

Bioinformatics for Beginners 2022



***Bioinformatics Training
& Education Program***

Table of Contents

Home

● Bioinformatics for Beginners: RNA-Seq	18
● Course Description:	18
● Module 1: Unix and Biowulf (Sep 27th - Oct 18th, 2022)	18
● Module 2: RNA-Seq Analysis (Oct 20th - Nov 29th, 2022)	18
● Module 3: Gene Ontology and Pathway Analysis (Dec 1st - Dec 13th, 2022)	19
● Course requirements:	19
● Who can take this course?	19
● What materials are needed to take this course?	20

Module1 - Unix/Biowulf

Lesson 1: Introduction to Unix and the Shell	22
● Lesson Objectives	22
● Why Bioinformatics?	22
● Useful concepts / guidance from statistics	22
● Terms to know	22
● Additional guidance	23
● What is Unix?	23
● Why learn Unix?	23
● A few things about the Unix shell...	23
● How is Unix different from other operating systems?	24
● How much Unix do I need to know to get started?	24

● Getting started with DNAnexus	24
● What is DNAnexus?	24
● Obtaining a DNAnexus account	24
● Finding the course and getting started with the GOLD system	25
● Getting started outside of DNAnexus	27
● Working with command line from a macbook	27
● Working with command line from a Windows computer	28
● Working with the Biostars software on Biowulf	29
● What is conda?	29
● Activating / deactivating a conda environment	30
● The importance of documentation	30
● Help Session	30

Lesson 2: Navigating file systems with Unix **31**

● Quick review	31
● Lesson Objectives	31
● A word about mistakes	31
● How can we overcome mistakes?	31
● File system	31
● Some useful unix commands to navigate our file system and tell us some things about our files	32
● Getting Started	32
● First Unix command (ls)	32
● Anatomy of a command	33
● Where am I? (pwd)	34
● More on the home directory	35
● Creating files (touch)	36
● The nano editor is a text editor useful for small files.	36
● Choosing good names for your files and directories.	36

● Removing files with rm	37
● Creating (mkdir) and removing (rmdir) directories	38
● Removing directories (rmdir)	38
● Getting around the Unix directory structure (cd)	38
● Getting back to removing directories (rmdir)	40
● Moving and renaming files and directories, all with one command (mv)	40
● Viewing file content	41
● Copying files (cp)	42
● Help! (man)	42
● Additional Resources	43
● Help Session	43

Lesson 3: Introduction to Biowulf 44

Lesson 4: Useful Unix 45

● Lesson 3 Review	45
● Learning Objectives	45
● Flags and command options - making programs do what they do	45
● Use of wildcards	47
● Using tab complete for less typing	48
● Access your history with the "up" and "down" arrows on your keyboard	48
● Keyboard shortcuts	48
● cat, head, and tail	49
● Working with file content (input <, and output >, append >>)	50
● Combining commands with pipe (). Where the heck is pipe anyway?	51
● Using what we've learned, all together now.	52
● Finding information in files with grep	52
● Performing repetitive actions with Unix	54
● Permissions - when all else fails check the permissions	55

● Help Session	56
----------------	----

Lesson 5: Working on Biowulf **57**

● Lesson 4 Review	57
● Lesson Objectives	57
● Working on Biowulf	57
● Login using ssh	57
● Batch Jobs	58
● Multi-threaded jobs and sbatch options	59
● Some slurm commands	59
● Standard error and standard output	60
● Partitions	60
● Walltime	61
● Submit an actual job	62
● Swarm-ing on Biowulf	64
● Let's create a swarm job	65
● Running Interactive Jobs	65
● Exiting from Biowulf	66
● So you think you know Biowulf?	66
● Help Session	66

Lesson 6: Downloading data from the SRA **67**

● Lesson 5 Review:	67
● Learning Objectives:	67
● Create a project directory	67
● A brief word about sequence formats	68
● What is the SRA?	68
● SRA terms to know	68
● Using fastq-dump	68

● Using seqkit stat	71
● fasterq-dump	72
● Using prefetch	72
● Navigating NCBI SRA	72
● Where can we get an accession list or run information for a particular project?	72
● E-utilities	75
● Using the "parallel" tool for automating downloads	75
● European Nucleotide Archive (ENA)	77
● Help Session	77

Lesson 7: Downloading the RNA-Seq Data and Dataset Overview **78**

● Lesson Review	78
● Learning Objectives	78
● Getting Project files	79
● UHR and HBR data	79
● Downloading the data	79
● A brief introduction to awk and sed	83
● What is awk?	83
● What is sed?	84
● Help Session	85

Introduction to RNASeq

RNA-SEQ Overview **87**

● What is RNASEQ ?	87
● RNASEQ - WorkFlow	87
● Generating the Data - Experimental Design	88
● Only Sequence the RNA of interest	88

● Remember	88
● What question(s) are you asking?	89
● Read Choices	89
● Replicates	89
● Data Analysis Questions	90
● Best Practice Guidelines from Bioinformatic Core (CCBR):	90
Generating the Data	92
● General Rules for Sample Preparation	92
● Sample Amounts Guidelines	92
● RNA-Seq Sample Recommendations (CCBR)	93
● Sequencing	93
● Illumina Sequencing Platforms	93
● Long Read Sequencing Platforms	94
Data Analysis Overview	96
● RNASEQ - Data Analysis WorkFlow	96
● Computational Considerations THE GOOD NEWS	96
● Computational Considerations THE BAD NEWS	97
● Computational Considerations The Challenges	97
● Computational Prerequisites	97
Data Analysis	98
● Data Quality Assessment (QC)	99
● Examples of some of the quantities measured in basic QC	100
● By the program FastQC	100
● By the program MultiQC	101
● Raw Sequence Cleanup	102

Alignment	103
● RNASeq Mapping Challenges	103
● Mapping Challenges	104
● RNASeq Mapping Solutions	104
● Common Aligners	106
● To Align or not to Align	106
● Typical Questions about alignment	106
● Answers	106
● Questions not asked	107
● Answers	107
● Special Consideration for Alternate Splicing Events	107
● Post Alignment QC Programs	107
● RSeQC example of plot types	108
● Post Alignment Cleanup Programs	109
Quantitation	110
● Counting as a measure of Expression	110
● Count Normalization	111
● Counting as a measure of Expression	112
● Log Transformed Data	113
● Common counting Programs	113
Differential Expression	114
● Normalization and Statistical Significance	114
● Replicates	114
● Multiple Testing Correction	114
● Count Matrix	115
● Contrast File	116
● Differential Expression Programs	116

● Differential Expression Output	117
Visualization	118
Resources	125
● Tertiary Analysis - Biological Meaning	125
● Software Solutions	125
● Public sources of RNA-Seq data	125
● NGS File Formats	126
● Utility Programs	126
● Web-Based Tools	126
● File Transfer Methods	126
● Further Reading	127

Module 2 - RNA sequencing

Lesson 9: Reference genomes and genome annotations used in RNA sequencing

● Lesson 8 Review	129
● Learning Objectives	129
● About the Human Brain Reference and Universal Human Reference data	129
● Where is my data?	130
● Where are the analysis tools?	133
● The reference genome and annotation files	134
● Visualizing the reference genome and annotations	139

Lesson 10: Introducing the FASTQ file and assessing sequencing data quality

● Lesson 9 Review	145
● Learning objectives	145
● HBR and UHR FASTQC reports	162

Lesson 11: Merging FASTQ quality reports and data cleanup	164
● Lesson 10 Review	164
● Learning objectives	164
● Merging individual FASTQC reports	164
● Trimming away adapters and poor quality reads	175
● Trimming with Trimmomatic	180
● Trimming with BBDuk	186
● MultiQC report for HBR and UHR dataset	191
● FASTQC reports for SRR1553606	191
Lesson 12: RNA sequencing review 1	192
● Learning objectives	192
● Accessing the Biostar handbook	192
● Review of RNA sequencing concepts	195
● RNA sequencing analysis considerations	195
● What are the files that we need for RNA sequencing analysis?	195
● Review of reference genome and annotation files	195
● Review of FASTQ files	196
Lesson 13: Aligning raw sequences to reference genome	198
● Lesson 11 Review	198
● Learning objectives	198
● Creating an index for the reference genome	198
● Performing alignment for one file using HISAT2	200
● Working with SAM file alignment outputs	211
● Alignment of HBR and UHR raw sequencing data with Bowtie2	213
● MultiQC reports with pre-alignment QC and post-alignment statistics	215
Lesson 14: Visualizing alignment results	216
● Lesson 13 Review	216

● Learning objectives	216
● Creating bigWig files	216
● Creating bigWig files - step 1, convert BAM to bedGraph	217
● Creating bigWig files - step 2, create a genome index	220
● Creating bigWig files - step 3, actually creating the bigWig files	220
● Visualizing alignment HBR and UHR alignment results with IGV	221

Lesson 15: Finding differentially expressed genes **235**

● Lesson 14 review	235
● Learning objectives	235
● Obtaining expression read counts from alignment results	235
● Normalization	238
● Differential expression analysis	239
● Visualizing differential expression results	242

Lesson 16: RNA sequencing review and classification based analysis **246**

● Review	246
● Tools for assessing sequencing data quality	246
● Tools for sequencing data cleanup	246
● Tools for alignment	249
● Obtaining expression read counts	249
● Differential expression analysis	250
● Learning objectives	250
● Constructing human chromosome 22 reference transcriptome	250
● Alignment of HBR and UHR raw sequencing data to human chromosome 22 transcriptome	252
● Obtaining differentially expressed genes	258
● Mapping transcript IDs to gene names	259
● Interpretation and comparison to alignment based RNA sequencing	264
● Visualizing gene expression	265

Module 3 - Pathway analysis and course review

Gene ontology and pathway analysis 268

● Objectives	268
● Where have we been and where are we going?	268
● What is gene ontology?	270
● What is a GO term?	271
● Approaches to gene set analysis / pathway analysis?	272
● Over-representation analysis (ORA)	272
● Functional Class Scoring (FCS)	272
● What is MSigDB and how does it relate to GSEA?	273
● Pathway Topology	273
● What tools are available?	273
● Other and related tools	274
● Other databases	274
● Importance of Gene IDs	276
● Resources:	276

Database for Annotation, Visualization and Integrated Discovery (DAVID) - an overview 278

● Lesson 17 review	278
● Learning objectives	278
● Background on DAVID	278
● What does DAVID do?	278
● Background gene list	279
● Basic statistics behind DAVID	279
● Data size limits	281

● Getting help	281
● Starting an analysis in DAVID	281
● Supplying input	282
● Results and interpretation	289
● Annotation Results Summary	289
● Disease Annotations	289
● Disease Annotation - chart view	290
● Disease Annotation - link to external resource	291
● Disease Annotation - gene view	291
● Disease Annotation - gene view results	292
● Disease Annotation - OMIM	292
● Pathways:	295
● Functional Annotation Chart	296
● Functional Annotation Clustering	299
● Functional Annotation Table	301
● Gene Functional Classification Result	302

Introduction to Qiagen Ingenuity Pathway Analysis 304

● Lesson objective	304
● Example data	304
● Accessing IPA	304
● IPA learning resources	304

Course Wrap-up 305

● Lesson Objectives	305
● RNA-Seq overview	305
● Review of resources	306
● BTEP Resources pages	306

● Course documentation and resources	306
● The Biostar Handbook	306
● Biostars on Biowulf	306
● Getting a Biowulf account	306
● Accessing Biowulf	307
● Where can I find training materials on Biowulf?	307
● What software is available on Biowulf?	307
● Using the Biostars Module	308
● Upcoming BTEP classes	309

Help Sessions

Lesson 2 Practice	311
-------------------	-----

Lesson 4 Practice	315
-------------------	-----

Lesson 5 Practice	317
-------------------	-----

● Login to Biowulf	317
● Let's run fastqc, a quality control program, on the files we downloaded from the SRA.	
● Using sbatch	317 ³¹⁷
● Accessing the Biostars module on Biowulf	319
● Additional practice materials from hpc.nih.gov	319

Lesson 6 Practice	320
-------------------	-----

● Find the data	320
● Download the data	320

Lesson 7 Practice	322
-------------------	-----

● Task 1: Create a new directory	322
----------------------------------	-----

● Task 2: Download the "Golden Snidget" reference data	322
● Task 3: Download the "Golden Snidget" reads	323

Lesson 9 Practice **325**

● Objectives	325
● Create a directory for the Golden Snidget analysis	325
● Where is my data?	326
● Find out some details about the Golden Snidget genome and transcriptome	328
● Visualizing Golden Snidget genome and transcriptome in IGV	331
● Load the Golden Snidget reference	331

Lesson 10 Practice **334**

● Objectives	334
● Where is my data?	334
● FASTQC reports for the Golden Snidget	346

Lesson 11 Practice **348**

● Objectives	348
● Merging Golden Snidget FASTQC reports into one	348
● Quality and adapter trimming	356
● MultiQC report for Golden Snidget	359

Lesson 12 Practice **360**

● Objectives	360
● Where is my data	360
● Getting to know the data	362
● Trimming	364
● QC reports for hcc1395 dataset	365
● Pre-trimming	365
● Post-trimming	366

Lesson 13 Practice	367
● Objectives	367
● Review of what we have done so far for the Golden Snidget dataset	367
● Aligning Golden Snidget sequencing - constructing genome index	368
● Constructing a text file with the Golden Snidget sample IDs	369
● Alignment of Golden Snidget FASTQ files - constructing the HISAT2 command	370
● Alignment of Golden Snidget FASTQ files - converting SAM files to BAM	372
Lesson 14 Practice	374
● Objectives	374
● About the data and launching IGV	374
● Testing your knowledge	376
● Bonus question	381
Lesson 15 Practice	383
● Objectives	383
● Scripts used for differential analysis	383
● Create a folder to store the Golden Snidget differential expression analysis results	383
● Create the design file	384
● Creating design.csv	384
● Obtaining the counts table	384
● Format the Golden Snidget counts table for differential expression analysis	385
● Visualizing expression	386
Lesson 16 Practice	387
● Objectives	387
● Create a directory for the classification based analysis output	387
● Indexing the reference transcriptome	387
● Obtaining expression counts	388
● Differential expression analysis	389

● Visualization	390
● Bonus question (if there is time)	391

Database for Annotation, Visualization and Integrated Discovery (DAVID) - practicing what we learned **393**

● Learning objectives	393
● Practicing what we learned	393
● Getting the data	393
● Upload the data to DAVID	393
● Results	394
● Gene Functional Classifier	394
● Potential Diseases	395
● Gene Ontology	396
● Functional Annotation Clusters	396

References

References	399
Biostars Software Reference	400
● Biostars Software Description	400

Additional Resources

Further readings and tutorials	403
● Additional Resources	403
● Working with Unix	403
● RNA-Seq	403

Logging into Biowulf	404
Accessing the Biostar Handbook	405
Using the Biostars Module for Biowulf	406
● Biostars on Biowulf	406
● How to use the Biostars Biowulf module	406
● Loading the module	406
● How to transfer data to and from Biowulf?	407

Installing IGV

Instructions for installing IGV	409
---------------------------------	-----

Questions and Answers

Questions and answers	411
● BTEP Bioinformatics for Beginners (September 13th, 2022 - December 13th, 2022)	411
● Questions and Answers	411

Bioinformatics for Beginners: RNA-Seq

Course Description:

This course was designed to teach the basic skills needed for bioinformatics, including working on the Unix command line. This course primarily focuses on RNA-Seq analysis. All steps of the RNA-Seq workflow, from raw data to differential expression and gene ontology analysis, are covered. However, many of the skills learned are foundational to most bioinformatics analyses and can be applied to the analysis of other types of next generation sequencing experiments.

This course is divided into three modules.

Module 1: Unix and Biowulf (Sep 27th - Oct 18th, 2022)

Lessons focus on developing command line skills, getting started and working on Biowulf (the NIH HPC cluster), and downloading data from NCBI.

- Lesson 1 - Introduction to Unix and the Shell (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=e19e1849d27f8b3f04aed4321ced7d2c>))
- Lesson 2 - Navigating file systems with unix (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=dea8f906e57eca037127a54c3b7e93b7>))
- Lesson 3 - Introduction to Biowulf (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=4a04321c200300b362ca7a5ff80c59f8>))
- Lesson 4 - Useful Unix (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=4b159fc6f848ba715346fbb80e0af75e>))
- Lesson 5 - Working on Biowulf (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=78a0cae7bb4a4827d3b810f9429a323e>))
- Lesson 6 - sra-tools, e-utilities, parallel (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=9c52e2e33df72d435d66fe8f21d33f0e>))
- Lesson 7 - Review; Downloading RNA-Seq Data (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=752ff24fd25f93a7b69ec295404e04a5>))

Module 2: RNA-Seq Analysis (Oct 20th - Nov 29th, 2022)

Lessons focus on RNA-Seq analysis including experimental design and best practices, quality control, trimming, alignment based methods, classification based methods, feature counts, and differential expression analysis.

- Lesson 8: Introduction to RNA-Seq (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=cc07b2529a448e51faf326399fa0d7f4>))

- Lesson 9: Introduction to the data (HBR, UHR) (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=efd124edc61536dadac4ea2d631c4fe0>))
- Lesson 10: NGS quality check (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=0b98531157512a78e8f4f2cce1c7d03e>))
- Lesson 11: Quality and adapter trimming (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=832b386b16123baca2fd8398feb9497e>))
- Lesson 12: Review (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=593764eda003f31b2766c2bef5f6c74c>))
- Lesson 13: Alignment based RNA-Seq, part 1 (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=4d877f2bc99d5ded9515f0ef4f5bb1ca>))
- Lesson 13: Alignment based RNA-Seq, part 2 (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=c96bec102980bab422beb4dfbb2b27c8>))
- Lesson 14: IGV (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=16f17a6a3f0854a24c238409d1ec584b>))
- Lesson 15: Differential Expression Analysis (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=88929fd74383ddfa010c3a568fec200e>))
- Lesson 16: Classification based RNA-Seq (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=97819965be99b59d53e59e26009b0072>))

Module 3: Gene Ontology and Pathway Analysis (Dec 1st - Dec 13th, 2022)

Lessons focus on gene ontology and pathway analysis.

- Lesson 17: Introduction to gene ontology and pathway analysis (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=5c6a9f415202606a12515f5bbb5c26b1>))
- Lesson 18: Functional enrichment with DAVID (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=e40b65d867f24c06718ce1cfd8689f07>))
- Lesson 19: Pathway analysis with Qiagen IPA (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=e2fb05d4f0479a491b6fc3fcb13490bd>))
- Lesson 20: Review and Course Wrap-up (Recording (<https://cbiit.webex.com/cbiit/ldr.php?RCID=69b4b4550053c2260a37253ad93ca83c>))

Course requirements:

Who can take this course?

There are no prerequisites to take this course. This course is open to NCI-CCR researchers interested in learning bioinformatics skills, especially those relevant to analyzing bulk RNA sequencing data.

What materials are needed to take this course?

To participate in this course, you will need a computer, a reliable internet connection, and a web browser. All classes and help sessions will be held virtually through Webex. In addition, this class will be taught with the GOLD learning environment on the DNAnexus platform. Learners will need to sign up for a DNAnexus account and complete the [registration survey \(https://www.surveymonkey.com/r/LWXPTKF\)](https://www.surveymonkey.com/r/LWXPTKF) to participate. Class registrants will receive a license to the *Biostar Handbook*.

Lesson content and practice questions can be found in these pages. Email ncibtep@nih.gov if you have any comments, questions, or concerns.

Module1 - Unix/Biwulf

Lesson 1: Introduction to Unix and the Shell

Lesson Objectives

1. Course overview.
2. Introduce Unix and describe how it differs from other operating systems.
3. Introduce and get set up on DNAnexus and the GOLD system.
4. Discuss ways to use the command line outside of the DNAnexus teaching environment.
5. Introduce conda.

Why Bioinformatics?

1. Analyze your own data
2. Expand scientific training and skills
3. Provide a path to a new career
4. Have a better understanding of how other people analyze data

Useful concepts / guidance from statistics

Terms to know

- **Normalization** - the process of scaling data to account for uncontrolled factors affecting variation.
- **Effect size** - "the quantitative measure of the magnitude of a phenomenon" (Biostar Handbook).
- **P-value** - "the probability for the observed effect size to be a product of random chance" (Biostar Handbook). **Statistical significance** is usually set to values below 0.05 or 0.01.
- **Adjusted p-value** adjusted for multiple testing / multiple comparisons. When repeating a test multiple times, the chances of getting a value by random chance increases.
- **Statistical power** - "reflects the ability of a test to produce the 'correct' prediction" (Biostar Handbook). This is impacted by sample size, effect size, and the applied statistical model.

- **Confidence interval** - the range of values that contains some true value at a defined probability (e.g., 95%).

Additional guidance

- **Statistical Models** - these should be appropriate for your experimental design. The methods you are using should be consistent with what's in related scientific literature. Different tests produce different results. Consult a statistician if possible.
- **Outliers** - try to identify early on. If you remove data, you should have sound rationale for doing so.
- **P-hacking and HARKing** - P-hacking is when you alter some form of the data analysis pipeline to get different results (e.g., using different statistical tests or including/excluding only subsets of the data). HARKing is modifying your hypotheses based on results. These are common in data science, and can be dangerous if (1) you aren't transparent about why something was done and what you did, (2) you fail to validate results, and (3) you don't explore alternative explanations.

Do not overinterpret the meaning of a p-value. p-value thresholds are fairly arbitrary.

What is Unix?

1. An operating system, just like Windows or MacOS
2. Something that is worth learning
3. Sometimes used interchangeably with Linux, which for our purposes, is just a version of Unix

Why learn Unix?

1. Many tools (like a bazillion) for biological data analysis are freely available and supported on Unix systems
2. Useful for working with big data, like genomic sequence files
3. To use the NIH High Performance Cluster (HPC) Biowulf for data analysis

A few things about the Unix shell...

1. It gives a command line interface where users can type commands
2. Also a scripting language, used to automate repetitive tasks

- The Bash shell (the Bourne Again SHell) is the most popular Unix shell.
- 3.

How is Unix different from other operating systems?

1. Does not use a Graphical User Interface (GUI) better known as a "point and click" environment.
2. The user has to learn a series of commands for interacting with a Unix system
3. BUT...a few commands, like the ones we will learn over the next several lessons, will allow us to employ a number of bioinformatics tasks

How much Unix do I need to know to get started?

As with any language, the learning curve for Unix can be quite steep. However, to get started analyzing data you really need to understand the following:

1. Directory navigation: what the directory tree is, how to navigate and move around with `cd`
2. Absolute and relative paths: how to access files located in directories
3. What simple Unix commands do: `ls`, `mv`, `rm`, `mkdir`, `cat`, `man`
4. Getting help: how to find out more on what a unix command does
5. What are "flags": how to customize typical unix programs `ls` vs `ls -l`
6. Shell redirection: what is the standard input and output, how to "pipe" or redirect the output of one program into the input of the other --- [Biostar Handbook \(https://www.biostarhandbook.com/introduction-to-unix.html\)](https://www.biostarhandbook.com/introduction-to-unix.html)

Many of these will be covered in Lesson 2.

Getting started with DNAnexus

What is DNAnexus?

DNAnexus provides a secure cloud based platform for the analysis and sharing of next generation sequencing data. This class will use a pre-built teaching environment, the GOLD platform, which includes all of the software needed installed and ready to go.

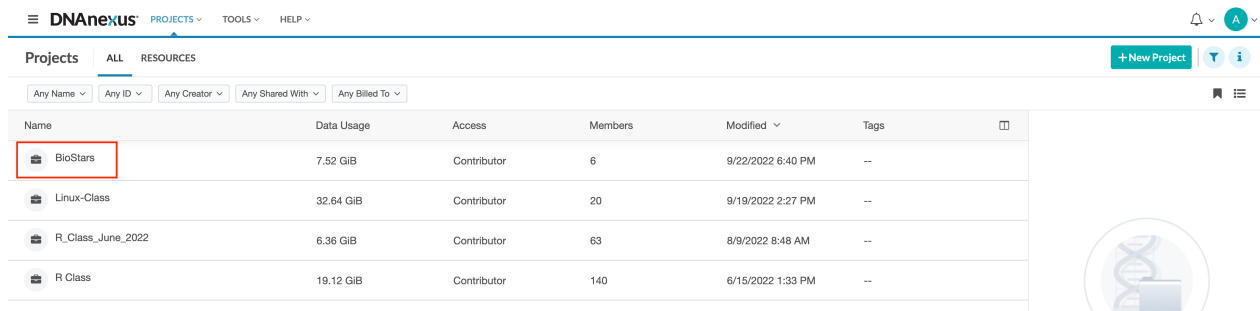
Obtaining a DNAnexus account

If you have not already created a free DNAnexus account, please do so [here \(https://platform.dnanexus.com/register\)](https://platform.dnanexus.com/register). Once you have obtained your free account, you will need to email us your username at ncibtep@nih.gov to obtain access to the course page and GOLD System.

Finding the course and getting started with the GOLD system

Step 1: Login to DNAnexus

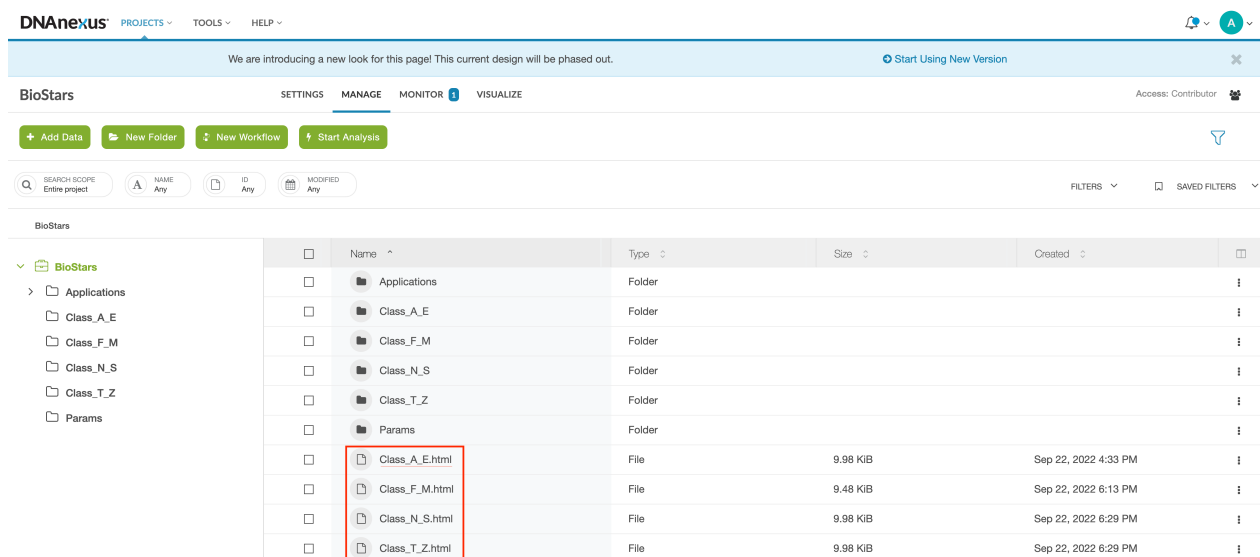
Step 2: Once you login, you should see the *Projects* page. If you have used DNAnexus previously, you may see more than one project listed. If this is your first time using DNAnexus, you will only see the project name for this course listed, **BioStars**. Double click on BioStars.



The screenshot shows the DNAnexus interface with the 'Projects' tab selected. A table lists several projects. The 'BioStars' project is highlighted with a red box. The table has columns for Name, Data Usage, Access, Members, Modified, and Tags.

Name	Data Usage	Access	Members	Modified	Tags
BioStars	7.52 GiB	Contributor	6	9/22/2022 6:40 PM	--
Linux-Class	32.64 GiB	Contributor	20	9/19/2022 2:27 PM	--
R_Class_June_2022	6.36 GiB	Contributor	63	8/9/2022 8:48 AM	--
R Class	19.12 GiB	Contributor	140	6/15/2022 1:33 PM	--

Step 3: Once you double click on the BioStars project, you will see a project directory containing multiple subdirectories and files. Select (double click) one of the *.html* files (e.g., *Class_LETTER_LETTER.html*). We have divided the class into four groups based on name. For example, if your first name begins with a letter A-E, select *Class_A_E.html*; if your first name begins with a letter F-M, select *Class_F_M.html*. You will need to double-click on the *.html* file.



The screenshot shows the DNAnexus interface for the 'BioStars' project. The 'MANAGE' tab is selected. A table lists the project's contents, including folders and files. The 'Class_A_E.html' file is highlighted with a red box. The table has columns for Name, Type, Size, and Created.

Name	Type	Size	Created
Applications	Folder		
Class_A_E	Folder		
Class_F_M	Folder		
Class_N_S	Folder		
Class_T_Z	Folder		
Params	Folder		
Class_A_E.html	File	9.98 KiB	Sep 22, 2022 4:33 PM
Class_F_M.html	File	9.48 KiB	Sep 22, 2022 6:13 PM
Class_N_S.html	File	9.98 KiB	Sep 22, 2022 6:29 PM
Class_T_Z.html	File	9.98 KiB	Sep 22, 2022 6:29 PM

Step 4: The *Class_LETTER_LETTER.html* file will open the GOLD platform application, and you will see a screen that looks like this:



Welcome to GOLD, an online learning platform presented by BTEP

Use the links below to a) login to your account and b) view files in the [public] folder

Peter Fitzgerald	Amy Stanelake	Carl McIntosh	Desiree Tillo	Alex Emmons	Joe Wu
<input type="text" value="peter"/>	<input type="text" value="amy"/>	<input type="text" value="carl"/>	<input type="text" value="des"/>	<input type="text" value="alex"/>	<input type="text" value="joe"/>
<input type="button" value="Files"/>	<input type="button" value="Files"/>	<input type="button" value="Files"/>	<input type="button" value="Files"/>	<input type="button" value="Files"/>	<input type="button" value="Files"/>

User	Login	View
First Last	<input type="text" value="Name"/>	<input type="button" value="Files"/>
First Last	<input type="text" value="Name"/>	<input type="button" value="Files"/>
First Last	<input type="text" value="Name"/>	<input type="button" value="Files"/>

Find your name and select from the Login column

At the top of the page you will see the instructors pictures and logins. You will need to find your name (First and Last) in the table below the instructors. Once you find your name click on the link associated with your name in the login column. **The name that you see in the login column will serve as your username in step 5.**

Step 5: The login link will open a terminal with a prompt to login. Login with your username (See step 4) and password (to be distributed in class).

A screenshot of a terminal window. At the top, the NIH logo and "NATIONAL CANCER INSTITUTE Center for Cancer Research" are displayed. Below that, it says "Bioinformatics Training and Education Program --- email BTEP at ncibtep@nih.gov". The terminal prompt shows "job-GGfqXv809vXZ6z1P7F3Xzp67 login: " followed by a cursor.

Step 6: Once you login at the terminal, you will see the following page:

Bioinformatics for Beginners 2022

Bioinformatics for Beginners: RNA-Seq

Course Description:

This course was designed to teach the basic skills needed for bioinformatics, including working on the Unix command line. This course primarily focuses on RNA-Seq analysis. All steps of the RNA-Seq workflow, from raw data to differential expression and gene ontology analysis, are covered. However, many of the skills learned are foundational to most bioinformatics analyses and can be applied to the analysis of other types of next generation sequencing experiments.

This course is divided into three modules.

Module 1: Unix and Biowulf

Lessons focus on developing command line skills, getting started and working on Biowulf (the NIH HPC cluster), and downloading data from NCBI.

- Lesson 1 - Introduction to Unix and the Shell

Expandable course documentation with terminal below.
Click the border and drag.

Command prompt

The course documentation is accessible at the top of the page and can be dragged up or down for viewing. The command line terminal accounts for the rest of the page. You may need to resize the screen to see the command prompt.

Now you should be logged onto the GOLD platform and ready for class.

Ending your DNAnexus session: if you are finished with the GOLDSYSTEM for the day, logout using

```
exit
```

Getting started outside of DNAnexus

Lessons 3 and 5 will focus on Biowulf and will require VPN access and the ability to connect remotely via a terminal on your local computer.

In addition, we want you to be able to get started analyzing data on your own without having to use the GOLD teaching environment. Most bioinformatics software will work with unix based systems (MacOS or Linux). Therefore, if you are working on a Windows operating system, you will need a work around.

Working with command line from a macbook

- Type `cmd + spacebar` and search for "terminal". Once open, right click on the app logo in the dock. Select `Options` and `Keep in Dock`.

- The default shell starting with Mac OSX version 10.14 is the `zsh` shell. While this is not really a problem, you can configure your computer to use the `bash` shell using the following:

```
chsh -s /bin/bash
```

- Follow [Biostar instructions \(https://www.biostarhandbook.com/computer-setup.html\)](https://www.biostarhandbook.com/computer-setup.html) if you want `bioinfo`, which contains the software used in the book, installed on your local machine. Regardless, you will likely need to install the Xcode compiler. You can search for and install this directly from the "App Store". Then install the additional Xcode command line tools from the terminal using

```
xcode-select --install
```

Working with command line from a Windows computer

If you are using a Windows operating system, Windows 10 or greater, you can use the [Windows Subsystem for Linux \(https://docs.microsoft.com/en-us/windows/wsl/\)](https://docs.microsoft.com/en-us/windows/wsl/) (WSL) for your computational needs.

The Windows Subsystem for Linux (WSL) is a feature of the Windows operating system that enables you to run a Linux file system, along with Linux command-line tools and GUI apps, directly on Windows, alongside your traditional Windows desktop and apps. --- [docs.microsoft.com \(https://docs.microsoft.com/en-us/windows/wsl/faq\)](https://docs.microsoft.com/en-us/windows/wsl/faq)

To install WSL, you will need to submit a help ticket to [service.cancer.gov \(https://service.cancer.gov/ncisp\)](https://service.cancer.gov/ncisp). There are multiple Linux distributions. We recommend new users install "Ubuntu".

If you do not plan to use your local machine for bioinformatics analyses, you can connect to the NIH HPC Biowulf using an SSH client. The secure shell (`ssh`) protocol is commonly used to connect to remote servers. More on Biowulf later.

Windows 10 or greater has a built in SSH client.

You can start an SSH session in your command prompt by executing `ssh user@machine` and you will be prompted to enter your password. ---[Windows documentation \(https://docs.microsoft.com/en-us/windows/terminal/tutorials/ssh?source=recommendations\)](https://docs.microsoft.com/en-us/windows/terminal/tutorials/ssh?source=recommendations)

To find the Command Prompt, type `cmd` in the search box (lower left), then press `Enter` to open the highlighted Command Prompt shortcut.

If this yields any major issues, try installing PuTTY (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>), Solar-PuTTY (https://www.solarwinds.com/free-tools/solar-putty?a_aid=BIZ-PAP-CMPRTCH&a_bid=1bd20791&CMP=BIZ-PAP-CMPR_PCW-SolarPutty-FSPTY-LM&data1=&data2=list), or MobaXterm (<https://mobaxterm.mobatek.net/>).

Note about command prompt and powershell: Just like the bash shell works effectively with a linux operating system, Windows also has shells to interact with the Windows operating system. Windows has two shells: the Command Prompt and the PowerShell. However, because most bioinformatics software is unix based, these shells will not be useful for bioinformatics scripting.

Working with the Biostars software on Biowulf

For your convenience, we have created a module on Biowulf that includes many of the same programs in the `bioinfo` environment from *The Biostar Handbook*. To use this module, please see the instructions documented under [Additional Resources](#).

What is conda?

The Biostar Handbook works with programs installed within a conda environment named `bioinfo`. Conda is commonly used for bioinformatics package installations.

Conda is often used for scientific software installation because...

Installing software is hard. Installing scientific software is often even more challenging. In order to minimize the burden of installing and updating software (data) scientists often install software packages that they need for their various projects system-wide.

Installing software system-wide has a number of drawbacks:

It can be difficult to figure out what software is required for any particular research project. It is often impossible to install different versions of the same software package at the same time. Updating software required for one project can often “break” the software installed for another project. --- [Pugh and Tocknell, Introduction to Conda for \(Data\) Scientists \(https://carpentries-incubator.github.io/introduction-to-conda-for-data-scientists/01-getting-started-with-conda/index.html\)](https://carpentries-incubator.github.io/introduction-to-conda-for-data-scientists/01-getting-started-with-conda/index.html)

Conda solves these problems by facilitating software installations, making the installation process far easier. As a package and environment management system, conda also enhances both the portability and reproducibility of scientific workflows by isolating software and their dependencies in “environments”. These environments do not interact with system wide programs and therefore do not reek havoc on your local machine due to software incompatibilities.

Conda runs on Windows, macOS, Linux and z/OS. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language. --- [docs.conda.io \(https://docs.conda.io/en/latest/\)](https://docs.conda.io/en/latest/)

Activating / deactivating a conda environment

The Biostar handbook has included bioinformatics software in a conda environment named `bioinfo`. If you followed Biostar Handbook instructions and created a `bioinfo` environment on your local computer, you will need to activate the environment to use installed software.

To activate a conda environment use

```
conda activate bioinfo
```

To deactivate your environment

```
conda deactivate
```

The importance of documentation

Analyzing next generation sequencing data requires a large number of computational steps. As you work, you should **ALWAYS** keep a record of what you are doing. Just as you keep a laboratory notebook, you should have a notebook for bioinformatics, in which you record the programs that you used, including version information, what commands you entered and their parameters, and any other code that you used to prepare or move around data. Your collaborators and future self will thank you. There are many options out there for code editors; both Visual Studio Code and Atom are fairly popular and approachable for beginners. Consider keeping a README.txt log per project or analysis step.

Help Session

1. Getting set up on DNAnexus
2. Getting everyone access to Biowulf via ssh.

Lesson 2: Navigating file systems with Unix

Quick review

- Unix is an operating system
- We use a unix shell (typically bash) to run many bioinformatics programs
- We need to learn unix to use non-GUI based tools and Biowulf

Lesson Objectives

- Learn the basic structure of a unix command
- Learn how to navigate our file system, including absolute vs relative directories
- Learn unix commands related to navigating directories, creating files and removing files or directories, and getting help

A word about mistakes

YOU WILL MAKE MISTAKES...but, it is okay. We all make mistakes, and mistakes are how we learn.

How can we overcome mistakes?

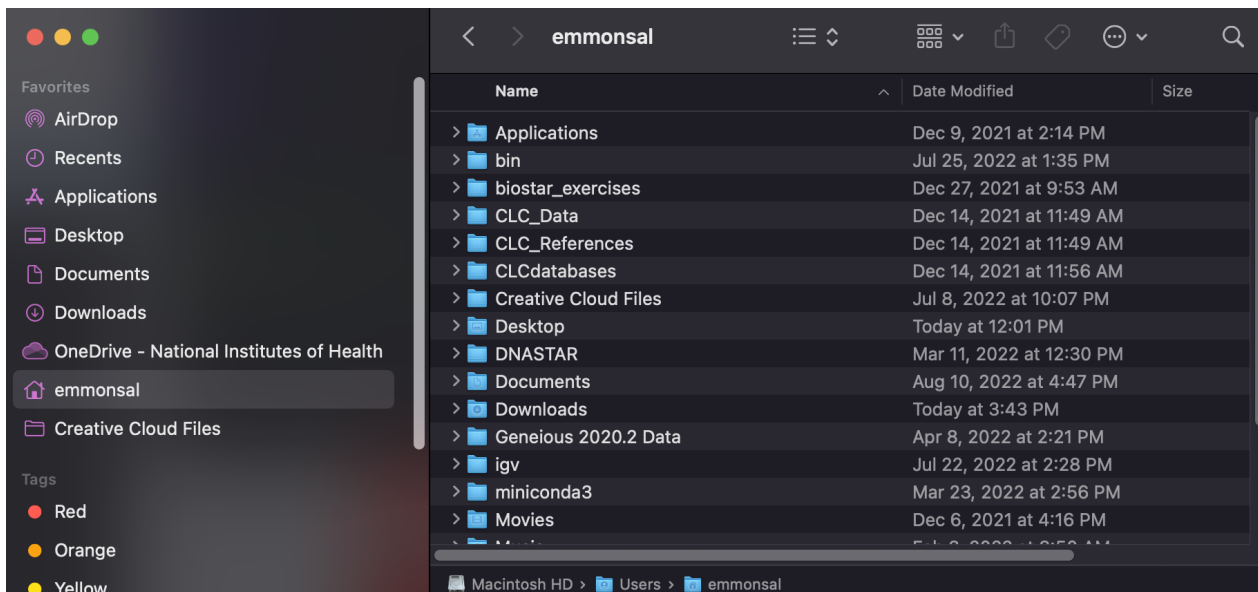
We practice. The more you use unix and bash scripting the better you will become.

You will need to learn how to troubleshoot error messages. Often this will involve googling the error in reference to the entered command. There are many forums that post help regarding specific errors (e.g., stack overflow, program repositories such as github).

File system

We manage files and directories through the operating system's file system. A directory is synonymous with a "folder", which is used to organize files, other directories, executables, etc.

On a Windows or Mac, we usually open and scroll through our directories and files using a GUI. For example, **Finder** is the default file management GUI from which we can access files or deploy programs on a macbook.



This same file system can be accessed and navigated via command line from the unix shell.

Some useful unix commands to navigate our file system and tell us some things about our files

1. `pwd` (print working directory)
2. `ls` (list)
3. `touch` (creates an empty file)
4. `nano` (basic editor for creating small text files)
5. using the `rm` command to remove files. Be careful!
6. `mkdir` (make a directory) and `rmdir` (remove a directory, must be empty of all files)
7. `cd` (change directory), by itself will take you home, `cd ..` (will take you up one directory), `cd /results_dir/exp1` (go directly to this directory)
8. `mv` (for renaming files or moving files)
9. `less` (for viewing files, "more" is the older version of this)
10. `man` (for viewing the man pages when you need help on a command)
11. `cp` (copy) for copying files

Getting Started

First Unix command (`ls`)

```
ls
```

You may see something like this:

```
public  reads.tar  sample.fasta  sample.fastq
```

The "ls" command "lists" the contents of the directory you are in. You may see files and other directories here.

How can you tell the difference between a file and a directory?

We can add some additional options to our command.

```
ls -lh
```

will show permissions and indicate directories (d). The -lh are flags. -l refers to listing in long format, while -h provides human readable file sizes.

Or, many systems offset directories and files using colors (e.g., blue for directories). If you don't see colored output, try the -G flag.

We can also label output by adding a marker to indicate files, directories, links, and executables using the -F flag.

```
ls -F
```

a terminal / = directory

a @ = link

a * = executable

Anatomy of a command

Using ls as an example, we can get an idea of the overall structure of a unix command.

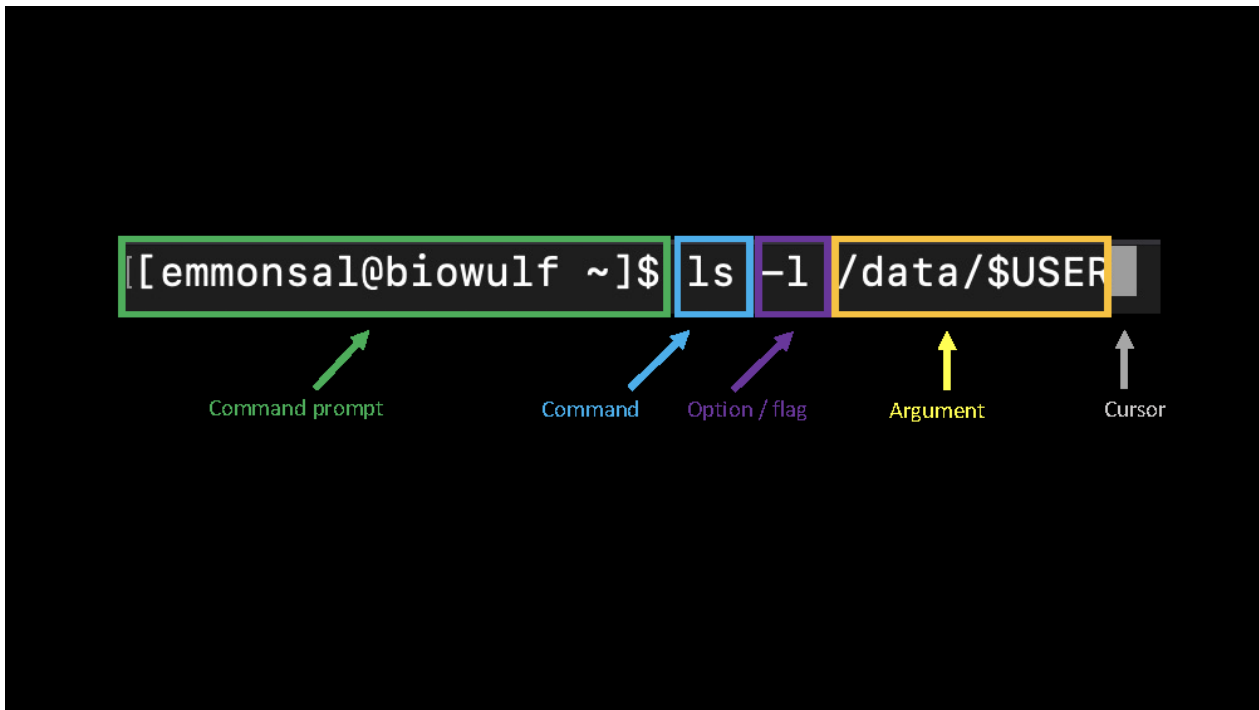


Image inspired by "Learn Enough Command Line to Be Dangerous"

The first thing we see is the command line prompt, usually \$ or %, which varies by unix system. The prompt let's us know that the computer is waiting for a command. Next we see the actual command, in this case, `ls`, telling the computer to list the files and directories. Most commands will have various options / flags that can be included to modify the command function. We can also supply an argument, which in the case of `ls` is optional. For example, here we supplied an alternative directory from which we are interested in listing files and directories. We hit `enter` after each command, and when the command has finished running, the command prompt will reappear prompting us to enter more commands.

Where am I? (pwd)

```
pwd
```

You should see something like this.

```
/home/username
```

where username is your name. This is your home directory - where you start from when you open a terminal. This is an example of a "path". The path tells us the location of a file or directory. Note: while Windows computers use a `\` as a path separator, unix systems use a `/`.

Therefore, the `pwd` command is very helpful for figuring out where you are in the directory structure. If you are getting "file not found" errors while trying to run something, it is a good idea to `pwd` and see if you are where you think you are. Type the `pwd` command and make a note of the directory you are in.

More on the home directory

We see that we are in our home directory. But where is that exactly?

The file system on any computer is hierarchical, with the top level of the file system, or root directory, being `/`.

See the following example file system:

Image from *swcarpentry/shell-novice: Software Carpentry: the UNIX shell, June 2019 (Version v2019.06.1)*

Image from swcarpentry/shell-novice: Software Carpentry: the UNIX shell, June 2019 (Version v2019.06.1)

At the top is the root directory that holds everything else. We refer to it using a slash character, `/`, on its own; this character is the leading slash in `/Users/nelle`.

Inside that directory are several other directories: `bin` (which is where some built-in programs are stored), `data` (for miscellaneous data files), `Users` (where users' personal directories are located), `tmp` (for temporary files that don't need to be stored long-term), and so on.

We know that our current working directory `/Users/nelle` is stored inside `/Users` because `/Users` is the first part of its name. Similarly, we know that `/Users` is stored inside the root directory `/` because its name begins with `/`.

Notice that there are two meanings for the `/` character. When it appears at the front of a file or directory name, it refers to the root directory. When it appears inside a path, it's just a separator.

Underneath `/Users`, we find one directory for each user with an account on Nelle's machine, her colleagues `imhotep` and `larry`.

Like other directories, home directories are sub-directories underneath `/Users` like `/Users/imhotep`, `/Users/larry` or `/Users/nelle`

Typically, when you open a new command prompt, you will be in your home directory to start. ---*swcarpentry/shell-novice: Software Carpentry: the UNIX shell* (<https://swcarpentry.github.io/shell-novice/02-filedir/index.html>)

Creating files (touch)

The `touch` command creates a file, but the file is empty, so it is not a command you will use very often, but good to know about.

```
touch file1.txt
touch file2.txt
ls
```

Now we see something like this.

```
file1.txt  file2.txt  public  reads.tar  sample.fasta  sample.1
```

The `nano` editor is a text editor useful for small files.

```
nano file2.txt
```

Let's put something in this file.

```
Unix is an operating system, just like Windows or MacOS. Linux is a l
```

Nano commands for saving your file and exiting nano:

1. control O - write file (equivalent to save as)
2. File name to write: file2.txt (Hit return/enter on your keyboard to save the file with this name).
3. control X - to Exit

This brings us to our next topic which is very important!

Choosing good names for your files and directories.

There should not be spaces in Unix file names or directories. Here is a good method to use:

Use the underscore (`_`) where a space would go, like this, to name a directory containing RNA-Seq data.

```
my_RNA_Seq_data
```

These are examples of file names:

```
brain_rna.fastq  
liver_rna.fastq
```

The first part of the file name provides info about the file, and the extension (.fastq) tells what kind of file it is. (Examples of file extensions are .csv, .txt, .fastq, .fasta and many more.)

More on file organization to come.

A word about file extensions

It's important to understand file extensions, to know what kinds of data you are working with.

.txt are text files. These are likely but not always tab delimited.

.tsv are tab delimited files.

.csv are "comma-separated values" - good for importing into MS Excel spreadsheets

.tar.gz indicates a tarred and zipped file - so it is a compressed file

.fastq tells you that these are FASTQ files, containing sequence data and quality scores

.fasta indicates FASTA formatted sequence data, either protein or nucleotide

Removing files with rm

Warning - a Unix system will delete something when you ask to delete it and there is usually no way of getting it back.

By adding the -i option, the system will ask if you're sure you want to delete. Generally speaking, when a file on a Unix system is deleted, it is gone.

You can modify your profile on a Unix system to always ask before deleting, this is a good idea when you're just getting started.

```
rm -i file1.txt
```

will remove a file we created.

Creating (`mkdir`) and removing (`rmdir`) directories

A couple things to note - this is a good time to give your directories meaningful names, which will help you keep things organized. Organization is key. I generally like to have a new directory per project, and within that directory, subdirectories separating raw data from analysis files. From there, each analysis would also get its own subdirectory. However, there are many ways to organize files and you should do whatever makes sense for your data and helps you (and others) stay organized.

For now, let's create a directory called `RNA_Seq_data`.

```
mkdir RNA_Seq_data
```

Removing directories (`rmdir`)

Directories must be completely empty of all files and other contents before you can delete them. There are ways to "recursively" remove files and directories using the `-r` option, but these can be dangerous. Keep in mind that once these files are deleted they are gone for good. Be extremely careful with the `-r` option. As beginners it can be safer to go to the directory and remove contents.

Getting around the Unix directory structure (`cd`)

This is a very helpful command used for moving around the directory structure.

It can be used to go to a specific directory. Let's "go to" the directory we just made, and make another directory within it.

```
cd RNA_Seq_data
pwd
mkdir exp_one
ls
cd exp_one
touch myseq.txt
ls
pwd
```

So, we've moved to the `RNA_Seq_data` directory, checked our directory with `pwd`, created a directory called `exp_one`, listed the contents of `RNA_Seq_data` so we can see the directory we just created, now we go to that directory with `cd`, create a file with `touch`, list the contents with `ls` and print our working directory.

By itself, the `cd` command takes you `home`. Let's try that, and then do a `pwd` to see where we are.

```
cd
pwd
```

We are now in our home directory.

```
/home/username
```

How can we go back to the `exp_one` directory we created? We need to give the "path" to that directory.

```
cd RNA_Seq_data/exp_one
pwd
ls
```

Check where you are with `pwd` and look at the contents of the directory with `ls`. What do you see? It should be the file "myseq.txt".

Here's another way to get around the directory structure using `cd`.

```
cd ~/RNA_Seq_data/exp_one
```

where the tilde `~` stands for your home directory.

How is this command different from the last one?

```
cd RNA_Seq_data/exp_one
```

The first `cd` command provides the full path to where you want to go, it is called an "absolute" path.

For the second version, you need to be in the directory that contains `/RNA_Seq_data`, or the command will not work. This is known as a "relative" path.

As a reminder, paths are the sequence of directories that hold your data. In this path...

```
~/RNA_Seq_data/exp_one
```

there is a directory named `exp_one`, within a directory named `RNA_Seq_data`, within our home directory.

You will become more comfortable with paths as you build up your directories and data.

Another way to use the `cd` command is to go up one level in the directory structure, like this.

```
cd ..
```

This can be very helpful as you move around the directory tree. There are many more ways to use the "cd" command.

Getting back to removing directories (`rmdir`)

Directories must be completely empty of all files and other contents before you can delete them. There are ways to "recursively" remove file and directories using the `-r` option, but these can be dangerous. Keep in mind that once these files are deleted they are gone for good. Be extremely careful with the `-r` option. As beginners it can be safer to go to the directory and remove contents.

What do you see when you try to remove this directory?

```
rmdir exp_one
```

What should we do? We need to remove the contents of a directory before we can remove the directory. Here's one safe option.

```
cd exp_one
ls
rm myseq.txt
ls
cd ..
ls
rmdir exp_one
```

Moving and renaming files and directories, all with one command (`mv`)

The `mv` command is a handy way to rename files if you've created them with a typo or decide to use a more descriptive name. For example:

```
cd
mv file2.txt README.txt
ls
```

Be careful when moving files, a mistake in the command can yield unexpected results.

```
mv README.txt RNA_Seq_data
cd RNA_Seq_data
ls
```

The `-i` interactive option will help keep you safe.

For example:

```
mkdir dir1
mkdir dir2
touch dir2/hello.txt
touch hello.txt
mv -i dir2/hello.txt hello.txt
```

Let's compare the following

```
mv dir1 dir2
cd dir2
mv dir1 dir3
```

Viewing file content

We can use the `less` command to view the contents of a file like this.

```
cd
less /data/sample.fasta
```

You'll need to type `q` to get out of `less` and back to the command line. Before the `less` command was available, the `more` command was commonly used to look at file content. The `less` command has more options for scrolling through files, so it is now the preferred command.

Copying files (cp)

This is similar to `mv` but will create an actual copy of a file. You will need to specify what you are copying (the source) and where you want to make the copy (the target).

For example:

Let's make a file in our home directory.

```
touch ~/file_to_copy.txt
```

Now, let's move that file to `RNA_Seq_data`.

```
cp ~/file_to_copy.txt ./RNA_Seq_data
```

Remember, the `.` is a relative path shortcut denoting our current directory.

We can also copy an entire directory using the recursive flag (`cp -r`).

```
cp -r RNA_Seq_data RNA_Seq_data_copy
```

Help! (man)

All Unix commands have a `man` or "manual" page that describes how to use them. If you need help remembering how to use the command `ls`, you would type:

```
man ls
```

There are quite a few flags/options that we can use with the `ls` command, and we can learn all about them on the `man` page. My favorite flags for `ls` are `-l` and `-h`. We will use flags often, and you won't get far in Unix without knowing about them. Try this:

```
cd  
ls -lh
```

We have already seen these flags, but as a reminder...

`-h` when used with the `-l` option, use unit suffixes (Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte) in order to reduce the number of digits to three or less using base 2 for sizes.

-l (The lowercase letter "ell".) List in long format. (See below). If the output is to a terminal, a total sum for all the file sizes is output on a line before the long listing.

Compare the results between these two commands.

```
cd
ls
ls -lh
```

Additional Resources

Software Carpentry: The Unix Shell (<https://swcarpentry.github.io/shell-novice/01-intro/index.html>)

Help Session

Practice navigating the file system and creating files. Instructions are [here](#).

Lesson 3: Introduction to Biowulf

I

Lesson 4: Useful Unix

For this lesson, you will need to login to the GOLD environment on DNAnexus.

Lesson 3 Review

- Biowulf is the high performance computing cluster at NIH.
- When you apply for a Biowulf account you will be issued two primary storage spaces: 1) `/home/$User` and 2) `/data/$USER`, with 16 GB and 100 GB of default disk space.
- Hundreds of pre-installed bioinformatics programs are available through the `module` system.
- Computational tasks on Biowulf should be submitted as a job (`sbatch`, `swarm`) or through an interactive session `sinteractive`.
- Do not run computational tasks on the login node.

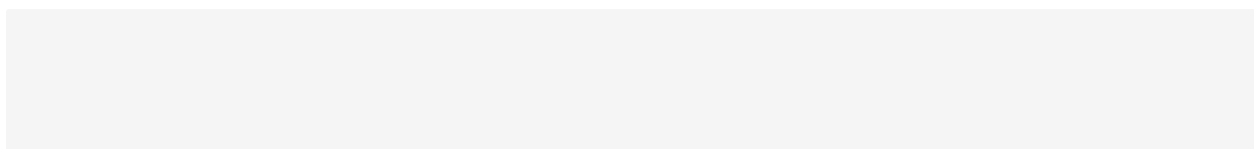
Learning Objectives

We are going to shift gears back to unix. We will focus on learning concepts that make working with unix particularly useful including:

- Flags and command options - making programs do what they do
- Wildcards (*)
- Tab complete for less typing
- Accessing user history with the "up" and "down" arrows on the keyboard
- `cat`, `head`, and `tail` for reading files
- Working with file content (input,output, and append)
- Combining commands with the pipe (`|`). Where the heck is pipe anyway?
- Finding information in files with `grep`
- Performing repetitive actions with Unix (`for` loop)
- Permissions - when all else fails check the permissions

Flags and command options - making programs do what they do

Compare the output from these commands:



```
ls
ls -S
ls -lh
```

`ls -h` (when used with `-l` option, prints file sizes in a human readable format with the unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte. This reduces the number of digits displayed.)

`ls -l` (list in long format). The column output is as follows:

1. file type
2. Content permissions
3. Number of hard links to content
4. Owner
5. Group owner
6. Content size (bytes)
7. Last modified date / time
8. File / directory name

`-S` (sort from largest to smallest file size)

What do you see when combining the `-h` and `-l` flags?

There are many flags you can use with `ls`. How would we find out what they are?

```
man ls
```

Or to see a more user friendly display, google to the rescue. Google "man ls unix" and see what you get. [Here's \(https://shaped.com/unix-ls/\)](https://shaped.com/unix-ls/) a useful, readable explanation of the "ls" command with examples.

Try combining some of the `ls` flags.

```
ls -lhS
ls -alt
```

`-a` (show hidden dot files .)

`-t` (sort by modification time with most recent listed first)

Flags and options add a layer of complexity to unix commands but are necessary to get the command or program to behave in the way you expect. For example, here is a command line for running "blastn" an NCBI/BLAST application.


```
blastn -db nt -query seq1.fasta -out results.out
```

What's going on in this command line? First, the BLAST algorithm is specified, in this case it is `blastn`, then the `-db` flag is used to choose the database to search against (`nt` for nucleotide). The `query` flag specifies the input sequence, which is in FASTA format, and the `-out` flag specifies the name of the output file.

Use of wildcards

Wildcard characters are a handy tool when working at the command line. Want to list all your FASTA files?

Use `*`:

```
ls /data/*.fasta
```

The `*` wildcard matches zero or more characters including spaces.

This example will work as long as all your FASTA files end in `.fasta`. But sometimes they don't.

```
ls /data/*.fa
```

or you want all the the FASTA and FASTQ files.

```
ls /data/*.f*
```

In addition to the asterisk (`*`) character, you can use other wildcards on the unix command line, not limited to the following:

`?` - matches a single character

`{ }` - used for multiple matches

`[]` - specify a range of characters or numbers. For example, `[a-z]` would specify all lower case letters and `[0-9]` would mean all digits.

To see some more practical examples of using wildcards, see [this article \(https://www.tecmint.com/use-wildcards-to-match-filenames-in-linux/\)](https://www.tecmint.com/use-wildcards-to-match-filenames-in-linux/) from tecmint.com and [this \(https://medium.com/@leedowthwaite/advanced-wildcard-patterns-most-people-dont-know-52f7fd608cb3\)](https://medium.com/@leedowthwaite/advanced-wildcard-patterns-most-people-dont-know-52f7fd608cb3) from the medium.com. This second article provides a nice discussion on how wildcards differ from regular expressions.

Using tab complete for less typing

Here's a good Unix trick to know - tab complete. Start typing the name of the file or directory you want, and hit the `tab` key. The system will auto-complete the name of the file or directory if the name is unique. It may only give a partial name if there is more than one file with a similar name in the directory. You can fill in the next part of the name and then try tab-complete again.

Let's create some files to test this.

```
touch file.txt
touch file.fasta
touch file.fastq
```

Start typing...

```
less f (hit tab)
```

What do you get? How does that differ from:

```
less file (hit tab)
```

The tab complete will save you lots of typing, and also help to figure out if you are where you think you are in the directory structure.

Access your history with the "up" and "down" arrows on your keyboard

Here's another Unix trick to make your life easier. Access previous commands with the up and down arrows on your keyboard. You can scroll backwards and forwards. This helps when you've got a typo or small mistake in your command lines that you can fix without retyping the whole thing.

Keyboard shortcuts

There are also a few handy keyboard shortcuts to make life on the command line easier. For example:

`ctrl c` to kill a running process
`ctrl l` to clear the screen
`ctrl a` skip to the beginning of a command
`ctrl e` skip to the end of a line

See more examples [here](https://unix.stackexchange.com/questions/255707/what-are-the-keyboard-shortcuts-for-the-command-line) (<https://unix.stackexchange.com/questions/255707/what-are-the-keyboard-shortcuts-for-the-command-line>).

cat, head, and tail

Who says Unix programmers don't have a sense of humor? Let me introduce `cat`, `head`, and `tail`. The `cat` command (short for "concatenate") is an extremely useful command for creating new files and viewing file contents. You can use it to open files for reading input and writing output. Or you can use it to copy several files into a new file. Also you can "append" the contents of a file to the end of another file.

This command reads the content of `sample.fasta` and outputs to standard output (i.e., the screen). This is not helpful for very large files, as it moves to the end of the file quickly. `Less` is a better command option for reading large files.

```
cat /data/sample.fasta
```

You can use `cat` to combine several files into one file, such as:

```
cat /data/seq1.fasta /data/seq2.fasta
```

Although, this again prints to standard output, the screen. To capture that output, we need to learn how to redirect output. (Coming up next!)

In the meantime, let's take a quick look at `head` and `tail`.

`head`

```
head /data/sample.fasta
```

`tail`

```
tail /data/sample.fasta
```

You can specify how many lines you would like to see (`-n`), or you can use the default value, which is 10.

```
head -n 20 /data/sample.fasta
```

Want to be sure of those 20 lines? `cat` again.

```
cat -n /data/sample.fasta
```

What does the `-n` flag do? How could you find out more about "cat"?

```
man cat
```

Working with file content (input `<`, and output `>`, append `>>`)

```
<  
>  
>>
```

Want to put the output from `cat`, `head`, or `tail` into a new file?

```
head -n 20 /data/seq1.fasta > smaller.fasta
```

Or we could put the last 20 lines into a file with `tail`.

```
tail -n 20 /data/seq1.fasta > smaller2.fasta
```

What if we want the first 20 lines and the last 20 lines in one file, with the first at the top and the last at the bottom? Use append, `>>` to paste the second file to the bottom of the first file. Let's try it.

```
head -n 20 /data/sample.fasta > smaller.fasta  
tail -n 20 /data/sample.fasta >> smaller.fasta
```

Keep in mind that if you input into the same file multiple times, you are overwriting the previous contents. For example, what is the final content of our file `covid.fasta`?

```
head -n 20 /data/seq1.fasta > covid.fasta
head -n 20 /data/seq2.fasta > covid.fasta
head -n 20 /data/seq3.fasta > covid.fasta
```

How many lines are now in "covid.fasta"? How can you check?

Let's try `wc`, short for word count.

```
wc covid.fasta
```

`wc` is a very useful function. Without opening a file, we can find out how many lines, words and characters are in it. Line counts are extremely useful to assess your data output.

If we created a file where we were expecting there to be 1000 lines of output? The `wc` command provides a quick way to check.

What happened to all of our content? The final results are from "seq3.fasta" only. The other two results files have been overwritten.

So, how would you get all three files into `covid.fasta`? You'll need to use `append`.

```
cat /data/seq1.fasta > covid.fasta
cat /data/seq2.fasta >> covid.fasta
cat /data/seq3.fasta >> covid.fasta
```

How could you test to see if the file has the expected number of lines?

```
wc covid.fasta
```

To input into a file

```
less < covid.fasta
```

Combining commands with pipe (|). Where the heck is pipe anyway?

The hardest thing about using pipe (|) is finding it on your keyboard!

The pipe symbol "|" (a.k.a., vertical bar) is way over on the right hand side of your keyboard, above the backslash \.

Pipe is used to take the output from one command, and use it as input for the next command, all in one command line. Let's look at some examples.

```
head -n 20 /data/sample.fasta | wc
```

Using what we've learned, all together now.

Let's say you've got a very large FASTA or FASTQ file, and you want to run an analysis on it. Before working on the whole file, it can be useful to set up a smaller test file instead.

Here's one way to do it.

```
cat /data/sample.fasta | head -n 20 > output.fasta
```

This combines several things we have learned about. The `cat` command opens the file `sample.fasta` for writing. The pipe `|` command is used to take that output and run it through the `head` command where we only want to see the first 20 lines, and we want them output `>` into a file called "output.fasta". Let's compare the files. How are they different?

```
ls -lh
```

and

```
less /data/sample.fasta  
less output.fasta
```

Finding information in files with grep

The `grep` utility is used to search files looking for a pattern match. It is used like this.

```
grep pattern options filename
```

As our first example we will look for restriction enzyme (EcoRI) sites in a sequence file (`eco.fasta`). The file has four EcoRI sites, but two of them are across the end of the line (and won't be found).

```
ls /data/eco.fasta  
grep -n GAATTC /data/eco.fasta
```

We can modify the "eco.fasta" file to remove the line breaks (`\n`) at the ends of the lines.

```
grep -v ">" /data/eco.fasta | tr -d "\n" | grep GAATTC
```

`-v` : This prints out all the lines that do not match the pattern

The unix "tr" (translate) command is used for translating or deleting characters.

Usage:

```
tr [option] set1 [set2]
```

`-d` : Deletes characters in the first set from the output

So this part of the command line is finding the line breaks "`\n`" and removing them.

The header lines (`>`) can also be removed.

And now we can see all four of the EcoRI sites.

What if we just wanted to count the occurrence of the EcoRI sites in the sequence?

```
grep -v ">" /data/eco.fasta | tr -d "\n" | grep GAATTC -c
```

`-c` : This prints only a count of the lines that match a pattern.

It is counting the entire line as one.

If we wanted to see each of the EcoRI sites listed.

```
grep -v ">" /data/eco.fasta | tr -d "\n" | grep GAATTC -o
```

And if we want to count the number of EcoRI sites.

```
grep -v ">" /data/eco.fasta | tr -d "\n" | grep GAATTC -o | wc
```

Let's create a word file that we can input to `grep`. We can input multiple restriction enzyme sites and search for all of them.

```
cd  
nano wordfile.txt
```

Put in the words (GAATTC, TTTT). Now we can use that file to find lines.

```
grep -v ">" /data/eco.fasta | tr -d "\n" | grep -f wordfile.txt -o |
```

We can find a list of options to be used with `grep` using `man`:

```
man grep
```

There are also existing programs to find motifs (patterns) in sequence data. For example:

Emboss - fuzznuc - a motif finding program (<http://emboss.sourceforge.net/apps/cvs/emboss/apps/fuzznuc.html>)

```
fuzznuc -help  
fuzznuc -pattern GAATTC -rformat2 tagseq eco.fasta -outfile /home/user
```

`-pattern` : Nucleotide pattern we are looking for

`-rformat2` : Report format

`tagseq` : tag sequence

`-outfile` : name and path to the results file

Performing repetitive actions with Unix

We can create a "for" loop to do iterative actions in Unix.

For each commands all on one line or separate lines: ("i" can be any variable name). These steps can be saved as a file, thereby creating a simple Unix script.

What does this "for loop" do?

```
for i in /data/*.fasta; do ls $i; done
```

What do these command lines do?

This one pulls out all the header ">" lines in the fasta files.

```
for i in /data/*.fasta; do echo $i; grep ">" $i; done
```

While this one just pulls out the ones from files named `seq*.fasta`.


```
for i in /data/seq*.fasta; do echo $i; grep ">" $i; done
```

Permissions - when all else fails check the permissions

Permissions dictate who can access your files and directories, and what actions they can perform. If you are consistently getting error messages from your command line and you're sure you are typing it correctly, it's worthwhile checking the permissions. So what does it all mean? How do we read the permissions information?

As we have seen, `ls -l`, provides information about file types, the owner of the file, and other permissions.

For example:

```
ls -l /data/sample.fasta
```

Here is an overview of what these permissions mean:

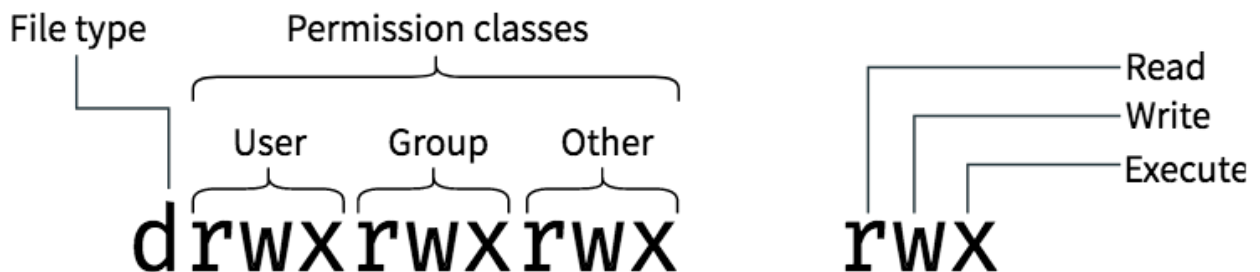


Image from [booleanworld.com](https://www.booleanworld.com/introduction-linux-file-permissions/) (<https://www.booleanworld.com/introduction-linux-file-permissions/>)

The first letter `d` indicates whether this is a directory or not - or some other special file type. The next 3 positions are the owner's/user's permissions. In this image, the owner can "read", "write" and "execute". So they can create files and directories here, read files here, and execute/run programs. The next 3 positions show the permissions for the "group". The last 3 positions shows permissions for everyone ("other").

You can modify permissions using `chmod`. Let's see this in action.

```
touch example.txt
chmod u-w example.txt
ls -l example.txt
```

Now, reassign write privileges to the user/owner

```
chmod u+w example.txt  
ls -l example.txt
```

Help Session

Let's complete a [Unix treasure hunt](#).

Lesson 5: Working on Biowulf

Lesson 4 Review

- Flags and command options
- Wildcards (*)
- Tab complete
- Accessing user history with the "up" and "down" arrows
- `cat`, `head`, and `tail`
- Working with file content (input,output, and append)
- Combining commands with the pipe (|)
- `grep`
- `for` loop
- File Permissions

Lesson Objectives

- Learn about the slurm system by working on Biowulf: batch jobs, swarms jobs, interactive sessions
- Retrieve data from NCBI through a batch job
- Learn how to troubleshoot failed jobs

NOTE: for this session, you will need to either login to your Biowulf account or use a student account.

Working on Biowulf

Now that we are becoming more proficient at the command line, we can use these skills to begin working on Biowulf. Today's lesson will focus on submitting computational jobs on the Biowulf compute nodes.

Login using `ssh`

Login to Biowulf. *Make sure you are on VPN.*

Open your Terminal if you are using a mac or the Command prompt if you are using a Windows machine.

```
ssh username@biowulf.nih.gov
```

When you log into Biowulf, you are automatically in your home directory (/home). This directory is very small and not suitable for large data files or analysis.

Use the "cd" command to change to the /data directory.

```
$ cd /data/$USER
```

where \$USER is an environment variable holding your username.

When working on Biowulf, you can not use any computational tools on the "login node". Instead, you need to work on a node or nodes that are sufficient for what you are doing.

To run jobs on Biowulf, you must designate them as interactive, batch or swarm. Failure to do this may result in a temporary account lockout.

Batch Jobs

Most jobs on Biowulf should be run as batch jobs using the "sbatch" command.

```
$ sbatch yourscript.sh
```

Where `yourscript.sh` is a shell script containing the job commands including input, output, cpus-per-task, and other steps. Batch scripts always start with `#!/bin/bash` or similar call. The sha-bang (`#!`) tells the computer what command interpreter to use, in this case the Bourne-again shell.

For example, to submit a job checking sequence quality using `fastqc` (MORE ON THIS LATER), you may create a script named `fastqc.sh`:

```
nano fastqc.sh
```

Inside the script, you may include something like this:

```
#!/bin/bash

module load fastqc
fastqc -o output_dir -f fastq seqfile1 seqfile2 ... seqfileN
```

where `-o` names the output directory

`-f` states the format of the input file(s)

and `seqfile1 ... seqfileN` are the names of the sequence files.

Note: `fastqc` is available via Biowulf's module system, and so prior to running the command, the module had to be loaded.

For more information on running batch jobs on Biowulf, please see: <https://hpc.nih.gov/docs/userguide.html> (<https://hpc.nih.gov/docs/userguide.html>)

Multi-threaded jobs and `sbatch` options

For multi-threaded jobs, you will need to set `--cpus-per-task`. You can do this at the command line or from within your script.

Example at the command line:

```
$ sbatch --cpus-per-task=# yourscript.sh
```

In your script:

```
#!/bin/bash
#SBATCH --job-name qc
#SBATCH --mail-type BEGIN,END
#SBATCH --cpus-per-task #

module load fastqc
fastqc -o output_dir -t $SLURM_CPUS_PER_TASK -f fastq seqfile1 seqfi`
```

Within the script we can use directives denoted by `#SBATCH` to support command line arguments such as `--cpus-per-task`. If included within the script, you will not need to call these at the command line when submitting the job. You should also pass the environment variable, `$SLURM_CPUS_PER_TASK` to the thread argument. Some other useful directives include `--job-name`, where you assign a name to the submitted job, and `--mail-type`, which you can use to direct `slurm` to send you an email when a job begins, ends, or both.

NOTE: the jobscript should always be the last argument of `sbatch`.

To see more `sbatch` options, use

```
sbatch --help
```

Some `slurm` commands

Once you submit a job, you will need to interact with the `slurm` system to manage or view details about submitted jobs.

Here are some useful commands for this purpose:

Command	Purpose	Pros & Cons
<code>squeue</code>	Information about jobs	Lots of options, odd syntax
<code>scancel</code>	Delete job(s)	Only way to cancel your jobs!
<code>sacct</code>	Job accounting for completed jobs	Useful to get a list of jobs you submitted recently (default is 'today'), odd syntax
<code>sjobs (Biowulf utility)</code>	Info about running and pending jobs	Informative but slow

Standard error and standard output

Output that you would expect to appear in the terminal (e.g., standard error and standard output) will not in batch mode. Rather, these will be written by default to `slurm#####.out` in the submitting directory, where `#####` represents the job id. These can be redirected using `--output=/path/to/dir/filename` and `--error=/path/to/dir/filename` on the command line or as an `#SBATCH` directive.

Partitions

Your job may be in a waiting phase ("Pending" or "PD") depending on available resources. You can specify a particular node partition using `--partition`.

Use `free` to see what's available.

Summary of partitions

Nodes available to all users	
norm	the default partition. Restricted to single-node jobs
multinode	Intended to be used for large-scale parallel jobs. Single node jobs are not allowed. See here for detailed information.
largemem	Large memory nodes. Reserved for jobs with memory requirements that cannot fit on the norm partition. Jobs in the largemem partition must request a memory allocation of at least 350GB.
unlimited	Reserved for jobs that require more than the default 10-day walltime. Note that this is a small partition with a low CPUs-per-user limit. Only jobs that absolutely require more than 10 days runtime, that cannot be split into shorter subjobs, or that are a first-time run where the walltime is unknown, should be run on this partition.
quick	For jobs < 4 hours long. These jobs are scheduled at higher priority. They may run on the dedicated quick partition nodes, or on the buy-in nodes when they are free.
gpu	GPU nodes reserved for applications that are built for GPUs.
visual	Small number of GPU nodes reserved for jobs that require hardware accelerated graphics for data visualization.
Buy-in nodes	
ccr*	for NCI CCR users
forgo	for individual groups from NHLBI and NINDS
persist	for NIMH users

Walltime

The default walltime, or amount of time allocated to a job, is 2 hours on the norm partition. To change the walltime, use `--time=d-hh:mm:ss`.

Here are the walltimes by partition:

```
batchlim
```

Partition	Walltime	
	Default	Maximum
norm	02:00:00	10-00:00:00
multinode	08:00:00	10-00:00:00
+turbo		08:00:00
interactive	08:00:00	1-12:00:00
gpu	02:00:00	10-00:00:00
unlimited	UNLIMITED	UNLIMITED
largemem	02:00:00	10-00:00:00
visual	08:00:00	1-12:00:00
quick	02:00:00	04:00:00
persist (NIMH)	UNLIMITED	UNLIMITED
ccr (NCI_CCR)	04:00:00	10-00:00:00
forgo (lab only)	1-00:00:00	3-00:00:00

You can change the walltime after submitting a job using

```
newwall --jobid <job_id> --time <new_time_spec>
```

Submit an actual job

Let's submit a batch job. We are going to download data from the **Sequence Read Archive (SRA)**, a public repository of high throughput, short read sequencing data. We will discuss the SRA a bit more in detail in Lesson 6. For now, our goal is to simply download multiple fastq files associated with a specific BioProject on the SRA. We are interested in downloading RNAseq files associated with **BioProject PRJNA578488**, which "aimed to determine the genetic and molecular factors that dictate resistance to WNT-targeted therapy in intestinal tumor cells".

We will learn how to pull the files we are interested in directly from SRA at a later date. For now, we will use the run information stored in `sra_files_PRJNA578488.txt`.

Let's check out this file.

```
less sra_files_PRJNA578488.txt
```


Now, let's build a script downloading a single run, `SRR10314042`, to a directory called `/data/$USER/testscript`.

```
mkdir /data/$USER/testscript
```

Open the text editor `nano` and create a script named `filedownload.sh`.

```
nano filedownload.sh
```

Inside our script we will type

```
#!/bin/bash
#SBATCH --cpus-per-task=6
#SBATCH --gres=lscratch:10

#load module
module load sratoolkit

fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314042
```

Notice our use of `SBATCH` directives inside the script.

`fasterq-dump` assigns 6 threads by default, so we are specifying `--cpus-per-task=6`. This can be modified once we get an idea of the CPUs needed for the job. `-t` assigns the location to be used for temporary files. Here, we are using `/lscratch/$SLURM_JOB_ID`, which is created upon job submission. In combination, we need to request local scratch space allocation, which we are setting to 10 GB (`--gres=lscratch:10`).

Remember:

Default compute allocation = 1 physical core = 2 CPUs
Default Memory Per CPU = 2 GB
Therefore, default memory allocation = 4 GB

Now, let's run the script.

```
sbatch filedownload.sh
```

Let's check our job status.

```
squeue -u $USER
```

Once the job status changes from `PD` (pending) to `R` (running), let's check the job status.

```
sjobs -u $USER
```

When the job completes, we should have a file called `SRR10314042.fastq`.

```
ls -lth
```

Now, what if we want to download all of the runs from `sra_files_PRJNA578488.txt`. We could use `GNU parallel`, which acts similarly to a for loop (MORE ON THIS NEXT LESSON), or we can submit multiple `fastq-dump` jobs in a job array, one subjob per run accession.

NOTE: There are instructions for running SRA-Toolkit on Biowulf [here \(https://hpc.nih.gov/apps/sratoolkit.html\)](https://hpc.nih.gov/apps/sratoolkit.html).

Swarm-ing on Biowulf

`Swarm` is for running a group of commands (job array) on Biowulf. `swarm` reads a list of command lines and automatically submits them to the system as sub jobs. To create a `swarm` file, you can use `nano` or another text editor and put all of your command lines in a file called `file.swarm`. Then you will use the `swarm` command to execute.

By default, each subjob is allocated 1.5 gb of memory and 1 core (consisting of 2 cpus) --- [hpc.nih.gov \(https://hpc.nih.gov/apps/swarm.html\)](https://hpc.nih.gov/apps/swarm.html)

```
$ swarm -f file.swarm
```

`Swarm` creates two output files for each command line, one each for `STDOUT (file.o)` and `STDERR (file.e)`. You can look into these files with the "less" command to see any important messages.

```
$ less swarm_jobid_subjobid.o  
$ less swarm_jobid_subjobid.e
```

View the `swarm` options using

```
swarm --help
```

For more information on swarm-ing on Biowulf, please see: <https://hpc.nih.gov/apps/swarm.html> (<https://hpc.nih.gov/apps/swarm.html>)

Let's create a swarm job

To retrieve all the files at once, you can create a swarm job.

```
nano set.swarm
```

Inside the swarm file type

```
#SWARM --threads-per-process 3
#SWARM --gb-per-process 1
#SWARM --gres=lscratch:10
#SWARM --module sratoolkit

fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314043
fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314044
fasterq-dump -t /lscratch/$SLURM_JOB_ID SRR10314045
```

There is advice for generating a swarm file using a `for` loop and `echo` in the [swarm user guide \(https://hpc.nih.gov/apps/swarm.html\)](https://hpc.nih.gov/apps/swarm.html).

Let's run our swarm file. Because we included our directives within the swarm file, the only option we need to include is `-f` for file.

```
swarm -f set.swarm
```

Running Interactive Jobs

Interactive nodes are suitable for testing/debugging cpu-intensive code, pre/post-processing of data, graphical application, and to GUI interface to application.

To start an interactive node, type "sinteractive" at the command line "\$" and press Enter/Return on your keyboard.

```
$ sinteractive
```

You will see something like this printed to your screen. You only need to use the `sinteractive` command once per session. If you try to start an interactive node on top of another interactive node, you will get a message asking why you want to start another node.

```
[username@biowulf ]$ sinteractive
salloc.exe: Pending job allocation 34516111
salloc.exe: job 34516111 queued and waiting for resources
```

```
salloc.exe: job 34516111 has been allocated resources
salloc.exe: Granted job allocation 34516111
salloc.exe: Waiting for resource configuration
salloc.exe: Nodes cn3317 are ready for job
srun: error: x11: no local DISPLAY defined, skipping
[username@cn3317 ]$
```

You can use many of the same options for `sinteractive` as you can with `sbatch`. The default `sinteractive` allocation is 1 core (2 CPUs) and 4 GB of memory and a walltime of 8 hours.

To terminate / cancel the interactive session use

```
exit
```

Exiting from Biowulf

To disconnect the remote connection on Biowulf, use

```
exit
```

So you think you know Biowulf?

Quiz yourself using the `hpc.nih.gov` [biowulf-quiz](https://hpc.nih.gov/training/intro_biowulf/biowulf-quiz/) (https://hpc.nih.gov/training/intro_biowulf/biowulf-quiz/).

Help Session

Let's submit some jobs on Biowulf.

Lesson 6: Downloading data from the SRA

For this lesson, you will need to login to the GOLD environment on DNAnexus.

Lesson 5 Review:

- The majority of computational tasks on Biowulf should be submitted as jobs: `sbatch` or `swarm`
- the SRA-toolkit can be used to retrieve data from the Sequence Read Archive

Learning Objectives:

1. Download data from the SRA with `fastq-dump`
 - split files into forward and reverse reads
 - download part, not all, of the data
2. Compare `fastq-dump` to `fasterq-dump`
3. Introduce `prefetch`
4. Look at XML-formatted data with `sra-stat`
5. Grab SRA run info and run accession information
6. Work with csv-formatted data using the `cut` command to isolate columns
7. Learn to automate data retrieval with the `parallel` command
8. Learn about alternative download options

For the remainder of this course, we will be working on the GOLD teaching environment in DNAnexus.

Create a project directory

Today, we will download data from the NCBI/SRA. Let's create a project directory named `biostar_class` and new folder within `biostar_class` named `sra_data` to store the data we are using today. We will use `biostar_class` for the remaining lessons in this course.

```
mkdir biostar_class
cd biostar_class
mkdir sra_data
cd sra_data
```

A brief word about sequence formats

- FASTA – commonly used text format for downstream analysis such as sequence similarity searches
- FASTQ – output format from NGS technologies with quality scores

What is the SRA?

The SRA (Sequence Read Archive) at NCBI is a large, public database of DNA sequencing data. The repository holds "short reads" generated by high-throughput next-generation sequencing, usually less than 1,000 bp.

We will download data from the SRA using the command line package `sra-toolkit`. Note, you can also download files directly from NCBI via a web browser.

Helpful documentation for the SRA:

1. [SRA handbook at NCBI \(https://www.ncbi.nlm.nih.gov/books/NBK47528/\)](https://www.ncbi.nlm.nih.gov/books/NBK47528/)
2. [SRA Toolkit Documentation \(NCBI\) \(https://github.com/ncbi/sra-tools/wiki\)](https://github.com/ncbi/sra-tools/wiki)

SRA terms to know

NCBI BioProject: PRJN#### (example: [PRJNA257197 \(https://www.ncbi.nlm.nih.gov/bioproject/PRJNA257197/\)](https://www.ncbi.nlm.nih.gov/bioproject/PRJNA257197/)) contains the overall description of a single research initiative; a project will typically relate to multiple samples and datasets.

NCBI BioSample: SAMN#### or SRS#### (example: SAMN03254300) describe biological source material; each physically unique specimen is registered as a single BioSample with a unique set of attributes.

SRA Experiment: SRX#### a unique sequencing library for a specific sample

SRA Run: SRR#### or ERR#### (example: SRR1553610) is a manifest of data file(s) linked to a given sequencing library (experiment). --- Biostar handbook (XIII SEQUENCING DATA)

Using fastq-dump

`fastq-dump` and `fasterq-dump` can be used to download FASTQ-formatted data. Both download the data in SRA format and convert it to FASTQ format.

```
fastq-dump SRR1553607
```

creates the file:

```
SRR1553607.fastq
```

Check the file to make sure it is in fastq format. How would you do this? What is FASTQ format?

What are "spots"?

Spots are a legacy term referring to locations on the flow cell for Illumina sequencers. All of the bases for a single location constitute the spot including technical reads (e.g., adapters, primers, barcodes, etc.) and biological reads (forward, reverse). In general, you can think of a spot as you do a read. For more information on spots, see the [linked discussion \(https://www.biostars.org/p/12047/\)](https://www.biostars.org/p/12047/) on Biostars. When downloading "spots", always split the spots into the original files using:

```
--split-files
```

For paired-end reads, we need to separate the data into two different files like this:

```
fastq-dump --split-files SRR1553607
```

which creates

```
SRR1553607_1.fastq  
SRR1553607_2.fastq
```

NOTE: There is an additional option `--split-3` that will split the reads into forward and reverse files and a third file with unmatched reads. Since many bioinformatic programs require matching paired end reads, this is the preferred option.

`fastq-dump` first downloads the data in SRA format, then converts it to FASTQ. If we want to work with a subset of the data, for example the first 10,000 reads, we can use `-X`:

```
fastq-dump --split-files -X 10000 SRR1553607 --outdir SRR1553607_sub:
```

We have additionally included `--outdir` to save our subsetted data to a different directory, so that we do not overwrite our previous downloads.

To see other `fastq-dump` options, use

```
fastq-dump --help
```

Other options of interest:

- `--gzip` or `bz ip2` allows you to compress the reads
- `--skip-technical` allows you to only download biological reads

To generate an XML report on the data that shows us the "spot" or read count, and the count of bases "base_count", including the size of the data file:

```
sra-stat --xml --quick SRR1553607
```

which yields this data in XML format:

```
<Run accession="SRR1553607" spot_count="203445" base_count="41095890"
  <Member member_name="A" spot_count="203445" base_count="41095890" t
  <Size value="25452196" units="bytes"/>
  <AlignInfo>
  </AlignInfo>
  <QualityCount>
    <Quality value="2" count="3523771"/>
    <Quality value="5" count="40006"/>
    <Quality value="6" count="30160"/>
    <Quality value="7" count="79961"/>
    <Quality value="8" count="80987"/>
    <Quality value="9" count="21904"/>
    <Quality value="10" count="63700"/>
    <Quality value="11" count="23695"/>
    <Quality value="12" count="42479"/>
    <Quality value="13" count="24910"/>
    <Quality value="14" count="22036"/>
    <Quality value="15" count="135747"/>
    <Quality value="16" count="56174"/>
    <Quality value="17" count="81514"/>
    <Quality value="18" count="123393"/>
    <Quality value="19" count="79256"/>
    <Quality value="20" count="269807"/>
    <Quality value="21" count="68295"/>
    <Quality value="22" count="125196"/>
    <Quality value="23" count="267733"/>
    <Quality value="24" count="287688"/>
    <Quality value="25" count="296721"/>
    <Quality value="26" count="338062"/>
    <Quality value="27" count="507890"/>
    <Quality value="28" count="226423"/>
    <Quality value="29" count="569612"/>
    <Quality value="30" count="813014"/>
    <Quality value="31" count="1309505"/>
```



```
<Quality value="32" count="812951"/>
<Quality value="33" count="2398417"/>
<Quality value="34" count="2324453"/>
<Quality value="35" count="10399039"/>
<Quality value="36" count="1250313"/>
<Quality value="37" count="2681177"/>
<Quality value="38" count="1210457"/>
<Quality value="39" count="2744683"/>
<Quality value="40" count="2268539"/>
<Quality value="41" count="5496222"/>
</QualityCount>
<Databases>
  <Database>
    <Table name="SEQUENCE">
      <Statistics source="meta">
        <Rows count="203445"/>
        <Elements count="41095890"/>
      </Statistics>
    </Table>
  </Database>
</Databases>
</Run>
```

where we can see the spot count, the base count, and the number of reads corresponding to quality scores.

```
spot_count="203445" base_count="41095890"
```

Side bar: (XML) is Extensible Markup Language, a document format that is both human and machine-readable. XML aims for: "simplicity, generality and usability across the Internet".

Using `seqkit stat`

We can get similar information from our downloaded files using a program called `seqkit` (<https://bioinf.shenwei.me/seqkit/>).

First, let's see what options are available?

```
seqkit --help
```

We can see that `seqkit stats` provides "simple statistics of FASTA/Q files". Let's try it out.

```
seqkit stat SRR1553607_1.fastq
```

fasterq-dump

We have already seen `fasterq-dump` in action (See Lesson 5). `fasterq-dump` is faster than `fastq-dump` because it uses multi-threading (default `--threads 6`). `--split-3` and `--skip-technical` are defaults with `fasterq-dump`, so you do not need to worry about specifying how you want the files to be split.

However, you can not grab a subset of the file like you can with `fastq-dump` nor can you compress the file during download, so `fasterq-dump` is not necessarily a replacement for `fastq-dump`.

Using prefetch

Both `fastq-dump` and `fasterq-dump` are faster when following `prefetch` (<https://github.com/ncbi/sra-tools/wiki/08.-prefetch-and-fasterq-dump>), and `fasterq-dump` paired with `prefetch` is the fastest way to pull the files from the SRA. There are alternative options, which we will mention later.

`prefetch` will first download all of the necessary files to your working directory. Runs are downloaded in the SRA format (compressed).

```
mkdir prefetch
cd prefetch
prefetch SRR1553607
ls -l
```

`fasterq-dump` will then convert the files to the `fastq` format.

```
fasterq-dump SRR1553607
```

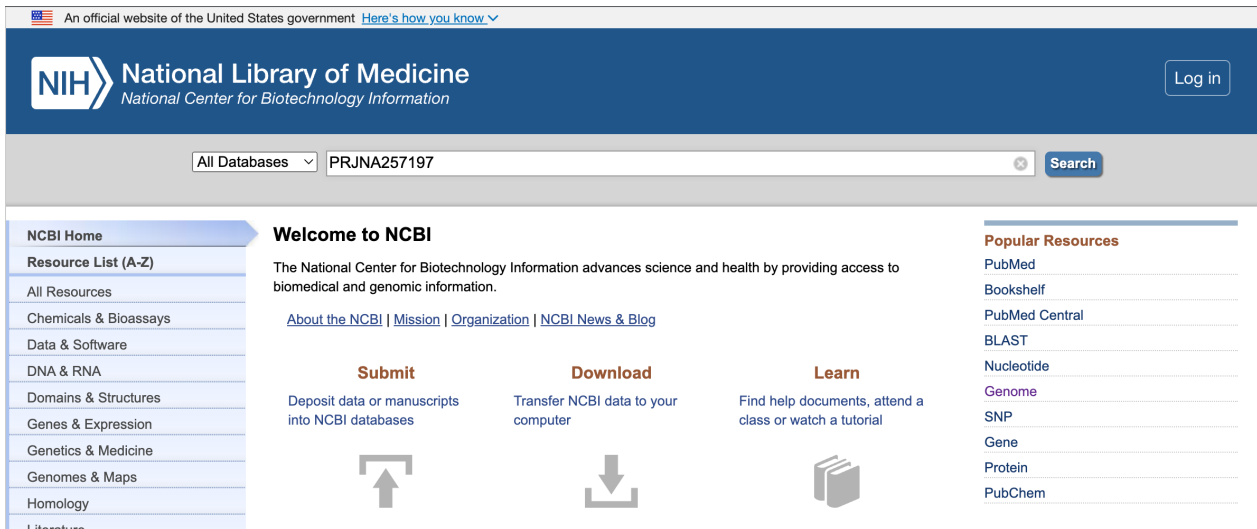
Navigating NCBI SRA

Where can we get an accession list or run information for a particular project?

Navigating the NCBI website can be challenging. From a user perspective, nothing is as straight forward as you would expect. For the sequence read archive (SRA), there are fortunately some options. There are the convoluted e-utilites, which can grab run information

without navigating to the NCBI website, or there is the [SRA Run Selector \(https://www.ncbi.nlm.nih.gov/Traces/study/\)](https://www.ncbi.nlm.nih.gov/Traces/study/). The Run Selector is nice because you can filter the results for a given BioProject and obtain only the accessions that interest you in a semi user-friendly format. We can search directly from the SRA Run Selector, or simply go to the main NCBI website and begin our search from there. Let's see the latter example, as this is likely the primary way you will attempt to find data in the future.

Step 1: Start from the NCBI homepage. Type the BioProject ID (PRJNA257197) into the search field.



The screenshot shows the NCBI homepage. At the top, there is a navigation bar with the NIH logo and the text "National Library of Medicine National Center for Biotechnology Information". A search bar is located below the navigation bar, containing the text "All Databases" and "PRJNA257197". To the right of the search bar is a "Search" button. Below the search bar, there is a "Welcome to NCBI" section with a description of the center's mission and a list of links: "About the NCBI", "Mission", "Organization", and "NCBI News & Blog". There are three main sections: "Submit" (Deposit data or manuscripts into NCBI databases), "Download" (Transfer NCBI data to your computer), and "Learn" (Find help documents, attend a class or watch a tutorial). To the right of these sections is a "Popular Resources" section with links to PubMed, Bookshelf, PubMed Central, BLAST, Nucleotide, Genome, SNP, Gene, Protein, and PubChem. On the left side, there is a "Resource List (A-Z)" section with links to All Resources, Chemicals & Bioassays, Data & Software, DNA & RNA, Domains & Structures, Genes & Expression, Genetics & Medicine, Genomes & Maps, Homology, and Literature.

Step 2: In the search results, select the entries next to the SRA.

Results found in 8 databases

BIOPROJECT

Zaire ebolavirus Genome sequencing

Zaire ebolavirus

Zaire ebolavirus sample sequencing from the 2014 outbreak in Sierra Leone, West Africa.

PRJNA257197

[BioSample](#) [PubMed](#)

Literature	Genes	Proteins
Bookshelf 1	Gene 0	Conserved Domains 0
MeSH 0	GEO DataSets 0	Identical Protein Groups 240
NLM Catalog 0	GEO Profiles 0	Protein 0
PubMed 1	HomoloGene 0	Protein Family Models 0
PubMed Central 9	PopSet 1	Structure 0

Genomes	Clinical	PubChem
Assembly 0	ClinicalTrials.gov 0	BioAssays 0
BioCollections 0	ClinVar 0	Compounds 0
BioProject 2	dbGaP 0	Pathways 0
BioSample 0	dbSNP 0	Substances 0
Genome 1	dbVar 0	
Nucleotide 0	GTR 0	
SRA 891	MedGen 0	
Taxonomy 0	OMIM 0	

Step 3: From the SRA results, select "Send results to Run Selector".

National Library of Medicine
National Center for Biotechnology Information
Log in

SRA Search

Create alert Advanced Help

Access
Public (891)

Source
RNA (891)

Library Layout
paired (891)

Platform
Illumina (891)

Strategy
other (891)

Data in Cloud
GS (891)
S3 (891)

File Type
bam (891)

[Clear all](#)

[Show additional filters](#)

Summary ▾ 20 per page ▾ Send to: ▾ [Filters: Manage Filters](#)

[Zaire ebolavirus Genome sequencing - BioProject](#)

Zaire ebolavirus sample sequencing from the 2014 outbreak in Sierra Leone, West Africa.

Genome sequencing project
Accession: PRJNA257197

View results as an expanded interactive table using the RunSelector. [Send results to Run selector](#)

Search results

Items: 1 to 20 of 891 << First < Prev Page 1 of 45 Next > Last >>

- [Zaire ebolavirus genome sequencing from 2014 outbreak in Sierra Leone: Sample W220.0](#)
- 1. 1 ILLUMINA (Illumina HiSeq 2500) run: 8.3M spots, 1.7G bases, 997.4Mb downloads
Accession: SRX994253
- [Zaire ebolavirus genome sequencing from 2014 outbreak in Sierra Leone: Sample W219.0](#)
- 2. 1 ILLUMINA (Illumina HiSeq 2500) run: 2.7M spots, 546.4M bases, 292.2Mb downloads
Accession: SRX994252
- [Zaire ebolavirus genome sequencing from 2014 outbreak in Sierra Leone: Sample W218.0](#)

Results by taxon

Top Organisms [\[Tree\]](#)

Zaire ebolavirus (856)

unidentified virus (35)

Search in related databases

Database	Access		all
	public	controlled	
BioSample			
BioProject	1		1
dbGaP			
GEO Datasets			

Find related data

Database:

Step 4: From there, you can download the metadata or accession list.

Copy and paste the accession list into a file using `nano`. Save to `runaccessions.txt`. Now use `head` to grab the first few results.

```
head -n 3 runaccessions.txt > runids.txt
```

E-utilities

{{Sdet}}

Using e-utilities to programmatically grab the run info{{Esum}}

`esearch` and `efetch`, can be used to query and pull information from [Entrez \(https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html\)](https://www.ncbi.nlm.nih.gov/Web/Search/entrezfs.html). They aren't the easiest to understand or use, but you can use them to get the same run info that we grabbed using the Run Selector with the following one liner:

```
esearch -db sra -query PRJNA257197 | efetch -format runinfo > runinfo.csv
```

Here we provide the BioProject ID to `esearch`, which queries the SRA. That info can be piped to `efetch`, which will pull down the run info in comma separated format, which we can save to file using `>`.

Then we can use a combo of `cat`, `cut`, `grep`, and `head` to grab only the accession info we are interested in:

```
cat runinfo.csv | cut -f 1 -d ',' | grep SRR | head > runids.txt
```

Here, we opened the file with `cat`, piped it to the `cut` command, piped it to `grep` for only the accession IDs (skipping the column header), and get only the first few results.

So what is `cut`? It is a unix command, that is used to cut out selected portions of the file (`man cut` for more info). The `-f` option specifies which field to cut (the first field), and `-d` tells the program which delimiter to use. We know this is a comma-separated value file, so we do `-d ','` to specify the comma.

{{Edet}}

Using the "parallel" tool for automating downloads

Now that we have accession numbers to work with, let's use `parallel` and `fastq-dump` to download the data.

GNU `parallel` executes commands in "parallel", one for each CPU core on your system. It can serve as a nice replacement of the `for` loop. See [Tool: Gnu Parallel - Parallelize Serial Command Line Programs Without Changing Them \(https://www.biostars.org/p/63816/\)](https://www.biostars.org/p/63816/)

```
cat runids.txt | parallel -j 1 fastq-dump -X 10000 --split-files {}
```

This gives us twenty files, two files for each run and 10 runs. It takes the place of the multiple `fastq-dump` commands we would need to use.

What's up with the curly braces `{}`?

The curly braces are default behavior for `parallel`. This is the default replacement string; when empty, it appends inputs. So the following gets the same result.

```
cat runids.txt | parallel fastq-dump -X 10000 --split-files
```

The real power of `parallel` comes when using something between the brackets such as `{.}`, `{/}`, and `{//}`, like this...

```
nano filelist
```

Include the following:

```
/dir/subdir/file.txt  
/other/list.csv  
/raw/seqs.fastq
```

Save the file -> `Ctrl O`, Yes, `Ctrl X`.

```
cat filelist | parallel echo {} "without extension" {.}
```

```
/dir/subdir/file.txt without extension /dir/subdir/file  
/other/list.csv without extension /other/list  
/raw/seqs.fastq without extension /raw/seqs
```

This replaces the extension.

```
cat filelist | parallel echo {} "without path" {/}
```

```
/dir/subdir/file.txt without path file.txt  
/other/list.csv without path list.csv  
/raw/seqs.fastq without path seqs.fastq
```

This removes the path.

```
cat filelist | parallel echo {} "with root" {/}
```

```
/dir/subdir/file.txt with root /dir/subdir  
/other/list.csv with root /other  
/raw/seqs.fastq with root /raw
```

This keeps only the path.

