## Contents

1 Instructor: Dr. Ross Whetten
   1.1 Course Description .................................................. 3

2 Indices and tables .......................................................... 55
CHAPTER 1

Instructor: Dr. Ross Whetten

1.1 Course Description

The Analysis of Deep Sequencing Data course is designed to introduce biologists to the Linux command-line computing environment, to cloud computing, and to open-source software for analysis of next-generation sequencing data. Class sessions consist of two-hour blocks, each beginning with presentation and discussion of a specific topic, followed by hands-on computing exercises using model datasets. A total of 45 two-hour blocks are scheduled over a 15-week period, and the course is offered once per calendar year. The importance of cloud computing is emphasized, due to the increasing demands for RAM and storage space required for analysis and storage of high-throughput DNA sequencing data, and the cost-effectiveness and flexibility provided by cloud computing solutions. Applications of sequencing discussed include genome sequencing (both de-novo and resequencing), transcriptome analysis, discovery of sequence and structural variations, ChIP-seq methods for mapping DNA-protein interactions, and genotyping by sequencing (GBS and RAD-seq methods). For each application of sequencing technology, discussion topics include experimental design strategies, methods for library construction, sources of experimental and biological variation, and analytical approaches available in open-source software packages. Computing exercises utilize the software discussed, and provide participants with the opportunity to carry out analysis of sample datasets using a virtual machine image through the NC State University Virtual Computing Lab. This Linux system is customized to provide the bioinformatics software described during the course, and is available for class participants to use at any time. The objective of the course is not to make course participants experts in every aspect of sequence analysis, but instead to empower participants to learn the specific skills they need by teaching basic skills in command-line Linux computing, and providing an introduction to the literature and on-line resources. The course is directed at graduate students, but has also attracted participation from faculty, post-doctoral researchers, and research technicians interested in expanding their skills in the area of sequence data analysis.
1.1.1 Semester Overview, 2021

Class meetings are hosted as zoom meetings from 8:30 to 10:20 am on Mondays, Wednesdays, and Fridays.

The Biostar Handbook is a resource for much of the reading.

The Piazza online forum page is available for questions and discussion.

Teaching assistant: Will Kohlway

Course Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>[Relevant Biostar Handbook Chapter]</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-22 Jan</td>
<td>Introduction to Linux and the command-line interface [1, 2, 4]</td>
<td></td>
</tr>
<tr>
<td>25-29 Jan</td>
<td>Sequencing instruments, Experimental design, Data preprocessing and quality control [8, 9, 10]</td>
<td></td>
</tr>
<tr>
<td>1-5 Feb</td>
<td>Error correction and alignment</td>
<td></td>
</tr>
<tr>
<td>8-12 Feb</td>
<td>Assembly - transcriptomes and genomes [22]</td>
<td></td>
</tr>
<tr>
<td>15-19 Feb</td>
<td>Re-sequencing, alignment, structural variation [17, 18, 23]</td>
<td></td>
</tr>
<tr>
<td>22-26 Feb</td>
<td>Discovery and genotyping of genetic variation</td>
<td></td>
</tr>
<tr>
<td>1-4 Mar</td>
<td>R and R Studio - lectures and exercises through Software Carpentry website, sections 1 - 8</td>
<td></td>
</tr>
<tr>
<td>5 Mar</td>
<td>Wellness Day (no class)</td>
<td></td>
</tr>
<tr>
<td>8-12 Mar</td>
<td>R and R Studio, continued - sections 9 - 12. Advanced: Data Carpentry for Genomics</td>
<td></td>
</tr>
<tr>
<td>15-19 Mar</td>
<td>Transcriptome analysis: differential gene expression,</td>
<td></td>
</tr>
<tr>
<td>Date Range</td>
<td>Course Content</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>22 &amp; 26 Mar</td>
<td>Genome analysis: ChIP-seq, DHS-seq, 3-D conformation (No Class March 24th)</td>
<td></td>
</tr>
<tr>
<td>29 Mar-2 Apr</td>
<td>Linux command-line tools: awk, sed, and bash [15]</td>
<td></td>
</tr>
<tr>
<td>5-9 Apr</td>
<td>CLC Genomics Workbench - data QC and pre-processing</td>
<td></td>
</tr>
<tr>
<td>12-16 Apr</td>
<td>CLC Genomics Workbench - RNA-seq analysis</td>
<td></td>
</tr>
<tr>
<td>19-23 Apr</td>
<td>CLC Genomics Workbench - genome assembly tutorial</td>
<td></td>
</tr>
<tr>
<td>26-30 Apr</td>
<td>CLC Genomics Workbench - variant analysis</td>
<td></td>
</tr>
</tbody>
</table>

**General background information and course resources**

- General advice on troubleshooting
- Course syllabus
- Class bash history files from instructor VCL instances
- Lior Pachter's list of sequencing-based assays: *Seq
- The R statistical programming environment
- Course resources
- Overview slides (old and outdated in some respects)
- Cloud computing with Amazon Web Services: setting up an AWS account and starting an instance
- OMICtools software database publication and website
- Qiagen webpage of tutorials for CLC workbench programs
- Colibri'read project webpage on reference-assembly-free programs for SNP and indel detection and moress
A flow-chart overview of DNA sequencing experiments

1.1.2 Readings and Resources

Course Notes

- A Google Doc version of notes for each class session. Students have view permission; instructors have edit access. This provides a way for instructors to write code that students can copy into a terminal window on another computer.

Global overview books and papers

- The Biostar Handbook - Bioinformatics Data Analysis Guide. Istvan Albert and others. Available online

Library construction and experimental design


• Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. Aird et al, Genome Biology 12:R18, 2011 Full Text
• RNA-seq differential expression studies: more sequence or more replication? Liu et al., Bioinformatics 30: 301 - 304, 2014. Publisher Web Site
• Power analysis and sample size estimation for RNA-seq differential expression. Ching et al., RNA 20: 1684 -1696, 2014. Publisher Web Site
• Guidance for RNA-seq co-expression network construction and analysis: safety in numbers. Ballouz et al., Bioinformatics 31: 2123 - 2130, 2014. Publisher Web Site
• Points of Significance: Nested designs. Krzywinski et al., Nature Methods 11: 977–978, 2014 Publisher Web Site
• Points of significance: Sources of variation Altman & Krzywinski. Nature Methods 12: 5 – 6, 2015 Publisher Web Site
• Compilation of DNA sequencing library preparation Methods: as a Poster & an extensive methods review PDF.
• Compilation of RNA sequencing library preparation Methods: as a Poster & an extensive methods review PDF.
• A Poster compiling Single-Cell sequencing methods.
• An overview of recent publications for cell biology and complex disease research with Illumina technology.

Data formats, data management, and alignment software tools

• The Sequence Alignment/Map format and SAMtools. Li et al, Bioinformatics 25(16):2078-9, 2009 PubMedCentral
• SAM format specification file
• BEDTools: a flexible suite of utilities for comparing genomic features. Quinlan and Hall, Bioinformatics 26:841-842, 2010. Publisher Website

Data quality assessment, filtering, and correction

• FastQC: a quality control tool for high-throughput sequence data. Home Page
• FASTX-toolkit: FASTQ/A short-reads pre-processing tools Home Page
• QuorUM: an error corrector for Illumina reads. Marçais et al. 2013 Arxiv preprint or 2015 PLoSOne paper
• Error correction of high-throughput sequencing datasets with non-uniform coverage. Medvedev et al., Bioinformatics 27(13):i137-41, 2011. PubMedCentral
• Characterization of the Conus bullatus genome and its venom-duct transcriptome. Hu et al., BMC Genomics 12:60, 2011 Full Text [Includes a novel strategy for estimating genome size from a partial transcriptome assembly and low-coverage (3x) genome sequence.]

De novo assembly

• Assembly of large genomes using second-generation sequencing. Schatz et al, Genome Res 20(9):1165-73, 2010. PubMedCentral
• High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Gnerre et al, PNAS 108(4): 1513-18, 2011 PubMedCentral
• Artemis: an integrated platform for visualization and analysis of high-throughput sequence-based experimental data. Carver et al, Bioinformatics 28(4):464 - 469, 2012 PubMedCentral
• Efficient de novo assembly of large genomes using compressed data structures. Simpson & Durbin, Genome Research 22:549-556, 2012 Full Text [Describes the String Graph Assembler (SGA), which assembled a human genome in less than 6 days using 54 Gb of RAM and a 123-processor compute cluster for calculation of an FM-index of the 1.2 billion reads]

Chromatin analysis

Bias Correction

• Identifying and mitigating bias in next-generation sequencing methods for chromatin biology. Meyer and Liu, Nat Rev Genetics 15: 709 - 721, 2014 Publisher Web Site
Chromatin Immunoprecipitation sequencing: ChIP-seq

- ChIP-seq and Beyond: new and improved methodologies to detect and characterize protein-DNA interactions. Furey, Nat Rev Genet 13: 840–852, 2012 Publisher Web Site

Chromatin conformation

- A decade of 3C technologies: insights into nuclear organization. de Wit & de Laat, Genes & Devel 26: 11-24, 2012 Publisher Website
- Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. Dekker et al, Nature Reviews Genetics 14: 390–403, 2013 Publisher Website

Transcriptome analysis

General considerations for RNA-seq library construction

- Molecular indexing enables quantitative targeted RNA sequencing and reveals poor efficiencies in standard library preparations. Fu et al, PNAS 111:1891–1896, 2014 Publisher Web Site

Assembly and comparison to genome

- A glance at quality score: implication for de novo transcriptome reconstruction of Illumina reads. Mbandi et al., Frontiers in Genetics 2014. Publisher Website
- Comprehensive analysis of RNA-Seq data reveals extensive RNA editing in a human transcriptome. Peng et al, Nature Biotechnology 30:253 - 260, 2012. PubMed [Several comments on this paper question whether the reported differences are in fact evidence of editing or are simply sequencing errors - the authors stand by their conclusions, but the controversy demonstrates the importance of robust data analysis methods.]
- De novo assembly and analysis of RNA-seq data. Robertson et al, Nature Methods 7:909-912, 2010 Full Text [Describes Trans-ABysS, a pipeline to use the ABysS parallel assembler for de novo transcriptome analysis.]

Differential expression analysis

- Robust adjustment of sequence tag abundance. Baumann & Doerge, Bioinformatics 2013 PubMed
Comparing genomes and assemblies; variant detection

- Toward better understanding of artifacts in variant calling from high-coverage samples. Heng Li, Bioinformatics 30, 2843, 2014 PubMedCentral


- Calling SNPs without a reference sequence. Ratan et al, BMC Bioinformatics 11:130, 2010 PubMedCentral


- Detecting and annotating genetic variations using the HugeSeq pipeline. Lam et al, Nature Biotechnology 30:226 - 229, 2012 Publisher Website, Home Page

- Genome-wide LORE1 retrotransposon mutagenesis and high-throughput insertion detection in Lotus japonicus. Ubarashi et al, Plant J 64:731-741, 2012. Publisher Website [This paper describes a 2-dimensional pooling strategy with barcoding to allow use of Illumina sequencing to screen for retrotransposon insertion mutations, and includes a software package called FSTpoolit for analysis of the resulting sequence reads.]


• Diversity Arrays Technology (DArT) and next-generation sequencing combined: genome-wide, high-throughput, highly informative genotyping for molecular breeding of Eucalyptus. Sansaloni et al., BMC Proceedings 5(Suppl 7):P54, 2011 Full Text


Restriction-site Associated DNA (RAD) markers


• Genome evolution and meiotic maps by massively parallel DNA sequencing: spotted gar, an outgroup for the teleost genome duplication. Amores at al, Genetics 188(4):799-808, 2011. PubMed


• RAD tag sequencing as a source of SNP markers in Cynara cardunculus L. Scaglione et al., BMC Genomics 13:3, 2012. Full Text

• Paired-end RAD-seq for de novo assembly and marker design without available reference. Willing et al., Bioinformatics 27(16):2187-93, 2011. Publisher Website


• Stacks: building and genotyping loci de novo from short-read sequences. Catchen et al., G3: Genes, Genomes, Genetics, 1:171-182, 2011. Home Page


• UK RAD Sequencing Wiki page, with bibliography and RADTools software download Home Page

Population Genomics

• PGDspider: an automated data conversion tool for connecting population genetics and genomics programs. Lischer & Excoffier, Bioinformatics 28: 298-299, 2012 Publisher Website
Workspace environments

Papers


Online resources

- The R statistical computing environment includes Bioconductor, a specialized set of tools for analysis of microarray and high-throughput sequencing data. Introductory materials from on-line or short workshops are widely available online; examples are Evomics2012 Bioconductor Tutorial, and Intro to Bioconductor. Materials from an advanced course on high-throughput genetic data analysis are at Seattle 2012 materials. Thomas Girke of UC-Riverside has written a very complete set of manuals describing the use of R and Bioconductor for analysis of genomic datasets, available at R and Bioconductor Manuals.