Note: `parallel` will default to 1 concurrent job per available CPU. On a shared system like `helix` with 48 CPUs, where users do run `fast(er)q-dump`, that can cause an overload. Even when submitting a job on `Biowulf`, you may not want to run as many concurrent jobs as there are allocated CPUs (e.g. if you ran `fasterq-dump` with 2 threads and you have 12 CPUs allocated -- jobs would be 6). Therefore, it is good practice to always specify the `-j` flag, which assigns the number of jobs for the `parallel` command.

European Nucleotide Archive (ENA)

Everything in the SRA is also in the ENA (See [PRJNA257197 on ENA \(https://www.ebi.ac.uk/ena/browser/view/PRJNA257197?show=reads\)](https://www.ebi.ac.uk/ena/browser/view/PRJNA257197?show=reads)). Files in the ENA share the same naming convention as the SRA but are stored directly as gzipped fastq files and bam files. You can easily use `wget` or `curl` to pull files directly from the ENA without using the `sratoolkit`. More on `wget` and `curl` in the next lesson. Check out the [ENA training modules \(https://ena-docs.readthedocs.io/en/latest/retrieval/file-download.html#\)](https://ena-docs.readthedocs.io/en/latest/retrieval/file-download.html#) for more information on downloading data from the ENA, including examples.

There is also a fantastic web service called [SRA Explorer \(https://sra-explorer.info/#\)](https://sra-explorer.info/#) that can be used to easily generate code for file downloads from the SRA, if you are looking to download a maximum of 500 samples. Use caution, the SRA-explorer is not associated with the ENA or NCBI, and program support / updates cannot be guaranteed.

Help Session

Let's search for and download some data.

Lesson 7: Downloading the RNA-Seq Data and Dataset Overview

Lesson Review

- `pwd` (print working directory)
- `ls` (list)
- `touch` (creates an empty file)
- `nano` (basic editor for creating small text files)
- using the `rm` command to remove files. Be careful!
- `mkdir` (make a directory) and `rmdir` (remove a directory, must be empty of all files)
- `cd` (change directory), by itself will take you home, `cd ..` (will take you up one directory), `cd /results_dir/exp1` (go directly to this directory)
- `mv` (for renaming files or moving files)
- `less` (for viewing files, "more" is the older version of this)
- `man` command (for viewing the man pages when you need help on a command)
- `cp` (copy) for copying files
- Flags and command options - making programs do what they do
- Wildcards (e.g., `*`)
- Tab complete - for less typing
- Accessing user history with the "up" and "down" arrows on the keyboard
- `cat`, `head`, and `tail` - print to screen, print first few lines to the screen, print last few lines to the screen
- Working with file content (`<`, `>`, `>>`)
- Combining commands with pipe (`|`). Where the heck is pipe anyway?
- Finding information in files with `grep`
- Performing repetitive actions with Unix (`for` loop), GNU `parallel`
- Permissions (`chmod`, `chown`)
- `wc` - number of lines (`-l`), words (`-w`), and bytes (`-c`, usually one byte per character); for number of characters use `-m`.
- `grep`- search files using regular expressions
- `cut` - cuts selected portions of a file
- `fastq-dump` and `fastq-dump` - SRA file download
- `ssh` - secure shell protocol for remote login to Biowulf / Helix

Learning Objectives

- Introduce the RNA-Seq data

- Use `wget` and `curl` to download files
- Learn to compress / decompress and unarchive files
- Learn `sed` and `awk` for file editing

Getting Project files

UHR and HBR data

For this class, we are going to work with data from and associated with two commercially available sets of RNA samples, Universal Human Reference (UHR) and Human Brain Reference (HBR).

- UHR - bulk RNA from 10 cancer cell lines.
- HBR - bulk RNA from 23 brains; subjects were Caucasian, both sexes, and mostly between 60 and 80 years of age.

The data are paired-end with three replicates from each set (UHR, HBR).

These data are from:

Informatics for RNA-seq: A web resource for analysis on the cloud. 11(8):e1004393. PLoS Computational Biology (2015) by Malachi Griffith, Jason R. Walker, Nicholas C. Spies, Benjamin J. Ainscough, Obi L. Griffith.

There is also an accompanying [tutorial \(https://rnabio.org/\)](https://rnabio.org/).

Downloading the data

Let's download the data and learn how to decompress it. First, we will create a place to store the data.

Go to the directory you created for working with class material. If you haven't created a class directory (`biostar_class`), do that now.

```
mkdir biostar_class
```

Now, change to that directory.

```
cd biostar_class
```

Create a directory for the data we are going to download.

```
mkdir -p RNA_Seq/raw_data
```

What does the `-p` flag do? Now, go to the `raw_data` directory you have created.

```
cd RNA_Seq/raw_data
```

Now that we're in the correct directory, we will use `curl` to download some bulk RNA-Seq data.

```
curl http://data.biostarhandbook.com/rnaseq/projects/griffith/griffith-data.tar.gz
```

Let's take a look at this Unix command line... The `curl` command is used to retrieve data from web sites. A similar command is `wget`. The Unix system you are working with may have either `curl` or `wget` installed. To see which is active on your system, just type the command at the command line like this...

```
wget
```

You may see an error like this if `wget` is not installed.

```
-bash: wget: command not found
```

Next, try the `curl` command.

```
curl
```

If `curl` is active on the system, you may see something like this...

```
curl: try 'curl --help' or 'curl --manual' for more information
```

We can do as the instructions say...

```
curl --help
```

and see information on the usage of `curl`. So it looks like `curl` is installed on this system.

Moving on. Let's take a look at this command line. We now know what `curl` means, but how about the rest of it. The URL `http://data.biostarhandbook.com/rnaseq/projects/griffith/griffith-data.tar.gz` represents the "path" to this data. As we have discussed, paths are a very important concept in Unix. An incorrect path can result in frustrating "file not found" errors.

The path to the file `griffith-data.tar.gz` is `data.biostarhandbook.com/rnaseq/projects/griffith`, which can be translated as - on the `data.biostarhandbook.com` server, there is a directory (folder) named `rnaseq` that contains `projects`, which contains `griffith` and subsequently the file `griffith-data.tar.gz`. Notice how there are no blank spaces in the path name - Unix does not handle spaces in file names, directories, or paths easily.

Another way to get to this data file is via your browser. Open a browser window and enter `http://data.biostarhandbook.com/rnaseq/projects/griffith`. You will see an index page listing all the directories at this location.

For

example:

Index of /rnaseq/projects/griffith/

../		
GRCh38.chrom22.cdna.fa.gz	26-Nov-2016 18:27	1M
griffith-align.sh	04-Dec-2017 16:03	1100
griffith-count.sh	04-Dec-2017 16:03	1530
griffith-data.tar.gz	05-Dec-2018 14:03	125M
griffith-getdata.sh	04-Dec-2017 15:51	1445
griffith-kallisto.sh	20-Jan-2017 17:15	2377

If you look closely, you will find a file named `griffith-data.tar.gz`. What happens if you click on this link? Does it download? Can you open a tar file in the Mac environment? How about on PC? How would you do it?

Okay, let's take a look at the file name `griffith-data.tar.gz`. What does the `.tar.gz` extension mean? `tar` refers to "tape archive" and is used to archive a set of files into a single file. The `tar` command can also be used to compress an archive using some form of compression. The `-z` flag, for example, compresses the archive using `gzip`, which results in the extension `.gz`. Note: `gzip` is a command on its own and can be run independently.

How do we deal with `tar.gz` files? On a Unix system, we `untar` and `unzip` the file using `tar` with the flags `-x`, `-v`, and `-f`. `tar` auto-detects the compression type, so nothing specific is needed to handle the compression type.

```
tar -xvf filename.tar
```

What does `-xvf` mean? If we check the `man` page for `tar`, we could find out...

```
man tar
```

`-x` means - extract to disk from the tar (tape archive),


```
+  
@@@DDDDDDFFFFFIIIII;??::::9?99?G8;)9/8'787.)77;@==D=?;?A>D?@BDC@?CC=?BI
```

Keep in mind, there are several Unix commands that can be used to look at the contents of files, each has it's own flags/options and is used slightly differently. For example:

```
less  
more  
cat  
head  
tail
```

Less, in particular, can also be used to examine zipped files with the help of `lesspipe`, on certain unix systems. On Biowulf, for example, you can use `less` to view compressed /archived files.

A brief introduction to awk and sed

What is awk?

A scripting language that can be used for manipulating data and generating reports.

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then performs the associated actions. ---<https://www.geeksforgeeks.org/awk-command-unixlinux-examples/> (<https://www.geeksforgeeks.org/awk-command-unixlinux-examples/>)

awk works line by line. The basic syntax is

```
awk 'CONDITION { ACTIONS }'
```

For each line, awk tries to match the `CONDITION`, and if that condition matches, it performs the `ACTIONS`. ---[Biostar Handbook, The Art of Bioinformatics Scripting](https://www.biostarhandbook.com/books/scripting/programming-with-awk.html) (<https://www.biostarhandbook.com/books/scripting/programming-with-awk.html>)

There can be multiple conditions and actions.

Let's see `awk` in action. Let's return to `runinfo.csv` from Lesson 6. We can use `awk` to print columns of interest.

For example,

```
cd ../../sra_data
awk -F ',' '{ print $1,$4,$7 }' runinfo.csv | head > awk_example.txt
```

Here, the action is to simply print the first `$1`, fourth `$4`, and seventh `$7` columns from `runinfo.csv`. Since there is no condition to be met, `awk` acts on all lines. The `-F` flag is used to specify the field separator. In this case, we are looking at a comma separated file, so we use `,`. If we also want the output to be comma separated, we need to use the special `awk` variable `OFS`.

```
awk -F ',' '{ OFS=","; print $1,$4,$7 }' runinfo.csv
```

There are many resources online for getting started with `awk`. There is a chapter in the *Biostar Handbook*, *IV Awk Programming* (<https://www.biostarhandbook.com/books/scripting/programming-with-awk.html>) in the *Art of Bioinformatics Scripting*. You may also find [this article series](https://catonmat.net/awk-one-liners-explained-part-one) (<https://catonmat.net/awk-one-liners-explained-part-one>) explaining `awk` one-liners handy.

What is `sed`?

`sed` stands for stream editor. Functions include searching, find and replace, and insertion / deletion.

`sed` is often used for its "find and replace" capabilities.

For example, let's replace "SRR" in `awk_example.txt` with "ACC".

```
sed 's/SRR/ACC/' awk_example.txt
```

Notice the single quotes containing our substitution phrase. The `s` specifies `sed`'s substitution command, while the `/s` separate the search pattern and the replacement string. The first occurrence in each line will be substituted. To substitute across all occurrences in a line use the global option `'s/SRR/ACC/g'`.

You can pair `sed` with regular expressions. For example, let's say we want to replace a few of the run accessions, those ending with a "17", "18", or "19", with "Unknown".

```
sed 's/SRR197291./Unknown/' awk_example.txt
```

The `.` in this context means any character.

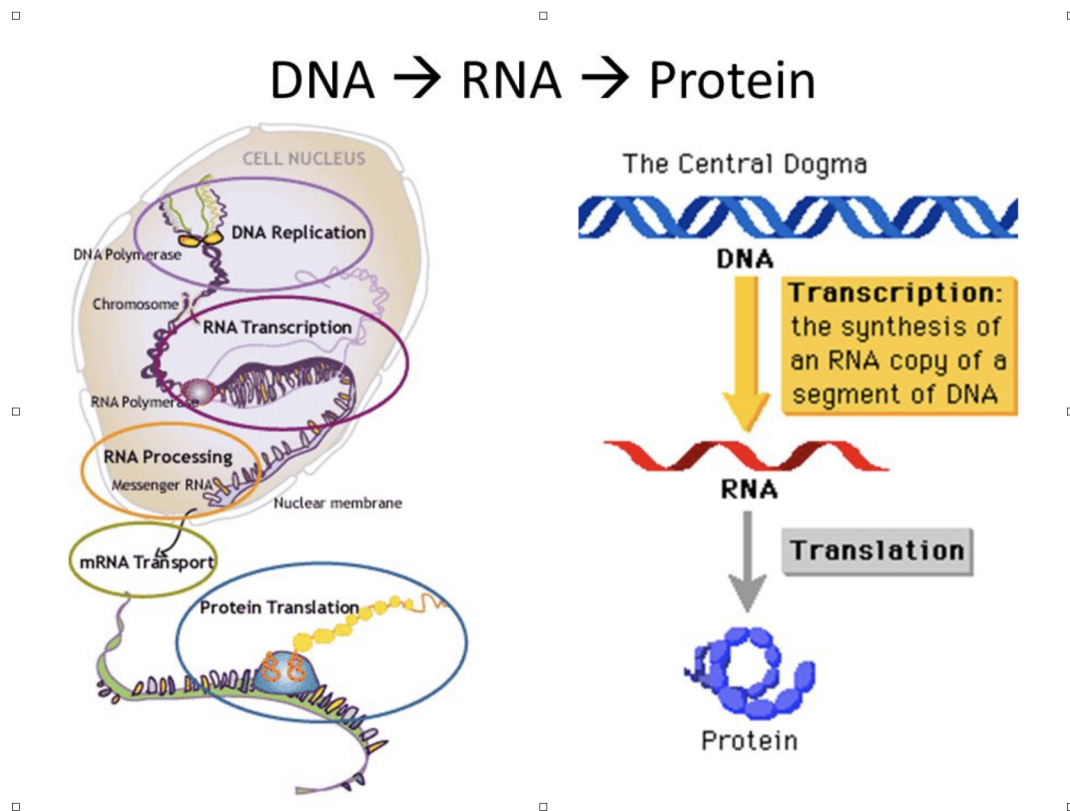
For more information, see <https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/> (*https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/*). Also, check out [these handy one-liners and their explanations](https://catonmat.net/sed-one-liners-explained-part-one) (*https://catonmat.net/sed-one-liners-explained-part-one*).

Help Session

For this help session, you will be downloading the Golden Snidget data. Practice materials are located [here](#).

Introduction to RNASeq

RNA-SEQ Overview



What is RNASEQ ?

RNA-Seq (RNA sequencing), uses next-generation sequencing (NGS) to reveal the presence and quantity of RNA in a biological sample at a given moment. (Wikipedia) Strictly speaking this could be any type of RNA (mRNA, rRNA, tRNA, snoRNA, miRNA) from any type of biological sample. For the purpose of this talk we will be limiting ourselves to mRNA.

Technically, with a few exceptions, we are not actually sequencing mRNA but rather cDNA.

(RNASeq is only valid within the context of Differential Expression)

RNASEQ - WorkFlow

A typical RNASEQ experiment involves several steps, only one of which falls within the realm of bioinformatics. Namely the Data Analysis step.

- Experimental Design
 - What question am I asking

- How should I do it (does it need to be done)
 - **Sample Preparation**
 - Sample Prep
 - Library Prep
 - Quality Assurance
 - **Sequencing**
 - Technology/Platform
 - Detail Choices
 - **Data Analysis (Computation)**
 - Quality Control
 - Alignment
 - Quantitation
 - Differential Expression
 - Biological Meaning
-

We will now examine each of these steps, highlight the major components of each, and touch briefly on some of the more critical steps and pitfalls.

Generating the Data - Experimental Design

A good experimental design is vital for the success of any RNA-SEQ experiment. Before you begin the experiment make sure you have a clear understanding of the technique and how to avoid costly mistakes that can produce unanalysable data. Listed below are some of the things you should consider before starting your experiment

Only Sequence the RNA of interest

- Remember ~90% of RNA is ribosomal RNA. Therefore enrich your total RNA sample by: polyA selection (oligo-dT affinity) of mRNA (eukaryote), or rRNA depletion - RiboZero is typically used (costs extra)

Remember

- RNA-SEQ looks at steady state mRNA levels which is the sum of transcription and degradation
- Protein levels are assumed to be driven by mRNA levels
- RNA-SEQ can measure relative abundance not absolute abundance
- RNA-SEQ is really all about sequencing cDNA

What question(s) are you asking?

Your answers to the following questions will dictate many of your choices with respect of the appropriate methodologies to be employed.

- Which gene are expressed?
- Which genes are differentially expressed?
- Are different splicing isoforms expressed?
- Are there novel genes or isoforms expressed?
- Are you interested in structural variants or SNPs, indels Are you interested in non-coding RNAs
- Does your interest lie in micro RNAs
- If this a standalone experiment, a pilot, or a “fishing trip”

Read Choices

For any NGS experiment you will have to make choices about the following sequencing options. Unfortunately, there is an inverse relationship between accuracy and cost.

- **Read Depth**
 - More depth needed for lowly expressed genes
 - Detecting low fold differences need more depth
- **Read Length**
 - The longer the length the more likely to map uniquely
 - Paired read help in mapping and junctions
- **Replicates**
 - Detecting subtle differences in expression needs more replicates
 - Detecting novel genes or alternate iso-forms need more replicates

*Increasing depth, length, and/or replicates increase costs

Replicates

Technical Replicates

- It's generally accepted that they are not necessary because of the low technical variation in RNASeq experiments

Biological Replicates (Always useful)

- Not strictly needed for the identification of novel transcripts and transcriptome assembly.
- Essential for differential expression analysis - must have 3+ for statistical analysis
- Minimum number of replicates needed is variable and difficult to determine:
 - 3+ for cell lines
 - 5+ for inbred samples
 - 20+ for human samples (rarely possible)

- More is always better

Data Analysis Questions

Make sure you have a clear plan for storing, managing and analysing your data. Also, ensure you have a method to capture all the pertinent metadata and document the data analysis steps that have been taken.

- Where will the primary data be stored (fastq)? Where will the processed data be stored (bam)? Who will do the primary analysis?
- Who will do the secondary analysis?
- Where will the published data be deposited and by who? (what metadata will they require)
- Are you doing reproducible science?

If you are not going to analyse the data yourself talk to the people who will be analyzing your data **BEFORE** doing the experiment*

Best Practice Guidelines from Bioinformatic Core (CCBR):

1. Factor in at least 3 replicates (absolute minimum), but 4 if possible (optimum minimum). Biological replicates are recommended rather than technical replicates.
2. Always process your RNA extractions at the same time. Extractions done at different times lead to unwanted batch effects.
3. There are 2 major considerations for RNA-Seq libraries: • If you are interested in coding mRNA, you can select to use the mRNA library prep. The recommended sequencing depth is between 10-20M paired-end (PE) reads. Your RNA has to be high quality (RIN > 8). • If you are interested in long noncoding RNA as well, you can select the total RNA method, with sequencing depth ~25-60M PE reads. This is also an option if your RNA is degraded.
4. Ideally to avoid lane batch effects, all samples would need to be multiplexed together and run on the same lane. This may require an initial MiSeq run for library balancing. Additional lanes can be run if more sequencing depth is needed.
5. If you are unable to process all your RNA samples together and need to process them in batches, make sure that replicates for each condition are in each batch so that the batch effects can be measured and removed bioinformatically.
6. For sequence depth and machine requirements, visit Illumina Sequencing Coverage website

For cost estimates, visit [Sequencing Facility pricing for NGS](#) For further assistance in planning your RNA-Seq experiment or to discuss specifics of your project, please contact us by email: CCBR@mail.nih.gov OR visit us during office hours on Fridays 10am to noon (Bldg37/

Room3041). For cost and specific information about setting up an RNA-Seq experiment, please visit the Sequencing Facility website or contact Bao Tran

Generating the Data

General Rules for Sample Preparation

Ignoring these simple guidelines will greatly increase the chances that your data will be unanalysable and/or your experiment unpublishable.

- Prepare all samples at the same time or as close as possible. The same person should prepare all samples
- Do not prepare “experiment” and “control” samples on different days or by different people. (Batch effects).
- Use high quality means to determine sample quality (RNA Integrity Number) (RIN >0.8) and quantity, and size (Tapestation, Qubit, Bioanalyzer)
- Don't assume everything will work the first time (do pilot experiments) or every time (prepare extra samples)

Sample Amounts Guidelines

Type of Library	Minimum DNA/RNA Requirement for Library Construction	Recommended DNA/RNA for Optimal Library Construction	Maximum Sample Volume Requirement for Library Construction	Additional Requirements
mRNA Sequencing	100 ng	1 µg	50 µL	RIN should be at least 8.0, DNase treated
mRNA ultralow Clontech	100 pg	10 ng	10 µL	RIN should be at least 8.0, DNase treated
microRNA Sequencing	100 ng	1 µg	6 µL	
Total RNA Sequencing	100 ng	1 µg	10 µL	DNase treated, FFPE and degraded RNA can be used
Total RNA ultralow	10 ng	1 µg	10 µL	DNase treated, FFPE and degraded RNA can be used

RNA-Seq Sample Recommendations (CCBR)

QC Metric Guidelines	mRNA	total RNA
RNA Type(s)	Coding	Coding + non-coding
RIN	8 [low RIN = 3' bias]	> 8
Single-end vs Paired-end	Paired-end	Paired-end
Recommended Sequencing Depth	10-20M PE reads	25-60M PE reads
FastQC	Q30 > 70%	Q30 > 70%
Percent Aligned to Reference	70%	> 65%
Million Reads Aligned Reference	7M PE reads (or > 14M reads)	16.5M PE reads (or > 33M reads)
Percent Aligned to rRNA	< 5%	< 15%
Picard RNAseqMetrics	Coding > 50%	Coding > 35%
Picard RNAseqMetrics	Intronic + Intergenic < 25%	Intronic + Intergenic < 40%

Sequencing

Illumina Sequencing Platforms

By far the most popular platforms for RNA-Seq experiments are the Illumina family of sequencers. All are Sequencing by Synthesis (SbS) and produce Short read lengths (50 to 300 bp).

Illumina

Sequencing by Synthesis (SbS)
/NovaSeq/HiSeq/NextSeq/MiSeq
Short read length (50 to 300 bp)

Selection driven by cost, precision,
speed, number of samples and
number of reads required

Consult with the Sequencing Core



Illumina
NovaSeq



Illumina
NextSeq



Illumina
MiSeq

Consult with the Sequencing Core as to which is most appropriate for your experiment. The appropriate selection will be driven by cost, precision, speed, number of samples and number of reads required

Long Read Sequencing Platforms

Long read platforms by PacBio and Oxford Nanopore are becoming more popular and offer significant advantages over short read technologies in certain circumstances.

PacBio

120,000 bases per molecule, with maximum read lengths > 200,000 bases. Good for repetitive regions and isomers, modified bases.

**PacBio Sequel II****Oxford Nanopore**

Direct DNA or RNA sequencing (Max length 2 Mb) Good for modified bases, repetitive regions, isomers, small genomes.



MinION



GridION

Consult with the Sequencing Cores

Oxford Nanopore

Data Analysis Overview

RNASEQ - Data Analysis WorkFlow

Mostly Computational intensive task requiring significant computer hardware.

- **Quality Control**
 - Sample quality and consistency
 - Is Trimming appropriate - quality/adaptors
 - **Alignment/Mapping**
 - Reference Target (Sequence and annotation) Alignment Program
 - Alignment Parameters
 - Mark Duplicates
 - Post-Alignment Quality Assurance
 - **Quantification** *Counting Method and Parameters
-

Generally less computational intensive task doable on a personal computer.

- **Quantification**
 - Differential Expression - statistics
 - **Visualization**
 - Visual inspection - IGV
 - Data representation - scatter, violin plots, heat-maps
 - **Biological Meaning**
 - Gene Set Enrichment
 - Pathway Analysis
-

Computational Considerations THE GOOD NEWS

For the most part the computational aspects have been taken care of for you.

(no need to develop new algorithms or code).

There are pre-built workflows that can automate many of the processes involved, and facilitate reproducibility.

Computational Considerations THE BAD NEWS

Like most of NGS data analysis, the complexity of RNA-Seq data analysis revolves around data and information management and the dealing with “unexpected” issues.

Consider the simplest experiment (Two conditions three replicates) 6-12 fastq starting files

6-12 quality control files

6-12 fastq files post trimming of adaptors 6 bam file, and 6 bam index files

6 gene count files

36-48 files minimum (big files)

Computational Considerations The Challenges

There is no single **best method** for RNA-Seq data analysis - it depends on your definition of best, and even then it varies over time and with the particular goals and specifics of a given experiment

It's for this reason that you should learn enough about the process to make “sensible choices” and to know when the results are reasonable and correct.

Treating an RNA-Seq (or any NGS) analysis as a black box is a “recipe for disaster” (or at least bad science). That's not to say that you need to know the particulars of every algorithm involved in a workflow, but you should know the steps involved and what assumptions and/or limitations are build into the whole workflow

Computational Prerequisites

These are considered appropriate if you are planning on doing all the data analysis yourself.

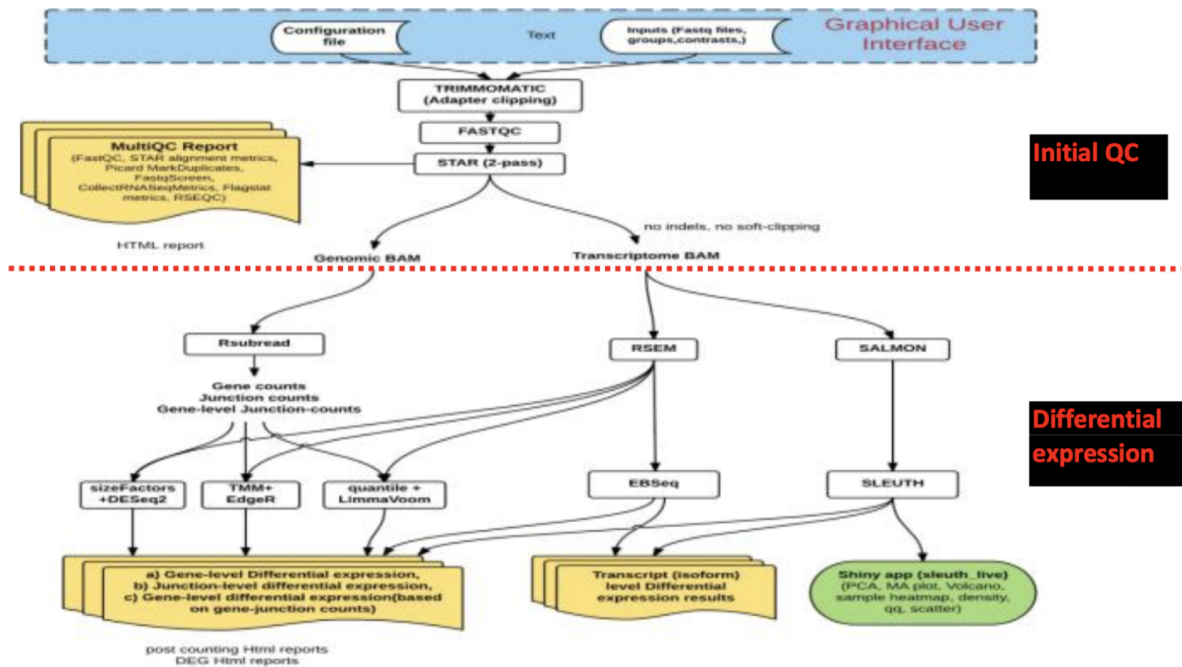
- High performance Linux computer (multi core, high memory, and plenty of storage)
- Familiarity with the “command line” and at least one programming/scripting language.
- Basic knowledge of how to install software
- Basic knowledge of R and/or statistical programming Basic knowledge of Statistics and model building

Data Analysis

Here are a pair of examples of RNASEQ complete workflows

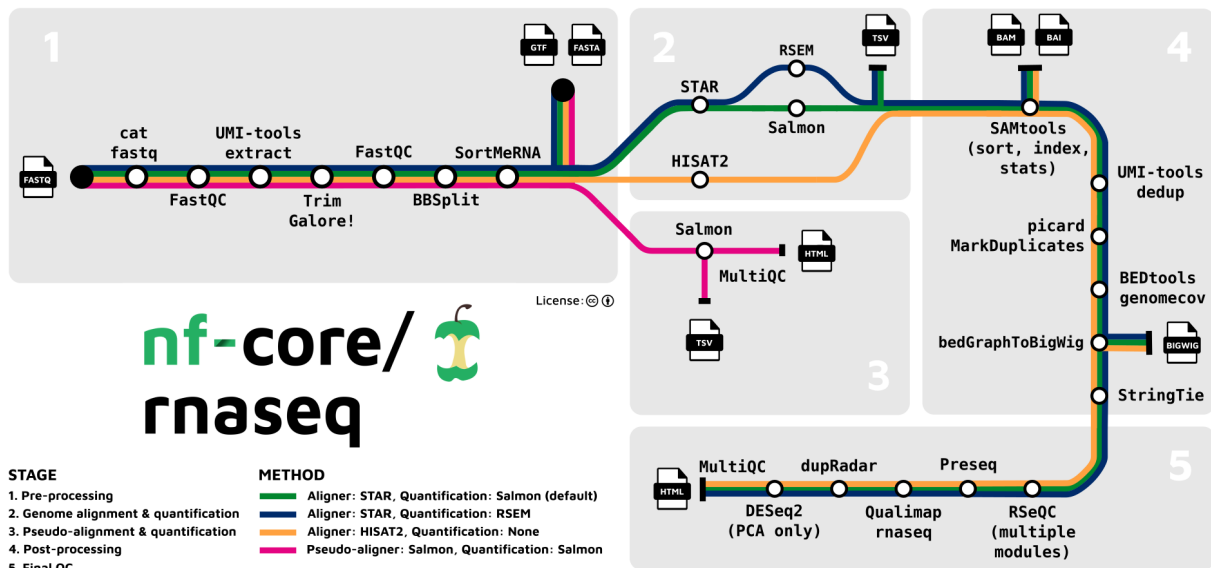
RNASEQ Pipeline from NCI CCBR

<https://github.com/CCBR/Pipeliner/blob/master/RNASeqDocumentation.pdf>



RNASEQ Nextflow Pipeline from nf-core

<https://nf-co.re/rnaseq>



Data Quality Assessment (QC)

Basic RNASEQ Quality Control (QC) examines the technical characteristics of the data produced by the sequencer. (It tells us nothing about whether the experiment worked. It answers the questions:

- Is the data of sufficiently high quality to be analyzed?
- Are there technical artifacts?
- Are there poor quality samples?
- It evaluate the following features
 - Overall sequencing quality scores and distributions
 - GC content distribution
 - Presence of adapter or contamination
- Sequence duplication levels

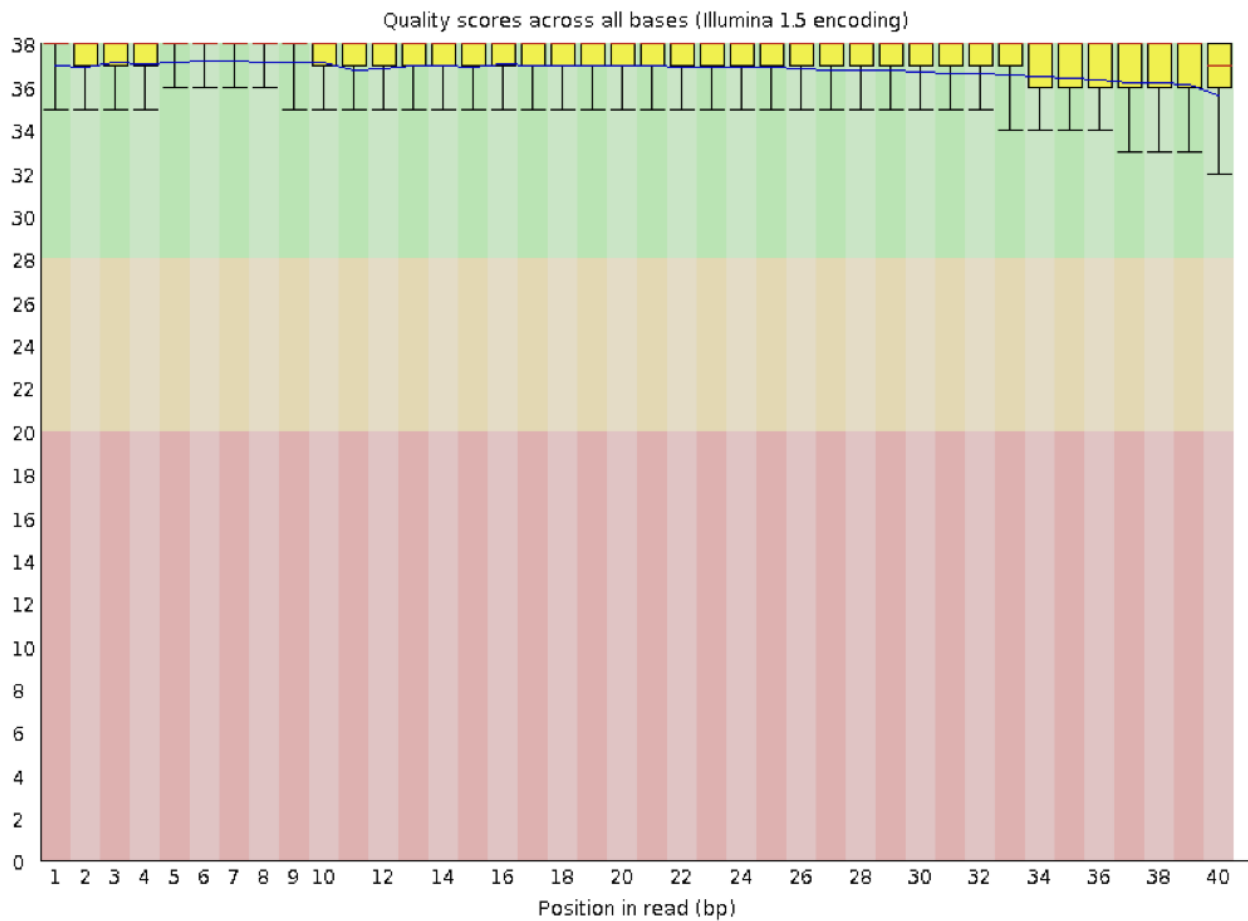
Data should be filtered, trimmed, or rejected as appropriate

Sequencing cores generally provide some/all of this analysis

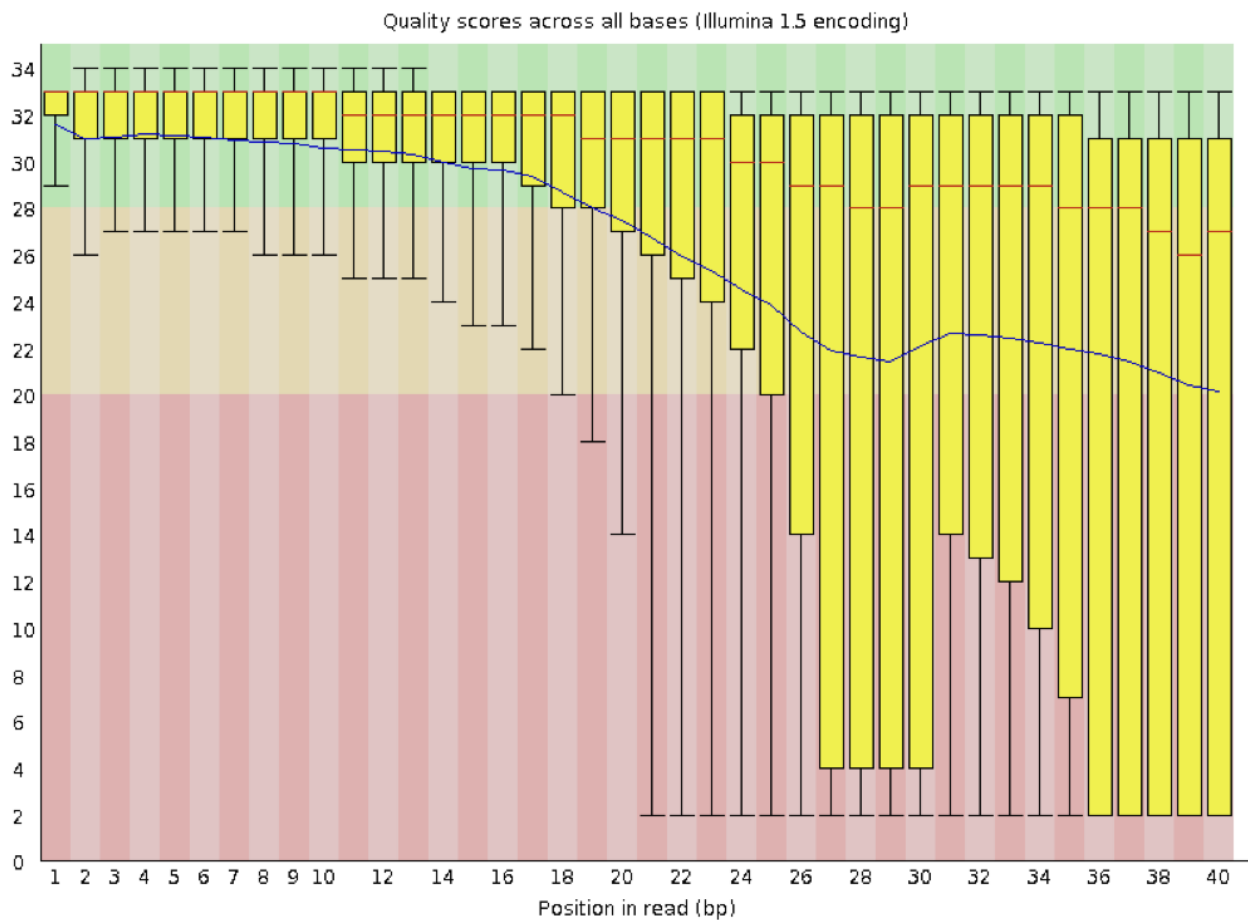
Examples of some of the quantities measured in basic QC

By the program FastQC

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html
(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html)



GOOD

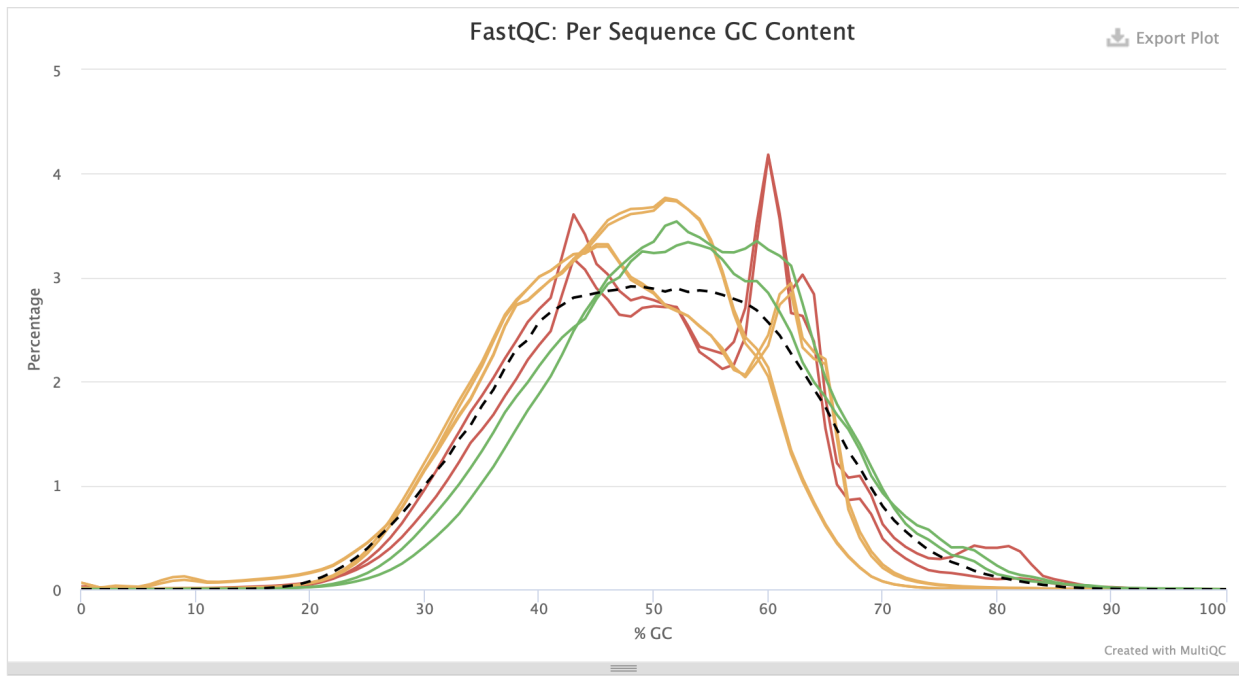


BAD

By the program MultiQC

https://multiqc.info/examples/rna-seq/multiqc_report.html

(https://multiqc.info/examples/rna-seq/multiqc_report.html)



Raw Sequence Cleanup

Trim and/or filter sequence to remove sequencing primers/adaptor and poor quality reads.
Example programs:

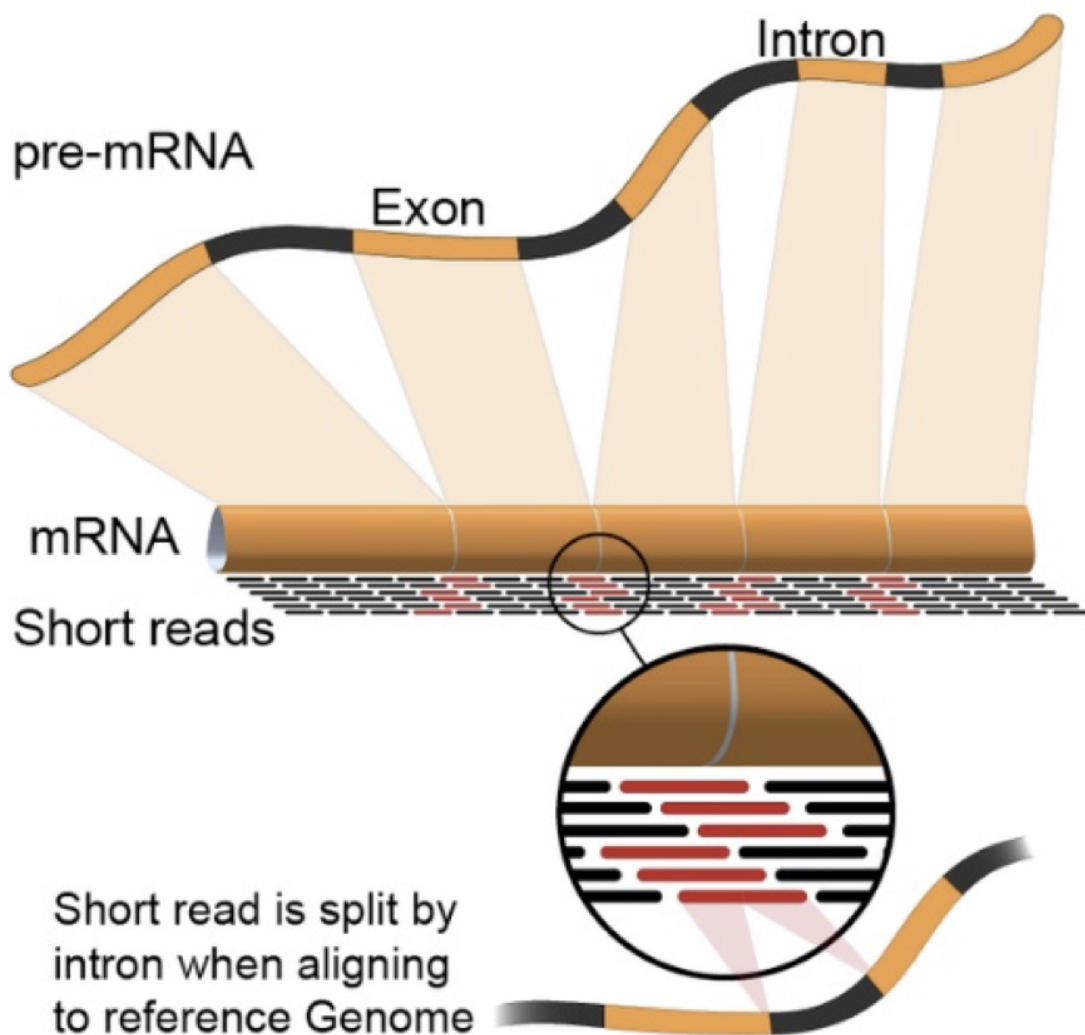
- **FASTX-Toolkit** is a collection of command line tools for Short-Reads FASTA/ FASTQ files preprocessing.
- **SeqKit** is an ultrafast comprehensive toolkit for FASTA/Q processing.
- **Trimmomatic** is a fast, multithreaded command line tool that can be used to trim and crop Illumina (FASTQ) data as well as to remove adapters.
- **TrimGalore** is a wrapper tool around Cutadapt and FastQC to consistently apply quality and adapter trimming to FastQ files, with some extra functionality for MspI-digested RRBS-type (Reduced Representation Bisulfite-Seq) libraries.
- **Cutadapt** finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

Alignment

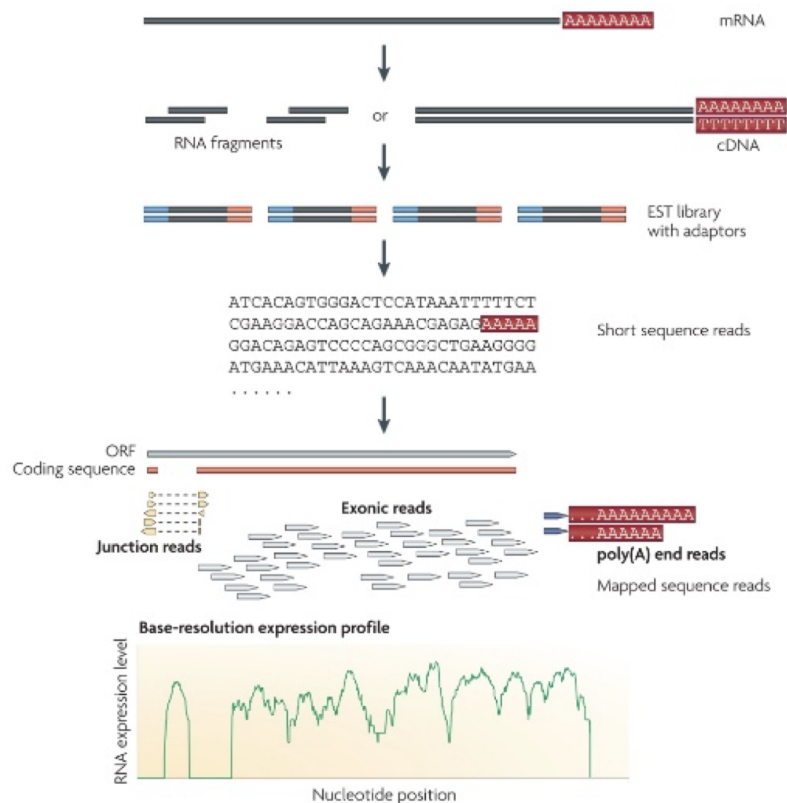
RNASeq Mapping Challenges

The majority of mRNA derived from eukaryotes is the result of splicing together discontinuous exons, and this creates specific challenges for the alignment of RNASeq data.

RNA-seq Alignment



RNA-seq protocol schematic



Mapping Challenges

- Reads not perfect
- Duplicate molecules (PCR artifacts skew quantitation)
- Multimapped reads - Some regions of the genome are thus classified as unmappable
- Aligners try **very** hard to align **all** reads, therefore fewest artifacts occur when all possible genomic locations are provided (genome over transcriptome)

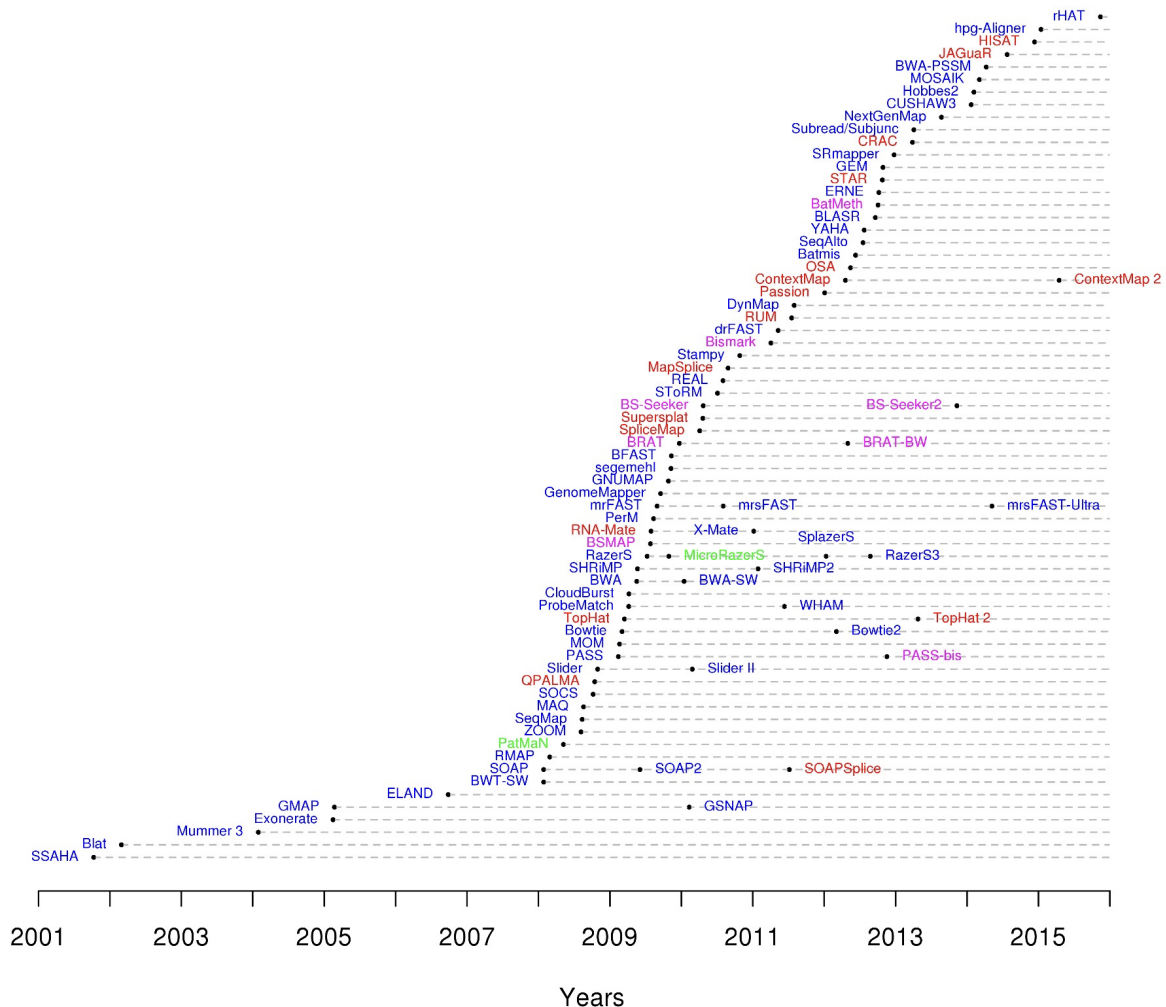
RNASeq Mapping Solutions

There are a number of specific solutions that have been devised to address the issues created by attempting to map mRNA to DNA genomes. Each of these has its advantages and disadvantages.

- Align against the transcriptome
 - Many/All transcriptomes are incomplete
 - Can only measure known genes
 - Won't detect non-coding RNAs

- Can't look at splicing variants
- Can't detect fusion genes or structure variants
- De novo assembly of RNASeq reads
 - Largely used for uncharacterized genomes
- Align against the genome using a splice-aware aligner
 - Most versatile solution
- Pseudo-Aligner - quasi mappers (Salmon and Kalisto)
 - New class of programs - blazingly fast
 - Map to transcriptome (not genome) and does quantitation
 - Surprisingly accurate except for very low abundance signals
 - With bootstrapping can give confidence values

The complexity of the problem of accurately mapping millions of reads against large genomes can be appreciated by looking at a time line of the development of different mapping programs.



Common Aligners

Most alignment algorithms rely on the construction of auxiliary data structures, called indices, which are made for the sequence reads, the reference genome sequence, or both. Mapping algorithms can largely be grouped into two categories based on properties of their indices: algorithms based on hash tables, and algorithms based on the Burrows-Wheeler transform

- Bowtie2 BWA/BWA-mem STAR
- HISAT
- HISAT2
- TopHat
- TopHat2

Tools for mapping high-throughput sequencing data Nuno A. Fonseca Johan Rung Alvis Brazma John C. Marioni Author Notes Bioinformatics, Volume 28, Issue 24, 1 December 2012, Pages 3169–3177, <https://doi.org/10.1093/bioinformatics/bts605>

To Align or not to Align

Aligners typically align against the entire genome and provide a output where the results can be visibly inspected (bam file via IGV). They must be used for detecting novel genes/transcripts. Quantitation of aligned reads to specific genes is typically done by separate program

PseudoAligners assign reads to the most appropriate transcript... can't find novel genes/transcripts or other anomalies. Generally much faster than aligner and are likely more accurate (Recent improvements in salmon have increased its accuracy, at the expense of being somewhat slower than the original)

Typical Questions about alignment

- What is the best aligner to use?
- What Genome version should I use?
- What Genome annotation should I use?

Answers

- STAR - (Salmon or Kallisto) - subjective
- Depends ! most recent or best annotated
- GeneCode with caveats - know what is being annotated and what is not and how it effects your results

Questions not asked

- What parameters should I use?

Answers

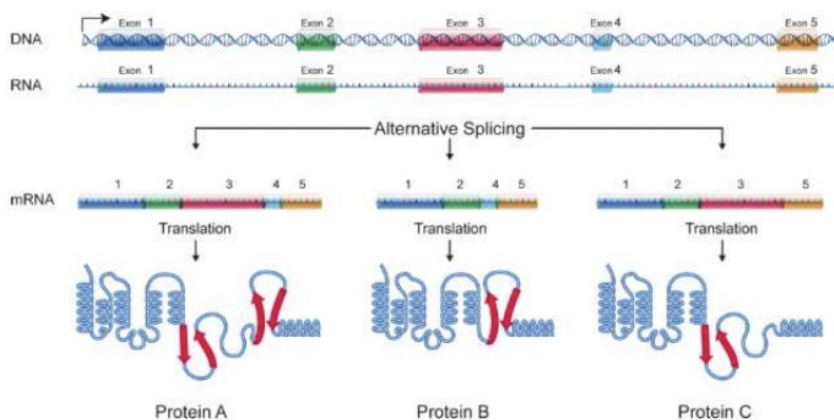
Most programs have lots of optional parameters that can tweak the results, but most are set to defaults that should work in most common situations. *(Don't touch what you don't understand - especially if it gets you, your favorite answer)*

Special Consideration for Alternate Splicing Events

To add to the mRNA mapping problem is the existence of alternate splicing events. Attempting to identify alternate splicing in RNA-SEQ data is not something for the novice to attempt! get professional help

RNA-Seq: Special Mapping Concerns

Alternate Splicing



genome.gov

Post Alignment QC Programs

RSeQC package provides a number of useful modules that can comprehensively evaluate high throughput sequence data especially RNA-seq data. "Basic modules" quickly inspect

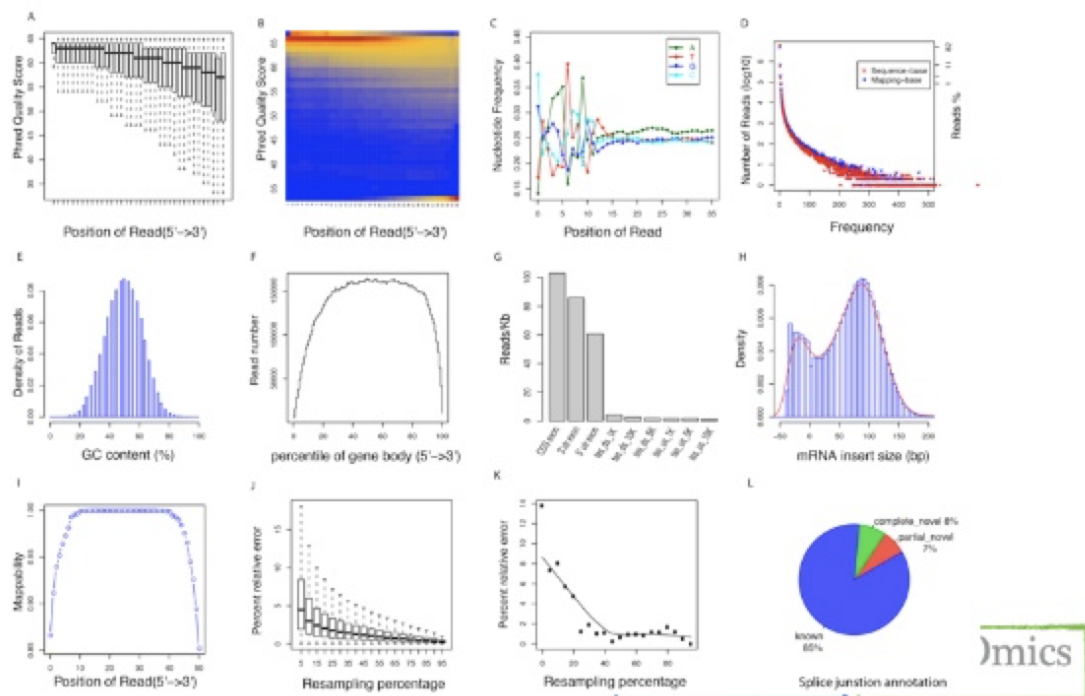
sequence quality, nucleotide composition bias, PCR bias and GC bias, while “RNA-seq specific modules” investigate sequencing saturation status of both splicing junction detection and expression estimation, mapped reads clipping profile, mapped reads distribution, coverage uniformity over gene body, reproducibility, strand specificity and splice junction annotation.

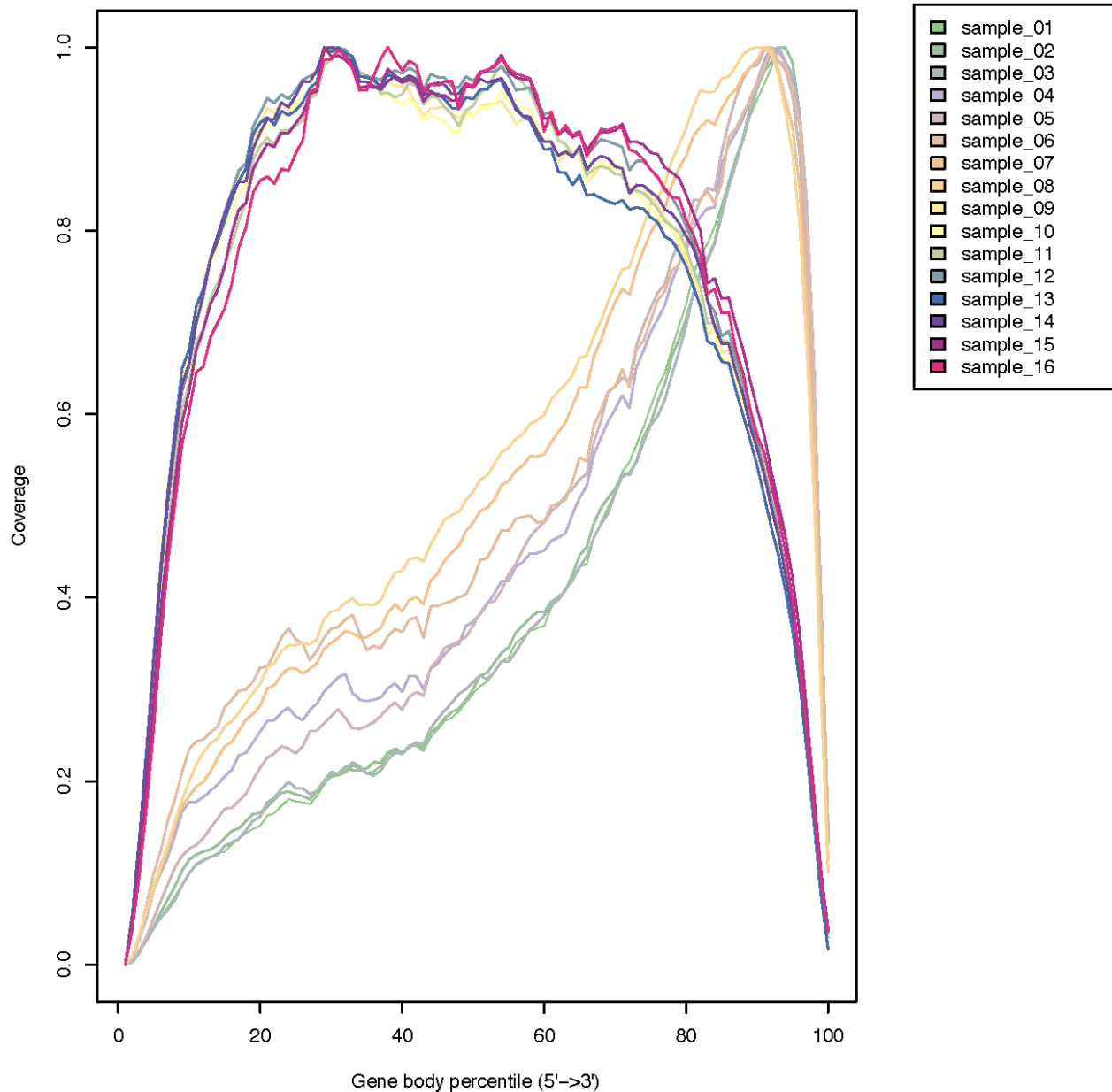
MultiQC is a modular tool to aggregate results from bioinformatics analyses across many samples into a single report.

Picard Tools - RNAseqMetrics is a module that produces metrics about the alignment of RNA-seq reads within a SAM file to genes

RSeQC example of plot types

RSEQC





Post Alignment Cleanup Programs

Picard is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/ BAM/CRAM and VCF. (mark pcr duplicates)

Samtools provide various utilities for manipulating alignments in the SAM/BAM format, including sorting, merging, indexing and generating alignments in a per-position format.

BamTools is a command-line toolkit for reading, writing, and manipulating BAM (genome alignment) files.

Quantitation

Counting as a measure of Expression

Most RNASEQ techniques deal with count data.

- The reads are mapped to a reference and the number of reads mapped to each gene/transcript is counted
- Read counts are roughly proportional to gene-length and abundance
- The more reads the better
 - Artifacts occur because of:
 - Sequencing Bias
 - Positional bias along the length of the gene Gene annotations (overlapping genes) Alternate splicing
 - Non-unique genes
 - Mapping errors

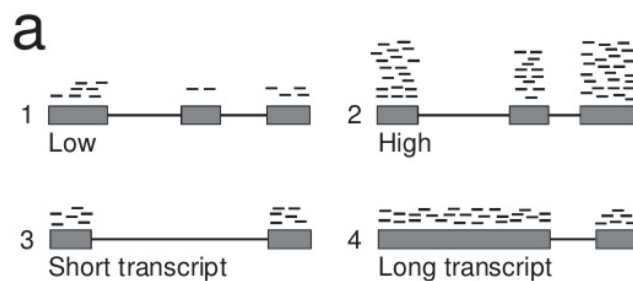
The typical steps in quantitation of mapped reads is as follows:

- Count mapped reads
- Count each read once (deduplicate)
- Discard reads that:
 - have poor quality alignment scores
 - are not uniquely mapped
 - overlap several genes
 - Have paired reads do not map together
- Remember to document what was done

Count Normalization

Count Normalization

- Number of reads aligned to a gene gives a measure of its level of expression
- Normalization of the count data
 - Sequencing depth
 - Length bias



Nature Methods 8, 469–477 (2011), Doi:10.1038/nmeth.1613

There are three metrics commonly used to attempt to normalize for sequencing depth and gene length.

- **RPKM = Reads Per Kilobase Million**

$$\begin{aligned} \text{Total Reads}/1,000,000 &= \text{PM} \\ \text{Gene read-count}/\text{PM} &= \text{RPKM} \\ \text{RPM}/\text{gene-length (kb)} &= \text{RPKM} \end{aligned}$$

- **FPKM = Fragments Per Kilobase Million**

FPKM is very similar to RPKM. RPKM was made for single-end RNA-seq, where every read corresponded to a single fragment that was sequenced. FPKM was made for paired-end RNA-seq.

- **TPM = Transcripts Per Million (Sum of all TPM in samples is the same)**

TPM is very similar to RPKM and FPKM. The only difference is the order of operations

$$\begin{aligned} \text{Gene read-count}/\text{gene-length (kb)} &= \text{RPK} \\ (\text{Sum all RPKs})/1,000,000 &= \text{PM} \\ \text{Gene RPK}/\text{PM} &= \text{TPM} \end{aligned}$$

<https://www.rna-seqblog.com/rpkm-fpkm-and-tpm-clearly-explained/>

Counting as a measure of Expression

An example of a counts matrix for RNASEQ data.

Name	Length	EffectiveLength	TPM	NumReads
ENSG00000121410.12_4	509.732	325.991	3.22494	322.674
ENSG00000268895.6_6	1823.71	1633.86	0.9255	464.119
ENSG00000148584.15_4	5354.1	5164.27	0	0
ENSG00000175899.14_4	4544.77	4354.95	0.039651	53
A2M-AS1	2592.39	2402.54	0.008136	5.999
A2ML1	1749	1561.55	0	0
SLC7A2	452	269.66	0	0
ENSG00000001461.12_NIPAL3	386	208.766	0	0
ENSG00000001497.12_LAS1	1715	1526.05	0	0
ENSG00000001617.7_SEMA3F	1023	833.15	0	0
ENSG00000003096.9_KLHL13	1457.48	1269.51	3.23046	1258.74



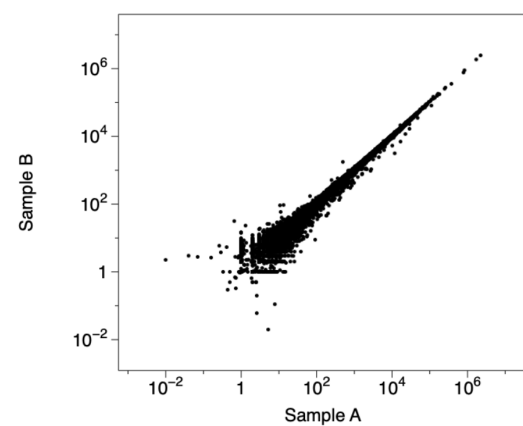
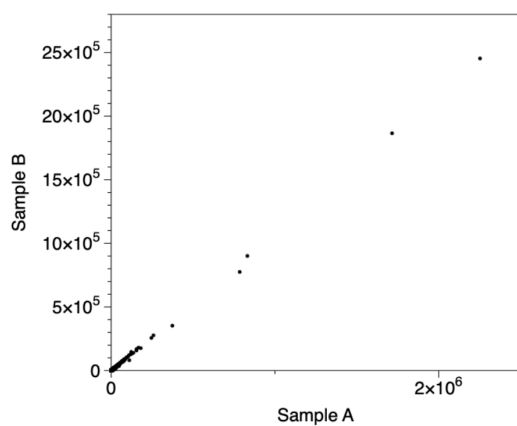
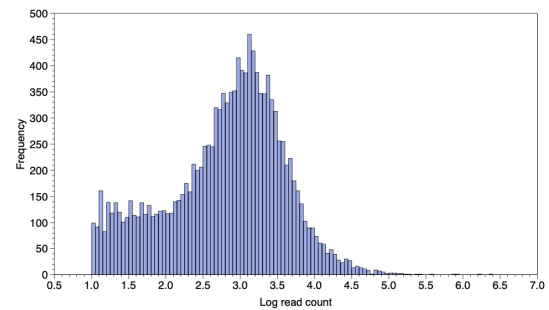
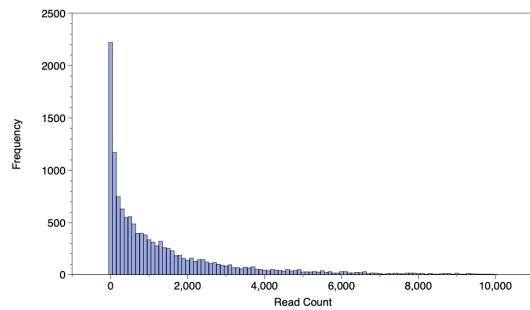
Different ways of annotating the genes



Not always integers -
may not be acceptable to some programs

Log Transformed Data

Because of its vast dynamic ranges RNASEQ data is typically log transformed in order to: provide better visualizations and to present analysis software with a more "normal distribution".



Common counting Programs

- Subread (featureCount)
- STAR (quantmode)
- HTseq (counts)
- RSEM (RNA-Seq by Expectation Maximization) Salmon, Kallisto - pseudoaligners
- Salmon (pseudo aligner and counter)

Differential Expression

Differential expression involves the comparison of normalized expression counts of different samples and the application of statistical measures to identify quantitative changes in gene expression between the different samples.

Normalization and Statistical Significance

Two Statistical Components:(Remember all statistical methods rely on various assumptions regarding the characteristics of the data...if they are not true all bets are off).

- **Normalization of counts** - the process of ensuring that values are expressed on the same scale (e.g. RPKM, FPKM, TPM, TMM). Corrects for variable gene length, read depth.
- **Differential Expression** - analysis of the difference in expression of genes under two conditions (pair wise comparison) - expressed as fold difference. A statistical test determines whether the observed difference is statistically significant (i.e. the likelihood of the observation is greater than that expected from random biological variability). Such analyses are typically based on a negative binomial distribution - expressed as P or corrected P value.

Replicates

Biological replicates are essential to derive a meaningful result. Don't mistake the high precision of the technique for the need for biological replicates.

If technical or biological variability exceeds that of the experimental perturbation you will get zero DEs.

Remember not all DE may be directly due to the experimental perturbation, but could be do to cascading effects of other genes.

Multiple Testing Correction

Differential Expression data **must** be corrected for multiple testing. Two common methods are the "Bonferroni procedure" and "Benjamini-Hochberg procedure". These forms of statistical correction will result in a "corrected pvalue", or a qvalue or FDR or padj (adjusted p value).

Note pvalues refer to the each gene, whereas an FDR (or qvalue) is a statement about a list. So using FDR cuff of 0.05 indicates that you can expect 5% false positives in the list of genes with an FDR of 0.05 or less.

Count Matrix

Data_matrix

Data_matrix	p53_rock_1	p53_rock_2	p53_rock_3	p53_rock_4	p53_IR_1	p53_IR_2	p53_IR_3	p53_IR_4	null_rock_1	null_rock_2	null_IR_1	null_IR_2
C330021F23RIK	83	67	52	117	52	43	38	38	96	71	54	71
CPS1	0	0	0	0	4	8	0	0	0	0	0	1
FAM171B	11	11	6	11	13	10	4	8	14	6	10	10
OLFR910	0	1	0	0	0	1	0	0	0	0	0	0
DYNLL2	462	413	294	529	330	206	317	293	312	275	409	663
NPEPL1	2361	1794	1563	1612	2296	1565	2969	3758	1904	1657	3200	3516
TRAJ2	4	6	6	4	9	13	5	4	7	4	5	2
SLC2A4	9	11	3	3	15	10	13	21	2	7	0	0
ZFP655	2874	2474	2006	2517	1640	1276	1881	1948	2666	2412	3157	3315
SLC8A1	1074	839	941	921	657	340	469	320	852	770	337	803
CY5R4	7431	6425	4866	6215	4502	3800	4170	4656	6602	5619	6059	6843
GM31123	0	0	0	0	0	0	0	0	0	0	0	0
CTDNEP1	1210	1105	869	1323	833	493	951	1094	1063	999	2069	2039
ETS1	44445	38606	27356	39522	10423	7905	8481	10543	42254	41214	20881	27334

Contrast File

Study_design

Study_Design	p53_rock_1	p53_rock_2	p53_rock_3	p53_rock_4	p53_IR_1	p53_IR_2	p53_IR_3	p53_IR_4	null_rock_1	null_rock_2	null_IR_1	null_IR_2
p53	wt	wt	wt	wt	wt	wt	wt	wt	null	null	null	null
Treatment	rock	rock	rock	rock	IR	IR	IR	IR	rock	rock	IR	IR

Study_design-1

Study_Design	p53	Treatment
p53_rock_1	wt	rock
p53_rock_2	wt	rock
p53_rock_3	wt	rock
p53_rock_4	wt	rock
p53_IR_1	wt	IR
p53_IR_2	wt	IR
p53_IR_3	wt	IR
p53_IR_4	wt	IR
null_rock_1	null	rock
null_rock_2	null	rock
null_IR_1	null	IR
null_IR_2	null	IR

Different programs require this file to be organized in different ways

Differential Expression Programs

Method	Normalization	Needs replicas	Input	Statistics for DE	Availability
edgeR	Library size	Yes	Raw counts	Empirical Bayesian estimation based on Negative binomial distribution	R/Bioconductor
DESeq	Library size	No	Raw counts	Negative binomial distribution	R/Bioconductor
baySeq	Library size	Yes	Raw counts	Empirical Bayesian estimation based on Negative binomial distribution	R/Bioconductor
LIMMA	Library size	Yes	Raw counts	Empirical Bayesian estimation	R/Bioconductor
CuffDiff	RPKM	No	RPKM	Log ratio	Standalone

Differential Expression Output

Final output is typically a rank order list of differentially expressed (DE) genes with expression values and associated p-values. Here are examples from the programs EdgeR and DESeq2

EDGER

Gene	LogFC	AveExpr	P-Value	FDR
*CA14	-6.72	4.31	1.406716E-10	0.000001
*MCF2L	-10.75	3.25	2.854327E-10	0.000001
*COL5A2	-6.12	4.28	3.678663E-10	0.000001
*TYRP1	-9.31	9.85	4.190114E-10	0.000001
*BCAN	-8.39	5.33	6.384088E-10	0.000001
*CSAG1	10.81	-0.56	7.095577E-10	0.000000

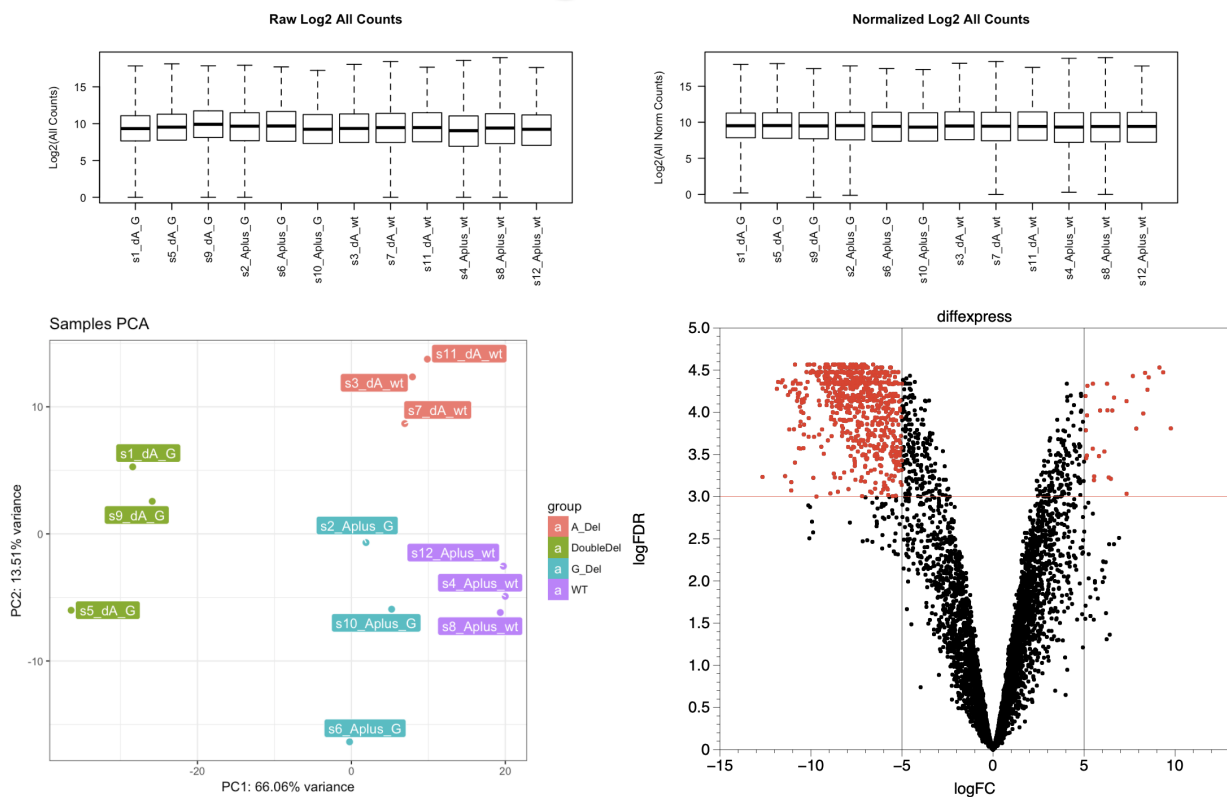
DESEQ2

Row-names	Symbol	log2FoldChange	padj	p53_mock_1	p53_mock_2	p53_mock_3	p53_mock_4	p53_IR_1	p53_IR_2	p53_IR_3	p53_IR_4
ENSMUSG000000000001	Gnai3;Gnai3	-0.4763	0.1737	11.584	11.565	11.609	11.621	11.399	11.338	11.997	11.927
ENSMUSG0000000000028	Cdc45;Cdc45	-0.4610	0.4125	8.024	7.575	7.668	7.295	7.736	7.675	7.906	7.873
ENSMUSG0000000000037	Scml2;Scml2	1.3780	0.1889	3.196	3.554	3.563	3.296	4.592	5.249	4.765	5.262
ENSMUSG0000000000056	Narf;Narf	-0.1732	0.8053	10.644	10.609	10.634	10.754	9.640	9.516	10.036	10.127
ENSMUSG0000000000058	Cav2;Cav2	-0.3945	0.6751	4.377	4.546	5.292	5.120	4.122	3.531	4.835	4.269
ENSMUSG0000000000088	Cox5a;Cox5a	-0.5847	0.2738	9.887	9.754	9.964	9.851	9.692	9.501	10.530	10.467
ENSMUSG0000000000120	Ngfr;Ngfr	0.7409	0.2168	7.519	7.746	7.625	8.458	8.053	8.149	7.435	7.406
ENSMUSG0000000000127	Fer;Fer	0.1804	0.7480	7.324	7.381	7.368	7.008	7.389	6.650	6.534	6.235
ENSMUSG0000000000142	Axin2;Axin2	0.0927	0.9124	5.542	5.920	5.396	5.510	6.008	6.281	5.351	5.484

Visualization

Here are a number of visual elements that are typically produce from RNASEQ data.

Normalization plots



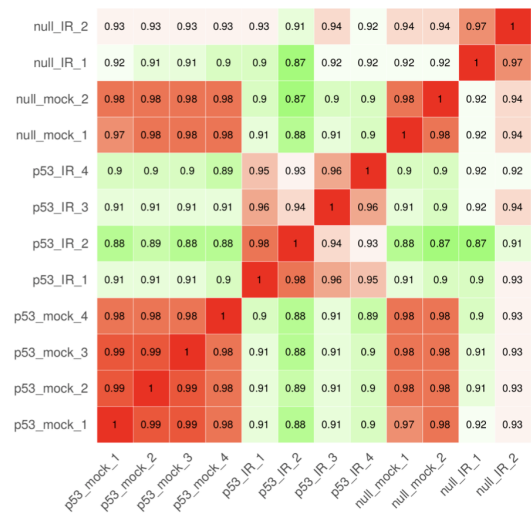
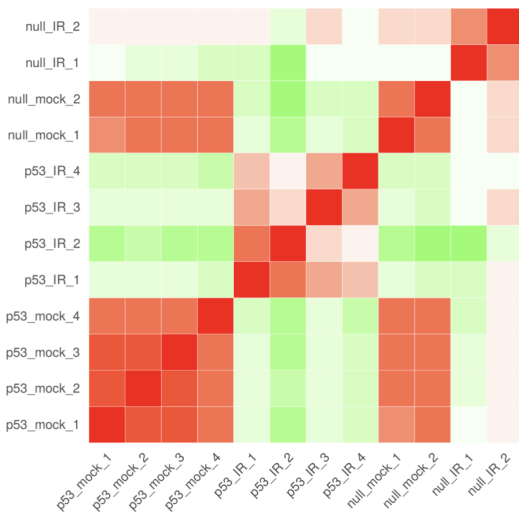
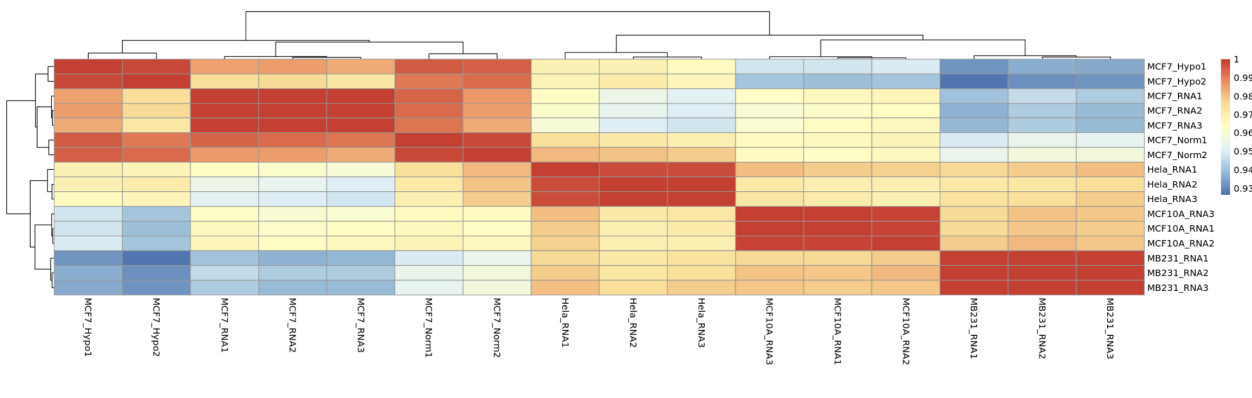
PCA and Volcano plots

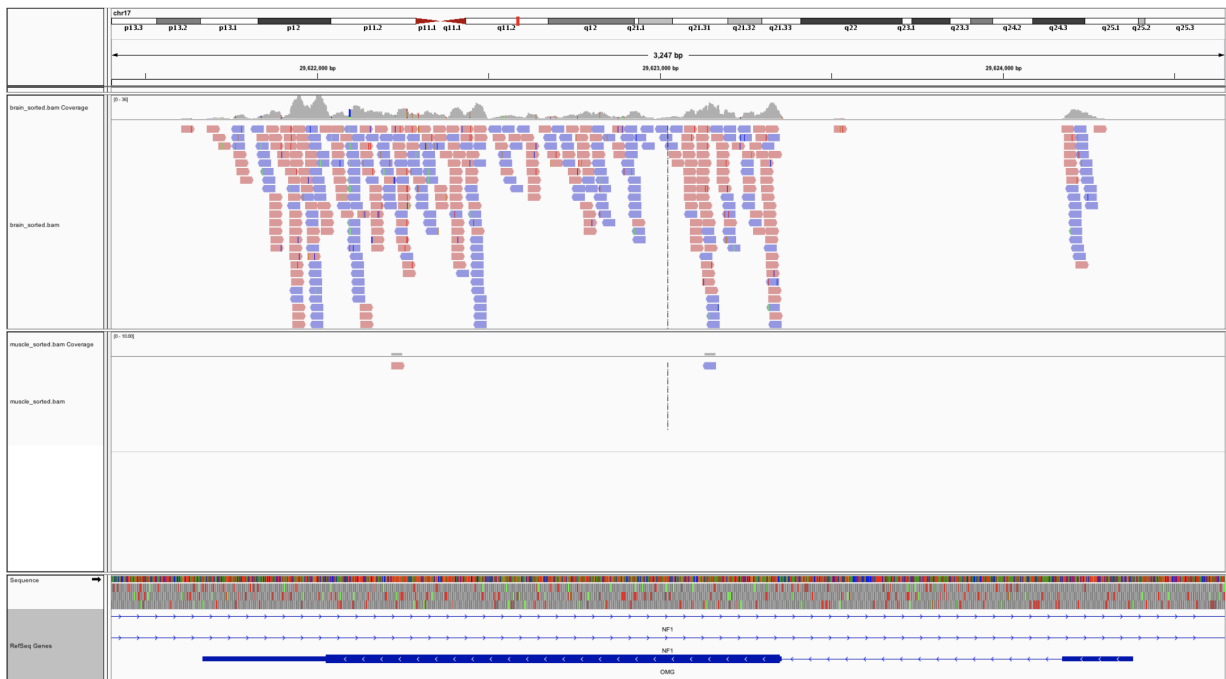
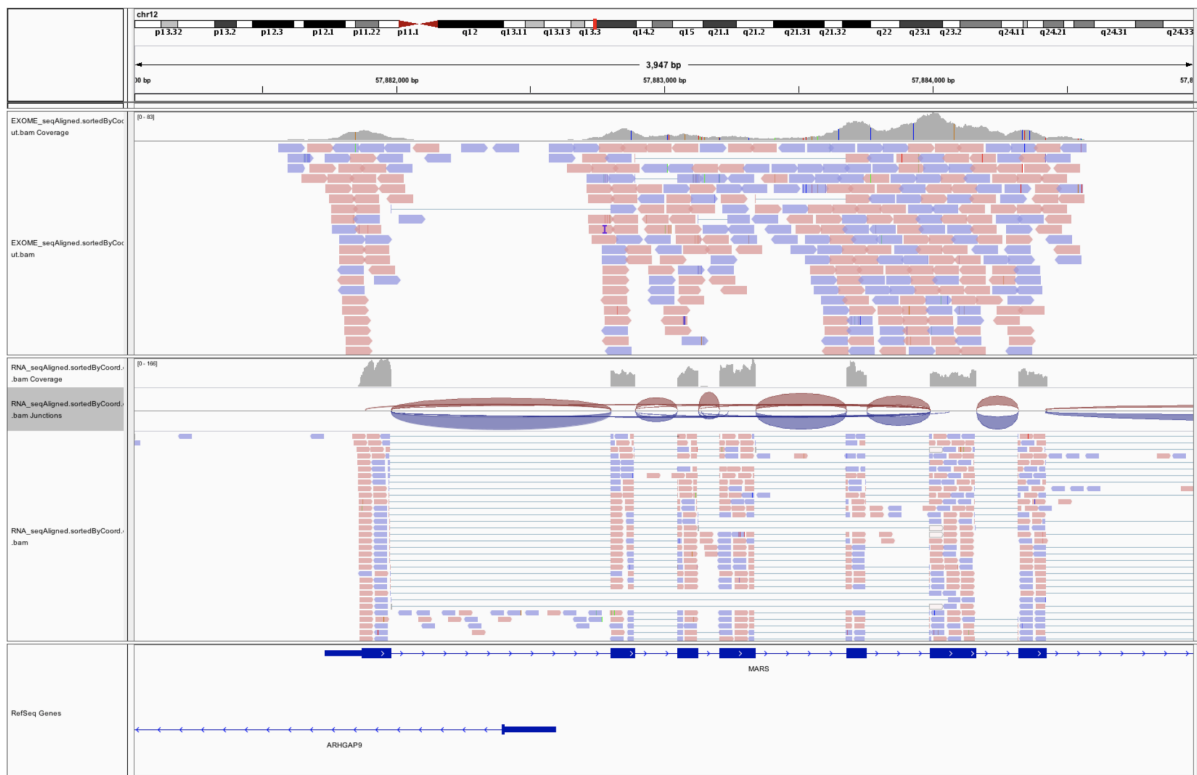
Scatter plot and correlation coefficients



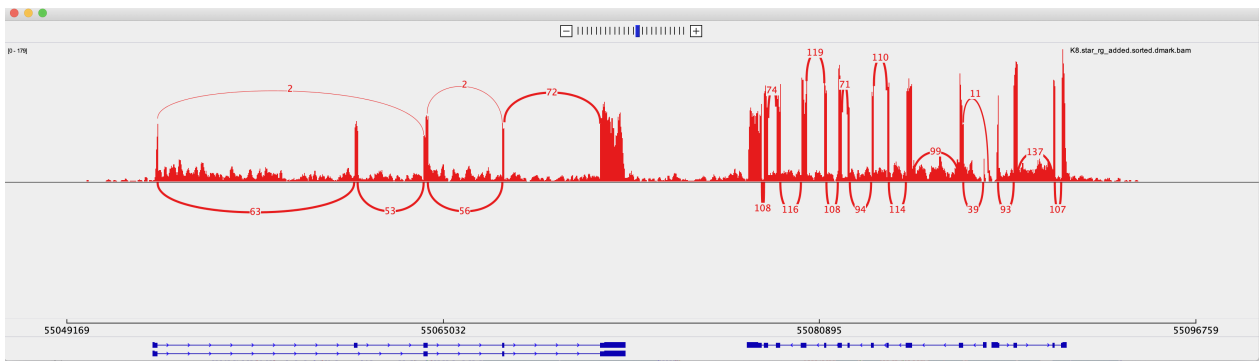
Heat

Maps



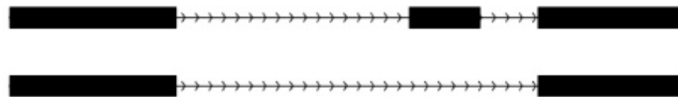
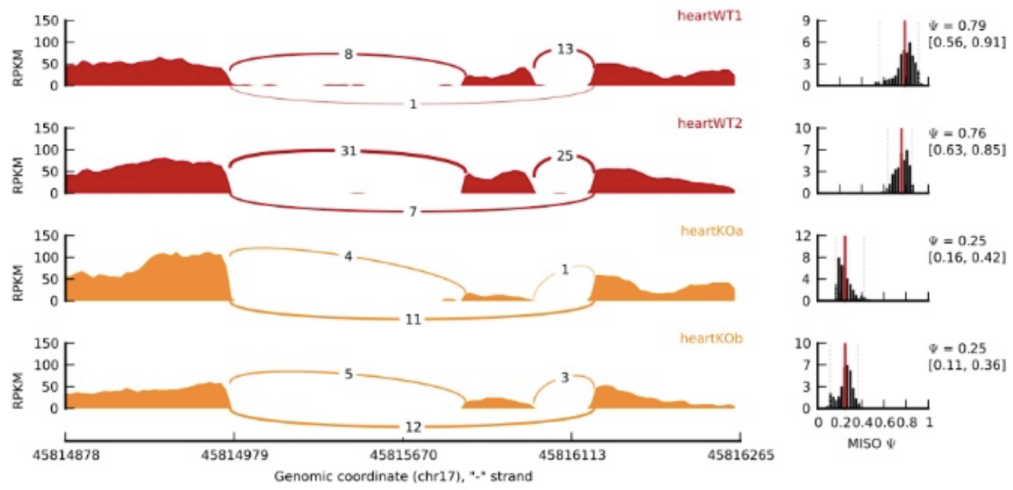






Visualizing Splicing

chr17:45816186:45816265:-@chr17:45815912:45815950:-@chr17:45814875:45814965:-



Visualization and Next step tools

Visualization

1. Integrated Genome Viewer (<https://www.broadinstitute.org/igv/>)

Further Annotation of Genes

1. DAVID (<http://david.abcc.ncifcrf.gov/tools.jsp>)
2. ConsensusPathdb (<http://cpdb.molgen.mpg.de/>)
3. NetGestalt (<http://www.netgestalt.org/>)
4. Molecular Signatures Database (<http://www.netgestalt.org/>)
5. PANTHER (<http://www.pantherdb.org/>)
6. Cognoscente (<http://vanburenlab.medicine.tamhsc.edu/cognoscente.shtml>)
7. Pathway Commons (<http://www.pathwaycommons.org/>)
8. Readctome (<http://www.reactome.org/>)
9. PathVisio (<http://www.pathvisio.org/>)
10. Moksiskaan (<http://csbi.ltdk.helsinki.fi/moksiskaan/>)
11. Weighed Gene Co-Expression Network Analysis (WGCNA)s
12. More tools in R Bioconductor

Resources

Tertiary Analysis - Biological Meaning

- Pathway Analysis
 - IPA (Qiagen - CCR License) Future talk
- Functional Analysis
 - Gene Set Enrichment Analysis (GSEA) <https://www.gsea-msigdb.org/gsea/index.jsp>
 - DAVID <https://david.ncifcrf.gov/>
 - Enrichr <https://maayanlab.cloud/Enrichr/>
- Other Types of Analysis
 - Genomic Location
 - Transcription Factor Enrichment Analysis
 - miRNA Enrichment Analysis

Software Solutions

CCR staff have access to a number of resources

- Biowulf (Helix) - CIT maintained large cluster with a huge software library (Unix command line)
- CCBP Pipeliner (Biowulf)
- Partek Flow (Local Web Service)
- DNAnexus (Cloud Solution)
- CLCBio Genomic Workbench (Small genomes)

Public sources of RNA-Seq data

- Gene Expression Omnibus (GEO) (<http://www.ncbi.nlm.nih.gov/geo/>)
- Both microarray and sequencing data
- Sequence Read Archive (SRA) (<http://www.ncbi.nlm.nih.gov/sra>)
- All sequencing data (not necessarily RNA-Seq)
- ArrayExpress (<https://www.ebi.ac.uk/arrayexpress/>)
- European version of GEO
- Homogenized data: MetaSRA, Toil, recount2, ARCHS4

NGS File Formats

- Sequence
 - FASTA, FastQ
- Alignment
 - SAM, BAM, CRAM
- Annotation
 - GTF, GFF, BED (BIGBED)
- Graphing
 - WIG (BIGWIG), BEDGRAPH

Utility Programs

- SeqKit
- FastQC, RSeQC, MultiQC
- Cutadapt, Fastp, Trimmomatic, TrimGalore STAR, Bowtie, Salmon
- Samtools, Picard, bedtools, bamtools
- R, Python
- IGV

Web-Based Tools

- BioJupies - Many analysis functions - generates Jupyter Notebook of results (<https://amp.pharm.mssm.edu/biojupies/>)
- IDEP92 - an integrated web application for differential expression and pathway analysis of RNA-Seq data

(<http://bioinformatics.sdstate.edu/idep92/>) Both allow analysis of many public datasets

File Transfer Methods

- Globus (<https://hpc.nih.gov/storage/globus.html>)
- HPCDME
- BOX
- OneDrive
- (s)FTP
- Network Drives
- Flash Drives

Further Reading

- RNA-seqlopedia - <https://rnaseq.uoregon.edu/>
- RNA-Seq by Example - <https://www.biostarhandbook.com/>

Module 2 - RNA sequencing

Lesson 9: Reference genomes and genome annotations used in RNA sequencing

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 8 Review

In Lesson 8, we learned about the basics of RNA sequencing, including experimental considerations and basic ideas behind data analysis. In lessons 9 through 17 we will learn how to analyze RNA sequencing data. We will start with quality assessment, followed by alignment to a reference genome, and finally identify differentially expressed genes.

Learning Objectives

In this lesson, we will continue to learn about RNA sequencing analysis using the [Human Brain Reference \(HBR\)](https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/) and [Universal Human Reference \(UHR\)](https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/) datasets (https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/). In particular, we will

- Get to know the HBR and UHR dataset
- Locate the data and analysis tools
- Introduce the reference genomes
- Provide a brief introduction to the Integrative Genome Viewer([IGV \(https://software.broadinstitute.org/software/igv/\)](https://software.broadinstitute.org/software/igv/)) - to focus on visualizing the reference genome and genomic features

About the Human Brain Reference and Universal Human Reference data

The Human Brain Reference (HBR) RNA sequencing data are derived from RNA extracted from

- 23 human brains
- brains are from both males and females, age ranging from 60 to 80 years

The Universal Human Reference data used RNA from 10 cancer cell lines.

These two experiments used the External RNA Control Consortium (ERCC) spike-in RNAs as controls. These internal standards provide a known quantity of RNA to evaluate the quality of RNA sequencing experiment. They can provide information on dynamic range, limits of detection, and reliability of differential expression results. To learn more about ERCC spike-ins refer to the following:

- <https://www.thermofisher.com/order/catalog/product/4456740> (<https://www.thermofisher.com/order/catalog/product/4456740>)
- <https://www.nist.gov/programs-projects/external-rna-controls-consortium> (<https://www.nist.gov/programs-projects/external-rna-controls-consortium>)
- ERCC seminar (<https://youtu.be/YVlrzKMJ2uc>)
- ERCC publication (<https://www.nature.com/articles/ncomms6125>)
- ERCC Bioconductor package (<https://www.bioconductor.org/packages/release/bioc/html/ercdashboard.html>)

Where is my data?

Here, let's download the HBR and UHR dataset to get acquainted with it.

First, we will use `pwd` to make sure we are in the home directory.

```
pwd
```

If we are in the home directory, we will see the following output displayed in the terminal where "username" is your username, or student id that you used to sign into the terminal.

```
/home/username
```

If not in the home directory use the command below to get back.

```
cd
```

Then create a folder called `biostar_class` and change into this folder.

```
mkdir biostar_class
```

```
cd biostar_class
```

Let's keep the analysis results of the HBR and UHR dataset to a folder called `hbr_uhr`, so we need to create this and then change into it.

```
mkdir hbr_uhr
```

```
cd hbr_uhr
```

Now it's time to download the HBR and UHR dataset. What are the two commands that we can use to download data from the web?

{{Sdet}}

Solution{{Esum}}

We can use `curl` OR `wget` to download data from the web.

{{Edet}}

Here, we will use `wget` to download the HBR and UHR dataset (remember, we should now be in the `~/biostar_class/hbr_uhr` directory).

Using the `wget` command we just need to enter the command, which is `wget`, and then provide the URL to the file.

```
wget http://data.biostarhandbook.com/rnaseq/projects/griffith/griffith
```

If `wget` does not work, try `curl`. If using `curl`, we need to make sure to specify an output file name using the `-o` option.

```
curl http://data.biostarhandbook.com/rnaseq/projects/griffith/griffith
```

If we now listed the content of the `~/biostar_class/hbr_uhr` directory (recall that `~` denotes home directory), we will see the file `griffith-data.tar.gz`. Recall that this is an archive of a collection of files that has been zipped (or compressed) to save on storage space.

```
ls
```

```
griffith-data.tar.gz
```

We can use the tar command to unpack the contents of griffith-data.tar.gz. In the tar command below, we include the following flags

- -x means - extract to disk from the tar (tape archive)
- -v means - produce verbose output. When using this flag tar will list each file name as it is read from the tar (tape archive), this allows us to see what is being extracted
- -f (file) means read the tar (tape archive) from or to the specified file

```
tar -xvf griffith-data.tar.gz
```

We do ls -l (where -l denotes list contents of a directory in the long view), we will see two additional folders

```
ls -l
```

```
-rw-rw-r-- 1 username username 131302475 Dec 5 2018 griffith-data.1  
drwxrwxr-x 1 username username      264 Oct 19 17:27 reads  
drwxrwxr-x 1 usernamels  username      60 Oct 19 17:27 refs
```

The sequencing reads for the HBR and UHR dataset reside in the reads directory. Below, we will list the content of the reads directory using ls -l where -l tells ls to list directory contents, but with one item per row. We will talk about these fastq or fq files later.

```
ls -l reads
```

```
HBR_1_R1.fq  
HBR_1_R2.fq  
HBR_2_R1.fq  
HBR_2_R2.fq  
HBR_3_R1.fq  
HBR_3_R2.fq  
UHR_1_R1.fq  
UHR_1_R2.fq  
UHR_2_R1.fq  
UHR_2_R2.fq  
UHR_3_R1.fq  
UHR_3_R2.fq
```

In this lesson, the focus is to get to know the reference genome and annotation files.

```
ls refs
```

In the refs folder, we have two fasta (fa) files. One is the reference genome for human chromosome 22 (see file 22.fa) and the other is the reference genome for ERCC spike-ins (see file ERCC92.fa). We will only be using 22.fa here.

We also have two gtf files that tells us about features that exist in a genome. Again, because we are not working with ERCC, we will only be using the 22.gtf file, which tells us about the features that exist on human chromosome 22.

```
22.fa 22.gtf ERCC92.fa ERCC92.gtf
```

Where are the analysis tools?

Throughout this module, we will be running various tools, including helper R scripts on the Unix command line to analyze the HBR and UHR RNA sequencing dataset. The helper R scripts were developed by the author of the Biostars Handbook.

The tools include those used for

- Sequencing data quality assessment
 - FASTQC
 - MultiQC
- Quality and adapter trimming of sequencing data
 - Trimmomatic
 - BBduk
- Alignment to genome or transcriptome
 - HISAT2
 - Salmon
- Obtain expression counts
 - featureCounts
 - combine_transcripts.r (R helper script used to combine count files from Salmon alignment)
- Differential expression analysis
 - deseq2.r (R helper script)
- Visualize gene expression
 - create_map.r (R helper script)
 - create_pca.r (R helper scripts)

For the non-R based tools, we can run them at the command line. You can see a full list of non-R based tools if you listed the contents of the /miniconda3/bin folder although there is no need to go into this folder to do anything.

```
ls /miniconda3/bin
```

The R helper scripts are a bit different. They are located in the folder `/usr/local/code`.

```
ls -l /usr/local/code
```

```
combine_genes.r
combine_transcripts.r
compare_results.r
create_heatmap.r
create_pca.r
create_tx2gene.r
deseq2.r
edger.r
filter_counts.r
mission-impossible.mk
parse_featurecounts.r
```

1. The path to the R helper scripts, `/usr/local/code`, has been exported as the environmental variable `CODE`.
2. If we `ls $CODE`, we should see the contents of the directory as well.
3. This prevents us from typing a long path when running the R helper scripts.
4. For example, if we wanted to use `deseq2.r`, we can type in the command line `Rscript $CODE/deseq2.r`

(Note we run the R helper scripts by starting off with the `Rscript` command.)

```
ls $CODE
```

The reference genome and annotation files

Now that we have downloaded the HBR and UHR dataset and know where analysis tools are, let's start learning about RNA sequencing, by first learning about our reference genome and annotation files. Let's stay in the `~/biostar_class/hbr_uhr/refs` folder for this, if not then change into this directory.

```
cd ~/biostar_class/hbr_uhr/refs
```



```
>chr22
TAAGATGTCCTATAATTTCTGTTTGGAAATATAAAATCAGCAACTAATATGTATTTTCAA
GCATTATAAATACAGAGTGCTAAGTTACTTCACTGTGAAATGTAGTCATATAAAGAACAT
AATAATTATACTGGATTATTTTTAAATGGGCTGTCTAACATTATATTTAAAGGTTTCATC
AGTAATTCATTATATCAAAATGCTCCAGGCCAGGCGTGGTGGCTTATGCCTGTAATCCCA
GCACCTTTGGGAGGTCGAAGTGGGCGGATCACTTGAGGTCAGGAGTTGGAGACTAGCCTGG
CCAACATGATGAAACCCCGTCTCTAATAATAATAATAAAAAAAAAATTAGCTGGGTGTGGT
GGTGGGCAACTGTAATCTCAGCTAATCAGGAGGCTGAGGCAGAAGAATTGCTTGAACGTG
GAAGACAGAGTTTACAGTGTGCCAAGATCACACCACCCTACTCCAACCTGGGTGACAGAG
CAAGACTCAGTCTCAAGGAAAAAAAAAAGCTCGAAAAATGTTTGCTTATTTTGGTAAAAT
```

Here, we find that the human chromosome 22 reference (22.fa) is a file with a header or definition line with the nucleotide sequence of that entire chromosome. But what is it good for? 22.fa contains the known sequences of human chromosome 22, thus it's a reference that we can compare other sequences to. For high throughput sequencing, we need the known sequences so that we can find out where in the genome each of the sequencing reads came from. The reference genome in a way acts like a template that we can follow to reconstruct the genome of the unknown. In other words, think of the reference genome as a picture of the completed puzzle that helps us assemble the actual puzzle, by allowing us to overlap the pieces to see if they fit the completed version.

A question we might ask about the reference genome is how big is the reference (ie. how many bases)? To answer this, we can use the tool `seqkit` and its `stats` feature.

Why do we need the size of the genome?

Prior to the sequencing experiment, the size of the genome will help us determine the number of reads needed to achieve a certain coverage (https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf (https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf)).

After the experiment, we could use the size of our genome along with other information to decide the computing resources (ie. time and memory) needed for our analysis. **Chromosome 22 is the second smallest in humans** (<https://medlineplus.gov/genetics/chromosome/22/>), so it would be faster to align to this than the entire human genome. For the sake of time in this class, that's why we chose to align to just this chromosome.

```
seqkit stats 22.fa
```

file	format	type	num_seqs	sum_len	min_len	avg_len	50
22.fa	FASTA	DNA	1	50,818,468	50,818,468	50,818,468	50

What is in the 22.gtf file? A gtf file is known as **Genome Transfer File**, which is essentially a tab delimited file (ie. columns in the file are separated by tab). It informs us of where different features such as genes, transcripts, exons, and coding sequences are found in a genome.

A gtf file is needed for couple of reasons.

First, even though we will map the RNA sequencing reads to a genome, we need to know which genomic features (ie. genes) the reads are aligning to generate some metric for expression (counts) and then perform differential expression analysis.

Second, because some of the sequencing reads can map to two exons, if we use a splice-aware-aligner, the information in the gtf file can be used to recognize the exon-exon boundaries (see Figure 1).

Otherwise, those reads that map to two exons will not be mapped and we end up losing information. See <https://useast.ensembl.org/info/website/upload/gff.html> (<https://useast.ensembl.org/info/website/upload/gff.html>) for required information in a gtf file.

Table 2 shows the first three lines of 22.gtf.

The "score" column values represent the confidence in a feature existing in that genomic position. In this example the score contains a dot ".", which represents a missing value.

The frame column provides reading frame information. Where a "." appears in the gtf table, this means that the information is not available.

In Table 2, we are looking at gene ENSG00000277248.1, where the first line tells us that the feature is a gene, and the lines below this shows the transcript products for the gene as well as exons associated for that transcript.

Table 2: Preview of information presented in a gtf file

CHROMOSOME	DATA SOURCE	FEATURE	START	END	SCORE	STRAND	FRAME	ATTRIBUTE
chr22	ENSEMBL	gene	10736171	10736283	.	-	.	gene_id "ENSG00000277248.1"; gene_type "snRNA"; gene_status "NOVEL"; gene_name "U2AF1";

CHROMOSOME	DATA SOURCE	FEATURE	START	END	SCORE	STRAND	FRAME	ATTRIBUTE
chr22	ENSEMBL	transcript	10736171	10736283	.	-	.	gene_id "ENSG00000277" transcript_id "ENST00000615" gene_type "snRNA" gene_status "NOVEL" gene_name "U2.14-201"; transcript_type "snRNA"; transcript_status "NOVEL"; transcript_name "U2.14-201"; level "basic"; transcript_support "NA";
chr22	ENSEMBL	exon	10736171	10736283	.	-	.	gene_id "ENSG00000277" transcript_id "ENST00000615" gene_type "snRNA" gene_status "NOVEL" gene_name "U2.14-201"; transcript_type "snRNA"; transcript_status "NOVEL"; transcript_name "U2.14-201"; exon_number 1; exon_id "ENSE00003736" level 3; tag "basic"; transcript_support "NA";

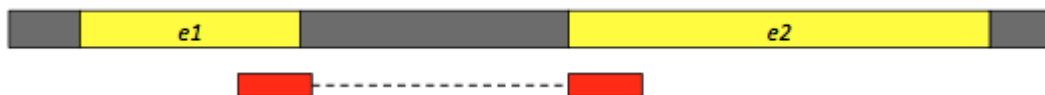


Figure 1: A sequencing read (red fragments) aligning two exons (e1 and e2). Modified from: <https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/rb-rnaseq/tutorial.html> (<https://training.galaxyproject.org/training-material/topics/transcriptomics/tutorials/rb-rnaseq/tutorial.html>)

To view the gtf or other tabular data in Unix, we can cat the file and then pipe or send the output to the column command, which allows us to print the columns so that they are nicely aligned. In this case, we use the -t option in column to determine the number of columns in the input and by default white space is used to separate the columns. Piping to less -S allows us to truncate some really long lines like those found in the attributes column of the gtf file and this also allows us to scroll horizontally to view additional columns. Hit Q to exit the cat command.

```
cat 22.gtf | column -t | less -S
```

```
chr22  ENSEMBL  gene           10736171  10736283  .  -  .  gene_id
chr22  ENSEMBL  transcript     10736171  10736283  .  -  .  gene_id
chr22  ENSEMBL  exon          10736171  10736283  .  -  .  gene_id
```

Visualizing the reference genome and annotations

One of the things we will be doing quite often is to visualize genomics data using some sort of genome browser. In this course series, we will use a popular one called Integrative Genome Viewer (IGV (<https://software.broadinstitute.org/software/igv/>)). Let's visualize human chromosome 22 in IGV as a way to start becoming acquainted with the software.

Here, we will be using IGV that is installed on our local machine.

Instructions to install IGV:

Submit ticket to service.cancer.gov to get IGV installed on your machine.

We also need to download 22.fa and 22.gtf to our local desktop. To do this copy both into the ~/public directory.

```
cp 22.fa ~/public
```

```
cp 22.gtf ~/public
```

From there, download 22.fa and 22.gtf to our local machine. **Note where the files were downloaded.**

When we open IGV, we will see the window shown in (Figure 2).

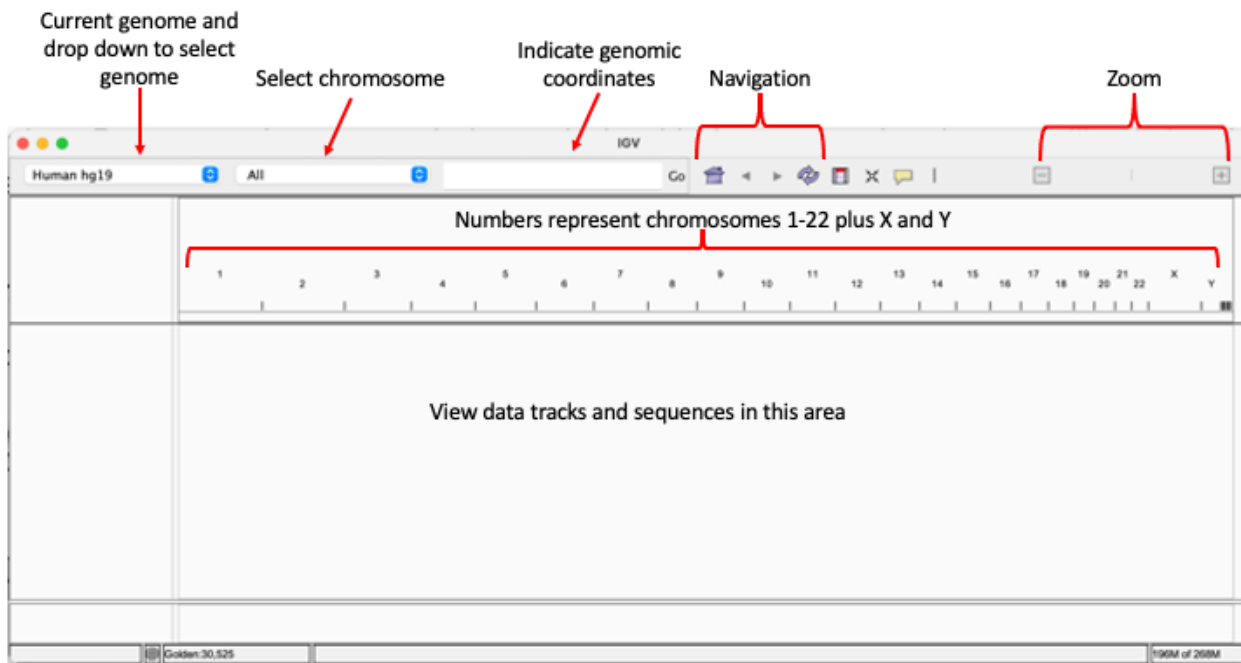


Figure 2

IGV comes preloaded with several genomes, but if we do not see the one we need in the drop down menu we can always load it from the Genomes drop down on the menu bar where it gives us some options including loading the genome from local file or from the web or URL (Figure 3). Compatible file formats include FASTA, JSON, or ".genome".

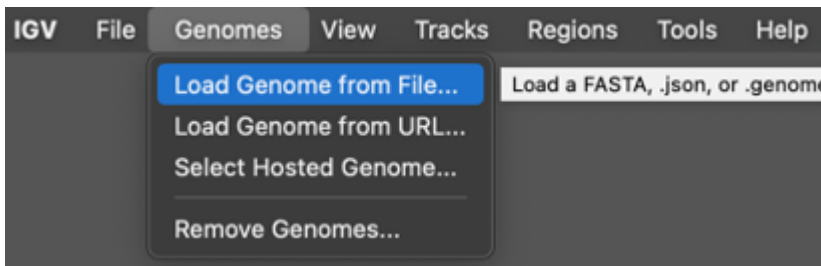


Figure 3:

To load data into IGV, we will choose one of the options in the File drop down in the menu bar (see Figure 4). Note that from the File drop down, we can take snap shots of our view and save as either PNG or SVG images.

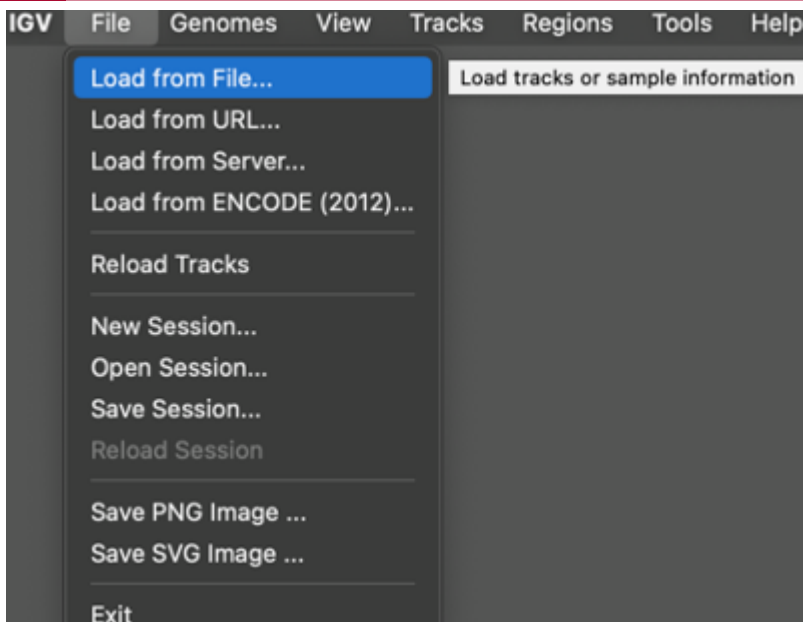


Figure 4:

Since IGV loaded hg19 upon startup, we will load the chromosome 22 reference genome using the Load from File option under the Genomes drop down in the menu bar (Figure 5).

Chromosome 22 loaded onto IGV

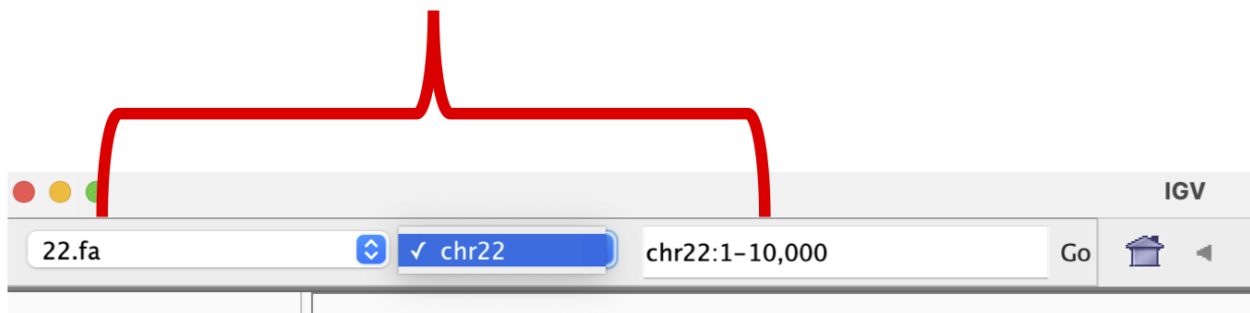


Figure 5:

Next, we will load the 22.gtf file onto IGV so we can see the genes aligned to the reference genome (Figure 6). The blue rectangles represent genes and transcripts, we can zoom in to look at ENSG00000280363.1 by searching for this gene ID in the Go box (we can actually search by coordinates, ID, or name).

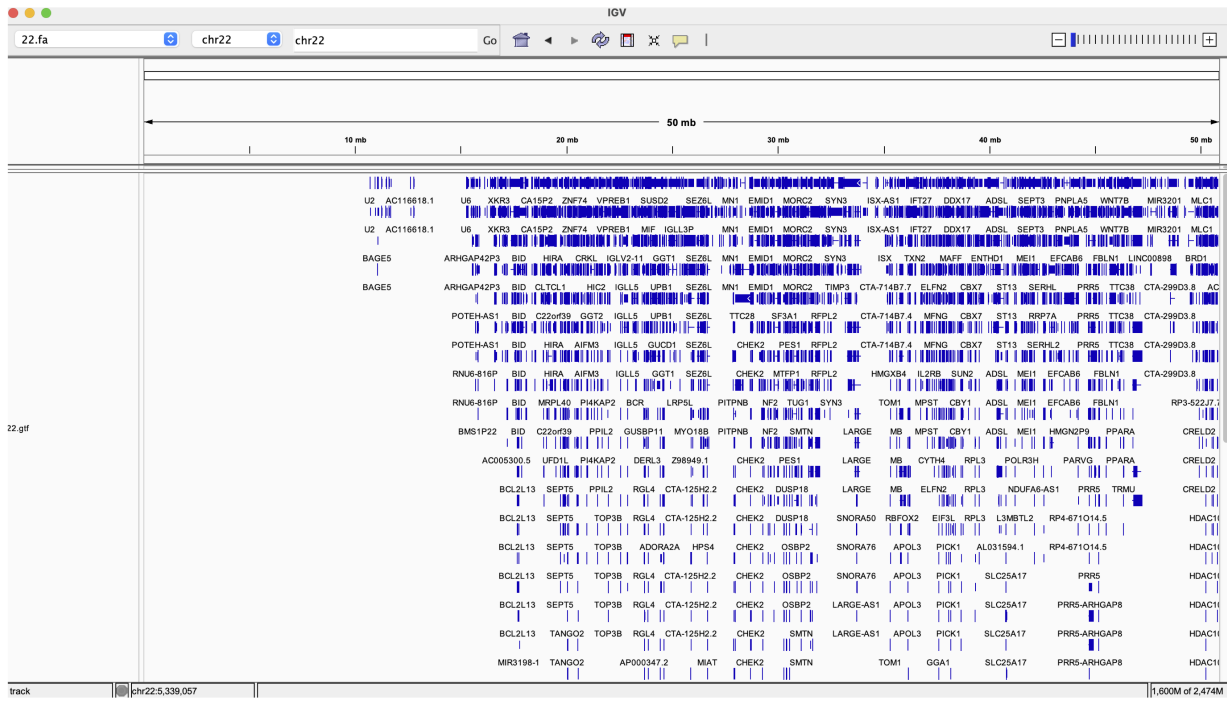


Figure 6

In Figure 7, we zoom into ENSG00000280363.1 on IGV, where ENSG00000280363.1 is the Ensembl accession number for gene CU104787.1. This gene overlaps ENSG00000279973.1, which is bage5. If we click on the solid blue line for each of the genes, a dialog box with more information will appear. The arrows on the genes point to the direction in which the gene was transcribed (these two genes were transcribed in different directions). Note that even though we search by gene ID in the Go box, the genomic coordinates will be displayed after IGV finds what we are looking for.

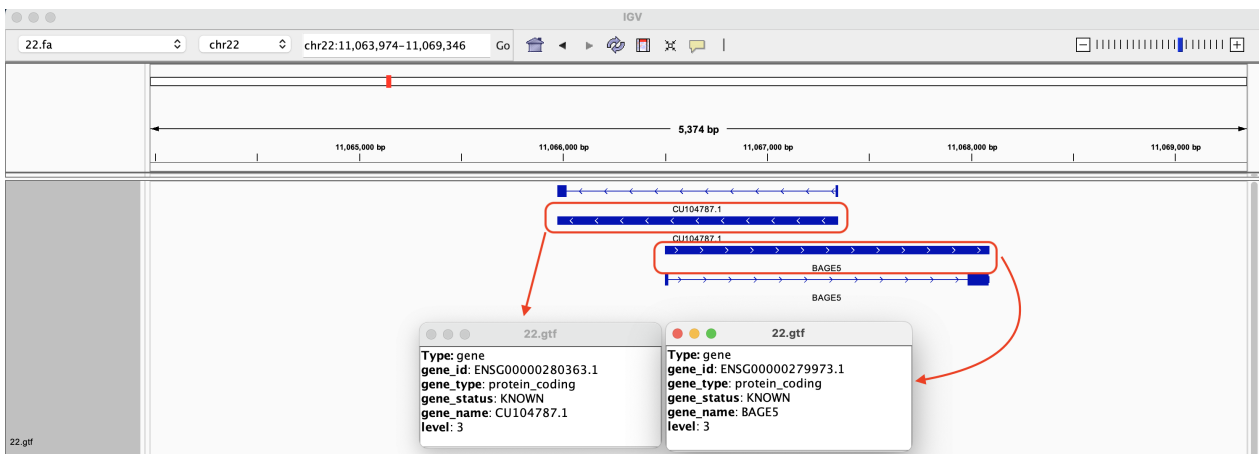


Figure 7

The transcripts are depicted by solid rectangles separated by lines (Figure 8). The solid rectangles are exons and the lines connecting are introns. If we click on the transcripts, a box pops up and we get more information regarding the transcript.

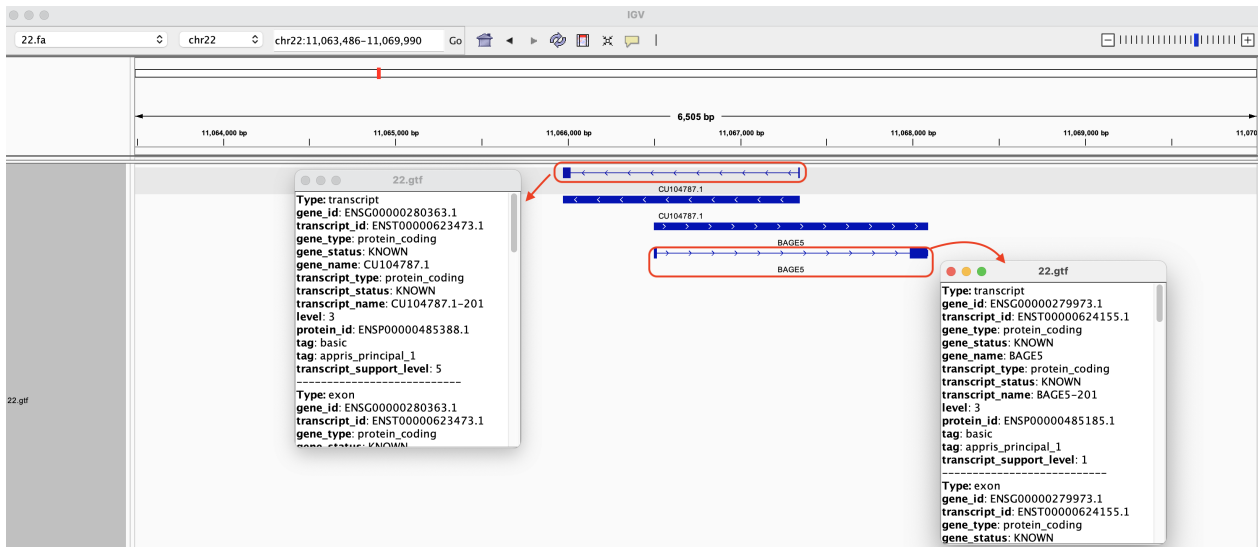


Figure 8

Within the exons, narrower solid rectangles represent untranslated regions or UTRs (Figure 9).

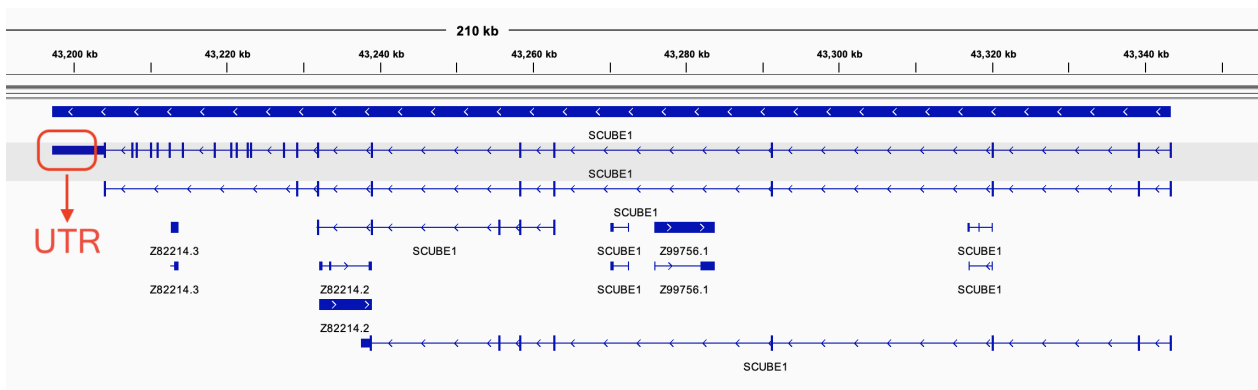


Figure 9

Among the many features of IGV, is the ability for users to zoom in close enough to see the bases (Figure 10).

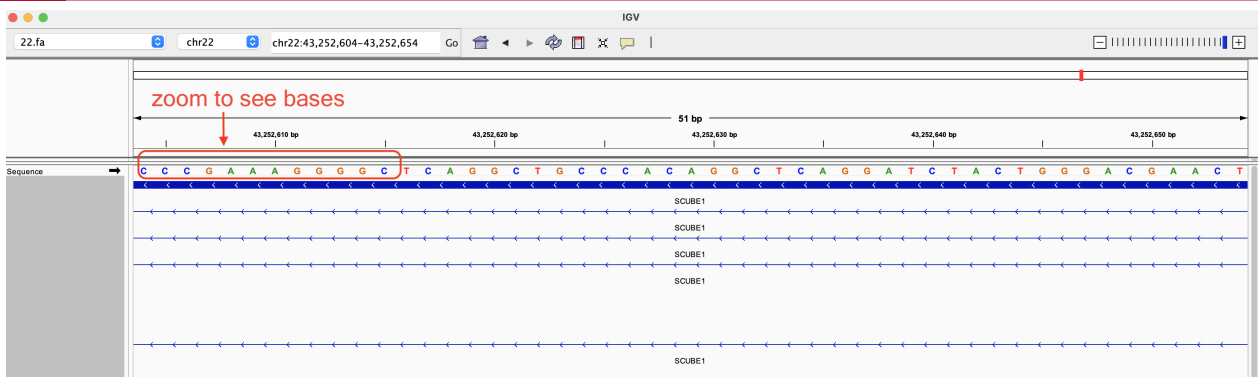


Figure 10

Lesson 10: Introducing the FASTQ file and assessing sequencing data quality

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 9 Review

In the previous lesson, we explored the reference genomes and genome annotation files that are needed in our analysis of the Human Brain Reference (HBR) and Universal Human Reference (UHR) RNA sequencing data.

Learning objectives

In lesson 9, we learned that reference genomes came in the form of FASTA files, which essentially store nucleotide sequences. In this lesson, we will learn about the FASTQ file, which is the file format that we get from our high throughput sequencing experiment. Our goals are to

- Learn about the structure of FASTQ files
- Use the command line to retrieve some basic FASTQ file statistics
- Use FASTQC, a prebuilt application, to generate quality metrics for FASTQ files
- Interpret quality check results generated from FASTQC

The skills learned can be applied to your own research and will be used when we learn more about RNA sequencing in subsequent lessons. In this lesson, we will continue to use the Human Brain Reference and Universal Human Reference datasets.

The directory in which the HBR and UHR dataset resides is `~/biostar_class/hbr_uhr`. Let's go ahead and change into this folder. Because we are talking about the sequencing data in this lesson, we will then need to change into the reads folder.

```
cd ~/biostar_class/hbr_uhr
```

```
cd reads
```

Let's now list (using `ls`) the contents of the reads folder. We use `-1` to make `ls` list one item per row.

```
ls -l
```

```
HBR_1_R1.fq
HBR_1_R2.fq
HBR_2_R1.fq
HBR_2_R2.fq
HBR_3_R1.fq
HBR_3_R2.fq
UHR_1_R1.fq
UHR_1_R2.fq
UHR_2_R1.fq
UHR_2_R2.fq
UHR_3_R1.fq
UHR_3_R2.fq
```

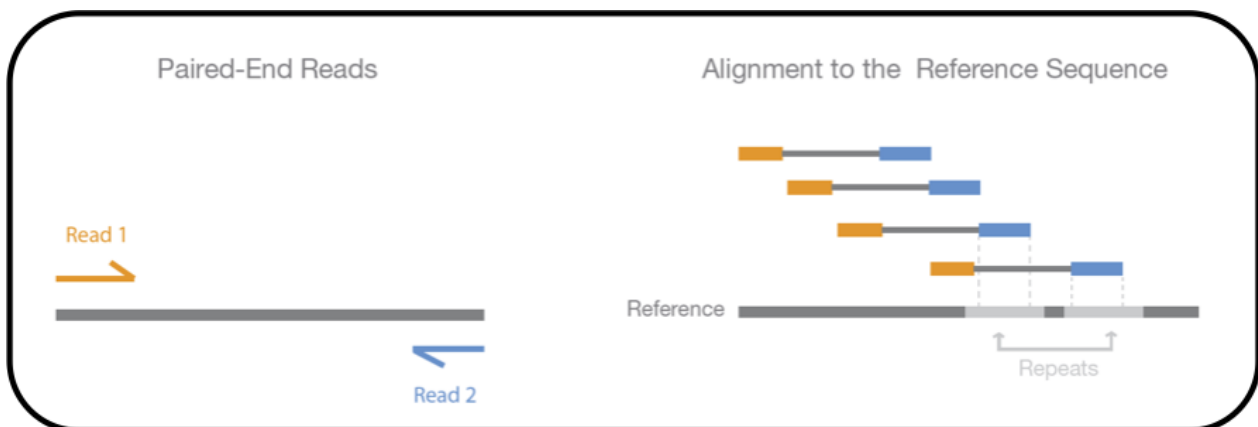


Figure 1: Paired end sequencing. Source: <https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/design-considerations/> (<https://www.ebi.ac.uk/training/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/design-considerations/>).

Each FASTQ file is composed of many sequences. We can use the seqkit tool and its stats function to get statistics on our FASTQ files. In the seqkit command below, we use * to denote wild card in order to have seqkit run stats for all FASTQ files.

```
seqkit stats HBR*.fq UHR*.fq
```

Our query of the stats for the FASTQ files generates the results below where we are informed of things such as the number of sequences (or reads) in a FASTQ file. For the HBR_1 biological replicates, both files in the pair (R1 and R2) have 118,571 sequences. The average length of the sequences in these files is 100 bases. Pairs in paired end sequencing should have the


```
grep @HWI HBR_1_R1.fq | wc -l
```

```
118571
```

```
grep @HWI HBR_1_R2.fq | wc -l
```

```
118571
```

Here we have learned how to use an existing application as well as stand alone commands to get some basic statistics from FASTQ files. However, doing this repeatedly for every FASTQ file can become cumbersome so we typically turn to tools like FASTQC to generate sequencing quality metrics.

FASTQC is a tool developed by the Babraham Institute (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>). Their documentation can be found at <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/> (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/>). A video tutorial can be found at <https://youtu.be/bz93ReOv87Y> (<https://youtu.be/bz93ReOv87Y>).

We can take a look at the instructions for running FASTQC by typing the following at our command prompt.

```
fastqc --help
```

Once the instructions have been pulled up, we see that to run FASTQC we simply do the following.

```
fastqc input1
```

If we have multiple FASTQ files then we can enter them in a series.

```
fastqc input1 input2 input3 ... inputN
```

Let's run FASTQC on the HBR and UHR sequencing data.

```
fastqc HBR_*.fq UHR_*.fq
```

FASTQC will print out the status of the analysis. Analysis status for HBR_1_R1.fq is shown below but note that this status will be printed for all inputs in the FASTQC command.

```
Started analysis of HBR_1_R1.fq
Approx 5% complete for HBR_1_R1.fq
Approx 10% complete for HBR_1_R1.fq
Approx 15% complete for HBR_1_R1.fq
Approx 20% complete for HBR_1_R1.fq
Approx 25% complete for HBR_1_R1.fq
Approx 30% complete for HBR_1_R1.fq
Approx 35% complete for HBR_1_R1.fq
Approx 40% complete for HBR_1_R1.fq
Approx 45% complete for HBR_1_R1.fq
Approx 50% complete for HBR_1_R1.fq
Approx 55% complete for HBR_1_R1.fq
Approx 60% complete for HBR_1_R1.fq
Approx 65% complete for HBR_1_R1.fq
Approx 70% complete for HBR_1_R1.fq
Approx 75% complete for HBR_1_R1.fq
Approx 80% complete for HBR_1_R1.fq
Approx 85% complete for HBR_1_R1.fq
Approx 90% complete for HBR_1_R1.fq
Approx 95% complete for HBR_1_R1.fq
Analysis complete for HBR_1_R1.fq
...
```

Now if we list the files in the hbr_uhr directory, we will see that each FASTQ file has a corresponding FASTQC html report and zip folder. We can view the html version of the QC report in a web browser and this is what most people would do. The QC results (text and figures that appears in the html report) are also available in the zip folder.

```
ls
```

```
HBR_1_R1.fq          HBR_1_R2_fastqc.zip  HBR_2_R2_fastqc.html  HBF
HBR_1_R1_fastqc.html HBR_2_R1.fq          HBR_2_R2_fastqc.zip  HBF
HBR_1_R1_fastqc.zip  HBR_2_R1_fastqc.html HBR_3_R1.fq          HBF
HBR_1_R2.fq          HBR_2_R1_fastqc.zip  HBR_3_R1_fastqc.html UHF
HBR_1_R2_fastqc.html HBR_2_R2.fq          HBR_3_R1_fastqc.zip  UHF
```

Let's unzip HBR_1_R1_fastqc.zip and take a peek at the content.

```
unzip HBR_1_R1_fastqc.zip
```



```
cd HBR_1_R1_fastqc
```

```
ls
```

In the HBR_1_R1_fastqc folder, the QC summary is available in the summary.txt file. The QC results is in fastqc_data.txt and the figures are in the Images folder.

Icons	fastqc.fo	fastqc_report.html
Images	fastqc_data.txt	summary.txt

```
cd Images
```

```
ls
```

adapter_content.png	per_sequence_gc_content.png
duplication_levels.png	per_sequence_quality.png
per_base_n_content.png	per_tile_quality.png
per_base_quality.png	sequence_length_distribution
per_base_sequence_content.png	

Now, let's change back into the ~/biostar_class/hbr_uhr/reads folder and move HBR_1_R1_fastqc.html to our ~/public directory and then right click to open it in a new browser tab. In the cp command below ~ denotes home directory.

```
cd ~/biostar_class/hbr_uhr/reads
```

```
cp HBR_1_R1_fastqc.html ~/public
```

If we now listed the contents of the ~/public directory, then we will see HBR_1_R1_fastqc.html.

```
ls ~/public
```

In the FASTQC report, the first thing we see is the Summary panel that allows us to navigate to different portions of the FASTQC report. In this summary panel, we see that we have one failed

module (red circle with x) and three warnings (yellow circle with !). In the main report pane, the first information is the Basic Statistics (Figure 2), which tells us

- Name of the FASTQ file that we are working with (HBR_1_R1.fq in this case)
- The number of sequences in the file
- Number of sequences flagged with poor quality (0 in the HBR_1_R1.fq file)
- The sequence length (ie. how many bases are in each sequencing read of the FASTQ file). For HBR_1_R1.fq, each sequencing read is composed of 100 bases and this concurs with the results from seqkit stats.

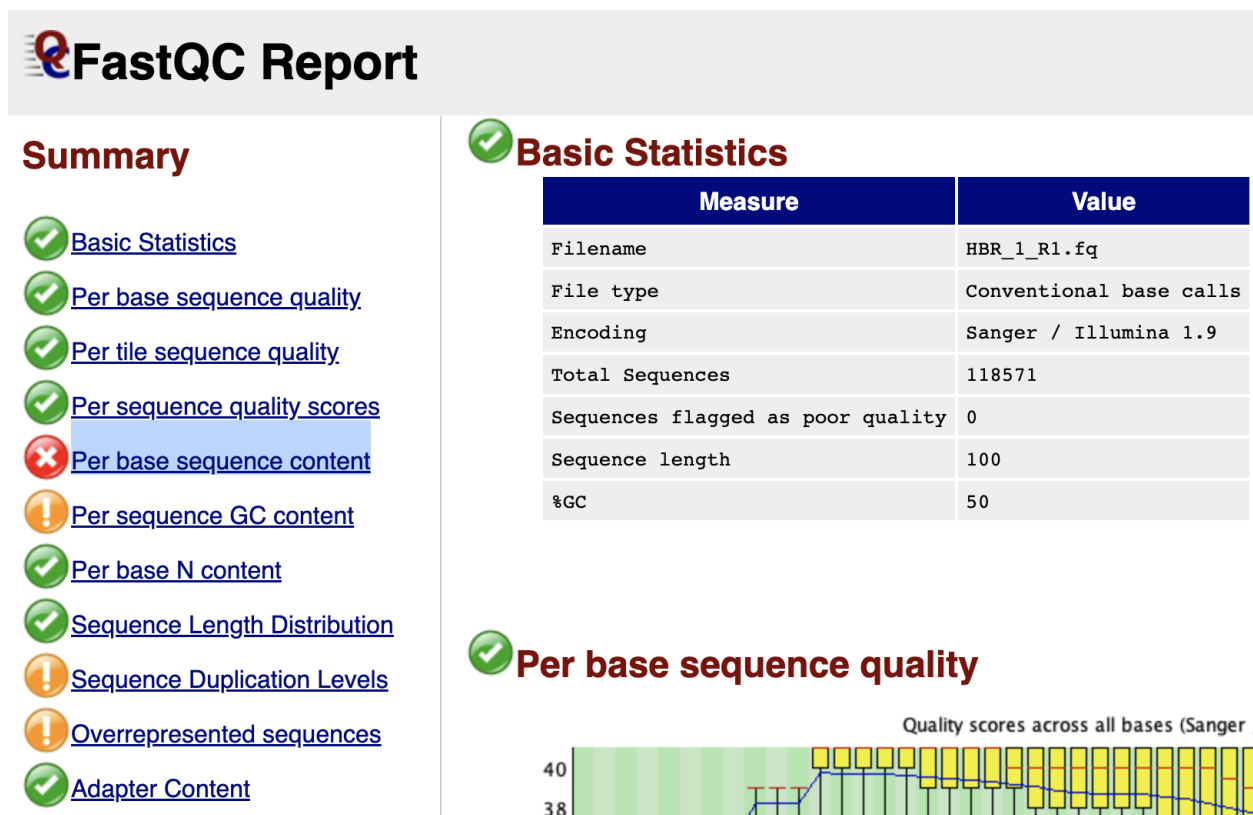


Figure 2

Next, we can see the "Per base sequence quality" plot (Figure 3).

This chart plots the error likelihood at each position averaged over all measurements.

- The vertical axis are the quality scores that you see in row 4 of the sequencing reads in a FASTQ file. These quality scores represent error probabilities, where:
 - 10 corresponds to 10% error (1/10),
 - 20 corresponds to 1% error (1/100),
 - 30 corresponds to 0.1% error (1/1000) and
 - 40 corresponds to one error every 10,000 measurements (1/10,000) that is an error rate of 0.01%

- The three colored bands (green, yellow, red) illustrate the typical labels assigned to these measure:
 - reliable (28-40, green)
 - less reliable (20-28, yellow)
 - and error prone (1-20, red)
- The yellow boxes contain 50% of the data, the whiskers indicate the 75% outliers.
- The red line inside the yellow boxes is the median quality score for that base.
- The blue line is the average quality score at a particular base.

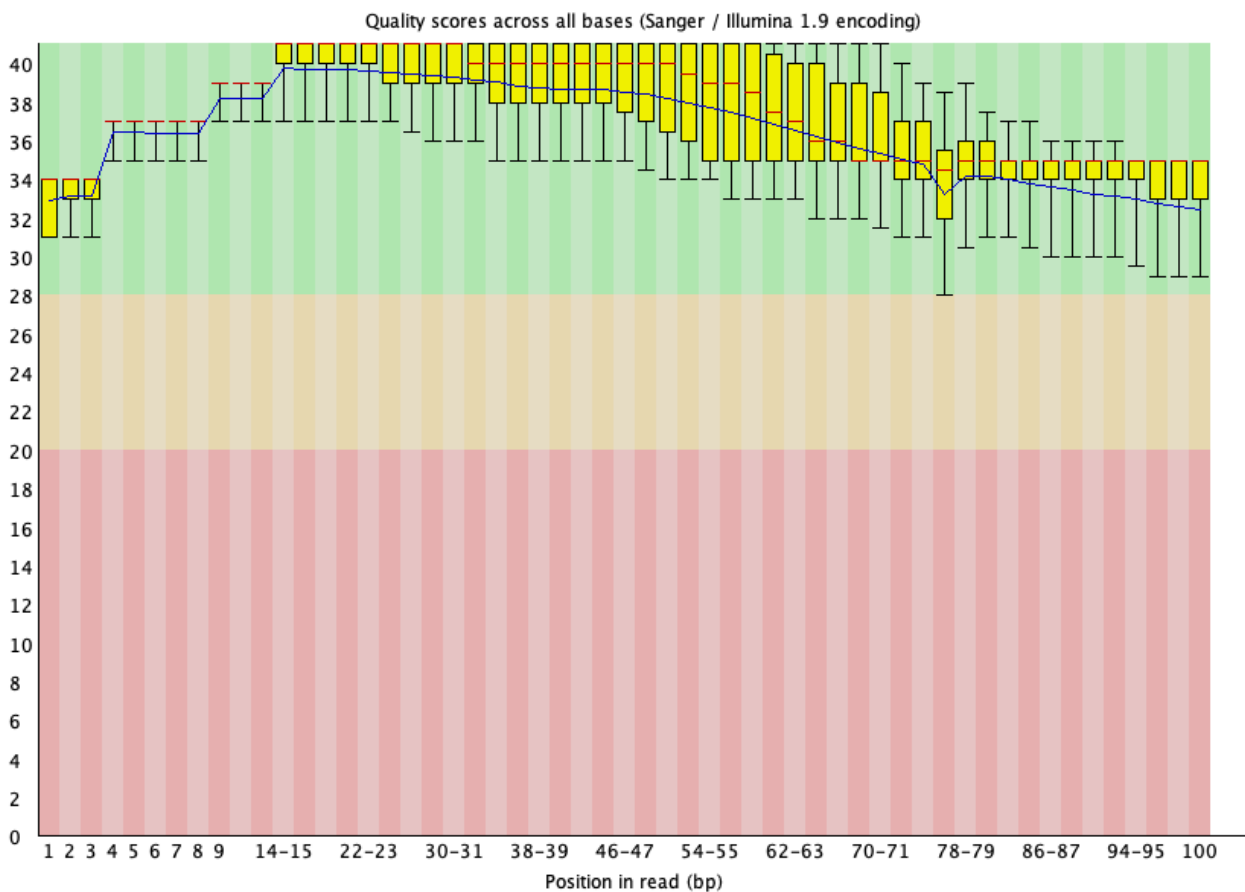


Figure 3

Next, in Figure 4, we have the "Per tile sequence quality" plot. This graph tells us whether something is wrong with a tile in the flow cell. Along the horizontal axis are the base positions. The vertical axis represents the tile number in which the read came from.

This plot compares the *average quality score of a tile* to the *average quality score of all tiles at a particular base position*. -- <https://sequencing.qcfail.com/articles/position-specific-failures-of-flowcells/> (<https://sequencing.qcfail.com/articles/position-specific-failures-of-flowcells/>)

The interpretation for this plot is as follows

- Colder colors indicate that the average quality score at a tile is at or above the average quality score of all tiles at a particular base position. So a plot that is entirely blue is good (Figure 4).
- Hotter colors indicate that the average quality score at a tile is below the average quality score of all tiles at a particular base position. So a plot with red indicates a part of the flow cell has problems (Figure 5).

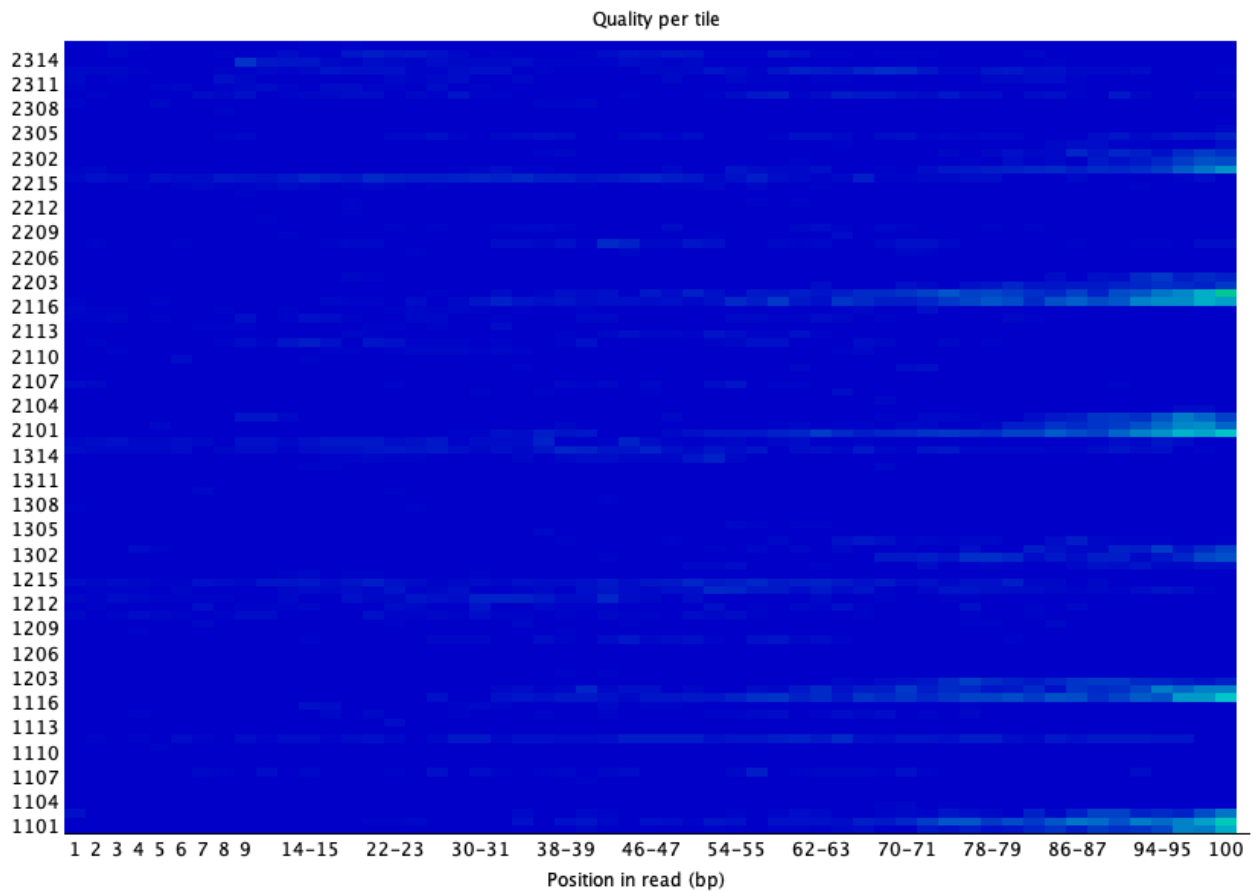


Figure 4

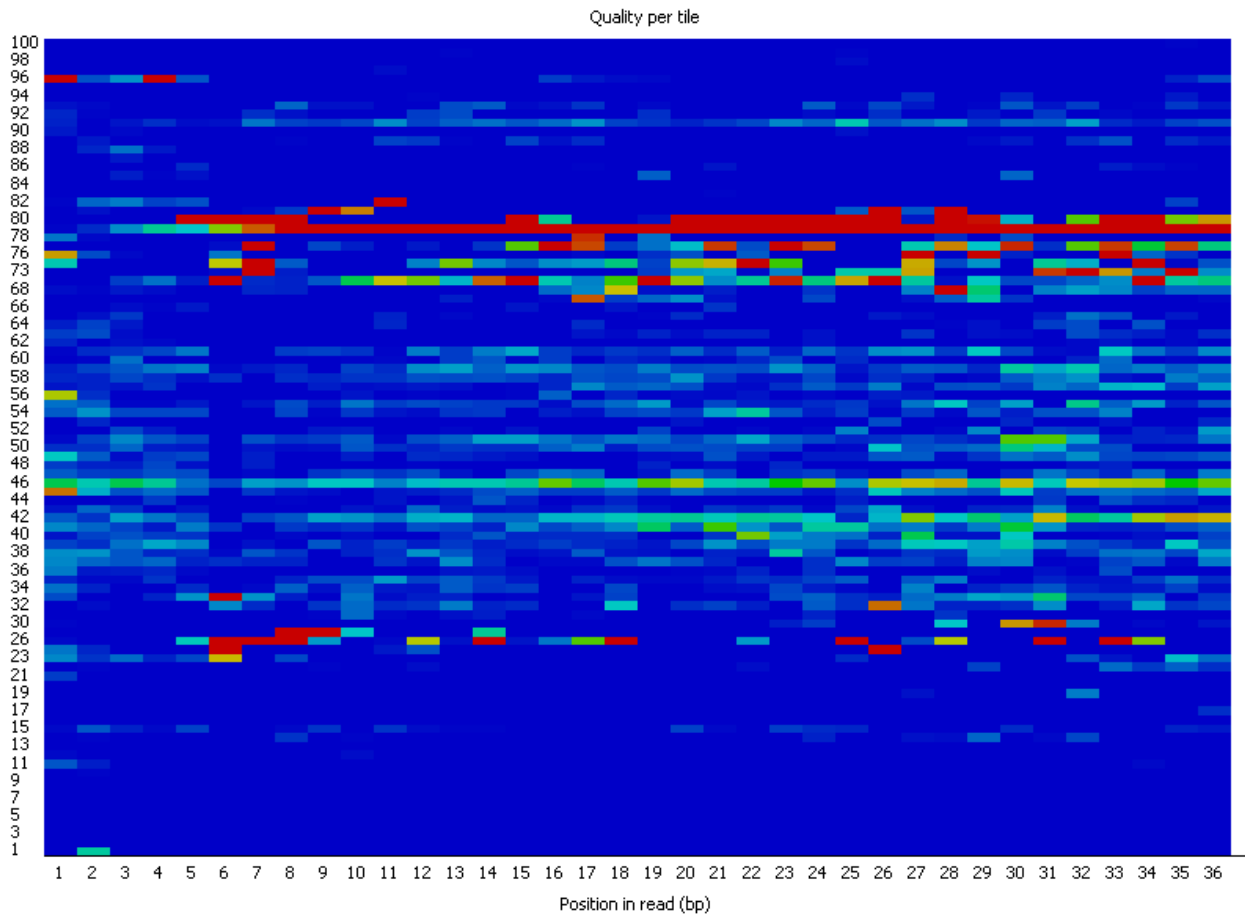


Figure 5. Source: <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/12%20Per%20Tile%20Sequence%20Quality.html> (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/12%20Per%20Tile%20Sequence%20Quality.html>)

Next, in Figure 6 we see a distribution of the sequence quality scores in the "Per sequence quality scores" plot, showing whether the quality of a subset of sequences is poor. Here our sequences have good quality, where most reads have quality that clusters at around 37.

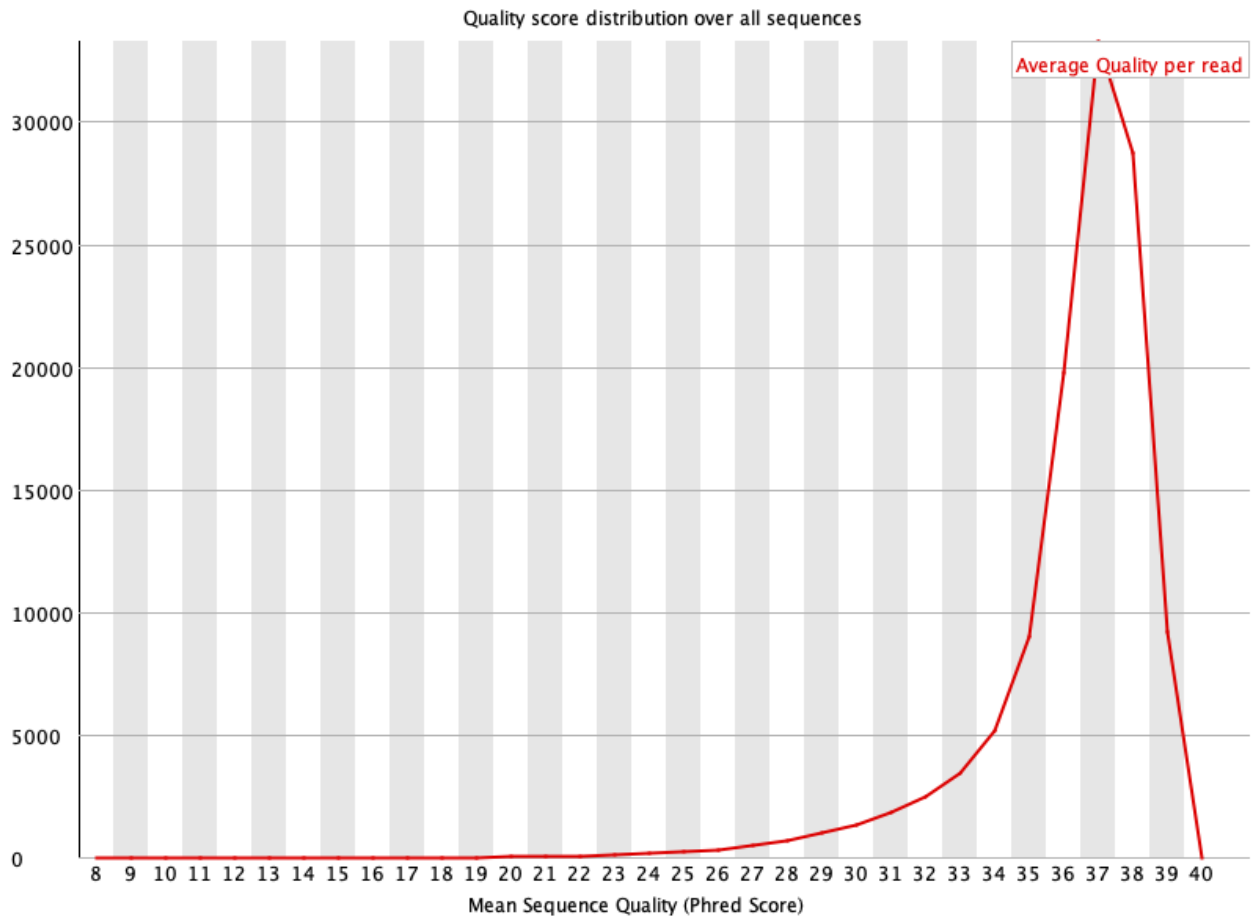


Figure 6

Figure 7 shows us the sequence make up along the bases of our reads in the "Per base sequence content" plot. If a library is random, then the percent composition of each nucleotide base (A,T,C,G) should be the same (~25%).

This module fails for HBR_1_R1.fq because difference between the percentage of A and the percentage of T is larger than 20 at a particular location **OR** the difference between the percentage of C and the percentage of G is larger than 20 at a particular location -- [Babraham bioinformatics Per base sequence content \(https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/4%20Per%20Base%20Sequence%20Content.html\)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/4%20Per%20Base%20Sequence%20Content.html).

In HBR_1_R1.fq, it looks like the difference between the percent composition of A and T at base position 2 is causing the failure. Unfortunately, this type of unevenness in base distribution at the beginning of a read is observed in RNA sequencing due to priming with random hexamers during the library preparation stage.

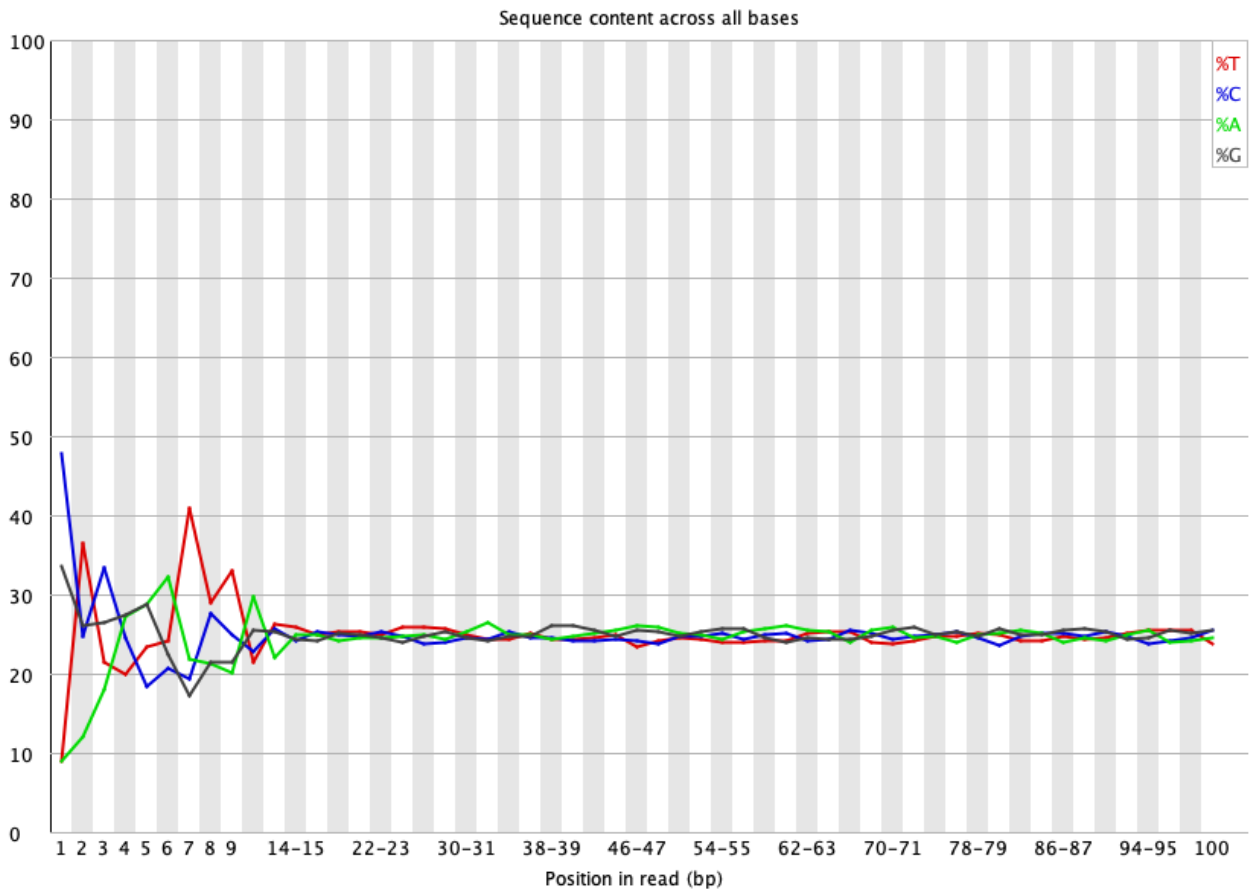


Figure 7

Figure 8 shows the GC content across each sequence compared to a normal distribution in what is called the "Per sequence GC content" plot. The GC content in HBR_1_R1.fq is off from the normal theoretical distribution.

The peak of this theoretical distribution is an estimate of the GC content of the underlying genome. Deviation from of the GC content from the theoretical distribution could be caused by contamination or sequencing bias. -- Babraham bioinformatics Per base sequence content (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/5%20Per%20Sequence%20GC%20Content.html>).

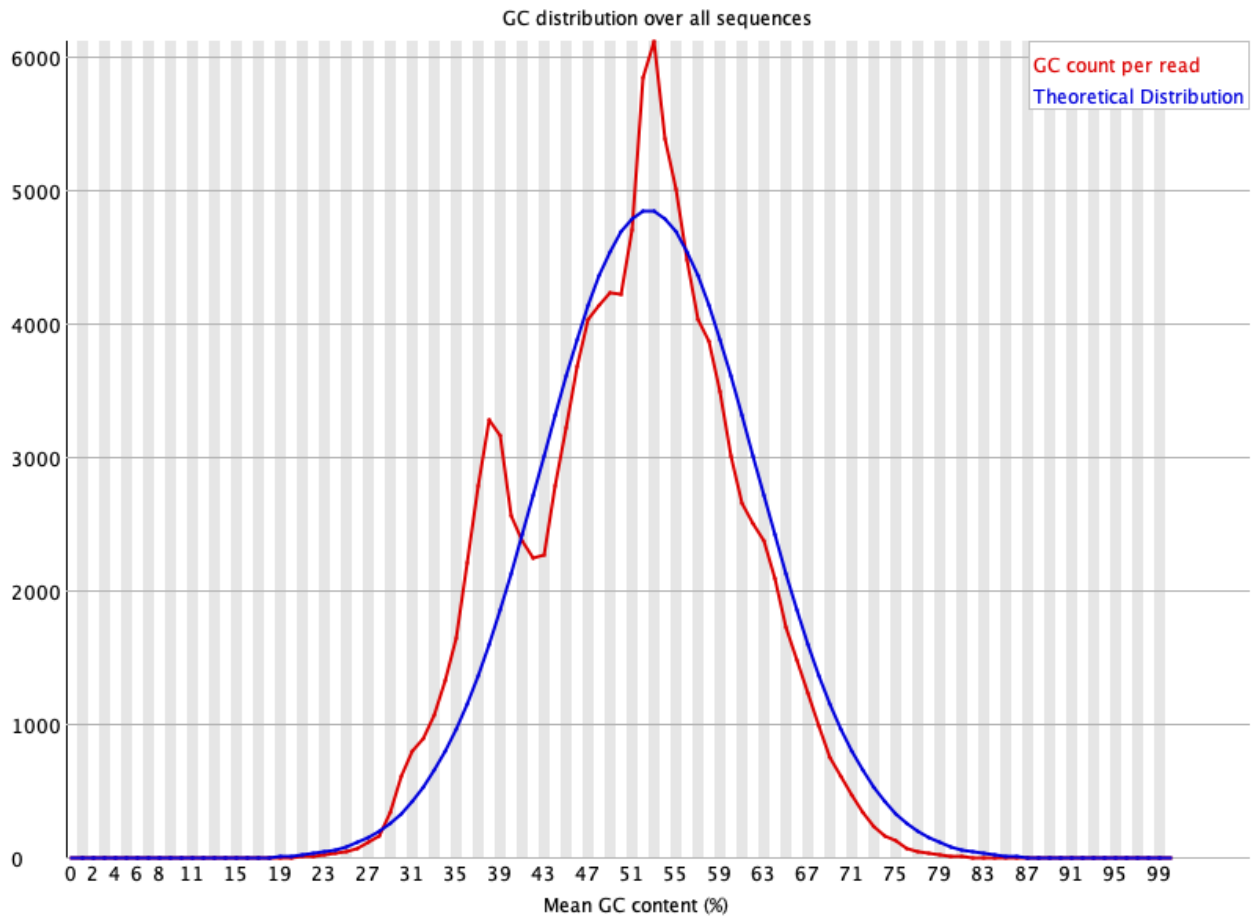


Figure 8

Whether we have any unknown bases in our sequencing reads is shown in Figure 9, which is the "Per base n content" plot.

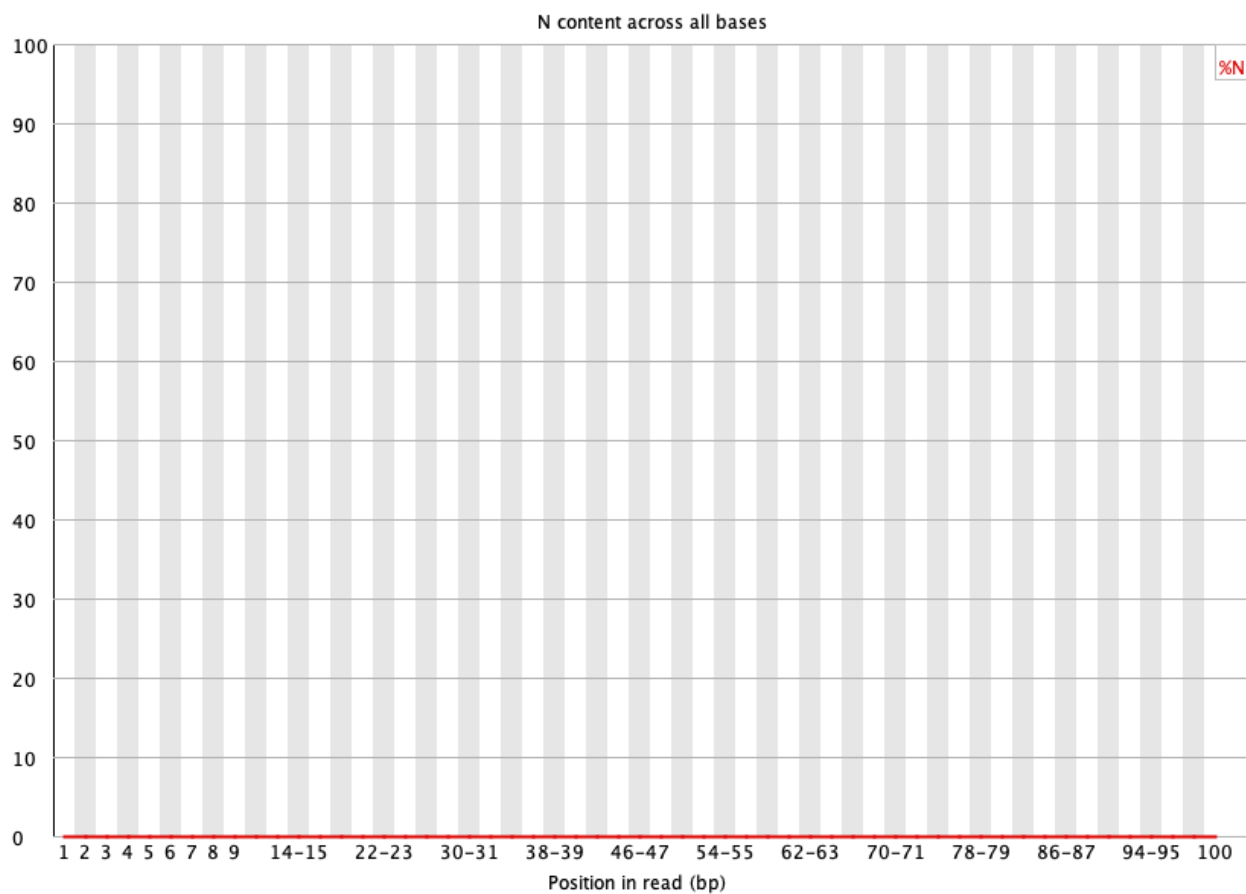


Figure 9

Figure 10 shows the sequence length distribution of HBR_1_R1.fq in what is known as the "Sequence Length Distribution" plot.

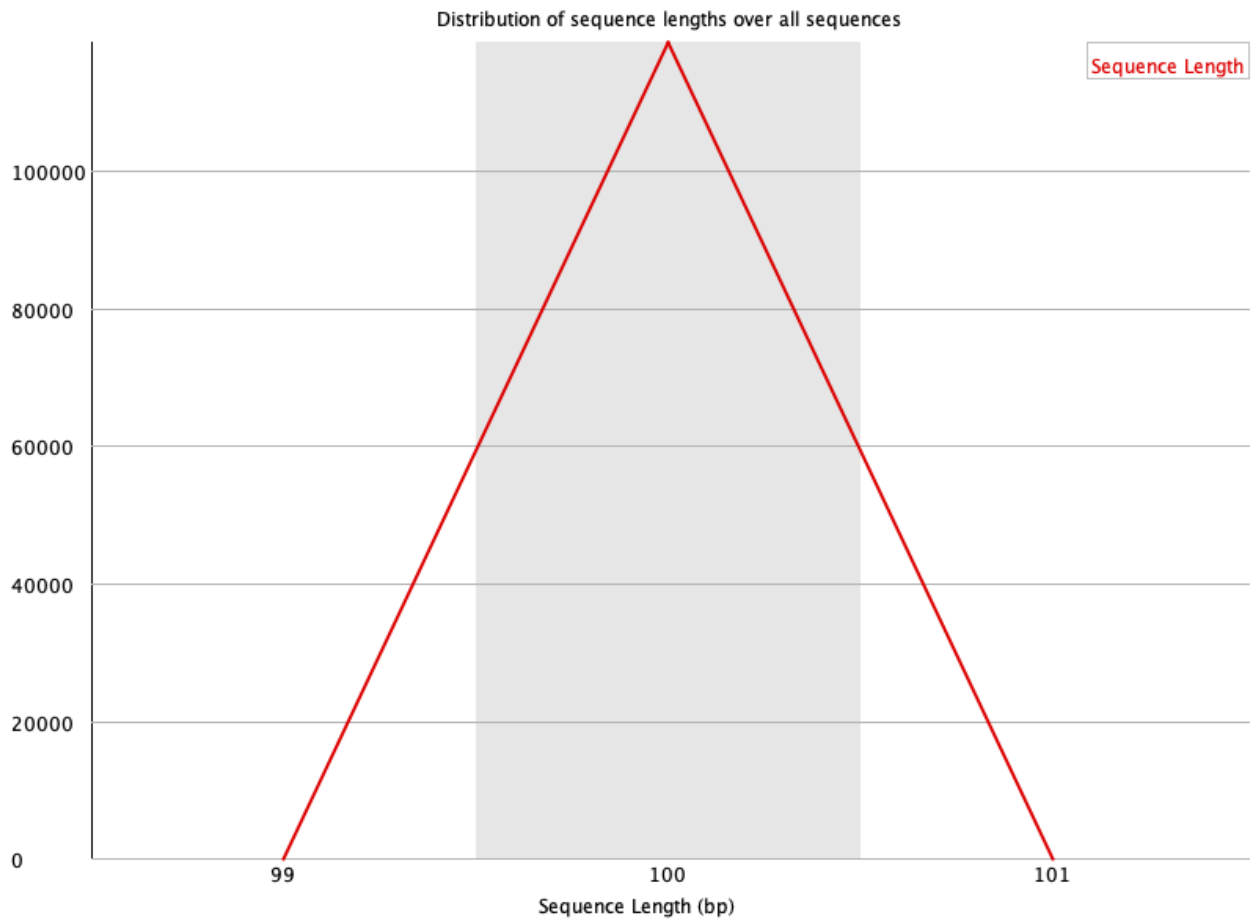


Figure 10

Figure 11 shows the sequence duplication levels. High levels of duplication may indicate an enrichment bias such as over-amplification in the PCR step. Otherwise, most sequences will occur only once.

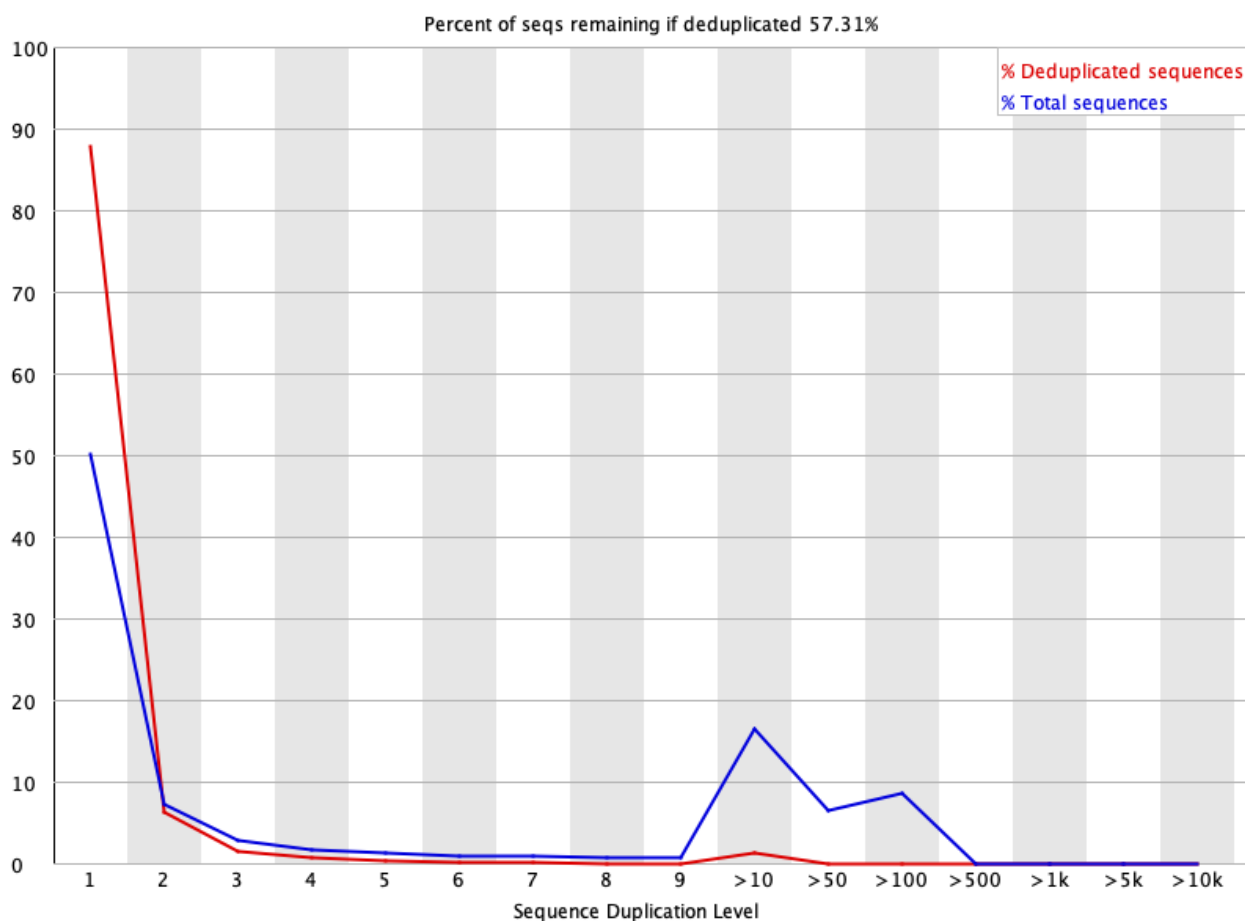


Figure 11

Figure 12 shows that we have some overrepresented sequences. As an exercise, let's copy one of the overrepresented sequences and BLAST it to find out what it is. Presence of overrepresented sequences may indicate enrichment artifact or the sequencing library is not very diverse. On the other hand, overrepresented sequences could have biological significance.

Click here to goto NCBI BLAST (<https://blast.ncbi.nlm.nih.gov/Blast.cgi>)

! Overrepresented sequences

Sequence	Count	Percentage	Possible Source
CTTATGTGATAGATGCCTCTTTAAATATCTAAGTGCTGGGTTATGAGT	444	0.37445918479223417	No Hit
CGCTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAAT	396	0.3339771107606413	No Hit
CCGCTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAA	388	0.3272300984220425	No Hit
CGGCTGTCGAGTTGTACGGCCGTTACGCCACGAGTCACGGGGTCTAACGC	382	0.3221698391680934	No Hit
CTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAATAT	381	0.3213264626257685	No Hit
CTGAGACAGAGTCGCTATCGTTATGTCTCCTTCCCGGGTCAAGGCGAAA	339	0.28590464784812475	No Hit
GCCTTATGTGATAGATGCCTCTTTAAATATCTAAGTGCTGGGTTATGA	295	0.24879607998583128	No Hit

Figure 12

Figure 13 tells us whether some of our sequencing reads have adapter content. Adapters sequences should be trimmed prior to alignment.

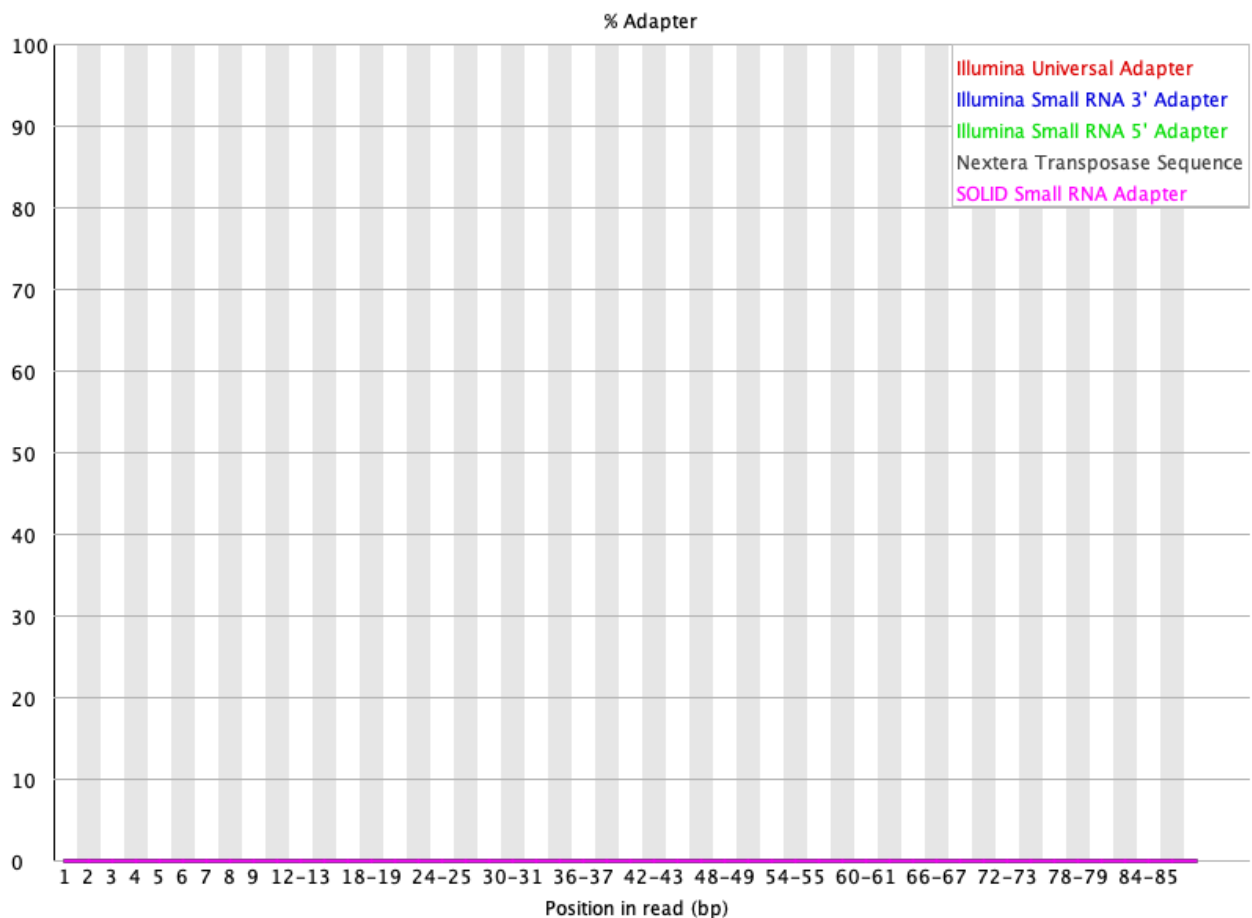


Figure 13

HBR and UHR FASTQC reports

See the links below for the HBR and UHR FASTQC reports.

[HBR_1_R1_fastqc.html](#)

[HBR_1_R2_fastqc.html](#)

[HBR_2_R1_fastqc.html](#)

[HBR_2_R2_fastqc.html](#)

[HBR_3_R1_fastqc.html](#)

[HBR_3_R2_fastqc.html](#)

[UHR_1_R1_fastqc.html](#)

[UHR_1_R2_fastqc.html](#)

[UHR_2_R1_fastqc.html](#)

[UHR_2_R2_fastqc.html](#)

[UHR_3_R1_fastqc.html](#)

[UHR_3_R2_fastqc.html](#)

Lesson 11: Merging FASTQ quality reports and data cleanup

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 10 Review

In the previous lesson, we learned about the structure of the FASTQ file, which stores our raw sequencing reads. Next, we learned to use a tool called FASTQC to assess the quality of each of the FASTQ files in the Human Brain Reference (HBR) and Universal Human Reference (UHR) dataset.

Learning objectives

As described in the Lesson 10 review above, we generated quality reports for each of the FASTQ files in the Human Brain Reference and Universal Human Reference dataset using FASTQC. However, interrogating 12 individual FASTQC reports is cumbersome. In this lesson, we will focus on the following.

- Merge FASTQC reports using a tool called MultiQC so that we can interrogate one report rather than multiple.
- Learn to perform quality and adapter trimming on FASTQ files.

The skills learned can be applied to your own research and will be used when we learn more about RNA sequencing in subsequent lessons. In this lesson, we will continue to work with the HBR and UHR datasets.

Merging individual FASTQC reports

We previously stored FASTQC results for the HBR and UHR raw sequencing data in the `~/biostar_class/hbr_uhr/QC` directory (recall that `~` denotes home directory). So before getting started, change into this folder.

```
cd ~/biostar_class/hbr_uhr/QC
```

FASTQC generated 12 html reports, here, we will merge them using the tool MultiQC.

MultiQC searches a given directory for analysis logs and compiles a HTML report. It's a general use tool, perfect for summarising the output from numerous bioinformatics tools. -- <https://multiqc.info> (*https://multiqc.info*).

MultiQC "knows" the report formats of many existing NGS tools: FastQC, cutadapt, bowtie2, tophat, STAR, kallisto, HISAT2, samtools, featureCounts, HTSeq, MACS2, Picard, GATK, etc. -- <https://wikis.utexas.edu/display/bioiteam/Using+MultiQC> (*https://wikis.utexas.edu/display/bioiteam/Using+MultiQC*).

MultiQC can be used to aggregate reports from pre-alignmnet quality check as well as metrics from other downstream steps of high throughput sequencing analysis. See <https://multiqc.info/docs/> (*https://multiqc.info/docs/*) for the tools that MultiQC can generate aggregate reports for.

Below, we will take a look at the MultiQC documentation to see how to run it.

```
multiqc --help
```

MultiQC allows users to input some options but mainly to run this application, we need to specify the directory that contains our analysis logs.

```
/// MultiQC  | v1.13.dev0
```

```
Usage: multiqc [OPTIONS] [ANALYSIS DIRECTORY]
```


MultiQC aggregates results from bioinformatics analyses across many samples into a single report.

It searches a given directory for analysis logs and compiles a HTML report. It's a general use tool, perfect for summarising the output from numerous bioinformatics tools.

To run, supply with one or more directory to scan for analysis results. For example, to run in the current working directory, use 'multiqc .'

While we can specify the path of our hbr_uhr directory to MultiQC, we are in it already so if we want to aggregate the FASTQC reports in this folder we can just specify the directory path with "." to tell MultiQC to look for files to aggregate here (in the present working directory). We use the --filename option to specify a name (multiqc_hbr_uhr) for the MultiQC report.

```
multiqc --filename multiqc_report_hbr_uhr .
```

```
/// MultiQC  | v1.13
```

```
|           multiqc | Search path : /home/joe/biostar_class/hbr_uhr/(
|           searching | _____
```

```
|          fastqc | Found 12 reports
|          multiqc | Compressing plot data
|          multiqc | Report      : multiqc_report.html
|          multiqc | Data       : multiqc_data
|          multiqc | MultiQC complete
```

After running MultiQC, we can use `ls` to list the contents of our folder. We see that we have a html file (`multiqc_report.html`) that we can open to view the quality assessment summary for all of our samples.

```
ls
```

```
HBR_1_R1_fastqc.html  HBR_2_R1_fastqc.html  HBR_3_R1_fastqc.html  HIR
HBR_1_R1_fastqc.zip  HBR_2_R1_fastqc.zip  HBR_3_R1_fastqc.zip  UHF
HBR_1_R2_fastqc.html  HBR_2_R2_fastqc.html  HBR_3_R2_fastqc.html  UHF
HBR_1_R2_fastqc.zip  HBR_2_R2_fastqc.zip  HBR_3_R2_fastqc.zip  UHF
```

Let's copy `multiqc_report.html` to our public directory so we can take a look at the contents of this report in our web browser.

```
cp multiqc_report_hbr_uhr.html ~/public/multiqc_report_hbr_uhr.html
```

Upon opening the MultiQC report, we see a navigation panel (similar to what we have with the individual FASTQC reports) that allows us to quickly move to different sections of the report. We are also provided with links to get help if we have questions about how to use the MultiQC reports. To the right of the report page, we have a tool box that allows us to do things like highlighting different samples in different colors for better visualization, rename samples, and export each of the individual QC plots as an image for inclusion in presentations and/or publications. MultiQC reports are interactive.

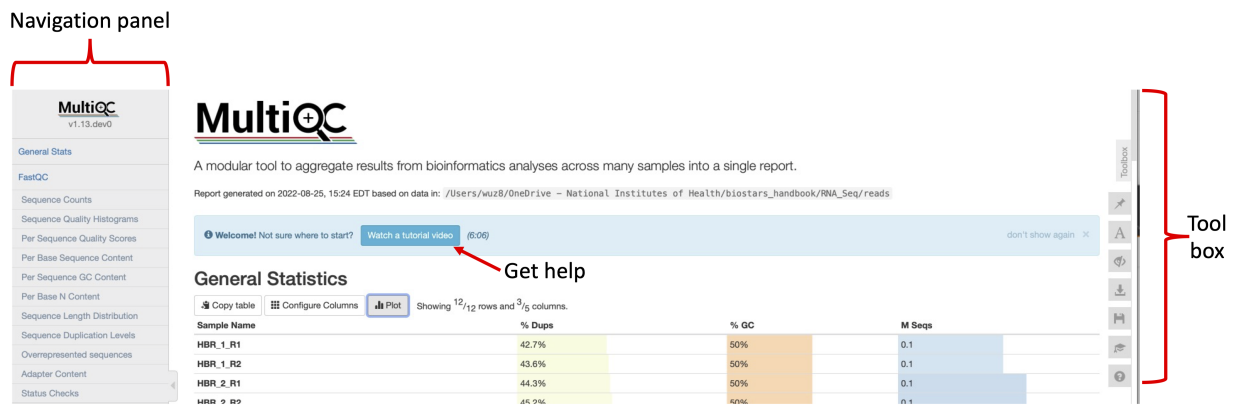


Figure 1

The next figure (Figure 2) shows some basic statistics about our samples, including percentage of duplicate reads, GC content, number of bases in the sequencing reads (or read length), percentage of modules that failed in the FASTQC report for that sample, and the total number of sequences in a FASTQ file (in Millions of sequences).

We can click on the Configure Columns tab to choose which columns we like to see in this table (Figure 3) and the plot button to visualize the data in graphical format.

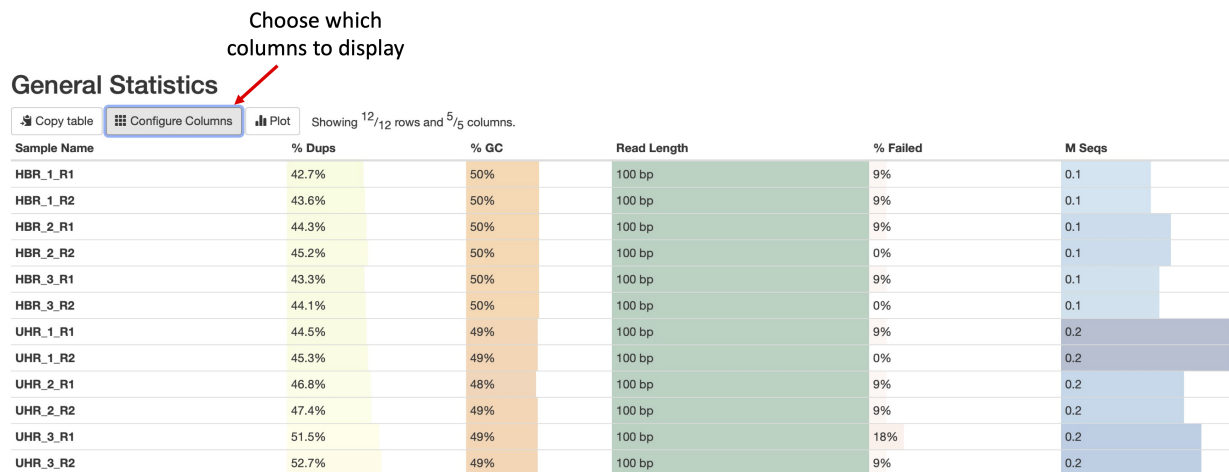


Figure 2

General Statistics: Columns

Uncheck the tick box to hide columns. Click and drag the handle on the left to change order.

Show All Show None

Sort	Visible	Group	Column	Description	ID	Scale
	<input checked="" type="checkbox"/>	FastQC	% Dups	% Duplicate Reads	percent_duplicates	None
	<input checked="" type="checkbox"/>	FastQC	% GC	Average % GC Content	percent_gc	None
	<input checked="" type="checkbox"/>	FastQC	Read Length	Average Read Length (bp)	avg_sequence_length	None
	<input checked="" type="checkbox"/>	FastQC	% Failed	Percentage of modules failed in FastQC report (includes those not plotted here)	percent_fails	None
	<input checked="" type="checkbox"/>	FastQC	M Seqs	Total Sequences (millions)	total_sequences	read_count

Close

Figure 3

The next plot (Figure 4) shows the break down of unique and duplicate reads for each FASTQ file. Again, duplication suggests some sort of enrichment bias. The default to this panel is to show the number of sequences but we can get a percentage breakdown by clicking on the Percentages tab.

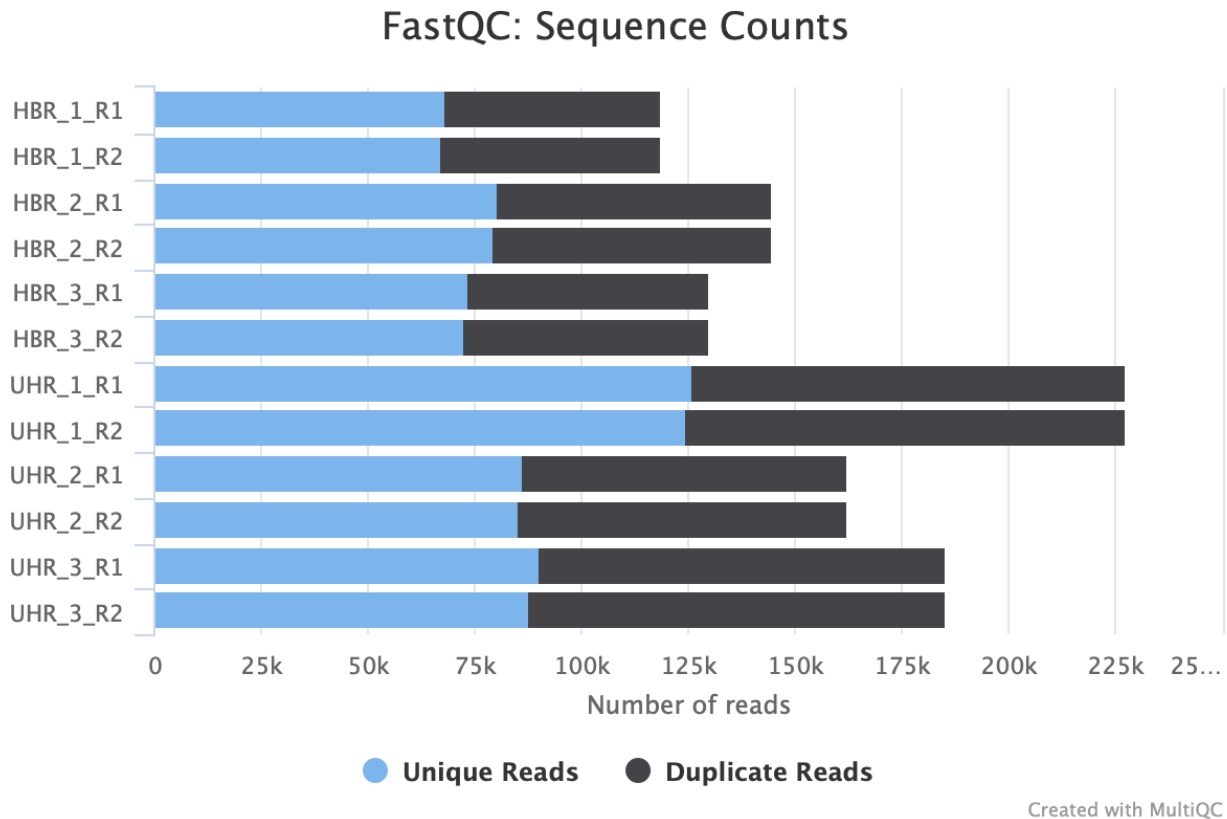


Figure 4

Figure 5 shows us the average quality score of the sequencing reads in FASTQ files along each base position. If we click on the green rectangle with the number 12 written in it, we can choose which sample we like to see in our plot (Figure 6). Regarding these boxes at the top of the QC plots, green means QC passed while orange and red indicate warning and failed, respectively.

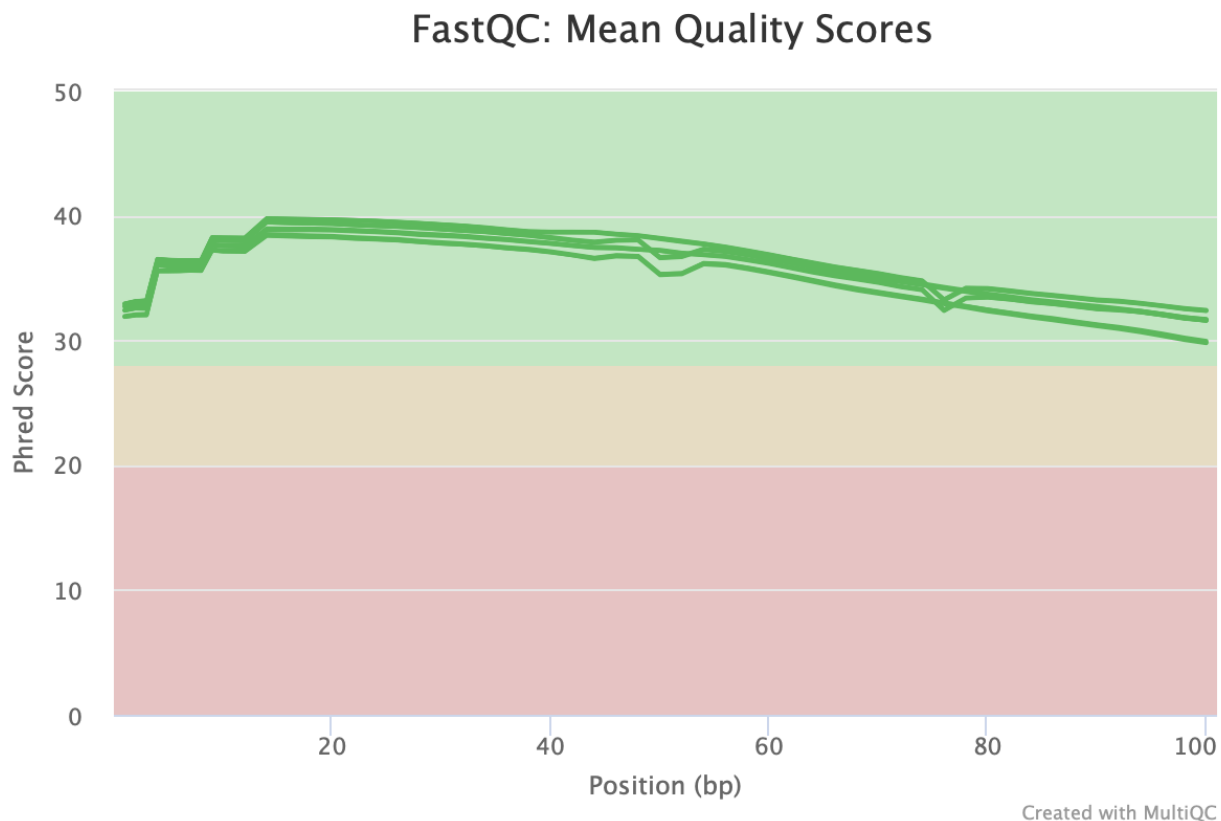


Figure 5

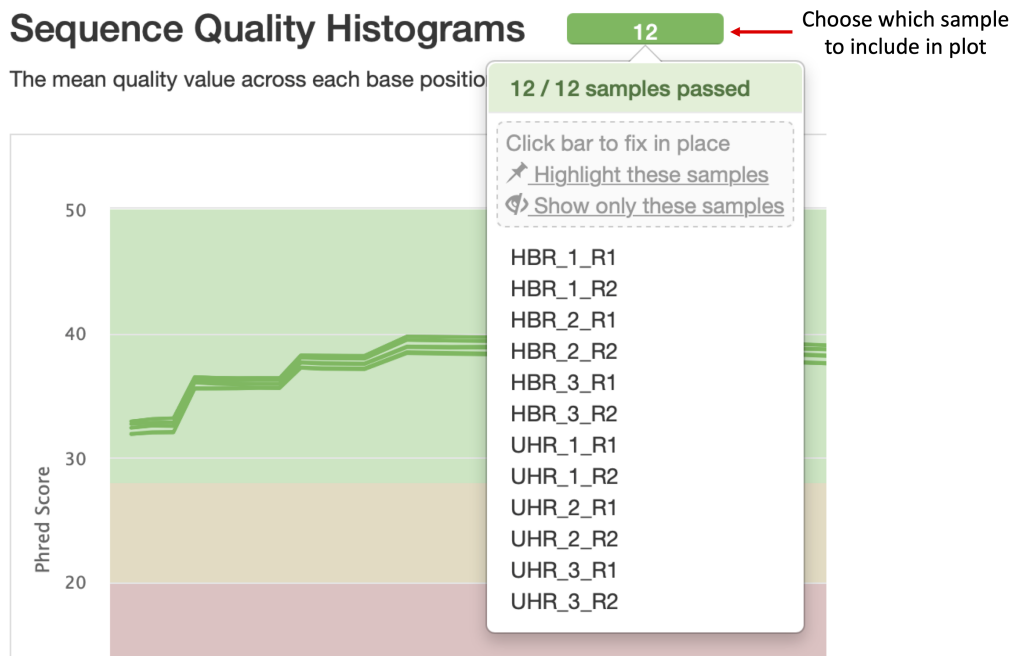


Figure 6

In Figure 7, we see the quality score distribution of each of our FASTQ files.

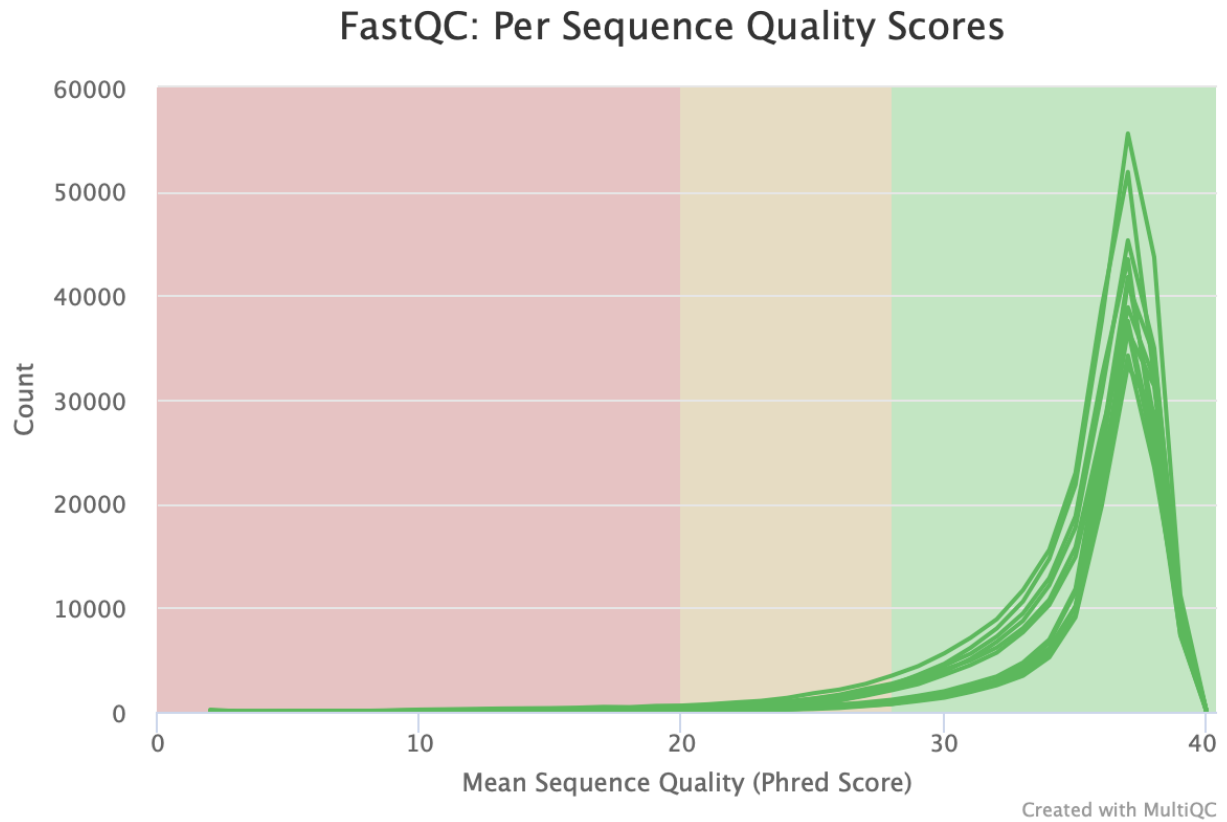


Figure 7

In Figure 8, we get an interactive heatmap of the percent composition of each nucleotide base (A,T,C,G) along the bases (horizontal axis) for each of the FASTQ files (vertical axis). If we hover over a tile, we will see the corresponding numbers. If we click on any row in this heatmap, we will get the base composition plot for just that sample (Figure 9).

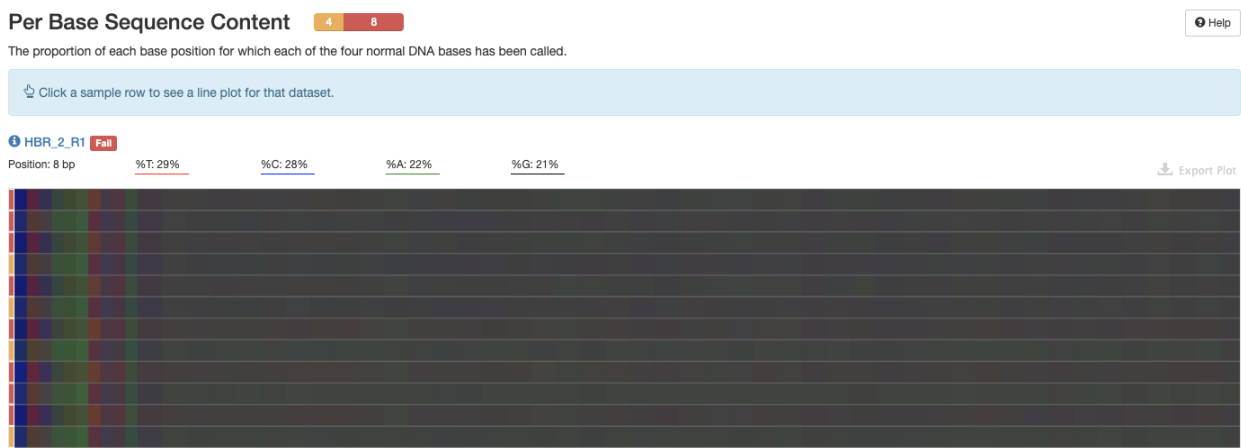


Figure 8

Per Base Sequence Content

4 8

Help

The proportion of each base position for which each of the four normal DNA bases has been called.