Manuals and contributed documentation for R are available at the R-project.org website, and video tutorials are also available on Youtube; those posted by Tutorlol are brief, clear, and to the point.

Materials from a series of mini-courses in R taught in 2010 at UCLA are available:

- Intro to programming and graphics
- Data manipulation and functions
- Graphics for exploratory data analysis
- Introductory statistics
- Linear regression

A Little Book of R for Bioinformatics is an on-line resource with information and exercises to provide practice in bioinformatics analysis of DNA sequences and other biological data in R. Many books on specific topics in R programming are also available through Amazon or other vendors.

Cloud computing resources

- The case for cloud computing in genome informatics. Lincoln Stein, Genome Biol. 11(5):207, 2010 Pubmed
- CloudBioLinux is an open-source project that provides a bioinformatics Linux system for cloud computing, pre-configured with a variety of software tools installed and ready to use.
- A tutorial on getting started with CloudBioLinux on the Amazon Web Services Elastic Compute Cloud (EC2)
- Deploying Galaxy on the Cloud slides from a presentation by Enis Afgan (Emory University) at the Bioinformatics Open Source Conference in Boston, July 2010
- A screencast that provides a step-by-step guide to starting a Galaxy cluster in the EC2 environment
- A webpage that has the same information in text form, and is the basis for the screencast
• The iPlant Collaborative, an NSF-funded project to create computational resources for plant biology research, provides access to cloud computing resources through Atmosphere
• SeqWare Query Engine: storing and searching sequence data in the cloud. OConnor et al, BMC Bioinformatics 11(Suppl 12):S2, 2010 Full Text

Links to Linux command-line tutorials and resources

Tutorials for AWK, a powerful tool for handling data tables
• A set of awk notes from Boston University
• Bruce Barnett’s awk tutorial
• Greg Goebel’s awk tutorial
• Executing an awk command from R to simplify data exploratory analysis, from Lex Nederbragt

Tutorials for bash shell scripting
• A tutorial at linuxconfig.org
• A Getting Started With Bash tutorial at hypexr.org
• Mendel Cooper’s Advanced Bash Shell-Scripting Guide

Tutorials for sed, the command-line stream editor + A tutorial at Rutgers + Peter Krumins claims to have the World’s Best Introduction to Sed; take a look and judge for yourself. + Bruce Barnett’s sed tutorial.

Links for Exercise Data

Links to other useful sites

• The SEQanswers online community has forums on several topics related to sequencing; the bioinformatics forum is the most active.
• The SEQanswers Software Wiki is a list of software for analysis of sequencing data
• Biostar is another online community for questions and answers on bioinformatics and computational genomics.
• Information on file formats used by the University of California - Santa Cruz Genome Browser is on the FAQ list
• A manual for the Integrated Genome Browser visualization tool is here
• Course materials for a short course entitled Introduction to R and Bioconductor, held in Seattle in Dec 2010
• Genomic Regions Enrichment of Annotations Tool - A web service to test for over-representation of specific ontology categories among genes near ChIP-seq peaks
• Ben Langmead, author of several tools for sequence analysis, has made course materials for a class in Computational Genomics available on Github.
• An open-source book called Introduction to Applied Bioinformatics has chapters on sequence alignment approaches and algorithms, for those interested in more detail about how that works.
• Next-gen-seq software - a list of software packages, both commercial and open-source, related to analysis of deep sequencing datasets
• Software from the Center for Bioinformatics and Computational Biology, University of Maryland - many useful programs, all open-source
1.1.3 Computing Hardware

**Global overview**

1. Accounts on the HPC are available to those with an NCSU Unity ID - see Get Access for information.
2. Connect to the HPC via login.hpc.ncsu.edu using PuTTY for a command-line interface
3. Create a job submission script file that contains the commands you want to execute
4. Submit the job script to the appropriate queue and wait until the job is complete
5. Transfer the output data back to your office workstation for further analysis, or write another job submission script to carry out more analysis on the HPC.

**Job submission and management**

A tool called Load Sharing Facility (LSF) is used to manage the queues of submitted computing jobs on the HPC cluster. See Running Applications for more information on batch job submission with LSF, and the factors considering in prioritizing submitted jobs in the queue. More information is available at the FAQ for HPC users; additional documentation on LSF commands can also be found via Google searches.

**Available software**

The most current list of software available on the HPC can be obtained by logging into the HPC and listing the contents of the directory /usr/local/apps - not all the listed programs are related to bioinformatics, but this will allow searching for a specific program of interest. As of March 1, 2016, installed software useful for biological research applications included a mix of open-source and commercial tools - examples are shown here, but the complete list is much longer.

<table>
<thead>
<tr>
<th>abyss-1.0.14</th>
<th>asreml2</th>
<th>BEAST.v1.4.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>blas-smrtanalysis</td>
<td>bowtie2</td>
<td>bwa</td>
</tr>
<tr>
<td>circos</td>
<td>cufflinks</td>
<td>cutadapt</td>
</tr>
<tr>
<td>EIGENSOFT-3.0</td>
<td>exonerate</td>
<td>FastQC</td>
</tr>
<tr>
<td>fastx_toolkit</td>
<td>java</td>
<td>maker</td>
</tr>
<tr>
<td>maq</td>
<td>mothur</td>
<td>mpiblast-1.4.0</td>
</tr>
<tr>
<td>mrbayes-3.1.1</td>
<td>mrbayes-3.1.2</td>
<td>MUMmer</td>
</tr>
<tr>
<td>ncbi-blast</td>
<td>ncbi-em64t</td>
<td>ncbi-sra</td>
</tr>
<tr>
<td>oases</td>
<td>octave-2.1.73</td>
<td>paup4b10-x86-linux-icc</td>
</tr>
<tr>
<td>paup4b10-x86-linux-icc.orig</td>
<td>paup4b10-x86-linux</td>
<td>paup4b10</td>
</tr>
<tr>
<td>perl, bioperl</td>
<td>plink</td>
<td>python-2.6.5</td>
</tr>
<tr>
<td>python-2.4.4</td>
<td>R-2.3.1</td>
<td>raxml</td>
</tr>
<tr>
<td>RepeatMasker</td>
<td>samtools</td>
<td>SOAPdenovo2</td>
</tr>
<tr>
<td>sparsehash-1.5.2</td>
<td>spss</td>
<td>SQLite</td>
</tr>
<tr>
<td>sqlite</td>
<td>stacks</td>
<td>tassel</td>
</tr>
<tr>
<td>tophat</td>
<td>trinity</td>
<td>velvet</td>
</tr>
</tbody>
</table>
Setting up a job submission script

• Similar to other shell scripts, batch job scripts start by specifying the shell that will process the script, e.g. 
  `#!/bin/bash` or another shell.

• The next lines specify options for the batch submission process, including details like the number of compute 
  cores required, the amount of time to be allocated to the batch job, the amount of memory needed, the job name, 
  and the output files to which errors and job outputs should be written.

• After all the options are specified, then the code that actually loads software modules and executes the desired 
  programs is included in the script. Depending on the number of different tasks to be executed, there may be 
  a need to load multiple software modules and ensure compatibility among the versions of different software 
  packages to be used in the analysis.

Setting up NCSU AFS and Drive Access

• AFS filespace and NCSU Drive filespace are two different storage options that are available to NCSU campus 
  community members (students, staff, and faculty). Access to these storage volumes is enabled on the VCL 
  instances used in class. Access to AFS filespace is enabled by default and a directory called AFS is created in the 
  home directory of each user; the NCSU Drive space can be mounted by entering the command `mount.mydrive` 
  in a terminal window and entering the appropriate Unity password in response to the prompt, after which the 
  NCSU Drive space is available at the path `/mnt/mydrive`. Both of these storage volumes can also be accessed 
  from Windows or Mac desktop or laptop computers; the OIT webpage linked above has information on how to 
  set up this access. Saving work from the VCL instance to these non-volatile storage options is important if you 
  want keep any files produced during a VCL work session, as all user-created files saved on the virtual machine 
  instance will be lost when the work session is terminated.


1.1.4 Introduction to Linux and the Command-line Interface

Objective

The objective of these class sessions is to introduce participants to the Linux computing environment, with a particular 
focus on the Unix environment provided as a virtual machine through the NC State Virtual Computing Laboratory. 
An introductory lecture on key elements of Linux system architecture and computing philosophy will be followed 
by hands-on computing exercises to provide experience in using command-line utilities to navigate the file system, 
manage files and directories, and carry out basic file processing tasks. Demonstrations of how these command-line 
utilities can be applied to sequence analysis tasks are integrated into the exercises.

Description

Introductory slides provide an introduction to the course objectives and the Linux operating system in the first class ses-

1.1. Course Description 15
Global Overview

Linux is the operating system of choice for computationally-intensive data analysis, because of its design and the efficiency with which it runs. Much open-source software for sequence data analysis is written for Linux, although there are an increasing number of Java-based programs that can run under Windows. A key element of the philosophy behind Unix and Linux operating systems is decomposition of tasks into simple categories – separate command-line utilities are available for separate tasks. Combining these simple individual tools into pipelines provides enormous flexibility for managing and processing data. Abstraction is another key concept in computing - generalizing from a specific case to a larger group of cases that all meet a specific set of criteria. The use of “wildcard” or meta-characters to specify groups of files is a simple example; this process is commonly called file globbing. Regular expressions are another powerful example of abstraction and generalization as key parts of computational thinking.

Key Facts

DNA sequence data and most results of analysis are stored in plain text format, often compressed using an open-source algorithm to reduce the size of the files stored on disk. A few dozen commands for manipulation of text files, executed either separately or in different combinations, provide an enormous range of data manipulation and analysis capabilities. A modest investment of time in learning basic file formats and commands for text file manipulation will pay large returns by enabling you to manage large data files and carry out basic analyses on the command line, without any specialized software for sequence analysis.

Exercises

1. An Introduction to Linux is a tutorial to guide participants through an 8-step introduction to the Linux operating system and virtual computing lab (VCL) access used for most class computing exercises. As an initial exercise before the tutorial, we’ll use the commands in fastq-dump.exercise.sh to explore some features of Linux and the virtual machine available through the VCL. Open the Intro to Linux PDF file and follow the directions to connect to a VCL instance, then open a browser in the VCL instance, navigate back to this page, and download the fastq-dump.exercise.sh file to the instance. Follow the directions in that file to compare the relative speed and resource requirements for different methods of achieving the same result.

2. A list of useful Linux commands is available as a handy reference.

3. The example files for the week1 quiz are at quiz_week1.tgz or quiz_week1_download_script.


5. A quick Quiz to gauge linux command line proficiency.

Additional Resources

Background information about Linux:

- A reminiscence called The Strange Birth and Long Life of Unix by Warren Toomey in 2011 commemorated the 40th anniversary of the beginning of Unix (and therefore, Linux) development.

- The Software Carpentry website has a series of tutorials with introductions to many aspects of Linux computing. The lessons entitled The Unix Shell and Programming with R are particularly relevant to this course, because we use the shell a lot throughout the course, and R is important in the section on transcriptome analysis.

- Analysis of Next-Generation Sequencing Data workshops (ANGUS) have been taught at Michigan State in the past, and in 2017 moved to UC-Davis. Course materials are available online. The class uses cloud computing instances that are configured differently, so the exercises won’t necessarily work exactly the same on the
VCL machine image we are using, but the course materials do contain useful information about bioinformatics applications and tools.

• More information on regular expressions is available at A Brief Introduction to Regular Expressions at The Linux Documentation Project webpage.

• The FileGlobbing.pdf and RegularExpressions.pdf documents also provide more information on these pattern-matching tools.

• The LocaleSettingDetails.pdf document covers localization options in UNIX, including the ‘C’ locale, and how it may affect alphabetical processes.