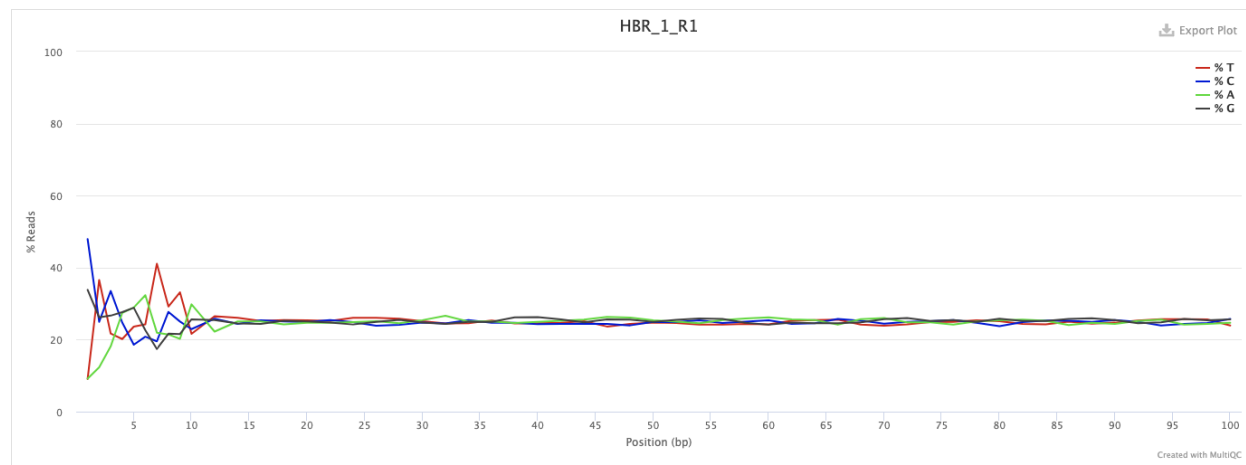
[Back to overview heatmap](#) « Prev Next »

Figure 9

The GC distribution for each of the FASTQ files is shown in Figure 10.

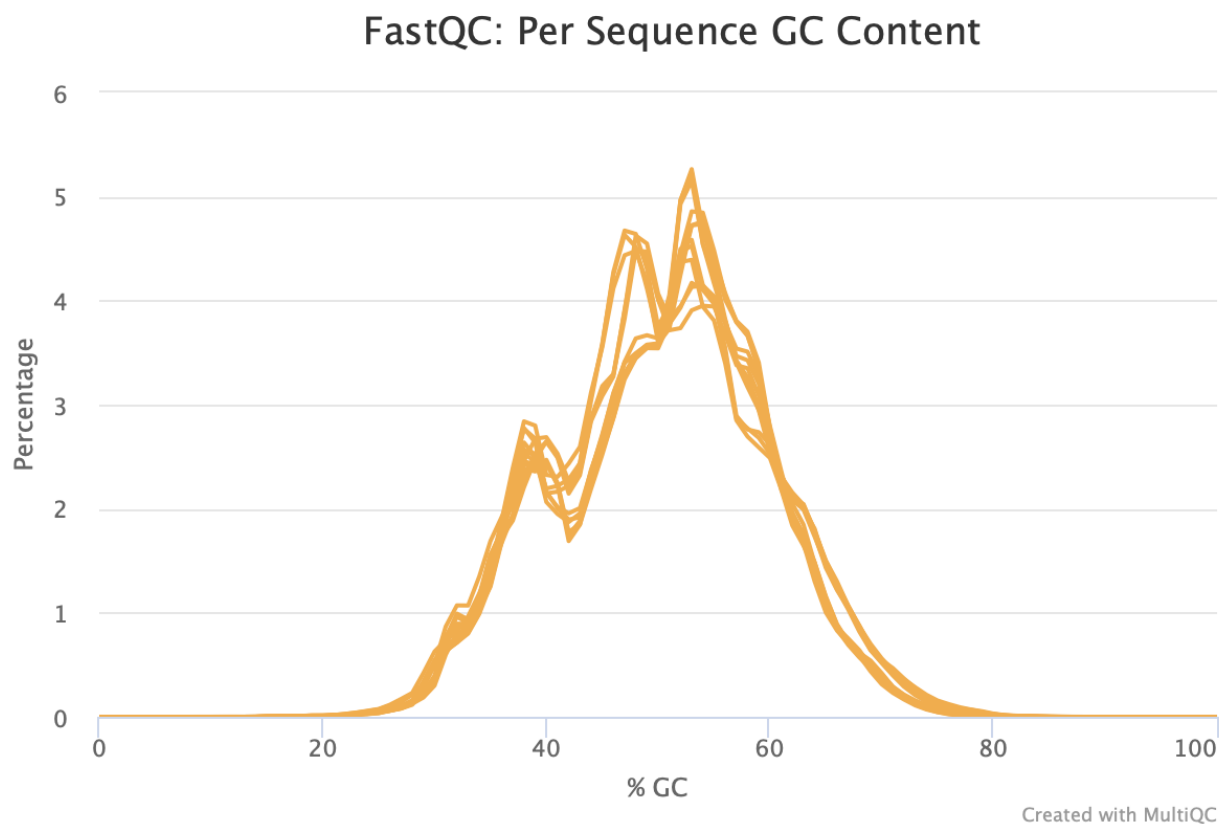


Figure 10

Figure 11 tells us that except for a few bases at the beginning of the read, we do not have unknown bases in our FASTQ files.

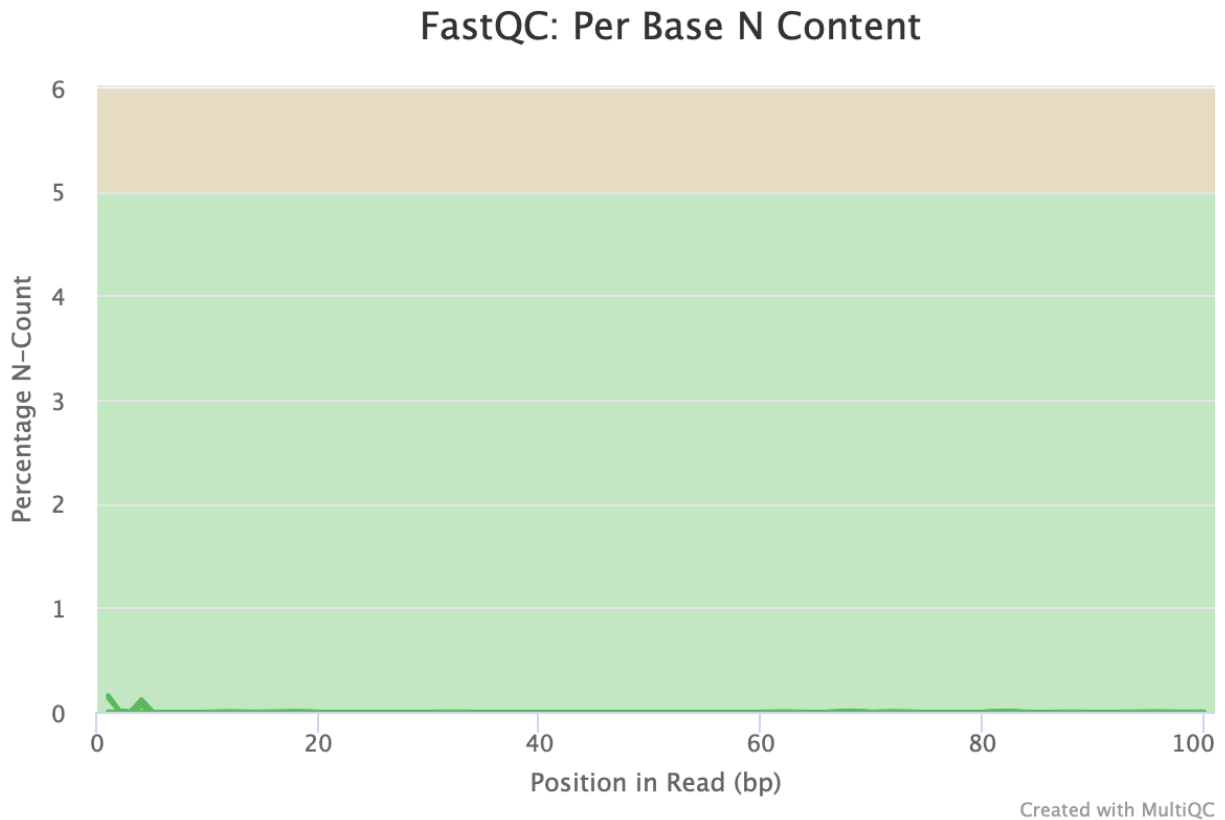


Figure 11

Figure 12 tells us that all of the reads in our FASTQ files have 100 bases, so no problems there.

Sequence Length Distribution

12

All samples have sequences of a single length (100bp).

Figure 12

We see the sequence duplication levels, overrepresented sequences, and adapter content information in Figures 13 through 15.

FastQC: Sequence Duplication Levels

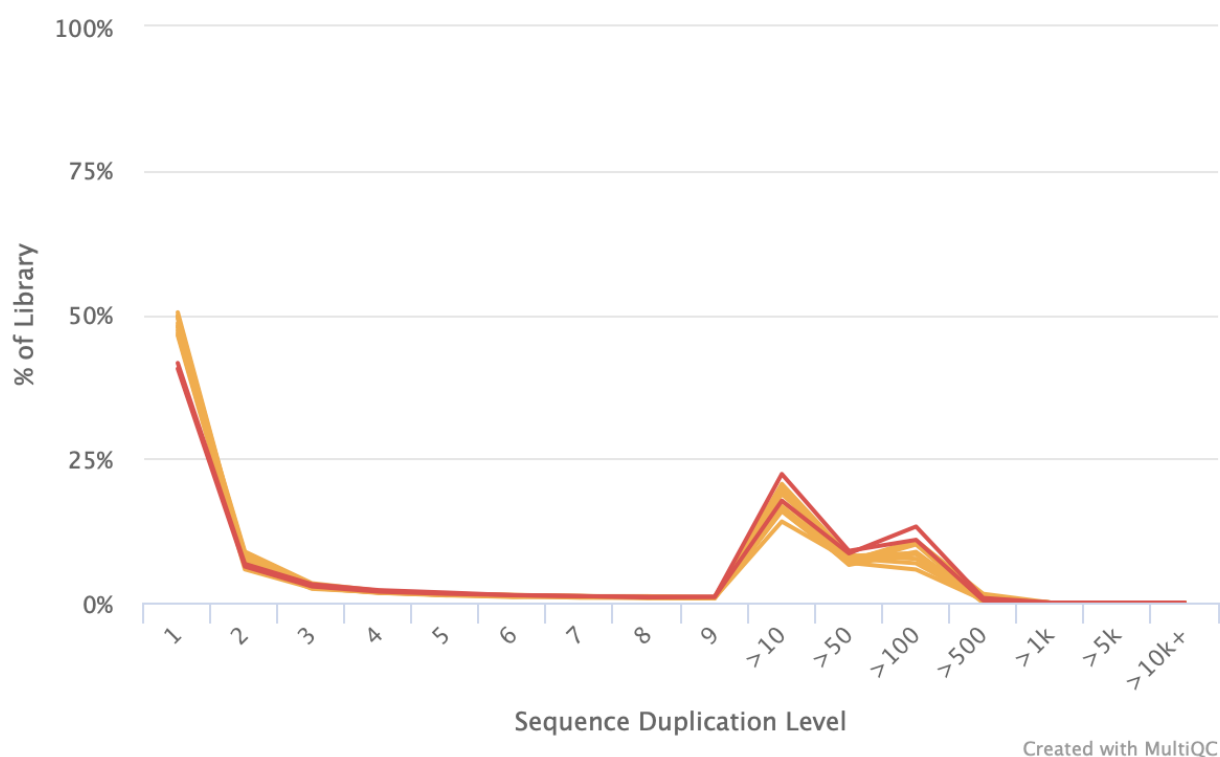


Figure 13

FastQC: Overrepresented sequences

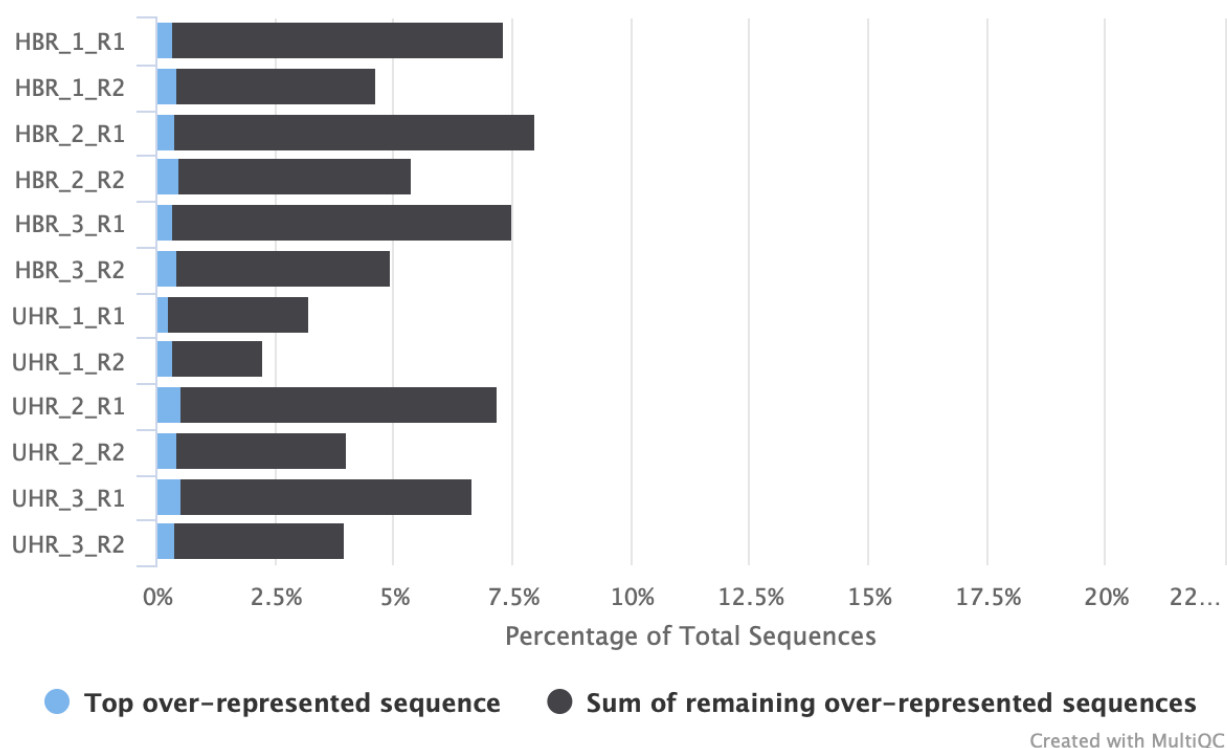


Figure 14

Adapter Content

12

The cumulative percentage count of the proportion of your library which has seen each of the adapter sequences at each position.

No samples found with any adapter contamination > 0.1%

Figure 15

At the end of the MultiQC report, we see a heatmap of the modules that have passed, warn, or failed status for each of the FASTQ files.

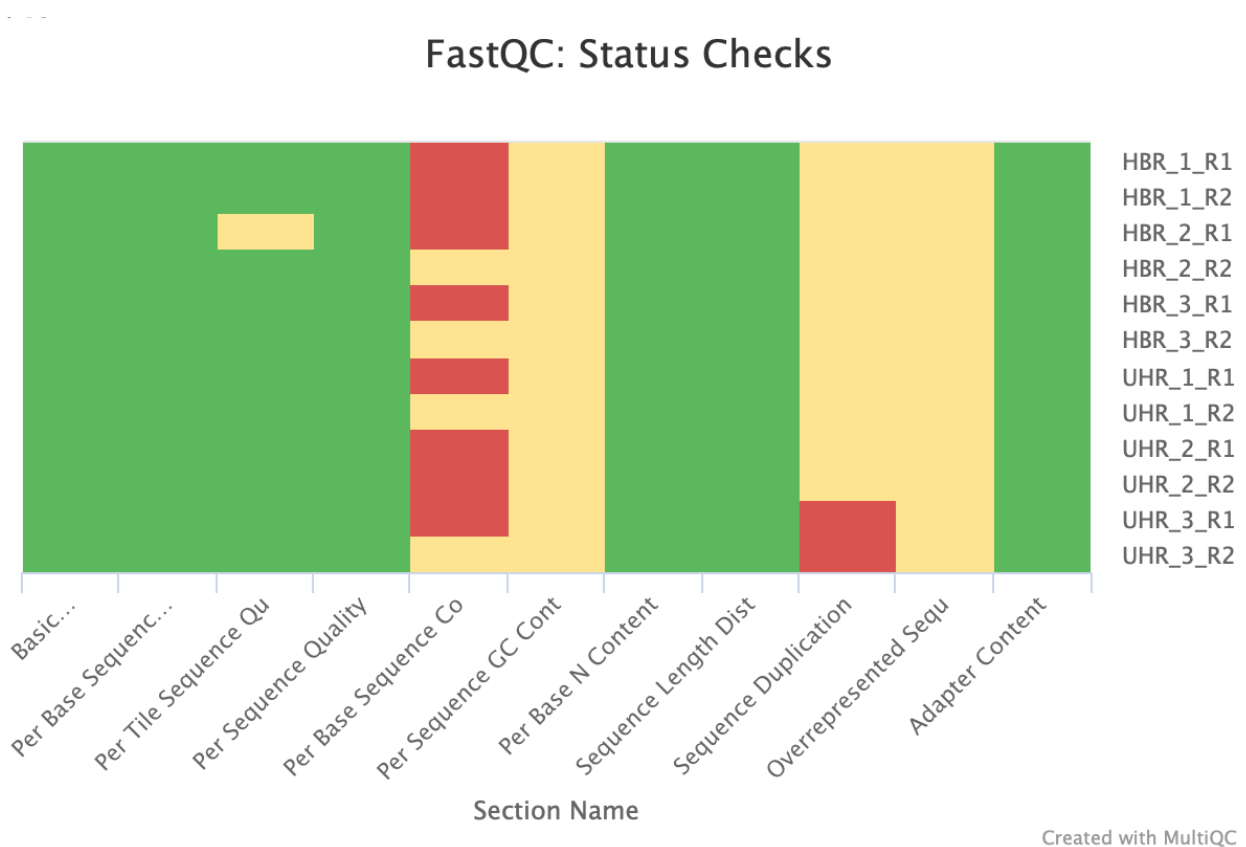


Figure 16

Trimming away adapters and poor quality reads

We will be downloading a FASTQ file from the Sequence Read Archive to learn about trimming. But first, go back to the `~/biostar_class` folder and then create a new directory named `trimming`.

```
cd ~/biostar_class
```

```
mkdir trimming
```

```
cd trimming
```

Let's now download the FASTQ files for SRR1553606, which was sequenced under the paired end format, so we will need to specify `--split-file` to separate read 1 and read 2. We specify `-X 10000` to retrieve only 10000 reads, otherwise the download will take longer.

```
fastq-dump --split-files -X 10000 SRR1553606
```

shows this output

```
Read 10000 spots for SRR1553606  
Written 10000 spots for SRR1553606
```

```
ls
```

```
SRR1553606_1.fastq    SRR1553606_2.fastq
```

Let's run FASTQC for both read 1 and read 2 of SRR1553606. We can use the wildcard (*) to get both files rather than inputting them separately.

```
fastqc SRR1553606_*.fastq
```

We can copy the FASTQC reports for SRR1553606 to the `~/public` directory to review them.

```
cp SRR1553606_1_fastqc.html ~/public
```

```
cp SRR1553606_2_fastqc.html ~/public
```

Initial QC of both FASTQ files for SRR1553606 shows failures and warnings. Figures 17 through 20 shows the per base sequence quality and adapter content for SRR1553606 read 1 and read 2. Here let's focus on removing adapters and poor quality reads. Adapters in particular will interfere with the alignment step. At the end of the reads in SRR1553606_2, we see that the 25th

- 75th percentile (the yellow box) of scores span a large range, with many of them in the orange and red regions, where reliability of the reads would come into question (Figure 20).

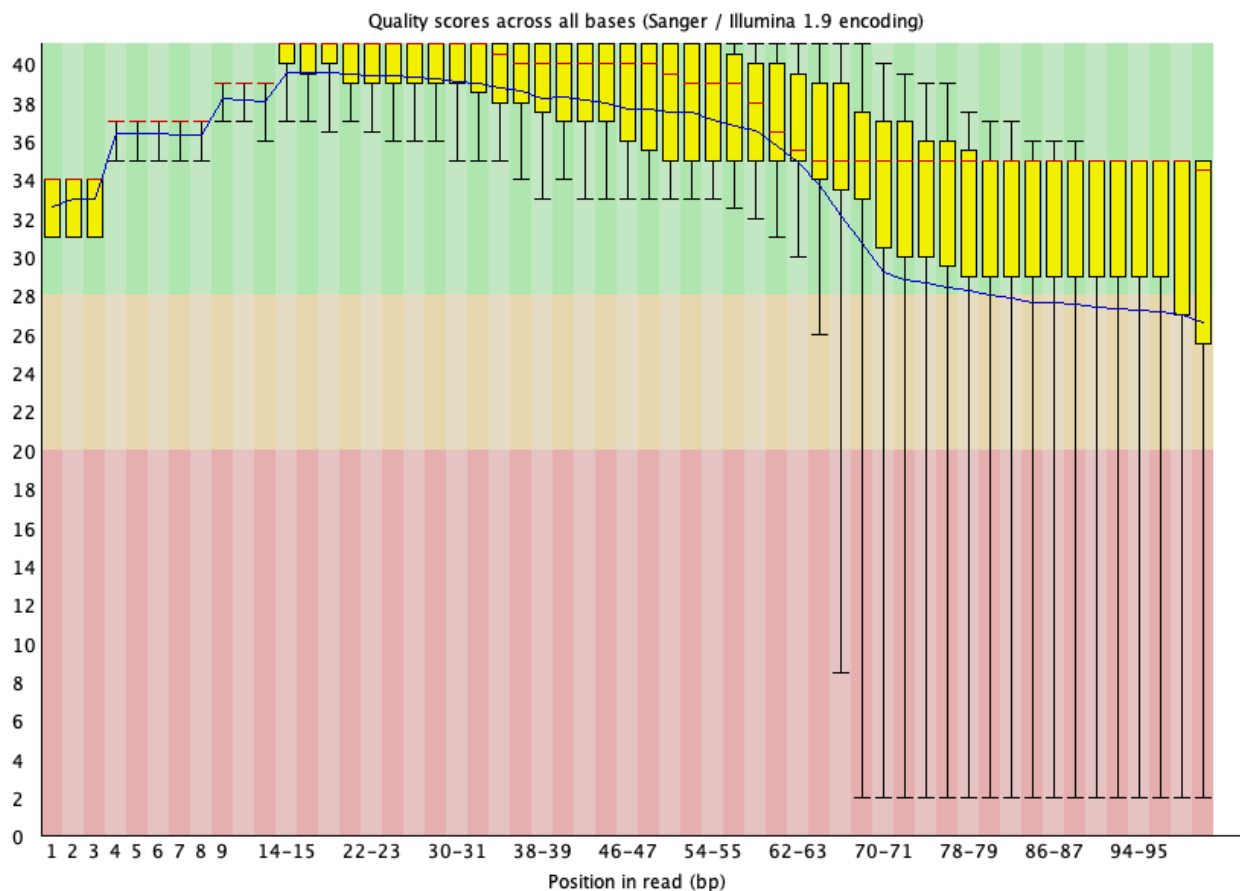


Figure 17: Per base quality for SRR1553606_1.fastq

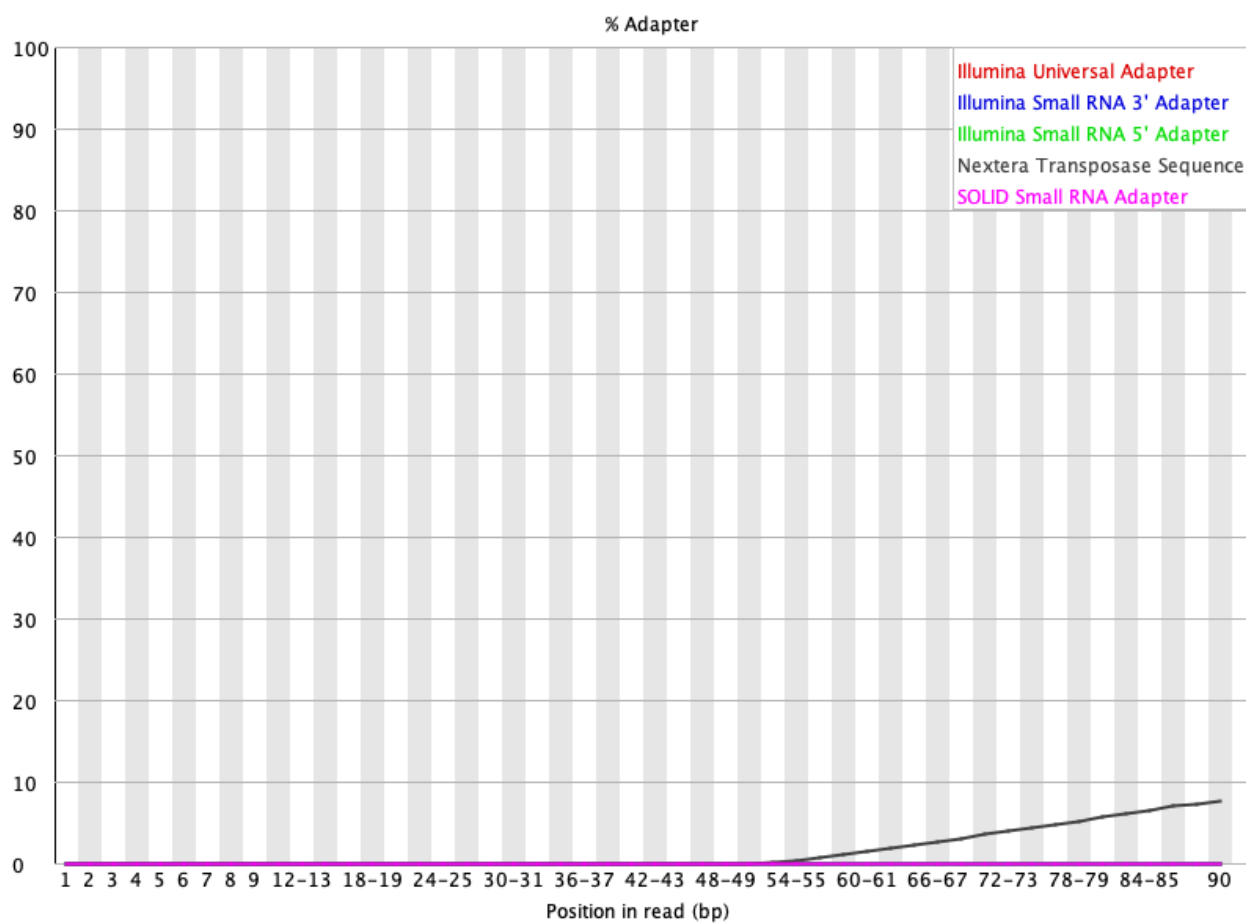


Figure 18: Adapter content for SRR1553606_1.fastq

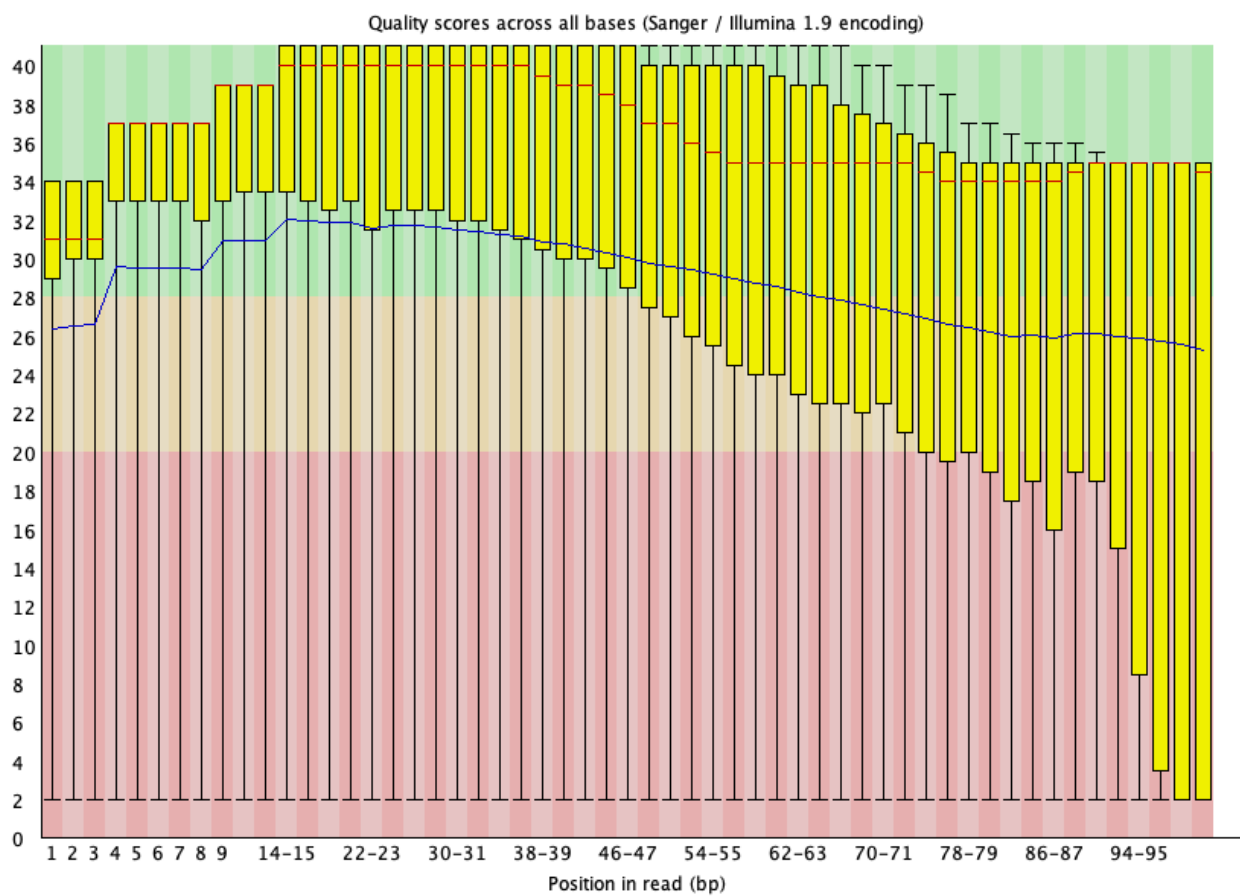


Figure 19: Per base quality for SRR1553606_2.fastq

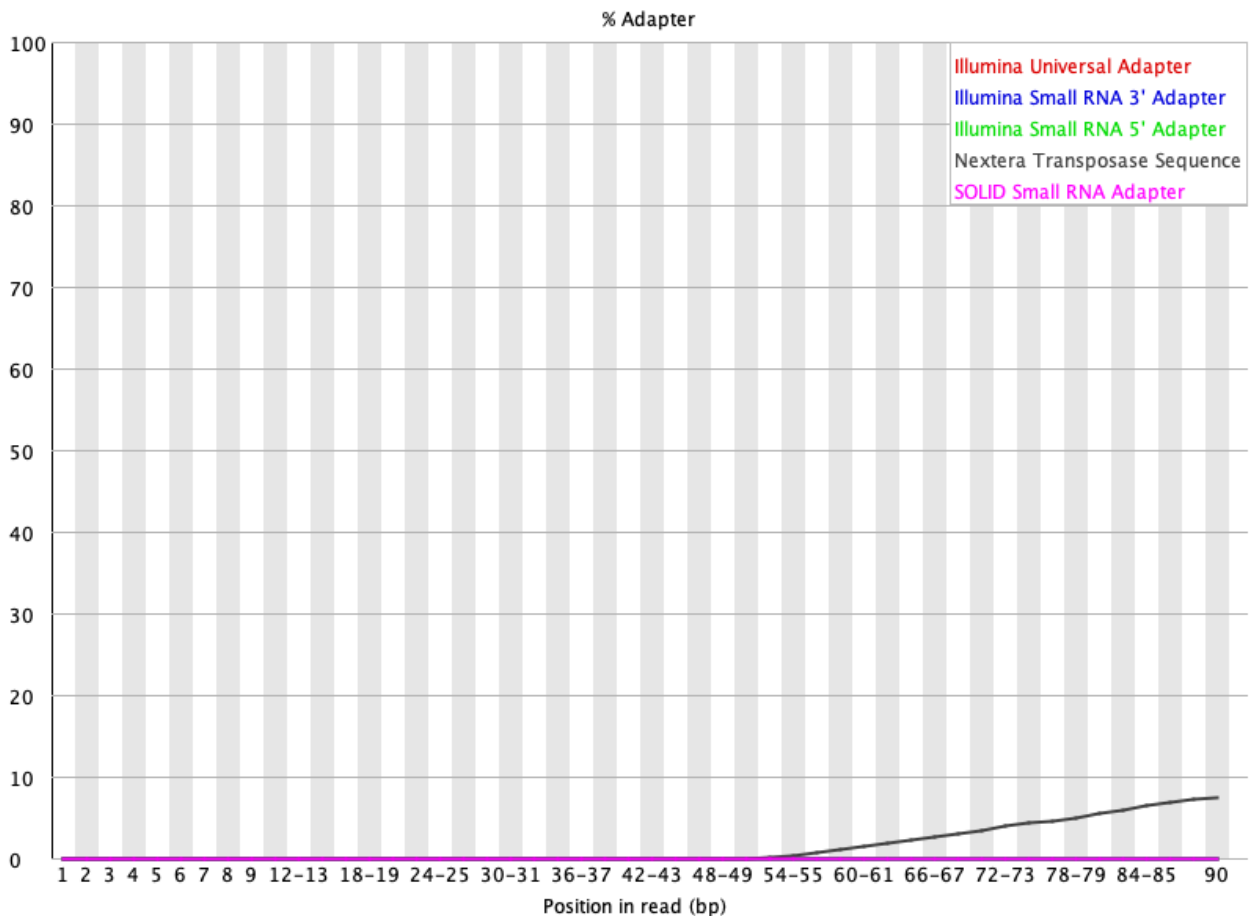


Figure 20: Adapter content for SRR1553606_2.fastq

Trimming with Trimmomatic

Let's use the tool Trimmomatic to clean up the adapters and the poor quality reads for SRR1553606. For help with Trimmomatic type `trimmomatic --help` at the command line.

Before getting started with using `trimmomatic`, let's create a file called `nextera.fa` which houses the nextera adapter sequence that we need to remove (from the FASTQC result, we have Nextera adapter contamination).

The command below will create a file called `nextera.fa` and open it in the nano editor. We can then copy and paste the sequence, then hit `control-x` to save and exit the editor.

```
nano nextera.fa
```

```
>nextera
CTGTCTTATACACATCTCCGAGCCCACGAGAC
```

We initiate the application by typing `trimmomatic` at the command line and the parameters are explained below.

- PE stands for paired end mode. We are dealing with sequencing data derived from paired end library preparation so we can use this option. If we have single end sequencing data then we can replace PE with SE. After specifying paired end (PE) mode
 - The files for read 1 and read 2 are entered
 - Following that, we enter the names of the trimmed FASTQ files (`SRR1553606_trimmed_1.fastq` and `SRR1553606_trimmed_2.fastq`). We need to specify two because we have two input files for paired end sequencing
 - Note that we also specify a file name for unpaired reads (`SRR1553606_trimmed_1_unpaired.fastq` and `SRR1553606_trimmed_2_unpaired.fastq`). Sometimes a read in one file may be successfully processed while the same read will not be successfully processed in the second file, thus, we place these reads in a separate file.
- The next portion to the `trimmomatic` command allows us to specify the quality score criteria for trimming. Here we use a sliding window (SLIDINGWINDOW), which scans the 5' end of the read and removes when the average quality of the window falls below a threshold.
 - We choose a window size of 4 reads
 - Quality threshold of 30
 - The final construction is `SLIDINGWINDOW:4:30`
- We then use the `ILLUMINACLIP` flag to specify the file to our adapter sequence where the numbers (2:30:5) that follows sets the criteria on how Trimmomatic would determine whether a portion of the read matches the adapter (see the Trimmomatic manual at http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf) for more).
- In the `MINLEN` argument, we specify 50 and Trimmomatic will remove reads that are less than 50 bases. We set this threshold because shorter reads will be difficult to map because they would potentially fall onto multiple regions of the genome.

```
trimmomatic PE SRR1553606_1.fastq SRR1553606_2.fastq SRR1553606_trimr
```

Trimmomatic will display the following as it runs.

```
TrimmomaticPE: Started with arguments:
  SRR1553606_1.fastq SRR1553606_2.fastq SRR1553606_trimmed_1.fastq SRR1553606_trimmed_2.fastq
Using Long Clipping Sequence: 'CTGTCTCTTATACATCTCCGAGCCCACGAGAC'
ILLUMINACLIP: Using 0 prefix pairs, 1 forward/reverse sequences, 0 for reverse
Quality encoding detected as phred33
Input Read Pairs: 10000 Both Surviving: 6301 (63.01%) Forward Only Surviving: 6301
TrimmomaticPE: Completed successfully
```

In the ls command below, we place * (wildcard) around trimmed to tell ls that we want any file with the word trimmed in it. We use * before and after so that ls will know there could be characters before trimmed and also after.

```
ls *trimmed*
```

```
SRR1553606_trimmed_1.fastq  
SRR1553606_trimmed_1_unpaired.fastq  
SRR1553606_trimmed_2.fastq  
SRR1553606_trimmed_2_unpaired.fastq
```

Run FASTQC on the trimmed FASTQ files.

```
fastqc SRR1553606_trimmed_1.fastq SRR1553606_trimmed_2.fastq
```

Copy FASTQC reports for the SRR1553606 trimmed data to the ~/public folder to view.

```
cp SRR1553606_trimmed_1_fastqc.html ~/public
```

```
cp SRR1553606_trimmed_2_fastqc.html ~/public
```

Our per base quality looks much better and adapters were removed after trimming using Trimmomatic. Note in the basic statistics portion of the FASTQC report for the trimmed files, we loss around 40% of the reads from original. Therefore, trimming is a balancing act between removing unwanted reads and keeping as much of the original information as possible to prevent our experiment from becoming a wasted effort.

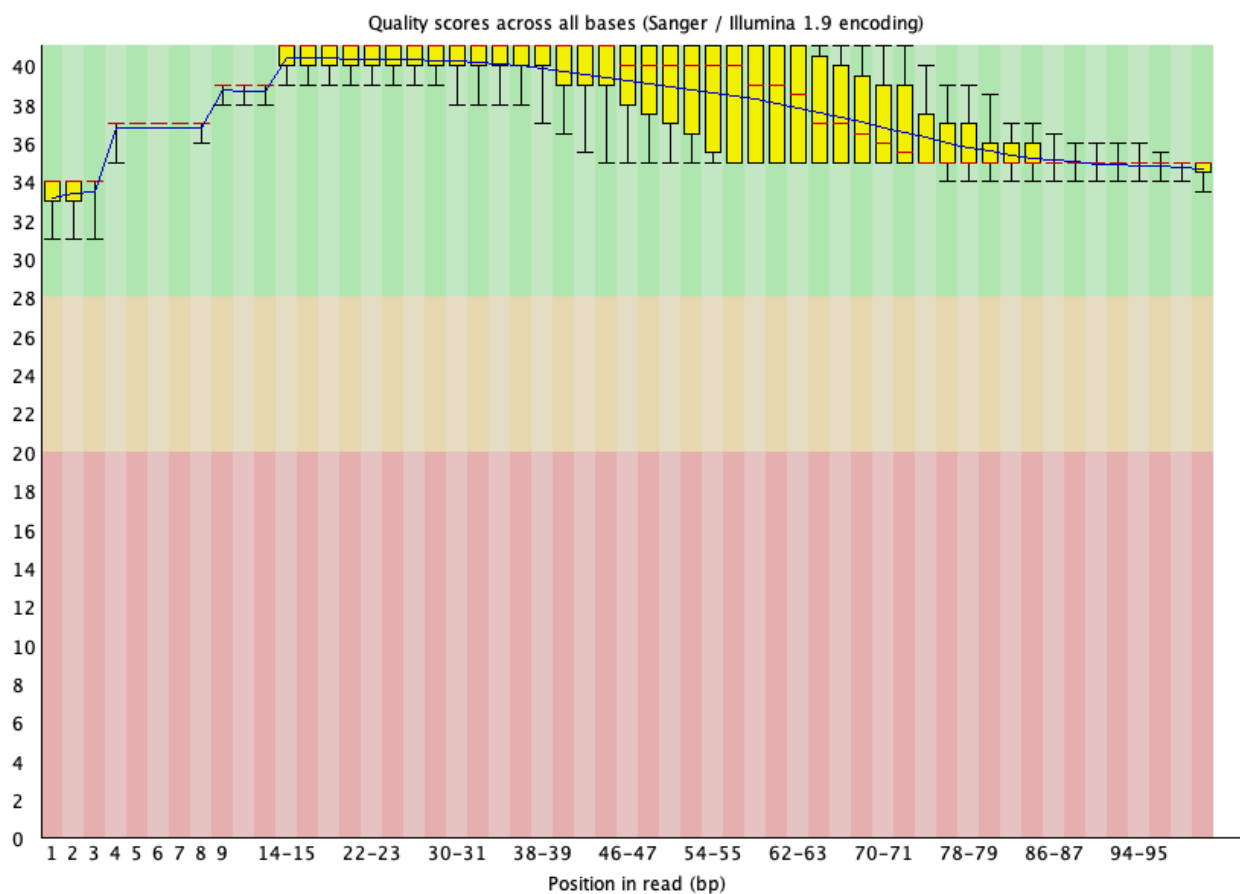


Figure 21: Per base quality for SRR1553606_1 after Trimmomatic trimming.

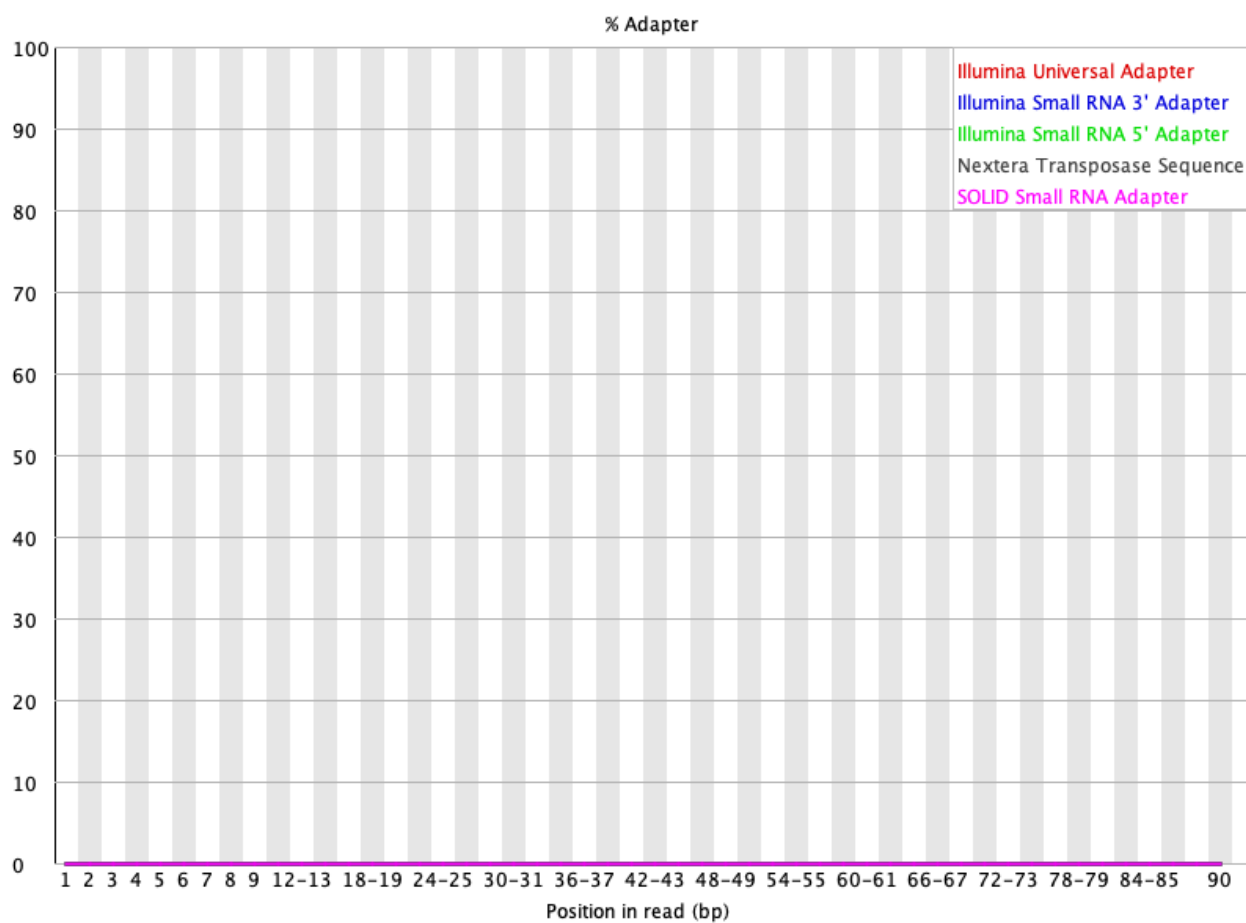


Figure 22: Adapter content for SRR1553606_1 after Trimmomatic trimming.

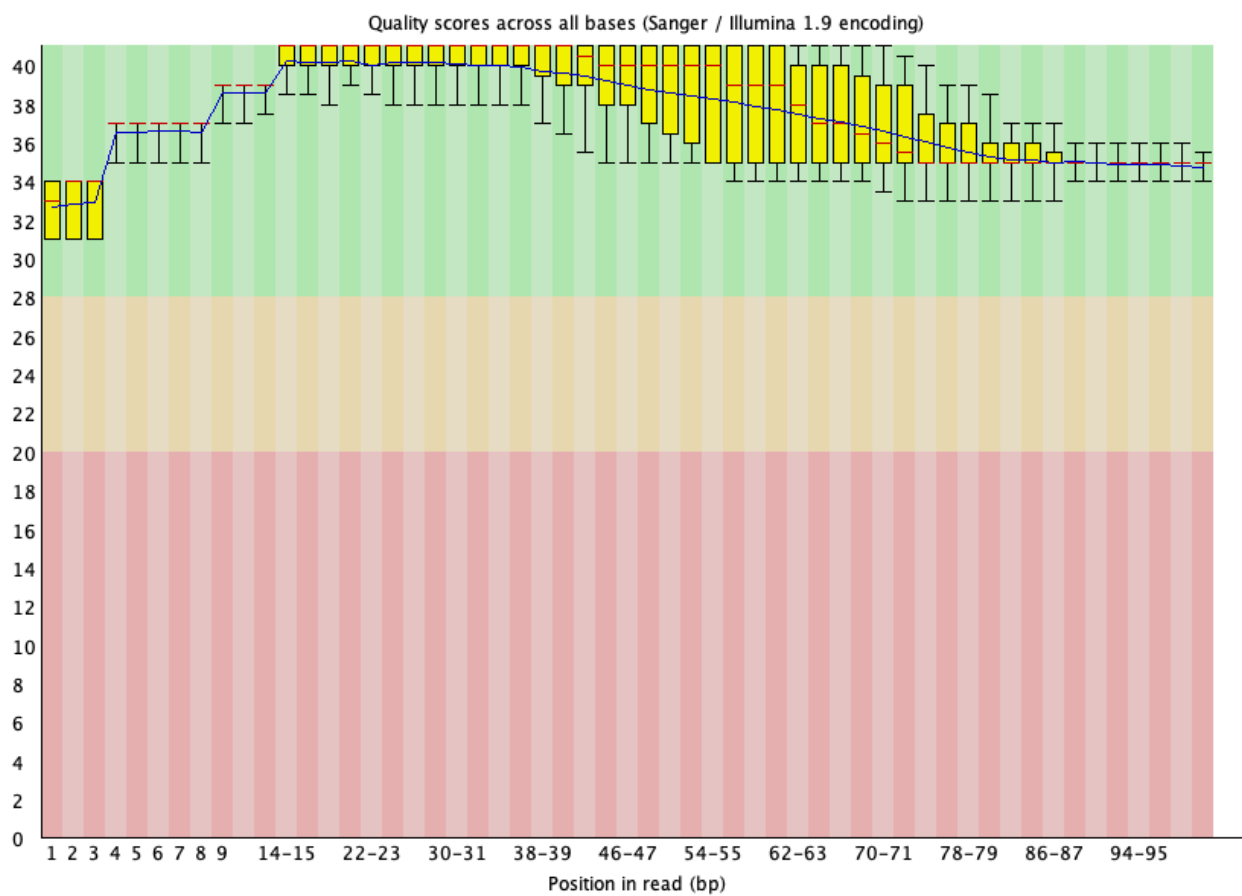


Figure 23: Per base quality for SRR1553606_2 after Trimmomatic trimming.

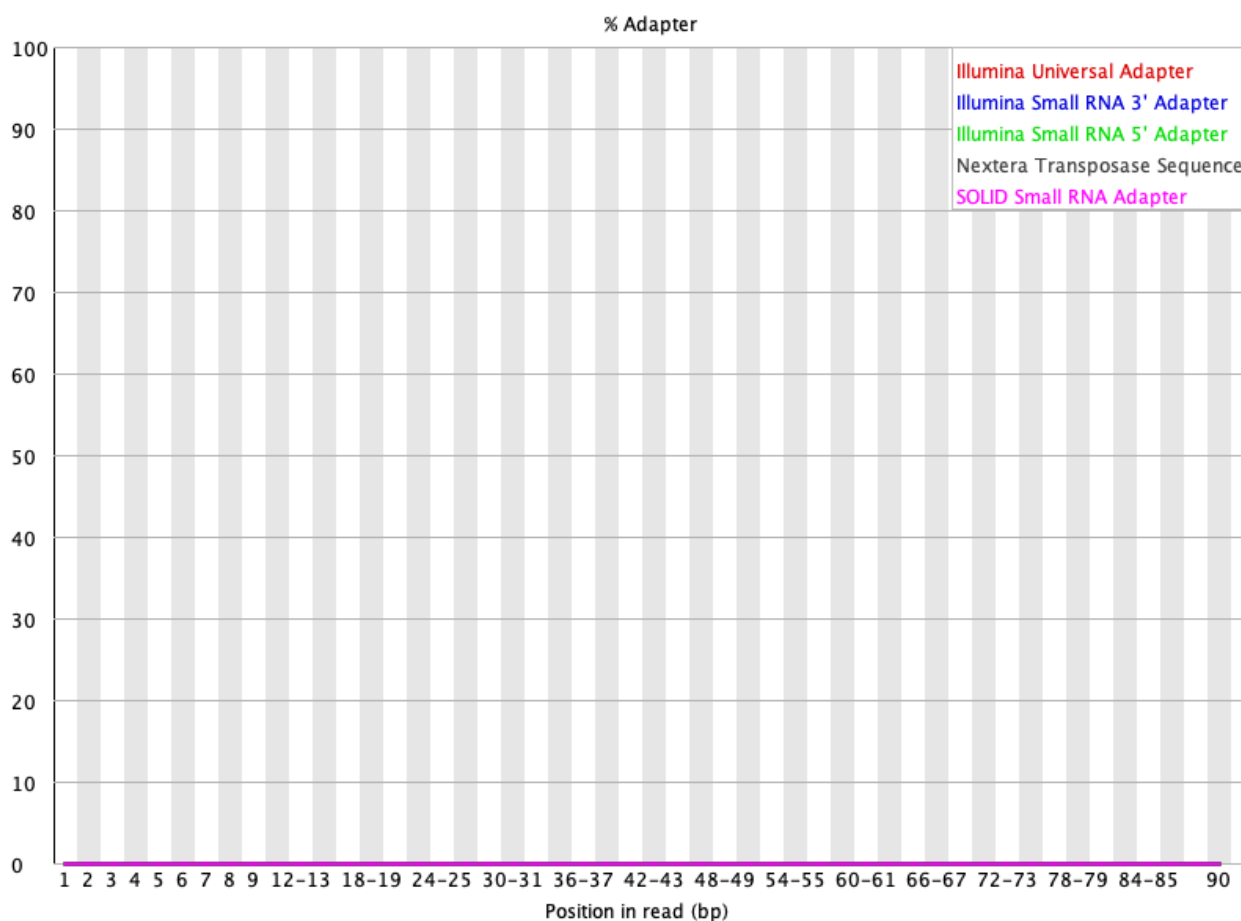


Figure 24: Adapter content for SRR1553606_2 after Trimmomatic trimming.

Trimming with BBDuk

BBDuk is another tool that can be used for adapter and quality trimming. In addition, BBDuk can be used to filter out contaminations, perform GC filtering, filter for length, etc. (see <https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbdduk-guide/> (<https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbdduk-guide/>)).

Let's run BBDuk to do the same adapter and quality trim as we did with Trimmomatic for the FASTQ files in SRR1553606.

To initiate BBDuk we type `bbduk.sh` at the command prompt

- Again, we are working with paired end sequencing so we provide read 1 after the "in=" argument and then read 2 after "in2=". Similarly, we specify the output file names for read 1 and read 2 after the "out=" and "out2=" arguments, respectively.
- The `qtrim` argument tells BBDuk we want to perform quality trimming. Setting `qtrim=r` means that BBDuk will trim from the right side of the sequence. We specify the quality threshold for trimming to 30 using the `trimq` argument.
- For adapter trimming, we specify the adapter sequence FASTA file (again, we will be using `nextera.fa` created earlier). Note that for adapter trimming, we use the `ktrim` option,

which essentially tells BBDuk to trim based on sequence matching rather than quality. Here, we set `ktrim=r` so that BBDuk will trim away bases to the right of the match. The parameters that follow `ktrim` are criteria that determine whether a portion of the sequencing read matches the adapter.

- See [BBDuk manual \(https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbdduk-guide/\)](https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/bbdduk-guide/) for more about arguments and parameters that can be used with this program.

```
bbduk.sh in=SRR1553606_1.fastq in2=SRR1553606_2.fastq out=SRR1553606_
```

As BBDuk is running, we see statistics such as the number of reads that are quality and/or adapter trimmed. We also see the number of reads that have been removed and the number of reads that remain.

```
Input is being processed as paired Started output streams: 0.005 seconds.  
Processing time: 0.148 seconds.
```

```
Input: 20000 reads 2020000 bases. QTrimmed: 4137 reads (20.69%) 276896 bases  
(13.>71%) KTrimmed: 5632 reads (28.16%) 448982 bases (22.>23%) Total  
Removed: 6356 reads (31.78%) 725878 bases (35.>93%) Result: 13644 reads  
(68.22%) 1294122 bases >(64.07%)
```

```
Time: 0.172 seconds. Reads Processed: 20000 116.38k reads/sec Bases  
Processed: 2020k 11.75m bases/sec
```

Running FASTQC on the BBDuk trimmed output, we see that BBDuk performs similar to Trimmomatic.

```
fastqc SRR1553606_bbdduk_1.fastq SRR1553606_bbdduk_2.fastq
```

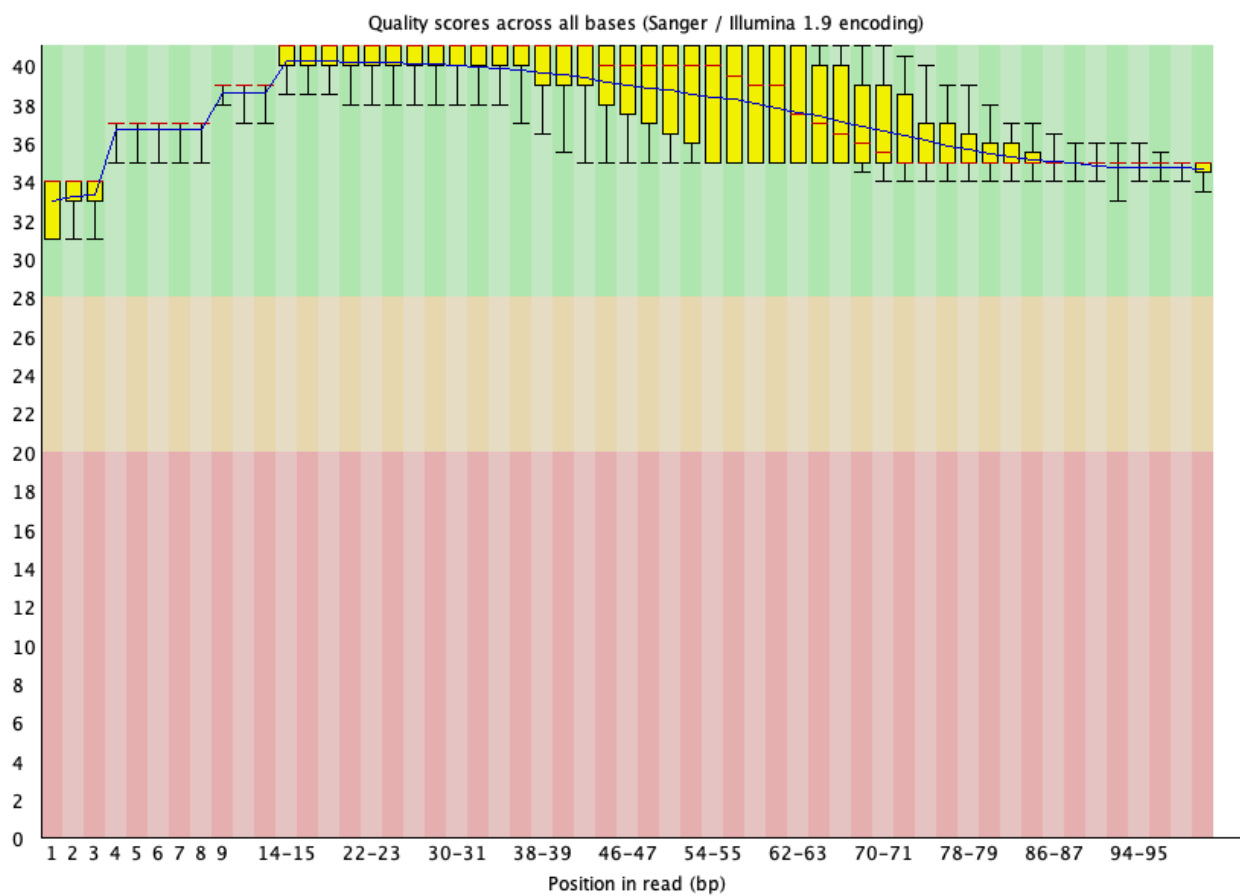


Figure 25: SRR1553606_1 per base quality after BBDuk trimming

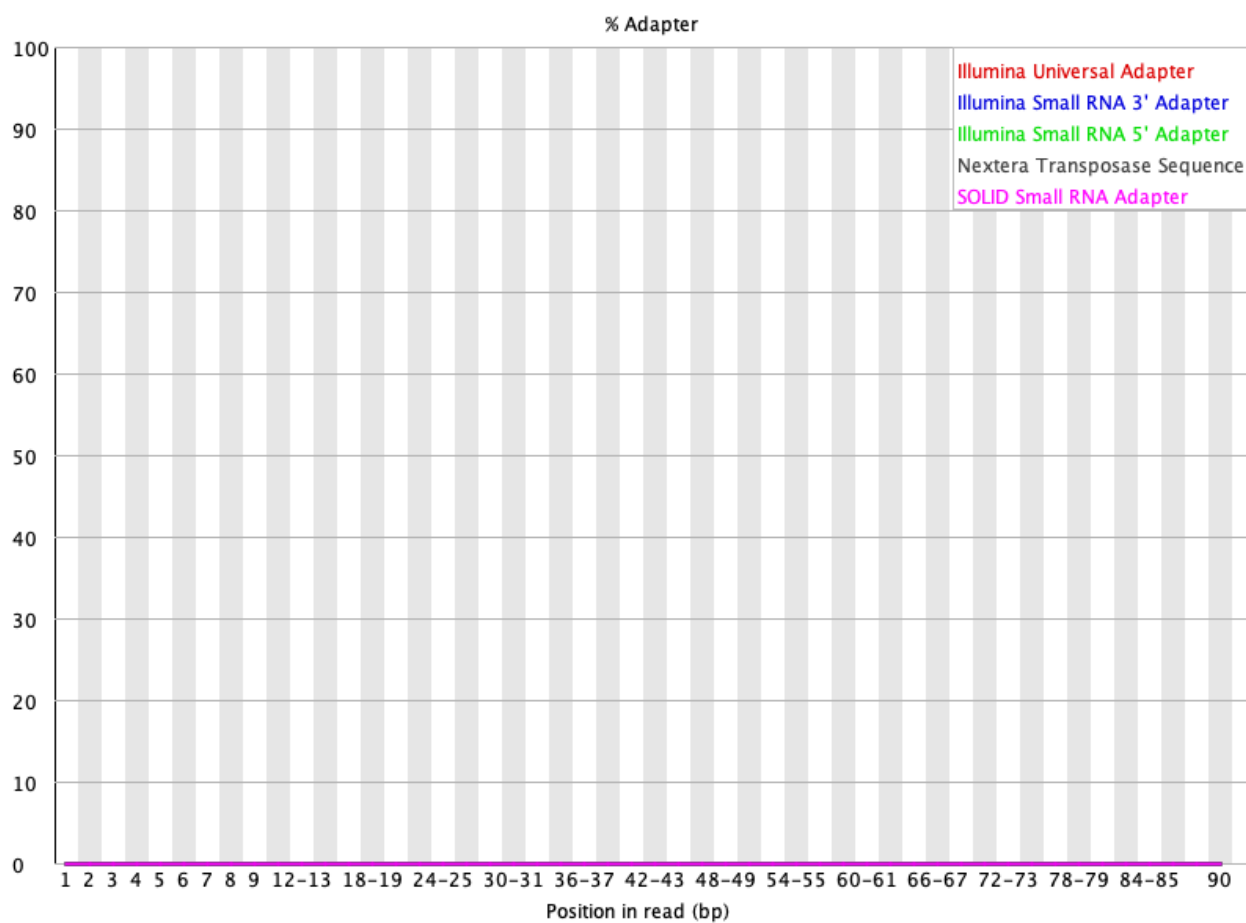


Figure 26: SRR1553606_1 adapter content after BBDuk trimming

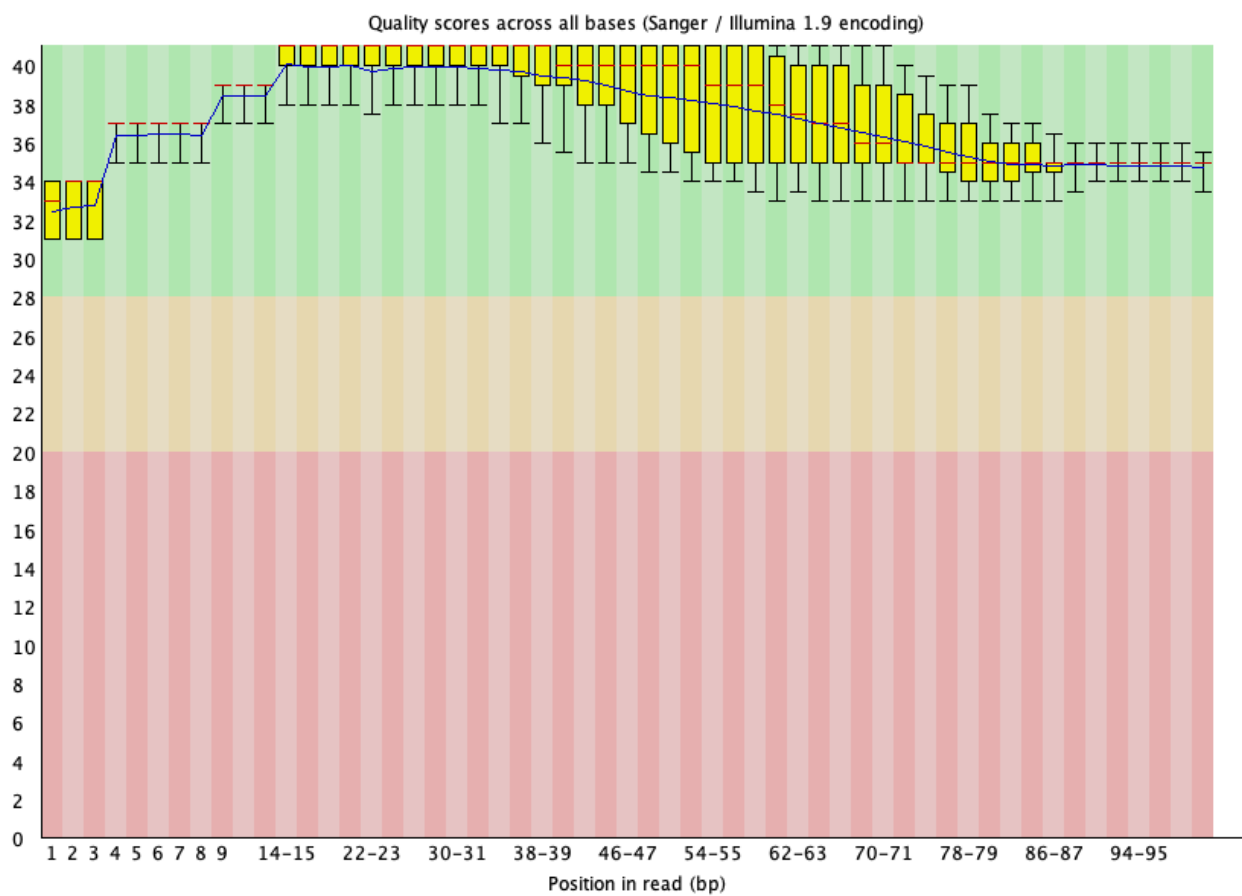


Figure 27: SRR1553606_2 per base quality after BBDuk trimming

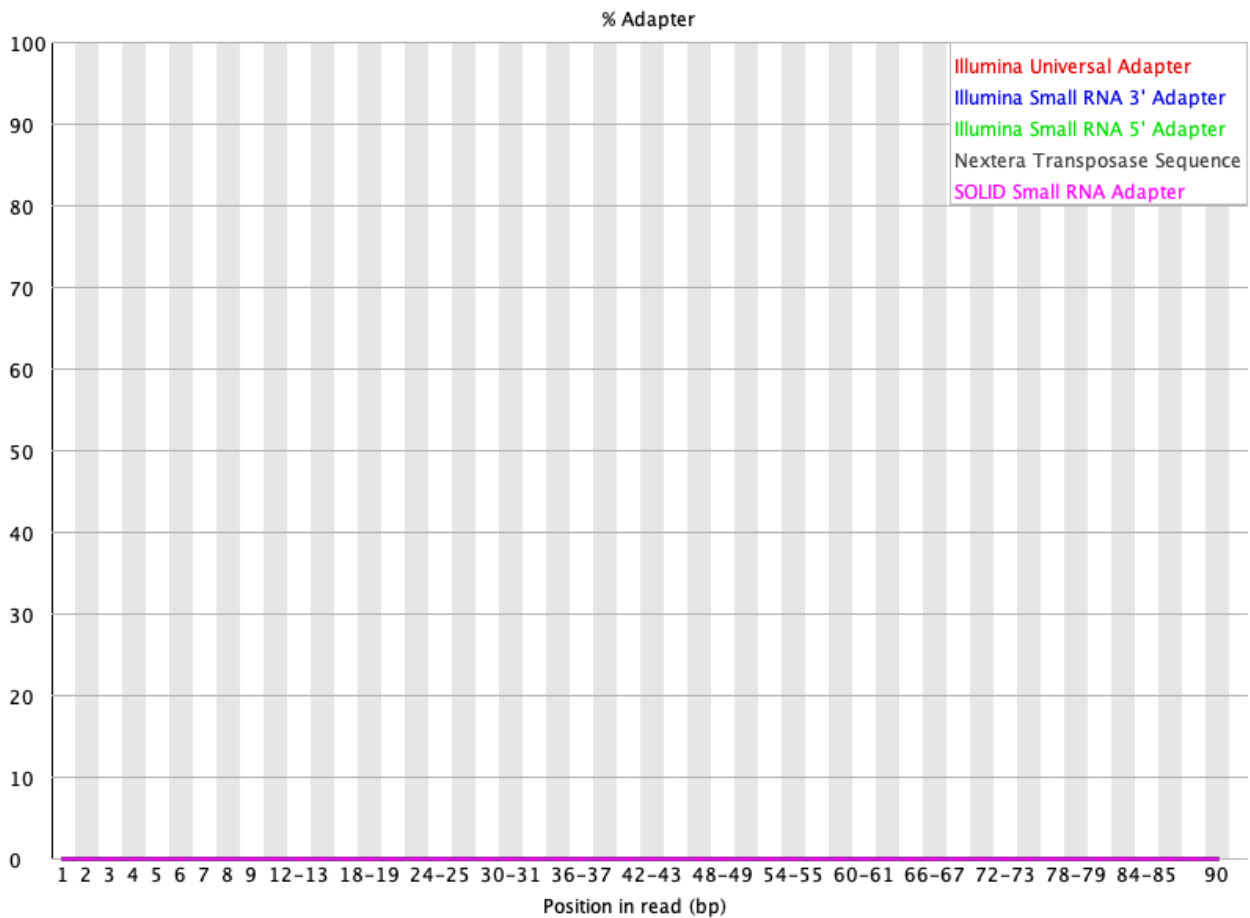


Figure 28: SRR1553606_2 adapter content after BBDuk trimming

MultiQC report for HBR and UHR dataset

HBR and UHR MultiQC report

FASTQC reports for SRR1553606

SRR1553606_1.fastq not trimmed

SRR1553606_2.fastq not trimmed

SRR1553606_1.fastq trimmed with Trimmomatic

SRR1553606_2.fastq trimmed with Trimmomatic

SRR1553606_1.fastq trimmed with BBDuk

SRR1553606_2.fastq trimmed with BBDuk

SRR1553606 MultiQC report - aggregate of untrimmed, Trimmomatic trimmed, and BBDuk trimmed results

Lesson 12: RNA sequencing review 1

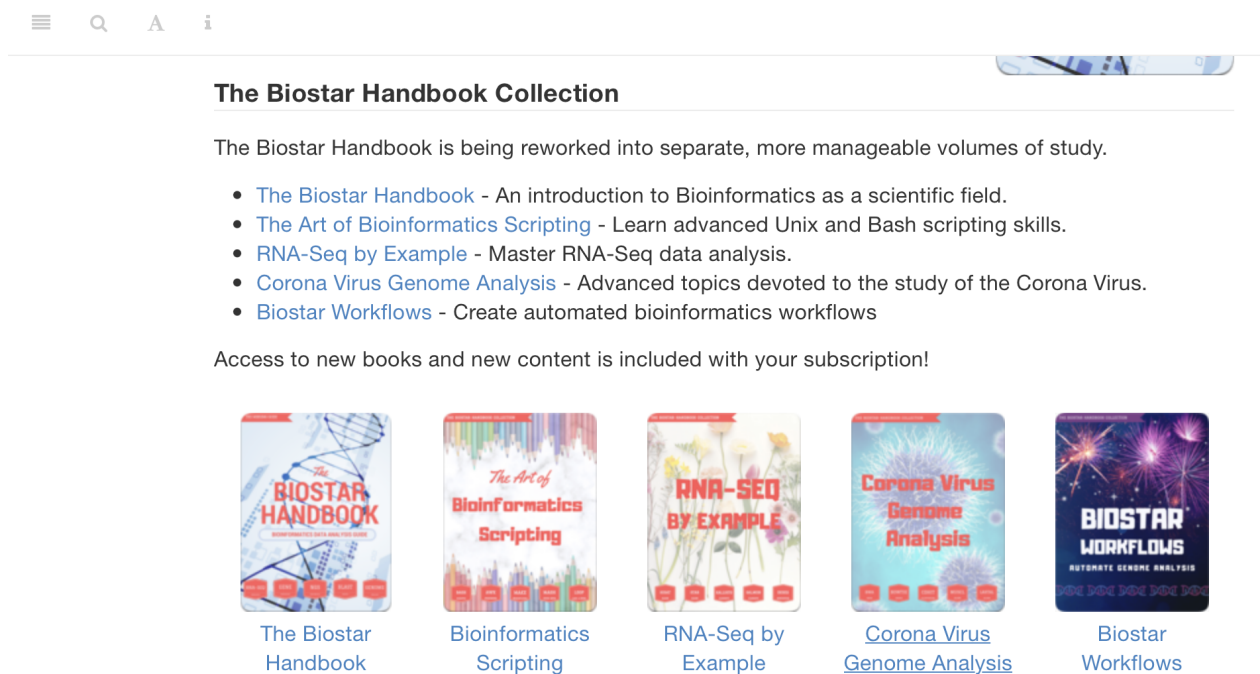
Learning objectives

Here, we will do a quick review of what we have learned about RNA sequencing in Lessons 8 through 11.

Accessing the Biostar handbook

The URL for the Biostar handbook is <https://www.biostarhandbook.com> (<https://www.biostarhandbook.com>).

Once you sign into this handbook, you will find that it is composed of several different books including one for RNA sequencing.



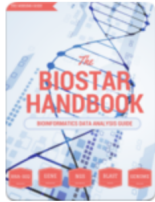
The screenshot shows the 'The Biostar Handbook Collection' page. At the top, there is a navigation bar with icons for a menu, search, and user profile. Below the navigation bar, the title 'The Biostar Handbook Collection' is displayed. A paragraph states: 'The Biostar Handbook is being reworked into separate, more manageable volumes of study.' This is followed by a bulleted list of five books: 'The Biostar Handbook - An introduction to Bioinformatics as a scientific field.', 'The Art of Bioinformatics Scripting - Learn advanced Unix and Bash scripting skills.', 'RNA-Seq by Example - Master RNA-Seq data analysis.', 'Corona Virus Genome Analysis - Advanced topics devoted to the study of the Corona Virus.', and 'Biostar Workflows - Create automated bioinformatics workflows'. Below the list, a note says 'Access to new books and new content is included with your subscription!'. At the bottom, five book covers are shown in a row, each with a caption below it: 'The Biostar Handbook', 'Bioinformatics Scripting', 'RNA-Seq by Example', 'Corona Virus Genome Analysis', and 'Biostar Workflows'.

The Biostar Handbook Collection


The Biostar Handbook is being reworked into separate, more manageable volumes of study.

- [The Biostar Handbook](#) - An introduction to Bioinformatics as a scientific field.
- [The Art of Bioinformatics Scripting](#) - Learn advanced Unix and Bash scripting skills.
- [RNA-Seq by Example](#) - Master RNA-Seq data analysis.
- [Corona Virus Genome Analysis](#) - Advanced topics devoted to the study of the Corona Virus.
- [Biostar Workflows](#) - Create automated bioinformatics workflows


Access to new books and new content is included with your subscription!




[The Biostar Handbook](#)



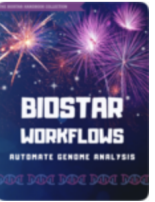
[Bioinformatics Scripting](#)



[RNA-Seq by Example](#)



[Corona Virus Genome Analysis](#)



[Biostar Workflows](#)

Scroll to the bottom of the page and you will find a button that says Access Your Account. Click this to sign in.

Manage your account

If you are logged in you may manage your account by clicking the link button below.

[Access Your Account](#)

You may also change your email or log out via this page.

Welcome to the Biostar Handbook

Please enter your email and password below.

Log in to access the book.

Email

Password

Log In **Cancel**

[Forgot your password?](#)

Because the Biostars handbook subscription is only good for 6 months, we recommend that you download either the PDF or eBook.

How to download the book



The book is available to registered users. The latest versions can be downloaded from:

- [RNA-Seq by Example, PDF](#)
- [RNA-Seq by Example, eBook](#)

Our books are updated frequently. We recommend accessing each book via the website as the web version will always contain the most recent and up-to-date content. A few times a year we send out emails that describe the new additions.

Review of RNA sequencing concepts

- Purpose of RNA sequencing and what biological questions can RNA sequencing answer
- Experimental considerations
 - Sample preparation
 - Replicates
 - Technical noise
 - Read depth
 - More depth for low expression genes
 - More depth for low expression differences between samples
 - RNA quality

RNA sequencing analysis considerations

What are the files that we need for RNA sequencing analysis?

{{Sdet}}

Solution{{Esum}}

- Reference genome or transcriptome
- Annotation files (gff or gtf) that tells us the genomic features (ie. gene, transcript, etc.)
- Raw sequencing data in FASTQ (or fq) format

{{Edet}}

Review of reference genome and annotation files

Why do we need a reference genome?

{{Sdet}}

Solution{{Esum}}

The reference genome serves as a "known" that guides us in constructing the genome of the unknown from sequencing data.

{{Edet}}

What file format is the reference genome in and what information does it contain?

{{Sdet}}

Solution{{Esum}}

The reference genome is in the fasta/fa format. These files will have extension fasta or fa, where the two extensions are used interchangeably.

A fasta file contains a definition line that starts with ">" followed by nucleotide sequences.

{{Edet}}

What is the annotation file used for?

{{Sdet}}

Solution{{Esum}}

The annotation file lists the features of a genome (ie. genes, transcripts, exons) along with their coordinates and other information. Annotations files are useful in RNA sequencing because it informs us of which gene or transcripts the aligned reads are overlapping and thus helps us generate a table of expression counts for our samples either on per gene or transcript basis.

{{Edet}}

Review of FASTQ files

What is a FASTQ file?

{{Sdet}}

Solution{{Esum}}

A fastq or fq file is the format for files that contain our sequencing data. Similar to a fasta file, which contains a header line that starts with ">" followed by sequence, the fastq file also contains a header line for each sequencing read that starts with "@". The sequencing read follows the metadata line, which is then followed by a "+" sign and a line that contains the quality score of each of the bases in a sequencing read.

{{Edet}}

What tool can we use to assess quality of sequencing data? And how do we aggregate several FASTQC reports into one.

Q



A

FASTQC

To aggregate multiple FASTQC reports, we can use MultiQC

Q

What type of data clean up can we perform on sequencing data prior to downstream analysis?

A

S

We can trim away adapters and low quality reads. Trimmomatic is a tool that can be used to do this.

Q

Lesson 13: Aligning raw sequences to reference genome

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 11 Review

In Lesson 11 we learned to aggregate multiple FASTQC reports into one using MultiQC, which allows us to easily interrogate the quality of sequencing data for multiple samples. We also learned to trim away adapters and poor quality reads in raw data using Trimmomatic (instructions for trimming using BBDuk are available in the Lesson 11 content for you to look at even though we did not go over this).

Learning objectives

In this lesson, we will continue to use the Human Brain Reference (HBR) and Universal Human Reference (UHR) data and we will

- Learn to align the sequencing data to reference genome using HISAT2, which is a splice aware aligner
- Look at post alignment QC
- Familiarize ourselves with the contents of alignment output
- Learn to use SAMTOOLS to work with alignment output
- Align sequences with BOWTIE2, which is not splice aware so we can visualize and compare to results obtained from HISAT2 using the Integrative Genome Viewer (IGV) in Lesson 14.

The skills learned in this lesson can be applied towards your own research and subsequent lessons in this course.

Creating an index for the reference genome

Let's start our exploration of sequencing read alignment by discussing the reference genome for human chromosome 22. For this, change into our `~/biostar_class/hbr_uhr/refs` folder. In Lesson 9, we discussed why we need a reference genome when we are working with high throughput sequencing data. In high throughput sequencing, we obtain many sequence reads in our experimental output and we do not know where in the genome these reads come from. The reference genome is a known standard that helps us reassemble these reads.


```
cd ~/biostar_class/hbr_uhr/refs
```

As a review, when we list the contents of this folder, we will see that it contains the reference genome (22.fa) and annotation (22.gtf) for human chromosome 22.

```
ls
```

```
22.fa 22.gtf ERCC92.fa ERCC92.gtf
```

The first step in alignment is to create an index for the reference genome. Think of an index as a table of contents in a book. If we are searching for something in a book, we can either search from beginning to end and depending on the size of the book, this could take a long time. Alternatively, we could use the table of contents to jump to and search only the relevant sections. Thus, an index allows the aligner to search more specifically and reduce computation time.

To align the HBR and UHR raw reads to chromosome 22 we will use a tool called **HISAT2** (<http://daehwankimlab.github.io/hisat2/manual/>), which is a splice aware aligner used for RNA sequencing. This aligner will be able to handle the alignment of reads that fall on two exons. We will use the build feature of HISAT2 to create our index. Other aligners will have their own algorithm for indexing the reference genome.

To build the index for human chromosome 22, we type `hisat2-build` at the command prompt, followed by the name of FASTA file for the reference genome (22.fa in our case) and then the base name (ie. file name without extension) that we would like to use for our index (here we choose 22 as the base name).

```
hisat2-build 22.fa 22
```

After the index has been generated, we can list the contents of our `biostar_class/hbr_uhr/refs` folder to see if anything has changed. We use the `-1` option with `ls` to list one item per row.

```
ls -1
```

Note that we now have HISAT2 genome indices, which are the 8 files that have extension "ht2".

```
22.1.ht2
22.2.ht2
22.3.ht2
22.4.ht2
22.5.ht2
```

```
22.6.ht2
22.7.ht2
22.8.ht2
```

Once the index for the human chromosome 22 reference has been created, change back into the ~/biostar_class/hbr_uhr folder.

```
cd ~/biostar_class/hbr_uhr
```

Then create a new folder called hbr_uhr_hisat2 and change into this. We will keep our alignment outputs in this folder.

```
mkdir hbr_uhr_hisat2
```

```
cd hbr_uhr_hisat2
```

Performing alignment for one file using HISAT2

To align FASTQ files for one sample, we construct the HISAT2 command with the following options.

- The "-x" flag prompts us to enter the base name (ie. without extension) of genome index. The HISAT2 index in the ~/biostar_class/hbr_uhr/refs directory, which is one directory back from our present working directory of ~/biostar_class/hbr_uhr/hbr_uhr_hisat2, so we can use ".." to specify go one directory back then go into refs.
- We specify files containing read 1 and read 2 of paired end sequencing after "-1" and "-2" flags, respectively. The reads are in ~/biostar_class/hbr_uhr/reads, which is one directory back from our present working directory of ~/biostar_class/hbr_uhr/hbr_uhr_hisat2, so we can use ".." to specify go one directory back then go into reads.
- Using the "-S" flag, we indicate that we want to save the alignment results in the SAM format where SAM stands for Sequencing Alignment Mapped.

```
hisat2 -x ../refs/22 -1 ../reads/HBR_1_R1.fq -2 ../reads/HBR_1_R2.fq
```

As HISAT2 is running the alignment, we get the alignment statistics below.

```
118571 (100.00%) were paired; of these:
  56230 (47.42%) aligned concordantly 0 times
  61769 (52.09%) aligned concordantly exactly 1 time
  572 (0.48%) aligned concordantly >1 times
```

```

----
56230 pairs aligned concordantly 0 times; of these:
  173 (0.31%) aligned discordantly 1 time
----
56057 pairs aligned 0 times concordantly or discordantly; of these:
  112114 mates make up the pairs; of these:
    112061 (99.95%) aligned 0 times
     48 (0.04%) aligned exactly 1 time
     5 (0.00%) aligned >1 times
52.75% overall alignment rate

```

Let's breakdown the alignment statistics shown above for the sample HBR_1.

The first line of the HISAT2 alignment statistics says 118571 reads (100.00%) were paired. Recall from FASTQC that read 1 and read 2 FASTQ files for HBR_1 have 118571 reads, each (Figures 1 and 2). So the first line in the HISAT2 alignment statistics is telling us that out of all the reads from read 1 and read 2 FASTQ files of HBR_1, we have 118571 pairs, which agrees with what we know. Also, this means we have 118571x2 or 237142 reads for the HBR_1 sample.

FastQC Report

Summary

- ✔ [Basic Statistics](#)
- ✔ [Per base sequence quality](#)
- ✔ [Per tile sequence quality](#)
- ✔ [Per sequence quality scores](#)
- ✘ [Per base sequence content](#)
- ! [Per sequence GC content](#)
- ✔ [Per base N content](#)
- ✔ [Sequence Length Distribution](#)
- ! [Sequence Duplication Levels](#)
- ! [Overrepresented sequences](#)
- ✔ [Adapter Content](#)

Basic Statistics

Measure	Value
Filename	HBR_1_R1.fq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	118571
Sequences flagged as poor quality	0
Sequence length	100
%GC	50

Per base sequence quality

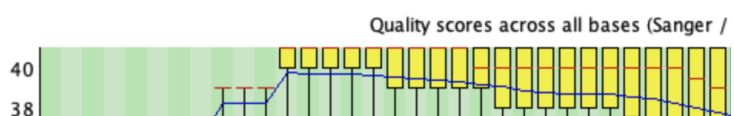












Figure 1

FastQC Report

Summary

-  [Basic Statistics](#)
-  [Per base sequence quality](#)
-  [Per tile sequence quality](#)
-  [Per sequence quality scores](#)
-  [Per base sequence content](#)
-  [Per sequence GC content](#)
-  [Per base N content](#)
-  [Sequence Length Distribution](#)
-  [Sequence Duplication Levels](#)
-  [Overrepresented sequences](#)
-  [Adapter Content](#)

Basic Statistics

Measure	Value
Filename	HBR_1_R2.fq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	118571
Sequences flagged as poor quality	0
Sequence length	100
%GC	50

Per base sequence quality



Figure 2

Following the first line of the HISAT2 alignment statistics, we will see some terminologies like concordant or discordant mapped reads. See Figure 3 for a visual explanation.

- A concordant read pair is defined as those that align in an expected manner where the reads are oriented towards one another and the distance between the outer edges is within expected ranges.
- A discordant read pair is defined as those that do not align as expected such as where the distance between the outer edges is smaller or larger than expected.

For the HBR_1 sample, we have

- 61769 pairs (123538 reads) that aligned concordantly once
- 572 pairs (1144 reads) that aligned concordantly more than once - these are mapped to multiple parts of the genome and likely cause by duplicated regions in the genome
- 173 paris (346 reads) that aligned discordantly once
- 48 reads that did align (neither concordantly or discordantly) once
- 5 reads that aligned (neither concordantly or discordantly) more than once

If we sum up the number of reads that mapped in the above break down and divide by the total number of reads in the two FASTQ files for the HBR_1 sample then we should get an overall alignment rate of 52.75%.

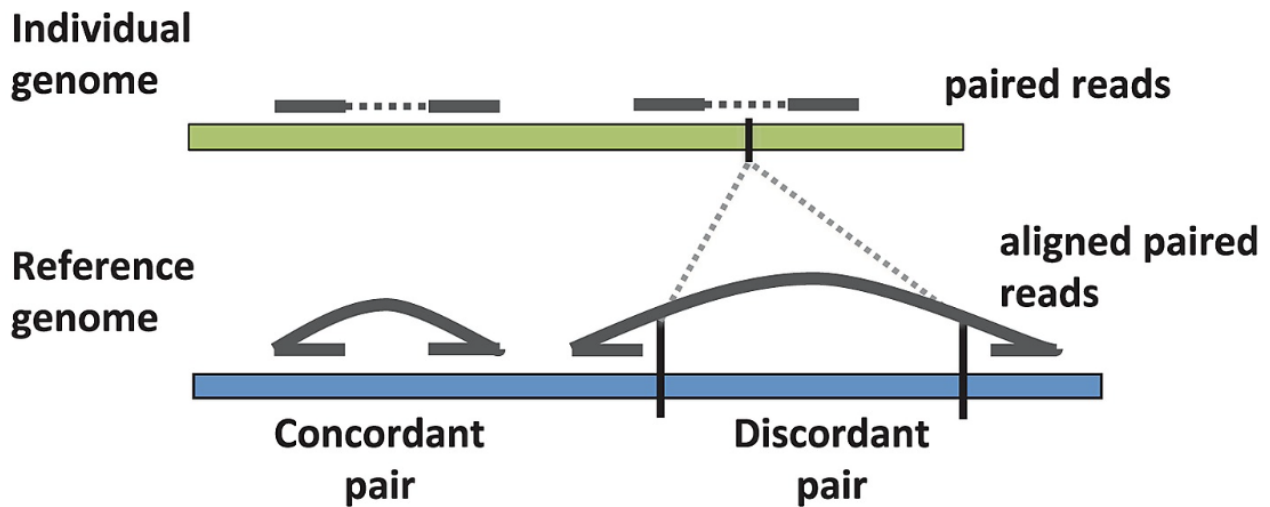


Figure 3: Source: Benjamin J. Raphael, Chapter 6: Structural Variation and Medical Genomics, PLOS Computational Biology, December 27, 2012 (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002821>)

If we include the option `--summary-file` in the HISAT2 command, we can specify a file name to save the alignment statistics.

Before moving further, let's use the `parallel` and `echo` commands to create a text file that contains the IDs for the HBR and UHR samples. This will allow us to align many FASTQ files at the same time.

To create the list of sample IDs for the HBR and UHR dataset we run the command below:

- `parallel` allows us to create the list of sample IDs all in one go (ie. multi-task rather than doing things in series)
- `echo` will print its arguments to the terminal (ie. `echo horse` will print `horse` in the terminal), in this case we want `echo` to print `{1}{2}`, where these can represent anything connected together by the underscore (the underscore is denoted by `_`). `{1}` denotes input 1 and `{2}` denotes input 2 such that input 1 is printed first and input 2 is printed second.
- the inputs in the command below are HBR and UHR, the sample groups in our dataset; "1 2 3" to denote 1, 2, or 3 replicates for each group (ie. we have samples HBR_1, HBR_2, HBR_3, UHR_1, UHR_2, UHR_3)
- one of the ways to specify input using `parallel` is the `:::` notation. here, after `{1}_{2}`, we specify the sample groups as the first input (HBR UHR) and then the replicate number (1, 2, or 3)
- we write this to a file called `ids.txt`, in the `~/biostar_class/hbr_uhr/reads` folder, where the `reads` folder is one directory up from our present working directory of `~/biostar_class/hbr_uhr/snidget_hisat2` so we can denote this using `..` (one directory up) and then specify the `reads` directory followed by the file name

```
parallel echo {1}_{2} ::: HBR UHR ::: 1 2 3 > ../reads/ids.txt
```

```
cat ../reads/ids.txt
```

```
HBR_1  
HBR_2  
HBR_3  
UHR_1  
UHR_2  
UHR_3
```

To align all of the HBR and UHR FASTQ files we can take advantage of the parallel command. Let's use the `--summary-file` option to store the alignment statistics. We will see why this is useful in a bit. In the command below, we

- read the sample names stored in `~/biostar_class/hbr_uhr/reads/ids.txt` using `cat`
- send this to `parallel` and the `hisat2` command is enclosed in double quotes within
- we use `{}` as a place holder for accepting the sample IDs provided by the `cat` command
- because we are dealing with paired-end sequencing, we append `_R1` and `_R2` to denote the first and second file in the pair

```
cat ../reads/ids.txt | parallel "hisat2 -x ../refs/22 -1 ../reads/{_
```

After alignment with HISAT2, let's list the contents of the `hbr_uhr_hisat2` directory to see what has changed.

```
ls -l
```

```
HBR_1.sam  
HBR_1_hisat2_summary.txt  
HBR_2.sam  
HBR_2_hisat2_summary.txt  
HBR_3.sam  
HBR_3_hisat2_summary.txt  
UHR_1.sam  
UHR_1_hisat2_summary.txt  
UHR_2.sam  
UHR_2_hisat2_summary.txt  
UHR_3.sam  
UHR_3_hisat2_summary.txt
```

If we print the alignment summary file for HBR_1 (HBR_1_hisat2_summary.txt) then we should see the same statistics that we saw earlier.


```
cat HBR_1_hisat2_summary.txt
```

Recall from Lesson 11 that we can include summaries from post alignment steps into MultiQC reports. This is why we create the summary files for the HISAT2 alignments for the HBR and UHR data. So let's run MultiQC again to generate a report that includes the HISAT2 alignment statistics. Make sure that we change the ~/biostar_class/hbr_uhr/ folder of our home directory and then construct the multiqc command below, where we specify the output filename using --filename. The output will be written in the QC directory. We use "." to tell multiqc to search ~/biostar_class/hbr_uhr for any QC or log files and it will search not only the present working directory but also sub-directories.

```
cd ~/biostar_class/hbr_uhr
```

```
multiqc --filename QC/multiqc_hbr_uhr_with_hisat2 .
```

Note that multiqc is now adding the alignment statistics in the report.

```
/// MultiQC  | v1.13

|           multiqc | Search path : /home/joe/biostar_class/hbr_uhr
|       searching | _____
|           snippy | Found 1 reports
|       bargraph | Tried to make bar plot, but had no data: snippy
|           bowtie2 | Found 6 reports
|           fastqc | Found 14 reports
|           multiqc | Compressing plot data
|           multiqc | Report      : qc/multiqc_with_hisat2.html
|           multiqc | Data       : qc/multiqc_with_hisat2_data
|           multiqc | MultiQC complete
```

We can copy multiqc_hbr_uhr_with_hisat2 to ~/public and then take a look at this file.

```
cp QC/multiqc_hbr_uhr_with_hisat2.html ~/public
```

After multiqc_hbr_uhr_with_hisat2.html has been copied to the ~/public directory, go back to the GOLD landing page (you can access it by clicking on the Class html links).

The screenshot shows the DNAnexus BioStars interface. The top navigation bar includes 'SETTINGS', 'MANAGE', 'MONITOR', and 'VISUALIZE'. The left sidebar shows a tree view with 'BioStars' expanded, containing 'Applications', 'Params', 'Sessions', and 'Test'. The main content area displays a file list for 'All Projects > BioStars'. The list includes folders for 'Params', 'Sessions', and 'Test', and several HTML files named 'Class_A_E.html' through 'Class_T_Z.html'. Each file entry shows its name, type/class, creation date and time, and the user who created it. At the bottom, it indicates '1-9 of 9 items'.

Name	Type / Class	Created	Created By
Params	Folder	--	--
Sessions	Folder	--	--
Test	Folder	--	--
Class_A_E.html	File	Nov 03 2022, 12:17 ...	user-mcintoshc
Class_F_M.html	File	Nov 03 2022, 3:01 PM	user-jwrows35z
Class_F_M.html	File	Nov 03 2022, 12:18 ...	user-mcintoshc
Class_N_S.html	File	Nov 03 2022, 12:21 ...	user-mcintoshc
Class_T_Z.html	File	Nov 03 2022, 12:24 ...	user-mcintoshc

At the GOLD landing page, scroll down the student table until you see your name and click the tab labeled File that is associated with your name.



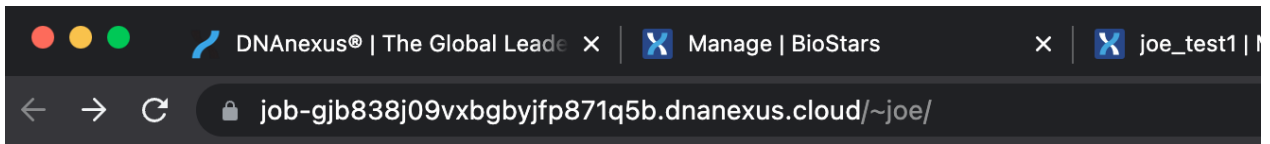
Welcome to GOLD, an online learning platform presented by BTEP

Use the links below to a) login to your account and b) view files in the [public] folder

pfitz	astank	cmcint	dtibb	arasmus	jwu
File	File	File	File	File	File

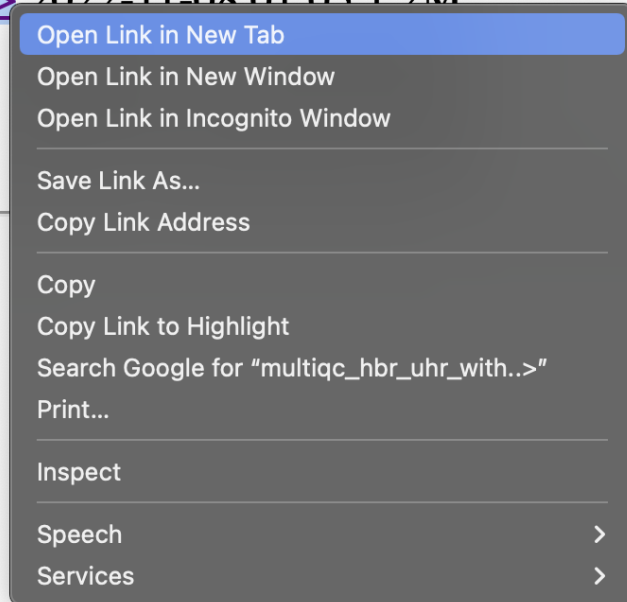
User	Login	View
Fan Yang	Fan	Files
Feng Zhu	Feng	Files
Gajendra Jogdand	Gajendra	Files
Gianluca Pegoraro	Gianluca	Files
Giovanni Maria	Giovanni	Files
Hima Makala	Hima	Files
Hoyoung Maeng	Hoyoung	Files
Jeeyoung Kang	Jeeyoung	Files
Jianping Li	Jianping	Files
Joe Wu	Joe	Files
Jung-Hyun Kim	Jung-Hyun	Files

You will then be taken to a page that where you access the files in your ~/public directory. You can either right click to download the files or view in a separate browser tab in the case of html files.



Index of /~joe

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 multiqc hbr uhr with..>	2022-11-08 01:05	1.2M	
 multiqc report snidg..>			
 multiqc report snidg..>			



In the navigation pane of the multiqc report, we now see a link to the hisat2 alignment statistics (Figure 4).

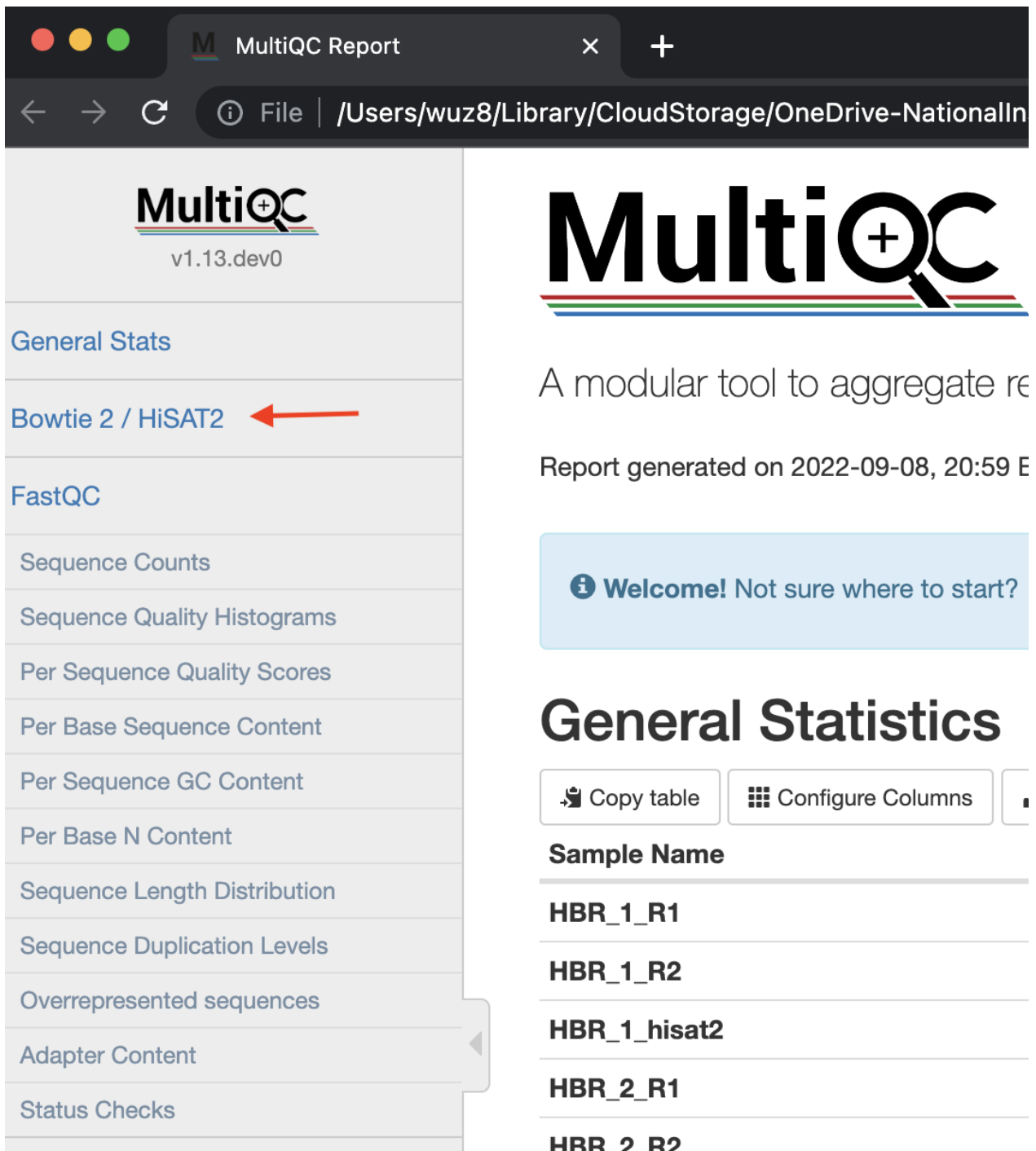


Figure 4

In the general statistics table, we now have a column indicating the overall alignment rate for each sample (Figure 5).