• One aspect of command-line use is knowing when to use a particular command, and when it is not needed. Many command-line utilities such as grep, cut, wc, sort, sed, and awk (among many others) accept filenames as arguments after the command, but will also accept input from stdin via a pipe. Other utilities, such as tr, do not accept a filename as an argument and only process data received from stdin. Some people prefer to use the cat command to put data into a pipeline, even when the command being used could read the filename as an argument, simply for the sake of consistency and style (see this StackOverflow discussion as an example), while purists argue that using a command when it is not required means running two processes when one will do. This is rarely a problem, but can lead to differences in the commands used to accomplish the same result. For example, in chapter 5 of the Biostar Handbook called Ontologies, the section called Understand the GO data includes some manipulations of a file called goa_human.gaf.gz, after downloading the file from the geneontology.org website.

```bash
# Uncompress the GO file:
gunzip goa_human.gaf.gz

# Remove the lines starting with ! and simplify the file name:
cat goa_human.gaf grep -v '!!' > assoc.txt

# The same result could be obtained in a single step using this alternative
gunzip goa_human.gaf.gz | grep -v '!!' > assoc.txt
```

Windows Subsystem for Linux in Windows 10

• Windows 10 offers an optional beta-release of Windows Subsystem for Linux, which allows running any of three different Linux-like command-line environments on Windows, although the Linux kernel itself is not installed. These provide a command-line bash shell environment with GNU utilities - see a tutorial on set-up or a Microsoft page.

Setting up an Amazon Web Service account to use Elastic Compute Cloud services:

• A 2013 guide to setting up an Amazon Web Services account is available for those interested in using cloud-based computing resources, and a 2013 guide to preparing and running a Cloudbiolinux instance on the Amazon Web Services Elastic Compute Cloud (AWS-EC2), is also available. The BIT815 course no longer uses AWS resources, so these documents have not been updated to reflect any recent changes in AWS procedures – users are cautioned to follow the instructions on the AWS website rather than those in these documents in case of any conflict.

Class Recordings

• Session 1: recorded January 20th 2021
• Session 2: recorded January 22th 2021. Audio only
1.1.5 Experimental Design

Global Overview

There are two kinds of sequencing experiments - those where the sequence itself is the objective (e.g. for genome or transcriptome assembly), and those where the sequence reads are merely a way of counting some other kind of phenomenon. Examples of the latter type of experiment include differential gene expression analysis (counting frequencies of transcripts), chromatin immunoprecipitation (counting frequencies of protein-DNA interactions), or chromatin accessibility (identifying differences in chromatin structure in intact nuclei). The design of a sequencing experiment should be based on the experimental objectives and analytical procedures, and consider all the potential sources of variation (technical, biological, and experimental) that may contribute to the variation in the final dataset. Data processing should evaluate the quality of the sequence reads in the context of the experimental design, and remove data that falls below an objective threshold of satisfactory quality. That threshold will be different for different experimental designs, so quality control measures must be based on the experimental design and experimental objectives.

Objective

Discuss sources of technical and biological variation in high-throughput DNA sequencing experiments, along with strategies for experimental designs that maximize the value of the resulting data for addressing the experimental question of interest. The issues in experimental design for sequencing experiments are not fundamentally different than those involved in experimental design in other contexts, such as agricultural field tests, and they are important for the same reasons. Poorly-designed sequencing experiments may yield data poorly suited for testing the intended hypothesis, reducing the overall value of the outcome.

Description

A central concern in many statistical analyses is evaluating the amount of variation within a group or category relative to the variation among groups or categories. The ability to identify sources of variation, and classify each source of variation either as within-group or among-group, is therefore important in the process of data analysis. Good experimental design is critical to maximize this ability to identify sources of variation. Each method for highly-parallel DNA sequencing has its own characteristics in terms of the type and frequency of sequencing errors observed. A good experimental design is based on the recognition that both biological and technical sources contribute to the variation observed in experimental results, and includes appropriate randomization and replication strategies to maximize the power of the experiment to detect differences in the variables of experimental interest. An example analysis of experimental sources of variation is shown below: Fig 3 from Reeb and Steibel (2013), which shows a multi-dimensional scaling analysis of two RNA-seq experiments with contrasting levels of biological and technical variation.
Plot (A) on the left shows combined technical and biological variation among unreplicated RNA-seq data of cell cultures from different male and female donors. Biological and technical variation are confounded, because only a single dataset was collected per cell culture. The plot on the right shows biological replicates of two inbred mouse strains called B6 and D2. The major clusters separated along Dimension 1 (the x-axis) are the two strains, while variation in Dimension 2 is technical variation among individual samples, RNA extractions, library preparations, and flowcell differences. The colors of the letters show which samples were run on the same flowcells. It is clear from this panel that differences among flowcells account for some but not all the differences observed among biological replicates of the same mouse strain. The libraries were not individually-indexed and pooled, so variation among individuals, RNA preps, and libraries is confounded with variation among lanes within flowcell and variation among different flowcells.

**Key Facts**

Indexing or barcoding strategies allow sequencing experiments to be carried out on a mixture of libraries from different experimental treatments. This allows use of orthogonal experimental designs that avoid confounding technical sources of variation with experimental sources of variation (Auer and Doerge, 2010). The experimental treatments or factors controlled by the experimenter can be randomly allocated to different sequencing runs, including different sequencing platforms if desired, if the experiment is large enough to require collection of data from more than a single experimental unit. Technical sources of variation need not be orthogonal to each other in the experimental design, if variation from technical sources is simply a nuisance factor in the analysis. If it is important to know the exact source of technical variation, such as whether poor data are due to a failure of library construction or to a failure of the sequencing service center, then keeping those factors from being confounded is valuable.

Preliminary trials using simulated data sets, or “plasmode” datasets derived from real data but with specific effects added, can be a powerful tool to guide experimental design (Reeb and Steibel, 2013). The ideal situation is to find an existing dataset in a public database that is derived from an experimental system similar to the intended experiment, so that the correlation structure and sources of technical and biological variation are likely to be similar to those encountered in the data to be collected. This is not always feasible, particularly for researchers working on experimental questions that have not yet been widely explored using highly-parallel sequencing as an experimental method.
An experimental design for any highly-parallel sequencing experiment needs to take into consideration the experimential objectives, the availability of biological material, and potential technical and biological sources of variation. The question of whether individual samples should be analyzed multiple times (technical replicates), or if the same amount of sequencing is better allocated to increasing the number of independent biological samples analyzed (biological replicates), depends in part on the experimental objectives and in part on the relative amount of variation expected to come from the different sources of variation. For example, experienced technical staff can typically produce very consistent libraries given adequate amounts of high-quality RNA, so in general there is little benefit to having multiple libraries made from a single RNA sample, because library preparation is not a major source of variation. As a general guide, it is usually more informative to have more biological replicates than to acquire the same amount of sequence data from more technical replicates, but this can vary depending on experimental priorities (Robles et al, 2012). Yang et al (2014) also report that more biological replicates is better - two is the minimum number, but two replicates do not provide all the benefit that can be obtained from three or four replicates.

**Exercises**

- Develop an experimental design for an RNA-seq experiment that involves comparison of samples treated in two different ways. Suppose that the treatment is time-consuming enough that only two experimental units can be processed per day, and that technical variation in the treatment from one day to the next is unavoidable. What experimental design would avoid confounding this technical variation with the biological variation that is the topic of interest in the experiment?

- If the experimental design were to be extended so that biological responses to the two treatments are to be compared across multiple genetic backgrounds, what would the best approach be to incorporate two different genetic entries into the experiment?

**Additional Resources**


### 1.1.6 Data Preprocessing and Quality Control

**Global Overview**

Data from high-throughput DNA sequencing platforms (such Illumina, Ion Torrent, Pacific Biosciences, or Oxford Nanopore) can contain a variety of experimental artifacts and low-quality data, so quality control and data preprocessing are always a good idea. Common artifacts in sequences from Illumina and Ion Torrent instruments include copies of the adapter sequences, because these instruments use solid-phase PCR to amplify the template DNAs using
in the sequencing reaction. FASTQ-format data contains both the DNA sequence data and an estimate of the probability of error for each base in the sequence; removing sequences (or regions of sequences) consisting of low-quality base calls is another common pre-processing step. The stringency of such quality-filtering or trimming steps can affect the results of downstream data analyses, and the optimal level of stringency for filtering or trimming may be different for different downstream analyses. For example, RNA-seq analysis of differential gene expression can be biased by too-stringent quality trimming (MacManes, 2014; Williams et al, 2016).

**Objective**

The objective of this session is to provide participants with experience in managing and processing DNA sequence data. A review of FASTQ format will be followed by practical exercises in data quality analysis with FastQC and data filtering with tools from the BBTools suite.

**Description**

The most common format for sequence data output from high-throughput instruments is FASTQ format (Cock et al., 2010), containing sequences coupled with their quality stored as text characters. The quality scores are -10 times the logarithm (to base 10) of the per-base probability of an incorrect nucleotide call at each position in the sequence, and are based on the PHRED quality score (Ewing and Green, 1998; Ewing et al., 1998). In the past, some instruments produced separate files containing DNA sequences (in Fasta format) and quality scores (*.qual files), so you may encounter these file formats as well. The numerical quality scores (typically ranging from around zero to around 40) are converted into single characters to save space. A two-digit quality score must be followed by a space to distinguish it from the score for the next nucleotide, so one value requires three characters, while a single text character contains the same information and need not be separated by a space from the next single character.

Unfortunately, multiple different conventions have been used for encoding quality data in text characters using the American Standard Code for Information Interchange (ASCII), and knowing which convention was used for any given dataset is important. The Wikipedia page on FASTQ format is the most up-to-date source of information about the different encoding schemes used in the past and the current encoding used by Illumina instruments (called Illumina 1.8+ in the figure).

Several tools for summarizing the frequency and identity of contaminating adapter sequences and the distribution of quality scores per base or per sequence are available; we will use the Java program FastQC for data quality summaries. Many different software tools exist for removing low-quality base calls from the 5′ end or 3′ ends of sequence reads, for trimming off sequencing adaptor sequences or other extraneous sequences, and for separating reads from a single dataset into multiple datasets based on the presence of “barcode” sequences at the beginning of the sequence reads. We will use both fastx-toolkit (a simple, lightweight but limited toolset) and BBTools (a Java-based and extremely comprehensive package of programs) for trimming and filtering reads.

**Key Facts**

DNA sequence data as produced from the instrument can vary in quality and read length, and some exploratory analysis and filtering of the data is essential before beginning the process of analyzing the sequence to achieve experimental objectives. Understanding the format in which sequence data are provided is essential, as there are (unfortunately) multiple versions of the “standard” FASTQ format in use. Converting FASTQ format to FASTA is simply a matter of stripping out the quality data and changing the character at the beginning of the sequence header line, but converting FASTA and quality score files into a single FASTQ format file requires converting numerical quality scores into characters using the ASCII lookup table.
Exercises

1. Sequence format conversion tools. The key difference between FASTA and FASTQ format is that FASTQ contains quality scores. If you have a FASTA file containing sequences, and no accompanying file of quality scores, there is no meaningful way to provide real quality scores for creation of a FASTQ file. If you have a separate .qual file of quality scores, you can use various tools to merge the FASTA sequences with the quality scores to produce a FASTQ format file. Search online for a user guide to the BBTools package of programs, which is installed on the VCL machine image and has a reformat.sh script that can perform this conversion as well as many others. Read through the help information to learn about the capabilities of the BBTools package of programs, or see Bushnell et al. (2017) for more details on the BBMerge tool from this package that can merge paired-end reads into longer single sequences.

2. Software: fastx-toolkit and FastQC. FastQC is a Java program that will run on Windows, Mac, or Linux, and is already installed on the VCL machine image - the link here is provided for those interested in installing the software on another machine. The fastx-toolkit package of programs is available in the Ubuntu repository, and is also already installed on the machine image. These programs differ in that FastQC has a nice graphic interface and produces pretty graphs describing various aspects of data quality, but has no capability to filter, trim, or otherwise modify sequence files to address problems made apparent in those pretty graphs. The fastx-toolkit package can also produce graphs, although they aren’t quite as pretty, but it does include programs capable of filtering and trimming sequence data files.

3. Using fastx-toolkit for exploratory analysis. The first exercise will use fastx-toolkit to analyze and process the c3.fq.gz sequence file, also found in the fullset.zip archive (Both files can be directly downloaded via console using the following code). The fastx-toolkit programs are simple and somewhat dated, but still useful. Download the QCexercises.sh shell script with exercises.

4. Run the script using the command bash QCexercises.sh from a terminal prompt. The shell script should create two image files (with a .png extension) in the directory where you run it - use one of the image viewer programs available under Graphics in the Application menu of the Linux system to view these image files, or just use the display command at a terminal prompt.

5. Open the QCexercises.sh file with the Geany programming text editor (listed under the Development item in the Applications menu) and read the code and comments to learn what the script does. You can run the script multiple times with different input sequence files, changing the name of the file to be used as input to the series of programs, or you can simply copy the commands from the script file to a terminal window and run each command from the command line, one step at a time.

6. Run FastQC on the same file, and compare the results. FastQC produces a report containing an HTML file that can be opened using a browser. By default FastQC tries to write the output to a directory in the same place where the sequence file is found - you must have write permission in that directory. You can specify an alternative place to save the output using the -o option on the command line.

7. Using fastx-toolkit to remove adaptor sequences and trim off low-quality bases. Find the web page with instructions on the individual programs in the fastx-toolkit package (remember, the information is out there - you just have to find it). Use the fastx_clipper and fastq_quality_filter programs to remove any copies of the Illumina sequencing adaptor (GATCAGAGACCTATGACGCCCTTCTGCTTGAAA) and to trim low-quality bases so that 80% of the bases in each read have a quality score of 15 or higher. Note that the fastx-toolkit programs accept input from STDIN and deliver output to STDOUT by default, so they are designed to be linked together in pipelines for processing data without the need to save intermediate files. See the QCexercises.sh script file
for an example of how to format the command to send decompressed fastq-format sequence to fastx-toolkit commands.

**NOTE: the -Q33 option is required** - fastx-toolkit by default assumes that fastq-format sequences have quality scores from an older version of the Illumina base-calling software, and if you don’t tell it to use the new version, it will return an error message.

8. Using BBTools programs to remove adaptor sequences and trim low-quality bases. The BBTools programs are installed in the /usr/local/bbmap directory of the Linux system, and this directory has been included in the $PATH environment variable, so you can run them by typing the name of the command at a terminal prompt, for example bbduk.sh to run the bbduk.sh program. Executing this command with no arguments will print a user guide for the command to the terminal screen, so this is one way to learn what options and arguments each command accepts. A web search will lead you to a BBTools User Guide at the DOE Joint Genome Institute, because the author (Brian Bushnell) is a bioinformatics specialist at JGI. NOTE: many of the BBTools programs are Java-based, so they can be used on any operating system that has Java installed, but you can read the user guides for all the commands without installing Java. By default, the bbduk.sh and bbduk2.sh programs do not use the same sliding window approach for quality trimming as does fastq_quality_filter, but setting the appropriate options during execution of either bbduk.sh or bbduk2.sh will allow that approach to be used. For more information about alternative ways of quality trimming, see this SeqAnswers Forum thread, and look for post #134.

9. Summarizing sequence data characteristics using FastQC. You can run FastQC either from the command line, providing the names of sequence files to be processed as arguments, or from a graphic user interface. Typing the command `fastqc` without providing an input filename will start the program in interactive mode, where you choose which file to analyze from the File menu, while providing a file “glob” using wildcard characters will run the program on every sequence file that matches the filename pattern from fullset.zip, e.g.

```
fastqc /fullset/[ct][123].fq.gz
```

Note that the FastQC program can process gzip-compressed sequence files without saving an uncompressed version - this is important for saving disk space when hundreds of gigabytes of compressed sequence files need to be processed.

### Additional Resources

- Wikipedia has information on FASTA and FASTQ sequence formats.
- The University of California - Santa Cruz Genome Browser site maintains a FAQ with information about many different file formats used in analysis of deep sequencing data.
- The fastx-toolkit webpage has information about the fastx-toolkit package of programs for quality control and manipulation of FASTA and FASTQ files.
- The FastQC webpage has information about the FastQC program, and details on FastQC output are provided in the **FastQC_details.pdf** document.
- Another program suitable for adapter trimming is called “flexbar” - this program can also split reads into different files based on the presence of specific “barcode” sequences detected in the sequence reads. Such barcodes are common in GBS and RAD-seq applications, and the ability to detect variable-length barcodes is somewhat unusual. The manual for flexbar is on Sourceforge, and the publication describing the software is also available.
- The BBtools suite of programs was announced on the SeqAnswers forum, and the correspondence between the program developer and users is archived as a resource for others to learn how to use the various tools in the suite. The announcements and correspondence are in separate threads for individual programs; the list of tagged posts can be viewed to see links to the individual threads. The software is available at the Sourceforge project page.
tools for managing the process of data QC and format conversion, including tools for filtering datasets of paired-end reads to find single reads where the paired-end read was removed by a quality-filtering step*).

- Cock PJ, Fields CJ, Goto N, Heuer ML, and Rice PM. (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res. 38: 1767–1771. PMID 20015970 (*Note: This is the only formal publication I know of that describes the different versions of the FASTQ sequence format, and it is not as up-to-date as the Wikipedia page on FASTQ format*).


- A sequencing-focused publication/news aggregate blog, QCfail.


### 1.1.7 Error Correction and Alignment

#### Global Overview

Sequences randomly sampled from genomic DNA can be assumed to be drawn from a uniform distribution across all possible sequences in the genome, although this assumption is typically violated at least to some degree, even with PCR-free libraries (Kozarewa et al, 2009). Based on that assumption, however, it is possible to analyze the distribution of frequencies of k-mers (short oligonucleotides, typically in the range from 15 to 31 bases) observed in sequence reads. If the average coverage of the genome is 50x, one expects to see most k-mers drawn from single-copy sequences in the genome around 50 times, with a sampling distribution that extends above and below that expected value. Errors in sequencing reads give rise to novel k-mers that typically appear much less frequently than the correct k-mer sequences, while k-mers drawn from sequences that are repeated in the genome (such as transposable elements) appear much more frequently than the single-copy k-mers. These differences can be used to filter out error k-mers and selectively remove reads containing sequencing errors from the dataset. The number of different k-mers detected, and the characteristics of the frequency distribution of k-mers, can be used to estimate the size of the genome and the content of repetitive DNA sequences (Li and Waterman, 2003).

The underlying assumption that all k-mers are sampled from a uniform distribution is grossly violated in RNA-seq data and some other data types, so k-mer counting for purposes of error correction requires modeling the frequency distributions of k-mers independently for transcripts of different abundance classes (Song and Florea, 2015). More sophisticated models can take base quality scores or k-mer abundance values into account, along with the connectivity of k-mers in the De Bruijn graph used for assembly (Durai and Schulz, 2019), and can combine normalization (reducing the computational load for transcriptome assembly) with error correction.

#### Objective

The objective of this session is to introduce concepts behind error correction algorithms, and to provide participants experience in simulating short-read sequence data, using error-correction programs, and aligning short read sequence data to a reference genome. The resulting alignment files will be compared to determine the effects of different error models during simulation, the value of error correction, and the outcomes of different alignment programs.

#### Description

Ultra-high-throughput DNA sequencing platforms typically have much higher error rates than Sanger sequencing, and the sequence variation introduced by sequencing errors must be taken into consideration in downstream analysis
of the DNA sequence data. Various approaches to error correction are possible, but each has disadvantages - some are too computationally-intensive for application to anything larger than a microbial genome, some make unjustified assumptions about the distribution of read coverage across the sequenced DNA template, and some lack sensitivity or specificity to resolve sequencing errors from true variants. Empirical datasets can be used to create statistical models of sequencing errors, and those error models can then be used to simulate sequencing data from a reference sequence for use in comparative analysis, so the results can be compared back to the starting genome to determine how well the error correction algorithm worked. It has been noted (Heydari et al, 2017) that the true test of how well error correction works is the quality of the resulting assembly, rather than the difference in the number of mismatches noted after alignment of uncorrected or corrected reads to an existing assembly, but this is a much more time-consuming test of error-correction methods. The same authors developed an error correction program aimed at improving the outcome of genome assembly by focusing specifically on read pairs near regions of repetitive sequence, and report improved assemblies for six eukaryotic genomes relative to those obtained from uncorrected reads (Heydari et al, 2019).

Key Facts

Simulation is an important tool for development of new software and comparison of available software tools for specific purposes. The assumptions made in creating simulated datasets often determine the relative performance of different analytical approaches, so it is important to know what assumptions are made during simulation and how realistic those assumptions are for real datasets. In the exercises below, simulated Illumina paired-end reads created from a reference bacterial genome (Lactobacillus helveticus strain DPC4571) are used. The simulated read files (sim.r1.fq.gz and sim.r2.fq.gz) and the reference bacterial genome file are provided in the DPC_4571 archive. GemReads.py (a Python script from the GemSIM package) was used to create the simulated Illumina reads, and this package is installed in the VCL system, so you can create your own simulated datasets from the reference genome, but this is too time-consuming to do in class. GemReads.py does not accept gzipped files as input, so you will have to unpack the compressed genome sequence file.

It is important to note that while we use simulation as a tool, it is not the primary focus of the exercise. Instead, the focus of the exercise is on the use of error correction software, and what the effects of using error-correction software may be on the outcome of the overall experiment.

Exercises

The text file error_correction_files.txt contains a list of commands for the direct download of files necessary for the following exercises. These commands are given for the benefit of those who use an SSH connection to a compute node or virtual machine instance, and don’t have a web-browser interface available from which to download the datafiles to the instance. For users who have access to a virtual instance with a graphic interface, starting a web browser and downloading each file directly from the link is probably easiest.

1. The first exercise will align RNA-seq reads to a bacterial genome, to provide some experience with alignment software and an opportunity to explore software tools used to summarize and manipulate Sequence Alignment and Mapping (SAM) format files. The RNA-seq reads come from Lactobacillus helveticus strain CNRZ32, while the reference genome is from strain DPC 4571, so some sequence differences detected in the alignments will be due to the strain divergence, and some differences due to sequencing errors or other sources of experimental noise in the RNA-seq data. You can also download a text file of steps to use in aligning the reads to the reference genome, then reviewing the results of the alignment.

2. The MaSuRCA (Maryland SuperRead - Celera Assembler) program is installed on the VCL machine image. This program uses k-mers detected in filtered and trimmed fastq sequence reads to expand typical paired-end reads from an Illumina sequencing instrument into what it calls “super-reads”. Download the K-merCounting_ErrorCorrection.sh shell script, open it with the Geany or SciTE text editor (in the Applications menu under Development), and review the commands and comments in the script file. These commands show how to use a simulation program called GemReads.py to simulate short sequence reads from the reference bacterial genome sequence. Type GemReads.py -h at a terminal prompt to get a list of options used to specify parameters of the simulation. The reference bacterial genome is 2.1 Mb - how many paired-end 100-nt Illumina reads would be
required to reach average nucleotide coverage of 50x? What nucleotide coverage would be provided by 300,000 pairs of 100-nt reads?

3. The MaSuRCA installation also installed the Jellyfish k-mer counting program, and the Quorum error correction program as part of the MaSuRCA package. Use the Jellyfish k-mer counting program to produce a file with frequency data kmers of length 20, as outlined in the K-merCounting_ErrorCorrection.sh script, then use the plot_histo.r script to produce a PNG image file with a plot of the frequency distribution.

4. Use the Quorum error correction program to correct errors in the simulated sequence data. Type the full path to the /usr/local/masurca/bin/quorum program, followed by the -h option, at a terminal prompt to get help on the correct syntax to use the program. Run Quorum to correct the sequence reads, and save the corrected reads to new files.

5. Use the BWA or Bowtie2 alignment programs to align the uncorrected and corrected sequence reads to the reference genome. Manuals for these two programs are available on Sourceforge - follow the links on the program names - and both programs are already installed on the VCL system.

6. Summarize the resulting SAM output files using the command-line tools grep, awk, cut, sort, and uniq, as described in SAMformatAndCLtools.pdf

7. For extra practice working with SAM alignment files, download the smallfiles.zip archive into your working directory and unpack the archive with the command unzip smallfiles.zip Use the command-line tools grep, awk, cut, sort, and uniq, as described in the SAMformatAndCLtools.pdf document, to analyze the smallRNA-seq.sam file of read alignments. The same types of analyses can be carried out on the sampleReadsSAM.tgz file.