General Statistics

Copy table | Configure Columns | Plot | Showing 19/18 rows and 6/6 columns.

Sample Name	% Aligned	% Dups	% GC	Read Length	% Failed	M Seqs
HBR_1_R1		42.7%	50%	100 bp	9%	0.1
HBR_1_R2		43.6%	50%	100 bp	9%	0.1
HBR_1_hisat2	52.8%					
HBR_2_R1		44.3%	50%	100 bp	9%	0.1
HBR_2_R2		45.2%	50%	100 bp	0%	0.1
HBR_2_hisat2	52.5%					
HBR_3_R1		43.3%	50%	100 bp	9%	0.1
HBR_3_R2		44.1%	50%	100 bp	0%	0.1
HBR_3_hisat2	52.7%					
UHR_1_R1		44.5%	49%	100 bp	9%	0.2
UHR_1_R2		45.3%	49%	100 bp	0%	0.2
UHR_1_hisat2	47.9%					
UHR_2_R1		46.8%	48%	100 bp	9%	0.2
UHR_2_R2		47.4%	49%	100 bp	9%	0.2
UHR_2_hisat2	51.3%					
UHR_3_R1		51.5%	49%	100 bp	18%	0.2
UHR_3_R2		52.7%	49%	100 bp	9%	0.2
UHR_3_hisat2	46.9%					

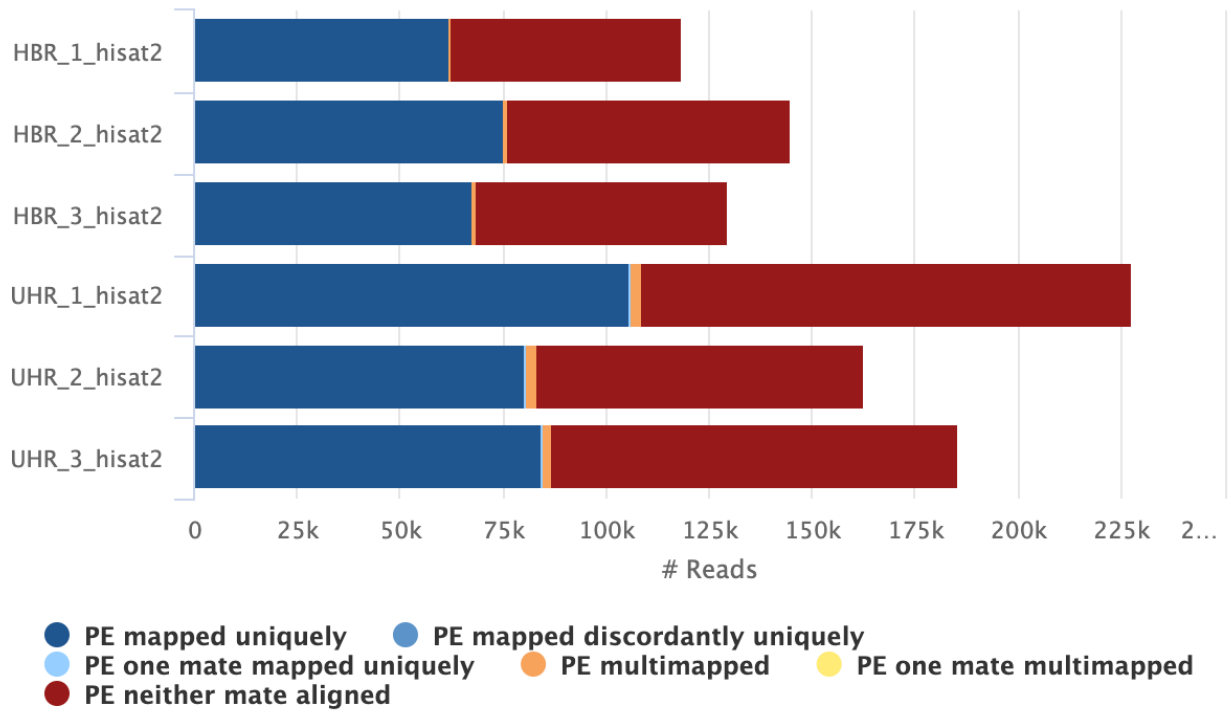
Figure 5

We also have a plot (Figure 6) that tells us

- How many pair of reads mapped uniquely (ie. only once)
- How many pair of reads mapped concordantly to more than one location
- How many pair of reads mapped uniquely but discordantly
- How many pair of reads did not align
- How many pair of reads where one of the pair mapped uniquely
- How many pair of reads where one of the pairs were multimapped

This information is the same that we see in the alignment statistics generated by HISAT2, except now we have combined it with our pre-alignment QC reports and this provides a nice way for us to keep the logs and summary of our analysis in one place so we can share with colleagues and collaborators.

Bowtie 2: PE Alignment Scores



Created with MultiQC

Figure 6

If we click on the bar for one of the samples, a dialogue box with alignment statistics appears and we can then see the numbers.

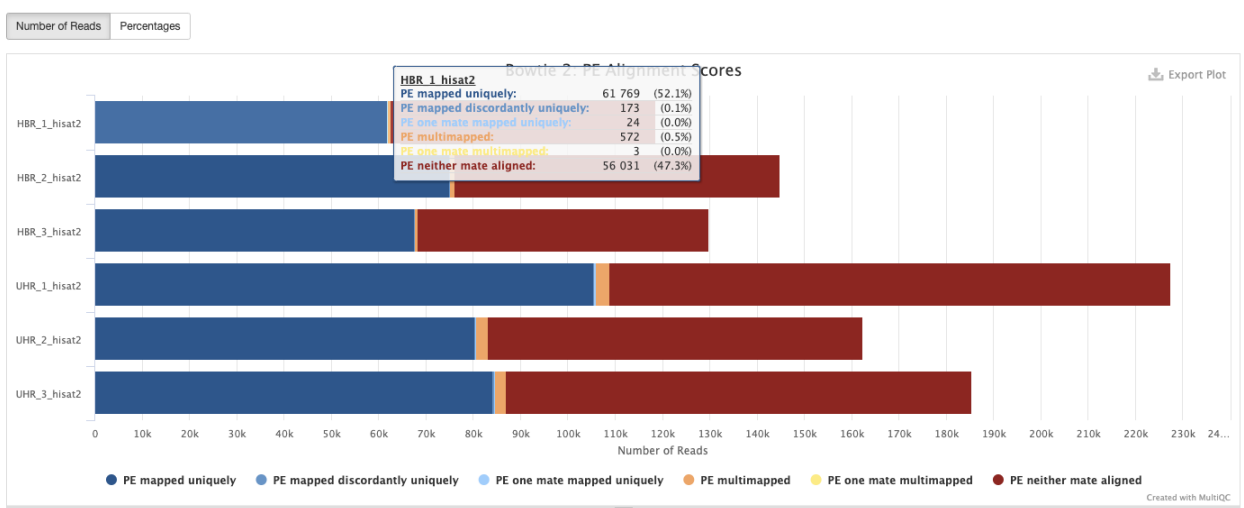


Figure 7

Working with SAM file alignment outputs

Change in the ~/biostar_class/hbr_uhr/hbr_uhr_hisat2 folder for this portion of the class.

```
cd ~/biostar_class/hbr_uhr/hbr_uhr_hisat2
```

Now that we have our SAM files generated for the HBR and UHR dataset. For your reference, information on the SAM file can be found [here \(https://samtools.github.io/hts-specs/SAMv1.pdf\)](https://samtools.github.io/hts-specs/SAMv1.pdf). A SAM file is tab delimited, so we can open it using Excel. I am going to copy HBR_1.sam to my ~/public folder so I can open this locally on my Excel to go over this SAM file (try to just watch what I do here).

SAM files always start off with metadata information and these lines start with @.

- @HD includes
 - SAM file format information (in this case version 1.0 as indicated by VN:1.0)
 - Whether the alignment file has been sorted (in this case no as indicated by SO:unsorted)
- @SQ provides reference information
 - SN denotes reference sequence name, which is chr22
 - LN is the reference length in bases, which is 50818468 as we found in Lesson 9 using seqkit stats
- @PG provides information about the program we used to generate the alignment
 - ID is the program record identifier
 - PN is the program name (HISAT2 in our case version 2.2.1 as indicated next to VN)
 - CL is the command line call used to generate the alignment

After the metadata information, each SAM file contains 11 fields.

1. QNAME or query template name, which is essentially the sequencing read that we want mapped onto a reference genome. If we grep the first sequence in the HBR_1.sam, then we should retrieve one of the reads in the original FASTQ files (try it on your own).
2. The second column is a FLAG that tells us a bit about the mapping. [Here is a good tool to help interpret these FLAGS \(https://broadinstitute.github.io/picard/explain-flags.html\)](https://broadinstitute.github.io/picard/explain-flags.html). These FLAGS can inform us of things like pair end read alignment orientation.
3. Column three contains the name of our reference genome.
4. Column four tells us the left most genomic coordinate where the sequencing read maps.
5. The mapping quality (MAPQ) is provided in the fifth column (the higher the number, the more confident we are of the map); a value of 255 in this column means that the mapping quality is not available.
6. Column six presents the CIGAR string, which tells us information about the match/mismatch, insertion/deletion, etc. of the alignment.

7. Column seven is the reference sequence name of the primary alignment of the NEXT read in the template. We will see an "=" if this name is the same as the current (which we would expect for paired reads)
8. The alignment position of the next read in the template is provided in column 8. When this is not available, the value is set to 0.
9. Column nine provides the template length (TLEN), which tells us how many bases of the reference genome does the sequencing read span.
10. The tenth column is just the sequencing read (some are written as the reverse complement so be cautious. The FLAGS in column two will tell us whether the sequence is reverse complemented).
11. The eleventh column is the Phred quality scores of the sequencing read.
12. For definition of optional fields, see <https://samtools.github.io/hts-specs/> (<https://samtools.github.io/hts-specs/SAMtags.pdf>).

The SAM file format is human readable so we will need to convert it to a machine readable format (Binary Alignment Map or BAM) before we can visualize alignment results using IGV and perform other downstream processes like obtaining read counts. To convert between SAM and BAM, we can use an application called **SAMTOOLS** (<http://www.htslib.org/>).

If we type samtools on the command line, we can see that this application has lots of features. For an alignment file to be of use, we will need to sort it by genomic position using the sort feature of SAMTOOLS. So we type the following command where the -o flag prompts us to enter the output file name (in this case it will be a BAM file). The last argument is the input SAM file, which is the SAM file that we want to sort.

```
samtools sort -o HBR_1.bam HBR_1.sam
```

Now that we have our BAM file for HBR_1 generated, we need to index it. Similar to the idea of indexing a reference genome, indexing the BAM file will allow the program that uses it to more efficiently search through it. For this we will use samtools index, where the -b flag tells SAMTOOLS to create the index from a BAM file. We include the extension ".bai" in the output file.

```
samtools index -b HBR_1.bam HBR_1.bam.bai
```

The sorting and indexing of our BAM files are perhaps the two necessary steps that allows us to visualize and move forward with our analysis. The above shows how to sort and index one BAM file at a time. Below, let's use the parallel command to take care of these tasks for all of our alignment outputs at one go.

Similar to how we constructed the hisat2 alignment, we use cat to send the sample IDs to parallel. The samtools command construct is enclosed in double quotes and {} acts as a place holder for accepting the sample IDs provided by the cat command.

```
cat ../reads/ids.txt | parallel "samtools sort -o {}.bam {}.sam"
```

```
cat ../reads/ids.txt | parallel "samtools index -b {}.bam {}.bam.bai"
```

Alignment of HBR and UHR raw sequencing data with Bowtie2

For RNA sequencing studies, we need to use a splice aware aligner to account for reads that map across exons. Bowtie2 is a commonly used aligner for DNA sequencing and is not splice aware. Let's align the HBR and UHR data to human chromosome 22 using Bowtie2 so we can see how the alignment results differ from HISAT2 when we visualize alignment results in Lesson 14.

To run Bowtie2, we will need to create a Bowtie2 specific index for chromosome 22, so change into the `~/biostar_class/hbr_uhr/refs` folder.

```
cd ~/biostar_class/hbr_uhr/refs
```

Then, we use `bowtie2-build` using `22.fa` as input (like we did with HISAT2) and assign to the index the base name of 22.

```
bowtie2-build 22.fa 22
```

Listing the contents of our `biostar_class/hbr_uhr` folder, we see the Bowtie2 indices for chromosome 22 has been created and these have extension `*.bt2`

```
ls -l
```

```
22.1.bt2
22.1.ht2
22.2.bt2
22.2.ht2
22.3.bt2
22.3.ht2
22.4.bt2
22.4.ht2
22.5.ht2
22.6.ht2
```

```
22.7.ht2
22.8.ht2
22.fa
22.gtf
22.rev.1.bt2
22.rev.2.bt2
ERCC92.fa
ERCC92.gtf
```

Before running Bowtie2 alignment, creates a folder called `hbr_uhr_bowtie2` in the `~/biostar_class/hbr_uhr` directory and change into.

```
cd ~/biostar_class/hbr_uhr
```

```
mkdir hbr_uhr_bowtie2
```

```
cd hbr_uhr_bowtie2
```

The construct for the command to alignment using Bowtie2 is similar to Hisat2, with the exception that we append `"_bowtie2"` to the output so that we know that it is from a Bowtie2 alignment. HISAT2 is actually based on Bowtie2 <http://daehwankimlab.github.io/hisat2/manual/> (<http://daehwankimlab.github.io/hisat2/manual/>).

In the command below

- read the sample names stored in `~/biostar_class/hbr_uhr/reads/ids.txt`
- send this to `parallel` and the `bowtie2` command is enclosed in double quotes within
- we use `{}` as a place holder for accepting the sample IDs provided by the `cat` command
- because we are dealing with paired-end sequencing, we append `_R1` and `_R2` to denote the first and second file in the pair

```
cat ../reads/ids.txt | parallel "bowtie2 -x ../refs/22 -1 ../reads/{}
```

And now we will sort the Bowtie2 SAM files and convert to BAM.

In the commands below, we use `cat` to send the sample IDs to `parallel`. The `samtools` command construct is enclosed in double quotes and `{}` acts as a place holder for accepting the sample IDs provided by the `cat` command.

```
cat ../reads/ids.txt | parallel "samtools sort -o {}_bowtie2.bam {}_t
```


Finally, we will index the the Bowtie2 alignment results.

```
cat ../reads/ids.txt | parallel "samtools index -b {}_bowtie2.bam {}_"
```

MultiQC reports with pre-alignment QC and post-alignment statistics

HBR and UHR MultiQC with pre- and post- alignment statistics

Lesson 14: Visualizing alignment results

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 13 Review

Previously, we used the application HISAT2 to align the raw sequencing data from the Human Brain Reference (HBR) and Universal Human Reference (UHR) dataset. We created sorted and indexed alignment output in the form of BAM files that we could use to visualize results in the Integrative Genome Viewer (IGV). We also used the splice unaware Bowtie2 aligner to map the HBR and UHR reads to chromosome 22 and will see how these results differ from HISAT2 when visualizing in IGV.

Learning objectives

In this lesson, we will continue to use the HBR and UHR dataset and focus on learning how to visualize the alignment outputs (both HISAT2 and Bowtie2) in IGV.

Creating bigWig files

Due to time constraints, we will not be going over the creation of bigWig files in class. The information below is for your reference and you can view these during your spare time.

In Lesson 9, we got a short introduction on what IGV can do. It allows us to visualize genomic data such as reference genomes and how features such as genes and transcripts align to them. A common thing to do after aligning raw sequencing reads is to visually inspect the results in IGV. In this lesson, we will do exactly this. In Lesson 11, we generated the BAM files, here we will generate an additional file that is used to visualize sequencing coverage in IGV. This file format is known as **bigWig** or **bw** (<https://genome.ucsc.edu/goldenPath/help/bigWig.html>). bigWig files are binary so not human readable and make visualization faster because the computer only needs to store in memory the content that is needed to be displayed. We will use bedtools and bedGraphToBigWig to generate the bigWig files for the HISAT2 and Bowtie2 alignment of the HBR and UHR dataset.

Be sure to change into the ~/biostar_class/hbr_uhr folder for this portion of the class.

```
cd ~/biostar_class/hbr_uhr
```

Creating bigWig files - step 1, convert BAM to bedGraph

Step 1 for generating bigWig files is to convert the BAM alignment results to a bedGraph (with extension bg) file that contains coverage along genomic regions.

Enhancing your vocabulary:

- BED file - this is also known as Browser Extensible Format and contains three columns, which are chromosome, start coordinate and end coordinate -- [see Ian's response in this Biostars forum \(https://www.biostars.org/p/113452/\)](https://www.biostars.org/p/113452/)
- bedGraph - this has the same first three columns as the BED file but also has a fourth column that could be anything such as sequencing coverage -- [also see Ian's response in this Biostars forum \(https://www.biostars.org/p/113452/\)](https://www.biostars.org/p/113452/)

Example BED file below -- <https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html> (<https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html>)

```
$ cat A.bed
chr1 10 20
chr1 20 30
chr2 0 500
```

Example bedGraph below -- <https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html> (<https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html>)

```
$ bedtools genomecov -ibam NA18152.bam -bg | head
chr1 554304 554309 5
chr1 554309 554313 6
chr1 554313 554314 1
chr1 554315 554316 6
chr1 554316 554317 5
chr1 554317 554318 1
chr1 554318 554319 2
chr1 554319 554321 6
chr1 554321 554323 1
chr1 554323 554334 7
```

To generate a bedGraph file from BAM alignment outputs from the HBR and UHR dataset, we will use an application called [bedtools](https://bedtools.readthedocs.io/en/latest/index.html) (<https://bedtools.readthedocs.io/en/latest/index.html>), which can be used for a range of tasks including compiling information on genomic intervals. We will use its [genomecov](https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html) (<https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html>), which calculates coverage over a genomic range with the following options:

- -ibam: prompts us to provide the name of the BAM input file

- `-split`: when applied to RNA sequencing data, do not count reads that map to introns [see this post on why we get reads that map to introns in RNA sequencing \(https://www.biostars.org/p/42890/\)](https://www.biostars.org/p/42890/)
- `-bg`: reports sequencing depth along a genomic interval rather than at each nucleotide position (See Figure 1, shows the ways we can get bedtools to output coverage information - where some options report coverage along an interval, some report at each base position. Bedtools also gives us the ability to fine tune how we count coverage along splice junctions with the `split` option)

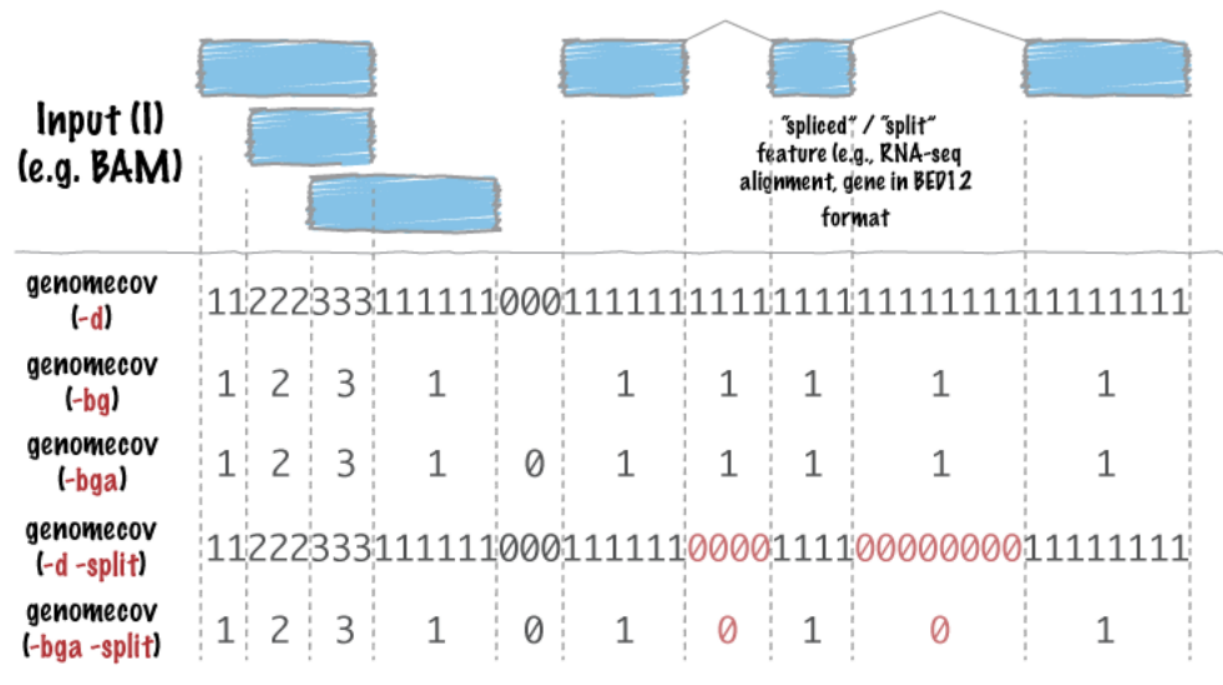


Figure 1: Source: <https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html> (<https://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html>)

Below, we take advantage of the parallel command to convert the BAM files from both HISAT2 and Bowtie2 alignments into bedGraph (bg) files for all of the samples in one go.

- `cat reads/ids.txt`: this will read the HBR and UHR sample IDs that are stored in a file called `ids.txt` in the `reads` folder of the `~/biostar_class/hbr_uhr` directory
- `|`: will pipe or send the output of `cat` to the next command, which is `parallel`
- we enclose the command that we want `parallel` to operate on in double quotes; here we are using `bedtools` and its `genomecov` subcommand, where the parameters `-ibam`, `-split`, and `-bg` were explained above
- `hbr_uhr_hisat2/{}.bam`: this is the path to our BAM file generated from aligning the HBR and UHR FASTQ files to genome
 - `hbr_uhr_hisat2` is the output directory for the HISAT2 alignment (`hbr_uhr_bowtie2` is the output directory for the Bowtie2 alignment)

- we use {} as a place holder to receive the HBR and UHR sample IDs from cat (ie. HBR_1, HBR_2, HBR_3, UHR_1, UHR_2, UHR3) and these IDs are appended by .bam to complete the full path of the input BAM file
- hbr_uhr_hisat2/{}_hisat2.bg:
 - we write the bedGraph output to the hbr_uhr_hisat2 for the HISAT2 alignment (hbr_uhr_bowtie2 for the Bowtie2 alignment)
 - again, {} acts as a place holder to receive the HBR and UHR sample IDs from cat, and the sample IDs are appended with _hisat2.bg for the HISAT2 alignment output (or _bowtie2.bg for Bowtie2 alignment output) to complete the full path of the output bedGraph (bg) file

First, create bg files from HISAT2 alignments.

```
cat reads/ids.txt | parallel "bedtools genomecov -ibam hbr_uhr_hisat2/{}_hisat2.bg"
```

Next, create bg files from Bowtie2 alignments.

```
cat reads/ids.txt | parallel "bedtools genomecov -ibam hbr_uhr_bowtie2/{}_bowtie2.bg"
```

Below we will look at the content of HBR_1_hisat2.bg sorted by sequencing depth (column 4) from highest to lowest using the column command.

- column: prints out tabular data nicely in the terminal
 - -t: figures out how many columns there are in the input, which is HBR_1_hisat2.bg in the folder hbr_uhr_hisat2 (full input path: hbr_uhr_hisat2/HBR_1_hisat2.bg)
- |: pipes or sends the output from column to sort, where
 - -k: allows us to specify the column to sort by, in this case we want to sort by column 4 (the coverage) and we want to sort numerically (n) and from highest to lowest (r) with the final construct of 4nr
- |: pipes or sends the sort output to less where -S allows us to scroll horizontally in the terminal to see columns that did not fit

```
column -t hbr_uhr_hisat2/HBR_1_hisat2.bg | sort -k 4nr | less -S
```

The first column is the chromosome, followed by the genomic coordinates and the sequencing depth.

```
chr22 42565097 42565102 567
chr22 42565102 42565137 566
chr22 42565137 42565142 565
chr22 42565142 42565167 564
```

```
chr22 42565167 42565169 562
chr22 42565096 42565097 531
```

Creating bigWig files - step 2, create a genome index

To proceed with converting the bedGraph files to bigWig, we need to first create an index of our genome using SAMTOOLS and its `faidx` feature. Where `faidx` will index/extract FASTA.

```
samtools faidx refs/22.fa
```

Listing the contents of our `refs` directory, we now see an index of the human chromosome 22 genome named `22.fa.fai`.

```
ls refs
```

Creating bigWig files - step 3, actually creating the bigWig files

After the index file for the genome has been created, we will use a tool called `bedGraphToBigWig` (<https://www.encodeproject.org/software/bedgraphtobigwig/>) to generate bigWig (bw) files from bedGraph (bg).

Again, we use `cat` and `parallel` where

- `cat reads/ids.txt`: this will read the HBR and UHR sample IDs that are stored in a file called `ids.txt` in the `reads` folder of the `~/biostar_class/hbr_uhr` directory
- `|`: will pipe or send the output of `cat` to the next command, which is `parallel`
- we enclose the command that we want `parallel` to operate on in double quotes; here we are using `bedGraphToBigWig`
- `hbr_uhr_hisat2/{}_hisat2.bg`: this is the path to our bg (bedGraph) file generated using `bedtools genomecov`
 - `hbr_uhr_hisat2` is the output directory for the HISAT2 alignment (`hbr_uhr_bowtie2` is the output directory for the Bowtie2 alignment)
 - we use `{}` as a place holder to receive the HBR and UHR sample IDs from `cat` (ie. `HBR_1`, `HBR_2`, `HBR_3`, `UHR_1`, `UHR_2`, `UHR3`) and these IDs are appended by `.bg` to complete the full path of the input BAM file
- `refs/22.fa.fai`: path for index of the human chromosome 22 reference
- `hbr_uhr_hisat2/{}_hisat2.bw`:
 - we write the bigWig output to the `hbr_uhr_hisat2` for the HISAT2 alignment (`hbr_uhr_bowtie2` for the Bowtie2 alignment)
 - again, `{}` acts as a place holder to receive the HBR and UHR sample IDs from `cat`, and the sample IDs are appended with `_hisat2.bw` for the HISAT2 alignment output

(or `_bowtie2.bw` for Bowtie2 alignment output) to complete the full path of the output bigWig (bw) file

Generate bigWig files for the HISAT2 alignment.

```
cat reads/ids.txt | parallel "bedGraphToBigWig hbr_uhr_hisat2/{}_hisat2.bw"
```

Generate bigWig files for the Bowtie2 alignment.

```
cat reads/ids.txt | parallel "bedGraphToBigWig hbr_uhr_bowtie2/{}_bowtie2.bw"
```

Visualizing alignment HBR and UHR alignment results with IGV

In this portion of the class, it is very important that you have IGV already opened on your computer. See Figure 1 and Figure 2 on how to load the relevant alignment outputs for HBR_1 and UHR_1 into IGV.

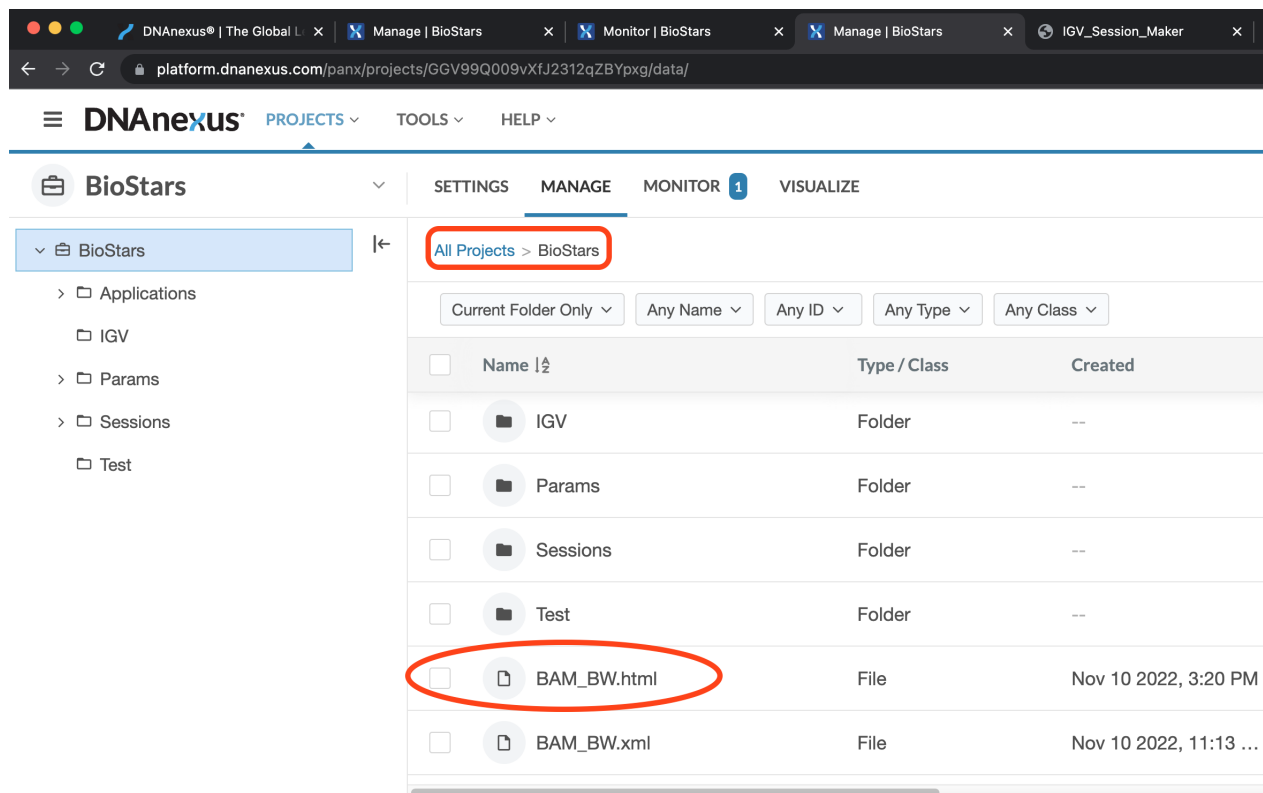
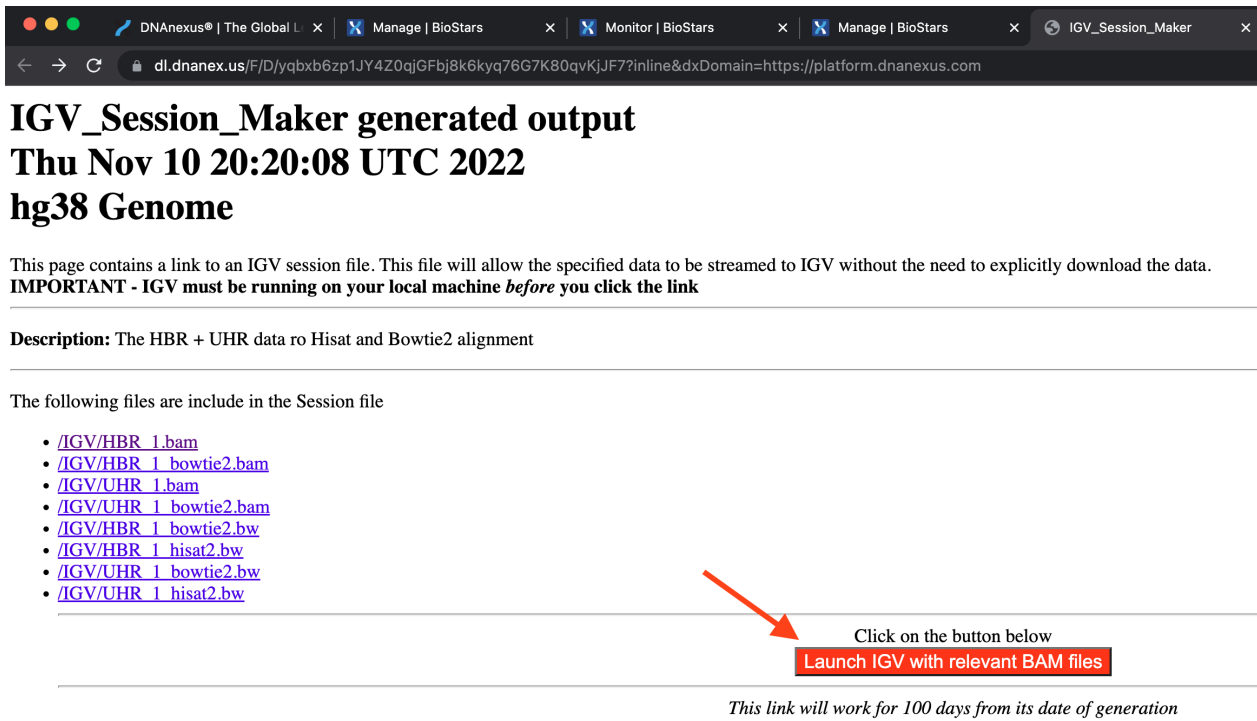


Figure 1: Click on the BAM_BW.html under All Projects -> BioStars to access the IGV launcher for the HBR and UHR dataset.



DLAnexus® | The Global L | Manage | BioStars | Monitor | BioStars | Manage | BioStars | IGV_Session_Maker

dl.dnanexus.us/F/D/yqxb6zp1JY4Z0qjGFbj8k6kyq76G7K80qvKJF7?inline&dxDomain=https://platform.dnanexus.com

IGV_Session_Maker generated output

Thu Nov 10 20:20:08 UTC 2022

hg38 Genome

This page contains a link to an IGV session file. This file will allow the specified data to be streamed to IGV without the need to explicitly download the data. **IMPORTANT - IGV must be running on your local machine *before* you click the link**

Description: The HBR + UHR data ro Hisat and Bowtie2 alignment

The following files are include in the Session file

- [/IGV/HBR_1.bam](#)
- [/IGV/HBR_1_bowtie2.bam](#)
- [/IGV/UHR_1.bam](#)
- [/IGV/UHR_1_bowtie2.bam](#)
- [/IGV/HBR_1_bowtie2.bw](#)
- [/IGV/HBR_1_hisat2.bw](#)
- [/IGV/UHR_1_bowtie2.bw](#)
- [/IGV/UHR_1_hisat2.bw](#)

Click on the button below

Launch IGV with relevant BAM files

This link will work for 100 days from its date of generation

Figure 2: Click the launch button to view the alignments.

We can also select the reference genome to use in the genome selection drop down menu. But here, we will stick with hg38 (Figure 3).

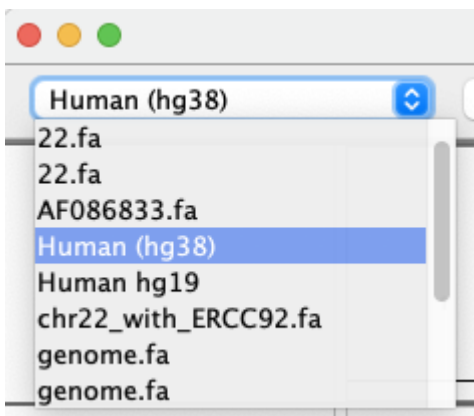


Figure 3

We can also load local data to IGV by going to "File" and then "Load from File" (Figure 4). But the alignment output has already been pre-loaded for us, so we will not need to load data from local.

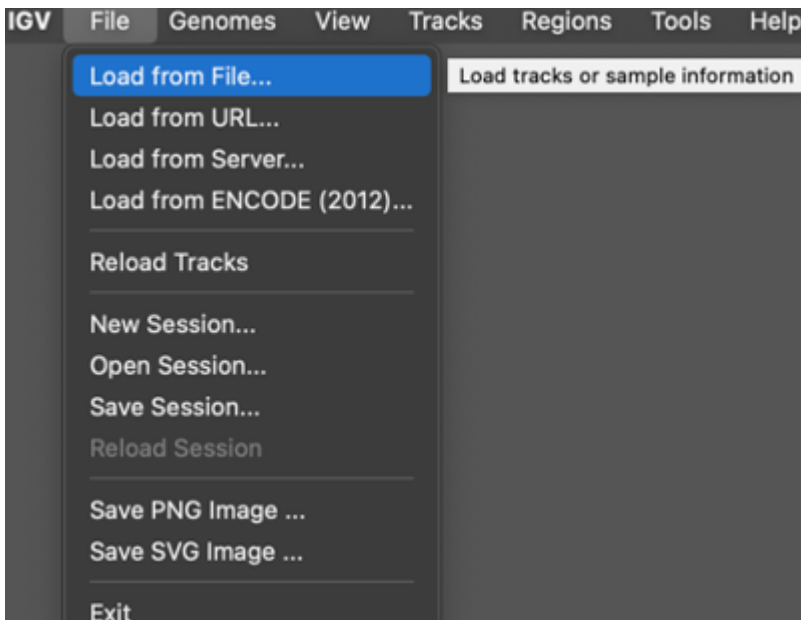


Figure 4

Unfortunately, the entire IGV browser is empty after we loaded the data, with the exception of some coverage at chromosome 22 on the bigWig tracks from hg38. This is because we aligned our HBR and UHR raw sequencing data only to chromosome 22.

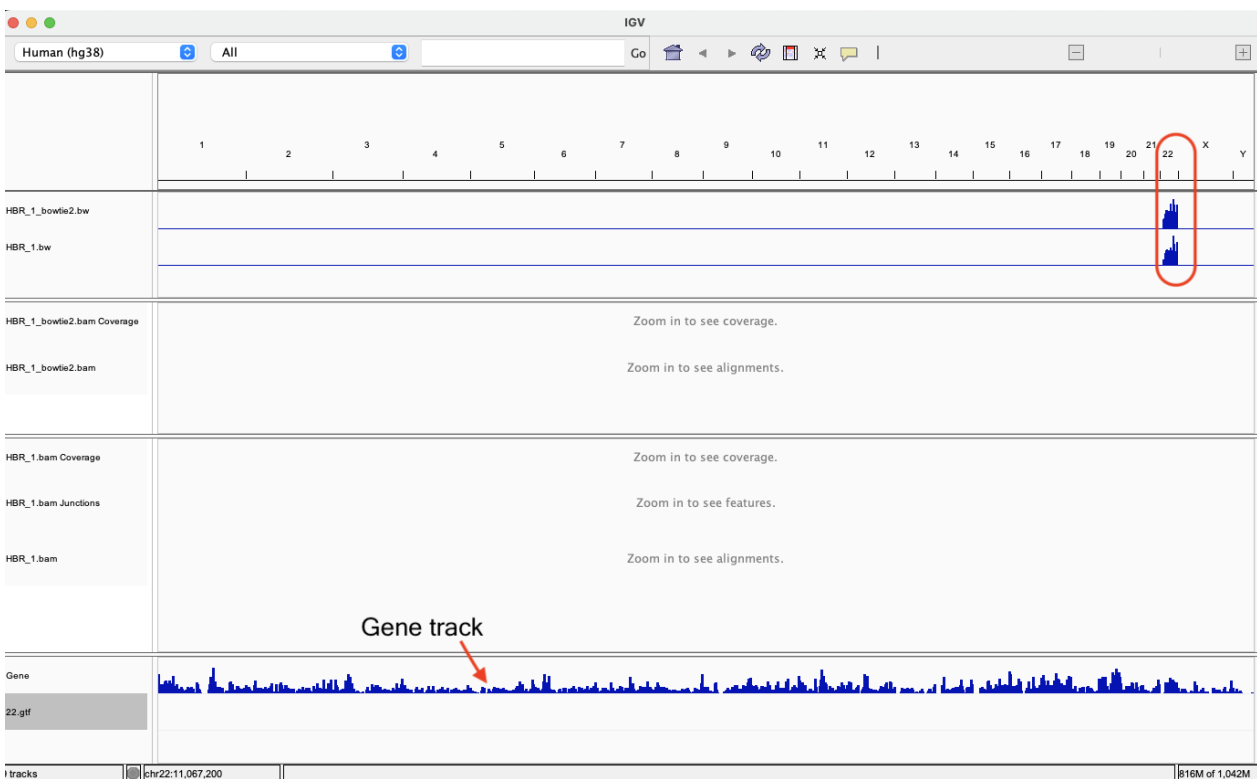


Figure 5

Once we select chr22 (Figure 6), we will begin to see more information in IGV.

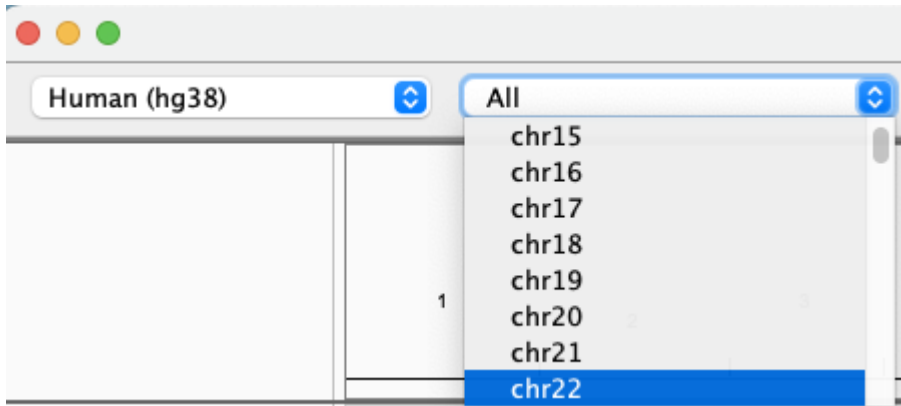


Figure 6

Let's now go over what the data tracks are

- On the top (Tracks 1) we have the bigWig or bw tracks for HBR_1 aligned with HISAT2 (HBR_1.bw) or aligned with Bowtie2 (HBR_1_bowtie2.bw).
- Tracks 2 - the BAM alignment for HBR_1 aligned with Bowtie2 (alignment and coverage tracks are separate)
- Tracks 3 - the BAM alignment for HBR_1 aligned with HISAT2 (alignment, splice junction, and coverage tracks are separate)
- On the bottom, we have the gene model track

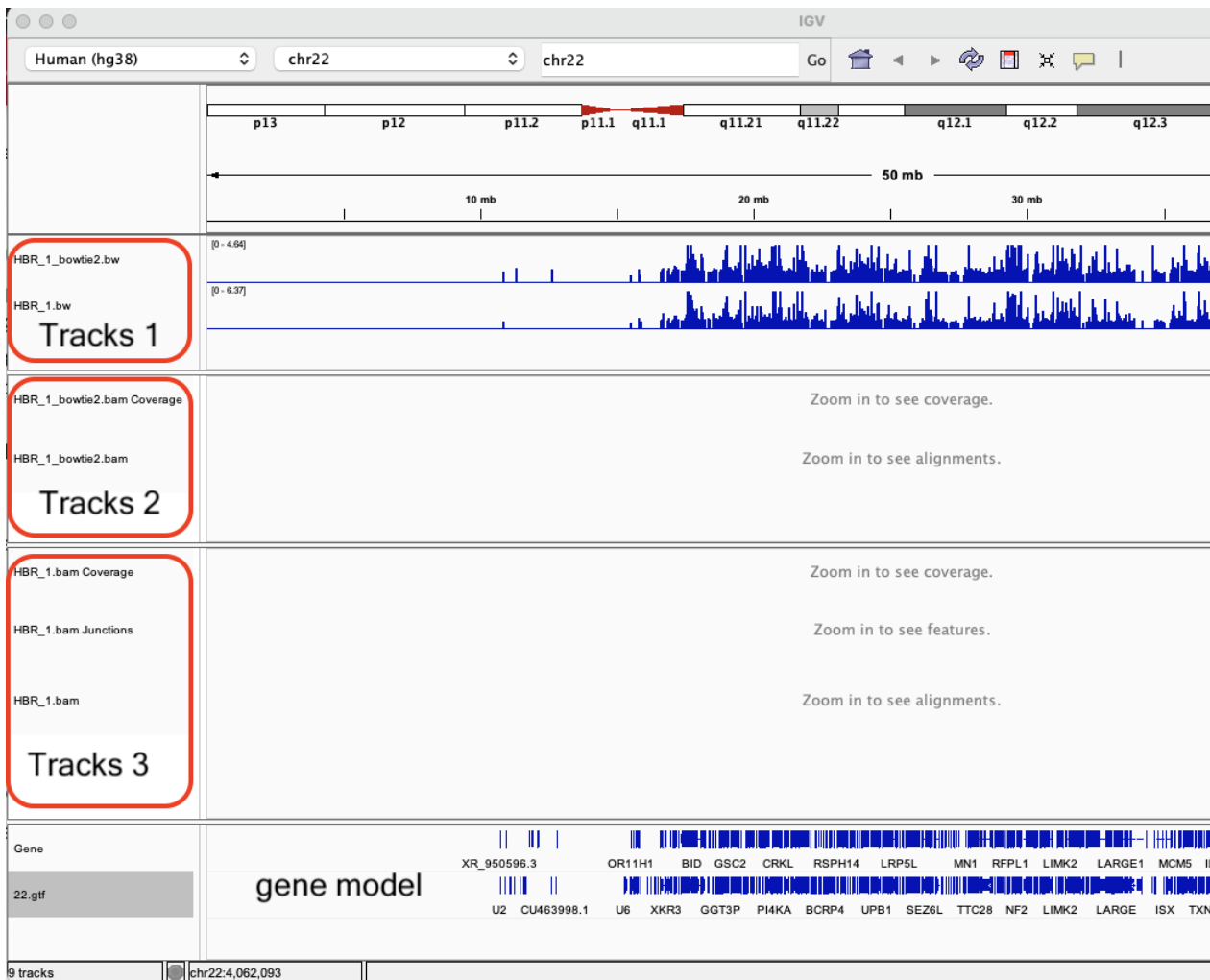


Figure 7

When the alignment results have loaded, let's go to chr22:41,377,179-41,390,187 (enter this in the search box and select Go) (Figure 6). Things to note are

- The TEF gene occupies this region
- Note that in the HISAT2 alignment track, some of the reads have lines connecting them. These reads are mapping across exons, so HISAT2, being splice aware can connect these. We do not see this feature in the Bowtie2 alignment.
- The HISAT2 alignment output also has a [splice junction track](https://software.broadinstitute.org/software/igv/splice_junctions) (https://software.broadinstitute.org/software/igv/splice_junctions), where the arc height and thickness is proportional to read depth.
- The bigWig (bw) tracks show only coverage information.



Figure 8

We should also note the difference in the coverage histogram between the HBR_1 HISAT2 and Bowtie2 alignments on the bigWig track (chr22:41,377,179-41,390,187).

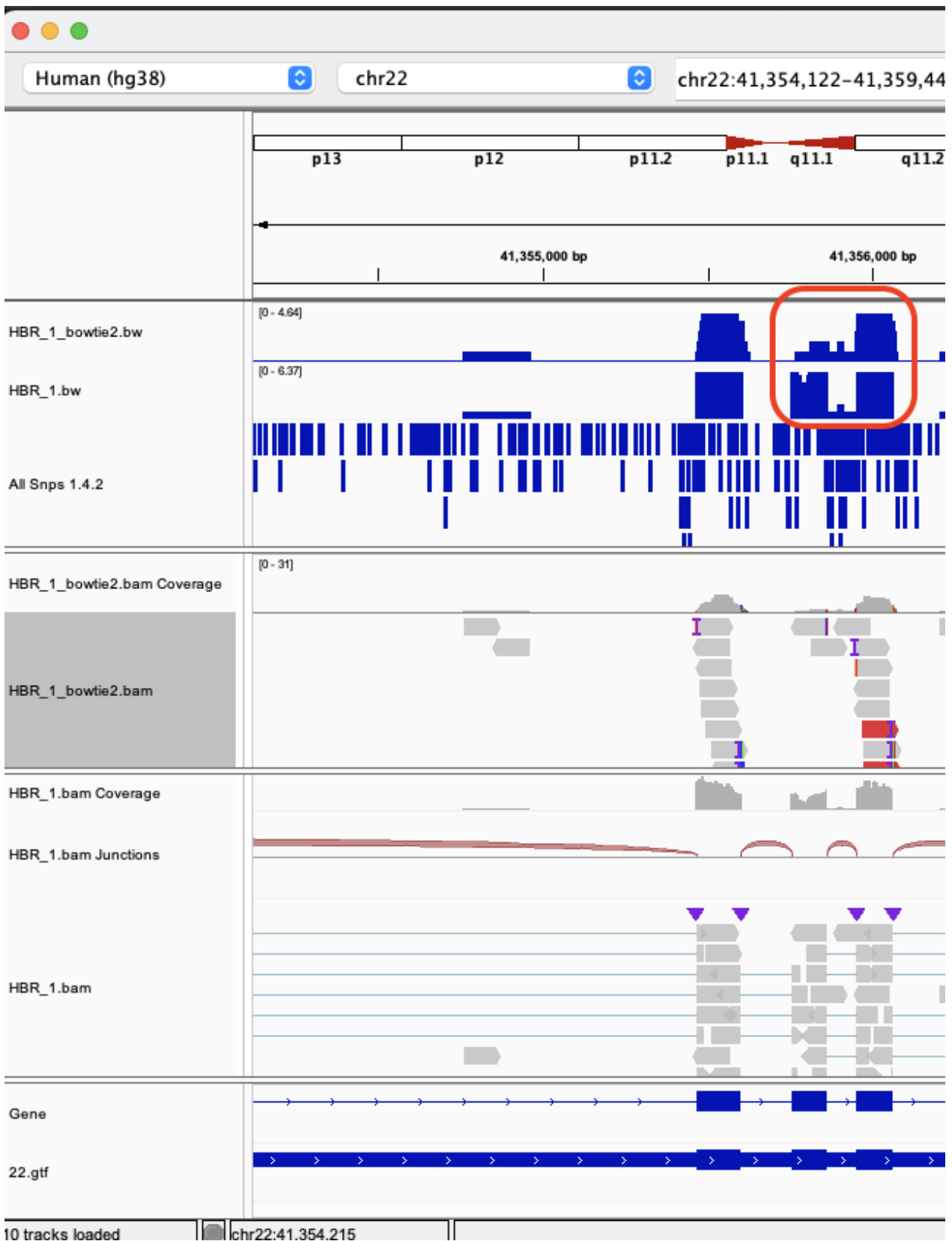


Figure 9

If we zoom in to chr22:41,387,655, we will see that the coverage track is color partly in blue and partly in red (Figure 10). This is showing a potential variant, where the reference is C (look at the sequences right above the TEF gene model) and some of the samples have T.

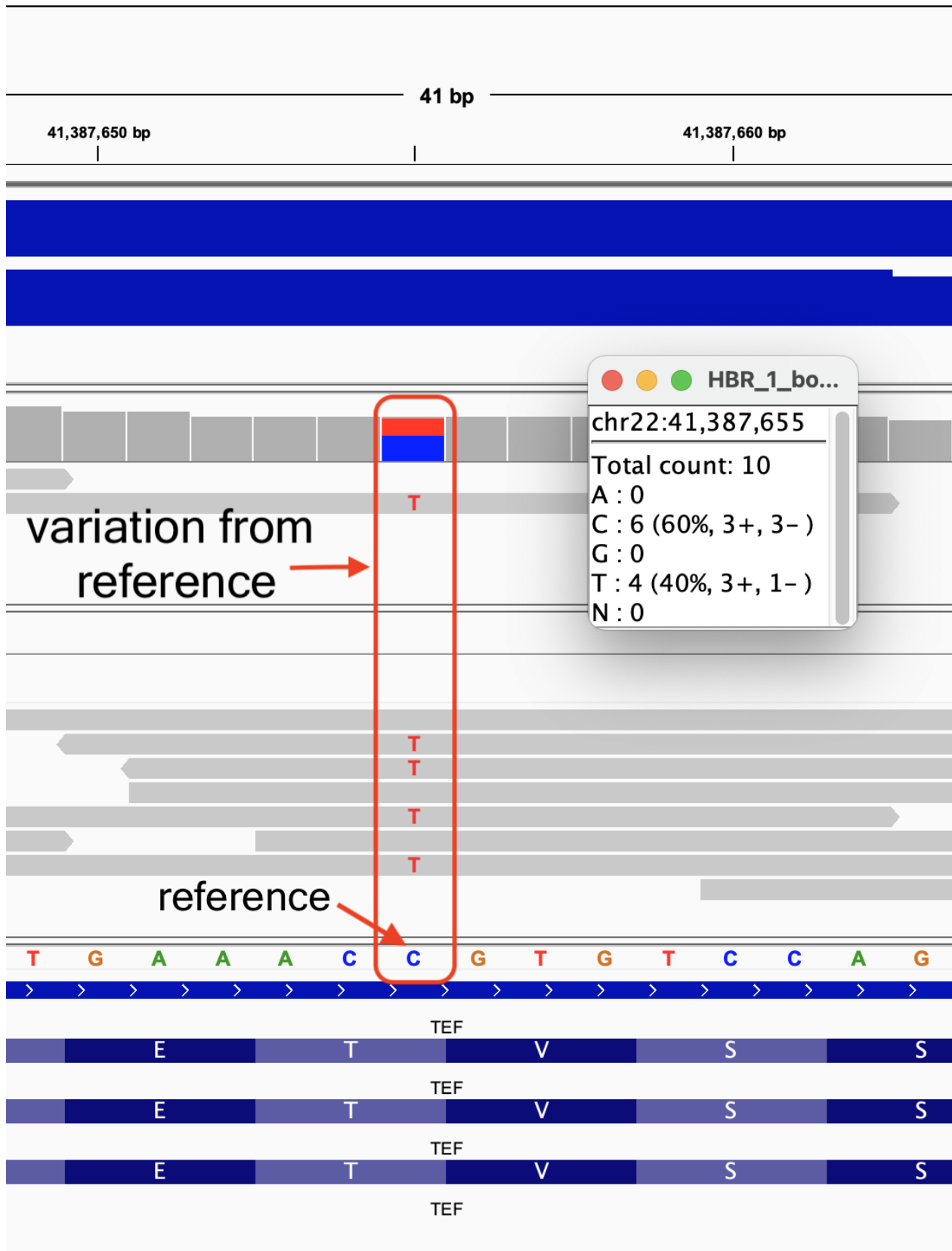


Figure 10

If we go back up to the "File" tab in the IGV menu bar, we can select to "Load from Server" (Figure 11) to bring in other information such as SNPs into IGV (Figure 12).

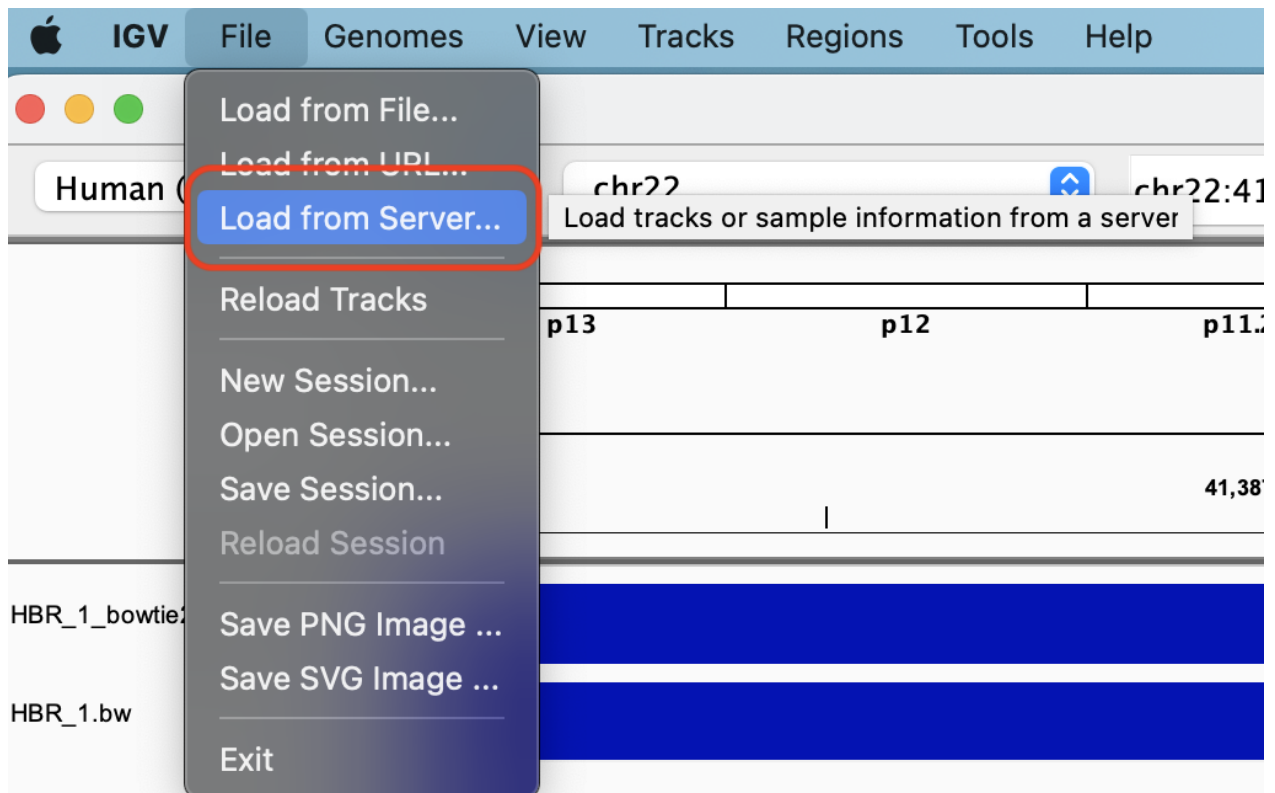


Figure 11

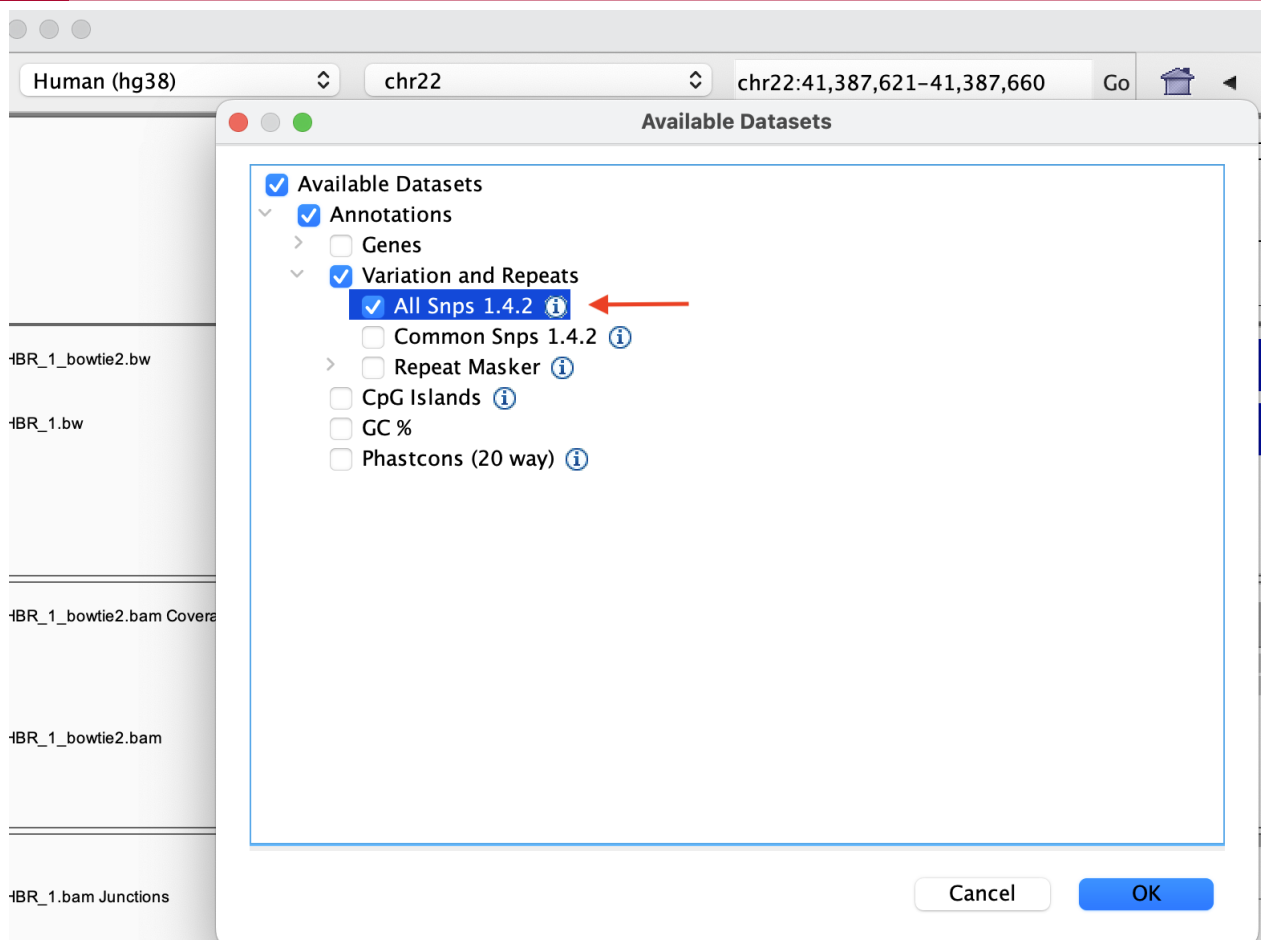


Figure 12

Once we click ok in the menu shown in Figure 12, we will see a SNP track beneath the bigWig tracks. Also, we will find that a SNP record aligns to the location where we found our potential variant.

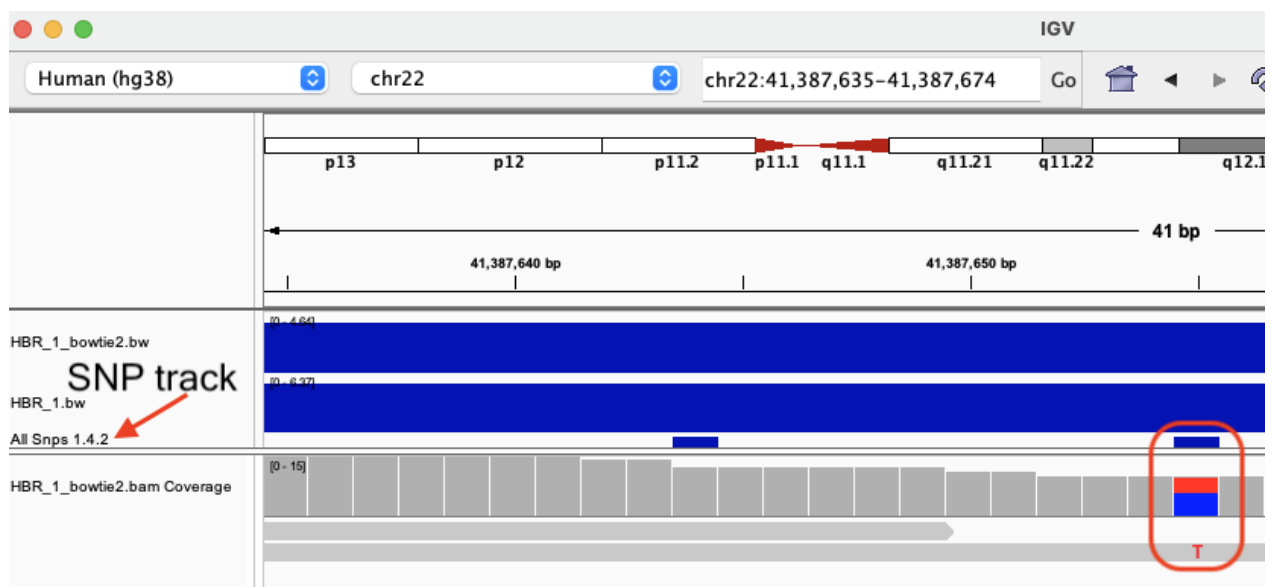


Figure 13

If we click on the SNP record, a dialog box containing more information about this variant appears.



Figure 14

Note that the presence of an "I" in the subject genome represents an insertion (click on it to see more details). In the example in Figure 15, we have a T insertion.

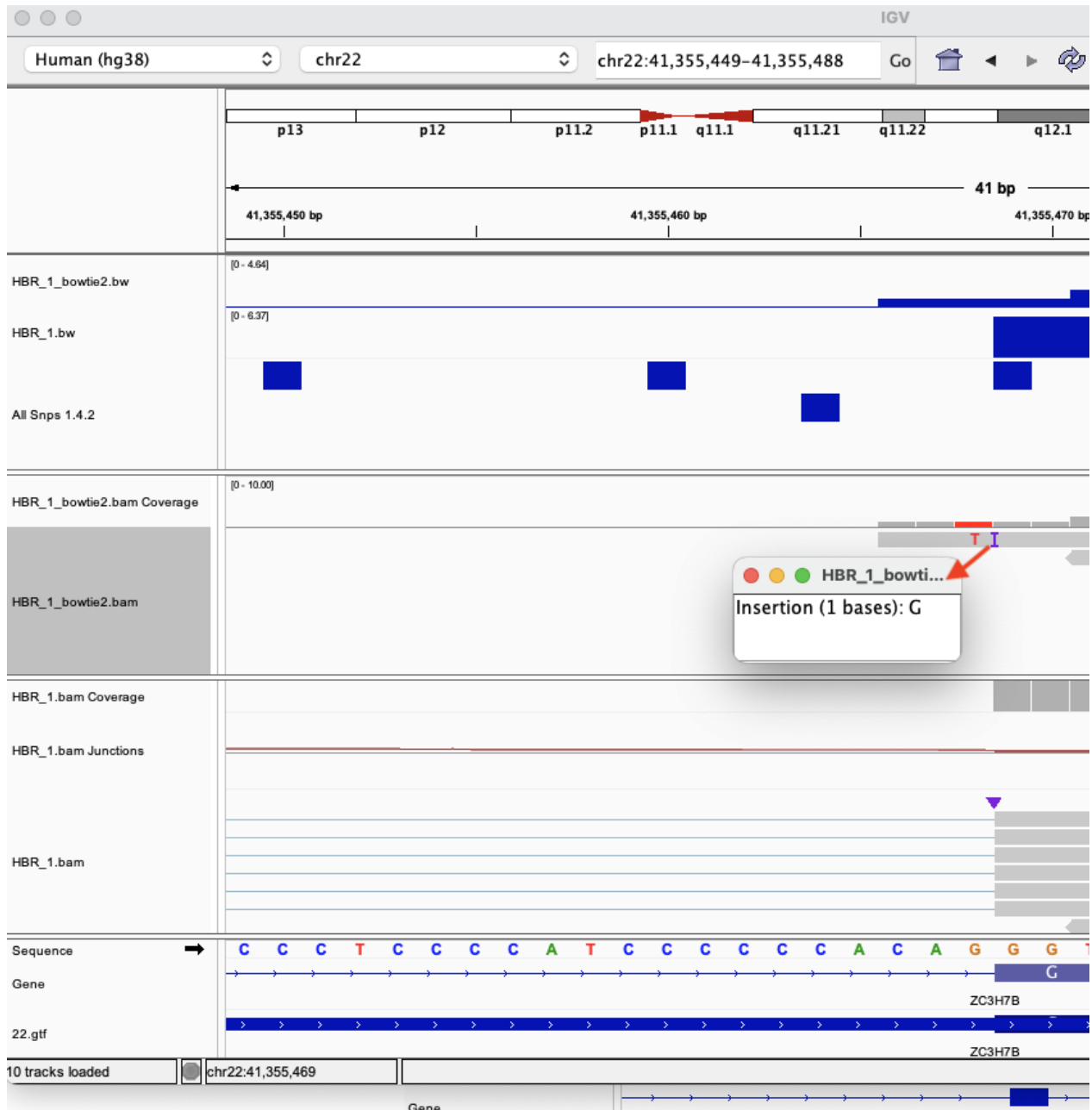


Figure 15

One of the things that we can do with IGV is to color or group reads according to certain criteria. For example, in paired end sequencing, the orientation of alignment for the pairs can alert us to potential structural variations so one of the things we can do is to go into our alignment (let's use the HBR_1 Bowtie2 alignment) by right clicking on the track, then select "Group alignments by" and then choose "pair orientation" (Figure 16).

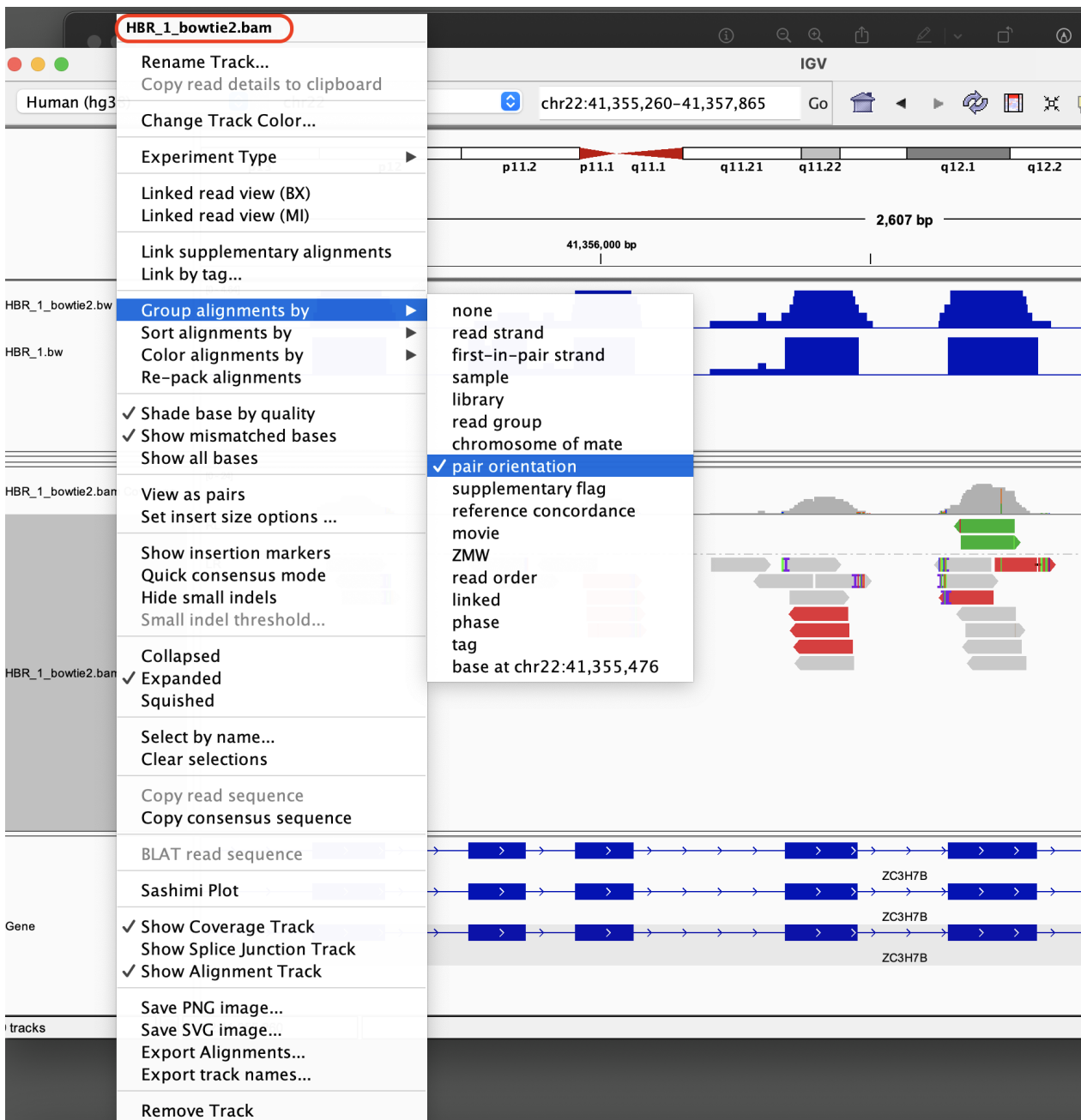


Figure 16

Grouping the alignments by pair orientation, we see that the alignment in the HBR_1_Bowtie2.bam track has been separated into two groups - a track labeled "RL" and one labeled "LR" (Figure 17). See https://software.broadinstitute.org/software/igv/interpreting_pair_orientations (https://software.broadinstitute.org/software/igv/interpreting_pair_orientations) for the definitions of the read pair orientations. In our example

- The alignments in the LR track are normal
- The alignments in the RL track suggest duplication or translocation when compared to the reference genome

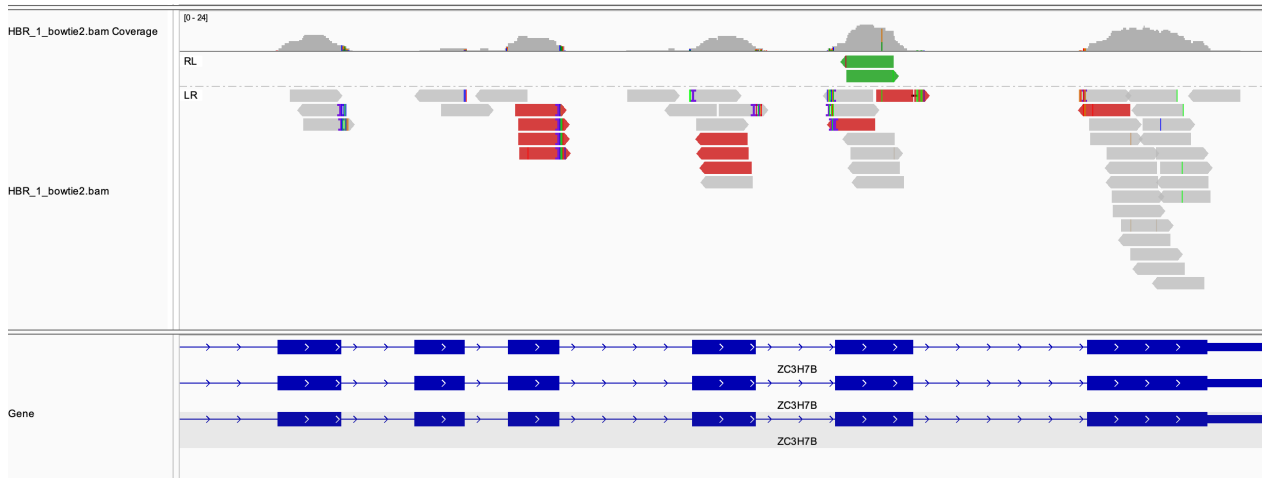


Figure 17

Lesson 15: Finding differentially expressed genes

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Lesson 14 review

In the previous lesson, we learned to visualize RNA sequencing alignment results in the Integrative Genome Viewer (IGV).

Learning objectives

In this lesson, we will identify genes that are differentially expressed between conditions. Here, we want to compare UHR samples to HBR samples, so we will be using existing applications to determine the ratio of expression of genes in the UHR samples versus the HBR samples (ie. UHR / HBR) and then we will get a metric to determine whether the expression change is statistically significant. Below are the two tasks for this lesson.

- Get expression counts for genes in the Human Brain Reference (HBR) and Universal Human Reference (UHR) dataset
- Complete our analysis of the HBR and UHR data by obtaining genes that are differentially expressed between the two groups
- Visualize gene expression

Before getting started, make sure that we are in the `~/biostar_class/hbr_uhr` directory.

```
cd ~/biostar_class/hbr_uhr
```

Obtaining expression read counts from alignment results

Let's talk about obtaining expression read counts using an application called `featureCounts`

First let's create a new directory in our `~/biostar_class/hbr_uhr` folder to store the counts.

```
mkdir hbr_uhr_deg_chr22
```

Next, change into the `~biostar_class/hbr_uhr_hisat2` folder

```
cd ~/biostar_class/hbr_uhr/hbr_uhr_hisat2
```

To run `featureCounts` we type in `featureCounts` at the command line followed by

- `-p`, specifies we want to count in the paired end mode since we are working with paired end sequencing
- `-a`, which prompts us to provide the annotation file (`22.gtf`)
- `-g`, which prompts us to specify attribute type in the GTF file (we choose to use `gene_names` so we can get expression by gene)
- `-o` prompts us to provide the output name
- finally, our input BAM files are provided

```
featureCounts -p -a ../refs/22.gtf -g gene_name -o ../hbr_uhr_deg_chr
```

After `featureCounts` finishes, we can go into the `hbr_uhr_deg_chr22` directory to see what we have.

```
cd ~/biostar_class/hbr_uhr/hbr_uhr_deg_chr22
```

```
ls -l
```

We have a file `hbr_uhr_chr22_counts.txt` with the expression counts and a summary. We will need the expression counts (ie. `hbr_uhr_chr22_counts.txt`) for differential expression analysis.

```
hbr_uhr_chr22_counts.txt
hbr_uhr_chr22_counts.txt.summary
```

If we take a look at the first two lines of `hbr_uhr_chr22_counts.txt`, we will see that `featureCounts` saves program information and command line call in the first line (we need to remove this line). The second line are the column headers. In the column headers, we do not need columns `Chr`, `Start`, `End`, `Strand`, and `Length`. All we want are the genes and expression for these genes in our samples.

```
head -2 hbr_uhr_chr22_counts.txt
```

```
# Program:featureCounts v2.0.1; Command:"featureCounts" "-a" "../ref:
Geneid Chr      Start  End      Strand  Length  HBR_1.bam      HBR_2
```

To remove the header line, we use the sed command below, where

- the option 1d indicates delete the first line (1 refers to the first line, d denotes delete)
- next we provide the input file
- we use ">" to indicate we want to save the output and then provide a name for the output (we will append "_no_header" to the original file name)

```
sed '1d' hbr_uhr_chr22_counts.txt > hbr_uhr_chr22_counts_no_header.txt
```

If we look at the first two rows of hbr_uhr_chr22_counts_no_header.txt, we should see the column headings and counts for the first gene. We still need to remove the columns Chr, Start, End, Strand, and Length. We will keep hbr_uhr_chr22_counts_no_header.txt because we need it for a downstream task.

```
head -2 hbr_uhr_chr22_counts_no_header.txt
```

Geneid	Chr	Start	End	Strand	Length	HBR_1.bam	HBR_2
U2	chr22	10736171		10736283		- 113	0

To remove the columns Chr, Start, End, Strand, and Length, we can use cut

- where -f1,7-12 tells the command that we want field 1 (the gene ids) and then fields 7-12, the expression counts for the samples
- the output from cut is then piped to tr '\t' ',', which will replace the original tab ('\t') separated columns with comma (',') separated columns. In other words, rather than using tabs to separate the columns of our counts table, we use commas, which is required for the differential expression analysis software. We can do tr ',' '\t' to go from comma separated columns to tab separated columns
- we save the new counts table as counts.csv

```
cut -f1,7-12 hbr_uhr_chr22_counts_no_header.txt | tr '\t' ',' > counts.csv
```

Now, we can use the column command to look at the HBR and UHR expression counts table. Hit q to exit the column command and return to the terminal.

- -t: identifies the number of columns in the input
- -s: tells column command that the columns in the input are separated by commas
- |: pipe or sends the output of column to less -S where -S allows us to scroll to see columns of the table that would not fit on the terminal

```
column -t -s ',' counts.csv | less -S
```

Geneid	HBR_1.bam	HBR_2.bam	HBR_3.bam	UHR_1.bam	UHR_2
U2	0	0	0	0	0
CU459211.1	0	0	0	0	0
CU104787.1	0	0	0	0	0
BAGE5	0	0	0	0	0

Normalization

Before diving into differential expression analysis, it is important to take a brief look at the concept of normalization. After obtaining expression counts, we will need to normalize the counts before performing differential expression analysis. This is an important step because normalization serves to remove technical variations amongst the samples to ensure that whatever difference we get in gene expression is due to biology.

Common sources of technical noise include the following and we would like to have them removed before continuing on with analysis.

- Differences in library size between the samples (ie. the sum of the counts between the samples are not the same). To understand this a bit better, think of a Western blot, where we have to load the same amount of protein or starting material (from each sample) into each lane in a gel so that we can compare protein expression.
- Gene length - the longer the gene the more reads it will get.
- Batch effect - when we process different samples on different days, locations, by different people etc. we are introducing technical noise.

While there are different approaches for normalization, we will not explore these today as it is a more advanced topic. Also, some of the differential expression analysis packages such as *DESeq2* (<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>) and *edgeR* (<https://bioconductor.org/packages/release/bioc/html/edgeR.html>) have their own normalization procedures and users are instructed to input only the raw integer counts when using these packages.

For now, we will use scripts written by the author of the Biostars handbook. These scripts can be run on the command line but they are R packages, so for those interested in learning R, please visit <https://btep.ccr.cancer.gov/on-line-classes-2022/> (<https://btep.ccr.cancer.gov/on-line-classes-2022/>) for our series of introductory R courses.

Differential expression analysis

Prior to differential expression analysis, we need to generate a design.csv file that contains the samples and their corresponding treatment conditions. Note that csv stands for comma separated value so the columns in these files are separated by commas. This file is already created for us and it lives in the design_file_hisat2 folder, which we should see in our home directory.

If we listed the contents of the design_file_hisat2 folder, we will see two files

```
ls ~/design_file_hisat2
```

```
design.csv  design.txt
```

Copy both files (design.csv and design.txt) to the ~/biostar_class/hbr_uhr/hbr_uhr_deg_chr22 folder, which should be the present working directory.

```
cp ~/design_file_hisat2/design.csv .
```

```
cp ~/design_file_hisat2/design.txt .
```

We will need the design.txt file when we do visualizations.

The design.csv file and design.txt file contain the same content, except the columns in the csv file are separated by commas and the txt file are separated by tabs. We can use cat to view these.

```
cat design.csv
```

```
sample,condition
HBR_1.bam,HBR
HBR_2.bam,HBR
HBR_3.bam,HBR
UHR_1.bam,UHR
UHR_2.bam,UHR
UHR_3.bam,UHR
```

```
cat design.txt
```

```
sample condition
HBR_1.bam      HBR
HBR_2.bam      HBR
HBR_3.bam      HBR
UHR_1.bam      UHR
UHR_2.bam      UHR
UHR_3.bam      UHR
```

To get differential expression for genes on chromosome 22, we will need make sure we are in ~/biostar_class/hbr_uhr/hbr_uhr_deg_chr22 folder (use pwd to confirm and cd into if needed) and run the deseq2.r script as shown below.

```
Rscript $CODE/deseq2.r
```

While the deseq2.r script is running, it will generate the following. Mainly, it's telling us that it is using the design.csv and counts.csv files as input and the output is stored in results.csv.

```
converting counts to integer mode
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
[1] "# Tool: DESeq2"
[1] "# Design:  design.csv"
[1] "# Input:   counts.csv"
[1] "# Output:  results.csv"
```

Since the differential analysis results are in csv format, we can open these in Excel. However, we can also view these in the terminal using the column command. And doing this we find that there are 13 columns in the differential analysis results table and they are described below. Hit q to get out of the column command.

```
column -t -s ',' results.csv | less -S
```

1. name: gene names.
2. baseMean: average expression across all samples (HBR+UHR).
3. baseMeanA: average expression across samples of treatment group A (HBR in this case).
4. baseMeanB: average expression across samples of treatment group B (UHR in this case).

5. foldChange: ratio of expression between UHR and HBR (ie. baseMeanB/baseMeanA)
6. log2FoldChange: log2 of the values in the foldChange column, this scales our fold change results uniformly. A fold change of 2 corresponds to a log2 fold change of 1, while a fold change of 1/2 corresponds to a log2 fold change of -1
7. lfcSE is the standard error of the log2 fold change.
8. stat: our test statistics, which is obtained by $\log_2\text{FoldChange}/\text{lfcSE}$ and is used to derive the statistical significance.
9. PValue: the uncorrected p-value of the likelihood of observing the effect of the size foldChange (or larger) by chance alone. This p-value is not corrected for multiple comparisons.
10. PAdj: Multiple comparison adjusted p-value.
11. FDR: the False Discovery Rate - this column represents the fraction of false discoveries for all the rows above the row where the value is listed. For example, if in row number 300 the FDR is 0.05, it means that if you cut the table at this row and accept all genes at and above it as differentially expressed then, $300 * 0.05 = 15$ genes out of the 300 are likely to be false positives. The values in this column are also called q-values.
12. falsePos: this column is derived directly from FDR and represent the number of false positives in the rows above.
13. The last six columns are normalized expression for each of the samples.

We can use the command below to sort by log2 fold change from largest to smallest, where

- column is used to print our tabular data nicely with columns aligned
- -t indicates to separate the columns by a tab
- -s ',' indicates that the columns in our file are originally separated by comma (ie. a csv file)
- sed 1q will print the first line of our table and then quits, preventing the first line from being included in the sort
- we use sort to sort things
 - -k: prompts us to provide column that we want to sort by, here it is column 6 (log2 fold change),
 - n indicates we want to sort numerically
 - r indicates in reverse order (from largest to smallest)

Hit q to exit and return to prompt

```
column -t -s ',' results.csv | (sed 1q; sort -k6nr) | less -S
```

name	baseMean	baseMeanA	baseMeanB	foldChange	log2F
PRAME	521.8	0	1043.6	4883.081	12.3
IGLC2	485.9	0	971.8	4538.584	12.1
IGLC3	386.2	0	772.5	3616.649	11.8
MY018B	63.8	0	127.7	595.615	9.2
PCAT14	145.5	0.8	290.3	375.805	8.6

CDC45	149.4	1.5	297.2	192.61	7.6
RP3-323A16.1	61	0.8	121.2	156.473	7.3
ZNF280A	10.1	0	20.3	94.53	6.6

Visualizing differential expression results

Plots condense complex and busy tabular data into a form that is easier to interpret. An expression heatmap is a common visualization used in RNA sequencing analysis. A heatmap shows numerical data on a color scale and is often combined with a dendrogram that shows how groups cluster. Here, we will generate an expression heatmap for HBR and UHR dataset using the script `create_heatmap.r`. The input for the script is `results.csv`, containing our differential gene expression findings generated by the `deseq2.r` script.

```
Rscript $CODE/create_heatmap.r
```

To view the expression heatmap, copy it to `~/public`. Go back to the GOLD landing page, click on the "Files" tab associated with your name and you will be taken to an index page. You can click on PDF files and view in them in the browser without having to download them.

```
cp heatmap.pdf ~/public/heatmap_hisat2.pdf
```

Figure 1 shows the expression heatmap for the chromosome 22 genes in the HBR and UHR samples. On the top of the plot, we have a color key, which indicates that down regulated genes have negative values and these are colored by shades of green and up regulated genes have positive values and these are colored by shades of red. The horizontal axis of the heatmap are labeled with the sample names, the right vertical axis are the genes, the left vertical axis is the dendrogram that shows clustering of genes based on expression. In this expression heatmap, it is clear that there are gene clusters that are up regulated or down regulated in one group but not the other.

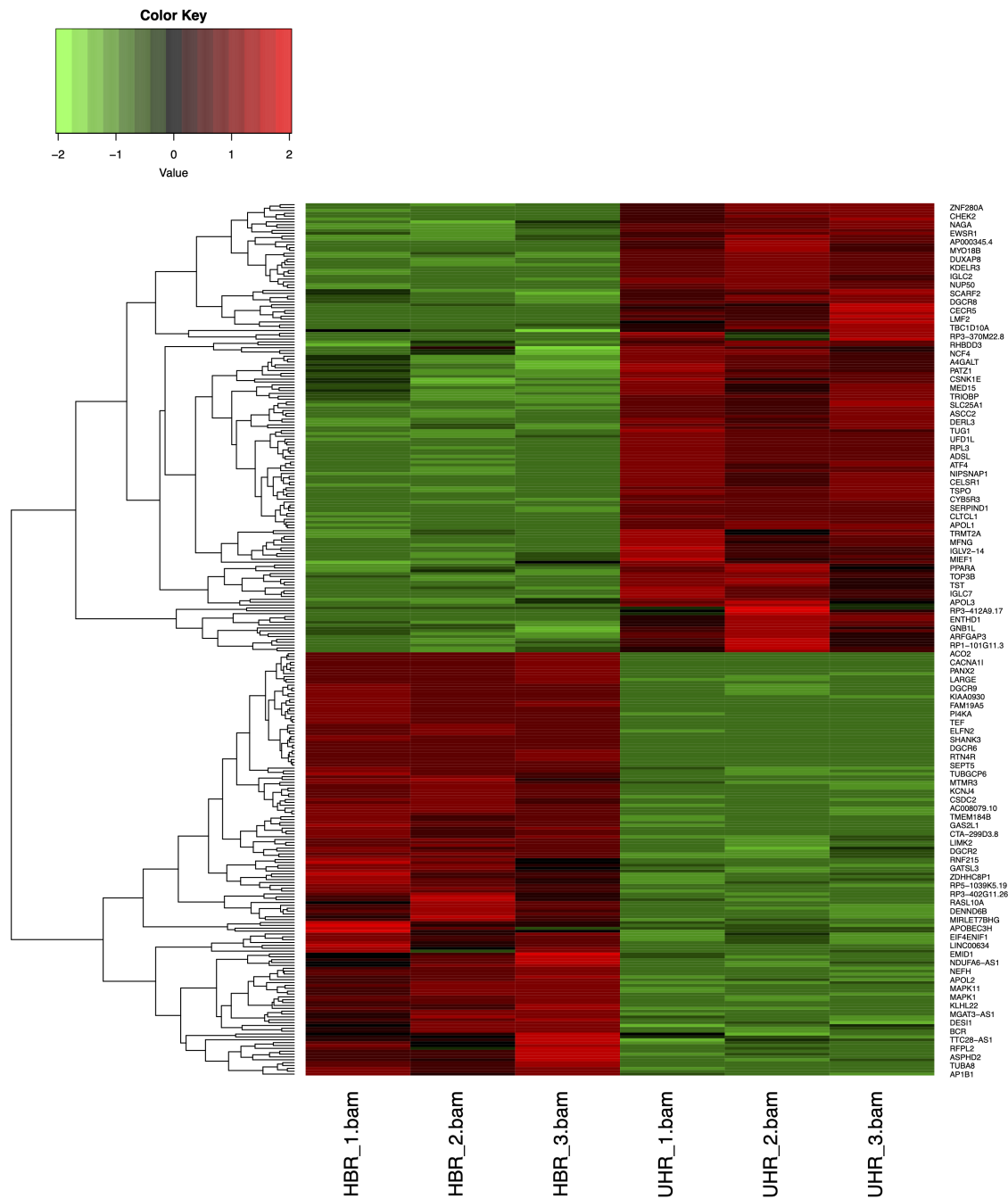


Figure 1

Another common visualization in RNA sequencing is Principal Component Analysis (PCA) plot. This helps us visualize clusters of samples in our data by transforming our data so that we can visualize along the axes that capture the largest variation in the data. We will use the `creat_pca.r` script and the command below to generate our PCA. The input for this script is our counts table (`counts.csv`) but we need a tab separated version of it. We also need `design.txt` (tab separated version of the design file) and the number of samples we have (6 in the case of the HBR and UHR dataset).

First use `cat` and `tr` to create tab separated counts table from `counts.csv`. The `tr` command replaces `,` with tabs (`\t`).

```
cat counts.csv | tr ',' '\t' > counts.txt
```

Then, we run the R script `create_pca.r` where the inputs are `counts.txt` (tab separated expression counts table), `design.txt`, and 6 (the number of samples).

```
Rscript $CODE/create_pca.r counts.txt design.txt 6
```

To view the PCA, copy it to `~/public`.

```
cp pca.pdf ~/public/pca_hisat2.pdf
```

The PCA plot is shown in Figure 2. The HBR samples are colored in red dots and UHR samples are colored in the green dots. The horizontal axis (PC1) is the one that captures the most variance or separation (79%) in our samples. PC2 on the vertical axis captures the second most variance or separation (7%). We see that along PC1, the HBR and UHR samples are clearly separated and we are able to see perhaps the biological difference between these samples. Along PC2, while the HBR samples cluster closely, we see that the UHR_2 sample is off by itself (away from the other two samples in this group). This could indicate some underlying biology of UHR_2 or maybe it's caused by some technical factor.

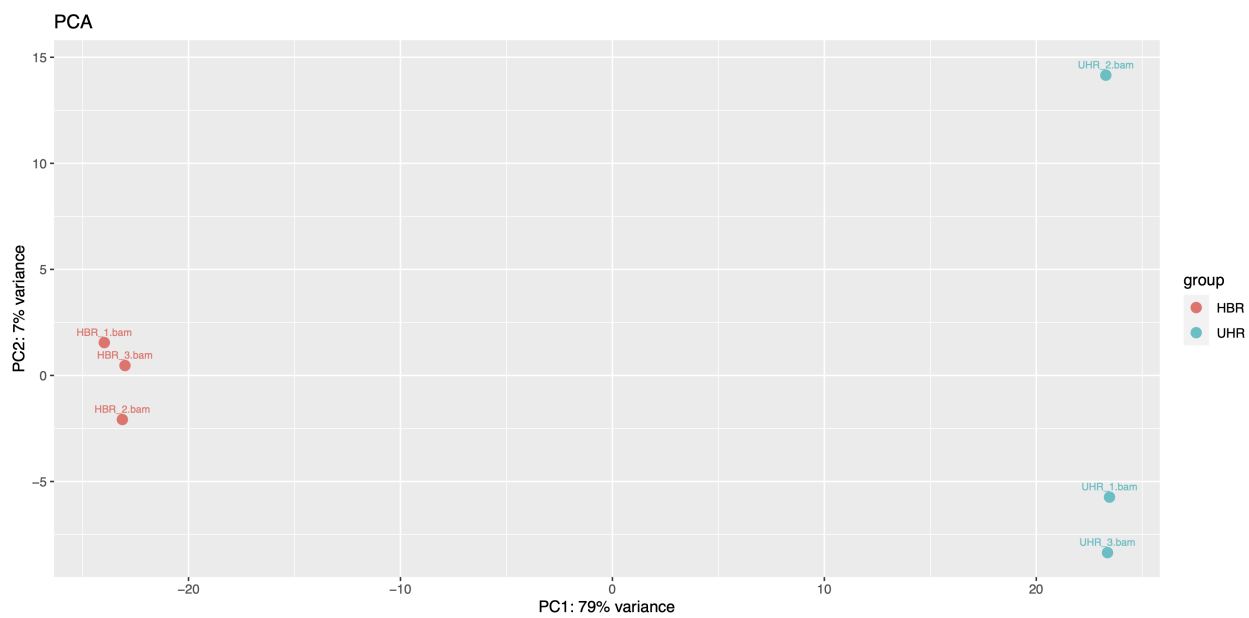


Figure 2

Lesson 16: RNA sequencing review and classification based analysis

Before getting started, remember to be signed on to the DNAnexus GOLD environment.

Review

In the previous classes, we learned about the steps involved in RNA sequencing analysis. We started off with assessing quality of raw sequencing data, then we aligned the raw sequencing data to genome, and finally we obtained expression counts and conducted differential expression analysis.

Tools for assessing sequencing data quality

- FASTQC to obtain quality metrics for individual FASTQ files. Recall that FASTQ files contain our sequencing data and each file has many sequencing reads. Each read is composed of four lines
 - Header, that starts with @
 - Actual sequence
 - "+"
 - Quality, which tells us the error likelihood of the base call
- MultiQC to aggregate multiple FASTQC outputs into one

Tools for sequencing data cleanup

When analyzing high throughput sequencing data, we will need to trim away adapters. Adapters help anchor the unknown sequencing template to the Illumina flow cell and can interfere with alignment. We may also want to trim away low quality reads. In this course, we learned to use *Trimmomatic*, which is a flexible trimming tool for Illumina data (<http://www.usadellab.org/cms/?page=trimmomatic>) to trim away low quality reads and adapters. Refer to the *Trimmomatic manual* (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf) and the *Trimmomatic publication* (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4103590/>) for details on how to use this tool.

Revisiting the construction of the trimmomatic command

1. Initiate with the command by typing trimmomatic at the prompt
2. Tell trimmomatic whether we are working with single end (SE) or paired end (PE) sequencing
3. Provide input files

4. Provide names of the output files including ones for unpaired reads (ie. those that survived the processing in one file but not the other)
5. Use SLIDINGWINDOW for quality trimming (see Figure 1)
 - specify window size (ie. how many bases the sliding window is composed of), in the example below we use a window size of 4 bases
 - specify the average quality score threshold for the window, in the example below we use 30
 - in the example below, we will scan starting from the 5' end of the read, 4 bases at a time and trim once the average quality within a 4-base window falls below 30
6. ILLUMINACLIP is used to trim away adapters and other Illumina-specific sequences, the parameters needed for ILLUMINACLIP are
 - FASTA file containing the adapter sequence
 - some numbers indicating a threshold on how Trimmomatic will determine whether adapter is present in a read (to put these numbers in context, think of how BLAST works), in the example below we use 2, 30, and 5
 - 2: seed mismatch threshold - "Specifies the maximum mismatch count which will still allow a full match to be performed. Short sections of each adapter (maximum 16 bp) are tested in each possible position within the reads. If this short alignment, known as the "seed" is a perfect or sufficiently close match, determined by the seed mismatch threshold, the entire alignment between the read and adapter is scored." -- *Trimmomatic manual* (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf)
 - 30: palindrome clip threshold - "Specifies how accurate the match between the two 'adapter ligated' reads must be for PE palindrome read alignment." -- *Trimmomatic manual* (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf)
 - 5: simple clip threshold - "Specifies how accurate the match between any adapter sequence must be against a read." -- *Trimmomatic manual* (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf)
 - Note that the above thresholds are essentially alignment scores (ie. reward for match and penalty for mismatch)
7. MINLEN is used to specify the minimum length of the trimmed sequence that we want to keep. In the example below, we set this to 50 bases. If a read falls below 50 bases, then it is dropped

Trimmomatic example: do not run this

```
trimmomatic PE SRR1553606_1.fastq SRR1553606_2.fastq SRR1553606_trimr
```

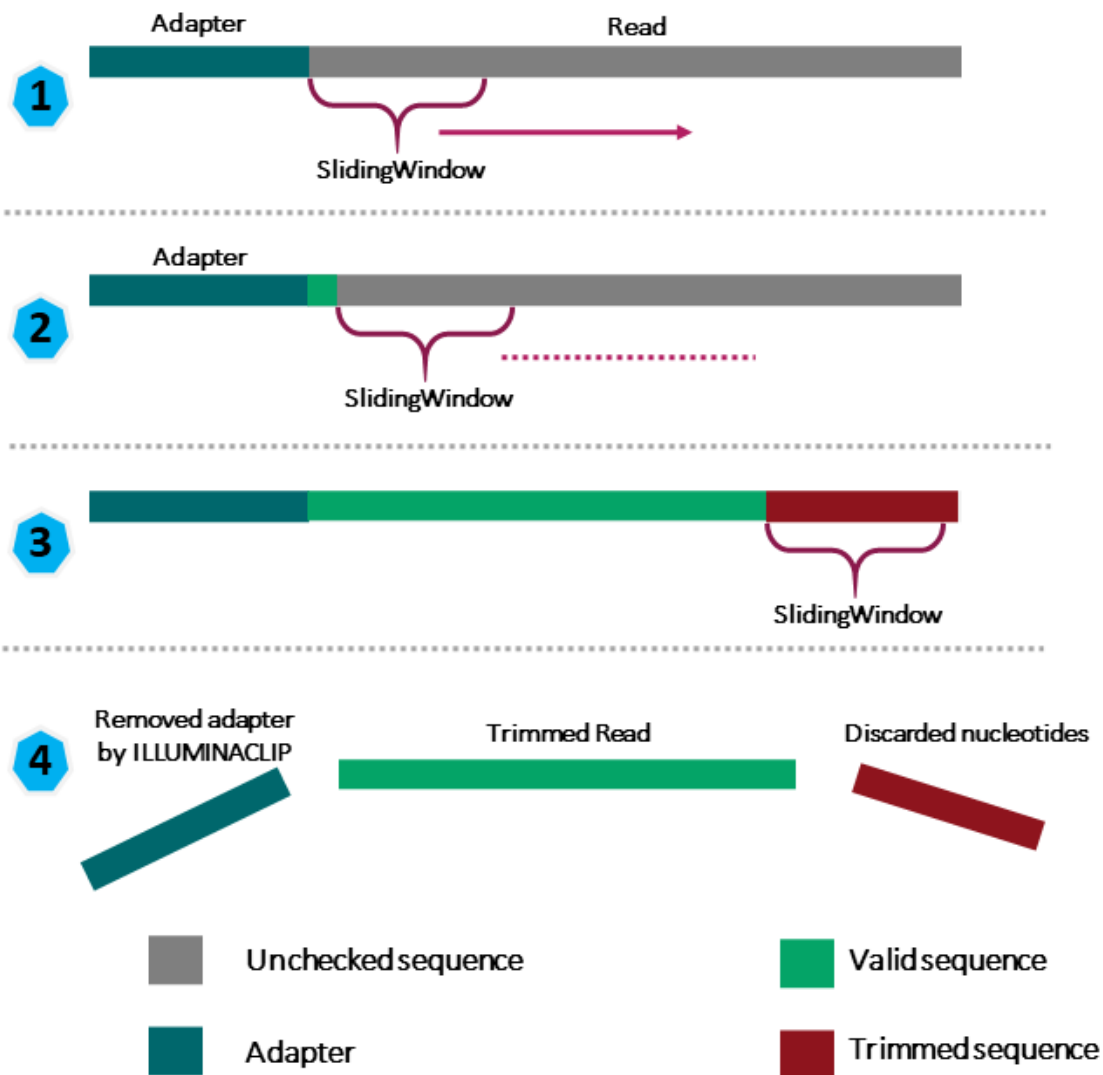


Figure 1: Source: <https://carpentries-incubator.github.io/metagenomics/03-trimming-filtering/index.html> (<https://carpentries-incubator.github.io/metagenomics/03-trimming-filtering/index.html>)

Additional resources for learning about Trimmomatic

<https://www.genepattern.org/modules/docs/Trimmomatic/#gsc.tab=0> (<https://www.genepattern.org/modules/docs/Trimmomatic/#gsc.tab=0>)

Trimmomatic before genome assembly (<https://youtu.be/nOJ5RGkKwnQ>)

FYI

Why are adapter sequences trimmed from only the 3' ends of reads? (<https://support.illumina.com/bulletins/2016/04/adapter-trimming-why-are-adapter-sequences-trimmed-from-only-the--ends-of-reads.html>)

We may also run FASTQC again after trimming to make sure that adapters have been removed and quality is good.