Additional Resources

- McElroy KE, Luciani F, Thomas T. (2012) GemSIM: general, error-model based simulator of next-generation sequencing data. BMC Genomics 13: 74. PMID 22336055 (Note: This paper describes software for simulation of sequence data that is useful for testing effects of error frequency on alignment and assembly).
- Hatem A, Bozdag D, Toland AE, Çatalyürek ÜV. 2013. Benchmarking short sequence mapping tools. BMC Bioinformatics 14:184. PMID 23758764 (A publication comparing eight different open-source or proprietary read-alignment programs on simulated and real data, including BWA and Bowtie2. The conclusion was that no single tool is optimal for every purpose or any dataset; the user must make an informed decision based on experimental system and objectives)

Multiple options are available for reconstruction of the sequences of RNA transcripts based on analysis of complement-
imentary DNA copies, depending on whether a high-quality reference genome assembly is available. If so, reference-
guided transcriptome assembly using a splice-aware sequencer aligner followed by resolution of alternative splicing
variants and export of consensus transcript sequences can be powerful (Trapnell et al., 2012). If there is no reference
genome assembly, or the available assembly is fragmented and poorly-annotated, a de-novo assembly of putative tran-
scripts from short-read sequences is one alternative. The availability of long-read sequencing methods such as Oxford
Nanopore Technologies and Pacific Biosciences has opened a third alternative, which is to obtain full-length sequences
of cDNAs and report those sequences without the requirement for assembly at all (Sharon et al., 2013; Bolisetty et al.,
2015; Minoche et al., 2015).

Objective

The objective of this session is to introduce participants to methods for assembly of a transcriptome, or a collection
of RNA transcripts present in a given RNA sample. The RNA-seq data used in the differential gene analysis exercise
will be used for this exercise as well, as those datasets are small enough to run on systems with only 8 Gb RAM.

Description

De-novo assembly of short DNA sequence reads into contiguous full-length copies of RNA transcripts is a complex
process, because there are multiple sources of variation in read coverage and in read sequences. Read coverage
is proportional to transcript abundance, but can also be affected by biases due to library construction methods and
biological factors. A key factor leading to variation in transcript sequences, and therefore variation in read sequences,
is alternative splicing of transcripts from the same transcription unit into different mature forms. Another factor
affecting read sequences is the increased error rate of RNA polymerases relative to DNA polymerases - RNA sequences
are inherently noisier than DNA sequences. Trinity (Grabherr et al, 2011) is a widely-used suite of programs for de-
ovo assembly of RNA-seq data; the Rockhopper program for assembly of bacterial RNA-seq data is described in a
recent report (Tjaden, 2015). One key difference between these two programs is the amount of memory required - the
recommended amount of RAM for Trinity is 1 Gb RAM per million reads in the dataset, while Rockhopper can carry
out assembly of datasets with hundreds of millions of reads using 2 Gb of RAM.

Construction of a de Bruijn graph is one approach used to solve the computational problem of assembling short DNA
sequence reads into accurate contigs that reflect the transcripts present in the original RNA sample used for library
construction. A key difference between Trinity and Rockhopper in the amount of memory required is that Rockhopper
builds an index of sequence contigs as they are assembled, so that the entire de Bruijn graph of all k-mers present in the
full set of reads does not have to be held in memory for the entire process of assembly. Slides from Lilian Matallana’s
lecture on de Bruijn graphs and their use in sequence assembly programs describe these different strategies in more
detail.

Key Facts

Paired-end sequencing reads are useful for assembly of eukaryotic transcriptomes, because the information they pro-
vide about the positions of sequences relative to each other within a transcript is valuable for correct assembly of
alternatively-spliced transcripts. In bacteria, splicing is less frequent, alternative splicing can be less of a concern,
and single-end reads can often be assembled into a reasonably accurate transcriptome. If alternative splicing is not
of interest, then single-end reads are useful for RNA-seq analysis of eukaryotic transcriptomes as well - now that
read-lengths of 100 to 150 nt are readily available, paired-end reads add little additional information regarding levels
of gene expression.
Exercise - reference-guided assembly

Direct download links for class can be found in the Transcriptome_Assembly.txt file.

- The Arabidopsis thaliana RNA-seq dataset used for the analysis of differential gene expression includes six sequence files found in the fullset.zip archive in the AtRNAseq archive.

- The HiSat2 aligner is already installed on the VCL system. If you want to install it on another computer, an executable binary can be downloaded from the link under the Releases heading on the right side of the program home page. This program is the successor to Bowtie and Tophat, the original programs developed for reference-guided sequence assembly.

- Create a folder for this exercise, and unpack the Atchromo5.fasta.gz reference sequence into that directory. Use the hisat2-build program (from the hisat2-2.0.5 directory) to build an alignment index from the uncompressed fasta-format sequence data.

- Unpack the three control-sample RNA-seq read files (named c1, c2 and c3) from the fullset.zip archive to a single fastq file, using the gzip -cd command with file globs.

- Align the fastq-format sequence reads to the reference sequence using the hisat2 program (using the –dta option), and pipe the output to samtools view to convert the SAM output into BAM, then to samtools sort to sort the output BAM data and save it to a file.

- Stringtie is already installed on VCL instances. If you are not working from the VCL you can download and install the Linux x86_64 binary version of the StringTie program from the link under the Obtaining and installing StringTie heading on the program home page.

- Execute the stringtie program using the sorted BAM file as input.

- Notes from class for reference guided assembly.

Exercise - de-novo assembly

- The Arabidopsis thaliana RNA-seq dataset can also be used for de-novo assembly, although RAM is a limiting factor on instances of the VCL Biostar_DNASeq machine image. The Trinity assembler archive can be downloaded from the project Github site. To compile, unpack the archive using the command tar -xzf trinityrnaseq-v2.9.1.FULL.tar.gz, then change to the trinityrnaseq-v2.9.1 directory and execute the make command to compile the program. Trinity requires Bowtie2, SAMtools, and Salmon, which should all be available on the VCL image.

- Rockhopper can be downloaded to the home directory of a VCL instance and run from the command line - for some reason the GUI version would not save the file of assembled transcripts when I tested it. All six files of RNA-seq data are from the same accession of Arabidopsis, so they can all be concatenated into a single file and provided as input to Rockhopper.

- A file of Arabidopsis thaliana RNA sequences (inferred from gene models in the TAIR 10 genome assembly: TAIR10.cDNA.fa.gz) is also available. The assembled transcripts can be compared with these predicted transcripts as a means of evaluating how good a job the Rockhopper assembler (which is designed for assembly of bacterial RNA-seq datasets) does with the plant RNA-seq data.

Additional Resources

Several papers have reported that the most reliable approach for transcriptome assembly for different organisms is to use multiple different programs for independent assemblies, followed by merging together of the resulting assembled contigs and selection of the most complete contigs as representatives for the final completed transcriptome.

• McManes, M.D. 2018 The Oyster River Protocol: a multi-assembler and kmer approach for de-novo transcriptome assembly. Peer J. 6:e5428. Full text This paper describes a set of criteria used to evaluate the relative quality of different transcriptome assemblies, using the software tools BUSCO, shmlast, Detonate, and TransRate.


• Gilbert, Donald 2013 Gene-omes built from mRNA seq not genome DNA. 7th annual arthropod genomics symposium. Notre Dame, Indiana. Poster

Correction of errors in RNA-seq reads requires consideration of the difference in relative abundance among transcripts in order to identify likely error-derived k-mers. Rcorrector is one software package capable of this process; the SEECER package described by Le et al (2013) is another.


• One strategy for reducing the amount of RAM required for transcriptome assembly by the Trinity software package is to carry out “digital normalization” of the RNA-seq dataset - this means adjusting the numbers of reads in the dataset to ensure more uniform representation of both abundant and rare transcripts, while removing sequencing errors. A detailed exercise is available, which uses AWS cloud computing instances to provide sufficient computing power to process a real dataset.

• Analysis of Next Generation Sequencing data (ANGUS) is a workshop series on high-throughput sequence data analysis; the 2017 workshop includes an exercise on transcriptome assembly with Trinity using cloud computing resources.

• Rana et al., 2016. Comparison of de-novo transcriptome assemblers and k-mer strategies using the killifish, Fundulus heteroclitus. PLoS One 11: e0153104. Full text


• Tjaden, B. (2015) De novo genome assembly of bacterial transcriptomes from RNA-seq data. Genome Biology 16:1 Full text


• Salzberg S, et al. (2012) GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Research 22:557–567. PubMedCentral This paper describes a set of experiments comparing different assembly programs on four genomes, and provides useful insights into the challenges of genome assembly.

• Magoc T and Salzberg S. (2011) FLASH: Fast Length Adjustment of Short Reads to improve genome assemblies. Bioinformatics 27:2957–2963. PubMedCentral This paper describes a software tool for joining paired-end reads obtained from DNA fragments short enough that the reads overlap at the ends. This is reported to improve the quality of assemblies created from the joined reads. An outline of an exercise with the FLASH assembler is available: FLASH_exercise.docx

1.1.9 Genome Sequencing and Assembly

Global Overview

De-novo assembly of a complete and accurate genome sequence from high-throughput sequencing data is a challenging computational problem for any genome that contains repeated DNA sequences longer than the typical sequencing read length (Simpson & Pop, 2015). The single-copy regions between repeated sequences can be assembled into contiguous stretches of sequences, or “contigs”. Construction and sequencing of “mate-paired” sequencing libraries allow retrieval of DNA sequences known to occur at a specific spacing in the genome, which can range from 2 kb to 35 or 40 kb. These data allow linking of contigs into larger “scaffolds” that contain gaps where the exact sequence is unknown but the approximate size is known. “Draft” genome assemblies often consist of many such scaffolds, and the unknown sequences in the gaps within scaffolds can represent half or more of the total length of the scaffolds. The process of “finishing” a genome sequence involves collecting sequence data to fill the gaps in scaffolds and ensure that all contigs are placed in the correct order and orientation relative to each other to form a complete continuous sequence of each chromosome of the genome of interest. This process is difficult and expensive, so few eukaryotic genomes larger than those of fungi are completely finished. Long-read sequencing methods, such as Pacific Biosciences and Oxford Nanopore Technologies single-molecule sequencing, provide reads long enough to bridge across many repetitive regions in eukaryotic genomes, and such long-read data have been used to improve existing assemblies of mammalian genomes (e.g. the gorilla genome, Gordon et al, 2016) and carry out de-novo assembly of fungal genomes (e.g. the genome of Verticillium dahliae, Faino et al, 2015).

Alternative approaches to scaffolding have been developed, based on the observation that DNA packaged into nucleosomes and higher-order chromatin shows a consistent pattern of long-range interactions both in vivo using nuclei isolated from living cells (Lieberman-Aiden et al, Science 326:289-293, 2009) or in vitro using DNA-histone complexes prepared from purified DNA and proteins (Putnam et al, Genome Research 26:342-350, 2016). In the past few years, methods to use these long-range interactions to aid in scaffolding genome assemblies and determining the linkage phase of genetic variants in diploid genomes have been developed (Burton et al, Nat Biotechnol 31:1119-1125, Kaplan & Dekker, Nat Biotechnol 31:1143-1147, and Selvaraj et al, Nat Biotechnol 31:1111-1118, all in 2013; Putnam et al in Genome Research, cited above).

Another means of detecting long-range linkages among DNA sequences is based on experimental strategies that result in labeling all sequence reads derived from individual long (50 kb to 250 kb) DNA fragments with the same unique barcode sequence, so that reads can be grouped into clusters of known relationships. The first commercially-available system to produce this type of library was from 10x Genomics (Mostovoy et al, Nat Methods 13:587-590, 2016), but there are now two additional commercial options and a method that can be applied by individual researchers in their own labs. The 10x Genomics system is based on microfluidic technology that creates an emulsion of aqueous droplets in oil, where each droplet contains a different 14-base barcode. The commercial service provided by BGI (Wang et al, Genome Research 29:798-808, 2019), the kit now available from Universal Sequencing (Chen et al, bioRxiv 2019), and the research method suitable for individual laboratories (Redin et al, Scientific Reports 9:18116, 2019) all use the Nextera DNA Flex bead-linked transposase kit available from Illumina (Bruinsma et al, BMC Genomics 19:722, 2018). These methods are known under the general heading of linked-read library methods, and the groups of reads all derived from the same original DNA fragment and marked with the same barcode are sometimes called “read clouds”. In addition to the value of these data for genome assembly, they are also very useful for experimental haplotyping of diploid samples to determine the phase of linkage of variant sites (Zhang et al, Gigascience 8:giz141, 2019).

Objective

The objective of this session is to introduce participants to important issues to consider in genome sequencing and assembly experiments. Simulation of sequencing reads from a bacterial genome will provide example datasets that will serve as input to an assembly program, and will allow testing the effect of sequencing error rate on assembly quality.
Description

De-novo assembly of short DNA sequence reads into a complete genome reference sequence is a challenging and computationally-intensive task for genomes larger than a few megabases, and can be difficult even for small bacterial genomes that happen to be rich in repeated DNA sequences. The key challenge in assembly is to correctly place DNA segments that occur only once in the genome relative to repeated sequences that occur multiple times. The Overlap-Layout-Consensus strategy used to assemble the human genome sequence from Sanger sequencing reads (500 to 900 nt in length) is not feasible for assembly from short reads (50 to 150 nt in length). An alternative approach, based on de Bruijn graph theory, breaks reads up into short fragments called k-mers, which are oligomers k nt in length, with k typically ranging from 19 to 75 depending on read length. These k-mers can be much more efficiently indexed and analyzed than longer reads, and very efficient mathematical methods are available for solving the problem of ordering k-mers into contiguous sequences (Pevzner et al, 2001). A recent publication described a hybrid approach, in which short paired-end reads are analyzed as k-mers, then extended into “super reads” by searching for unique overlapping k-mer sequences at the ends of each read. This process converts large numbers of short reads into a much smaller number of longer “super reads” that can be assembled using the OLC strategy (Zimin et al, 2013). A follow-up report by Zimin et al (2017) extends the “super-read” concept to allow incorporation of long single-molecule sequence reads from Pacific Biosciences or Oxford Nanopore sequencing platforms to form “mega-reads”, and describes assembly of the *Aegilops tauschii* genome to a higher degree of contiguity than previously reported.

Key Facts

The length and number of repetitive sequences in a particular genome determine the sequencing strategy to produce data likely to allow a relatively complete assembly. The objective is to have good coverage of the genome with pairs of reads that are far enough apart to guarantee that both reads of a pair are not within the same repetitive sequence element. If one read is within a repetitive element and the other read is in flanking sequence that is unique in the genome, then that read pair provides evidence locating one copy of the repetitive element within a specific unique-sequence region in the genome. Different genomes have differing lengths and arrangements of repetitive DNA sequences in the genome, so different strategies may be required for library construction and sequencing.

Different approaches have been proposed to evaluate the quality and completeness of de novo genome assemblies. Assemblahthon-1 (2011) and Assemblahthon-2 (2013) were multi-investigator projects to compare the utility of different software tools and analytical approaches to genome assembly. QUAST and REAPR are software tools for assessment of the quality of genome assemblies, both published in 2013. QUAST offers many metrics of assembly quality based on similarity to an existing reference genome assembly, and some metrics than can be applied to new genome assemblies that lack any existing reference sequence. REAPR assesses the degree of correctness of an assembly based on results of re-mapping of paired-end reads back to the assembly and finding regions that don’t match the expected distribution of paired-end sequences - see the manual for details. REAPR is installed on the VCL image, so it is available for use on instances running on the VCL. BUSCO, described in a 2015 publication, assesses assembly completeness by testing for the presence of conserved genes found in many existing genome assemblies of species from the relevant phylogenetic realm to the assembly being tested. The authors of the BUSCO publication maintain a website where data for evaluation of genome assemblies from various taxa of organisms can be downloaded.

- Sequencing library recommendations by Gnerre et al. (2011) for mammalian genomes:
  - Create both paired-end (180 bp insert, 2x100-nt reads) and mate-pair libraries of different (3 kb, 6 kb and 40 kb) insert lengths (see illumina’s webpage for information on mate-pair libraries)
- Variant discovery/genotyping
  - SNPs vs structural variants: SNPs are more abundant as sites, but insertion/deletion (indel) and rearrangement events can affect more nucleotides genome-wide
  - Common alleles vs rare: common alleles are easier to detect and effects can be estimated accurately; rare alleles can be so hard to find that specific strategies are needed to identify sufficient numbers of individuals to provide statistical power to estimate SNP effects

1.1. Course Description
Barcoding works well for genotyping specific individuals at SNPs with common alleles: pooled samples work well for identifying rare variants and estimating allele frequencies.

Exercise - assembly of a bacterial genome from simulated Illumina 100-nt PE reads

The GenomeAssembly shell script will guide the exercises for this class. You can download an archive of the bacterial genome and simulated reads from the web page, or use the following command in a terminal session:

```
ggID='1PWLCABfrEpxAeG0XOBwPsDBE_KxBqG3N';
echo "The file ID is $ggID";
ggURL='https://drive.google.com/uc?export=download';
filename="$(curl --insecure -sc /tmp/gcookie "$ggURL&id=${ggID}" | grep -o '="/name.*"' | sed 's/.* }//;s/<.a>.*//')";
gendcode="$(awk '/_warning_ {print $NF}' /tmp/gcookie)";
curl --insecure -LOJb /tmp/gcookie "$ggURL&confirm=${getcode}&id=${ggID}";
```

- Simulation of paired-end short reads from a bacterial genome sequence can be done with the GemReads.py program used previously, but that process takes some time. Two files containing simulated 100-nt paired-end reads from the *Lactobacillus helveticus* strain DPC4571 genome are included in the DPC4571.tgz archive mentioned in the previous paragraph: sim.r1.fq.gz and sim.r2.fq.gz. The GemReads.py program is installed on the VCL instance if you want to use it to simulate reads from another DNA reference sequence, or simulate reads with different characteristics from the example bacterial genome we are using.

- Use the `df` (remember “disk free”) command to see how much free space is left on your VCL instance - this is a useful practice before doing anything that generates large output files, because it is frustrating to start a large computing job and have it fail due to a lack of disk space to store output files.

- Map the simulated reads back to the reference genome sequence using the BWA aligner - execute the commands `bwa index` and `bwa mem` at a terminal prompt for an overview of the command-line options of the commands to create an index of the reference genome sequence and align the simulated reads to it, or read the manual to learn more of the details about how to carry out alignment of short reads to a reference genome. NOTE: BWA programs read from gzipped files, so you do not need to un-gzip the reference genome (DPC4571.fasta.gz) sequence file, or the sim.r1 and sim.r2 sequence read files. By default, BWA writes SAM-format output to STDOUT (the screen), so you need to redirect that to a file or another command in order to save it. In order to save space, it is most efficient to pipe the SAM output to samtools sort to sort the BAM file so it is ready for use in other downstream applications. The BWA and samtools packages are installed in the search path, so you can use these programs without specifying a complete path to the executable files. Entering either the `bwa mem` or the `samtools sort` command at a terminal prompt without other arguments will return a brief help message describing the key parameters and options available for those programs.

- The MaSuRCA assembler tgz archive has already been unpacked, compiled, and installed in the `/usr/local/masurca` directory of the VCL machine image.

- Use the MaSuRCA assembler to assemble the simulated reads into a genome assembly, following the instructions given in the MaSuRCA Quick Start Guide. The average insert size and standard deviation of insert sizes of the simulated paired-end reads is available from the information scrolled to the screen by the BWA mem program during the alignment process, or in the `KmerCounting_ErrorCorrection.sh` script in the section that describes the GemReads.py command used to simulate the reads.

- Comparison of the genome assembly to the genome reference sequence is possible using whole-genome alignment with MUMmer v.3. This package of programs is installed in the `/usr/local/MUMmer3.23/` directory; look at the list of programs and type `nucmer -h` at a terminal prompt to see the options available for the nucmer sequence alignment program.

- Assembly quality metrics and Assemblathon-1: Outline and notes
Additional Resources

  This paper describes a novel strategy for local assembly of Illumina or other short paired-end sequencing reads into “super reads” that can then be assembled using a modified version of an Overlap - Layout - Consensus assembler.

  This paper provides an overview of metrics for assessment of the completeness of de-novo genome assemblies, along with a discussion of potential sources of bias of different approaches.

  This paper describes a software package that uses results from Mummer v3 Nucmer, delta-filter & show-snps programs to classify sequence differences, which are presented in GFF3 format so they can be visualized in a genome browser.


  This paper provides recommendations for different types of Illumina libraries and appropriate depths of sequencing for best results with the ALLPATHS assembler. While this approach was the state-of-the-art in genome assembly for a period of time, it is no longer considered the optimal approach.

  This paper describes a set of experiments comparing different assembly programs on four genomes, and provides useful insights into the challenges of genome assembly.


1.1.10 Re-sequencing, Alignment, Structural Variation

Objective

The objective of this session is to introduce participants to re-sequencing of genomic DNA or cDNA produced from RNA. The term “re-sequencing” indicates that an assembled and annotated genome sequence is available for use as a reference in analysis of the sequence reads. This means that discovery of new sequence information is typically not the primary goal of these experiments. Instead, the sequence reads are used as a measure of some other biological property of interest. A variety of experimental methods have been developed that use massively-parallel DNA sequencing to measure specific aspects of genome structure or transcriptome activity: the accessibility of chromatin to digestion by nucleases, binding of specific proteins to DNA, three-dimensional interactions between chromosomes in the nucleus, and levels of gene expression are some examples of properties that can be measured in this way. After data preprocessing and quality control, alignment of reads to the reference genome sequence provides results in the form of Sequence
Alignment and Mapping (SAM/BAM) format alignment files, which are then processed by additional software tools to produce the measurements of experimental interest.

Description

Alignment of short DNA sequence reads to a genome reference sequence can be done by several approaches. One is to produce a “hash table”, either of the sequence reads or of the reference genome sequence. A hash table consists of key - value pairs, where the key is a short DNA sequence (a \textit{k-mer}) and the value is the number of times that sequence occurs in the reference genome or the sequence reads. This hash table can then be used to compare sequence reads to the genome to identify regions of the genome that most closely match the DNA sequence in each read. An alternative approach relies on a mathematical transformation called a Burrows-Wheeler Transform, which allows construction of an index of the reference genome sequence. This index is then used to identify the genomic location to which each sequence is most similar. We will not delve into the mathematics of these methods; see the Wikipedia page on short read alignment software for links to many different alignment programs.

Key Facts

The length and number of repetitive sequences in a particular genome determine the sequencing strategy to produce data likely to allow a relatively complete assembly, and also help determine the best strategy for detecting variation among individuals in genome sequence. Short-read sequencing is very useful for detecting substitutions or insertion/deletion events (indels) of one to a few basepairs, but is not as sensitive for detecting larger indels (>50 bp), or rearrangements such as inversions, translocations, or duplications. Long-read sequencing has much greater sensitivity for detecting these larger-scale phenomena, but is still a relatively new approach for structural analysis of genomes of higher eukaryotes because of the higher relative cost of long-read methods relative to Illumina sequencing. This cost structure is changing with the increases in the accuracy and yield of reads from PacBio and Oxford Nanopore Technologies instruments. A recent comparison of software designed to detect structural variation in short read resequencing data (Cameron et al, 2019) concluded that programs designed to do local reassembly of reads around regions of putative structural variation give better results than programs based on other algorithms. An in-depth study in Drosophila comparing the number of structural variants caused by transposable elements using short-read resequencing data versus long-read de-novo genome assembly reported that de-novo genome assembly detected hundreds more variant sites in regions that could be recognized as comparable between the reference assembly and their two new assemblies (Ellison and Cao, 2019). While comprehensive analysis aimed at identifying all structural variants among a group of genomes will probably be best done by de-novo genome assembly with long reads and other data types, it may be possible to survey a subset of structural variation more cost-effectively using short read data given the appropriate data types and software tools. Linked-reads, as produced from libraries made using 10x Genomics Chromium method, the BGI stSLR method, Universal Sequencing’s TELL-Seq kit, or the method of Redin et al (2019), can be used to detect structural variation, with advantages relative to standard short read data in sensitivity and relative to long-read sequencing in cost-effectiveness (Elyanow et al, 2018, Zhang et al, 2020).

Variant discovery/genotyping

- SNPs vs structural variants: SNPs are usually more abundant in terms of numbers of sites, but indels and rearrangement events can affect more nucleotides genome-wide. A report in Nature Genetics (Chiang et al, 2017) suggests that structural variants often have larger effects on relative expression levels of nearby genes than do SNPs.