Tools for alignment

One of the challenges in analyzing high throughput sequencing is to reconstruct the genome of the unknown by using a known (ie. reference). The next step in analysis is to align our sequencing data to a reference genome. We used HISAT2 (splice aware) and visually compared the alignment results obtained from Bowtie2 (not splice aware) using the Integrative Genome Viewer (IGV). For RNA sequencing, we should use splice aware aligners to account for reads that map across two exons.

Obtaining expression read counts

After alignment of sequencing data to genome, we will need to count how many reads aligned to which gene. Using the tool featureCounts, we were able to do this. This tool takes as input our BAM alignment files and also an annotation file (gtf/gff) that tells us location of features (ie. genes, transcripts) in a genome. Note that for paired end sequencing we should include the additional flags below.

featureCounts version 2.0.1 (available on DNAnexus and Biostar environment on Biowulf)

-p: If specified, fragments (or templates) will be counted instead of reads. This option is only applicable for paired-end reads; single-end reads are always counted as reads.

featureCounts version 2.0.3 (default when using just Biowulf)

-p: Specify that input data contain paired-end reads. To perform fragment counting (ie. counting read pairs), the '--countReadPairs' parameter should also be specified in addition to this parameter.

--countReadPairs: Count read pairs (fragments) instead of reads. This option is only applicable for paired-end reads.

"For paired end reads, you should count read pairs (fragments) rather than reads because counting fragments will give you more accurate counts. There are several reasons why you cannot get the fragment counts by simply dividing the counts you got from counting reads by two. One reason is that a fragment with two mapped reads will give you two counts when you count reads, but a fragment with only one mapped read will only contribute one count (this fragment should get 1 count in fragment counting but it ended up with 0.5 count when you count reads instead of fragments). Another reason is that some reads may be found to be overlapping with more than one gene and therefore were not counted, but the corresponding fragments may be counted because the ambiguity was solved by longer fragment length." -- [Wei Shi \(https://support.bioconductor.org/p/67534/\)](https://support.bioconductor.org/p/67534/)

Differential expression analysis

After obtaining the expression counts for each gene, we can run helper R scripts provided by the author of the Biostar Handbook to obtain differential gene expression results. In our lessons, we used `deseq2.r`. We also generated visualizations that show us how genes cluster by expression (heatmap) and how samples cluster together (PCA).

Learning objectives

An alternative to aligning raw sequencing data to a reference genome is to map them to a reference transcriptome. In this lesson, we will use the HBR and UHR datasets, and learn about this approach for analyzing RNA sequencing data and discuss some advantages and drawbacks. We will do the following

- Construct the human chromosome 22 reference transcriptome using FASTA file for the chromosome 22 reference genome and gtf file
- Align sequencing reads to the reference transcriptome
- Obtain differential expression

Before getting started, be sure to change into the `~/biostar_class/hbr_uhr` directory.

```
cd ~/biostar_class/hbr_uhr
```

Constructing human chromosome 22 reference transcriptome

While we can always download reference genomes and reference transcriptomes from repositories such as NCBI or Ensembl, we will use `gffread` (<https://anaconda.org/bioconda/>)

gffread) to create one from the chromosome 22 genome (22.fa) that we have used when analyzing the HBR and UHR data via alignment to the genome.

Make a reference transcriptome using *gffread* with the following options:

- `-w` tells *gffread* to write a FASTA file with sequences of all exons from a transcript, which is followed by the output file name (in this example, we are storing the reference transcriptome, `22_transcriptome.fa` in the `refs` folder so we need to specify that path as well since we are currently in the `~/biostar_class/hbr_uhr` folder)
- `-W` tells *gffread* to include the exon coordinates in the header of each sequence (ie. the sequencing header that starts with `>`)
- `-g` prompts us to enter the reference genome FASTA file (`22.fa` in our case)
- the `gtf` (`22.gtf`) annotation file is provided at the end

```
gffread -w refs/22_transcriptome.fa -W -g refs/22.fa refs/22.gtf
```

Using `head` we can view the first few transcripts in human chromosome 22.

```
head refs/22_transcriptome.fa
```

Here, we can see that each transcript has a header line that starts with `>` followed by the actual sequence of the transcript. On the header line we have the transcript ID, which starts with ENST (they are from Ensembl) and genomic coordinates for the transcripts.

```
>ENST00000615943.1 loc:chr22|10736171-10736283|- exons:10736171-10736283|
ATCACTTCTCGGCCTTTTGGCTAAGATCAACTGTAGTATCTGTTGTTATTAATATAATATTGTATATT(
ACCAATTGTCAATACAAGGCTGTTTGTATCTGATATGAACCAA
>ENST00000618365.1 loc:chr22|10936023-10936161|- exons:10936023-10936161|
AGCATGCCAGTTAATTTGAAATTTTCAGATAAACAAATACTTTTTTTCAGTGTAAGTATATCCCATACA/
ATTTGGGACATGCTTATACTAAAATATTATTCCTTATTTATCTGAAATTGAAATTTAACTGGGTATTA(
>ENST00000623473.1 loc:chr22|11065974-11067346|- exons:11065974-11067346|
GCTGCAGGCAGTGTTCTTCTGTGTCTGCTCACCGAGCTGCTCCGAGCCCGGCTT
>ENST00000624155.1 loc:chr22|11066501-11068089|+ exons:11066501-11068089|
ATGGCAGCCGGAGCGGTTTTTCTGGCATTGTCTGCCAGCTGCTCCAAGCCAGACTGATGAAGGAGGA(
```

Alignment of HBR and UHR raw sequencing data to human chromosome 22 transcriptome

Before we can align the HBR and UHR raw sequencing data to human chromosome 22 transcriptome, we need to create an index of this transcriptome (like we did with the genome). This will make the alignment proceed more efficiently. To do this we will use the index option of *salmon* (<https://salmon.readthedocs.io/en/latest/salmon.html>) where

- -t prompts us to input the FASTA file corresponding to the reference transcriptome (22_transcriptome.fa)
- -i prompts us to specify the name of a folder that will contain the indexing output

```
salmon index -t refs/22_transcriptome.fa -i refs/22_transcriptome.idx
```

Let's create a folder, salmon to store our alignment outputs with salmon

```
mkdir salmon
```

After the index has been generated, we can use salmon quant with the options below to generate our expression counts table. Below is how we would construct the command for one sample, however, we have 6 samples in the HBR and UHR dataset so we can turn to the parallel command to align all of these at the same time.

- -l prompts us to specify the library type and specifying "A" will allow salmon to automatically infer library type (ie. if the library is paired end)
- --validateMappings helps to make alignment more accurate

Selective alignment, first introduced by the --validateMappings flag in salmon, and now the default mapping strategy (in version 1.0.0 forward), is a major feature enhancement introduced in recent versions of salmon. When salmon is run with selective alignment, it adopts a considerably more sensitive scheme that we have developed for finding the potential mapping loci of a read... -- <https://salmon.readthedocs.io/en/latest/salmon.html> (<https://salmon.readthedocs.io/en/latest/salmon.html>).

- For paired end sequencing, we specify read 1 and read 2 after the flags -1 and -2, respectively
- -o prompts us to specify a name of the folder where the output will be stored (we want our output to be stored in the folder salmon, which was created earlier)

```
salmon quant -i refs/22_transcriptome.idx -l A --validateMappings -1
```

Because we have 6 samples, we want to get Salmon to quantify them in one go so we will need to create a text file called `ids.txt` that contains the sample names for this dataset.

```
cd ~/biostar_class/hbr_uhr/reads
```

```
parallel echo {1}_{2} ::: HBR UHR ::: 1 2 3 > ids.txt
```

Now, go back up one directory to `~/biostar_class/hbr_uhr`

```
cd ..
```

To align multiple files, we will use the following command

```
cat reads/ids.txt | parallel "salmon quant -i refs/22_transcriptome."
```

Now if we change into the salmon directory and list the content, we should see the folders below, which contain the transcriptome alignment output for each sample in the HBR and UHR dataset (Salmon produces an output folder for each sample).

```
cd salmon
```

```
ls -l
```

```
HBR_1_SALMON  
HBR_2_SALMON  
HBR_3_SALMON  
UHR_1_SALMON  
UHR_2_SALMON  
UHR_3_SALMON
```

Let's take a look at the contents of the HBR_1 alignment results folder.

```
cd HBR_1_SALMON
```

```
ls -l
```

```
aux_info
cmd_info.json
libParams
lib_format_counts.json
logs
quant.sf
```

If we need to recall how we ran the salmon alignment, we can see this in `cmd_info.json`, where `cmd` stands for command line (so this file provides command line information).

```
cat cmd_info.json
```

```
"salmon_version": "1.7.0",
"index": "22_transcriptome.idx",
"libType": "A",
"validateMappings": [],
"mates1": "HBR_1_R1.fq",
"mates2": "HBR_1_R2.fq",
"output": "salmon/HBR_1_SALMON",
"auxDir": "aux_info"
```

If we look at the `salmon_quant.log` file in the `logs` directory, we can get some information such as overall alignment rate.

```
cd logs
```

```
cat salmon_quant.log
```

```
Mapping rate = 34.6147%
```

The expression counts are available in the file `quant.sf` so to take a look at the HBR_1 Salmon quantifications we need to go up one folder (ie. `~/biostar_class/hbr_uhr/salmon/HBR_1_SALMON`).

```
cd ..
```


Below, we use the column command to show the first few lines of column 4 in quant.sf file for HBR_1, which contains the count data.

- column is used to print our tabular data, which is quant.sf for HBR_1 nicely with columns aligned
 - -t finds the number of columns in the data
- we use | to pipe the column output to sed, where
 - 1q will print the first line of our table and then quit, preventing the first line from being included in the sort
 - then we use sort where
 - -k prompts us to specify the column we like to sort by (column 4 containing the count data in this case)
 - we want to sort column 4 numerically so we use n after the 4 to indicate this
 - we also want to sort column 4 from largest to smallest so we include r, with the final construct being "-k 4nr" for sorting column 4 numerically from largest to smallest

Hit q to get out of the column command and return to the prompt

```
column -t quant.sf | (sed 1q; sort -k 4nr) | head
```

The columns in the quant.sf Salmon output file are described below.

- Name: contains transcript IDs
- Length: transcript length (ie. number of bases in the transcript)
- EffectiveLength: the computed effective length of the target transcript. It takes into account all factors being modeled that will effect the probability of sampling fragments from this transcript, including the fragment length distribution and sequence-specific and gc-fragment bias (if they are being modeled) -- https://salmon.readthedocs.io/en/latest/file_formats.html (https://salmon.readthedocs.io/en/latest/file_formats.html)
- TPM: this stands for Transcripts Per Million (these are normalized expression and we discussed normalization in the previous lesson). Here TPM is way to scale the library size or total number of counts per sample to one million (ie. for a given sample, how many reads will fall onto a particular gene if the total reads we have is one million?)

Name	Length	EffectiveLength	TPM	NumReads
ENST00000365188.1	283	116.921	123216.364027	282.000
ENST00000248975.5	1825	1651.938	29248.111343	945.757
ENST00000401609.5	1289	1115.938	16045.787239	350.501
ENST00000417718.6	569	395.946	12253.336005	94.968
ENST00000216146.8	1442	1268.938	11781.878918	292.646
ENST00000396680.2	1419	1245.938	11320.546468	276.091
ENST00000215909.9	560	386.946	9409.708625	71.271

```
ENST00000396821.7 4791 4617.938 9210.768077 832.591
ENST00000328933.9 4391 4217.938 8779.491093 724.866
```

Let's now change back in the ~/biostar_class/hbr_uhr/salmon folder

```
cd ~/biostar_class/hbr_uhr/salmon
```

We will next combine the expression for all of the HBR and UHR samples into one csv file. To do this, we need a design.csv file like we did with the alignment to genome analysis. As a review, the design file has two columns, where one column provides the sample names, and the other informs of the condition with which the samples were treated. Fortunately, the design file has been created already and they should be in the folder design_file_salmon in our home directory. For now, we want design.csv.

```
ls ~/design_file_salmon
```

```
design.csv
design.txt
```

To work with the design.csv file, we will need to copy it to the ~/biostar_class/hbr_uhr/ folder (so let's change into this)

```
cd ~/biostar_class/hbr_uhr/
```

```
cp ~/design_file_salmon/design.csv .
```

To view the contents of the design.csv file, we can use cat

```
cat design.csv
```

```
sample,condition
HBR_1_SALMON,HBR
HBR_2_SALMON,HBR
HBR_3_SALMON,HBR
UHR_1_SALMON,UHR
UHR_2_SALMON,UHR
UHR_3_SALMON,UHR
```

Note that the sample names in the design file matches the names of the salmon alignment output folders.

Now that we have the design file, run the `combine_transcripts.r` script to get a table with the expression for all samples. This script takes as input the `design.csv` file and the folder that contains the salmon output for all of the samples (ie. the salmon directory).

```
Rscript $CODE/combine_transcripts.r design.csv salmon
```

```
[1] "# Tool: Combine transcripts"
[1] "# Sample:  design.csv"
[1] "# Data dir:  salmon"
reading in files with read_tsv
1 2 3 4 5 6
[1] "# Results:  counts.csv
```

Below we show the first 4 lines of the merged salmon counts table, using the `column` command, where

- `column` is used to print tabular data nicely aligned in our terminal
 - `-t` obtains the number of columns in our input file (which is `counts.csv`)
 - `-s` allows us to specify the column separator, which is a `,` because we are working with a csv or comma separated value file, where the columns in the file are separated by `,`
- we use `|` to send the output from `column` to `sed`, where
 - `1q` will print the first line of our table and then quit, preventing the first line from being included in the sort
 - then we use `sort` where
 - `-k` prompts us to specify the column we like to sort by (column 2 the HBR_1 counts)
 - we want to sort column 2 numerically so we use `n` after the 2 to indicate this
 - we also want to sort column 2 from largest to smallest so we include `r`, with the final construct being `"-k 2nr"` for sorting column 2 numerically from largest to smallest

Hit `q` to get out of the `column` command and return to the prompt

```
column -t -s ',' counts.csv | (sed 1q; sort -k 2nr) | head -n 4
```

```
ensembl_transcript_id  HBR_1_SALMON  HBR_2_SALMON  HBR_3_SALMON  UHR_
ENST00000255882.10    1091.338      1127.926      986.09         326
```

ENST00000248975.5	945.757	1254.448	1141.565	237
ENST00000396821.7	832.591	994.288	903.553	185

Obtaining differentially expressed genes

After the merged expression counts table has been created, we can proceed with differential expression analysis. Let's use DESeq2 again for this. But first, let's move counts.csv (the merged salmon expression table) to the folder salmon so we can keep all our salmon outputs in one place. Let's move the design.csv file as well.

```
mv counts.csv salmon/
```

```
mv design.csv salmon/
```

Change back into the salmon directory so we can run differential analysis.

```
cd salmon
```

```
Rscript $CODE/deseq2.r
```

```
converting counts to integer mode
estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
[1] "# Tool: DESeq2"
[1] "# Design:  design.csv"
[1] "# Input:   counts.csv"
[1] "# Output:  results.csv"
```

As we seen previously, the deseq2.r script writes the differential analysis output to a file called results.csv. To view the differential analysis results let's do the following. Hit q to exit column and return to the prompt.

- column is used to print tabular data nicely aligned in our terminal
 - -t obtains the number of columns in our input file (which is results.csv)

- `-s` allows us to specify the column separator, which is a `,` because we are working with a csv or comma separated value file, where the columns in the file are separated by `,`

```
column -t -s ',' results.csv | less -S
```

name	baseMean	baseMeanA	baseMeanB	foldChange	log2
ENST00000390323.2	1182.9	0	2365.7	11306.612	13.1
ENST00000328933.9	480.4	939.3	21.6	0.023	-5.4
ENST00000390325.2	234.8	0	469.6	2244.479	11.1
ENST00000329492.5	203.7	399.9	7.5	0.019	-5.8
ENST00000396425.7	431.4	822.7	40.2	0.049	-4.4
ENST00000454349.6	145.8	291.6	0	0.001	-10

Mapping transcript IDs to gene names

Note that we now have differential expression by transcripts and our first column contains the transcript IDs. But what genes do these transcripts map to? We will need to do some data wrangling to find out.

First let's sort the `results.csv` file by transcript ID and save it as `results_id_sorted.txt` to denote that this is a sorted version. To obtain the `results_id_sorted.txt` file, we

- use `grep` to find ENST, which is the prefix to the transcript ids in the `results.csv` file
- use `|` to send the `grep` results to `column` to generate a nicely aligned print out - note that in the `column` command, even though we use `-s` to specify that the columns in the input are comma separated, `column` will print the results to the terminal as a tab separated table
- use `|` to send the `column` output to `sort`
- finally, `>` save the sorted output as `results_id_sorted.txt`

```
grep ENST results.csv | column -t -s ',' | sort > results_id_sorted.txt
```

Next, we go back to the `~/biostar_class/hbr_uhr` folder.

```
cd ~/biostar_class/hbr_uhr
```

We have the `22.gtf` file that tells us the genes, transcripts, exons and other features that reside on human chromosome 22. There is tool called `gtfToGenePred` that can help us extract the transcripts and gene names from the `gtf` file.

We can pull up the documentation for gtfToGenePred if we just type "gtfToGenePred" in the command prompt.

```
gtfToGenePred - convert a GTF file to a genePred
usage:
  gtfToGenePred gtf genePred

options:
  -genePredExt - create a extended genePred, including frame
    information and gene name
  -allErrors - skip groups with errors rather than aborting.
    Useful for getting information about as many errors as possible
  -ignoreGroupsWithoutExons - skip groups contain no exons rather
    generate an error.
  -infoOut=file - write a file with information on each transcript
  -sourcePrefix=pre - only process entries where the source name is
    specified prefix. May be repeated.
  -impliedStopAfterCds - implied stop codon in after CDS
  -simple - just check column validity, not hierarchy, resulting
  -geneNameAsName2 - if specified, use gene_name for the name2 field
    instead of gene_id.
  -includeVersion - include gene_version and/or transcript_version attribute
    in the corresponding identifiers.
```

If you remember in 22.gtf, under the attribute column we have things like gene id, gene name, etc (see the table below). We will use the option geneNameAsName2 to pull the gene names. We will also be using the genePredExt option to create the genePred file.

CHROMOSOME	DATA SOURCE	FEATURE	START	END	SCORE	STRAND	FRAME	ATTRIBUTE
chr22	ENSEMBL	gene	10736171	10736283	.	-	.	gene_id "ENSG00000277 gene_type "snR gene_status "NO gene_name "U2 3;

CHROMOSOME	DATA SOURCE	FEATURE	START	END	SCORE	STRAND	FRAME	ATTRIBUTE
chr22	ENSEMBL	transcript	10736171	10736283	.	-	.	gene_id "ENSG0000027711"; transcript_id "ENST0000061511"; gene_type "snRNA"; gene_status "NOVEL"; gene_name "U2.14-201"; transcript_type "snRNA"; transcript_status "NOVEL"; transcript_name "U2.14-201"; level "basic"; transcript_support "NA";
chr22	ENSEMBL	exon	10736171	10736283	.	-	.	gene_id "ENSG0000027711"; transcript_id "ENST0000061511"; gene_type "snRNA"; gene_status "NOVEL"; gene_name "U2.14-201"; transcript_type "snRNA"; transcript_status "NOVEL"; transcript_name "U2.14-201"; exon_number 1; exon_id "ENSE000037361"; level 3; tag "basic"; transcript_support "NA";

Here, we will use the `genePredExt` and `geneNameAsName2` to get gene names included in our output.

Note that in the `gtfToGenePred` command below, we are saving the `genePred` output to the `refs` folder.

```
gtfToGenePred -genePredExt -geneNameAsName2 refs/22.gtf refs/22_exter
```

Taking a glance at the 22_extended.genePred file, we see that the first column has the transcript IDs, followed by strand and genomic coordinate information. The 12th column has the gene names. So columns 1 and 12 are what we want. The genePred format actually stands for gene prediction, which is exactly what it does, it tells us information about genes and their transcripts.

```
ENST00000615943.1 chr22 - 10736170 10736283 10736283 10736283
ENST00000618365.1 chr22 - 10936022 10936161 10936161 10936161
ENST00000623473.1 chr22 - 11065973 11067346 11065973 11067346
ENST00000624155.1 chr22 + 11066500 11068089 11066500 11068089
ENST00000422332.1 chr22 + 11124336 11125705 11125705 11125705
ENST00000614148.1 chr22 - 11253604 11253719 11253719 11253719
ENST00000621672.1 chr22 + 11456855 11456937 11456937 11456937
```

Here, we use `cut` to extract column 1 (transcript ID) and column 12 (gene name) of the genePred file and save it to 22_transcript_to_gene.txt.

```
cut -f1,12 refs/22_extended.genePred > refs/22_transcript_to_gene.txt
```

```
column -t refs/22_transcript_to_gene.txt | head
```

```
ENST00000615943.1 U2
ENST00000618365.1 CU459211.1
ENST00000623473.1 CU104787.1
ENST00000624155.1 BAGE5
ENST00000422332.1 ACTR3BP6
ENST00000612732.1 5_8S_rRNA
ENST00000614148.1 AC137488.1
ENST00000614087.1 AC137488.2
ENST00000621672.1 CU013544.1
```

Note that some genes may appear more than once because they can have multiple transcript products. As an example SLC25A17.

```
grep SLC25A17 refs/22_transcript_to_gene.txt
```



```
ENST00000263255.10 SLC25A17
ENST00000491545.5 SLC25A17
ENST00000435456.6 SLC25A17
ENST00000544408.5 SLC25A17
ENST00000402844.7 SLC25A17
ENST00000447566.5 SLC25A17
ENST00000420970.5 SLC25A17
ENST00000430221.5 SLC25A17
ENST00000427084.5 SLC25A17
ENST00000458600.5 SLC25A17
ENST00000443810.5 SLC25A17
ENST00000412879.5 SLC25A17
ENST00000426396.5 SLC25A17
ENST00000434193.5 SLC25A17
ENST00000478550.1 SLC25A17
ENST00000449676.5 SLC25A17
ENST00000434185.1 SLC25A17
```

Now let's sort `22_transcript_to_gene.txt`, which contains our transcript ID to gene name mapping by transcript ID and save it as `22_transcript_to_gene_id_sorted.txt` to denote that it is sorted.

```
cat refs/22_transcript_to_gene.txt | sort > refs/22_transcript_to_gene_id_sorted.txt
```

Finally, we can change back into the salmon output directory (`cd salmon`) to paste (the command that we will use is called `paste`) the sorted transcript ID to gene name mapping file (`22_transcript_to_gene_id_sorted.txt`) to the sorted differential expression analysis results (`results_id_sorted.txt`) and save as `results_with_gene_names.txt`.

```
cd salmon
```

The input for the `paste` command below are the two files that we want to paste together.

```
paste ../refs/22_transcript_to_gene_id_sorted.txt results_id_sorted.txt > results_with_gene_names.txt
```

Again, to view `results_with_gene_names.txt` we can use the `column` command (note that in the `column` command below, we did not specify the column separator because they are already separated by tabs)

```
column -t results_with_gene_names.txt | less -S
```

We should also insert column headers. To this use the sed command where inside the single quotes, "1i" tells it to insert into the first line the text that follows. Because results_with_gene_names.txt is tab separated, we would need to separate the column header by tabs using "\t" following each heading.

```
sed '1i TRANSCRIPT_NAME\tGENE_NAME\tTRANSCRIPT_ID\tbaseMean\tbaseMear
```

The sed command is known as a stream editor. It has many functions including printing of specific lines, deletion of lines, substitutions, and inserstion of lines.

Interpretation and comparison to alignment based RNA sequencing

Let's now take a look at our final differential analysis results table (results_with_gene_names_labeled.txt), using the SLC2A11 gene as an example and below we

- use the column command to print the contents of results_with_gene_names_labeled.txt nicely aligned in the terminal where -t finds out how many columns are in the input.
- use | to send the column output to sed where
 - 1q prints the first line and quits
 - then we grep to search for SLC2A11
- use | to send the output to another sed where
 - 1q prints the first line and quits, thus preventing the first line from being sorted
 - use -k to denote we want to sort by a specific column (column 8, which is log2FoldChange)
 - n to sort column 8 numerically and r to sort from largest to smallest
- less -S allows us to scroll through the table horizontally as well as vertically

```
column -t results_with_gene_names_labeled.txt | (sed 1q; grep SLC2A11:
```

Below, we see the transcripts derived from SLC2A11. This is where classification based analysis is beneficial - it allows us to look at transcript level expression differences between conditions such that even though we are talking about the same gene, different transcript isoforms maybe expressed under different conditions or between tissue types. On the other hand, in alignment based analysis, we are aligning everything to the genome so we are getting aggregate gene level expression information.

TRANSCRIPT_NAME	GENE_NAME	TRANSCRIPT_ID	baseMean
ENST00000611880.4	SLC2A11	ENST00000612482.4	12.5
ENST00000418102.5	SLC2A11	ENST00000418170.5	11.9
ENST00000403208.7	SLC2A11	ENST00000403222.7	3.3

ENST00000473357.1	SLC2A11	ENST00000473487.6	2.6
ENST00000423972.5	SLC2A11	ENST00000424008.1	0.6
ENST00000489322.5	SLC2A11	ENST00000489424.5	0.6
ENST00000482576.1	SLC2A11	ENST00000482652.1	7.2
ENST00000436643.5	SLC2A11	ENST00000436663.1	0.3
ENST00000405847.5	SLC2A11	ENST00000405878.5	348.2
ENST00000405286.6	SLC2A11	ENST00000405309.7	17.7
ENST00000255830.7	SLC2A11	ENST00000255830.7	0
ENST00000316185.8	SLC2A11	ENST00000316185.8	0
ENST00000440493.5	SLC2A11	ENST00000440562.1	0
ENST00000461809.5	SLC2A11	ENST00000461833.1	0
ENST00000467660.5	SLC2A11	ENST00000467672.5	0
ENST00000472526.1	SLC2A11	ENST00000472575.5	0
ENST00000492516.1	SLC2A11	ENST00000492538.1	0
ENST00000405340.6	SLC2A11	ENST00000405409.6	460.8
ENST00000473740.5	SLC2A11	ENST00000473782.1	0.2
ENST00000486907.1	SLC2A11	ENST00000487165.5	5.6
ENST00000491948.5	SLC2A11	ENST00000491967.1	1.6
ENST00000345044.10	SLC2A11	ENST00000345044.10	19

A drawback to the classification based approach is that we are mapping to a known database of transcripts. This may prevent us from discovering expression of novel splice isoforms.

Visualizing gene expression

Let's now create a gene expression heatmap for results generated using the classification based approach. We have our results.csv and design.csv file in our salmon directory so we just need to do the following to create the heatmap.

```
Rscript $CODE/create_heatmap.r
```

To view the heatmap, copy it to the ~/public directory.

```
cp heatmap.pdf ~/public/hbr_uhr_heatmap_salmon.pdf
```

To generate the PCA plot, we need a tab separated version of the design file, which we can copy over from ~/biostar_class/design_file_salmon folder. Since we are in ~/biostar_class/hbr_uhr/salmon we can use "." to denote copy to here, our present working directory.

```
cp ~/design_file_salmon/design.txt .
```

Let's use `cat` to view `design.txt`. Again, the difference between `design.csv` and `design.txt` is that the columns are separated by commas in the csv file and by tabs in the txt file.

```
cat design.txt
```

```
sample condition
HBR_1_SALMON   HBR
HBR_2_SALMON   HBR
HBR_3_SALMON   HBR
UHR_1_SALMON   UHR
UHR_2_SALMON   UHR
UHR_3_SALMON   UHR
```

To generate the PCA plot, we will also need a tab separated version of the counts table, which we can generate using the code below where

- `cat` is used to print contents of a file (ie. `counts.csv`)
- `tr` is used to replace commas (denoted by ',') in this file with tabs (denoted by '\t')

```
cat counts.csv | tr ',' '\t' > counts.txt
```

Then, we run the `create_pca.r` script like we did with the alignment based analysis method.

```
Rscript $CODE/create_pca.r counts.txt design.txt 6
```

To view the `pca`, copy it to the public directory.

```
cp pca.pdf ~/public/hbr_uhr_pca_salmon.pdf
```

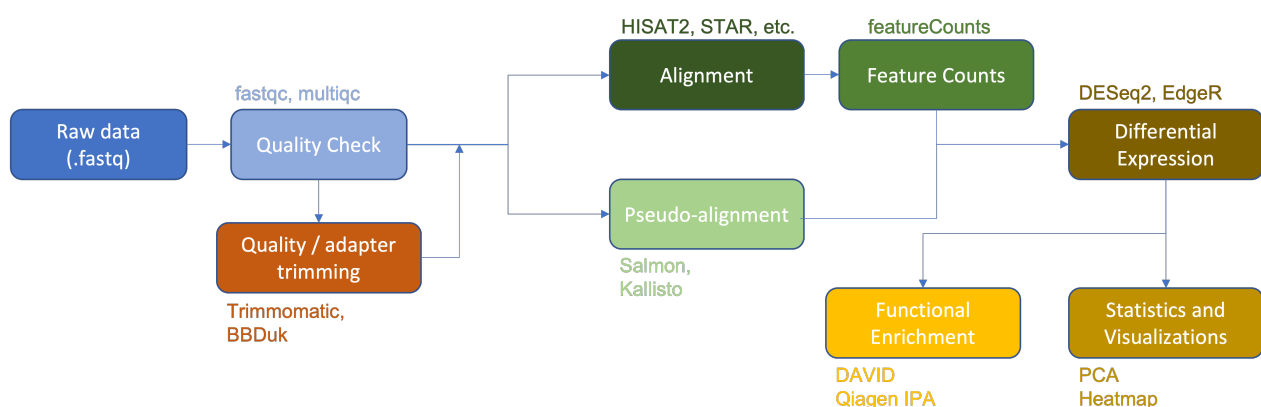
**Module 3 - Pathway
analysis and course
review**

Gene ontology and pathway analysis

Objectives

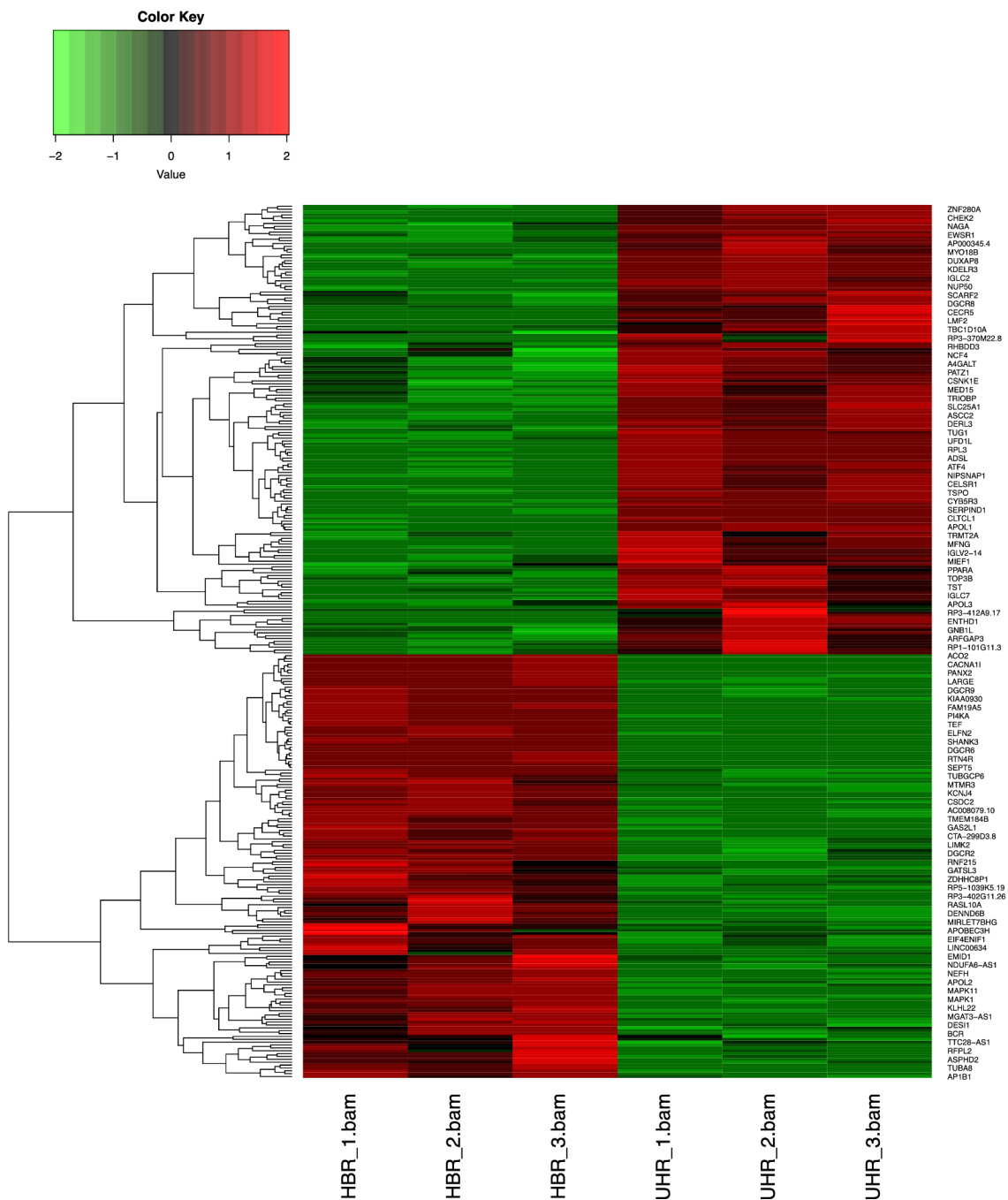
1. Determine potential next steps following differential expression analysis.
2. Tour geneontology.org and understand the three main ontologies.
3. Learn about different methods and tools related to functional enrichment and pathway analysis.
4. Get familiar with databases commonly used by popular functional enrichment tools.

Where have we been and where are we going?



Thus far we have:

1. Downloaded raw RNA-Seq data (.fastq files).
2. Examined raw data quality using `fastqc` and `multiqc`.
3. Performed adapter and quality trimming using `Trimmomatic`.
4. Aligned the raw sequences to a reference genome (human chromosome 22 from the GRCh38 version of the human reference genome) using `HISAT2`.
5. Viewed and compared alignments using `IGV`.
6. Generated a gene count matrix using `featureCounts`.
7. Performed differential expression analysis (`DESeq2`, `EdgeR`).
8. Generated a heatmap of differentially expressed genes.



Heatmap of differentially expressed genes (HBR vs UHR).

You now have a potentially large list of differentially expressed genes. Now what? If you are like most biologists, you are interested in understanding these genes within their biological context.

To do that, we can examine gene ontology and perform some type of functional enrichment analysis or pathway analysis.

These types of analyses exploit the use of gene sets, and not all gene sets represent a pathway. Gene sets, which are collections of genes "formed on the basis of shared biological or functional properties as defined by a reference knowledge base. Knowledge bases are

database collections of molecular knowledge which may include molecular interactions, regulation, molecular product(s) and even phenotype associations" Mathur et al. 2018 (<https://biodatamining.biomedcentral.com/articles/10.1186/s13040-018-0166-8>).

Whereas, a pathway is not a simple list of genes but rather includes an interaction component usually related to a specific mechanism, process, etc.

What is gene ontology?

Many of the tools used to understand functional enrichment will use sets of GO terms, examining GO enrichment. What do we mean by GO?

The Gene Ontology (GO) provides a framework and set of concepts for describing the functions of gene products from all organisms. --- <https://www.ebi.ac.uk/ols/ontologies/go> (<https://www.ebi.ac.uk/ols/ontologies/go>).

This is manually curated by team members of the GO consortium.

There are two parts to the gene ontology: (Check out <https://www.youtube.com/watch?v=6Am2VMbyTm4> (<https://www.youtube.com/watch?v=6Am2VMbyTm4>) for a more detailed overview)

1. the ontology (the GO terms and their hierarchical relationship) - form a directed, acyclic graph structure (nodes = GO terms, edges = relationships)
2. the annotations (the annotated genes linked to various GO terms)

Approaches to gene set analysis / pathway analysis?

Functional enrichment and pathway analysis have broad and varying definitions. For our purposes, there are three general approaches: 1. Over-Representation Analysis (ORA), 2. Functional Class Scoring (FCS), and 3. Pathway Topology (PT) (Khatri et al. 2012 (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375#pcbi-1002375-t001>)).

Examining genes in a set allows us to:

1. increase the statistical power in our analysis
2. ease interpretation
3. predict new roles for genes
4. better integrate data from different methods

Over-representation analysis (ORA)

statistically evaluates the fraction of genes in a particular pathway found among the set of genes showing changes in expression --- Khatri et al. 2012 (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375#pcbi-1002375-t001>)

From this, ORA determines which pathways are over or under represented by asking "are there more annotations in the gene list than expected?"

- tests based on hypergeometric, chi-square, or binomial distribution
- includes GO enrichment methods
- prioritizes a subset of genes using an arbitrary, user determined threshold
- doesn't require the data, just the gene identifiers
- Example tools include DAVID and Qiagen IPA

Functional Class Scoring (FCS)

Includes 'gene set scoring' methods such as GSEA, which first compute DE scores for all genes measured, and subsequently compute gene set scores by aggregating the scores of contained genes. --- Geistlinger et al. 2021 (<https://academic.oup.com/bib/article/22/1/545/5722384?login=true>)

GSEA

- ignore gene position and role
- do not pre-select genes (considers all gene expression) and so you must include data with gene identifiers for ranking
 - ranking by magnitude of change in gene expression between conditions
 - determines where genes from a gene set fall in the ranking
 - creates a running sum statistic

- uses a permutational approach to determine significance
- Broad Institute software but also available using web-based tools, R, and Qlucore (proprietary).
- also considered a strategy encompassing a range of methods
 - self-contained methods vs competitive methods

What is MSigDB (<https://www.gsea-msigdb.org/gsea/msigdb/index.jsp>) and how does it relate to GSEA?

- curated by the Broad Institute
- 33196 gene sets to be used in GSEA (not all gene sets are related to pathways)
 - larger, themed collections
- human and mouse

Pathway Topology

ORA and FCS discard a large amount of information. These methods use gene sets, and even if the gene sets represent specific pathways, structural information such as gene product interactions, positions of genes, and types of genes is completely ignored. Pathway topology methods seek to rectify this problem.

PT methods are mostly considered network based.

Some examples:

Impact analysis (iPathwayGuide)

constructs a mathematical model that captures the entire topology of the pathway and uses it to calculate a perturbation for each gene. Then, these gene perturbations are combined into a total perturbation for the entire pathway and a p-value is calculated by comparing the observed value with what is expected by chance. (<https://advaitabio.com/ipathwayguide/more-accurate-pathway-rankings-using-impact-analysis-instead-of-enrichment/>)

- Other tools include Pathway-Express, SPIA, NetGSA, etc. (See [Nguyen et al. 2019](https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1790-4) (<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1790-4>) for a review of PT methods.)

What tools are available?

This is neither a comprehensive list of tools nor an endorsement of certain tools, but rather a list of semi-popular tools with different approaches. Note: there are a ton of tools out there. Be aware of the background methods used and the quality of results returned.

Gene set analysis tools

Tool	Author	Year	Citations ¹	Availability	Gene sets	Methods ²
WEBGESTALT	Zhang <i>et al.</i> [73]	2005	1423	Web server	GO, KEGG, +20 more	ORA, GSEA
GOSTATS	Falcon and Gentleman [74]	2007	1437	R package	GO	ORA
G:PROFILER	Reimand <i>et al.</i> [75]	2007	534	Web server	GO, KEGG, +7 more	ORA
GENETRAIL	Backes <i>et al.</i> [76]	2007	360	Web server	GO, KEGG, +28 more	ORA, GSEA
DAVID	Huang <i>et al.</i> [8]	2009	19 569	Web server	GO, KEGG, +38 more	ORA
GORILLA	Eden <i>et al.</i> [77]	2009	1881	Web server	GO	ORA
TOPPGENE	Chen <i>et al.</i> [78]	2009	1200	Web server	GO, KEGG, +45 more	ORA
CLUSTER-PROFILER	Yu <i>et al.</i> [10]	2012	1305	R package	GO, KEGG, +8 more	ORA, GSEA
PANTHER	Mi <i>et al.</i> [79]	2013	1405	Web server	GO, +2 more	ORA, GSEA
ENRICHR	Chen <i>et al.</i> [9]	2013	1246	Web server	GO, KEGG, +33 more	ORA

¹Google Scholar, July 2019.

²Detailed summary of implemented methods in [Supplementary Methods S1.2](#).

Table from Geistlinger *et al.* 2020 (<https://academic.oup.com/bib/article/22/1/545/5722384>)

Other and related tools

- Gene Set Enrichment Analysis (GSEA) (<http://software.broadinstitute.org/gsea/>)
- EnrichmentMap (<https://apps.cytoscape.org/apps/enrichmentmap>)
- REVIGO (<http://revigo.irb.hr/>) (reducing and visualizing gene ontology)
- Pathview (<https://pathview.uncc.edu/>)
- iPathwayGuide (<https://advaitabio.com/ipathwayguide/>) (proprietary)
- Qiagen IPA (<https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/ipa/>) (proprietary, CCR license)
- Qlucore (<https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/qlucore/>) (proprietary, CCR license)
- GeneMANIA (<https://apps.cytoscape.org/apps/genemania>)
- CellNetAnalyzer (<http://www2.mpi-magdeburg.mpg.de/projects/cna/cna.html>)
- PARADIGM (<http://paradigm.five3genomics.com>)

Other databases

There are many databases devoted to relating genes and gene products to pathways, processes, and other phenomenon. Again, the following is not meant to be a comprehensive list.

Kyoto Encyclopedia of Genes and Genomes (KEGG) (<https://www.genome.jp/kegg/>)

- curated database
- biological pathways
- molecular interaction networks
- Very nice pathway maps
- Restricted licenses

Reactome (<https://reactome.org/>)

The Reactome Knowledgebase systematically links human proteins to their molecular functions, providing a resource that functions both as an archive of biological processes and as a tool for discovering novel functional relationships in data such as gene expression studies or catalogs of somatic mutations in tumor cells. --- Jassal et al. 2019 (<https://academic.oup.com/nar/article/48/D1/D498/5613674>)

- curated database including metabolism, signaling, and other biological processes
- human specific
- also includes disease superpathways
- built-in pathway analysis tool

Pathway Commons (<https://www.pathwaycommons.org/>)

- a meta-database of pathways from other pathway databases
- standardized format

PANTHER (<http://www.pantherdb.org/pathway/>)

The PANTHER (Protein ANALYSIS THrough Evolutionary Relationships) Classification System was designed to classify proteins (and their genes) in order to facilitate high-throughput analysis. The core of PANTHER is a comprehensive, annotated “library” of gene family phylogenetic trees. --- pantherdb.org/about.jsp (<http://pantherdb.org/about.jsp>)

- database of signaling pathways

WikiPathways (<https://www.wikipathways.org/index.php/WikiPathways>)

- community driven meta-database of pathways

NDEX (<https://home.ndexbio.org/index/>)

The NDEX Project provides an open-source framework where scientists and organizations can store, share, manipulate, and publish biological network knowledge. - [ndexbio.org](https://home.ndexbio.org/about-ndex/) (<https://home.ndexbio.org/about-ndex/>)

HumanCyc (<https://humancyc.org/>) (See BioCyc)

HumanCyc provides an encyclopedic reference on human metabolic pathways, the human genome, and human metabolites. --- humancyc.org (<https://humancyc.org/>)

Pathguide (<http://pathguide.org/?organisms=all&availability=all&standards=all&order=alphabetic&DBID=none>)

- Looking for a specific database? Pathguide contains a resource list of pathways searchable by organism and resource type.
- Most recent update was 2017

Importance of Gene IDs

To use various tools for functional analysis, you will need a list of annotated genes. Gene annotations come in a variety of flavors and not all flavors are compatible with every tool. For example, Gene Ontology (GO) is associated with Entrez, Ensemble, and official gene symbols (assigned by the HUGO Gene Nomenclature Committee (HGNC)).

Note: Genome builds will have differences in the names and coordinates of genomic features, which will impact gene ID conversions. See [this tutorial \(https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/Genomic_annotations.md\)](https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/Genomic_annotations.md) from the Harvard Chan Bioinformatics Core.

Some tools to help with annotation / conversion:

- [g:Convert \(https://biit.cs.ut.ee/gprofiler/convert\)](https://biit.cs.ut.ee/gprofiler/convert)
- Ensembl Biomart
- [AnnotationHub \(https://bioconductor.org/packages/release/bioc/html/AnnotationHub.html\)](https://bioconductor.org/packages/release/bioc/html/AnnotationHub.html)

Resources:

- [Functional enrichment and comparison with R \(https://alexslemonade.github.io/refinebio-examples/03-rnaseq/pathway-analysis_rnaseq_01_ora.html\)](https://alexslemonade.github.io/refinebio-examples/03-rnaseq/pathway-analysis_rnaseq_01_ora.html) .
- [ClusterProfiler, pathview, and good introductory information \(https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/functional_analysis_2019.md\)](https://github.com/hbctraining/Training-modules/blob/master/DGE-functional-analysis/lessons/functional_analysis_2019.md)
- [Article on the impact of the evolving GO \(https://www.nature.com/articles/s41598-018-23395-2\)](https://www.nature.com/articles/s41598-018-23395-2)
- [Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges, PLOS Computation Biology, 2012 \(https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375\)](https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002375)
- [Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap \(https://www.nature.com/articles/s41596-018-0103-9\)](https://www.nature.com/articles/s41596-018-0103-9)
- [Toward a gold standard for benchmarking gene set enrichment analysis, Briefings in Bioinformatics, 2021 \(https://academic.oup.com/bib/article/22/1/545/5722384\)](https://academic.oup.com/bib/article/22/1/545/5722384)
- [Enrichment analysis resource list from UCSF \(https://guides.ucsf.edu/bistats/pathenrich\)](https://guides.ucsf.edu/bistats/pathenrich)

- Introductory lectures on functional enrichment and R (<https://diytranscriptomics.com/project/lecture-10>)

Database for Annotation, Visualization and Integrated Discovery (DAVID) - an overview

Lesson 17 review

In the previous class, we got an overview of functional and pathway analysis, which help to put RNA sequencing results into biological context by informing us of things like biomolecular pathway, biological function, cellular localization, etc. of genes in our study. We were also introduced to tools that could help us perform these analyses.

Learning objectives

This lesson will provide an overview of the [Database for Annotation, Visualization and Integrated Discovery \(DAVID\)](https://david.ncifcrf.gov/home.jsp) (<https://david.ncifcrf.gov/home.jsp>). We will

- Provide some background on DAVID, including
 - what it does
 - statistical methods that it uses
 - some expected outputs
 - data size limits
 - how to get help
- Run an example and interpret results
 - [starting-an-analysis-in-david](#)
 - [supplying input](#)
 - [results](#)
 - [annotation results summary](#)
 - [functional annotation chart](#)
 - [functional annotation cluster](#)
 - [functional annotation table](#)
 - [gene classifier](#)

Background on DAVID

What does DAVID do?

This tool was created and is maintained by the [Laboratory of Human Retrovirology and Immunoinformatics](https://frederick.cancer.gov/research/laboratory-human-retrovirology-and-immunoinformatics) (<https://frederick.cancer.gov/research/laboratory-human-retrovirology-and-immunoinformatics>) at Frederick National Lab.

DAVID is used for functional analysis. Given an input gene list, DAVID will inform us of the following.

- Whether genes in an input gene list are associated with diseases and links out to resources such as NCBI's [MedGen \(https://www.ncbi.nlm.nih.gov/medgen/C2931781\)](https://www.ncbi.nlm.nih.gov/medgen/C2931781)
- Molecular functions that genes perform
- Biological pathways in which genes participate
- Other annotations (ie. cellular location, tissue expression, etc.) that the genes map to

Background gene list

DAVID compares the overlap of user provided gene list to an annotation to the overlap of a background gene list to the same annotation. Thus, DAVID is using the Fisher exact test to determine if the overlap of genes in the user input to a particular annotation is statistically different from what we would observe in the background. See [Table 2 in Huang et al, Nature Protocols 2009 \(https://www.nature.com/articles/nprot.2008.211/tables/2\)](https://www.nature.com/articles/nprot.2008.211/tables/2) for more information on the background gene set but essentially, the default background of the genome-wide genes is appropriate for studies that involve a genome-wide survey. However, DAVID provides custom background gene sets and users can specify their own.

Caution

"...make sure that the genes in your list are found in the background set that you have selected in DAVID otherwise, DAVID will ignore them." -- [DAVID FAQ \(https://david.ncifcrf.gov/content.jsp?file=FAQs.html#14\)](https://david.ncifcrf.gov/content.jsp?file=FAQs.html#14)

Basic statistics behind DAVID

Fisher Exact Test

DAVID performs over representation analysis (ORA) at its core, which aims to find enriched molecular functions, pathways, or other annotations represented by the input gene list. In other words, many genes in the list map onto those molecular functions, pathways, or annotations.

With DAVID, we are essentially looking at contingency tables (Figure 1). The example in Figure 1 shows the number of user input genes and background genes (selected from the whole genome) that fall onto a particular pathway (ie. p53 signaling). However, how certain can we be that the number of user input genes that map to the pathway is observed not by random chance. In other words, do user input genes fall onto a pathway more often as compared to the background or expected. DAVID uses the Fisher exact test to help us decide whether what we are observing is due to chance.

	User Genes	Genome	
In Pathway	3	37	40
Not in Pathway	297	29663	29960
	300	29700	30000

Exact p -value = 0.007. Since p -value < 0.05, this user's gene list is specifically associated (enriched) in the p53 signaling pathway by more than random chance.

Figure 1: Contingency table showing the number of user input genes and background genes from the genome that fall onto a certain pathway. [DAVID help documentations \(https://david.ncifcrf.gov/helps/functional_annotation.html#bonfer\)](https://david.ncifcrf.gov/helps/functional_annotation.html#bonfer)

Below are some resources for you to learn about or review this statistical procedure.

- Fisher exact test from Wikipedia (https://en.wikipedia.org/wiki/Fisher%27s_exact_test)
- Hypergeometric distribution from Wikipedia (https://en.wikipedia.org/wiki/Hypergeometric_distribution)
- Fisher exact test from Pathway Commons (https://www.pathwaycommons.org/guide/primers/statistics/fishers_exact_test/)
- Hypergeometric distribution from Pathway Commons (<https://www.pathwaycommons.org/guide/primers/statistics/distributions/#hypergeometric>)
- EASE score (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC328459/>)

Pathway Commons also provides a statistics primer that discusses those methods that are relevant to pathway analysis.

- Pathway Commons Statistics Primer (<https://www.pathwaycommons.org/guide/primers/statistics/>)

Multiple Testing Correction

A problem that arises with enrichment analysis is the need to perform multiple statistical tests across many gene sets. In short, type I errors or false positives increases as the number of tests performed increases -- [Pathway Commons multiple testing \(https://www.pathwaycommons.org/guide/primers/statistics/multiple_testing/\)](https://www.pathwaycommons.org/guide/primers/statistics/multiple_testing/). Users can choose to use either Bonferroni, Benjamini-Hochberg, or False Discovery Rate (FDR) to correct for multiple testing.

Reducing Redundancy

"Due to the redundant nature of annotations, Functional Annotation Chart presents similar/relevant annotations repeatedly. It dilutes the focus of the biology in the report. To reduce the redundancy, the Functional Annotation Clustering report groups/displays similar annotations together which makes the biology clearer and more focused..." -- [DAVID help documents \(https://david.ncifcrf.gov/helps/functional_annotation.html#summary\)](https://david.ncifcrf.gov/helps/functional_annotation.html#summary)

DAVID uses the *Kappa statistic* (<https://www.sciencedirect.com/topics/medicine-and-dentistry/kappa-statistics>) is used to measure the level of similarities in genes between annotations and then applies *fuzzy heuristic clustering* (https://david.ncifcrf.gov/helps/functional_classification.html#heuristic) to cluster groups of similar annotations.

Data size limits

"The goal of DAVID's design is to be able to efficiently upload and analyze a list consisting of ≤ 3000 genes. All DAVID tools have been tested with lists in this range and should return results in a few seconds to no more than a few minutes. If running time is longer than a few minutes, please contact the DAVID Bioinformatic Team for help. Please note that Functional Annotation Clustering and Gene Functional Classification have a 3000 gene limit." -- **DAVID FAQ** (<https://david.ncifcrf.gov/helps/FAQs.html>)

Getting help

DAVID question and forum (<https://david-bioinformatics.freeforums.net/>)

DAVID FAQ (<https://david.ncifcrf.gov/content.jsp?file=FAQs.html>)

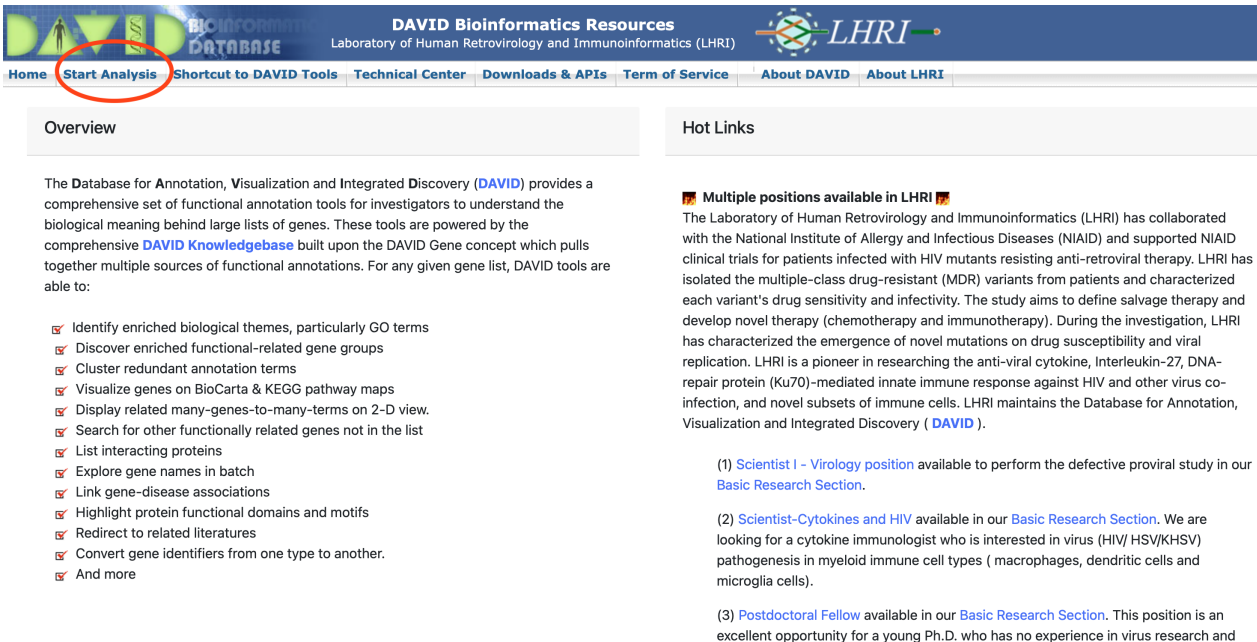
DAVID help documentations (https://david.ncifcrf.gov/helps/functional_annotation.html#bonfer)

DAVID quick start tutorial (<https://david.ncifcrf.gov/helps/tutorial.pdf>)

Starting an analysis in DAVID

USE GOOGLE CHROME TO INTERACT WITH DAVID

Click on the Start Analysis button to initiate an analysis, this will take us to the Analysis Wizard.



DAVID Bioinformatics Resources
Laboratory of Human Retrovirology and Immunoinformatics (LHRI)

Home **Start Analysis** Shortcut to DAVID Tools Technical Center Downloads & APIs Term of Service About DAVID About LHRI

Overview

The Database for Annotation, Visualization and Integrated Discovery (DAVID) provides a comprehensive set of functional annotation tools for investigators to understand the biological meaning behind large lists of genes. These tools are powered by the comprehensive **DAVID Knowledgebase** built upon the DAVID Gene concept which pulls together multiple sources of functional annotations. For any given gene list, DAVID tools are able to:

- Identify enriched biological themes, particularly GO terms
- Discover enriched functional-related gene groups
- Cluster redundant annotation terms
- Visualize genes on BioCarta & KEGG pathway maps
- Display related many-genes-to-many-terms on 2-D view.
- Search for other functionally related genes not in the list
- List interacting proteins
- Explore gene names in batch
- Link gene-disease associations
- Highlight protein functional domains and motifs
- Redirect to related literatures
- Convert gene identifiers from one type to another.
- And more

Hot Links

Multiple positions available in LHRI

The Laboratory of Human Retrovirology and Immunoinformatics (LHRI) has collaborated with the National Institute of Allergy and Infectious Diseases (NIAID) and supported NIAID clinical trials for patients infected with HIV mutants resisting anti-retroviral therapy. LHRI has isolated the multiple-class drug-resistant (MDR) variants from patients and characterized each variant's drug sensitivity and infectivity. The study aims to define salvage therapy and develop novel therapy (chemotherapy and immunotherapy). During the investigation, LHRI has characterized the emergence of novel mutations on drug susceptibility and viral replication. LHRI is a pioneer in researching the anti-viral cytokine, Interleukin-27, DNA-repair protein (Ku70)-mediated innate immune response against HIV and other virus co-infection, and novel subsets of immune cells. LHRI maintains the Database for Annotation, Visualization and Integrated Discovery (DAVID).

- (1) **Scientist I - Virology position** available to perform the defective proviral study in our **Basic Research Section**.
- (2) **Scientist-Cytokines and HIV** available in our **Basic Research Section**. We are looking for a cytokine immunologist who is interested in virus (HIV/ HSV/KHSV) pathogenesis in myeloid immune cell types (macrophages, dendritic cells and microglia cells).
- (3) **Postdoctoral Fellow** available in our **Basic Research Section**. This position is an excellent opportunity for a young Ph.D. who has no experience in virus research and

Supplying input

Tasks to do at the Analysis Wizard:

1. provide our input gene list (either copy paste or upload as a text file)
2. specify gene identifier type (gene identifiers could be gene symbol, Ensembl IDs, Entrez IDs, Genbank IDs, Refseq IDs, etc.)
3. specify whether we are providing an input gene list or background gene list
4. submit the list for analysis

Here, we are going to provide the genes that are upregulated in the UHR sample with respect to the HBR samples. These genes were obtained by filtering the differential expression table for \log_2 fold change ≥ 1 and false discovery rate of ≤ 0.05 . The genes are in the file [hbr_uhr_deg_chr22_up_genes.txt](#).

Step 1: After attaching hbr_uhr_deg_chr22_up_genes.txt as the input gene list in the DAVID Analysis Wizard, choose OFFICIAL_GENE_SYMBOL as the identifier type and then specify Homo sapiens in the Select species box that appears. Next, specify that we are providing an input gene list and then click on Submit List.

The screenshot shows the DAVID Analysis Wizard interface. The top navigation bar includes links for Home, Start Analysis, Shortcut to DAVID Tools, Technical Center, Downloads & APIs, Term of Service, About DAVID, and About LHRI. The main content area is titled "Analysis Wizard" and features a left sidebar with four steps: Step 1: Enter Gene List, Step 2: Select Identifier, Step 2a: Select species, Step 3: List Type, and Step 4: Submit List. In Step 1, there are two options: "A: Paste a list" with a text input field and a "Clear" button, and "B: Choose From a File" with a "Choose File" button and a "Multi-List File" checkbox. The "Choose File" button is highlighted with a blue box, and the file name "hbr_uhr_deg_chr22_up_genes.txt" is visible. Below the file selection, the "OFFICIAL_GENE_SYMBOL" identifier is selected in a dropdown menu, and "Homo sapiens" is selected in the species dropdown. In Step 3, "Gene List" is selected with a radio button. In Step 4, the "Submit List" button is visible. On the right side of the main content area, there is a blue arrow pointing left with the text "Step 1. Submit your gene list through left panel." Below this, an example list of gene IDs is provided: 1007_s_at, 1053_at, 117_at, 121_at, 1255_g_at, 131_at, 1316_at, 1320_at, 1405_i_at, 1431_at, 1438_at, 1487_at, 1494_f_at, and 1598_g_at. A "Copy/paste IDs to 'box A' -> Select Identifier as 'Affy_ID' -> List Type as 'Gene List' -> Click 'Submit' button" instruction is also present. At the top right of the main content area, there are links for "Tell us how you like the tool" and "Contact us for questions".

Step 2: After submitting the gene list, DAVID will tell us that we have successfully submitted the gene list and that we are using the Homo sapiens genome as background. We can then select which analysis tool we like to run. Notice that there is a Gene ID Conversion Tool. This is used to convert the input gene list to a set of IDs that are recognized by DAVID in the event that we do not know or DAVID does not recognize the identifier type in our input.

The screenshot shows the DAVID Analysis Wizard interface. The top navigation bar includes links for Home, Start Analysis, Shortcut to DAVID Tools, Technical Center, Downloads & APIs, Term of Service, About DAVID, and About LHRI. The main content area is titled "Analysis Wizard" and features a "Gene List Manager" on the left and a list of analysis tools on the right. Two red boxes highlight key steps: "Step 1. Successfully submitted gene list" and "Step 2. Analyze above gene list with one of DAVID tools". Red arrows point to "Unknown(18)" in the species list and "View Unmapped Ids" at the bottom left.

Analysis Wizard
DAVID Bioinformatics Resources, NIAID/NIH

Home Start Analysis Shortcut to DAVID Tools Technical Center Downloads & APIs Term of Service About DAVID About LHRI

Upload List Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
Homo sapiens(97)
Unknown(18)

Select Species

List Manager [Help](#)

hbr_uhr_deg_chr22_u

Select List to:
Use Rename
Remove Combine
Show Gene List

[View Unmapped Ids](#)

Step 1. Successfully submitted gene list
Current Gene List: hbr_uhr_deg_chr22_up_genes
Current Background: Homo sapiens

Step 2. Analyze above gene list with one of DAVID tools
Which DAVID tools to use?

- ⇒ Functional Annotation Tool
 - Functional Annotation Clustering
 - Functional Annotation Chart
 - Functional Annotation Table
- ⇒ Gene Functional Classification Tool
- ⇒ Gene ID Conversion Tool
- ⇒ Gene Name Batch Viewer

[Tell us how you like the tool](#)
[Contact us for questions](#)

Step 2 alternative: Here, we re-upload hbr_uhr_deg_chr22_up_genes.txt and then select Not Sure in the Select Identifier drop down menu. This will take us to the Gene ID Conversion Tool.

[Home](#) [Start Analysis](#) [Shortcut to DAVID Tools](#) [Technical Center](#) [Downloads & APIs](#) [Term of Service](#)

Upload **List** **Background**

Upload Gene List

[Demolist 1](#) [Demolist 2](#)

[Upload Help](#)

Step 1: Enter Gene List

A: Paste a list

Clear

Or

B: Choose From a File

Choose File [hbr_uhr_deg_...2_up_genes.txt](#)

Multi-List File ?

Step 2: Select Identifier

Not Sure

Step 3: List Type

Gene List

Background

Step 4: Submit List

Submit List

Analysis Wizard

← Step 1. Submit your gene list through left panel.

An example:

Copy/paste IDs to "**box A**" -> Select Identifier as "**Affy_ID**" -> List Type a button

1007_s_at
1053_at
117_at
121_at
1255_g_at
1294_at
1316_at
1320_at
1405_i_at
1431_at
1438_at
1487_at
1494_f_at
1598_g_at

Gene ID conversion - specify ID type to convert to: We have an option to choose a range of IDs to convert our gene list to but in this example, we have chosen the ENSEMBL_GENE_ID. Remember to specify the species where our genes came from (Homo sapiens in this case). When ready, hit Submit to Conversion Tool.

Gene ID Conversion Tool
DAVID Bioinformatics Resources, NIAID/NIH

Home Start Analysis Shortcut to DAVID Tools Technical Center Downloads & APIs Term of Service About DAVID About LHRI

Upload List Background

Gene ID Conversion Tool

[Help and Tool Manual](#)

Upload Gene List

[Demolist 1](#) [Demolist 2](#)
[Upload Help](#)

Step 1: Enter Gene List

A: Paste a list

Clear

Or

B: Choose From a File

Choose File No file chosen

Multi-List File

Step 2: Select Identifier

AFFYMETRIX_3PRIME_IVT_ID

Step 3: List Type

Gene List

Background

You are either not sure which identifier type your list contains, or less than 80% of your list has mapped to your chosen identifier type. Please use the Gene Conversion Tool to determine the identifier type.

Option 1:
Convert the gene list to

For species:

Submit to Conversion Tool






Option 2:

Gene ID conversion: Once we hit Submit to Conversion Tool, we will be taken to the page below. We can choose to convert all genes or convert each gene individually. Some of the genes may not be in the DAVID database, so the Gene ID conversion tool will not be able to convert those. We would need to do some data wrangling to find identifiers for those genes not in the DAVID database.

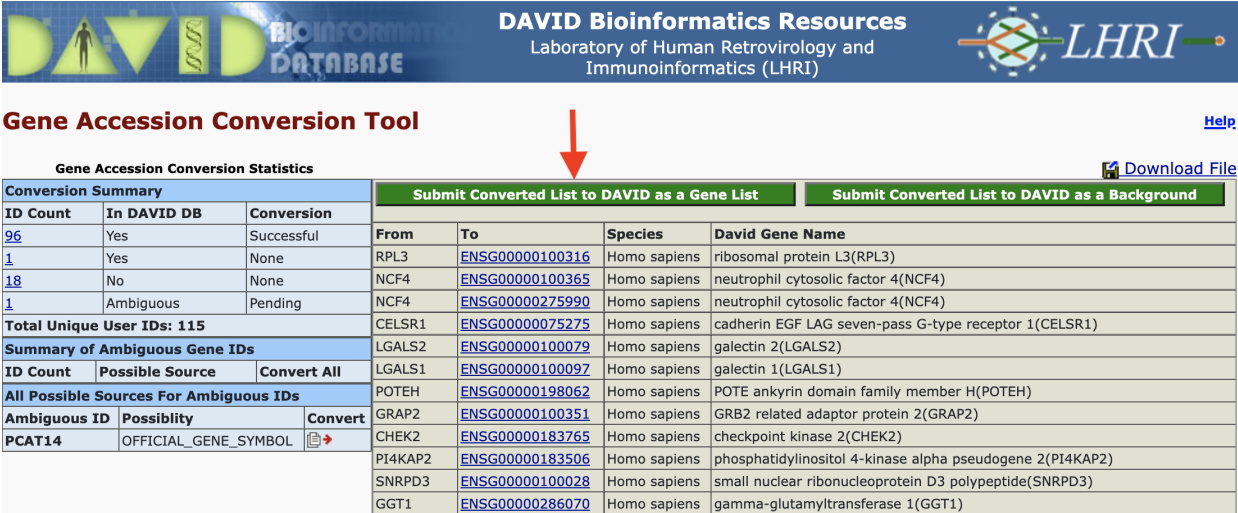


Gene Accession Conversion Tool

Gene Accession Conversion Statistics

Conversion Summary		
ID Count	In DAVID DB	Conversion
0	Yes	Successful
0	Yes	None
18	No	None
97	Ambiguous	Pending
Total Unique User IDs: 115		
Summary of Ambiguous Gene IDs		
ID Count	Possible Source	Convert All
97	OFFICIAL_GENE_SYMBOL	
All Possible Sources For Ambiguous IDs		
Ambiguous ID	Possibility	Convert
IL2RB	OFFICIAL_GENE_SYMBOL	
LGALS2	OFFICIAL_GENE_SYMBOL	
LGALS1	OFFICIAL_GENE_SYMBOL	
EIF3L	OFFICIAL_GENE_SYMBOL	

Gene ID conversion - send converted IDs back to input: After conversion, we can send the new list back to DAVID either as input or background. Here, we will submit as input.

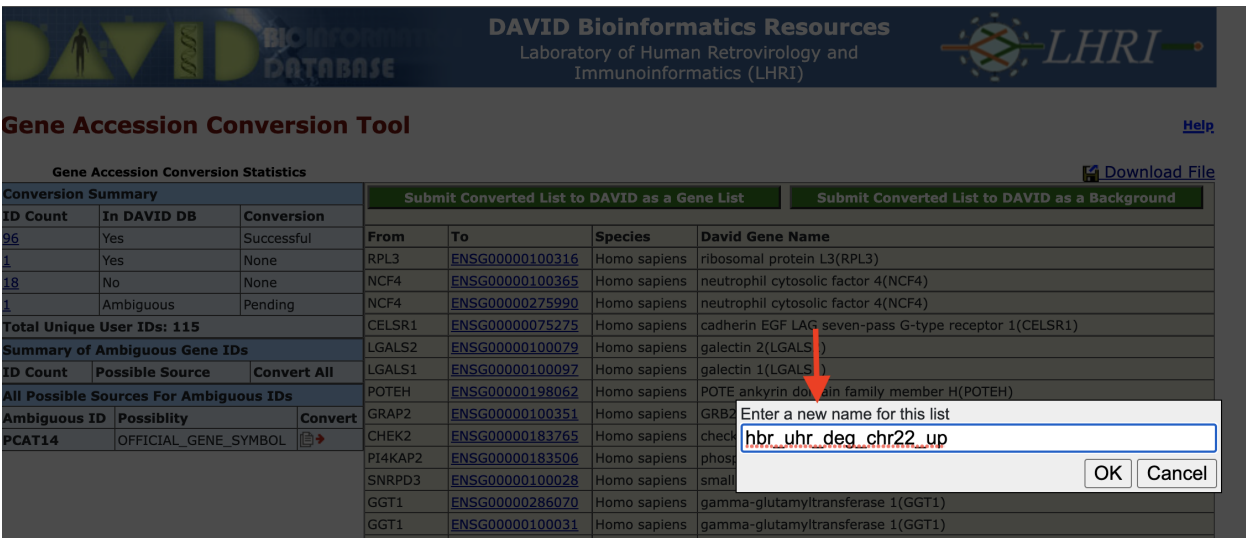


Gene Accession Conversion Tool [Help](#)

Gene Accession Conversion Statistics [Download File](#)

Conversion Summary			Submit Converted List to DAVID as a Gene List		Submit Converted List to DAVID as a Background	
ID Count	In DAVID DB	Conversion	From	To	Species	David Gene Name
96	Yes	Successful	RPL3	ENSG00000100316	Homo sapiens	ribosomal protein L3(RPL3)
1	Yes	None	NCF4	ENSG00000100365	Homo sapiens	neutrophil cytosolic factor 4(NCF4)
18	No	None	NCF4	ENSG00000275990	Homo sapiens	neutrophil cytosolic factor 4(NCF4)
1	Ambiguous	Pending	CELSR1	ENSG00000075275	Homo sapiens	cadherin EGF LAG seven-pass G-type receptor 1(CELSR1)
Total Unique User IDs: 115			LGALS2	ENSG00000100079	Homo sapiens	galectin 2(LGALS2)
Summary of Ambiguous Gene IDs			LGALS1	ENSG00000100097	Homo sapiens	galectin 1(LGALS1)
ID Count	Possible Source	Convert All	POTEH	ENSG00000198062	Homo sapiens	POTE ankyrin domain family member H(POTEH)
All Possible Sources For Ambiguous IDs			GRAP2	ENSG00000100351	Homo sapiens	GRB2 related adaptor protein 2(GRAP2)
Ambiguous ID	Possibility	Convert	CHEK2	ENSG00000183765	Homo sapiens	checkpoint kinase 2(CHEK2)
PCAT14	OFFICIAL_GENE_SYMBOL		PI4KAP2	ENSG00000183506	Homo sapiens	phosphatidylinositol 4-kinase alpha pseudogene 2(PI4KAP2)
			SNRPD3	ENSG00000100028	Homo sapiens	small nuclear ribonucleoprotein D3 polypeptide(SNRPD3)
			GGT1	ENSG00000286070	Homo sapiens	gamma-glutamyltransferase 1(GGT1)

Gene ID conversion - name the converted ID list: DAVID will give us the option to name the converted gene list. We will name its hbr_uhr_deg_chr22_up. Note that the Gene ID Conversion Tool was opened in a separate tab. After we submit the converted gene list, go back to the Analysis Wizard tab to continue with the analysis.



Gene Accession Conversion Tool [Help](#)

Gene Accession Conversion Statistics [Download File](#)

Conversion Summary			Submit Converted List to DAVID as a Gene List		Submit Converted List to DAVID as a Background	
ID Count	In DAVID DB	Conversion	From	To	Species	David Gene Name
96	Yes	Successful	RPL3	ENSG00000100316	Homo sapiens	ribosomal protein L3(RPL3)
1	Yes	None	NCF4	ENSG00000100365	Homo sapiens	neutrophil cytosolic factor 4(NCF4)
18	No	None	NCF4	ENSG00000275990	Homo sapiens	neutrophil cytosolic factor 4(NCF4)
1	Ambiguous	Pending	CELSR1	ENSG00000075275	Homo sapiens	cadherin EGF LAG seven-pass G-type receptor 1(CELSR1)
Total Unique User IDs: 115			LGALS2	ENSG00000100079	Homo sapiens	galectin 2(LGALS2)
Summary of Ambiguous Gene IDs			LGALS1	ENSG00000100097	Homo sapiens	galectin 1(LGALS1)
ID Count	Possible Source	Convert All	POTEH	ENSG00000198062	Homo sapiens	POTE ankyrin domain family member H(POTEH)
All Possible Sources For Ambiguous IDs			GRAP2	ENSG00000100351	Homo sapiens	GRB2 related adaptor protein 2(GRAP2)
Ambiguous ID	Possibility	Convert	CHEK2	ENSG00000183765	Homo sapiens	checkpoint kinase 2(CHEK2)
PCAT14	OFFICIAL_GENE_SYMBOL		PI4KAP2	ENSG00000183506	Homo sapiens	phosphatidylinositol 4-kinase alpha pseudogene 2(PI4KAP2)
			SNRPD3	ENSG00000100028	Homo sapiens	small nuclear ribonucleoprotein D3 polypeptide(SNRPD3)
			GGT1	ENSG00000286070	Homo sapiens	gamma-glutamyltransferase 1(GGT1)
			GGT1	ENSG00000100031	Homo sapiens	gamma-glutamyltransferase 1(GGT1)

Enter a new name for this list

Results and interpretation

Annotation Results Summary

Once we have submitted our gene list for analysis, DAVID takes us to the Annotation Summary Results page. This page confirms the name of our gene list (hbr_uhr_deg_chr22_up) and the background gene list that we are using (Homo sapiens). We can navigate the Annotation Summary Results page to obtain different insights to our data.

The screenshot displays the DAVID Functional Annotation Tool interface. The top navigation bar includes links for Home, Start Analysis, Shortcut to DAVID Tools, Technical Center, Downloads & APIs, Term of Service, About DAVID, and About LHRI. The main content area is titled "Annotation Summary Results" and shows the following information:

- Current Gene List:** hbr_uhr_deg_chr22_up
- Current Background:** Homo sapiens
- 96 DAVID IDs** (with a "Check Defaults" checkbox and a "Clear All" button)
- Selected Annotations:**
 - Disease (2 selected)
 - Functional_Annotations (6 selected)
 - Gene_Ontology (3 selected)
 - General_Annotations (0 selected)
 - Interactions (1 selected)
 - Literature (0 selected)
 - Pathways (3 selected)
 - Protein_Domains (4 selected)
 - Tissue_Expression (0 selected)
- Combined View for Selected Annotation:**
 - Functional Annotation Clustering
 - Functional Annotation Chart
 - Functional Annotation Table

On the left side, the "Gene List Manager" section allows users to select species (currently set to Homo sapiens) and manage the gene list (hbr_uhr_deg_chr22_u). The "Select List to:" section includes buttons for Use, Rename, Remove, Combine, and Show Gene List.

Disease Annotations

One of the first insights is that DAVID informs whether our input genes play a role in diseases. DAVID pulls disease annotations from different sources. Clicking on the Chart button will take us to a chart view showing the disease records from a given database in which our input genes map.

- DISGENET (<https://www.disgenet.org>)
- GAD_DISEASE
- OMIM (<https://www.omim.org>)

- UniProt - UP_KW_DISEASE where KW stands for keyword (<https://www.uniprot.org/keywords/KW-9995>)

Annotation Summary Results

[Help and Tool Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up

96 DAVID IDs

Current Background: Homo sapiens

Check Defaults

Clear All

Disease (2 selected)

Category	Percentage	Count	Chart
<input type="checkbox"/> DISGENET	56.2%	54	
<input type="checkbox"/> GAD_DISEASE	65.6%	63	
<input type="checkbox"/> GAD_DISEASE_CLASS	65.6%	63	
<input checked="" type="checkbox"/> OMIM_DISEASE	29.2%	28	
<input checked="" type="checkbox"/> UP_KW_DISEASE	22.9%	22	

Functional_Annotations (6 selected)

Gene_Ontology (3 selected)

General_Annotations (0 selected)

Interactions (1 selected)

Literature (0 selected)

Pathways (3 selected)

Protein_Domains (4 selected)

Tissue_Expression (0 selected)

Disease Annotation - chart view

In the chart view, we are presented with the disease(s) found in a particular database that our input genes map to. Clicking on one of the disease terms sends us to NCBI' MedGen.

Functional Annotation Chart

[Help and Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up

Current Background: Homo sapiens

96 DAVID IDs

Options

Rerun Using Options

Create Sublist

8 chart records

Download File


Sublist	Category	Term	RT	Genes	Count	%	P-Value	Benjamini
<input type="checkbox"/>	DISGENET	Malignant neoplasm of breast	RT		13	13.5	1.2E-2	1.0E0
<input type="checkbox"/>	DISGENET	Psychotic Disorders	RT		4	4.2	1.8E-2	1.0E0
<input type="checkbox"/>	DISGENET	Mammary Neoplasms, Experimental	RT		4	4.2	5.2E-2	1.0E0
<input type="checkbox"/>	DISGENET	Nonorganic psychosis	RT		3	3.1	5.4E-2	1.0E0
<input type="checkbox"/>	DISGENET	Nonalcoholic Steatohepatitis	RT		3	3.1	5.6E-2	1.0E0
<input type="checkbox"/>	DISGENET	Non-alcoholic Fatty Liver Disease	RT		3	3.1	5.6E-2	1.0E0
<input type="checkbox"/>	DISGENET	Steatohepatitis	RT		3	3.1	8.3E-2	1.0E0
<input type="checkbox"/>	DISGENET	Fatty Liver	RT		3	3.1	8.3E-2	1.0E0

74 gene(s) from your list are not in the output.

Disease Annotation - link to external resource

In the chart view shown above, we clicked on Malignant neoplasm of the breast, and this took us to the corresponding record in NCBI's MedGen. MedGen is NCBI's database that contains organized information pertaining to human gene-disease relationships.

 An official website of the United States government [Here's how you know](#) ▾


National Library of Medicine
National Center for Biotechnology Information


MedGen
[Create alert](#) [Limits](#) [Advanced](#)

[Full Report](#) ▾ [Send t](#)

Malignant tumor of breast
 MedGen UID: 651 • Concept ID: C0006142 • Neoplastic Process

Synonyms: Breast cancer; Malignant breast neoplasm

SNOMED CT: Malignant neoplasm of breast (254837009); Malignant tumor of breast (254837009); Breast cancer (254837009); CA - Breast cancer (254837009)



Related genes:  BRIP1, PALB2, CHEK2, RB1CC1, PPM1D, RAD54L, XRCC3, TP53, STK11, RAD51D, RAD51C, RAD51, PTEN, PIK3CA, PHB1, SLC22A18, NQO2, KRAS, HMMR, ESR1, CDH1, CASP8, BRCA2, BRCA1, BARD1, ATM, AKT1

Monarch Initiative: MONDO:0007254

OMIM®: 114480

Disease Annotation - gene view

If we click on the blue bar next to the chart button, we will be taken to a gene view of the disease terms.


Functional Annotation Tool
DAVID Bioinformatics Resources, NIAID/NIH


[Home](#) [Start Analysis](#) [Shortcut to DAVID Tools](#) [Technical Center](#) [Downloads & APIs](#) [Term of Service](#) [About DAVID](#) [About LHRI](#)

Upload List Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)






- Use All Species -
 Homo sapiens(103)

Select Species

Annotation Summary Results [Help and Tool Manual](#)


Current Gene List: hbr_uhr_deg_chr22_up **96 DAVID IDs**

Current Background: Homo sapiens **Check Defaults** [Clear All](#)


Disease (2 selected)				
<input type="checkbox"/> DISGENET	56.2%	54	Chart	
<input type="checkbox"/> GAD_DISEASE	65.6%	63	Chart	
<input type="checkbox"/> GAD_DISEASE_CLASS	65.6%	63	Chart	
<input checked="" type="checkbox"/> OMIM_DISEASE	29.2%	28	Chart	
<input checked="" type="checkbox"/> UP_KW_DISEASE	22.9%	22	Chart	

Disease Annotation - gene view results

The gene view lists disease(s) that the gene may play a role in.



DAVID Bioinformatics Resources
 Laboratory of Human Retrovirology and Immunoinformatics (LHRI)



Functional Annotation Table

[Help and Manual](#)


Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
 96 DAVID IDs

54 record(s) [Download File](#)


ENSG ID	Gene Name	Related Genes	Species
ENSG00000242247	ADP ribosylation factor GTPase activating protein 3(ARFGAP3)		Homo sapiens
DISGENET	Malignant neoplasm of breast,		
ENSG00000133466	C1q and TNF related 6(C1QTNF6)		Homo sapiens
DISGENET	Diabetes Mellitus, Insulin-Dependent, Diabetes, Autoimmune, Brittle diabetes, Diabetes Mellitus, Ketosis-Prone, Diabetes Mellitus, Sudden-Onset,		
ENSG00000100038	DNA topoisomerase III beta(TOP3B)		Homo sapiens
DISGENET	Schizophrenia,		
ENSG00000185838	G protein subunit beta 1 like(GNB1L)		Homo sapiens
DISGENET	Bipolar Disorder, Malignant neoplasm of breast, Psychotic Disorders, Schizophrenia, Nonorganic psychosis,		
ENSG00000075218	G2 and S-phase expressed 1(GTSE1)		Homo sapiens
DISGENET	Liver carcinoma,		
ENSG00000128342	LIF interleukin 6 family cytokine(LIF)		Homo sapiens
DISGENET	Spontaneous abortion, Abortion, Tubal, Delirium, Mental Depression, Depressive disorder, Blastocyst Disintegration, Embryo Resorption, Cardiomegaly, Female infertility, Polycystic Ovary Syndrome, Sterility, Postpartum, Subfertility, Female, Non-alcoholic Fatty Liver Disease, Embryo Death, Embryo Loss, Female sterility, Embryo Disintegration, Sclerocystic Ovaries, Weight decreased, Cardiac Hypertrophy, Female Urogenital Diseases, Nonalcoholic Steatohepatitis, Early Pregnancy Loss, Miscarriage,		
ENSG00000185022	MAF bZIP transcription factor F(MAFF)		Homo sapiens
DISGENET	Juvenile-Onset Still Disease, Juvenile arthritis, Juvenile psoriatic arthritis, Polyarthrits, Juvenile, Rheumatoid Factor Negative, Polyarthrits, Juvenile, Rheumatoid Factor Positive,		
ENSG00000169184	MN1 proto-oncogene, transcriptional regulator(MN1)		Homo sapiens
DISGENET	Craniosynostosis, Leukemia, Myelocytic, Acute, Acute Myeloid Leukemia, M1, Feeding difficulties, Polymicrogyria, Global developmental delay, Familial meningioma, hearing impairment, Central hypotonia, Abnormal corpus callosum morphology, Abnormality of the cerebellum Acute Myeloid Leukemia (AML-M2), Deformity of facial bone, Intellectual Disability, Abnormality of the face,		

Disease Annotation - OMIM

The chart view for some annotation databases such as OMIM will not have records. This is because there were no diseases that met the statistical threshold.



Functional Annotation Tool
 DAVID Bioinformatics Resources, NIAID/NIH



[Home](#) [Start Analysis](#) [Shortcut to DAVID Tools](#) [Technical Center](#) [Downloads & APIs](#) [Term of Service](#) [About DAVID](#) [About LHRI](#)

Upload | **List** | **Background**

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
- Homo sapiens(103)

Select Species





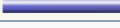
Annotation Summary Results

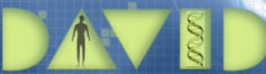
[Help and Tool Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up **96 DAVID IDs**


Current Background: Homo sapiens **Check Defaults** **Clear All**

Disease (2 selected)

Disease	Percentage	Count	Chart
<input type="checkbox"/> DISGENET	56.2%	54	
<input type="checkbox"/> GAD_DISEASE	65.6%	63	
<input type="checkbox"/> GAD_DISEASE_CLASS	65.6%	63	
<input checked="" type="checkbox"/> OMIM_DISEASE	29.2%	28	
<input checked="" type="checkbox"/> UP_KW_DISEASE	22.9%	22	



DAVID Bioinformatics Resources
 Laboratory of Human Retrovirology and Immunoinformatics (LHRI)



Functional Annotation Chart

[Help and Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
96 DAVID IDs


Options

Thresholds: Count EASE
 Display: Fold Enrichment Bonferroni Benjamini FDR Fisher Exact LT,PH,PT # of Records


0 chart records

from your list are not in the output.

However, we can still click on the gene view to see what disease individual genes may play a role in.



Functional Annotation Tool
 DAVID Bioinformatics Resources, NIAID/NIH



[Home](#) [Start Analysis](#) [Shortcut to DAVID Tools](#) [Technical Center](#) [Downloads & APIs](#) [Term of Service](#) [About DAVID](#) [About LHRI](#)

Upload List Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
Homo sapiens(103)

Select Species

List Manager [Help](#)

hbr_uhr_deg_chr22_u

Select List to:

Annotation Summary Results

[Help and Tool Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
96 DAVID IDs
Check Defaults

Disease (2 selected)

Disease	Percentage	Count	Chart
<input type="checkbox"/> DISGENET	56.2%	54	<input type="button" value="Chart"/>
<input type="checkbox"/> GAD_DISEASE	65.6%	63	<input type="button" value="Chart"/>
<input type="checkbox"/> GAD_DISEASE_CLASS	65.6%	63	<input type="button" value="Chart"/>
<input checked="" type="checkbox"/> OMIM_DISEASE	29.2%	28	<input type="button" value="Chart"/>
<input checked="" type="checkbox"/> UP_KW_DISEASE	22.9%	22	<input type="button" value="Chart"/>

Functional_Annotations (6 selected)

Gene_Ontology (3 selected)

General_Annotations (0 selected)

Interactions (1 selected)

Literature (0 selected)

Pathways (3 selected)

Protein_Domains (4 selected)

Tissue_Expression (0 selected)

For diseases that are annotated in OMIM, clicking on the corresponding link in the gene view will take us to the OMIM record.

Functional Annotation Table

[Help and Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up

Current Background: Homo sapiens

96 DAVID IDs

28 record(s)

[Download File](#)

ENSG00000169184	MN1 proto-oncogene, transcriptional regulator(MN1)	Related Genes	Homo sapiens
OMIM_DISEASE	Meningioma , CEBALID syndrome ,		
ENSG00000128340	Rac family small GTPase 2(RAC2)	Related Genes	Homo sapiens
OMIM_DISEASE	Immunodeficiency 73A with defective neutrophil chemotaxis and leukocytosis, Immunodeficiency 73B with defective neutrophil chemotaxis and lymphopenia, Immunodeficiency 73C with defective neutrophil chemotaxis and hypogammaglobulinemia,		
ENSG00000184058	T-box transcription factor 1(TBX1)	Related Genes	Homo sapiens
OMIM_DISEASE	Tetralogy of Fallot, DiGeorge syndrome, Velocardiofacial syndrome, Conotruncal anomaly face syndrome,		
ENSG00000100106	TRIO and F-actin binding protein(TRIOBP)	Related Genes	Homo sapiens
OMIM_DISEASE	Deafness, autosomal recessive 28,		
ENSG00000100219	X-box binding protein 1(XBP1)	Related Genes	Homo sapiens
OMIM_DISEASE	Major affective disorder-7, susceptibility to,		
ENSG00000196236	X-prolyl aminopeptidase 3(XPNPEP3)	Related Genes	Homo sapiens
OMIM_DISEASE	Nephronophthisis-like nephropathy 1,		
ENSG00000239900	adenylosuccinate lyase(ADSL)	Related Genes	Homo sapiens
OMIM_DISEASE	Adenylosuccinate deficiency,		
ENSG00000128274	alpha 1,4-galactosyltransferase (P blood group)(A4GALT)	Related Genes	Homo sapiens
OMIM_DISEASE	Blood group, P1Pk system, P(2) phenotype, Blood group, P1Pk system, p phenotype, NOR polyagglutination syndrome,		
ENSG00000198951	alpha-N-acetylgalactosaminidase(NAGA)	Related Genes	Homo sapiens
OMIM_DISEASE	Schindler disease, type I, Schindler disease, type III, Kanzaki disease,		

Here, we clicked on Meningioma and was taken to the OMIM record for this disease where we see MN1 as one of the genes involved in the disorder.





#607174

Table of Contents

Title

Phenotype-Gene Relationships

Clinical Synopsis

Text

Description

Clinical Features

Inheritance

Cytogenetics

Mapping

Molecular Genetics

See Also

References

Contributors

Creation Date

Edit History

607174

MENINGIOMA, FAMILIAL, SUSCEPTIBILITY TO

Phenotype-Gene Relationships

Location	Phenotype	Phenotype MIM number	Inheritance	Phenotype mapping key	Gene/Locus	Gene/Locus MIM number
10q23.31	{Meningioma}	607174	AD	3	PTEN	601728
10q24.32	{Meningioma, familial, susceptibility to}	607174	AD	3	SUFU	607035
17q21.2	{Meningioma, familial, susceptibility to}	607174	AD	3	SMARCE1	603111
22q12.1	Meningioma	607174	AD	3	MN1	156100
22q12.2	Meningioma, NF2-related, somatic	607174		3	NF2	607379
22q13.1	Meningioma, SIS-related	607174	AD	3	PDGFB	190040

Clinical Synopsis

PheneGene Graphics



ICD+

Pathways:

We see similar organization of information for other annotations. For instance, DAVID pulls biomolecular pathway information from several sources such as KEGG.

Functional Annotation Tool
DAVID Bioinformatics Resources, NIAID/NIH

Home | Start Analysis | Shortcut to DAVID Tools | Technical Center | Downloads & APIs | Term of Service | About DAVID | About LHRI

Upload | List | Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
Homo sapiens(103)

Select Species

List Manager [Help](#)
hbr_uhr_deg_chr22_u

Select List to:
Use | Rename
Remove | Combine
Show Gene List

Annotation Summary Results [Help and Tool Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up **96 DAVID IDs**
Current Background: Homo sapiens **Check Defaults**

Disease (2 selected)
 Functional_Annotations (6 selected)
 Gene_Ontology (3 selected)
 General_Annotations (0 selected)
 Interactions (1 selected)
 Literature (0 selected)
 Pathways (3 selected)

Annotation	Percentage	Count	Chart
<input checked="" type="checkbox"/> BBID	1.0%	1	<input type="button" value="Chart"/>
<input checked="" type="checkbox"/> BIOCARTA	13.5%	13	<input type="button" value="Chart"/>
<input type="checkbox"/> EC_NUMBER	25.0%	24	<input type="button" value="Chart"/>
<input checked="" type="checkbox"/> KEGG_PATHWAY	45.8%	44	<input type="button" value="Chart"/>
<input type="checkbox"/> REACTOME_PATHWAY	57.3%	55	<input type="button" value="Chart"/>
<input type="checkbox"/> WIKIPATHWAYS	47.9%	46	<input type="button" value="Chart"/>

Protein_Domains (4 selected)
 Tissue_Expression (0 selected)

Red annotation categories denote DAVID defined defaults

Clicking on the chart view for KEGG pathways we can see what pathways within this database our input genes participate in. The column labeled RT denotes related terms (ie. related pathways). Under the Genes column, we can click on the blue bar to view the genes that map to a pathway. The count column tells us how many genes in our list participate in a particular pathway.

DAVID Bioinformatics Resources
Laboratory of Human Retrovirology and Immunoinformatics (LHRI)

Functional Annotation Chart [Help and Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
96 DAVID IDs

Options

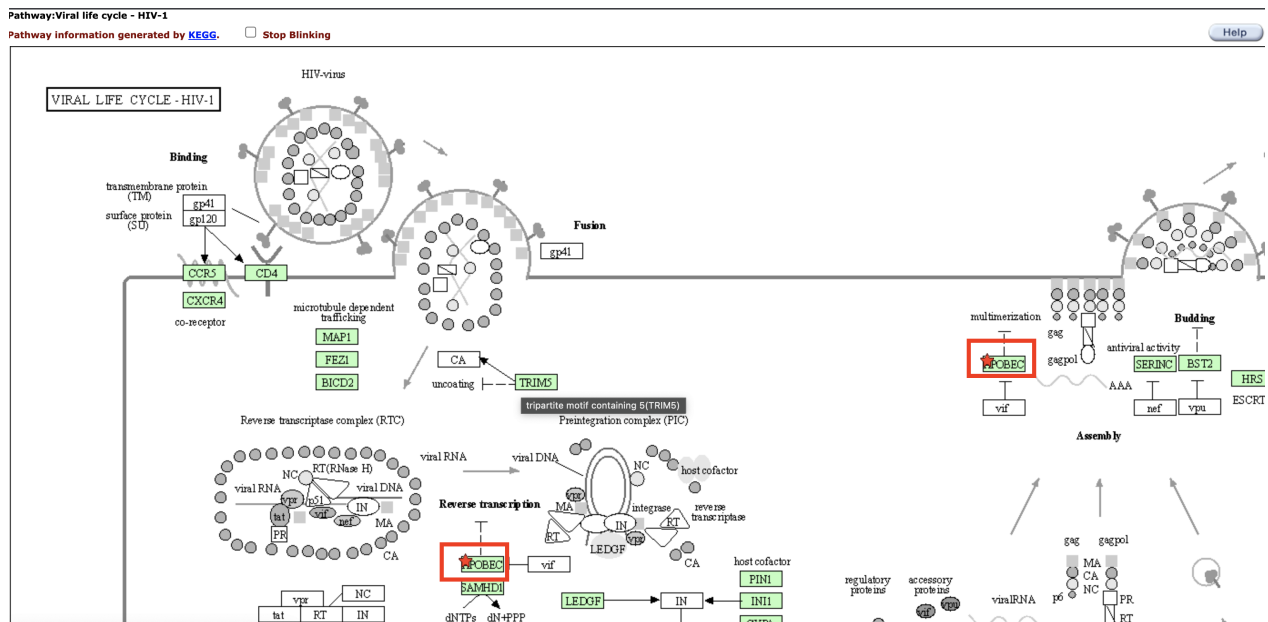
Rerun Using Options | Create Sublist

6 chart records [Download File](#)

Sublist	Category	Term	RT	Genes	Count	%	P-Value	Benjamini
<input type="checkbox"/>	KEGG_PATHWAY	Viral life cycle - HIV-1	RT	<input type="button" value="Bar"/>	5	5.2	3.2E-4	4.3E-2
<input type="checkbox"/>	KEGG_PATHWAY	Human immunodeficiency virus 1 infection	RT	<input type="button" value="Bar"/>	6	6.2	4.9E-3	3.3E-1
<input type="checkbox"/>	KEGG_PATHWAY	Human T-cell leukemia virus 1 infection	RT	<input type="button" value="Bar"/>	5	5.2	2.9E-2	1.0E0
<input type="checkbox"/>	KEGG_PATHWAY	Sulfur relay system	RT	<input type="button" value="Bar"/>	2	2.1	4.1E-2	1.0E0
<input type="checkbox"/>	KEGG_PATHWAY	Sulfur metabolism	RT	<input type="button" value="Bar"/>	2	2.1	5.1E-2	1.0E0
<input type="checkbox"/>	KEGG_PATHWAY	Glycosphingolipid biosynthesis - globo and isoglobo series	RT	<input type="button" value="Bar"/>	2	2.1	7.6E-2	1.0E0

81 gene(s) from your list are not in the output.

Clicking on a pathway under the Term column in Figure 21 takes us to the pathway record in KEGG or which ever database we are viewing. The input gene that participate in a pathway are labeled with a blinking red star.



Functional Annotation Chart

Chart Report is an annotation term focused view which lists annotation terms and their associated genes under study. -- DAVID help documents (https://david.ncifcrf.gov/helps/functional_annotation.html#summary)

Clicking on the Functional Annotation Chart button, we are taken to the page that lists all of the functional annotations that are input genes map onto. Note that for our gene list (hbr_uhr_deg_chr22_up), we get 133 chart records as a default.

Functional Annotation Tool
DAVID Bioinformatics Resources, NIAID/NIH

Home Start Analysis Shortcut to DAVID Tools Technical Center Downloads & APIs Term of Service About DAVID About LHRI

Upload List Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
Homo sapiens(103)

Select Species

List Manager [Help](#)

hbr_uhr_deg_chr22_u

Select List to:

Use Rename
Remove Combine
Show Gene List

Annotation Summary Results

[Help and Tool Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens

96 DAVID IDs
Check Defaults Clear All

- Disease** (2 selected)
- Functional_Annotations** (6 selected)
- Gene_Ontology** (3 selected)
- General_Annotations** (0 selected)
- Interactions** (1 selected)
- Literature** (0 selected)
- Pathways** (3 selected)
- Protein_Domains** (4 selected)
- Tissue_Expression** (0 selected)

Red annotation categories denote DAVID defined defaults

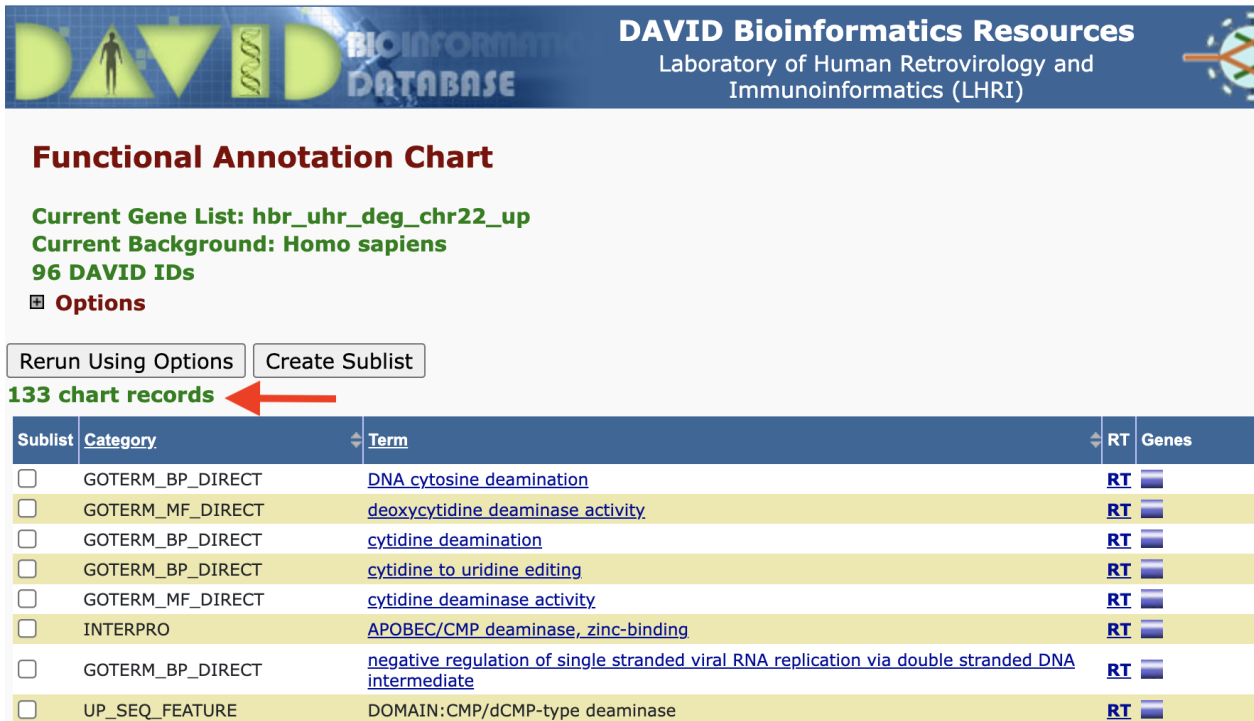
Combined View for Selected Annotation

Functional Annotation Clustering

Functional Annotation Chart

Functional Annotation Table

Adding/removing records from the Functional Annotation Chart: But remember that there are boxes that we can check if we expanded on the annotation categories. If we checked the boxes corresponding to DISGENET, GAD_DISEASE, and GAD_DISEASE_CLASS, and check the Functional Annotation Chart again, we see that we have a few additional annotation records. Thus, what we see in the Functional Annotation Chart is customizable.