- Common alleles vs rare: common alleles are easier to detect and effects can be estimated accurately; rare alleles can be so hard to find that specific strategies are needed to identify sufficient numbers of individuals to provide statistical power to estimate SNP effects

- In-read barcoding of individual samples, as is common in GBS or RAD-seq approaches, works well for genotyping specific individuals at SNPs with common alleles: sequencing pooled samples without individual IDs works well for identifying rare variants and estimating allele frequencies.
Many software packages are available for identifying structural variants in alignments of short-read (usually Illumina) sequence data to a reference genome (Alkan et al., 2011). Layer et al. (2014) describe Lumpy, a program that detects structural variation in whole-genome sequencing data by synthesizing information from different kinds of read alignment results - split reads (where a single read aligns to two different locations), discordant read pairs (where the paired-end reads from a single fragment align to locations inconsistent with that expected based on the fragment sizes in the sequencing library), or differences in read depth (due perhaps to variation in copy number of a particular sequence in the sample genome relative to the reference). A more recent publication by Becker et al. (2018) describes a Python tool for structural variant detection that uses an ensemble of different structural variant detection programs - BreakDancer (Chen et al., 2009), BreakSeq (Lam et al., 2010, cnMOPS (Klambauer et al., 2012), CNVnator (Abyzov et al., 2011), Delly (Rausch et al., 2012), Hydra (Quinlan et al., 2010), and Lumpy. An extension of this approach (Zarate et al., BioRxiv 2018) uses parallelization to run some or all of the SV callers BreakDancer, BreakSeq, CNVnator, Delly, Lumpy, and Manta (Chen et al., 2016), on a multi-core computer in roughly the same time required to run a single program. Two comparison of results from different software tools have recently been published; one reported results from 69 different packages (Kosugi et al., 2019), while another comparison selected a subset of 10 programs based on different algorithms or combinations of algorithms (Cameron et al., 2019). The latter paper reports, perhaps not surprisingly, that one of the better programs for detection of structural variants in Illumina short read data is Gridss, a software package previously described by the same research group (Cameron et al., 2017).

Efficient analysis of data from population-level whole-genome sequencing projects in humans is essential, due to the size of the human genome and the computational intensity of sequence alignment and variant detection. Johnston et al. (2017) describe two new programs, PEMapper and PECaller, to improve the efficiency of these analyses. Smith et al. (2017), describe the Genome Rearrangement Omni Mapper (GROM) program to efficiently identify all classes of sequence variation including SNPs, structural variants, indels, and copy-number variants.

Firtina & Alkan (2016) report that changing the order of sequence reads in a FASTQ file, followed by alignment of the reads to the human reference genome and SNP calling with various software tools shows discordance in identified variants, ranging from less than 1% to around 25%. This observation suggests that any variant-calling routine should be tested for sensitivity to read order in the input FASTQ files, although for large datasets this repeatability analysis would be extremely time-consuming.

Variant normalization, or reconciling differences in alignments and SNP or indel calls in low-complexity regions, is an important topic for researchers interested in making a complete catalog of genetic variation in a population. A seminal paper by Tan et al. (2015) introduced software for variant normalization using a VCF file and the reference sequence. Bayat et al. (2017) reported additional concerns regarding the normalization algorithm used by other software and offered another software package they believe does a better job.

Exercises

1. Data suitable for Gridss analysis, derived from a Drosophila melanogaster resequencing experiment, are available for download: reference genome archive, blacklist file, and reads.bam file (or if you are having issues downloading the bam file use this link) of Illumina reads aligned to the reference genome. These reads are from Sequence Read Archive accession SRR2033228 - the reason for providing the BAM file rather than the raw read data is to save the considerable time required to align the reads to the reference genome. Notes are also available describing the steps in preparing the BAM file and running Gridss.

2. Erik Garrison, lead author of the Freebayes SNP caller, has an exercise in alignment and variant calling on his Github page, using both E. coli and human datasets downloaded from web sources, for class please use the updated exercise. The software required for this tutorial is installed in the VCL machine image. A set of slides from a presentation Garrison gave in 2015 describing the Freebayes SNP caller and how it is used in the 1000 Genomes project exploring human genome diversity are also available.

3. Data from an exercise presented at a 2016 Canadian bioinformatics workshop are in Module3.tar.gz. You can directly download the module archive with the following terminal command in an SSH session:
Web pages describe the steps required to align samples of reads from normal and tumor samples to a reference human genome sequence, then analyze the resulting alignments to identify rearrangements. Before executing this exercise, you must create a “virtual environment” in which you install Python v2.7 and the numpy (Numerical Python) module, because the Lumpy program relies on those dependencies. Create the virtual environment with

```bash
conda create --name=py2 python=2.7
```

- you will have to respond to a question confirming the installation. After creation of the virtual environment is complete, activate the environment using

```bash
source activate py2
```

- you will see that the prompt changes to include (py2), so you can tell from the terminal prompt which virtual environment is in use. Install the numpy module in the terminal window running the virtual environment, using

```bash
conda install numpy
```

- this will also ask for confirmation. Normally the creation of a virtualenv and installation of modules would only need to be done once, but because everything in the home directory is lost when a VCL instance is shut down, these steps must be repeated with each new instance that is started.

3.1. Download a shell script that will carry out the commands given in the webpages linked above, edited to reflect the differences in file paths and configuration of the VCL instance using

```bash
ggID='1CTWJGeBctKpQ7XFgsVcK9UP7nFU9y2Vj'; echo "The file ID is $ggID"; ggURL='https://
˓→drive.google.com/uc?export=download'; filename="$(curl --insecure -sc /tmp/gcookie "
˓→$ggURL\id=$\(ggID\)" | grep -o '"uc-name.*</span>' | sed 's/>.*/s/<.a> .*/'))"; \\
˓→getcode="$(awk '/_warning_/ {print $NF}' /tmp/gcookie)"; curl --insecure -LOJb /tmp/
˓→gcookie "$\(ggURL\)&confirm=$\(getcode\)\&id=$\(ggID\)"
```

**IMPORTANT NOTES**

The full human reference genome sequence is too large to work in the alignment step, so after you unpack the Module3.tar.gz archive, you will need to change into the Module3 directory and extract the reference sequences for chromosomes 3, 6, 9 and 12 (because those have rearrangements on them) to a new file. The module3.sh script includes the command

```bash
bioawk -cfastx '{if($name==3 || $name==6 || $name==9 || $name==12) {print ">$name\n
˓→$seq}}' human_g1k_v37.fasta | fold | gzip > chr36912.fa.gz
```

to do this extraction; other ways are possible as well. The process will take a few minutes, so don’t assume that something is wrong if you don’t get a terminal prompt back right away after entering this command.

After extracting the subset of 4 chromosomes from the complete reference genome, the script will delete the files related to the complete human genome reference sequence and index files, to free up disk space. The next step is to create a BWA index before aligning reads to the four chromosomes of interest. The script uses the command

```bash
bwa index -p subset chr36912.fa.gz
```

to create an index with the name ‘subset’. This will take several minutes, so don’t be impatient.
3.2. Map the normal tissue-derived and tumor-derived reads back to the reference genome sequence, piping the SAM-
format output from the BWA mem aligner to samtools sort to sort the BAM file by reference position so alignment
viewers can efficiently display the resulting alignments. The module3.sh script uses the following command line:

```
bwa mem -t8 -p subset reads.tumour.fastq | samtools sort -o tumour.bam -
```

The alignment will take a few minutes for the tumor-derived reads. A modified version of the same
command is used to align the normal-tissue-derived reads to the same reference, convert the output to
BAM, and sort the output BAM file. After both BAM files are complete, the script uses the samtools
index command to produce index files for each of them. If you don’t know how to use the samtools
index command (and no one is born knowing this sort of thing), try typing `samtools index -h` at a
terminal prompt to see what information is available, or do a Google search.

3.3. The command to produce files of discordant reads from the BAM alignments uses the “flag” column of SAM
format, which is a numerical value that contains answers for 12 different yes-or-no questions. The Explain SAM flags
web page has a list of the 12 properties of reads that make up the flag value; if the value 1294 is entered in the box, the
 corresponding properties of the reads are identified. The samtools view -F1294 option means “do not show reads with
flags containing any of these values”, effectively excluding reads with the checked characteristics from the output.

3.4. The command to produce files of split reads uses a script called extractSplitReads_BwaMem in the scripts sub-
directory of the Module3 directory - make sure you use the correct path when you try to execute this command, and
pay attention to the permissions on the files in the scripts subdirectory. How can you change the permissions to allow
execution of all those script files?

3.5. The LUMPY program is installed in the VCL machine image and the path to the executable program is in the
search PATH variable, so you should be able to execute that program without concern about what path to use to the
program. The paths to the input files, and the names of the input files, however, must match those present on your
instance of the machine image.

### Additional Resources

- Information on the Sequence Alignment and Mapping (SAM) format is available at a University of Michigan
  wiki, at Dave’s Wiki, and in the SAM format specification.

- Quality control of alignment files is a valuable preliminary step before investing significant time and effort in
  analysis. A package called `indexcov` is available to efficiently summarize coverage of different genomic regions
  within a single sample, or uniformity of coverage across multiple samples, beginning with alignments in BAM
  or CRAM formats. See Indexcov: fast coverage quality control for whole-genome sequencing. GigaScience
  6:1-6, 2017

- Genomic rearrangements in Arabidopsis considered as quantitative traits. Imprialou et al, Genetics 205:1425-
  1441, 2017. This paper describes a strategy for mapping likely locations of structural rearrangements in a
  segregating population of recombinant inbred lines using low-coverage (0.3x) whole-genome resequencing.

- LUMPY: a probabilistic framework for structural variant discovery. Chiang et al, Genome Biology 15:R84,
  2014.

- CNVnator: An approach to discover, genotype, and characterize typical and atypical CNVs from family and

- Canvas: versatile and scalable detection of copy number variants. Roller et al., Bioinformatics 32: 2375-2377,
  2016.


1.1.11 Discovering and Genotyping Genetic Variation

Global Overview

Different individuals of the same species often have slightly different DNA sequences in nuclear and organellar genomes. Single-nucleotide polymorphisms (SNPs) are differences of single nucleotide bases in DNA, and these may occur at frequencies ranging from less than 10^-4 to greater than 10^-2, depending on the genetic diversity of the species and the divergence among the samples being compared. Sequence variants may also involve multiple adjacent nucleotides (MNPs), and both SNPs and MNPs may be either substitution, insertion, or deletion events. In addition to variation at the level of the nucleotide sequence, structural variation such as inversions and translocations may also be present. The quality of an available reference sequence is an important contributor to analysis of structural variation in genomic sequencing data, and tools have been developed that are reported to work well with fragmented and incomplete draft assemblies (Wang et al., 2017).

Objective

The objective of this class session is to describe and practice methods for discovering and genotyping DNA sequence variation in samples drawn either from structured mating designs or from natural populations. The most cost-effective approach to this challenge, for species with large genomes (i.e., greater than a few hundred megabases) is reduced-representation sequencing, originally carried out using Sanger sequencing (Altschuler et al., 2000). For species with genomes of less than a few hundred megabases, including most microbes as well as many model species and some crop plants, low-coverage whole-genome sequencing can be used as a method of genotyping (e.g., Andolfatto et al., 2011; Au et al., 2013; Huang et al., 2010). A variety of methods exploiting high-throughput DNA sequencing platforms for reduced-representation strategies have been described in the past several years.

Description

A common strategy for reduced-representation sequencing is based on digestion of genomic DNA with one or two restriction enzymes, followed by ligation of adapters bearing oligonucleotide “barcode” sequences. Individual samples identified by different barcode sequences on the adapters can be pooled together for subsequent size-selection, amplification, and sequencing. The sequence data from different samples are then analyzed and assigned to the respective samples using the DNA sequences of the barcode section of the adapter. This method was developed in parallel by two different groups, and is known as Restriction-site Associated DNA Sequencing (RAD-seq, Baird et al. 2008) by one group and as Genotyping By Sequencing (GBS, Elshire et al., 2011) by the other group. The original protocols for RAD-seq and GBS were different, as were the implementation strategies, but the two procedures have become more similar over time until now the distinction is more historical than actual. Several alternative methods for recovery of a specific subset of genomic fragment were reviewed by Mertes et al. (2011); among these are hybridization capture, where synthetic oligonucleotide “bait” sequences are used to selectively enrich a sequencing library for target sequences of interest, and various PCR-based methods for selective amplification of a subset of target sequences.