Functional Annotation Chart

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
96 DAVID IDs
 Options

Rerun Using Options Create Sublist

133 chart records ←

Sublist	Category	Term	RT	Genes
<input type="checkbox"/>	GOTERM_BP_DIRECT	DNA cytosine deamination	RT	
<input type="checkbox"/>	GOTERM_MF_DIRECT	deoxycytidine deaminase activity	RT	
<input type="checkbox"/>	GOTERM_BP_DIRECT	cytidine deamination	RT	
<input type="checkbox"/>	GOTERM_BP_DIRECT	cytidine to uridine editing	RT	
<input type="checkbox"/>	GOTERM_MF_DIRECT	cytidine deaminase activity	RT	
<input type="checkbox"/>	INTERPRO	APOBEC/CMP deaminase, zinc-binding	RT	
<input type="checkbox"/>	GOTERM_BP_DIRECT	negative regulation of single stranded viral RNA replication via double stranded DNA intermediate	RT	
<input type="checkbox"/>	UP_SEQ_FEATURE	DOMAIN: CMP/dCMP-type deaminase	RT	

Functional Annotation Clustering

Functional annotation clustering works to cluster annotations that share similar genes. If we click on Functional Annotation Clustering in the Annotation Summary Results page then we can see the functional annotation clusters that our input genes map to.

Functional Annotation Clustering

Current Gene List: **hbr_uhr_deg_chr22_up**
 Current Background: **Homo sapiens**
 96 DAVID IDs

Options Classification Stringency **Medium**

16 Cluster(s) [Download File](#)

Annotation Cluster 1	Enrichment Score: 3.95	G	Count	P_Value	Benjamini
<input type="checkbox"/> GOTERM_BP_DIRECT	DNA cytosine deamination	RT	5	6.9E-8	3.9E-5
<input type="checkbox"/> GOTERM_MF_DIRECT	deoxycytidine deaminase activity	RT	5	8.3E-8	1.8E-5
<input type="checkbox"/> GOTERM_BP_DIRECT	cytidine deamination	RT	5	1.6E-7	3.9E-5
<input type="checkbox"/> GOTERM_BP_DIRECT	cytidine to uridine editing	RT	5	1.6E-7	3.9E-5
<input type="checkbox"/> GOTERM_MF_DIRECT	cytidine deaminase activity	RT	5	1.9E-7	2.1E-5
<input type="checkbox"/> INTERPRO	APOBEC/CMP deaminase, zinc-binding	RT	5	4.2E-7	7.0E-5
<input type="checkbox"/> GOTERM_BP_DIRECT	negative regulation of single stranded viral RNA replication via double stranded DNA intermediate	RT	5	5.8E-7	1.1E-4
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN: CMP/dCMP-type deaminase	RT	5	7.8E-7	1.2E-4
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN: CMP/dCMP-type deaminase 1	RT	4	8.1E-7	1.2E-4
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN: CMP/dCMP-type deaminase 2	RT	4	8.1E-7	1.2E-4
<input type="checkbox"/> INTERPRO	Cytidine deaminase-like	RT	5	9.8E-7	7.0E-5
<input type="checkbox"/> INTERPRO	CMP/dCMP deaminase, zinc-binding	RT	5	9.8E-7	7.0E-5
<input type="checkbox"/> GOTERM_BP_DIRECT	DNA demethylation	RT	5	1.9E-6	2.8E-4
<input type="checkbox"/> GOTERM_BP_DIRECT	negative regulation of transposition	RT	5	3.3E-6	4.1E-4
<input type="checkbox"/> GOTERM_BP_DIRECT	innate immune response	RT	11	2.7E-4	2.8E-2
<input type="checkbox"/> KEGG_PATHWAY	Viral life cycle - HIV-1	RT	5	3.2E-4	4.3E-2
<input type="checkbox"/> GOTERM_CC_DIRECT	P-body	RT	5	7.7E-4	1.4E-1
<input type="checkbox"/> UP_SEQ_FEATURE	ACT_SITE: Proton donor	RT	6	3.9E-3	3.6E-1
<input type="checkbox"/> KEGG_PATHWAY	Human immunodeficiency virus 1 infection	RT	6	4.9E-3	3.3E-1
<input type="checkbox"/> UP_KW_BIOLOGICAL_PROCESS	Antiviral defense	RT	5	5.1E-3	2.8E-1
<input type="checkbox"/> GOTERM_MF_DIRECT	hydrolase activity	RT	6	8.1E-3	4.3E-1
<input type="checkbox"/> UP_KW_BIOLOGICAL_PROCESS	Innate immunity	RT	7	1.4E-2	2.9E-1

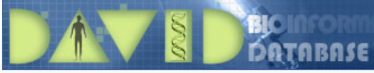

We can fine tune how DAVID clusters the annotations using the parameters below.

- "Clustering Stringency (lowest → highest): A high-level single control to establish a set of detailed parameters involved in functional classification algorithms. In general, the higher stringency setting generates less functional groups with more tightly associated genes in each group, so that more genes will be unclustered. The default setting is Medium, which gives balanced results for most cases based on our studies. Customization allows you to control Advanced options." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)
- "Similarity Term Overlap (any value ≥ 0 ; default = 4): The minimum number of annotation terms overlapped between two genes in order to be qualified for kappa calculation. This parameter is to maintain necessary statistical power to make the kappa value more meaningful. The higher the value, the more meaningful the result is." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)

- "Similarity Threshold (any value between 0 to 1; default = 0.35): The minimum kappa value to be considered significant. A higher setting will lead to more genes going unclustered, which leads to a higher quality functional classification result with fewer groups and fewer gene members. Kappa value of 0.3 starts giving meaningful biology based on our genome-wide distribution study. Anything below 0.3 has a good chance to be noise." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)
- "Initial Group Members (any value ≥ 2 ; default = 4): The minimum gene number in a seeding group, which affects the minimum size of each functional group in the final cluster. In general, the lower value attempts to include more genes in functional groups, and may generate a lot of small size groups." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)
- "Final Group Members (any value ≥ 2 ; default = 4): The minimum gene number in one final group after a 'cleanup' procedure. In general, the lower value attempts to include more genes in functional groups and may generate a lot of small size groups. It cofunctions with previous parameters to control the minimum size of functional groups. If you are interested in functional groups containing only 2 or 3 genes, you need to set it to a very low value. Otherwise, the small group will not be displayed and the genes will go unclustered." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)
- "Multi-linkage Threshold (any value between 0% to 100%; default = 50%): This parameter controls how seeding groups merge with each other, i.e. two groups sharing the same gene members over the percentage will become one group. A higher percentage, in general, gives sharper separation (i.e. it generates more final functional groups with more tightly associated genes in each group). In addition, changing the parameter does not cause additional genes to go unclustered." -- [DAVID functional classification documentations \(https://david.ncifcrf.gov/helps/functional_classification.html\)](https://david.ncifcrf.gov/helps/functional_classification.html)

Functional Annotation Table

Provides a gene-centric view which lists the genes and their associated annotation terms... -- [DAVID help documents](#) (https://david.ncifcrf.gov/helps/functional_annotation.html#summary)

Functional Annotation Table

[Help and Manual](#)

Current Gene List: hbr_uhr_deg_chr22_up
Current Background: Homo sapiens
96 DAVID IDs

92 record(s) [Download File](#)

ENSG00000242247	ADP_ribosylation_factor_GTPase_activating_protein_3(ARFGAP3)	Related Genes	Homo sapiens
BIOCARTA	ADP-Ribosylation Factor,		
GOTERM_BP_DIRECT	intracellular protein transport, protein secretion, vesicle-mediated transport, COPII coating of Golgi vesicle, regulation of catalytic activity,		
GOTERM_CC_DIRECT	Golgi membrane, Golgi apparatus, cytosol, membrane,		
GOTERM_MF_DIRECT	GTPase activator activity, protein binding, metal ion binding,		
INTERPRO	Arf GTPase activating protein,		
KEGG_PATHWAY	Endocytosis,		
SMART	ArfGap,		
UP_KW_BIOLOGICAL_PROCESS	Protein transport, Transport, ER-Golgi transport,		
UP_KW_CELLULAR_COMPONENT	Golgi apparatus, Membrane, Cytoplasm,		
UP_KW_DOMAIN	Coiled coil, Zinc-finger,		
UP_KW_LIGAND	Metal-binding, Zinc,		
UP_KW_MOLECULAR_FUNCTION	GTPase activation,		
UP_KW_PTM	Phosphoprotein,		
UP_SEQ_FEATURE	COMPBIAS:Basic and acidic residues, COMPBIAS:Polar residues, DOMAIN:Arf-GAP, REGION:Disordered, ZN_FING:C4-type,		
ENSG00000133486	C1q and TNF related 6(C1QTNF6)	Related Genes	Homo sapiens
GOTERM_CC_DIRECT	collagen trimer, extracellular space, integral component of membrane, macromolecular complex,		
GOTERM_MF_DIRECT	protein binding, identical protein binding,		
INTERPRO	Complement C1q protein, Collagen triple helix repeat, Tumour necrosis factor-like domain,		
SMART	C1Q,		
UP_KW_CELLULAR_COMPONENT	Membrane, Secreted,		
UP_KW_DOMAIN	Collagen, Signal, Transmembrane, Transmembrane helix,		
UP_KW_PTM	Glycoprotein,		
UP_SEQ_FEATURE	CARBOHYD:N-linked (GlcNac...) asparagine, DOMAIN:C1q, DOMAIN:Collagen-like, REGION:Disordered, TRANSMEM:Helical,		
ENSG00000100206	DNA meiotic recombinase 1(DMC1)	Related Genes	Homo sapiens
GOTERM_BP_DIRECT	DNA recombinase assembly, ovarian follicle development, oocyte maturation, DNA repair, mitotic recombination, synapsis, reciprocal meiotic recombination, male meiosis I, spermatogenesis, spermatid development, female gamete generation, strand invasion, meiotic cell		

Gene Functional Classification Result

DAVID can also generate gene clusters where those gene that cluster together share common annotations.

Gene Functional Classification Result

Current Gene List: hbr_uhr_deg_chr22_up
 Current Background: Homo sapiens
 96 DAVID IDs

Options: Classification Stringency Medium

2 Cluster(s)

Gene Group 1	Enrichment Score: 3.42	RG	T	Heatmap
1	ENSG00000128394	apolipoprotein B mRNA editing enzyme catalytic subunit 3F(APOBEC3F)		
2	ENSG00000243811	apolipoprotein B mRNA editing enzyme catalytic subunit 3D(APOBEC3D)		
3	ENSG00000244509	apolipoprotein B mRNA editing enzyme catalytic subunit 3C(APOBEC3C)		
4	ENSG00000179750	apolipoprotein B mRNA editing enzyme catalytic subunit 3B(APOBEC3B)		
5	ENSG00000239713	apolipoprotein B mRNA editing enzyme catalytic subunit 3G(APOBEC3G)		
Gene Group 2	Enrichment Score: 1.12	RG	T	Heatmap
1	ENSG00000211679	immunoglobulin lambda constant 3 (Kern-Oz+ marker)(IGLC3)		
2	ENSG00000211677	immunoglobulin lambda constant 2(IGLC2)		
3	ENSG00000211666	immunoglobulin lambda variable 2-14(IGLV2-14)		
4	ENSG00000211685	immunoglobulin lambda constant 7(IGLC7)		

87 gene(s) from your list are not in the output.

We can view the cluster information as a heatmap where green represents an association between the gene and an annotation. The black represents no reported association between the gene and an annotation. On the bottom horizontal axis, we have the gene names.

2D View

■ corresponding gene-term association positively reported ■ corresponding gene-term association not reported yet

Download File

Gene	immunoglobulin lambda variable 2-14(IGLV2-14)	immunoglobulin lambda constant 2(IGLC2)	immunoglobulin lambda constant 7(IGLC7)	immunoglobulin lambda constant 3 (Kern-Oz+ marker)
KW-0086-Bence-Jones protein				
GO:0072562-blood microparticle				
GO:0071753-IgM immunoglobulin complex				
GO:0071742-IgE immunoglobulin complex				
GO:0071738-IgD immunoglobulin complex				
GO:0071745-IgA immunoglobulin complex				
GO:0071735-IgG immunoglobulin complex				
GO:0070062-extracellular exosome				
GO:002250-adaptive immune response				
GO:0045087-innate immune response				
tolerability complex, conserved site				
SM00407:IGc1				
IPR003597:Immunoglobulin C1-set				
571-immunoglobulin complex, circulating				
ositive regulation of B cell activation				
GO:0006910-phagocytosis, recognition				
0034987-immunoglobulin receptor binding				
complement activation, classical pathway				
50853-B cell receptor signaling pathway				
009897-external side of plasma membrane				
0:0042742-defense response to bacterium				
DISUFID:Interchain (with heavy chain)				
GO:0006911-phagocytosis, engulfment				
KW-0391-Immunity				
GO:0003823-antigen binding				
GO:0005615-extracellular space				
KW-1280-Immunoglobulin				
GO:0005576-extracellular region				
KW-0964-Secreted				
KW-1064-Adaptive immunity				
DOMAIN:Ig-like				
IPR013783:Immunoglobulin-like fold				
KW-1015-Bisulfide bond				
KW-0393-Immunoglobulin domain				
IPR007110:Immunoglobulin-like domain				
GO:0005886-plasma membrane				
KW-0472-Membrane				
KW-1003-Cell membrane				

We can obtain a table view of the gene clustering results.



Term Report

[Help and Manual](#)

Gene Cluster 1

Current Gene List: **hbr_uhr_deg_chr22_up_genes**

Current Background: **Homo sapiens**

5 DAVID IDs

119 term records

[Download File](#)

Category	Term	RT	Genes in Group	Count	LT	PH	PT	%	P-Value	Fold Enrichment	Genes not in Group
GOTERM_BP_DIRECT	DNA cytosine deamination	RT	5	5	10	19308	100.0	3.6E-14	1.5E3		
GOTERM_MF_DIRECT	deoxycytidine deaminase activity	RT	5	5	10	18869	100.0	4.0E-14	1.5E3		
GOTERM_BP_DIRECT	cytidine deamination	RT	5	5	12	19308	100.0	8.6E-14	1.3E3		
GOTERM_BP_DIRECT	cytidine to uridine editing	RT	5	5	12	19308	100.0	8.6E-14	1.3E3		
GOTERM_MF_DIRECT	cytidine deaminase activity	RT	5	5	12	18869	100.0	9.4E-14	1.3E3		
INTERPRO	APOBEC/CMP deaminase, zinc-binding	RT	5	5	14	19204	100.0	1.8E-13	1.1E3		
GOTERM_BP_DIRECT	negative regulation of single stranded viral RNA replication via double stranded DNA intermediate	RT	5	5	16	19308	100.0	3.1E-13	9.7E2		
UP_SEQ_FEATURE	DOMAIN:CMP/dCMP-type deaminase	RT	5	5	17	20559	100.0	3.2E-13	9.7E2		
INTERPRO	Cytidine deaminase-like	RT	5	5	17	19204	100.0	4.2E-13	9.0E2		
INTERPRO	CMP/dCMP deaminase, zinc-binding	RT	5	5	17	19204	100.0	4.2E-13	9.0E2		
GOTERM_BP_DIRECT	DNA demethylation	RT	5	5	21	19308	100.0	1.0E-12	7.4E2		
GOTERM_BP_DIRECT	negative regulation of transposition	RT	5	5	24	19308	100.0	1.8E-12	6.4E2		
UP_SEQ_FEATURE	DOMAIN:CMP/dCMP-type deaminase 2	RT	4	5	5	20559	80.0	2.8E-11	2.5E3		
UP_SEQ_FEATURE	DOMAIN:CMP/dCMP-type deaminase 1	RT	4	5	5	20559	80.0	2.8E-11	2.5E3		
GOTERM_CC_DIRECT	P-body	RT	5	5	97	20562	100.0	4.7E-10	1.7E2		
KEGG_PATHWAY	Viral life cycle - HIV-1	RT	5	5	63	8164	100.0	3.2E-9	1.0E2		
UP_SEQ_FEATURE	ACT_SITE:Proton donor	RT	5	5	238	20559	100.0	1.8E-8	6.9E1	(1)	
UP_KW_BIOLOGICAL_PROCESS	Antiviral defense	RT	5	5	135	11223	100.0	2.0E-8	6.7E1		
GOTERM_BP_DIRECT	defense response to virus	RT	5	5	233	19308	100.0	2.1E-8	6.6E1		

Introduction to Qiagen Ingenuity Pathway Analysis

Lesson objective

In this lesson, we will continue to learn about pathway analysis using the Qiagen Ingenuity Pathway Analysis (IPA) package. This class will be taught by Qiagen field application scientist.

Example data

For the lesson, we will work with the differential expression analysis results from the human brain reference (HBR) and universal human reference (uhr).

[HBR-UHR differential expression results](#)

For the practice session, we will use the differential expression analysis results from the hcc1395 dataset.

[HCC1395 differential expression results](#)

Accessing IPA

To access IPA, see the [BTEP Bioinformatics Resources for CCR scientists \(https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/ipa/\)](https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/software/ipa/) website.

IPA learning resources

[IPA webinars \(https://digitalinsights.qiagen.com/webinars-and-events/\)](https://digitalinsights.qiagen.com/webinars-and-events/)

[IPA trainings hosted by BTEP \(https://btep.ccr.cancer.gov/on-line-classes-2022/\)](https://btep.ccr.cancer.gov/on-line-classes-2022/)

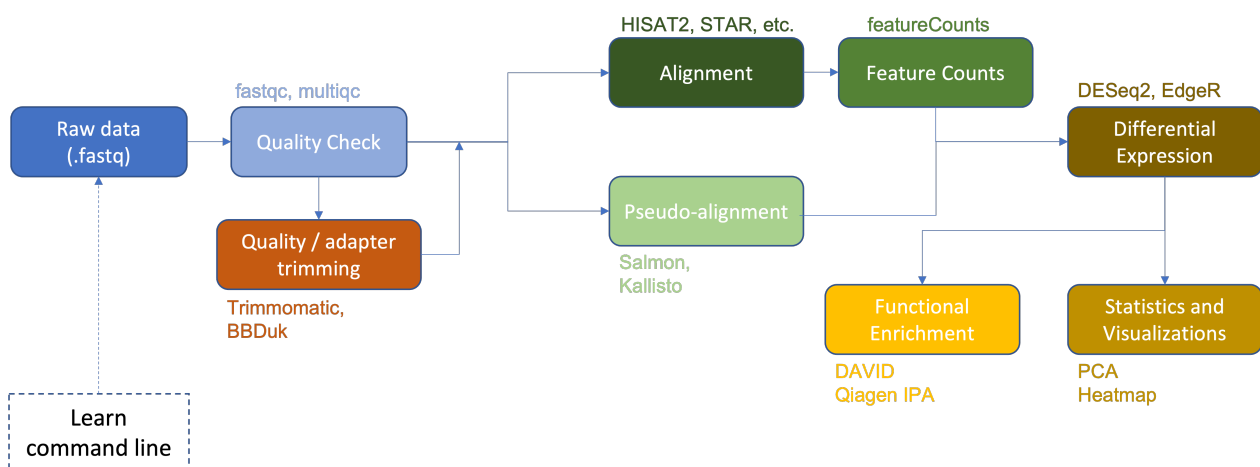
Course Wrap-up

This lesson concludes the Bioinformatics for Beginners course series. Please email us any time at ncibtep@nih.gov for help with your bioinformatics questions or concerns.

Lesson Objectives

1. Short course overview.
2. Review BTEP and course resources.
3. Learn how to login and access the Biostars module.
4. Discuss upcoming BTEP courses
5. Q & A

RNA-Seq overview



Thus far we have:

1. Learned how to interact with our computer via command line.
2. Downloaded raw RNA-Seq data (.fastq files).
3. Examined raw data quality using `fastqc` and `multiqc`.
4. Performed adapter and quality trimming using `Trimmomatic`.
5. Aligned the raw sequences to a reference genome (human chromosome 22 from the GRCh38 version of the human reference genome) using `HISAT2`.
6. Viewed and compared alignments using `IGV`.
7. Generated a gene count matrix using `featureCounts`.
8. Performed differential expression analysis (`DESeq2`, `EdgeR`).
9. Generated a heatmap of differentially expressed genes.

10. Performed functional enrichment using DAVID and Qiagen IPA.

Review of resources

BTEP Resources pages

BTEP seeks to inform and empower researchers, so that you can ultimately tackle some data analyses on your own. We offer a number of resources to accomplish this goal. Take a look at our [resources documentation \(https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/\)](https://btep.ccr.cancer.gov/docs/resources-for-bioinformatics/) to get an idea of the many bioinformatics resources available to help you reach your analysis and training objectives.

Course documentation and resources

The documentation you are currently reading will be accessible in the future from the [BTEP website \(https://btep.ccr.cancer.gov/\)](https://btep.ccr.cancer.gov/).

For this course, there are additional resources worthy of note under the *Additional Resource* tab, including *Further Readings and Tutorials*, instructions for *Logging into Biowulf*, instructions for *Accessing the Biostar Handbook*, and instructions on *Using the Biostars Module for Biowulf*.

The Biostar Handbook

As a reminder, when you registered for the course, you also gained a 6 month subscription to the [Biostar Handbook Collection \(https://www.biostarhandbook.com/index.html\)](https://www.biostarhandbook.com/index.html). This is a fantastic resource covering a range of bioinformatics topics, not just RNA-Seq. At the minimum, consider downloading the book volumes, available as pdfs, before your subscription disappears.

Biostars on Biowulf

For your convenience, we have created a module on Biowulf that includes many of the same programs in the `bioinfo` environment from *The Biostar Handbook*. Instructions for using this module can be found at [Additional Resources](#). Let's briefly review some key points about Biowulf and then take a look at using the Biostars module.

Getting a Biowulf account

The first step to working on Biowulf is getting a Biowulf account. If you intend to analyze your own data, most of you will need a Biowulf account some time in the future.

All NIH employees in the NIH Enterprise Directory (NED) are eligible for a Biowulf account. There is a charge of \$35 per month associated with each account, which is pretty nominal. The instructions for obtaining an account are [here \(https://hpc.nih.gov/docs/accounts.html\)](https://hpc.nih.gov/docs/accounts.html).

Accessing Biowulf

Once you have your Biowulf account, you can connect remotely to Biowulf using a secure shell (SSH) protocol. If you have a macbook, you will need to open the `terminal` application. If you are using Windows 10 or >, you can use `ssh` from the powershell or command prompt. If this fails, consider installing **PuTTY** (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>) and using PuTTY.

The login node will be used to submit jobs to run on the compute nodes that make up Biowulf or to request an interactive job on a compute node. It can also be used for editing / compiling code, file management on a small scale, and file transfer on a small scale.

```
ssh username@biowulf.nih.gov
```

“username” = NIH/Biowulf login username.

If this is your first time logging into Biowulf, you will see a warning statement with a yes/no choice. Type “yes”. Type in your password at the prompt. **NOTE: The cursor will not move as you type your password!**

Where can I find training materials on Biowulf?

Two introductory lessons on Biowulf were presented in this course series. You can access those [here](#) and [here](#). The first contains an introduction to Biowulf including information regarding storage space and the module system. The second provides an introduction to job submission on Biowulf using `swarm` and `sbatch`.

For more information and more detailed training documentation, see [hpc.nih.gov/training/ \(https://hpc.nih.gov/training/\)](https://hpc.nih.gov/training/). Also, the hpc user guides are important resources. Most of your initial questions can be answered simply by referring to and reading [the user guides \(https://hpc.nih.gov/docs/user_guides.html\)](https://hpc.nih.gov/docs/user_guides.html).

What software is available on Biowulf?

Hundreds of scientific programs, databases, and packages are maintained on the NIH HPC (Biowulf). These are mostly accessible as modules.

To see a list of available software in modules use

```
module avail
module avail [appname|string|regex]
module -d
```

To load a module

```
module load appname
module load appname/version
```

To see loaded modules

```
module list
```

To unload modules

```
module unload appname
module purge #(unload all modules)
```

Note: you may also create and use your own modules.

For a list of available software, see hpc.nih.gov/apps/ (<https://hpc.nih.gov/apps/>). For more information about working with modules, see hpc.nih.gov/apps/modules.html (<https://hpc.nih.gov/apps/modules.html>).

Using the Biostars Module

As mentioned above, [instructions for using the Biostar module on Biowulf can be found in the course documents](#). The Biostars module loads the software used in this course series. A list of the software included in the module can be found [in the course documentation](#). Most of the tools / programs in this list can be loaded as independent modules.

Let's see how we can use the Biostars module to work with course materials.

First, as we have already logged into Biowulf, we need to get an interactive session.

1. Use `sinteractive` to work on an interactive node. This will result in 4GB of memory and 2 CPUs.

Note: If you are planning to use the `sratoolkit` to download data from the SRA, you will need to allocate local scratch space (`sinteractive --gres=lscratch:30`).

`sinteractive` allows us access to a Biowulf computational mode in an interactive fashion rather than relying on job submission scripts via `sbatch`.

- Once we have an interactive node, we need to run the `run_biostars.sh` script.
- 2.

```
source /data/classes/BTEP/apps/biostars/1.0/run_biostars.sh
```

This will do the following:

1. Runs a set of commands to setup the terminal environment.
2. Creates a data directory environmental variable (`$DATA`) where you can gain access to course data and an environmental variable (`$CODE`) where you can find specific scripts associated with the course, for example, the R wrapper scripts.
3. Runs `module use /data/classes/BTEP/apps/modules`
4. Runs `module load biostars`

Note: If you want to use the biostars module for other purposes or you want to submit jobs via `sbatch`, skip **Step 2**. You can load the module with the following:

1. ``module use /data/classes/BTEP/apps/modules``
2. ``module load biostars``
3. ``module help biostars``

Let's check out `$DATA` and `$CODE`.

```
ls -l $DATA
ls -l $CODE
```

Upcoming BTEP classes

- Introduction to R
 - January 23rd (M,W 1-2 pm)
- Unix and Biowulf
 - January 24th (T,Th 1-2pm)
- Introduction to Bioinformatics Resources
 - January 12th (Th 1-2 pm)

Help Sessions



Lesson 2 Practice

The instructions that follow were designed to test the skills you learned in Lesson 2. Thus, the primary focus will be navigating directories and manipulating files.

1. Let's navigate our files using the command line. Begin in your home directory.
2. Copy the `Practice_L2` directory (`/data/Practice_Sessions/Practice_L2`) and all of its contents to your home directory.

{{Sdet}}

Solution{{Esum}}

```
cp -r /data/Practice_Sessions/Practice_L2 ~
```

{{Edet}}

3. Change directories to `Practice_L2/`.

{{Sdet}}

Solution{{Esum}}

```
cd ./Practice_L2
```

{{Edet}}

4. List the contents of `Practice_L2`. If there are files, when were they last modified and what is the file size?

{{Sdet}}

Solution{{Esum}}

```
ls -lh
```

The `-lh` flag will allow you to obtain a list of directory content in long format, which provides information including the date the file was last modified and the file size.

{{Edet}}

5. Rename `sample_names.txt` to `treatment_groups.txt`.

```
{{Sdet}}
```



```
Solution{{Esum}}
```

```
mv sample_names.txt treatment_groups.txt
```

```
{{Edet}}
```

6. Within `Practice_L2`, create a new directory called `Analysis`.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
mkdir Analysis
```

```
{{Edet}}
```

7. Copy `treatment_groups.txt` to the newly created `Analysis` directory.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cp treatment_groups.txt ./Analysis
```

```
{{Edet}}
```

8. List the contents of the `Analysis` directory.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
ls -l ./Analysis
```

```
{{Edet}}
```

9. Change directories to `Analysis`. What is the path to `Analysis`?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cd Analysis
pwd
```



{{Edet}}

10. Change to the data directory.

{{Sdet}}

Solution{{Esum}}

```
cd ../data
```

{{Edet}}

11. List the contents of data.

{{Sdet}}

Solution{{Esum}}

```
ls
```

{{Edet}}

12. View A_R1.fastq.

{{Sdet}}

Solution{{Esum}}

```
less A_R1.fastq
```

{{Edet}}

13. Copy and paste the first line of A_R1.fastq into a new file called Line1.txt using keyboard shortcuts and nano.

14. Move Line1.txt to Practice_L2 using a relative file path.

{{Sdet}}

Solution{{Esum}}

```
mv Line1.txt ..
```



{{Edet}}

15. Change to the `Practice_L2` directory.

{{Sdet}}

Solution{{Esum}}

```
cd ..
```

{{Edet}}

16. Remove the `Analysis` directory.

{{Sdet}}

Solution{{Esum}}

```
rm -i -r Analysis
```

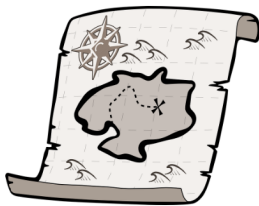
{{Edet}}



Lesson 4 Practice

For today's practice, we are going to embark on a Unix treasure hunt created by the **Sanders Lab** (<https://sanderslab.github.io/code/>) at the University of California San Francisco. Note: the treasure hunt materials can be obtained directly from the Sanders lab code repository linked above.

UNIX treasure hunt tutorial



This perl script will install a series of directories and clues that teaches basic UNIX command line skills including `cd`, `ls`, `grep`, `less`, `head`, `tail`, and `nano`. Run the perl script from the command line on a UNIX based machine (e.g. Mac or Linux) using the command: `perl treasureHunt_v2.pl`. Then use `ls` to find the first clue. A PDF of command line commands is also available to download.

- Source
- Manual

To begin create a directory called `treasure_hunt` in your home directory and run the perl script in `/data/Practice_Sessions/Practice_L4` from the `treasure_hunt` directory.

{{Sdet}}

Solution{{Esum}}

```
mkdir treasure_hunt
cd treasure_hunt
perl /data/Practice_Sessions/Practice_L4/treasureHunt_v2.pl
ls -l
```

{{Edet}}

Read the first clue and begin.

Recommendation: Create an environment variable to store the path to the treasure hunt directory to facilitate movement through the directory.

{{Sdet}}

Solution{{Esum}}

```
THUNT=`pwd`  
echo $THUNT
```



{{Edet}}

When you have found the treasure, answer or do the following:

1. How many words are in the last line of the file containing the treasure?

{{Sdet}}

Solution{{Esum}}

```
tail -n 1 openTheBox.txt | wc -w
```

{{Edet}} 2. Save the last line to a new file called `finallyfinished.txt` without copying and pasting.

{{Sdet}}

Solution{{Esum}}

```
tail -n 1 openTheBox.txt > finallyfinished.txt
```

{{Edet}}

3. Now append the first line to the same file that you just saved the last line.

{{Sdet}}

Solution{{Esum}}

```
head -n 1 openTheBox.txt >> finallyfinished.txt
```

{{Edet}}

Congratulations! You have found the treasure and have gained some useful unix practice throughout your hunt.



Lesson 5 Practice

The following can be used to practice skills learned in Lesson 5.

Login to Biowulf

If you are already logged in, exit the remote connection and reconnect. Remember, you must be on the NIH network.

{{Sdet}}

Solution{{Esum}}

```
ssh username@biowulf.nih.gov
```

{{Edet}}

Let's run *fastqc* (<https://hpc.nih.gov/apps/fastqc.html>), a quality control program, on the files we downloaded from the SRA.

Using sbatch

Start a new script, named `fastqc.sh` in the same directory in which you downloaded data from Lesson 5.

The command you will include in the script is as follows:

```
mkdir fastqc
fastqc -o ./fastqc/ -t 4 *.fastq
```

This command will output the `fastqc` results to a directory named `fastqc` inside the current working directory. It will also run using four threads and will run on all `fastq` files in your working directory.

You need edit this script in order to submit as a job on Biowulf. What is missing?

{{Sdet}}

Solution{{Esum}}



```
#!/bin/bash
#SBATCH --cpus-per-task=4

module load fastqc
mkdir fastqc
fastqc -o ./fastqc/ -t 4 *.fastq
```

{{Edet}}

Submit the job.

{{Sdet}}

Solution{{Esum}}

```
sbatch fastqc.sh
```

{{Edet}}

How can we check on our job? What is the job's status? How much memory is it using?

{{Sdet}}

Solution{{Esum}}

```
squeue -u $USER
sjobs -u $USER
```

{{Edet}}

How can we cancel this job?

{{Sdet}}

Solution{{Esum}}

```
scancel job-id
```

where `job-id` is the id of the job. Check the output of `squeue -u $USER` if you are unsure what the job id is.

{{Edet}}

Accessing the Biostars module on Biowulf

We have created a Biostars module on Biowulf for your convenience. Instructions for using this module are [here](#), and the software included in the module are listed [here](#).

Let's get an interactive session and see how we can use the module.

```
sinteractive
```

Also, we have created a script to set up your shell and load the module, use

```
source /data/classes/BTEP/apps/biostars/1.0/run_biostars.sh
```

This creates a \$DATA environment variable holding the path to many of the files from class. We try to update this, but please let us know if something is missing.

```
ls $DATA
```

Let's try a command.

```
fastqc -h
```

Additional practice materials from hpc.nih.gov

- Submit a job using `sbatch`. (https://hpc.nih.gov/training/intro_biowulf/hands-on-sbatch.html)
- Submit a multi-threaded job. (https://hpc.nih.gov/training/intro_biowulf/hands-on-multi.html)
- Submit a `swarm` job. (https://hpc.nih.gov/training/intro_biowulf/hands-on-swarm.html)



Lesson 6 Practice

The following was designed to practice skills learned in lesson 6.

Find the data

Here (https://journals.asm.org/doi/full/10.1128/mSystems.00065-20?rfr_dat=cr_pub+0pubmed&url_ver=Z39.88-2003&rfr_id=ori%3Arid%3Acrossref.org) is a paper examining the relationship between the oral microbiome and nasopharyngeal carcinoma. Where can you find the associated data?

Notice that the data was originally submitted to the ENA.

Download the data

Make a directory called `Lesson6_practice` and change directories.

{{Sdet}}

Solution{{Esum}}

```
mkdir Lesson6_practice
cd Lesson6_practice
```

{{Edet}}

Navigate to the **NCBI website** (<https://www.ncbi.nlm.nih.gov/>) and grab the accession information for the associated BioProject.

1. Download the first 10 samples using `fastq-dump` with `parallel`.

{{Sdet}}

Solution{{Esum}}

You can download the accession list directly from the SRA Run Selector.

From the command line:

```
esearch -db sra -query PRJEB37445 | efetch -format runinfo | cut -f :  
cat accessions.txt | parallel -j 1 fastq-dump --split-3 {}
```

Or the same command with prefetch...

```
cat accessions.txt | parallel -j 1 'prefetch {} | fastq-dump --split-
```

{{Edet}}

1. Download five samples with the aid of the sra-explorer.

Navigate to the ENA. How might you go about downloading the data?



Lesson 7 Practice

In Lesson 7, you learned how to download and work with archived and compressed files. To practice what you have learned, we will use the ERCC spike in control data, which Istvan Albert, creator of the Biostar Handbook, has reframed as the "Golden Snidget", "a magical golden bird with fully rotational wings."

Task 1: Create a new directory

Create a directory called `golden` and change directories.

{{Sdet}}

Solution{{Esum}}

```
mkdir golden
cd golden
```

{{Edet}}

Task 2: Download the "Golden Snidget" reference data

Now, grab the reference files from <http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz>. This time try out `wget`. If you aren't sure how to use `wget`, how might you find out?

{{Sdet}}

Solution{{Esum}}

```
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz
```

What does the `-nc` flag do?

```
man wget
```

or

```
wget --help
```



-nc stands for "no clobber", which keeps `wget` from downloading and overwriting an existing file of the same name. {{Edet}}

Unpack the reference genome (`golden.genome.tar.gz`).

{{Sdet}}

Solution{{Esum}}

```
tar -xvf golden.genome.tar.gz
```

{{Edet}}

What did this produce?

Just for fun, let's rearchive and zip the data we just packed, name it `funtimes.ref.tar.gz`. How might we do this using `tar`? Check the help information. Alternatively, try google.

{{Sdet}}

Solution{{Esum}}

```
tar --help
tar -czvf funtimes.ref.tar.gz refs
```

{{Edet}}

What are the file sizes of `golden.genome.tar.gz` and `funtimes.ref.tar.gz`?

Task 3: Download the "Golden Snidget" reads

Get the reads from <http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz>. You will also need to unpack the file.

{{Sdet}}

Solution{{Esum}}

```
wget -nc http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz
tar -xvf golden.reads.tar.gz
```

{{Edet}}



What did this produce? List the file contents.

{{Sdet}}

Solution{{Esum}}

```
ls -lh
```

{{Edet}}

Lastly, compress the fastq files using `gzip`.

{{Sdet}}

Solution{{Esum}}

```
gzip reads/*.fq
```

{{Edet}}



Lesson 9 Practice

Objectives

In this practice session, we will apply our knowledge to

- learn about the reference genome and annotation file for the Golden Snidget dataset
- visualize the Golden Snidget genome using the Integrative Genome Viewer (IGV) - instructor will demo this and you can practice on your own after getting IGV installed.

Create a directory for the Golden Snidget analysis

Before getting started, let's create a folder called snidget within the ~/biostar_class directory to conduct our analysis. To do this, we need to first go to the ~/biostar_class folder, how do you check where you are and change into this folder.

{{Sdet}}

Solution{{Esum}}

```
pwd
```

If you are not in the ~/biostar_class folder, then change into it.

```
cd ~/biostar_class
```

{{Edet}}

Next, we will create the directory snidget within the ~/biostar_class folder. Take a moment to see if you can do this.

{{Sdet}}

solution{{Esum}}

```
mkdir snidget
```

{{Edet}}

Now, we need to change into the "snidget" directory

```
{{Sdet}}
```



```
solution{{Esum}}
```

```
cd snidget
```

```
{{Edet}}
```

Where is my data?

The Golden Snidget reference genome is located at <http://data.biostarhandbook.com/books/rnaseq/data/golden.genome.tar.gz>. Can you download and extract?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

Download

```
wget http://data.biostarhandbook.com/books/rnaseq/data/golden.genome
```

OR

```
curl http://data.biostarhandbook.com/books/rnaseq/data/golden.genome
```

Unpack

```
tar -xvf golden.genome.tar.gz
```

```
{{Edet}}
```

After unpacking `golden.genome.tar.gz`, what do you see?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
ls -l
```



```
total 60
-rw-rw-r-- 1 joe joe 57462 Feb  5  2020 golden.genome.tar.gz
drwxrwxr-x 1 joe joe   70 Oct 25 00:06 refs
```



In addition to the `golden.genome.tar.gz` file, we have a `refs` folder. The `refs` folder contains the reference genome (`genome.fa`), reference transcriptome (`transcriptome.fa`), and annotations (`features.gff`) for the Golden Snidget.

```
ls refs
```

```
features.gff  genome.fa  transcripts.fa
```

{{Edet}}

The Golden Snidget sequencing reads are located at <http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.tar.gz>. Can you download and extract?

{{Sdet}}

Solution{{Esum}}

Download

```
wget http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.t
```

OR

```
curl http://data.biostarhandbook.com/books/rnaseq/data/golden.reads.t
```

Unpack

```
tar -xvf golden.reads.tar.gz
```

{{Edet}}

After downloading and unpacking `golden.reads.tar.gz`, what do you see?

{{Sdet}}

Solution{{Esum}}

```
ls -l
```



We see the two tar.gz files that were downloaded and a new folder called reads.

```
total 117384
-rw-rw-r-- 1 joe joe      57462 Feb  5  2020 golden.genome.tar.gz
-rw-rw-r-- 1 joe joe 120138017 Oct 25 00:18 golden.reads.tar.gz
drwxrwxr-x 1 joe joe      336 Oct 25 00:19 reads
drwxrwxr-x 1 joe joe       70 Oct 25 00:06 refs
```

The reads folder contains the FASTQ (fq) files for this dataset. We will be working with these in the next lesson.

```
ls reads
```

```
BORED_1_R1.fq  BORED_2_R1.fq  BORED_3_R1.fq  EXCITED_1_R1.fq  EXCITEI
BORED_1_R2.fq  BORED_2_R2.fq  BORED_3_R2.fq  EXCITED_1_R2.fq  EXCITEI
```

{{Edet}}

Find out some details about the Golden Snidget genome and transcriptome

Now that we have downloaded the Golden Snidget reference files let's take a moment to get to know the references. First, change into the refs folder. How do we do this from the ~/biostar_class/snidget directory.

{{Sdet}}

Solution{{Esum}}

```
cd refs
```

{{Edet}}

How many sequences are in the Golden Snidget genome?

How many bases are in the Golden Snidget genome (ie. what is the genome size for the Golden Snidget)?

{{Sdet}}

Solution{{Esum}}

```
seqkit stats genome.fa
```

file	format	type	num_seqs	sum_len	min_len
/data/golden/refs/genome.fa	FASTA	DNA	1	128,756	128,756

The Golden Snidget genome has 1 sequence composed of 128,756 bases.

{{Edet}}

How many transcripts does the Golden Snidget have?

How many bases does the longest transcript have?

How many bases does the shortest transcript have?

{{Sdet}}

Solution{{Esum}}

```
seqkit stats transcripts.fa
```

file	format	type	num_seqs	sum_len	min_len
/data/golden/refs/transcripts.fa	FASTA	DNA	92	82,756	273

The Golden Snidget has 92 transcripts.

The longest transcript is 2,022 bases.

The shortest transcript is 273 bases.

{{Edet}}

Note: if you grep for > in Unix, be sure to put quotes around it.

grep ">" NOT grep >

Is there an alternative way to get the number of transcripts in the Golden Snidget (ie. without using seqkit)?

{{Sdet}}

Solution{{Esum}}

```
grep ">" transcripts.fa | wc -l
```

```
92
```

{{Edet}}

The goal of the Golden Snidget RNA sequencing experiment is to find genes that are differentially expressed when the Golden Snidget is EXCITED compared to when it is BORED. Looking at the transcript names, what can you tell about a particular transcript in the EXCITED and BORED state? What would you expect the differential gene expression analysis to tell us when we get to this later on in the course? You will need to take a look at the features.gff file for this.

{{Sdet}}

Solution{{Esum}}

```
less features.gff
```

Note: hit Q to exit less.

Look at the gene or transcript names on the last column of the annotations file (gene names and transcripts names are the same in this dataset). Take for example AAA-750000-UP-4, the transcript name is telling us that

- In the BORED state, there are 750000 copies of the transcript
- In the EXCITED state, the expression of this transcript is UP 4 times ($4 \times 750000 = 3000000$ copies)
- Where the expression in the EXCITED state is down we will see DOWN in the transcript names

{{Edet}}

Visualizing Golden Snidget genome and transcriptome in IGV



Let's open IGV locally on our computer. Then we will copy the Golden Snidget refs folder to our public directory so we can download and use these locally. Remember the location on your computer to which the files were downloaded. See if you can remember how to copy the Golden Snidget reference to the public directory. Hint, it may be easier to do this from the ~/biostar_class/snidget directory.

{{Sdet}}

Solution{{Esum}}

```
cd ~/biostar_class/snidget
```

```
cp -r refs ~/public
```

{{Edet}}

After we have successfully copied the refs folder to public, click to open it and right click on each of the files and click

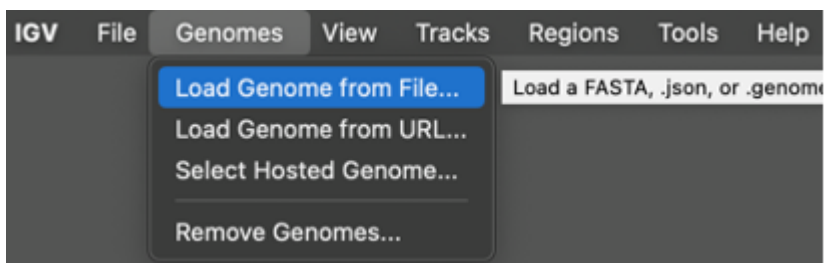
- "Save link as" (if on Google Chrome or Firefox) - include the appropriate file extension when saving
- "Download Linked File as" (if on Safari)

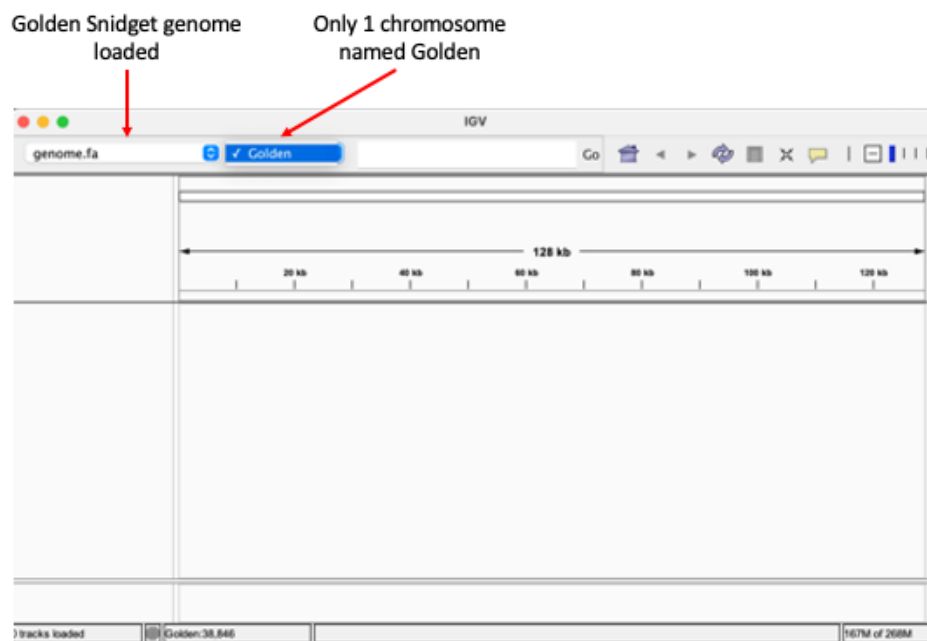
Load the Golden Snidget reference

The first step in using IGV is to load our reference genome. Take some time to see if you recall how to do this.

{{Sdet}}

Solution{{Esum}}



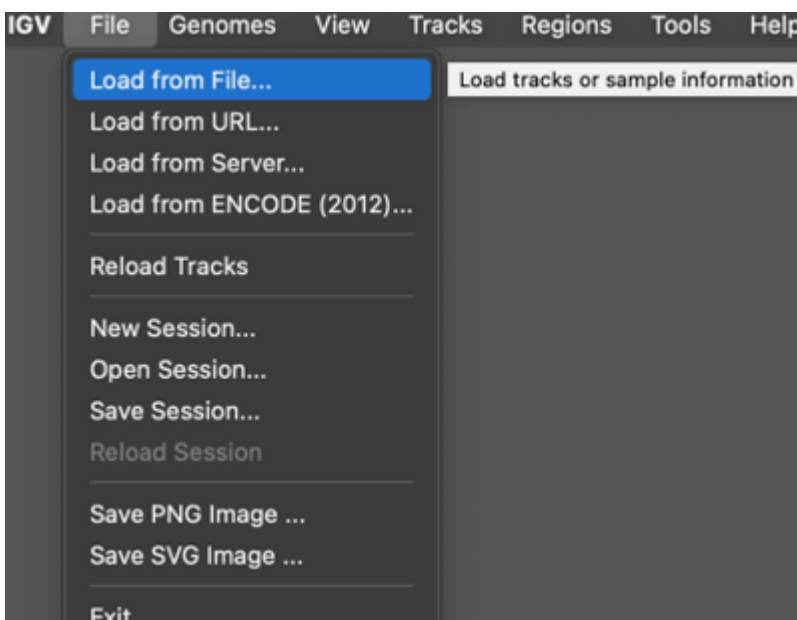


{{Edet}}

After loading the genome, let's view the transcripts in IGV and see how they line up in the genome. Take a moment to see if you recall how to do this.

{{Sdet}}

Solution{{Esum}}



Then choose the features.gff file and the result will look like the image below.

We can jump to different parts of the genome by typing in new coordinates or a gene name

This top track is the genome

features.gff

The features.gff file is now shown as a track

Blue bars represent genes from the Golden Snidget

AAA-750000-UP-4 ABA-187500-UP-4 ACA-46875-UP-4

{{Edet}}

Take sometime to explore IGV (zoom in, search for a transcript, pan around...)



Lesson 10 Practice

Objectives

In this lesson, we introduced the structure of the FASTQ file and learned to assess quality of raw sequencing data using FASTQC. Here, we will practice what we learned using the Golden Snidget dataset.

Where is my data?

Recall that the Golden Snidget data resides in `~/biostar_class/snidget` folder. Can you change into the folder and find where the sequencing reads are (ie. in which folder they are located)?

{{Sdet}}

Solution{{Esum}}

```
cd ~/biostar_class/snidget
```

The sequencing reads are in the reads folder

```
cd reads
```

{{Edet}}

How many sequencing (fq) files do we have?

{{Sdet}}

Solution{{Esum}}

```
ls
```

```
12
```

{{Edet}}



From the names of the FASTQ (fq) files, are these from paired or single end sequencing?

{{Sdet}}

Solution{{Esum}}

```
Paired
```

{{Edet}}

Can you find the first sequencing read in the file BORED_1_R1.fq? If you can, can you identify the sequencing header line and the quality score line?

{{Sdet}}

Solution{{Esum}}

```
head -4 BORED_1_R1.fq
```

{{Edet}}

From what you know about the structure of FASTQ files, how many reads are in BORED_1_R1.fq? There are two ways you can find out.

{{Sdet}}

Solution{{Esum}}

```
grep @HWI BORED_1_R1.fq | wc -l
```

```
112193
```

We can also use seqkit stats

```
seqkit stat BORED_1_R1.fq
```

{{Edet}}



What can we do to get stats such as the number of reads and read length for all of the Golden Snidget FASTQ files?

{{Sdet}}

Solution{{Esum}}

```
seqkit stats BORED_*.fq EXCITED_*.fq
```

file	format	type	num_seqs	sum_len	min_len	avg_len
BORED_1_R1.fq	FASTQ	DNA	112,193	11,219,300	100	100
BORED_1_R2.fq	FASTQ	DNA	112,193	11,219,300	100	100
BORED_2_R1.fq	FASTQ	DNA	137,581	13,758,100	100	100
BORED_2_R2.fq	FASTQ	DNA	137,581	13,758,100	100	100
BORED_3_R1.fq	FASTQ	DNA	123,093	12,309,300	100	100
BORED_3_R2.fq	FASTQ	DNA	123,093	12,309,300	100	100
EXCITED_1_R1.fq	FASTQ	DNA	237,018	23,701,800	100	100
EXCITED_1_R2.fq	FASTQ	DNA	237,018	23,701,800	100	100
EXCITED_2_R1.fq	FASTQ	DNA	158,009	15,800,900	100	100
EXCITED_2_R2.fq	FASTQ	DNA	158,009	15,800,900	100	100
EXCITED_3_R1.fq	FASTQ	DNA	196,673	19,667,300	100	100
EXCITED_3_R2.fq	FASTQ	DNA	196,673	19,667,300	100	100

{{Edet}}

How do we visualize quality metrics for the Golden Snidget sequencing reads?

{{Sdet}}

Solution{{Esum}}

```
fastqc BORED_*.fq EXCITED_*.fq
```

{{Edet}}

Look at the quality check report for BORED_1_R1.fq. But first copy it to the ~/public folder.

{{Sdet}}

Solution{{Esum}}

```
cp BORED_1_R1_fastqc.html ~/public
```



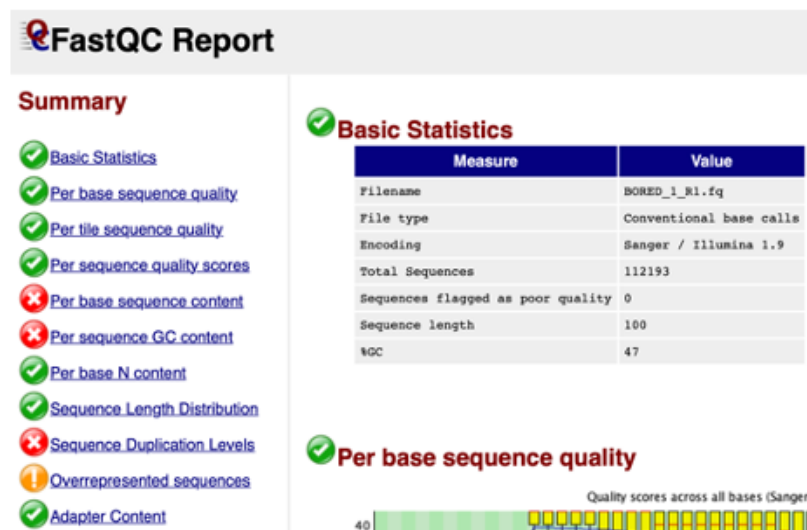
{{Edet}}

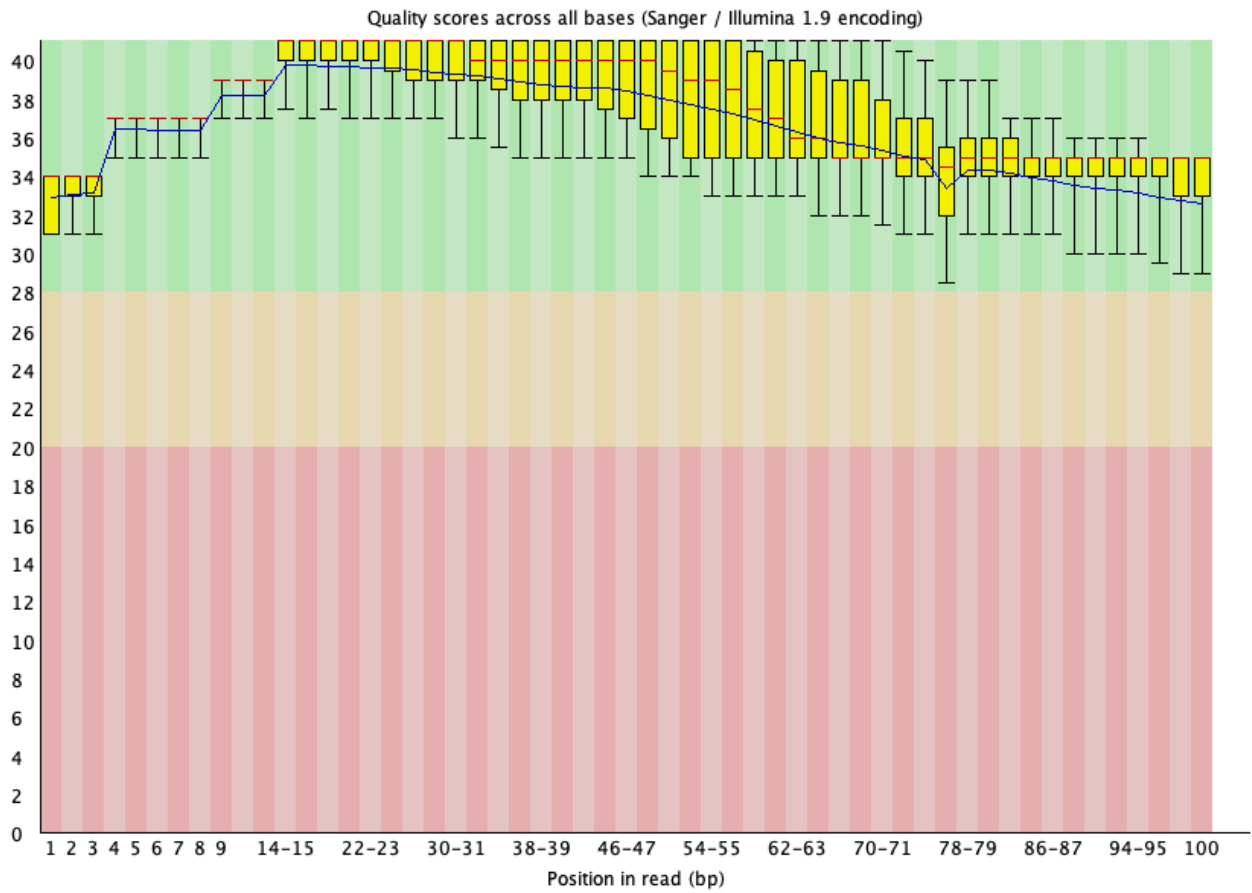
- How many modules passed QC?
- Do the total number of sequences and does sequence length match those obtained from seqkit stats?
- How are the qualities of the sequencing reads?
- What are the overrepresented sequences? BLAST one of the sequences to find out.
- Is there adapter contamination?

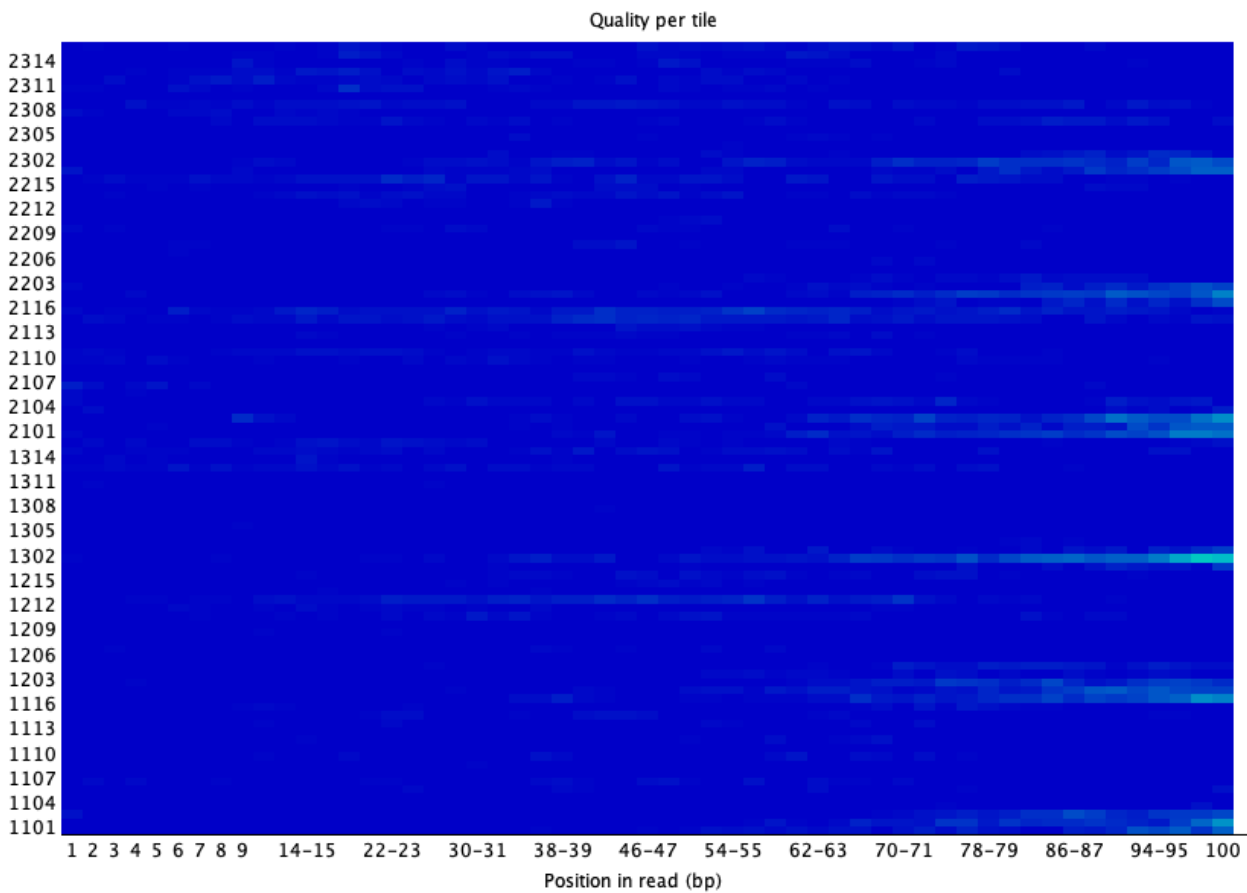
{{Sdet}}

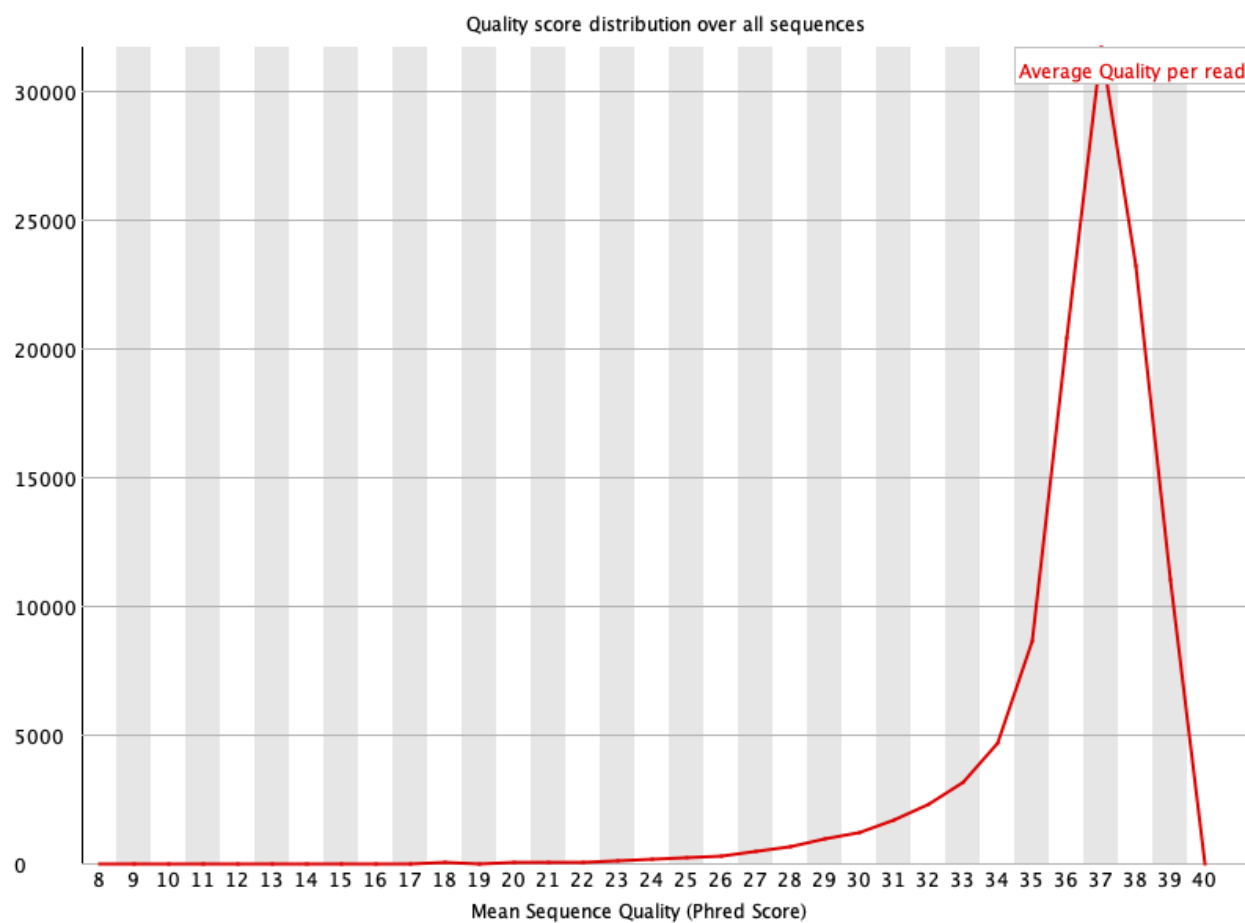
Solution{{Esum}}

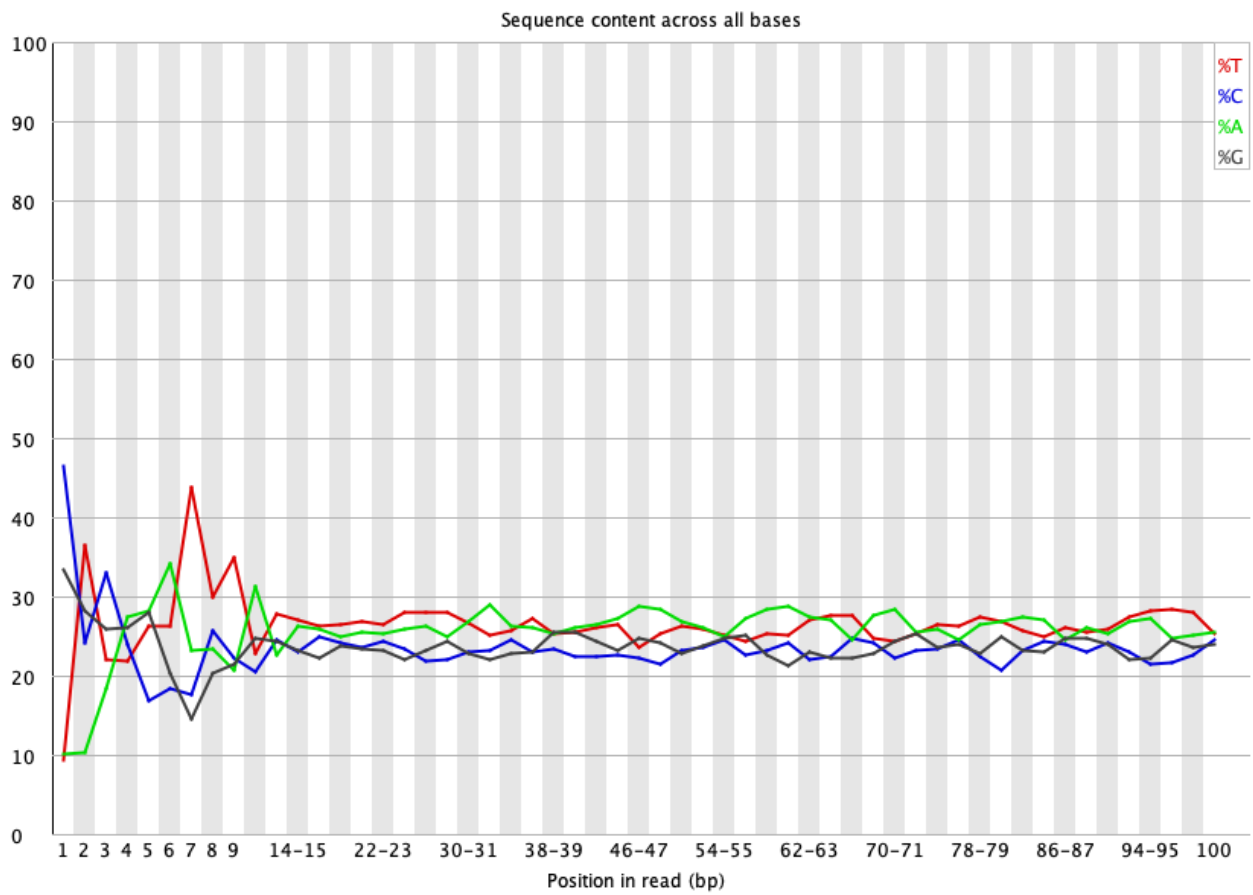
FASTQC results for BORED_1_R1.fq

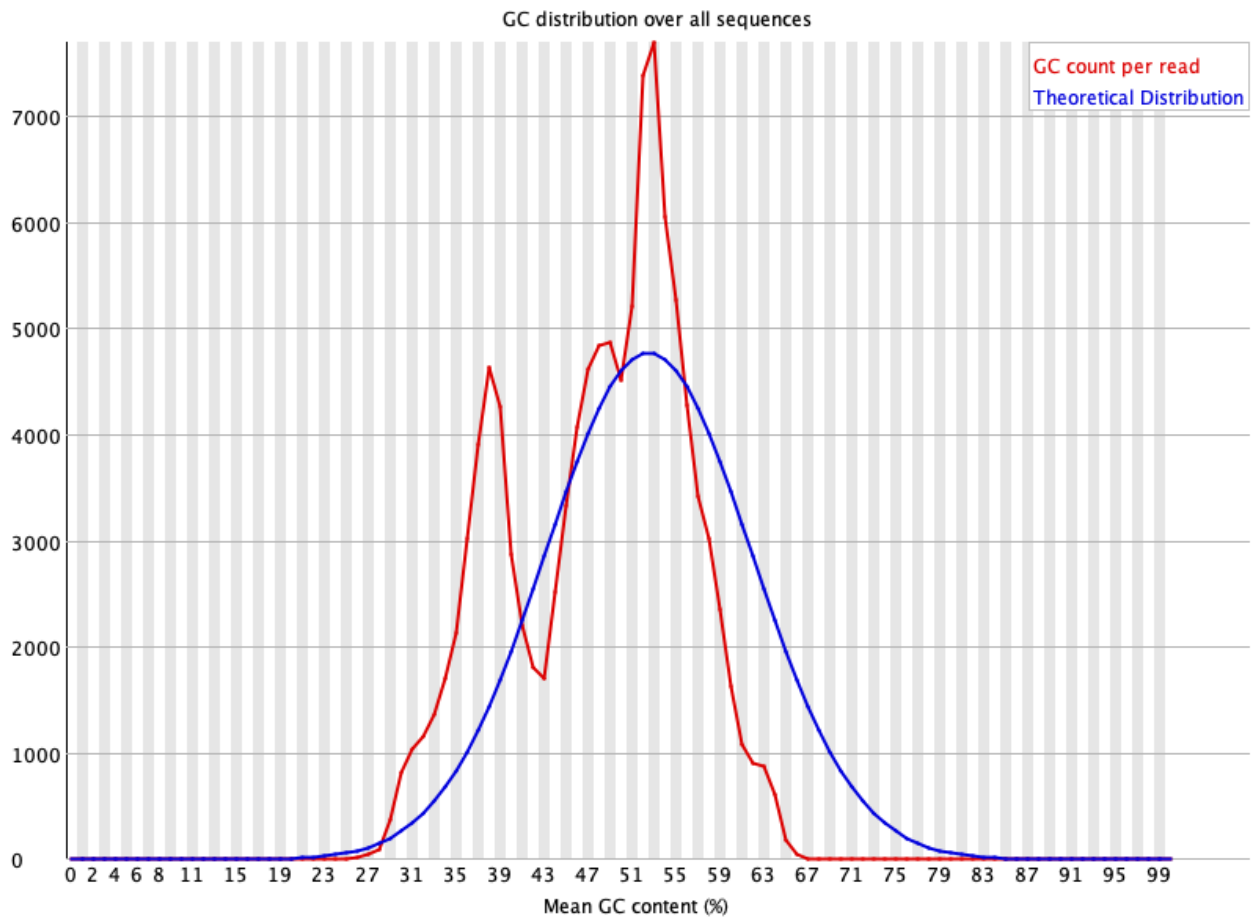


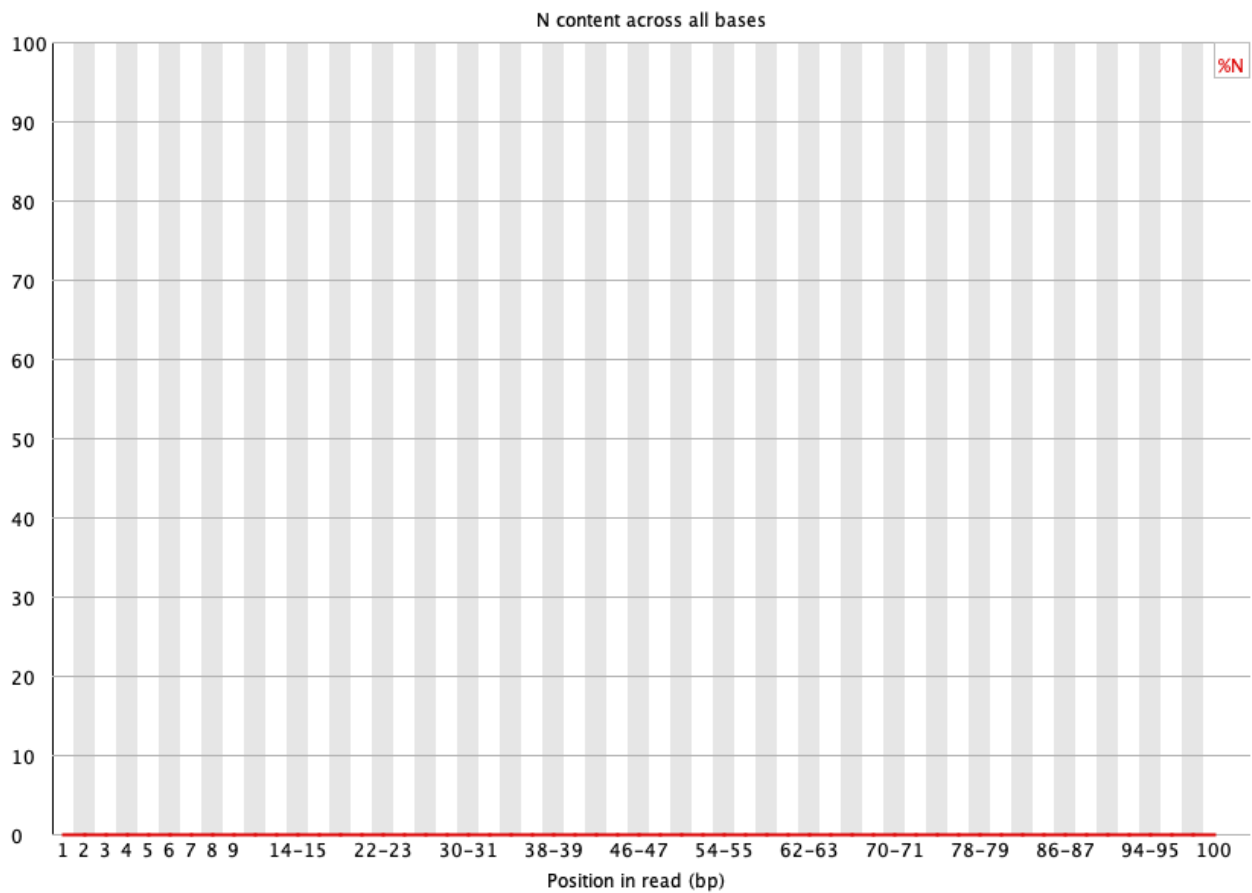


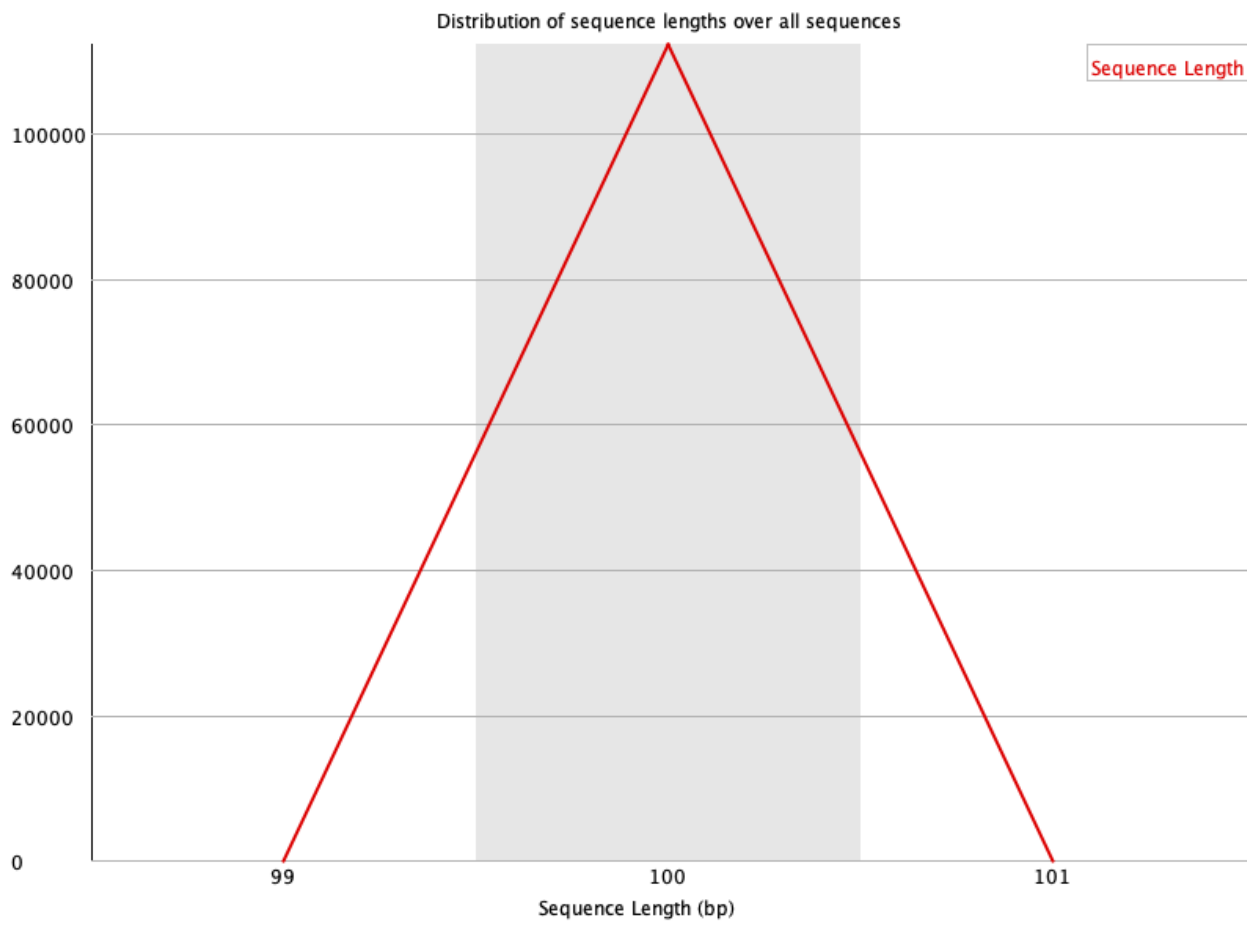


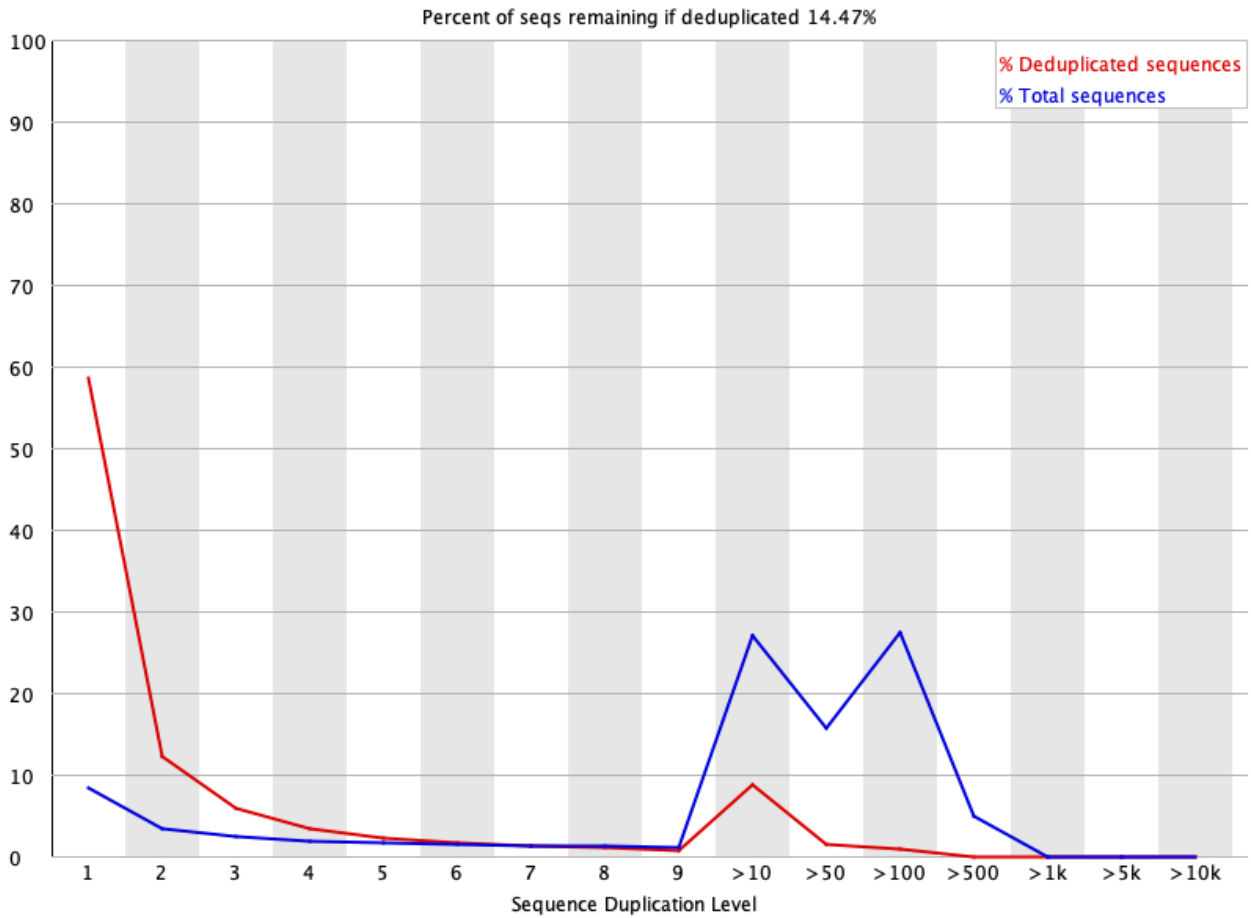






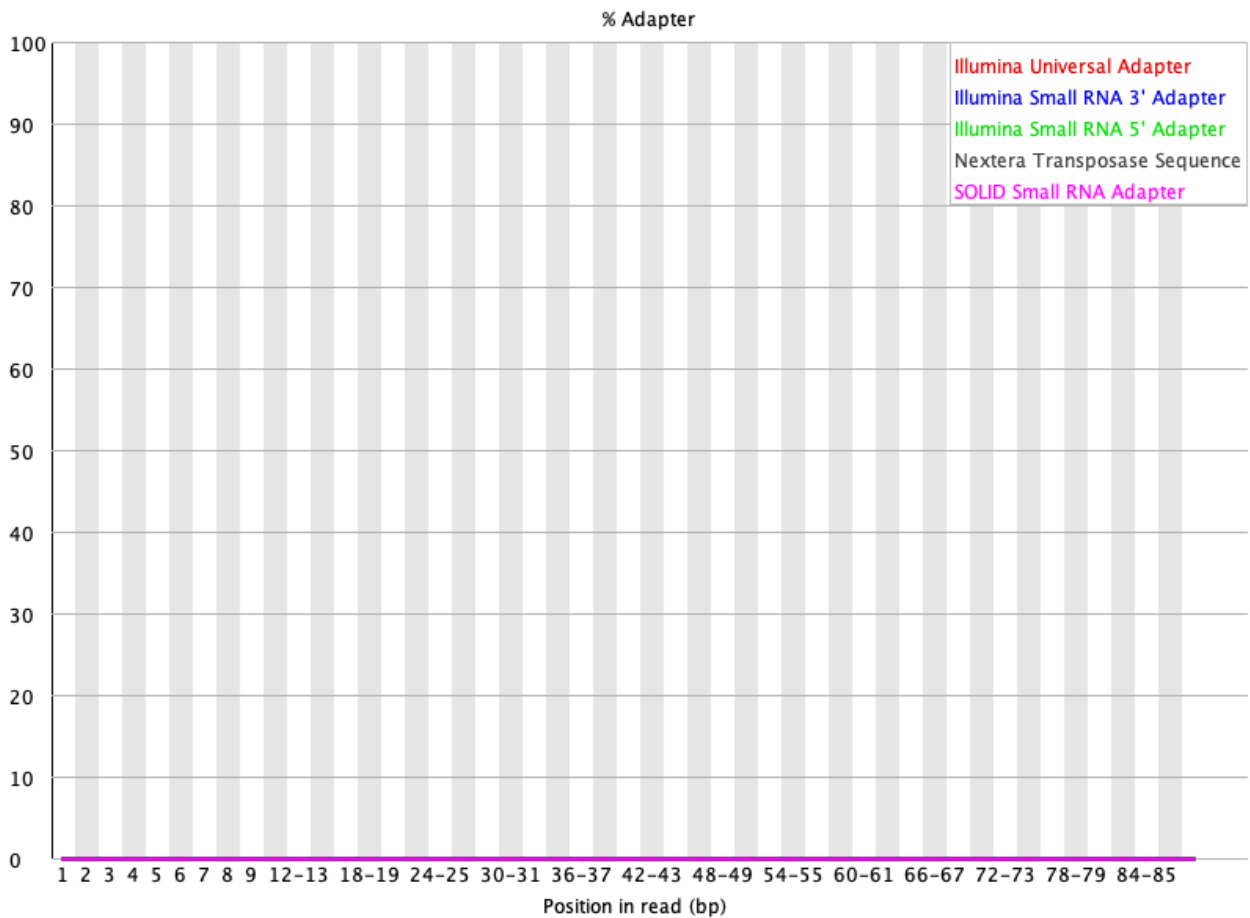






! Overrepresented sequences

Sequence	Count	Percentage	Possible Source
CTTATGTGATAGATGCCTCTTTAAAATATCTAAGTGCTGGGGTTATGAGT	894	0.7968411576479815	No Hit
CCGCTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAA	783	0.697904503846051	No Hit
CTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAATAT	773	0.688991291791823	No Hit
CGCTTTGATATTCTCTGCATCCTATTTAGGGCTATTGATATTTAACAAAT	769	0.6854260069701319	No Hit
CGGCTGTCGAGTTGTACGGCCGTTTCAGCCACGAGTCACGGGGTCTAACGC	757	0.6747301525050583	No Hit
CTGAGACAGAGTCGCTATCGTTATGTCTCCTFCCCGGGTCAAGGCGAAA	649	0.578467462319396	No Hit



{{Edet}}

FASTQC reports for the Golden Snidget

[BORED_1_R1_fastqc.html](#)

[BORED_1_R2_fastqc.html](#)

[BORED_2_R1_fastqc.html](#)

[BORED_2_R2_fastqc.html](#)

[BORED_3_R1_fastqc.html](#)

[BORED_3_R2_fastqc.html](#)

[EXCITED_1_R1_fastqc.html](#)

[EXCITED_1_R2_fastqc.html](#)

[EXCITED_2_R1_fastqc.html](#)

[EXCITED_2_R2_fastqc.html](#)

EXCITED_3_R1_fastqc.html

EXCITED_3_R2_fastqc.html



Lesson 11 Practice

Objectives

In this lesson, we learned to

- merge multiple FASTQC reports into one
- perform data cleanup (quality and adapter trimming) to prepare our sequencing reads for downstream analysis. Here, we will put what we learned to practice.

Merging Golden Snidget FASTQC reports into one

Before getting started, we should change into the `~/biostar_class/snidget/QC` directory where the Golden Snidget sequencing data and FASTQC reports were saved. How do we do this?

{{Sdet}}

Solution{{Esum}}

```
cd ~/biostar_class/snidget/QC
```

{{Edet}}

How do we merge the FASTQC results from the Golden Snidget dataset into one?

{{Sdet}}

Solution{{Esum}}

Since we are in the `snidget` folder, which contains our FASTQC results, we can use `."` to denote "here in this folder" because MultiQC will look for output logs in the specify folder.

```
multiqc --filename multiqc_report_snidget .
```

{{Edet}}

Next copy the MultiQC output to the public directory. Do you remember how to do this?

{{Sdet}}



Solution{{Esum}}

```
cp multiqc_report_snidget.html ~/public/multiqc_report_snidget.html
```

{{Edet}}

Can you configure the Golden Snidget MultiQC output's General Statistics table to show the percentage of modules that failed?

In the General Statistics table of the Golden Snidget MultiQC report, can you assign different colors to distinguish the FASTQ files for the BORED and EXCITED groups?

In the overrepresented sequences plot, how many samples have warnings and how many failed?

{{Sdet}}

Solutions{{Esum}}

Golden Snidget FASTQC files MultiQC results.

Navigation panel

MultiQC
v1.13.dev0

General Stats
FastQC
Sequence Counts
Sequence Quality Histograms
Per Sequence Quality Scores
Per Base Sequence Content
Per Sequence GC Content
Per Base N Content
Sequence Length Distribution
Sequence Duplication Levels
Overrepresented sequences
Adapter Content
Status Checks

MultiQC
A modular tool to aggregate results from bioinformatics analyses across many samples into a single report.
Report generated on 2022-08-25, 15:24 EDT based on data in: /Users/wuz8/OneDrive - National Institutes of Health/biostars_handbook/RNA_Seq/reads

Welcome! Not sure where to start? [Watch a tutorial video](#) (6:06) [don't show again](#) ✕

General Statistics
Copy table [Configure Columns](#) [Plot](#) Showing 12/12 rows and 3/15 columns.

Sample Name	% Dups	% GC	M Seqs
HBR_1_R1	42.7%	50%	0.1
HBR_1_R2	43.6%	50%	0.1
HBR_2_R1	44.3%	50%	0.1

Navigation panel

Tool box

Get help

Choose which
columns to display

General Statistics

Copy table **Configure Columns** Plot Showing 12/12 rows and 5/5 columns.

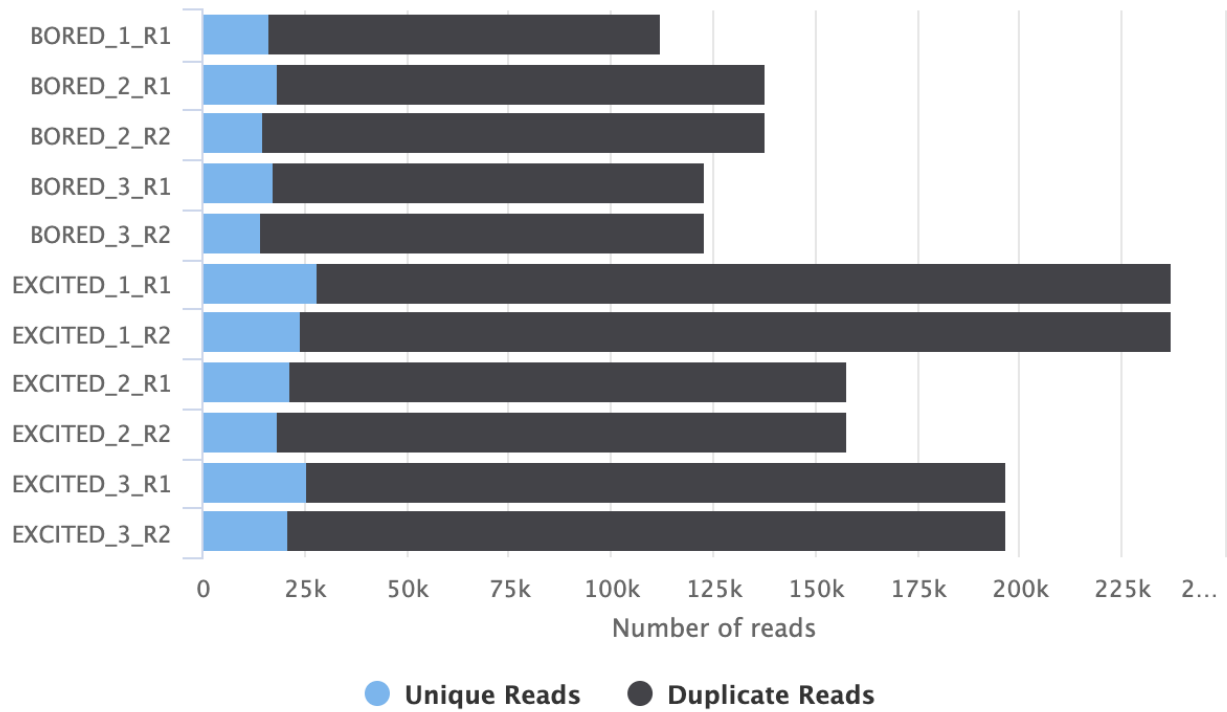
Sample Name	% Dups	% GC	Read Length	% Failed	M Seqs
HBR_1_R1	42.7%	50%	100 bp	9%	0.1
HBR_1_R2	43.6%	50%	100 bp	9%	0.1
HBR_2_R1	44.3%	50%	100 bp	9%	0.1
HBR_2_R2	45.2%	50%	100 bp	0%	0.1
HBR_3_R1	43.3%	50%	100 bp	9%	0.1
HBR_3_R2	44.1%	50%	100 bp	0%	0.1
UHR_1_R1	44.5%	49%	100 bp	9%	0.2
UHR_1_R2	45.3%	49%	100 bp	0%	0.2
UHR_2_R1	46.8%	48%	100 bp	9%	0.2
UHR_2_R2	47.4%	49%	100 bp	9%	0.2
UHR_3_R1	51.5%	49%	100 bp	18%	0.2
UHR_3_R2	52.7%	49%	100 bp	9%	0.2

General Statistics: Columns ×

Uncheck the tick box to hide columns. Click and drag the handle on the left to change order.

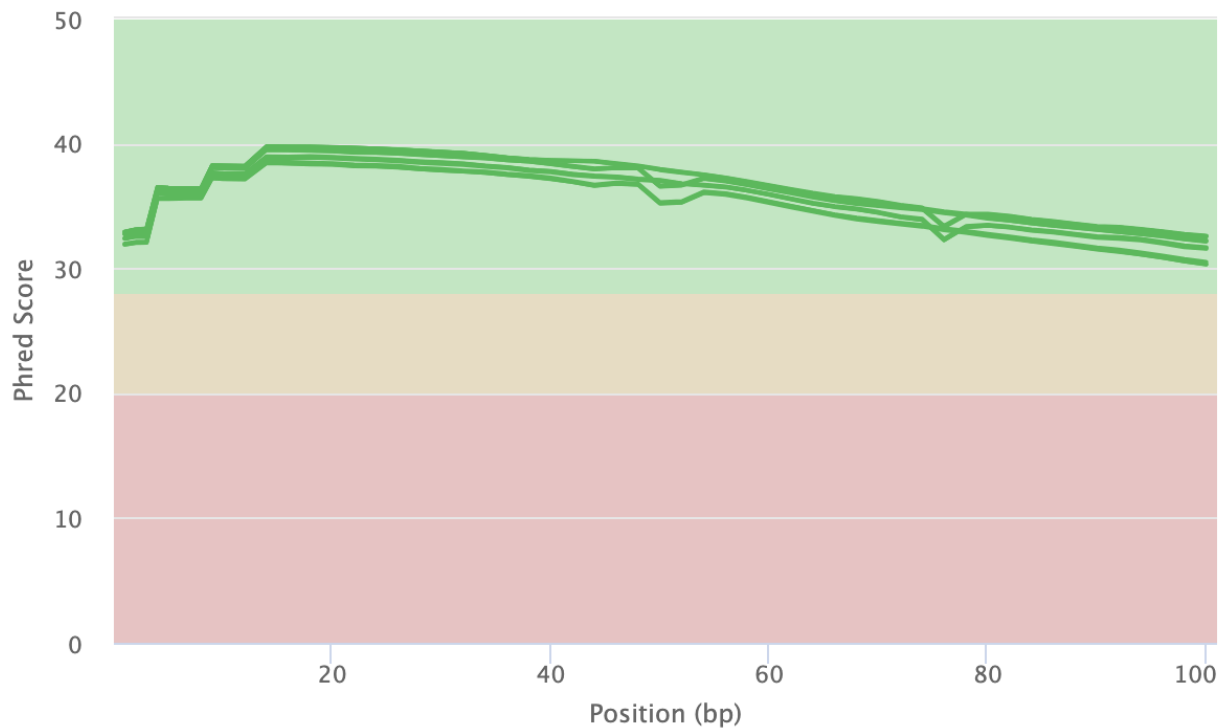
Sort	Visible	Group	Column	Description	ID	Scale
	<input checked="" type="checkbox"/>	FastQC	% Dups	% Duplicate Reads	percent_duplicates	None
	<input checked="" type="checkbox"/>	FastQC	% GC	Average % GC Content	percent_gc	None
	<input checked="" type="checkbox"/>	FastQC	Read Length	Average Read Length (bp)	avg_sequence_length	None
	<input checked="" type="checkbox"/>	FastQC	% Failed	Percentage of modules failed in FastQC report (includes those not plotted here)	percent_fails	None
	<input checked="" type="checkbox"/>	FastQC	M Seqs	Total Sequences (millions)	total_sequences	read_count

FastQC: Sequence Counts



Created with MultiQC

FastQC: Mean Quality Scores

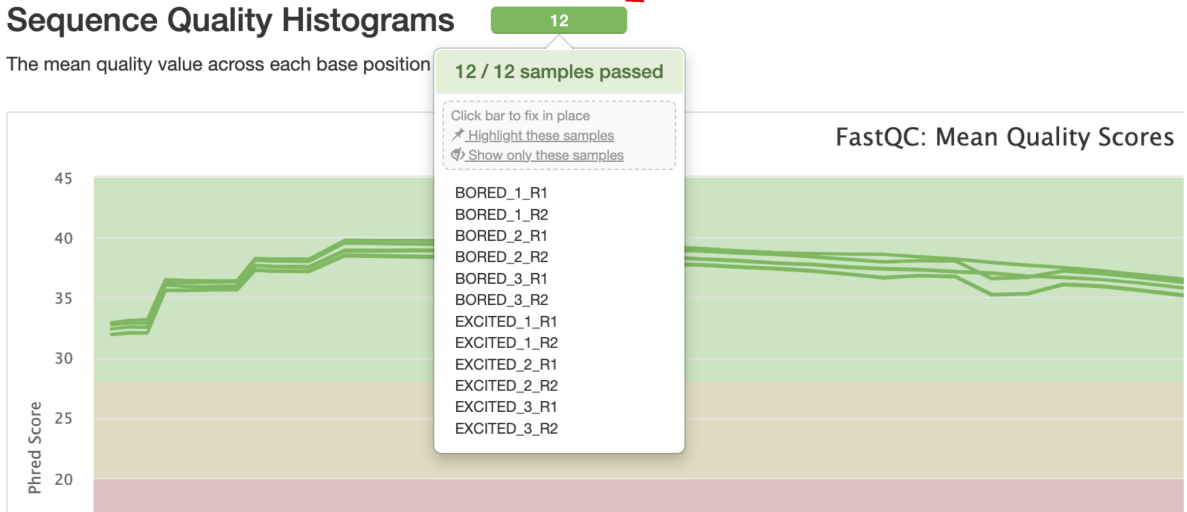


Created with MultiQC

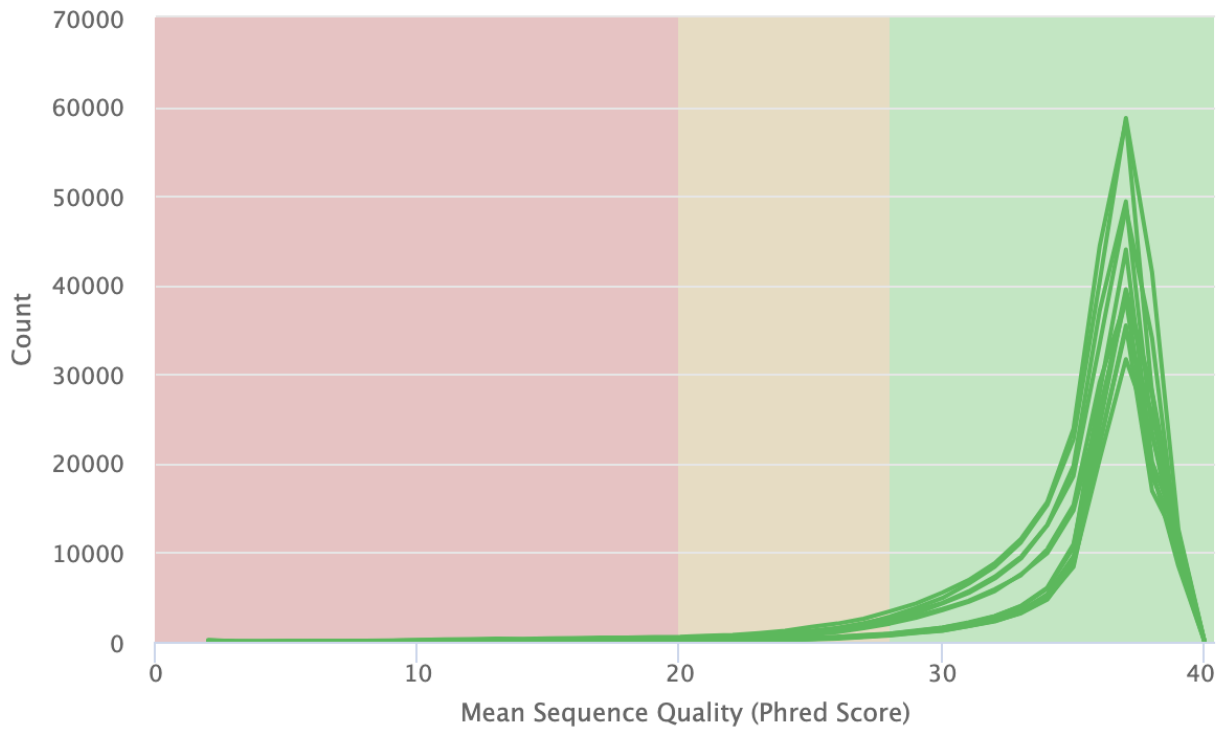
Click here to choose which samples to include in plot

Sequence Quality Histograms

The mean quality value across each base position



FastQC: Per Sequence Quality Scores



Per Base Sequence Content

12

Help

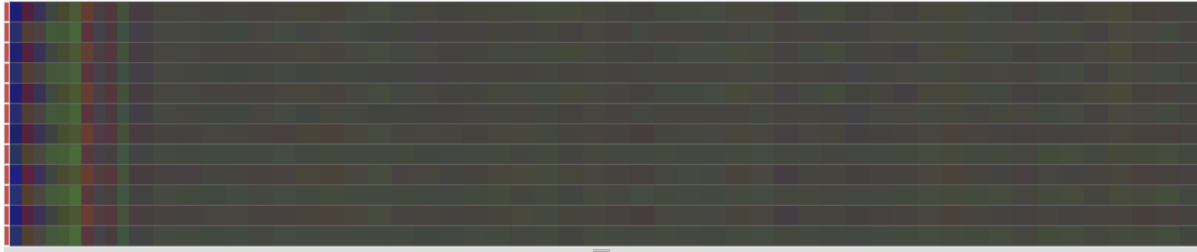
The proportion of each base position for which each of the four normal DNA bases has been called.

Click a sample row to see a line plot for that dataset.

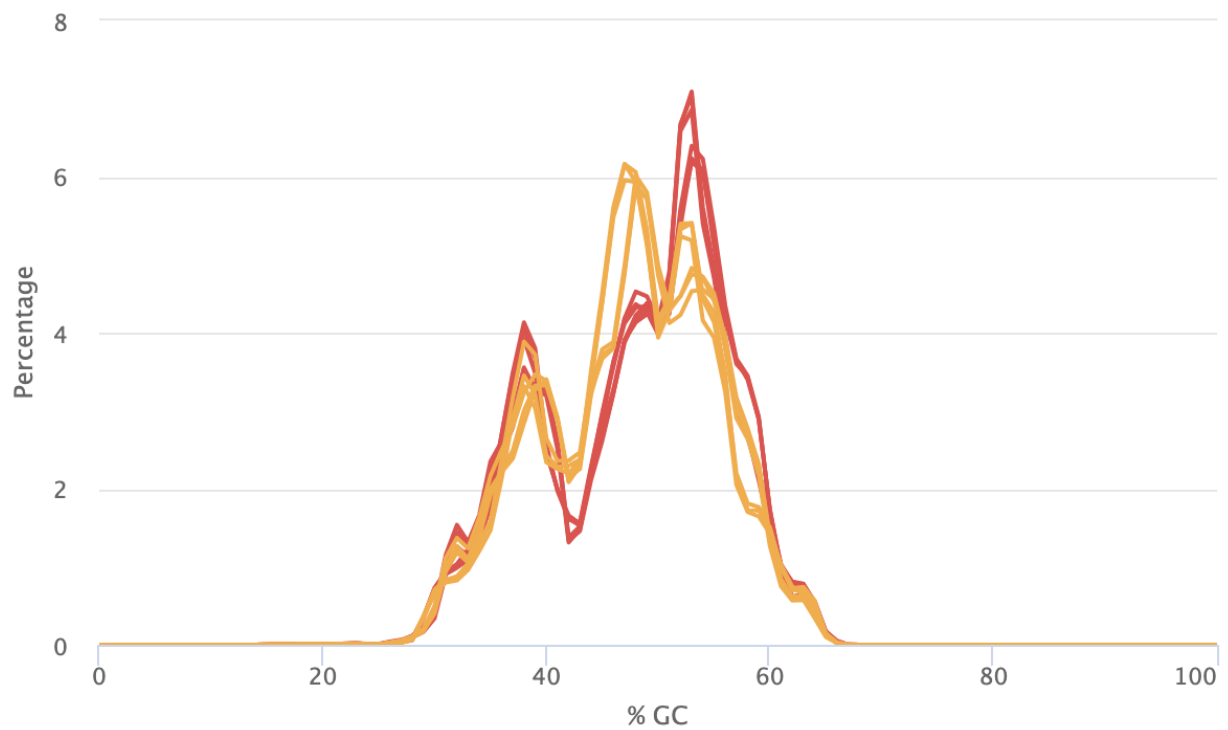
Rollover for sample name

Position: - %T: - %C: - %A: - %G: -

Export Plot

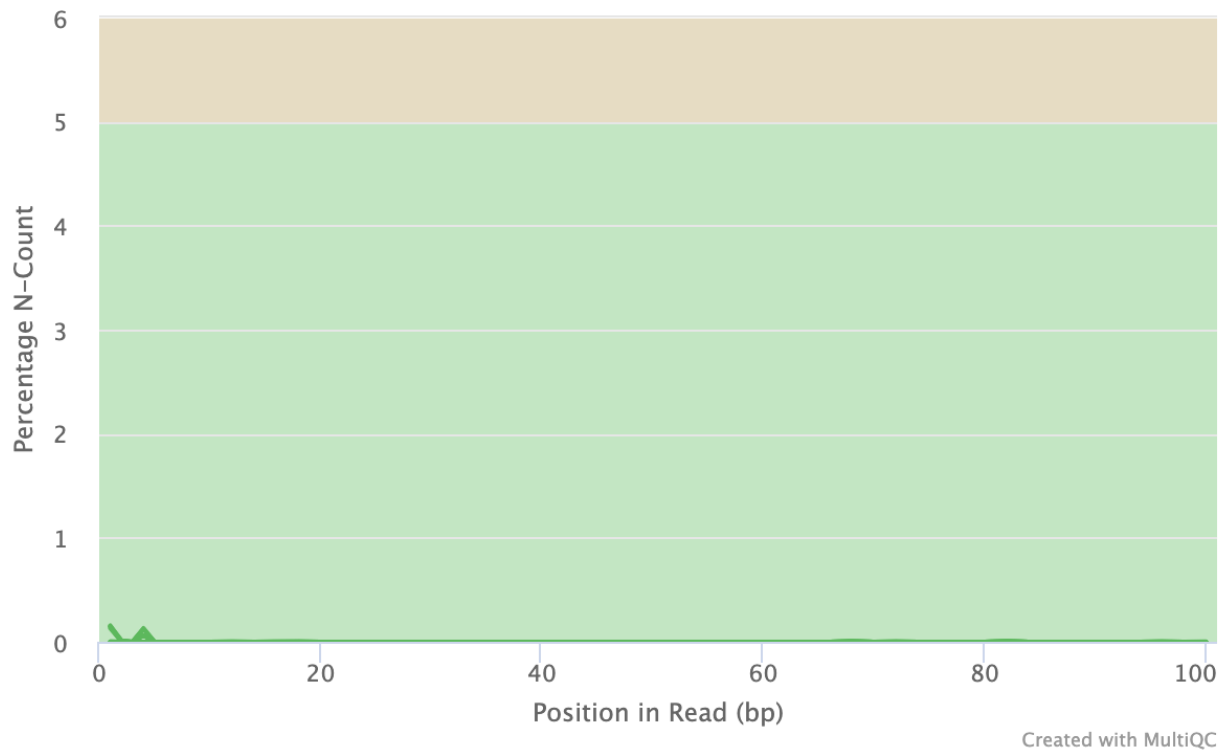


FastQC: Per Sequence GC Content



Created with MultiQC

FastQC: Per Base N Content



Sequence Length Distribution 12

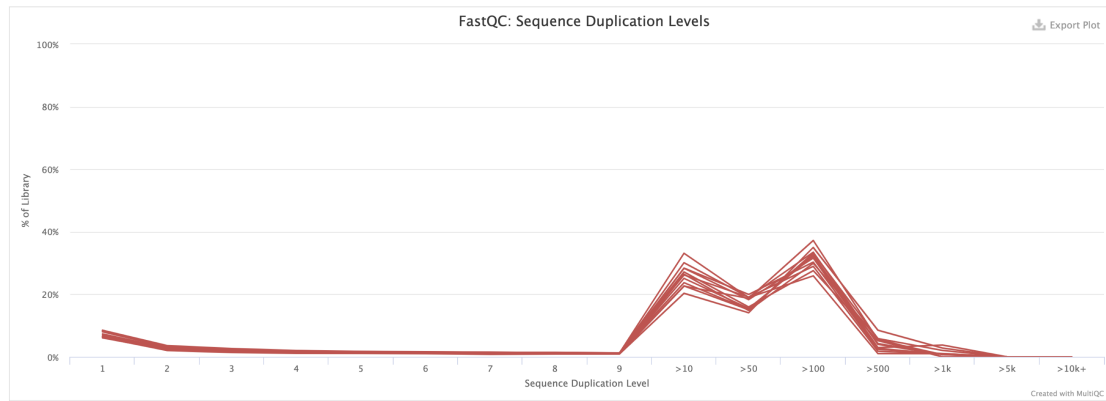
All samples have sequences of a single length (100bp).

Sequence Duplication Levels 12

The relative level of duplication found for every sequence.

[Help](#)

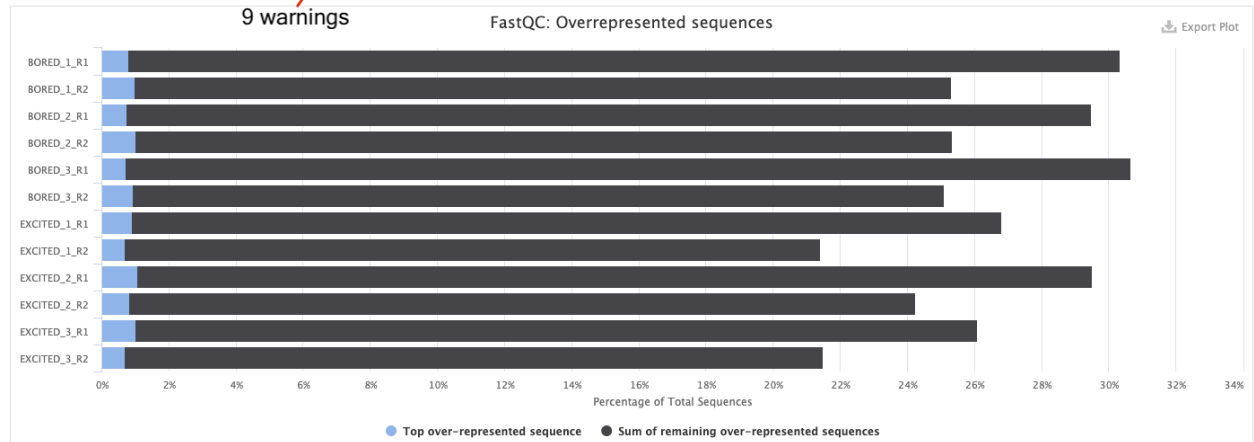
Y-Limits: on



Overrepresented sequences 9 3 ← 3 failed

The total amount of overrepresented sequences found in each library.

Help



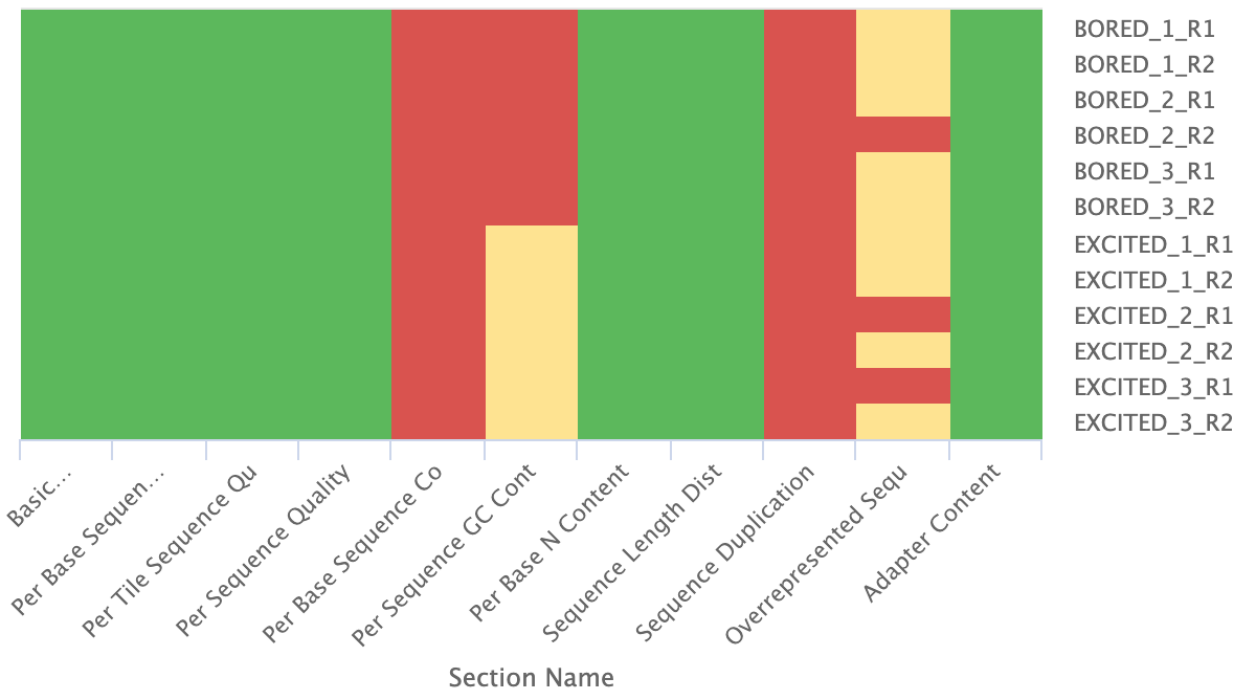
Adapter Content 12

The cumulative percentage count of the proportion of your library which has seen each of the adapter sequences at each position.

Help

No samples found with any adapter contamination > 0.1%

FastQC: Status Checks



- BORED_1_R1
- BORED_1_R2
- BORED_2_R1
- BORED_2_R2
- BORED_3_R1
- BORED_3_R2
- EXCITED_1_R1
- EXCITED_1_R2
- EXCITED_2_R1
- EXCITED_2_R2
- EXCITED_3_R1
- EXCITED_3_R2

Quality and adapter trimming



Let's go back to the `biostar_class` directory and create a folder called `practice_trimming` for this exercise. How do we do this?

{{Sdet}}

Solution{{Esum}}

This depends on where you are currently (ie. your present working directory is). From there go back to the `biostar_class` folder.

```
cd ~/biostar_class
```

```
mkdir practice_trimming
```

{{Edet}}

After the "practice_trimming" directory has been created, change into this directory. How do we do this?

{{Sdet}}

Solution{{Esum}}

```
cd practice_trimming
```

{{Edet}}

Next, download a FASTQ file from NCBI/SRA to practice trimming with.

```
fastq-dump --split-files -X 10000 SRR1553606
```

Once the download is complete the message below will appear.

```
Read 10000 spots for SRR1553606  
Written 10000 spots for SRR1553606
```

How many FASTQ files were downloaded? And from the file names, is this from paired or single end sequencing.

{{Sdet}}

Solution{{Edet}}

```
ls
```

Two FASTQ files were downloaded and this is paired end sequencing.

{{Edet}}

Let's run FASTQC for the these files. Do you recall how to do this?

{{Sdet}}

Solution{{Esum}}

```
fastqc SRR1553606_*.fastq
```

{{Edet}}

Copy the FASTQC outputs (html files) to the public directory.

{{Sdet}}

Solution{{Esum}}

```
cp SRR1553606_*_fastqc.html ~/public
```

{{Edet}}

How is the quality and are there adapter contamination for the FASTQ files in SRR1553606? If yes, can we trim away the adapters and poor quality reads? FYI, for this exercise our adapter sequence is below (can we create an input file called nextera_adapter.fa with the adapter sequence?).

```
>nextera
CTGTCTCTTATACACATCTCCGAGCCCACGAGAC
```



{{Sdet}}

Solution{{Esum}}

The answer is the quality for both FASTQ files is not great and we can remove the poor quality reads and the adapters.

```
nano nextera_adapter.fa
```

Copy and paste the adapter sequence into nano, hit control x and save to exit.

```
>nextera
CTGTCTCTTATACACATCTCCGAGCCCACGAGAC
```

```
trimmomatic PE SRR1553606_1.fastq SRR1553606_2.fastq SRR1553606_trimr
```

{{Edet}}

Run FASTQC on the trimmed output. Any improvements?

{{Sdet}}

Solution{{Esum}}

```
fastqc SRR1553606_trimmed_1.fastq SRR1553606_trimmed_2.fastq
```

{{Edet}}

What is another tool that we can use to perform quality and adapter trimming on FASTQ files?

{{Sdet}}

Solution{{Esum}}

BBDuk


```
bbduk.sh in=SRR1553606_1.fastq in2=SRR1553606_2.fastq out=SRR1553606_
```

{{Edet}}

MultiQC report for Golden Snidget

[Golden Snidget MultiQC report](#)



Lesson 12 Practice

Objectives

In this practice session, we will work with something new, which is a dataset from the Griffith lab RNA sequencing tutorial. Here, we will have a chance to practice what we have learned up until this point of the course series including

- Creating of new directory
- Copying of files
- Assaying sequencing data quality
- Trimming of sequencing data

Where is my data

This data set is from the [Griffith lab RNA sequencing tutorial \(https://rnabio.org\)](https://rnabio.org) and are kept at <http://genomedata.org/rnaseq-tutorial/practical.tar>. This dataset is derived from a study that profiled the transcriptome of HCC1395 breast cancer cell line and the HCC1395BL lymphoblastoid line, which makes this a tumor versus normal transcriptome comparison -- [Griffith lab \(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/\)](https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/).

To begin to work with this dataset, create a directory called hcc1395 in the ~/biostar_class folder and then change into it. See if you can do this.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

Make sure we are in the ~/biostar_class folder

```
pwd
```

If not, then change into it

```
cd ~/biostar_class
```

Create the hcc1395 folder

```
mkdir hcc1395
```

Then change into the hcc1395 folder ?

```
cd hcc1395
```

{{Edet}}

Next, we will need to download the data, which lives at <http://genomedata.org/rnaseq-tutorial/practical.tar>. How do we do this? What is the format (ie. extension) of the downloaded file and how do we unpack it?

{{Sdet}}

Solution{{Esum}}

We can either use wget or curl to download

```
wget http://genomedata.org/rnaseq-tutorial/practical.tar
```

If using curl, remember to specify the output

```
curl http://genomedata.org/rnaseq-tutorial/practical.tar -o practical
```

After downloading, list the directory content in the long view to make sure we successfully downloaded something.

```
ls -l
```

```
total 355132
-rw-rw-r-- 1 joe joe 363653120 Oct 23 2018 practical.tar
```

We have downloaded an archive (tar) of the practice data. We can use the tar command to unpack.

```
tar -xvf practical.tar
```

List the directory again to see what was added.

```
ls -l
```

We have the fastq files in fastq.gz format (ie. they are gzipped but we are will not be unzipping these as our tools can work with the gzipped versions).

```
total 710268
-rw-rw-r-- 1 joe joe 25955505 Mar 18 2017 hcc1395_normal_rep1_r1.fastq.gz
-rw-rw-r-- 1 joe joe 30766759 Mar 18 2017 hcc1395_normal_rep1_r2.fastq.gz
-rw-rw-r-- 1 joe joe 25409781 Mar 18 2017 hcc1395_normal_rep2_r1.fastq.gz
-rw-rw-r-- 1 joe joe 30213083 Mar 18 2017 hcc1395_normal_rep2_r2.fastq.gz
-rw-rw-r-- 1 joe joe 25132378 Mar 18 2017 hcc1395_normal_rep3_r1.fastq.gz
-rw-rw-r-- 1 joe joe 30174637 Mar 18 2017 hcc1395_normal_rep3_r2.fastq.gz
-rw-rw-r-- 1 joe joe 30361801 Mar 18 2017 hcc1395_tumor_rep1_r1.fastq.gz
-rw-rw-r-- 1 joe joe 35887220 Mar 18 2017 hcc1395_tumor_rep1_r2.fastq.gz
-rw-rw-r-- 1 joe joe 29769613 Mar 18 2017 hcc1395_tumor_rep2_r1.fastq.gz
-rw-rw-r-- 1 joe joe 35254974 Mar 18 2017 hcc1395_tumor_rep2_r2.fastq.gz
-rw-rw-r-- 1 joe joe 29472281 Mar 18 2017 hcc1395_tumor_rep3_r1.fastq.gz
-rw-rw-r-- 1 joe joe 35241854 Mar 18 2017 hcc1395_tumor_rep3_r2.fastq.gz
-rw-rw-r-- 1 joe joe 363653120 Oct 23 2018 practical.tar
```

{{Edet}}

Getting to know the data

Can you print the first sequencing read (from header line to quality score line) in hcc1395_normal_rep1_r1.fastq.gz?

{{Sdet}}

Solution{{Esum}}

```
zcat hcc1395_normal_rep1_r1.fastq.gz | head -4
```

{{Edet}}

How many sequencing reads are in hcc1395_normal_rep1_r1.fastq.gz?

{{Sdet}}

Solution{{Esum}}

```
zcat hcc1395_normal_rep1_r1.fastq.gz | grep @ | wc -l
```

```
331958
```

{{Edet}}



Assessing sequencing data quality

For this portion of the practice, create a folder called **qc** within `~/biostar_class/hcc1395` (which should be the present working directory) to store the FASTQC outputs.

{{Sdet}}

Solution{{Esum}}

```
mkdir qc
```

```
cd qc
```

{{Edet}}

Can you generate quality reports for the FASTQ files in this dataset?

{{Sdet}}

Solution{{Esum}}

We use `-o` to specify output directory in the FASTQC command below. Since we want the FASTQC output to be written in the present working directory (which is `~/biostar_class/hcc1395/qc`) we can just use `."` to denote this ("`."` means here in the current directory).

```
fastqc -o . ../hcc1395_normal*.fastq.gz ../hcc1395_tumor*.fastq.gz
```

{{Edet}}

How do you view the FASTQC reports?

{{Sdet}}

Solution{{Esum}}

Copy them to the `~/public` directory

```
cp *.html ~/public
```

{{Edet}}

Examine the FASTQC reports. How is the quality of the sequencing data? Are there any adapter contaminations?

```
{{Sdet}}
```



```
Solution{{Esum}}
```

While some of the sequence quality scores were not great especially in the read 2 files, we definitely need to trim for adapters.

```
{{Edet}}
```

Can you merge the FASTQC files for this dataset into an MultiQC report?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
multiqc --filename multiqc_report_hcc1395 .
```

```
{{Edet}}
```

Trimming

For this exercise, go back to the `~/biostar_class/hcc1395` folder and create a new directory called `trimmed_data`.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cd ~/biostar_class/hcc1395
```

```
mkdir trimmed_data
```

```
cd trimmed_data
```

```
{{Edet}}
```

The adapters for this dataset can be obtained at http://genomedata.org/rnaseq-tutorial/illumina_multiplex.fa so first download these.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
wget http://genomedata.org/rnaseq-tutorial/illumina_multiplex.fa
```



or

```
curl http://genomedata.org/rnaseq-tutorial/illumina_multiplex.fa -o .
```

{{Edet}}

Let's trim using the FASTQ files for hcc1395_normal_rep1 using the following thresholds:

- Input files are
 - hcc1395_normal_rep1_r1.fastq.gz
 - hcc1395_normal_rep1_r2.fastq.gz
- Quality - use SLIDINGWINDOW of 4 reads and a quality threshold of 30
- Adapter - use ILLUMINACLIP:illumina_multiplex.fa:2:30:5
- Minimum sequence length to keep - MINLEN:50

{{Sdet}}

Solution{{Esum}}

```
trimmomatic PE ../hcc1395_normal_rep1_r1.fastq.gz ../hcc1395_normal_r
```

{{Edet}}

QC reports for hcc1395 dataset

Pre-trimming

FASTQC report for hcc1395_normal_rep1_r1.fastq.gz

FASTQC report for hcc1395_normal_rep1_r2.fastq.gz

FASTQC report for hcc1395_normal_rep2_r1.fastq.gz

FASTQC report for hcc1395_normal_rep2_r2.fastq.gz

FASTQC report for hcc1395_normal_rep3_r1.fastq.gz

FASTQC report for hcc1395_normal_rep3_r2.fastq.gz

FASTQC report for hcc1395_tumor_rep1_r1.fastq.gz

FASTQC report for hcc1395_tumor_rep1_r2.fastq.gz

FASTQC report for hcc1395_tumor_rep2_r1.fastq.gz

FASTQC report for hcc1395_tumor_rep2_r2.fastq.gz

FASTQC report for hcc1395_tumor_rep3_r1.fastq.gz

FASTQC report for hcc1395_tumor_rep3_r2.fastq.gz

MultiQC report for hcc1395

Post-trimming

FASTQC report for hcc1395_normal_rep1_trimmed_r1.fastq.gz

FASTQC report for hcc1395_normal_rep1_trimmed_r2.fastq.gz

Lesson 13 Practice

Objectives

In this lesson we learned how to align raw sequencing reads to reference and to process alignment results for downstream analysis. Here, we will test our knowledge by continuing with the Golden Snidget dataset.

Review of what we have done so far for the Golden Snidget dataset

Before we start, remember to change into the `~/biostar_class/snidget` folder and then list the content to review what is available.

```
cd ~/biostar_class/snidget
```

```
ls -l QC
```

```
total 117384
drwxrwxr-x 1 joe joe      1188 Oct 31 23:29 QC
-rw-r--r-- 1 joe joe    57462 Oct 27 00:30 golden.genome.tar.gz
-rw-r--r-- 1 joe joe 120138017 Oct 27 00:30 golden.reads.tar.gz
drwxr-xr-x 1 joe joe      336 Oct 27 00:30 reads
drwxr-xr-x 1 joe joe       70 Oct 27 00:30 refs
```

In the `~/biostar_class/snidget` folder, we have the tar.gz files for the Golden Snidget reference genome and the sequencing data. These were unpacked previously to give the refs and reads folder. The QC folder contains our FASTQC and MultiQC reports for the unaligned sequencing data.

```
ls -l
```

```
BORED_1_R1_fastqc.html
BORED_1_R1_fastqc.zip
BORED_1_R2_fastqc.html
BORED_1_R2_fastqc.zip
```

```
BORED_2_R1_fastqc.html
BORED_2_R1_fastqc.zip
BORED_2_R2_fastqc.html
BORED_2_R2_fastqc.zip
BORED_3_R1_fastqc.html
BORED_3_R1_fastqc.zip
BORED_3_R2_fastqc.html
BORED_3_R2_fastqc.zip
EXCITED_1_R1_fastqc.html
EXCITED_1_R1_fastqc.zip
EXCITED_1_R2_fastqc.html
EXCITED_1_R2_fastqc.zip
EXCITED_2_R1_fastqc.html
EXCITED_2_R1_fastqc.zip
EXCITED_2_R2_fastqc.html
EXCITED_2_R2_fastqc.zip
EXCITED_3_R1_fastqc.html
EXCITED_3_R1_fastqc.zip
EXCITED_3_R2_fastqc.html
EXCITED_3_R2_fastqc.zip
multiqc_report_snidget.html
multiqc_report_snidget_data
```

Aligning Golden Snidget sequencing - constructing genome index

Here, we are going to align the Golden Snidget sequencing files to its genome. Recall that we are working with RNA sequencing data. Given HISAT2 and Bowtie2 as the options for aligners, which is more ideal? One of the aligners is splice aware the other is not.

{{Sdet}}

Solution{{Esum}}

We should use HISAT2 (the splice aware aligner)

{{Edet}}

For today, because we are working with alignment, let's create a folder called `snidget_hisat2` to store the results for the HISAT2 alignment. Take a moment to create this folder.

{{Sdet}}

Solution{{Esum}}



```
mkdir snidget_hisat2
```

{{Edet}}

Next, we need to change into the refs directory. How do we do this?

{{Sdet}}

Solution{{Esum}}

```
cd refs
```

{{Edet}}

Besides the raw sequencing files and reference genome, what other information do we need to complete the alignment (hint: it has something to do with the reference genome) and how do we do this?

{{Sdet}}

Solution{{Esum}}

We need to index the reference genome

```
hisat2-build genome.fa genome
```

{{Edet}}

Constructing a text file with the Golden Snidget sample IDs

To help us align all of the FASTQ files in one go, we should create in the reads directory a file with the sample IDs names for the Golden Snidget.

First, change into the ~/biostar_class/snidget/reads directory.

```
cd ~/biostar_class/snidget/reads
```



Then, how do we use the parallel command to construct the text file with the Golden Snidget sample IDs? Also, save this file as ids.txt.

```
{{Sdet}}
```

```
Solutions{{Esum}}
```

```
parallel echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids.txt
```

Listing the contents of the reads directory, we see that we have the file ids.txt.

```
ls
```

```
BORED_1_R1.fq  BORED_2_R1.fq  BORED_3_R1.fq  EXCITED_1_R1.fq  EXCITEI  
BORED_1_R2.fq  BORED_2_R2.fq  BORED_3_R2.fq  EXCITED_1_R2.fq  EXCITEI
```

Now, if we print out the contents of ids.txt using cat, then we see the sample IDs for the Golden Snidget dataset.

```
cat ids.txt
```

```
BORED_1  
BORED_2  
BORED_3  
EXCITED_1  
EXCITED_2  
EXCITED_3
```

```
{{Edet}}
```

Alignment of Golden Snidget FASTQ files - constructing the HISAT2 command

Now that our HISAT2 indices have been built and the text file with the sample IDs has been generated, we can do the actual alignment.

First, change into the ~/biostar_class/snidget/snidget_hisat2 folder

```
cd ~/biostar_class/snidget/snidget_hisat2
```



We are going to align all off the FASTQ files for the Golden Snidget in one go. Also, remember as we construct our command to align the FASTQ files that

- our reference genome is in the refs folder
- the FASTQ files are in the reads folder
- the text file (ids.txt) containing the sample IDs are in the reads folder

Remember to save the alignment status as a text file so we can view later (save this as sample-name_hisat2_summary.txt).

{{Sdet}}

Solution{{Esum}}

```
cat ../reads/ids.txt | parallel "hisat2 -x ../refs/genome -1 ../reads/
```

{{Edet}}

Now that the alignment is done, what are the overall alignment rates? Were there a lot of discordant alignments?

{{Sdet}}

Solution{{Esum}}

Overall alignment rate for each sample is 100%. For the most part the reads aligned concordantly.

{{Edet}}

Can we include the HISAT2 alignment statistics in a MultiQC report? Hint, change back into the ~/biostar_class/snidget directory and save this MultiQC report to the QC folder.

```
cd ~/biostar_class/snidget
```

{{Sdet}}

Solution{{Esum}}

Yes

```
multiqc --filename QC/multiqc_report_snidget_post_alignment .
```



Now listing, the contents of the QC folder we will see the MultiQC report that includes the post alignment statistics (multiqc_report_snidget_post_alignment.html).

```
ls QC
```

```
BORED_1_R1_fastqc.html  BORED_2_R1_fastqc.zip  BORED_3_R2_fastqc.htmr
BORED_1_R1_fastqc.zip  BORED_2_R2_fastqc.html  BORED_3_R2_fastqc.zip
BORED_1_R2_fastqc.html  BORED_2_R2_fastqc.zip  EXCITED_1_R1_fastqc.l
BORED_1_R2_fastqc.zip  BORED_3_R1_fastqc.html  EXCITED_1_R1_fastqc.:
BORED_2_R1_fastqc.html  BORED_3_R1_fastqc.zip  EXCITED_1_R2_fastqc.l
```

Copy multiqc_report_snidget_post_alignment.html to the ~/public directory to take a look.

```
cp QC/multiqc_report_snidget_post_alignment.html ~/public
```

{{Edet}}

Alignment of Golden Snidget FASTQ files - converting SAM files to BAM

Recall in the lesson that SAM files are human readable so we will need to convert these to sorted and indexed BAM files, which are machine readable for downstream steps in our analysis. How do we do this? Remember as we are working, we need to go back to the ~/biostar_class/snidget/snidget_hisat2 directory so change into this before starting this next exercise.

```
cd ~/biostar_class/snidget/snidget_hisat2
```

{{Sdet}}

Solution{{Esum}}

First, we sort the SAM file and convert it to BAM.

```
cat ../reads/ids.txt | parallel "samtools sort -o {}.bam {}.sam"
```

Second, we index the BAM file

```
cat ../reads/ids.txt | parallel "samtools index -b {}.bam {}.bam.bai"
```

{{Edet}}

Lesson 14 Practice

Objectives

Here, we will practice using the Integrative Genome Viewer (IGV) to visualize the hcc1395 RNA sequencing alignment results.

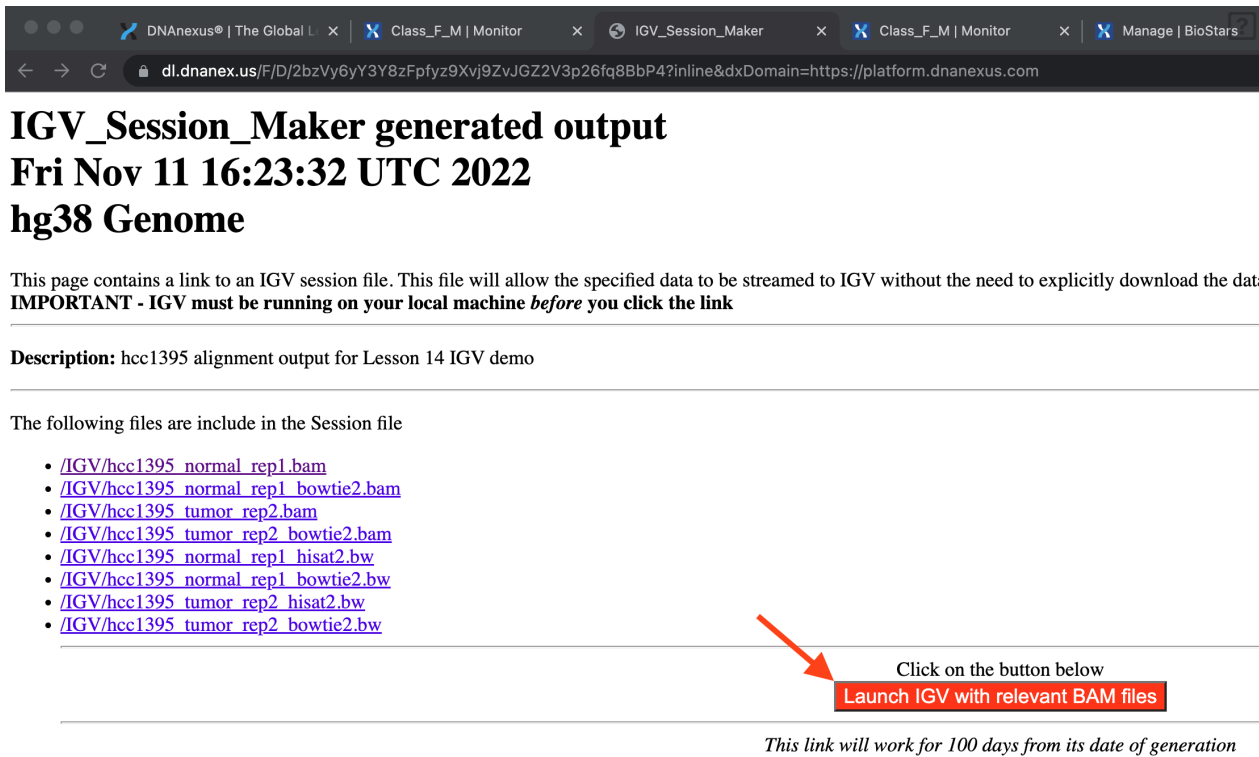
About the data and launching IGV

We were introduced to the hcc1395 RNA sequencing data in [Lesson 12 practice session \(https://btep.ccr.cancer.gov/docs/b4b/Module2_RNA_Sequencing/Lesson12_practice/#where-is-my-data\)](https://btep.ccr.cancer.gov/docs/b4b/Module2_RNA_Sequencing/Lesson12_practice/#where-is-my-data). This study compared the transcriptome of hcc1395 normal and cancer cell lines so it's a normal versus tumor comparison. This dataset was obtained from [rnabio \(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/\)](https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/) and was subsetted to reads that map to human chromosome 22. In Lesson 12 practice, we did QC for and trimming of the FASTQ files for this dataset. Even though we have not gone over the alignment step, the alignment output has been pre-generated for this IGV exercise. See Figure 1 and Figure 2 on how to view the hcc1395 alignment output in IGV.

The screenshot shows the DNAnexus web interface. The browser address bar displays the URL: `platform.dnanexus.com/panx/projects/GGV99Q009vXfJ2312qZBYpxg/data/`. The interface includes a navigation menu with 'PROJECTS', 'TOOLS', and 'HELP'. The main content area is titled 'BioStars' and has tabs for 'SETTINGS', 'MANAGE', 'MONITOR' (with a '1' notification), and 'VISUALIZE'. A left sidebar shows a tree view with 'BioStars' selected, containing sub-items: 'Applications', 'IGV', 'Params', 'Sessions', and 'Test'. The main panel shows a file list for 'All Projects > BioStars'. At the top of the list are filters: 'Current Folder Only', 'Any Name', 'Any ID', and 'Any Type'. The file list has columns for 'Name' and 'Type / Class'. The file 'hcc1395_igv.html' is circled in red.

<input type="checkbox"/>	Name ↕	Type / Class
<input type="checkbox"/>	BAMS.xml	File
<input type="checkbox"/>	Class_A_E.html	File
<input type="checkbox"/>	Class_F_M.html	File
<input type="checkbox"/>	Class_N_S.html	File
<input type="checkbox"/>	Class_T_Z.html	File
<input type="checkbox"/>	hcc1395_igv.html	File
<input type="checkbox"/>	hcc1395_igv.xml	File

Figure 1: Click on the `hcc1395_igv.html` under All Projects -> BioStars to access the IGV launcher for the `hcc1395` dataset.



IGV_Session_Maker generated output
Fri Nov 11 16:23:32 UTC 2022
hg38 Genome

This page contains a link to an IGV session file. This file will allow the specified data to be streamed to IGV without the need to explicitly download the data. **IMPORTANT - IGV must be running on your local machine before you click the link**

Description: hcc1395 alignment output for Lesson 14 IGV demo

The following files are include in the Session file

- [/IGV/hcc1395_normal_rep1.bam](#)
- [/IGV/hcc1395_normal_rep1_bowtie2.bam](#)
- [/IGV/hcc1395_tumor_rep2.bam](#)
- [/IGV/hcc1395_tumor_rep2_bowtie2.bam](#)
- [/IGV/hcc1395_normal_rep1_hisat2.bw](#)
- [/IGV/hcc1395_normal_rep1_bowtie2.bw](#)
- [/IGV/hcc1395_tumor_rep2_hisat2.bw](#)
- [/IGV/hcc1395_tumor_rep2_bowtie2.bw](#)

Click on the button below
Launch IGV with relevant BAM files

This link will work for 100 days from its date of generation

Figure 2: Click the launch button to view the alignments for samples hcc1395_normal_rep1 and hcc1395_tumor_rep2.

In IGV, we are visualizing the alignments for samples hcc1395_normal_rep1 and hcc1395_tumor_rep2 to get a glimpse of gene expression differences in normal versus tumor. We also want to compare the alignment results derived from HISAT2 (splice aware) and Bowtie2 (splice unaware). The Bowtie2 BAM filenames have "_bowtie2" appended.

Testing your knowledge

Which reference genome are we using in this IGV session to view the alignment results for samples hcc1395_normal_rep1 and hcc1395_tumor_rep2?

On what chromosome are the sequencing data mapping to?

{{Sdet}}

Solution{{Esum}}

We are using human hg38 and the reads map to chromosome 22



{{Edet}}

The peaks in the bigWig (bw) tracks, what do these represent?

{{Sdet}}

Solution{{Esum}}

The peaks in the bigWig (bw) tracks represents sequencing coverage

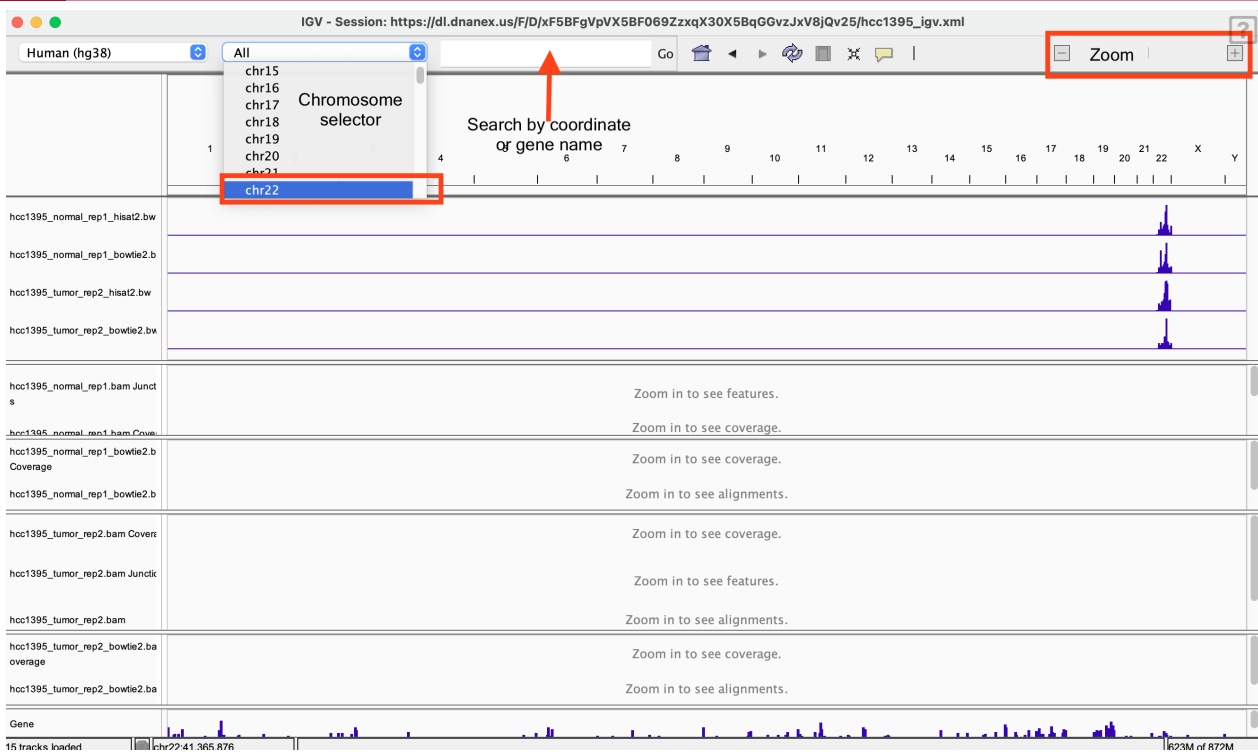
{{Edet}}

Upon opening of IGV, the bam tracks are empty, how do we zoom in to start viewing information?

{{Sdet}}

Solution{{Esum}}

We can either select by chromosome, search by gene name or coordinates, or use the zoom feature



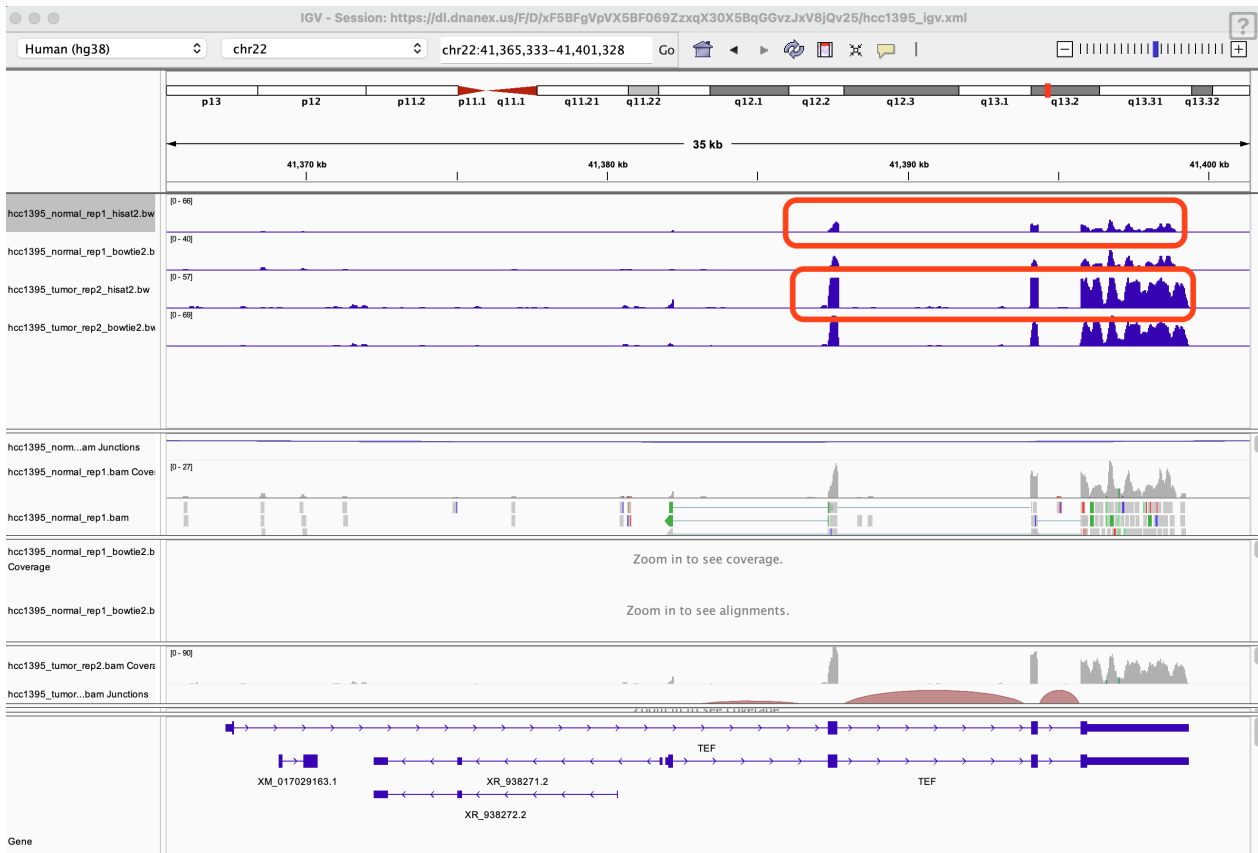
{{Edet}}

Search for the TEF gene, does there appear to be difference in gene expression between the normal and tumor sample?

{{Sdet}}

Solution{{Esum}}

It appears that the tumor sample is expressing more TEF than normal



{{Edet}}

You might have to remove the hcc1395_tumor_rep2 tracks for this but what is the difference between the HISAT2 and Bowtie2 alignment for the hcc1395_normal_rep1 sample?

{{Sdet}}

Solution{{Esum}}

The HISAT2 alignment have the extra splice junction track and the reads that map across exons are connected by lines



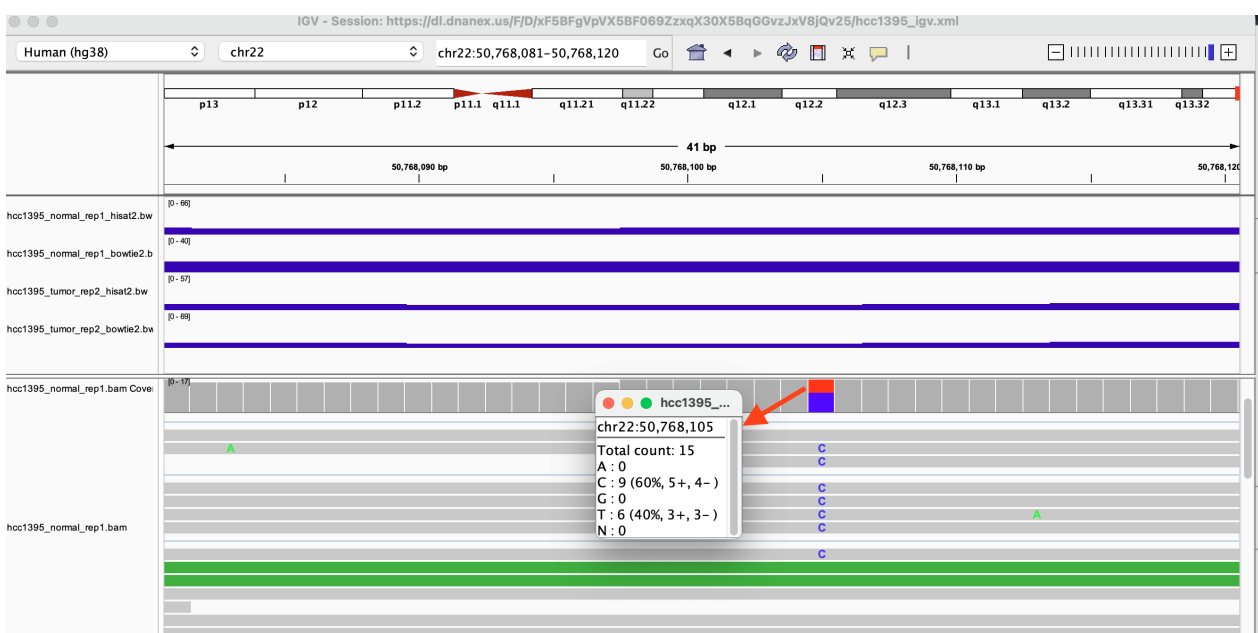
{{Edet}}

In position 50,768,105 on chromosome 22, what is the blue-red color in the hcc1395_normal_rep1 BAM coverage track telling us?

{{Sdet}}

Solution{{Esum}}

It's telling us that there could be a potential single nucleotide variant as we observed C's in addition to T's at this position. There is a T in the reference.



`{{Edet}}`

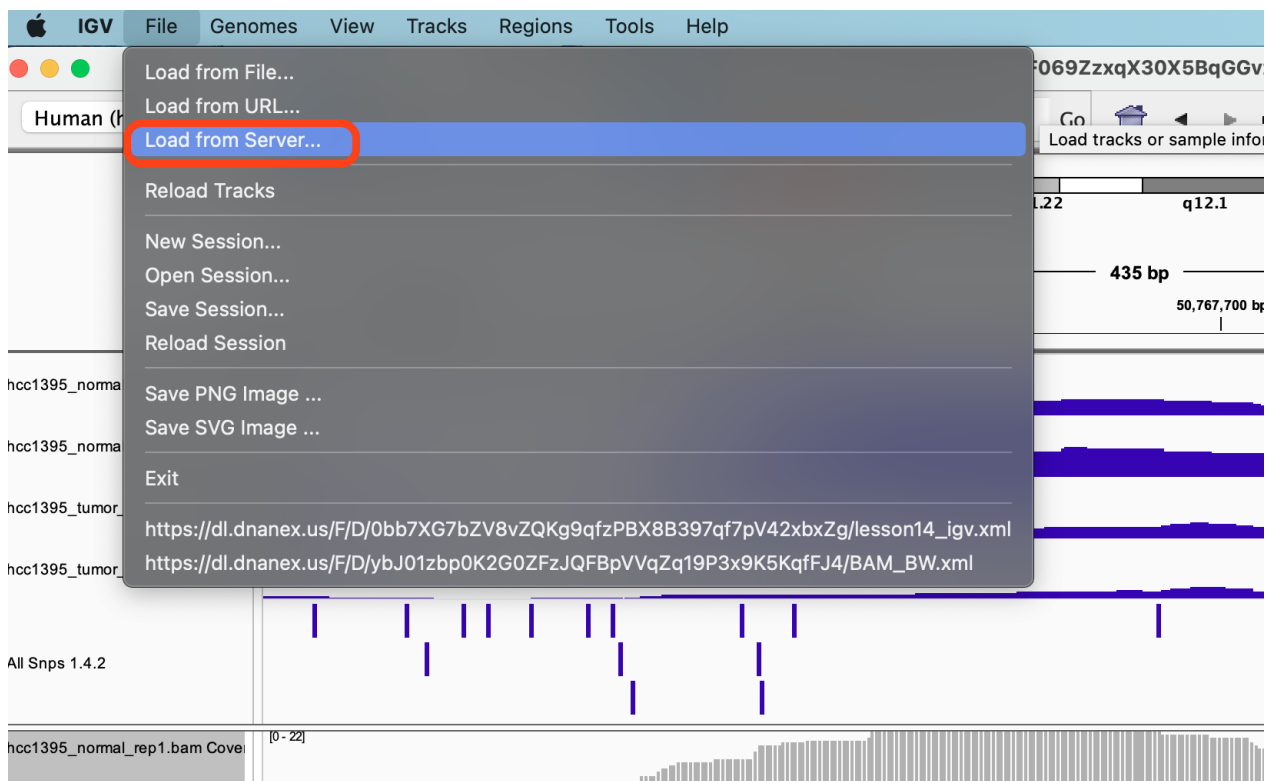
Bonus question

Work on this if you choose to, at your own leisure, or if time permits.

How would you confirm that there is a potential single nucleotide variant at position 50,768,105 on chromosome 22 for the hcc1395_normal_rep1 sample?

`{{Sdet}}`Solution`{{Esum}}`

Goto File and select Load from server



In the Available Datasets box, choose Annotations -> Variation and Repeats -> All Snps

The screenshot shows the IGV interface with the 'Available Datasets' dialog box open. The dialog box has the following structure:

- Available Datasets
 - Annotations
 - Genes
 - Variation and Repeats
 - All Snps 1.4.2 ⓘ
 - Common Snps 1.4.2 ⓘ
 - Repeat Masker ⓘ
 - CpG Islands ⓘ
 - GC %
 - Phastcons (20 way) ⓘ

The background interface shows the IGV window with the following elements:

- Menu bar: IGV, File, Genomes, View, Tracks, Regions, Tools, Help
- Session URL: https://dl.dnanex.us/F/D/xF5BFgVpVX5BF069ZzqxX30X5BqC
- Species: Human (hg38)
- Chromosome: chr22
- Region: chr22:50,767,463-50,767,896
- Genomic track: p13, p12, p11.2, p11.1, q11.1, q11.21, q11.22, q12.1
- Track list: hcc1395_normal_rep1_hisat2, hcc1395_normal_rep1_bowtie, hcc1395_tumor_rep2_hisat2, hcc1395_tumor_rep2_bowtie, All Snps 1.4.2, hcc1395_normal_rep1.bam C, hcc1395_normal_rep1.bam, hcc1395_normal_rep1_bowtie, Coverage

We see that there is indeed a potential SNP here and if we click on it we will see more information about the potential SNP.

{{Edet}}



Lesson 15 Practice

Objectives

Previously, we performed QC on the Golden Snidget RNA sequencing data, aligned the sequencing reads to its genome, and obtained expression counts. We can now finally perform differential expression analysis, to find out which genes are differentially expressed between the EXCITED and BORED states of the Golden Snidget.

Scripts used for differential analysis

Recall that the scripts used for differential expression analysis are in the folder `/usr/local/code`. We can also access this folder using the environmental variable `CODE`. How many scripts are in this folder (find out by not using the full path to the folder containing the scripts).

{{Sdet}}

Solution{{Esum}}

```
ls -l $CODE | wc -l
```

{{Edet}}

Make sure we change into `~/biostar_class/snidget` before starting.

```
cd ~/biostar_class/snidget
```

Create a folder to store the Golden Snidget differential expression analysis results

First, create a folder to store the Golden Snidget differential expression analysis results. Name this folder `snidget_deg`.

{{Sdet}}

Solution{{Esum}}

```
mkdir snidget_deg
```



```
{{Edet}}
```

Create the design file

Create the design.csv file using the nano editor. Recall that the design files contain nothing more than a column with sample names and a column informing of sample treatment condition. Some of the R helper scripts require a csv version of this, where the columns are separated by comma. Here, we create both before moving on.

Creating design.csv

```
nano design.csv
```

Copy the text below to the nano editor, hit control-x and save to return to the terminal.

```
sample,condition
BORED_1.bam,BORED
BORED_2.bam,BORED
BORED_3.bam,BORED
EXCITED_1.bam,EXCITED
EXCITED_2.bam,EXCITED
EXCITED_3.bam,EXCITED
```

Obtaining the counts table

Change into `~/biostar_class/snidget/snidget_hisat2/` when running `featureCounts` to obtain the expression counts table. The annotation file for this dataset is in `~/biostar_class/snidget/refs` and is named `features.gff`. How would we construct `featureCounts` to obtain an expression counts table for the Golden Snidget?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cd ~/biostar_class/snidget/snidget_hisat2/
```

```
featureCounts -p -a ../refs/features.gff -g gene_name -o ../snidget_?
```

{{Edet}}

Format the Golden Snidget counts table for differential expression analysis

Remember that the `deseq2.r` script requires that the expression counts table be in csv format. It is currently in tab delimited format as generated by `featureCounts`. There is also a header line that we need to get rid of in the counts table. Finally, recall that our expression counts table is stored as `counts.txt` in the `~/biostar_class/snidget/snidget_deg` directory, so change into this before moving forward.

```
cd ~/biostar_class/snidget/snidget_deg
```

Do you remember how to remove the header line in the counts table?

{{Sdet}}

Solution{{Esum}}

```
sed '1d' counts.txt > counts_no_header.txt
```

Save the counts table without header, we will need it later.

{{Edet}}

Next, how do we remove columns 2 through 6 of the counts table and convert it from tab delimited to csv?

{{Sdet}}

Solution{{Esum}}

```
cut -f1,7-12 counts_no_header.txt | tr '\t' ',' > counts.csv
```

Again, save the counts table without header, we will need it later.

{{Edet}}

Now that the correctly formatted counts table is generated. Let's see if we can remember how to run `deseq2.r` to generate the differential expression results.

{{Sdet}}

Solution{{Esum}}

```
Rscript $CODE/deseq2.r
```

{{Edet}}

Take a look at the `results.csv` file, which contains the differential expression analysis output. Can we sorted by largest to smallest fold change? In the sorted results table, what do you notice? How well do the fold change results match expected?

{{Sdet}}

Solution{{Esum}}

```
column -t -s ',' results.csv | (sed 1q; sort -k 6nr) | less -S
```

{{Edet}}

Visualizing expression

Let's create an expression heatmap. How do we do this? Looking at the heatmap, do the treatments (ie. BORED and EXCITED) cluster well together?

{{Sdet}}

Solution{{Esum}}

```
Rscript $CODE/create_heatmap.r
```

We will need to copy the result, `heatmap.pdf` to the public folder to view. And the BORED and EXCITED groups do cluster together.

{{Edet}}



Lesson 16 Practice

Objectives

In this lesson, we learned about the classification based approach for RNA sequencing analysis. In this approach, we are aligning our raw sequencing reads to a reference transcriptome rather than a genome. Here, we will get to practice what we learned using the Golden Snidget dataset.

Create a directory for the classification based analysis output

Before getting started, let's create an output directory for the classification based analysis results in the ~/biostar_class/snidget folder.

```
cd ~/biostar_class/snidget
```

```
mkdir -p classification_based/salmon
```

Indexing the reference transcriptome

To align raw sequencing reads to a reference transcriptome, we will need a reference transcriptome (ie. the sequences of known transcripts in FASTA format). Fortunately, a reference transcriptome is included with the Golden Snidget dataset, so we will just need to index the transcriptome using salmon. Do you recall how to do this? Note, we will store the transcriptome index in the ~/biostar_class/snidget/refs folder.

{{Sdet}}

Solution{{Esum}}

```
cd refs
```

```
salmon index -t transcripts.fa -i transcripts.idx
```

{{Edet}}



Obtaining expression counts

Next, we need to generate the counts (ie. number of reads that map to a transcript). But first, change back into the `~/biostar_class/snidget` folder and then take a moment to think about how you can do this for all of the samples in one go. Hint, it will be easier if you go into `~/biostar_class/snidget/reads` and create a text file called `ids.txt` that contains the sample names for this data set.

{{Sdet}}

Solution{{Esum}}

```
cd ~/biostar_class/snidget/reads
```

```
parallel echo {1}_{2} ::: BORED EXCITED ::: 1 2 3 > ids.txt
```

Now, go back up one directory to `~/biostar_class/snidget`

```
cd ~/biostar_class/snidget
```

```
cat reads/ids.txt | parallel "salmon quant -i refs/transcripts.idx -"
```

{{Edet}}

After getting the counts, change into the `classification_based` folder. How do we do this from the `~/biostar_class/snidget` directory.

{{Sdet}}

Solution{{Esum}}

```
cd classification_based
```

{{Edet}}

To proceed further with the analysis we need a `design.csv` file with the sample names, which are essentially the names of salmon output folders and the treatment condition of the samples. Take a moment to create the `design.csv` file.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
nano design.csv
```

Copy and paste the text below into the nano editor, hit control x and then save to go back to the terminal.

```
sample,condition
BORED_1_SALMON,BORED
BORED_2_SALMON,BORED
BORED_3_SALMON,BORED
EXCITED_1_SALMON,EXCITED
EXCITED_2_SALMON,EXCITED
EXCITED_3_SALMON,EXCITED
```

```
{{Edet}}
```

Remember salmon quant generates one folder for each sample and saves the counts for each sample in that particular folder. We will need to combine these count files into one. Do you remember how to do this?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

We need to use the script `combine_transcripts.r`

```
Rscript $CODE/combine_transcripts.r design.csv salmon
```

```
{{Edet}}
```

Differential expression analysis

Next, how do we generate the differential expression results?

```
{{Sdet}}
```



```
Solution{{Esum}}
```

```
Rscript $CODE/deseq2.r
```

```
{{Edet}}
```

Visualization

Can we generate an expression heatmap?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
Rscript $CODE/create_heatmap.r
```

```
{{Edet}}
```

Next, let's generate the Principal Components Analysis plot. But first, we need to convert the counts.csv and design.csv files to their tab delimited counterparts. Remember in Lesson 14 that we used the tr command to convert a tab delimited text file to a comma separated file. Can you do the opposite?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cat counts.csv | tr ',' '\t' > counts.txt
```

```
cat design.csv | tr ',' '\t' > design.txt
```

```
{{Edet}}
```

After generating the tab delimited expression counts table and design file, take a moment to recall how we would generate the Principal Components Analysis plot.


```
{{Sdet}}
```



```
Solution{{Esum}}
```

```
Rscript $CODE/create_pca.r counts.txt design.txt 6
```

```
{{Edet}}
```

Take a look at the differential expression results from the classification based approach. Do the gene expression changes reflect the expected?

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
column -t -s ',' results.csv | less -S
```

```
{{Edet}}
```

Bonus question (if there is time)

Can we combine just the fold change columns in the alignment based and classification based differential analysis results to compare the two?

Hint: first copy the results.csv file from the alignment based method in the ~/biostar_class/snidget/snidget_deg folder to ~/biostar_class/snidget and name it snidget_deg_genome.csv. Then, copy the results.csv file from the ~/biostar_class/snidget/classification_based folder to the ~/biostar_class/snidget folder and rename it snidget_deg_transcriptome.csv. Then make use of the cut, sed, sort, and paste commands to get a merged file. Work in the ~/biostar_class/snidget directory for this exercise.

```
{{Sdet}}
```

```
Solution{{Esum}}
```

```
cd ~/biostar_class/snidget
```

```
cp ~/biostar_class/snidget/snidget_deg/results.csv snidget_deg_genome.csv
```

```
cp classification_based/results.csv snidget_deg_transcriptome.csv
```

```
cut -f1,5 -d ',' snidget_deg_transcriptome.csv | (sed -u 1q; sort) |
```

```
cut -f1,5 -d ',' snidget_deg_genome.csv | (sed -u 1q; sort) | less -f
```

```
paste snidget_deg_genome_sorted.csv snidget_deg_transcriptome_sorted
```

The left half of the table are fold changes derived from the alignment based analysis. The right half of the table are fold changes derived from classification based analysis.

```
column -t -s ',' snidget_deg_genome_vs_transcriptome.csv | less -S
```

{{Edet}}



Database for Annotation, Visualization and Integrated Discovery (DAVID) - practicing what we learned

Learning objectives

In this practice session, we will practice using DAVID.

Practicing what we learned

To help us apply what we learned during the lesson, we are going to use DAVID to obtain some functional annotation information on upregulated genes in the HCC1395 dataset. Recall that the HCC1395 dataset is derived from a study that profiled the transcriptome of HCC1395 breast cancer cell line and the HCC1395BL lymphoblastoid line, which makes this a tumor versus normal transcriptome comparison -- [Griffith lab RNA sequencing course \(https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/\)](https://rnabio.org/module-01-inputs/0001/05/01/RNAseq_Data/)

Getting the data

The genes that exhibit higher expression in the tumor tissue (ie. \log_2 fold change ≥ 1 and false discovery rate ≤ 0.05) have been subsetted into the file [hcc1395_deg_chr22_up_genes.txt](#).

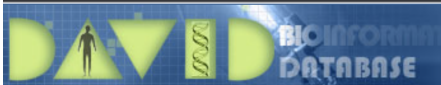

Upload the data to DAVID

Before uploading the data, open `hcc1395_deg_chr22_up_genes.txt` to see what the content looks like. These should resemble gene symbols so select "OFFICIAL_GENE_SYMBOL" as the identifier type in Step 2. However, as a hint, DAVID might not recognize these and may divert you to the gene identifier conversion tool. If it does divert you, convert these to ENSEMBL_GENE_ID. At the gene ID conversion tool, how many IDs are available for conversion? Thus, out of the 157 genes that we are starting with, how many can we use in functional annotation analysis?

{{Sdet}}

Solution{{Esum}}






We are left with 108 genes

DAVID Bioinformatics Resources
Laboratory of Human Retrovirology and Immunoinformatics (LHRI)

Gene Accession Conversion Tool [Help](#)

Gene Accession Conversion Statistics

Conversion Summary		
ID Count	In DAVID DB	Conversion
0	Yes	Successful
0	Yes	None
49	No	None
108	Ambiguous	Pending
Total Unique User IDs: 157		
Summary of Ambiguous Gene IDs		
ID Count	Possible Source	Convert All
108	OFFICIAL_GENE_SYMBOL	 →
All Possible Sources For Ambiguous IDs		
Ambiguous ID	Possibility	Convert
LGALS2	OFFICIAL_GENE_SYMBOL	 →
NDUFA9P1	OFFICIAL_GENE_SYMBOL	 →
IFT27	OFFICIAL_GENE_SYMBOL	 →
CCDC157	OFFICIAL_GENE_SYMBOL	 →

{{Edet}}

Convert the gene IDs and then submit the converted IDs as a gene list called hcc1359_deg_chr22_up.

Results

Gene Functional Classifier

Using the Gene Functional Classifier to group genes with similar annotations, how many clusters do we get?

{{Sdet}}

Solution{{Esum}}

Gene Functional Classification Tool
DAVID Bioinformatics Resources, NIAID/NIH

Home Start Analysis Shortcut to DAVID Tools Technical Center Downloads & APIs Term of Service About DAVID About LHRI

Upload List Background

Gene List Manager

Select to limit annotations by one or more species [Help](#)

- Use All Species -
Homo sapiens(105)

Select Species

List Manager [Help](#)
hcc1359_deg_chr22_u

Select List to:
Use Rename
Remove Combine
Show Gene List

Gene Functional Classification Result [Help and Tool Manual](#)

Current Gene List: **hcc1359_deg_chr22_up**
Current Background: **Homo sapiens**
100 DAVID IDs

Options Classification Stringency Medium

Rerun using options Create Sublist

1 Cluster(s) [Download File](#)

Gene Group 1	Enrichment Score: 0.36	RG
1 <input type="checkbox"/> ENSG00000166897, ENSG00000243902	extracellular leucine rich repeat and fibronectin type III domain containing 2(ELFN2)	
2 <input type="checkbox"/> ENSG00000073150	pannexin 2(PANX2)	
3 <input type="checkbox"/> ENSG00000100191	solute carrier family 5 member 4(SLC5A4)	
4 <input type="checkbox"/> ENSG00000221890	neuronal pentraxin receptor(NPTXR)	
5 <input type="checkbox"/> ENSG00000198792	transmembrane protein 184B(TMEM184B)	
6 <input type="checkbox"/> ENSG00000100258	lipase maturation factor 2(LMF2)	
7 <input type="checkbox"/> ENSG00000099960	solute carrier family 7 member 4(SLC7A4)	

93 gene(s) from your list are not in the output.

{{Edet}}

Potential Diseases

Now, run the Functional Annotation Tool. Look at the gene wise view for DISGENET, are there any genes in our input that map to cancer?

{{Sdet}}

Solution{{Esum}}

DAVID Bioinformatics Resources
Laboratory of Human Retrovirology and Immunoinformatics (LHRI)

Functional Annotation Table [Help and Manual](#)

Current Gene List: **hcc1359_deg_chr22_up**
Current Background: **Homo sapiens**
100 DAVID IDs

46 record(s) [Download File](#)

ENSG00000182858	ALG12 alpha-1,6-mannosyltransferase(ALG12)	Related Genes	Homo sapiens
DISGENET	Congenital disorder of glycosylation type 1G,		
ENSG00000128266	G protein subunit alpha z(GNAZ)	Related Genes	Homo sapiens
DISGENET	Liver carcinoma,		
ENSG00000100139	MICAL like 1(MICALL1)	Related Genes	Homo sapiens
DISGENET	Malignant neoplasm of breast,		
ENSG00000197182	MIRLET7B host gene(MIRLET7BHG)	Related Genes	Homo sapiens
DISGENET	Lung Neoplasms, Malignant neoplasm of lung,		
ENSG00000169184	MN1 proto-oncogene, transcriptional regulator(MN1)	Related Genes	Homo sapiens
DISGENET	Craniosynostosis, Leukemia, Myelocytic, Acute, Acute Myeloid Leukemia, M1, Feeding difficulties, Polymicrogyria, Global developmental delay, Familial meningioma, hearing impairment, Central hypotonia, Abnormal corpus callosum morphology, Abnormality of the cerebellum, Acute Myeloid Leukemia (AML-M2), Deformity of facial bone, Intellectual Disability, Abnormality of the face,		

{{Edet}}

Gene Ontology ?

Go under the Gene Ontology section in the Annotation Summary Results and expand on it. Click the chart view for GOTERM_BP_DIRECT to look at some biological processes that the upregulated genes in the dataset map to. What are some of the processes? Does it make sense that we get genes that are expressed higher in the tumor samples mapping to these processes?

{{Sdet}}

Solution{{Esum}}

- cellular response to glucose starvation
- cellular response to nutrient
- angiogenesis
- glucose transmembrane transport

Current Gene List: hcc1359_deg_chr22_up

Current Background: Homo sapiens

100 DAVID IDs

Options

Rerun Using Options

Create Sublist

18 chart records

Sublist	Category	Term
<input type="checkbox"/>	GOTERM_BP_DIRECT	cellular response to glucose starvation
<input type="checkbox"/>	GOTERM_BP_DIRECT	social behavior
<input type="checkbox"/>	GOTERM_BP_DIRECT	cellular response to nutrient
<input type="checkbox"/>	GOTERM_BP_DIRECT	negative regulation of axon regeneration
<input type="checkbox"/>	GOTERM_BP_DIRECT	negative regulation of ERK1 and ERK2 cascade
<input type="checkbox"/>	GOTERM_BP_DIRECT	positive regulation of transcription from RNA polymerase II promoter in response to endoplasmic reticulum stress
<input type="checkbox"/>	GOTERM_BP_DIRECT	anterior/posterior pattern specification
<input type="checkbox"/>	GOTERM_BP_DIRECT	mRNA processing
<input type="checkbox"/>	GOTERM_BP_DIRECT	canonical Wnt signaling pathway
<input type="checkbox"/>	GOTERM_BP_DIRECT	muscle organ development
<input type="checkbox"/>	GOTERM_BP_DIRECT	cholesterol homeostasis
<input type="checkbox"/>	GOTERM_BP_DIRECT	angiogenesis
<input type="checkbox"/>	GOTERM_BP_DIRECT	ERAD pathway
<input type="checkbox"/>	GOTERM_BP_DIRECT	DNA methylation involved in gamete generation
<input type="checkbox"/>	GOTERM_BP_DIRECT	long term synaptic depression
<input type="checkbox"/>	GOTERM_BP_DIRECT	MAPK cascade
<input type="checkbox"/>	GOTERM_BP_DIRECT	glucose transmembrane transport
<input type="checkbox"/>	GOTERM_BP_DIRECT	endothelial cell proliferation

{{Edet}}

Functional Annotation Clusters

How many annotation clusters were generated using default parameters?

{{Sdet}}

Solution{{Esum}}



15 clusters were generated

DAVID
LHRI

DAVID Bioinformatics Resources
 Laboratory of Human Retrovirology and Immunoinformatics (LHRI)

Functional Annotation Clustering

[Help and Manual](#)

Current Gene List: hcc1359_deg_chr22_up
Current Background: Homo sapiens
100 DAVID IDs

Options **Classification Stringency** Medium

15 Cluster(s) [Download File](#)

Annotation Cluster	Enrichment Score		Count	P_Value	Benjamini
Annotation Cluster 1	Enrichment Score: 2.42	G			
<input type="checkbox"/> SMART	CH	RT	4	2.7E-3	1.5E-1
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:Calponin-homology (CH)	RT	4	4.4E-3	7.2E-1
<input type="checkbox"/> INTERPRO	Calponin_homology_domain	RT	4	4.7E-3	9.2E-1
Annotation Cluster 2	Enrichment Score: 1.12	G			
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:EGF-like 8; calcium-binding	RT	3	4.5E-3	7.2E-1
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:EGF-like 7; calcium-binding	RT	3	5.7E-3	7.2E-1
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:EGF-like 3; calcium-binding	RT	3	1.1E-2	1.0E0
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:EGF-like 2; calcium-binding	RT	3	2.4E-2	1.0E0
<input type="checkbox"/> INTERPRO	EGF-type aspartate/asparagine hydroxylation site	RT	3	6.8E-2	1.0E0
<input type="checkbox"/> INTERPRO	EGF-like calcium-binding	RT	3	1.0E-1	1.0E0
<input type="checkbox"/> SMART	EGF_CA	RT	3	1.0E-1	1.0E0
<input type="checkbox"/> INTERPRO	Insulin-like growth factor binding protein, N-terminal	RT	3	1.1E-1	1.0E0
<input type="checkbox"/> SMART	EGF	RT	3	1.9E-1	1.0E0
<input type="checkbox"/> UP_SEQ_FEATURE	DOMAIN:EGF-like	RT	3	2.1E-1	1.0E0
<input type="checkbox"/> UP_KW_DOMAIN	EGF-like domain	RT	3	2.5E-1	1.0E0
<input type="checkbox"/> INTERPRO	Epidermal growth factor-like domain	RT	3	2.7E-1	1.0E0
<input type="checkbox"/> UP_KW_LIGAND	Calcium	RT	7	3.6E-1	1.0E0
<input type="checkbox"/> GOTERM_MF_DIRECT	calcium ion binding	RT	5	4.4E-1	1.0E0
Annotation Cluster 3	Enrichment Score: 0.94	G			

{{Edet}}

Look thoroughly through the annotations, are any expected annotations and unexpected ones given that this is a tumor versus normal comparison dataset?

References

References

Content for this course series was adapted / inspired by the following sources:

1. *The Biostar Handbook* (<https://www.biostarhandbook.com/index.html>), 2nd Edition, Istvan Albert.
2. Malachi Griffith, Jason R. Walker, Nicholas C. Spies, Benjamin J. Ainscough, Obi L. Griffith. 2015. Informatics for RNA-seq: A web resource for analysis on the cloud. *PLoS Comp Biol.* 11(8):e1004393.
3. Gabriel A. Devenyi (Ed.), Gerard Capes (Ed.), Colin Morris (Ed.), Will Pitchers (Ed.), Greg Wilson, Gerard Capes, Gabriel A. Devenyi, Christina Koch, Raniere Silva, Ashwin Srinath, ... Vikram Chhatre. (2019, July) swcarpentry/shell-novice: Software Carpentry: the UNIX shell, June 2019 (Version v2019.06.1). Zenodo. <http://doi.org/10.5281/zenodo.3266823> (<http://doi.org/10.5281/zenodo.3266823>)
4. NIH HPC group: <https://hpc.nih.gov/training/> (<https://hpc.nih.gov/training/>)

Biostars Software Description

To complement this course, we have created a Biostars module on Biowulf. The following programs have been included in that module:

Apps/Dependencies:

bbtools (<https://hpc.nih.gov/apps/bbtools.html>)

bcftools (<http://samtools.github.io/bcftools/bcftools.html>)

bedtools (<https://hpc.nih.gov/apps/bedtools.html>)

bio (<https://www.bioinfo.help/index.html>)

bioawk (<https://hpc.nih.gov/apps/bioawk.html>)

blastn (<https://hpc.nih.gov/apps/Blast.html>)

bowtie2 (<https://hpc.nih.gov/apps/bowtie2.html>)

bwa (<https://hpc.nih.gov/apps/bwa.html>)

cd-hit (<https://hpc.nih.gov/apps/cd-hit.html>)

csvkit (<https://hpc.nih.gov/apps/csvkit.html>)

csvtk (<https://github.com/shenwei356/csvtk>)

cutadapt (<https://hpc.nih.gov/apps/cutadapt.html>)

datamash (<https://www.gnu.org/software/datamash/>)

efetch (<https://biopython.org/docs/1.75/api/Bio.Entrez.html#>)

emboss (<http://emboss.open-bio.org>)

entrez-direct (<https://www.ncbi.nlm.nih.gov/books/NBK179288/>)

fastqc (<https://hpc.nih.gov/apps/fastqc.html>)

featureCounts (<https://academic.oup.com/bioinformatics/article/30/7/923/232889>)

freebayes (<https://hpc.nih.gov/apps/freebayes.html>)

gffread (<https://github.com/gperte/gffread>)

hisat2 (<https://hpc.nih.gov/apps/hisat.html>)

kallisto (<https://hpc.nih.gov/apps/kallisto.html>)

mafft (<https://hpc.nih.gov/apps/mafft.html>)

minimap2 (<https://hpc.nih.gov/apps/minimap2.html>)

multiqc (<https://hpc.nih.gov/apps/multiqc.html>)

parallel (<https://hpc.nih.gov/apps/parallel.html>)

picard (<https://hpc.nih.gov/apps/picard.html>)

python (<https://www.python.org>)

R/Rscript (<https://hpc.nih.gov/apps/R.html>)

readseq (<https://www.ebi.ac.uk/Tools/sfc/readseq/>)

salmon (<https://hpc.nih.gov/apps/salmon.html>)

samtools (<https://hpc.nih.gov/apps/samtools.html>)

seqkit (<https://hpc.nih.gov/apps/seqkit.html>)

seqtk (<https://hpc.nih.gov/apps/seqtk.html>)

snpEff (<https://hpc.nih.gov/apps/snpEff.html>)

sra-tools (<https://github.com/ncbi/sra-tools>)

STAR (<https://hpc.nih.gov/apps/STAR.html>)

subread (<https://hpc.nih.gov/apps/subread.html>)

trimmomatic (<https://hpc.nih.gov/apps/trimmomatic.html>)

ucsc-bedgraphtobigwig (https://hpc.nih.gov/apps/Genome_Browser.html)

wget (<https://www.gnu.org/software/wget/>)

R packages:

biomaRt (<https://bioconductor.org/packages/release/bioc/html/biomaRt.html>)

DESeq2 (<https://bioconductor.org/packages/release/bioc/html/DESeq2.html>)

edgeR (<https://bioconductor.org/packages/release/bioc/html/edgeR.html>)

GenomeInfoDb ([https://bioconductor.org/packages/release/bioc/html/\[GenomeInfoDb.html\]](https://bioconductor.org/packages/release/bioc/html/[GenomeInfoDb.html]))

ggplot2 (<https://ggplot2.tidyverse.org>)

gplots (<https://cran.r-project.org/web/packages/gplots/index.html>)

pheatmap (<https://www.rdocumentation.org/packages/pheatmap/versions/1.0.12/topics/pheatmap>)

RColorBrewer (<https://cran.r-project.org/web/packages/RColorBrewer/index.html>)

tidyverse (<https://bioconductor.org/packages/release/bioc/vignettes/destiny/inst/doc/tidyverse.html>)

tximport (<https://bioconductor.org/packages/release/bioc/html/tximport.html>)

Additional Resources

Additional Resources

Working with Unix

1. Introduction to Unix from the *Bioinformatics Workbook* (<https://bioinformaticsworkbook.org/Appendix/Unix/unix-basics-1.html#gsc.tab=0>)
2. Unix from Happy Belly Bioinformatics (<https://astrobiomike.github.io/unix/>)
3. Brandies PA, Hogg CJ (2021) Ten simple rules for getting started with command-line bioinformatics. *PLoS Comput Biol* 17(2): e1008645. <https://doi.org/10.1371/journal.pcbi.1008645> (<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008645>)
4. Linux Command List (from fosswire.com)

RNA-Seq

1. RNA-seq Bioinformatics Course (<https://rnabio.org/>) from the Griffith lab.

Logging into Biowulf

1. You must be on VPN (Virtual Private Network) or the NIH campus internet to access Biowulf. [Request VPN at NCI \(https://service.cancer.gov/ncisp?id=kb_article_view&sys_kb_id=68c557251bbbc110932da8efe54bcb39\)](https://service.cancer.gov/ncisp?id=kb_article_view&sys_kb_id=68c557251bbbc110932da8efe54bcb39).
2. If you have a Biowulf account but have not used it in 60 days, please send email to staff@hpc.nih.gov and ask them to unlock your account.
3. If you do not have a Biowulf account: You will be assigned a Student ID and Password for class help sessions.
4. **WARNING:** When you log into Unix systems, you will not be able to see your password as you type it. The cursor does not move, and you may think you are doing something wrong. This is a safety feature on Unix systems- someone looking over your shoulder can not see your password. Just type in the password and it will work.
5. If this is the first time you are logging into Unix/Biowulf, you will see a Security Alert as soon as you log-in. Answer “**yes**” to the security prompt.
6. Logging into Biowulf from a Mac:
 - a. Open the Terminal program in Applications/ (Try cmd spacebar to open a search tool and type Terminal)
 - b. `ssh username@biowulf.nih.gov` where username is your username or student id if you are using a student account
7. Logging into Biowulf from a PC. You may need to go to service.cancer.gov to install one of these options if not included.
 - a. Windows 10/11 has OpenSSH
 - b. Windows 10 OS has built-in SSH using the Power Shell
 - c. Older OS will need to download PuTTY (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>). Under “Alternative binary files” , download 64-bit x86 putty.exe (this will work for most installations).

Accessing the Biostar Handbook

This course follows along closely with the [Biostar Handbook \(https://www.biostarhandbook.com/index.html\)](https://www.biostarhandbook.com/index.html) content, and course registration comes with a *Biostar Handbook* license. You will receive an email from hello@biostarhandbook.com with information detailing how to set up your account.

Your account will give you access to all five volumes of the *Biostar Handbook*:

1. *The Biostar Handbook*
2. *The Art of Bioinformatics Scripting*
3. *RNA-Seq by Example*
4. *Corona Virus Genome Analysis*
5. *Biostar Workflows*

While we will cover some of the topics in these volumes, we only scratch the surface. Feel free to use your license to learn more about bioinformatics.

If you decide to tackle the concepts in these volumes on Biowulf, consider using the [Biostars on Biowulf module](#).

Biostars on Biowulf

To complement this course, there is a module available on Biowulf with installed programs associated with the Biostar Handbook. During class, we will work on the command line on the GOLD system on DNAnexus. However, this system will not be available outside of class time. There are two options for catching up on class work or working on practice problems outside of class. (1) You can install a Biostars conda environment on your local computer (See Biostar Handbook for [instructions \(https://www.biostarhandbook.com/computer-setup.html\)](https://www.biostarhandbook.com/computer-setup.html)). (2) You can use the Biowulf HPC cluster and the Biostars Biowulf module. For option 2, you will need to obtain a [Biowulf account \(https://hpc.nih.gov/docs/accounts.html\)](https://hpc.nih.gov/docs/accounts.html).

How to use the Biostars Biowulf module

Loading the module

1. Access the NIH network; this will require you to VPN if off campus.
2. Connect to Biowulf

```
ssh user_name@biowulf.nih.gov
```

 where `user_name` is your NIH username.

3. Use `sinteractive` to work on an interactive node. This will result in 4GB of memory and 2 CPUs. Alternatively, you may submit jobs by making scripts and submitting jobs via `sbatch` (See lesson 5).

Note: If you are planning to use the `sratoolkit` to download data from the SRA, you will need to allocate local scratch space (`sinteractive --gres=lscratch:30`).

4. If using an interactive node, run the following

```
source /data/classes/BTEP/apps/biostars/1.0/run_biostars.sh
```

This will do the following:

1. Runs a set of commands to setup the terminal environment.
2. Creates a data directory environmental variable (`$DATA`) where you can gain access to course data.
3. Runs `module use /data/classes/BTEP/apps/modules`
4. Runs `module load biostars`

- If you want to use the biostars module for other purposes or you want to submit jobs via `sbatch`, skip Step 4. You can load the module with the following:

```
1. module use /data/classes/BTEP/apps/modules
2. module load biostars
3. module help biostars
```

Course files found on DNAnexus will be made accessible on Biowulf. The path to course files has been assigned to the environment variable `$DATA`, which is automatically set when you run Step 4.

How to transfer data to and from Biowulf?

There is extensive documentation on transferring data to and from Biowulf at hpc.nih.gov (<https://hpc.nih.gov/docs/transfer.html>). If you simply would like to view .html files or other output, or you want to copy small files, consider [mounting HPC system directories to your local computer](https://hpc.nih.gov/docs/hpcdrive.html) (<https://hpc.nih.gov/docs/hpcdrive.html>).

Installing IGV

Instructions for installing IGV

The *Integrative Genome Viewer (IGV)* (<http://software.broadinstitute.org/software/igv/home>) is an open-source genome browser created and maintained by the Broad Institute. We will be using this to visualize genomic data.

To obtain IGV, please submit a ticket through [service.cancer.gov](https://service.cancer.gov/ncisp?id=nci_home) (https://service.cancer.gov/ncisp?id=nci_home) (Figure 1).

Figure 1

Once you are at the landing page, click on Order a Service and you will be prompted to log in with your NIH credentials. After logging in you will see a page that lists service categories (Figure 2). From here, look under the Categories pane and select Software.

Figure 2

Next, you will be taken to the software services page where you will select Install Software Not Managed by CBIIT.

Figure 3

Fill out the relevant information and submit (Figure 4).

Figure 4

Note to make sure you can at least get IGV opened (ie. test it) before ending the help session with service.cancer.gov.

Questions and Answers

Questions and answers

BTEP Bioinformatics for Beginners (September 13th, 2022 - December 13th, 2022)

Questions and Answers

Below, you will find questions and answers brought up in the course polls for the BTEP Bioinformatics for Beginners course series that took place from September 13th, 2022 to December 13th, 2022.

Question 1: Normalization - when to normalize and what type of normalization

Answer to Question 1: Normalization is always needed in RNA sequencing because this helps to remove technical effects so that we are comparing only the biological differences between experimental conditions. Technical factors include things like batch effects, differences in sequencing depth (library size) between samples, gene length, and library composition. Various techniques that normalized based on library size and gene length were discussed in class. Refer to the documentation on [RNA sequencing quantitation \(https://btep.ccr.cancer.gov/docs/b4b/RNASeq_Overview/06.Quantitation/#count-normalization\)](https://btep.ccr.cancer.gov/docs/b4b/RNASeq_Overview/06.Quantitation/#count-normalization).

Other types of normalization method include quantile normalization, which has been found to work well for RNA sequencing. Further, some differential expression analysis packages have their own normalization scheme so all users need to do is to provide the raw integer expression counts.

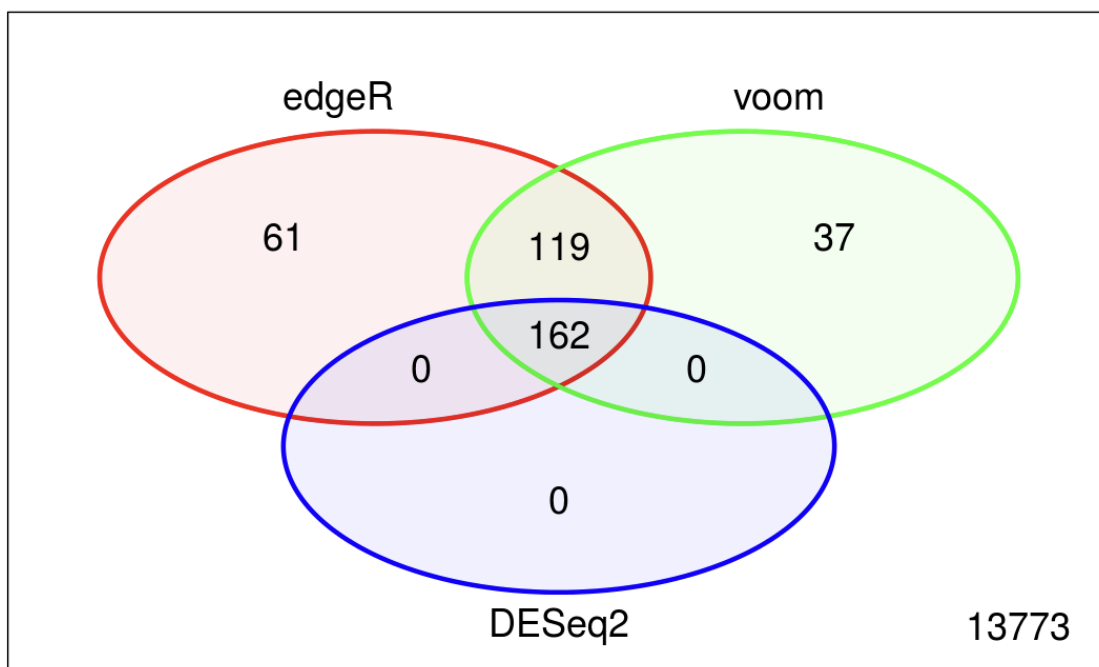
Question 2: Question regarding normalization sent by a class participant

“Hi-just a followup comment. The experience of the bioinformatics people that my lab works with is that normalization in RNAseq is everything and you are wasting your time if you don't do it properly. Especially if your treatment leads to an increase in RNA content of the cell-like resting vs activated T cells. The truth is, most everything goes up under these conditions but most software adjusts things such that half genes go up and half go down. In fact most of what is indicated to go down under these conditions are things that just don't go up as much as other genes. Anyway, that was a long explanation for why understanding how to do this properly is likely to be crucial. In my opinion, anyway. Thanks very much”

Answer to Question 2: All normalization methods (and consequently differential expression methods) contain assumptions about the data you are analyzing, and there is no single statistical method that can completely capture biological phenomena. It's the job of the analyst to be aware of these assumptions and whether they apply to the data you are analyzing. For

example, the most commonly used methods in the detection of differential gene expression (e.g. DeSeq2 and edgeR) assume that most genes are not differentially expressed and start with a correction for library size. This assumption might not apply to your experimental setup: If you suspect your treatment would result in differences in the total amount of signal (RNA yield) between your two samples (e.g. activated vs. resting t-cells, overexpression of cMyc), normalizing by library size wouldn't be accurate to the biology of your sample since you'd be removing this difference by normalizing by library size. If you want to capture these absolute differences in signal then you would have to modify your experimental approach (e.g. include a spike-in control - see <https://journals.asm.org/doi/full/10.1128/MCB.00970-14> (<https://journals.asm.org/doi/full/10.1128/MCB.00970-14>)).

An additional note on normalization: As stated above most common methods of differential gene expression assume most genes are not differentially expressed. When this is true, as in the case of "classical" RNAseq experiments, then the normalization (and indeed analysis methods) make only minor differences. In this example the majority of the identified DEGs are common to the three analysis methods (DeSeq2, edgeR, and limma-voom) and these use slightly different normalizations.



Question 3: Strategies for batch correction (when to use it, what approaches)?

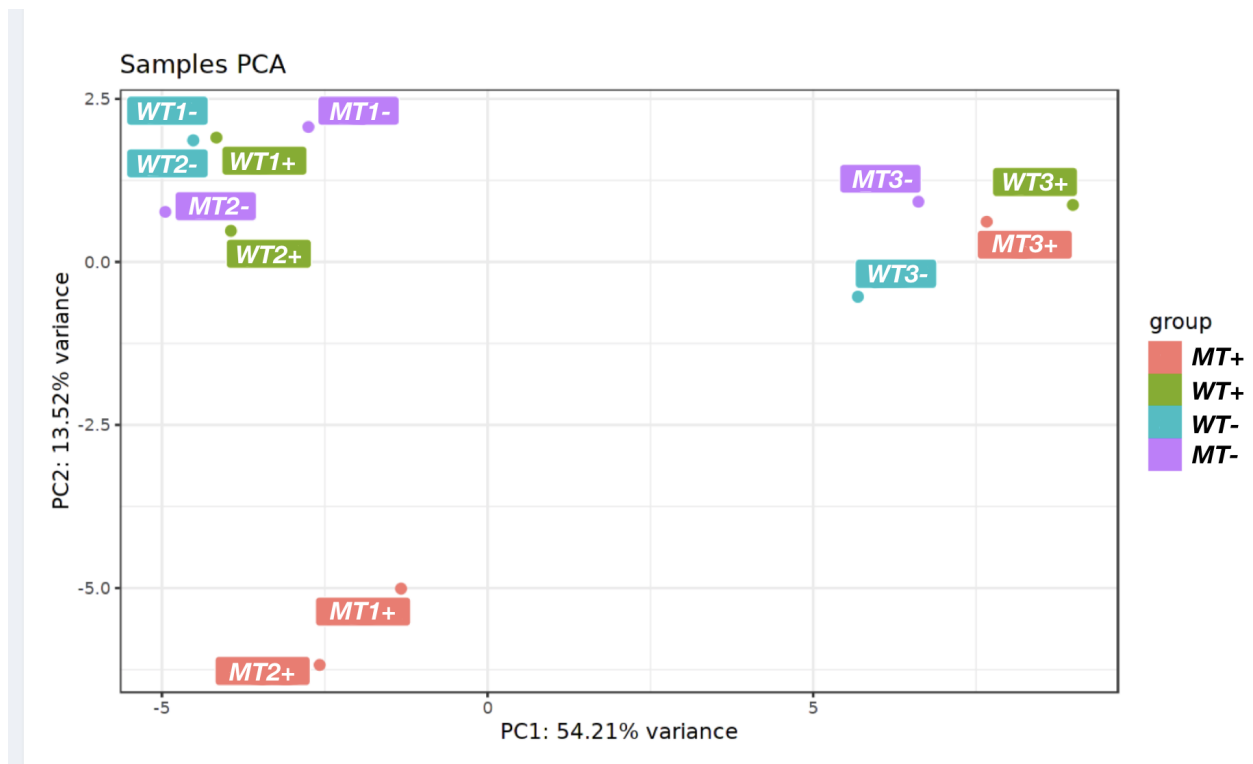
Answer to Question 3: Batch effects are caused by variation between samples that are not due to your experimental design (i.e. technical variation). The best approach to take regarding batch correction is to design your experiments in such a way that you can avoid it entirely: if possible, isolate and prepare your samples on the same day, use the same reagents and locations for preparing your samples). If you must have batches and can't prepare your samples on the same day, make sure they're not confounded: have representatives of your biological groups

(e.g. controls and treatments) in each batch, and keep track of any batches in your experimental records and metadata so you can identify batch effects if they exist in your data (see here for a nice overview (https://hbctraining.github.io/Intro-to-rnaseq-fasrc-salmon-flipped/lessons/02_experimental_planning_considerations.html)).

You can identify batch effects through an initial exploratory analysis of your data (e.g. hierarchical clustering, PCA analysis).

An example of how PCA can be used to identify batch effects: When labelled according to sample type (Wildtype +/- drug and Mutant +/- drug) replicates 3 are outliers, and make no sense. However, when knowledge of when the different samples were processed (two "batches") is added, it is apparent that replicates 3 were all done on a different day and represent batch effects that must be factored into the analysis (and it looks like the drug only affected the mutant).

PCA before batch correction:

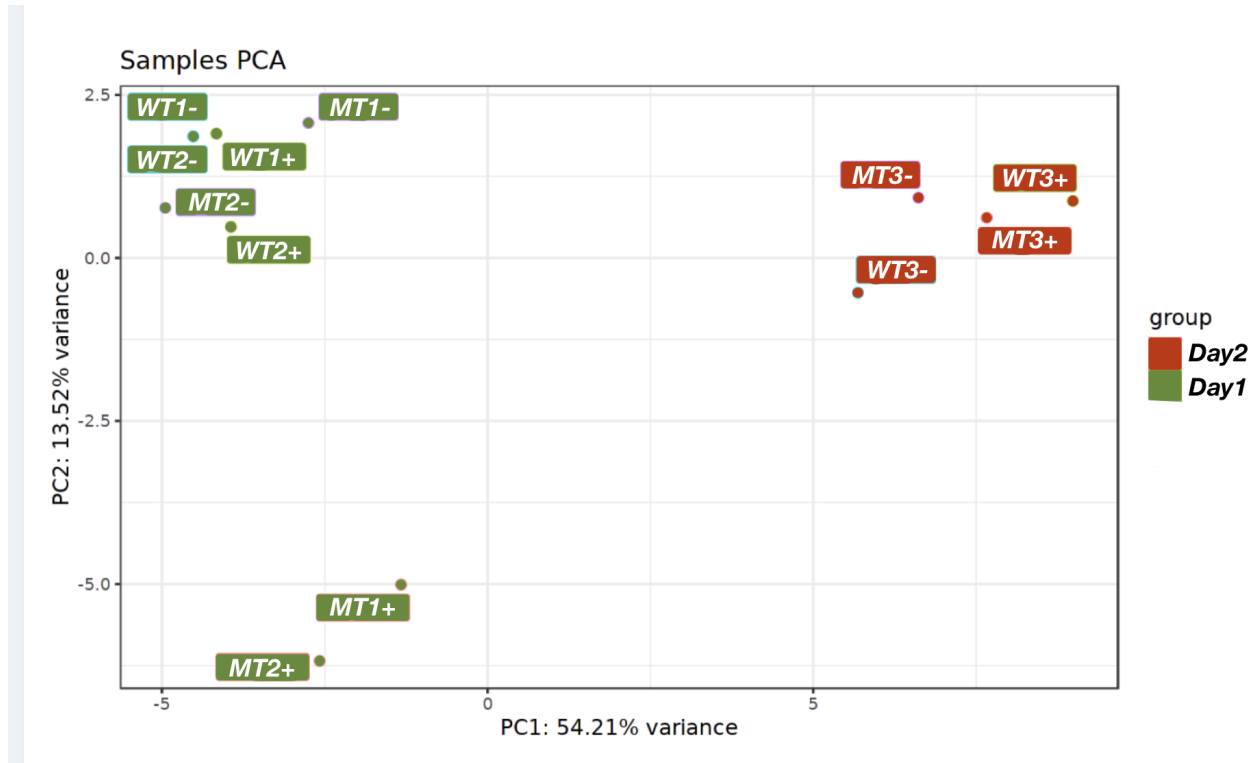


PCA

after

batch

correction:



Tools to adjust your data for batch effects exist: there are two common methods, ComBAT and sva which can be found in the Bioconductor package [sva](https://www.bioconductor.org/packages/release/bioc/html/sva.html) (<https://www.bioconductor.org/packages/release/bioc/html/sva.html>). A review on identifying and removing batch effects is reviewed in [this article here](https://www.nature.com/articles/nrg2825) (<https://www.nature.com/articles/nrg2825>), however, I would recommend consulting a bioinformatician with experience in this before venturing out to do this on your own.

Question 4: When working with large data sets what is the expectations for time and storage space?

Answer to Question 4: This depends on your project but FASTQ files can be many gigabytes in size and you have FASTQ files for many samples, the space can add up. You can always [add more storage space to your Biowulf account](https://hpc.nih.gov/storage/) (<https://hpc.nih.gov/storage/>). The amount of time it takes to complete your analysis depends on the pipeline that you are using and whether you have to create your own pipeline. If creating your own analysis pipeline, you will also need to optimize the run time of your analysis. Reach out to one of our expert analysts to discuss and plan before starting your study.

Question 5: What methods are available for transferring fastq files from my disk to Biowulf?

Answer to Question 5:

- Mount HPC drives to local machine (slow)
- Use secure copy (scp)
- Use SFTP client (Filezilla, Fugu, or command line SFTP)
- Globus for very large datasets (ie. FASTQ files)

See [Transferring data to/from the NIH HPC systems \(https://hpc.nih.gov/docs/transfer.html\)](https://hpc.nih.gov/docs/transfer.html) for more about transferring files to and from Biowulf.

Biowulf file transfer tutorials on YouTube

- [mounting \(https://youtu.be/H8ZksTK3EtE?t=88\)](https://youtu.be/H8ZksTK3EtE?t=88)
- [secure copy or scp \(https://youtu.be/H8ZksTK3EtE?t=258\)](https://youtu.be/H8ZksTK3EtE?t=258)
- [Globus \(https://youtu.be/mg9-a1OuDqo\)](https://youtu.be/mg9-a1OuDqo)

While there are several approaches for transferring file to Biowulf from local, if you are working with many FASTQ files, Globus would be the goto solution.

Question 6: What should I do after downloading RNAseq data from public website(e.g. TCGA) and check for data status?

Answer to Question 6: This depends on the study. For instance, you may find BAM files in a TCGA RNA sequencing study or expression counts. So depending on the data available, this will determine your starting point. When working with public datasets, it is very important to learn as much about a study as you can prior to working with data (ie. do not just download and blindly analyze).

Question 7: What are the various form of data format or quality of public data (GEO, TCGA, Depmap, UK data, etc)?

Answer to Question 7: This varies depending on the repository and study. For instance, GEO does not have a standard set of data that investigators need to submit. Thus, in GEO studies, you may find those with only sequencing data (FASTQ files), expression data, differentially expression results, or a combination.

Question 8: If I want to merge two kinds of GEO study, what should I check in terms of data quality, format, etc?

Answer to Question 8: Start with raw data if at all possible, do some of the QC steps that we did in the class if they're applicable, and process the data uniformly (same programs, pipelines, etc).