Key Facts

Repetitive DNA is a significant challenge for restriction-enzyme-based reduced-representation sequencing methods, particularly for species without a reference genome sequence. The choice of restriction enzymes used to reduce genome complexity should be based on the desired numbers of loci, the frequency of restriction sites in repetitive DNA sequences (including organelle genomes), and the expected level of DNA sequence variation. In the absence of a reference genome sequence, empirical studies of the fraction of genomic DNA found in specific size classes in products from digestion with different single-digest and double-digest reactions can be used to estimate the yield of fragments in a double-digest RAD-seq or GBS experiment (Peterson et al., 2012). If a draft genome assembly of the target species or a related species is available, simulation can be used to compare the expected yield of restriction fragments from enzyme digestion with different single- or double-digest combinations of enzymes (Herten et al., 2015, Mora-Márquez et al., 2017). Artifacts in identification of SNP and indel variants from short-read sequencing data are common, and
different approaches have been taken to develop software that avoids some of these artifacts. Comparisons of different software packages on well-studied real datasets have reported differences among software tools in susceptibility to various types of artifacts (Li, 2014). Verdu et al (2016) reported results of a study to evaluate possible contributions of paralogous or repetitive sequences to a RAD/GBS experiment using the reads2SNPs software of Gayral et al (2013), and suggest strategies that may be useful to others dealing with similar challenges.

The Variant Call Format (VCF) file type is widely used to store information about locations of genetic variants, although it uses coordinates based on some sort of reference sequence as a means of identifying locations of variant positions. A common starting point for identification of genetic variant positions is a set of BAM files with alignments of sequence reads from several different individuals. Several software packages can analyze such a dataset to identify candidate genetic variants and save the results in VCF format, including SAMtools mpileup, Freebayes, and the Genome Analysis Took Kit (GATK). Formal specifications of SAM, BAM, and VCF file formats are maintained on Github at https://github.com/samtools/hts-specs.

**Exercises**

A tar archive containing the files for class can be downloaded with the link below.

```
ggID='1b8vlQAhk6lQtAcplqu01Ee2y-xXFw40' ; ggURL='https://drive.google.com/uc?export=download' ; filename="$(curl --insecure -sc /tmp/gcookie "${ggURL}&id=${ggID}" | grep -o '="uc-name.*</span>' | sed 's/.*,//;s/<a> .*//')" getcode="$(awk '/warning_/ {print $NF}' /tmp/gcookie)" ; curl --insecure -LOJb /tmp/gcookie "${ggURL}&confirm=${getcode}&id=${ggID}"
```

1. We will complete an exercise in analysis of a reduced-representation sequencing dataset from a linkage mapping population (two heterozygous outbred individuals and 93 F1 progeny) both without the use of a reference genome, and with the use of the reference genome. The RADseq archive contains an archive of bamfiles for the 95 samples and the reference sequence for LG2 the spotted gar genome for use in the exercise with a reference genome. The complete genome sequence is available either from NCBI, or from the Broad Institute; I used the Broad Institute version. Only LG2 is needed, because the sample dataset includes only loci that map to a single linkage group. The annotation for LG2 is in the same directory; this comes from the GFF3 file for the spotted gar genome (available from Ensembl), for use in identifying SNPs or other variants within or near annotated genes.

2. Sample datasets are available for download for both the TASSEL GBS pipeline and for the STACKS software package. The TASSEL package is written in Java, and will run on the OpenJDK version available on most Linux distributions, as well as on Windows or Mac computers. TASSEL v5 requires Java 8, which is installed on the VCL machine image as the OpenJDK version, so those interested in using the most recent version of the TASSEL GBS pipeline can download and install that software package. The STACKS software package is written in C++, and can be compiled on Linux or on Mac OSX systems.

3. After completing the de-novo and reference-based STACKS analyses (output archives for de-novo and reference-based can be downloaded with these links to save time, Direct Download links for SSH users), additional exercises with the R package vcfR and the command-line tool VCFtools are available. A complete script for all commands in the VCFtools tutorial is available, but you will learn more by going through the tutorial step-by-step. Notes from class are also available. These tools are useful in filtering, summarizing, and subsetting selected data from VCF files. Another option is to explore the use of Freebayes, an alternative program for calling SNPs from BAM alignment files for a set of samples. The BAM files used for the samtools mpileup exercise can also be used for a Freebayes run, and the output VCF files compared. To speed up the Freebayes analysis, use the –use-best-n-alleles 4 option to limit the number of possible alleles the program considers at each site. Freebayes uses a Bayesian approach that considers the data from all individuals in a population to identify variant sites in each individual, and will use a list of the 93 BAM files as input for genotyping much as the SAMtools mpileup program does. Type freebayes -h at a terminal prompt for detailed instructions on command-line options for Freebayes; the general form of the command to run Freebayes is
4. As with SAM and other file formats for genomic data, the VCF format specifies some columns that are mandatory and must contain particular kinds of data, and allows individual software developers considerable freedom to expand on these required fields by adding additional information. In VCF files, the variable fields are the INFO column (which contains summary data about a specific variant across all samples) and the FORMAT string (which specifies data that is available about a variant for each sample with non-missing data at that site) at each genotyped sample, as well as the columns (beginning with column 10) that contain data for each locus from individual samples. One of the vignettes for the vcfR package has a nice overview of the structure of VCF files, although the examples use R and the vcfR package and may not be useful for those unfamiliar with R.

Additional Resources

Other software packages for analysis of GBS/RAD-seq data have been reported, including Unified Network-Enabled Analysis Kit (UNEAK, Lu et al 2013), PyRAD (Eaton, 2014), and AftrRAD (Sovic et al, 2015). A key distinction among these is that in the original versions, some (PyRAD and AftrRAD) allow detection of insertion-deletion (indel) variants as well as substitution events, while others (UNEAK, TASSEL, and STACKS) only considered SNP events. Versions of STACKS after v1.38 (dated April 18, 2016) include the ability to do gapped alignments, and should therefore be able to detect indels in addition to SNPs. Similarly, TASSEL has moved completely to a reference-based analysis format that also allows detection of small indels. Note that a posting to the TASSEL Google group on Feb 12, 2015 announced that the UNEAK package for species without a reference genome available is no longer being developed.

"Edward S. Buckler" <esb33@cornell.edu>: Feb 12 09:00PM

The UNEAK pipeline is no longer under development, as it is much better to generate a poor quality pseudo-reference genome (2x300bp reads and makes 100,000s of small contigs).

Sorry-
Ed

Slides with an overview of GBS - by Keith Merrill

Software links

- Bedtools documentation
- VCFtools documentation
- STACKS manual
- TASSEL v5 GBS pipeline v2 manual
- simRAD R package
- ddRADseq package Github repository
Papers:


1.1.12 R and R Studio

Objective

Provide an opportunity to learn basic R usage with lessons from Software Carpentry, and applications of R and Bioconductor to genomic data analysis with lessons from Data Carpentry. R (v 3.4.4), RStudio (v 1.1.383), and Bioconductor (v. 3.6) are already installed on the VCL image. If you log in using a Remote Desktop connection, you can find RStudio under the Applications menu in the Development category. If you normally log into a VCL instance using SSH,
then you have a choice of either using R in the terminal window on the VCL instance, or using RStudio running on
the laptop that you use to connect to the VCL. The Software Carpentry and Data Carpentry tutorials are written with
the assumption that you will be using RStudio, so if you choose to work in a terminal through SSH to the VCL, you
will not be able to carry out all the exercises in the tutorials.

Background

Base R is designed to load all data into RAM, and this works fine for small datasets. Large genomic datasets aren’t
suitable for this model of memory management, however, so a new approach to data access was developed for genomic
analysis in R. The Bioconductor environment provides tools that allow R to access specific parts of large datasets with-
out loading the entire file into memory, and a large number of R packages for genomic data analysis are now available
through the Bioconductor package management system. Materials for a two-day workshop on R and Bioconductor for
genomic data analysis are available - the introductory material reviews basic R functions and data types, then expands
into Bioconductor-specific topics.

More links to conference presentations and workshop materials related to Bioconductor are available on the Biocon-
ductor Courses and Conference Talks web page - note that each set of presentation materials is linked to a specific
version of Bioconductor and R in the right-most column, so the most recent course materials will assume that you
have the corresponding versions of Bioconductor and R installed on your system if you plan to carry out the exercises
demonstrated.

Workshop Links

Software Carpentry

Data Carpentry for Genomics

Additional Resources

PubMed Central This paper is the recommended citation for those who publish research results developed with
the use of Bioconductor packages. It is not the most useful source of information about how to analyze data
using Bioconductor but instead provides a global overview of the goals and strategies of the development team.

e1004525, 2015. Full Text This paper is focused on meeting the requirements of funding agencies for data
management plans associated with specific research projects. The guidelines presented are not dependent on
any specific mechanism for organizing and managing data on a particular computing platform, but instead
on the larger-scale and longer-term questions of how the data will be made available to the larger scientific
community upon completion of the research.

2009 Full Text This paper provides one perspective on factors to consider in creating a directory hierarchy
for storing data, scripts, analysis results, and other information related to computational biology projects. A
Google search for ‘bioinformatics directory structure’ returns many links to pages with other perspectives, and
personal preferences are important in determining what will work best for any particular person. Uniformity of
naming conventions is one unifying theme, however, because having a consistent system of naming directories
and files makes it easier to manage data with bash scripts and regular expressions.

• R & Bioconductor Manual by Thomas Girke from UC Riverside. This webpage provides a very complete
introduction to R and Bioconductor, with specific examples of many useful tools and analytical methods applied
to genomic datasets.

Last modified 20 February 2019. Edits by Ross Whetten, Will Kohlway, & Maria Adonay
1.1.13 Transcriptome Analysis: Differential Gene Expression and Annotation

Global Overview

Transcriptome analysis is a very broad topic that covers a spectrum from initial characterization of expressed genes in a non-model species with no other genomic information available, to detailed analysis of alternative splicing and gene expression among organs, tissues, or even individual cells of a model organism for which a well-annotated reference genome sequence is known. As previously noted, if the objective of a sequencing experiment is simply discovery of the sequence itself, then experimental design considerations may be less critical, but if the objective is to use the sequencing experiment as a quantitative measure of some biological process (such as gene expression or alternative splicing differences among individuals, developmental stages, or treatments), then an appropriate experimental design is essential.

Objective

The objective of this session is to introduce participants to the additional complexity of analyzing transcriptomes by deep sequencing, above the already complex process of analyzing genome structure. RNA transcripts vary both in abundance, in primary sequence, and in the outcome of splicing processes. All these types of variation can have important biological effects, and may be of interest in a “transcriptomics” experiment. Annotation is the process of describing functional elements in genomes, including regions that are transcribed and processed into various kinds of functional RNA molecules. Messenger RNAs that encode proteins are an abundant class of functional RNA molecules, but by no means the only one - long non-coding RNAs, microRNAs, ribosomal RNAs and transfer RNAs are other important classes of functional RNA molecules.

Description

RNA-seq experiments are growing in popularity as a means of characterizing the transcriptome, detecting alternative splicing events, and measuring differences in gene expression between samples of different types. The importance of good experimental design in conducting RNA-seq experiments is emphasized in the first paper in the recommended reading, by Auer and Doerge. Any experiment in which differences between samples are to be interpreted in a biological context should take seriously the need for good experimental design. The most reliable conclusions will result from a well-replicated design in which the experimental treatments are orthogonal to nuisance factors. Slides with an overview of RNA-seq workflows and some discussion of experimental design and analysis strategies are available. Engström et al (2013) compared the performance of several different programs designed to align RNA-seq reads to reference genome sequences, and provide a thorough discussion of the advantages and disadvantages of the programs tested. More recently, Conesa et al (2016) surveyed best practices for RNA-seq experiments, and noted that no single workflow is optimal for every experiment.

The exercise in RNA-seq data analysis will follow the description in the EdgeR user’s guide or the DESeq2 vignette on your own or in class. The exercise is based on an experiment reported by Cumbie et al. (2011), and involves comparison of gene expression levels in Arabidopsis plants inoculated with a bacterial pathogen or mock-inoculated with sterile solution. The complete data from the experiment are downloaded from NCBI SRA during the exercise using the At_RNAseq.sh script saved in the AtRNAseq archive; R scripts to run differential gene expression analysis with DESeq2 or edgeR packages are in the same directory. The same directory also contains smaller datasets consisting primarily of reads that map to chromosome 5 of the Arabidopsis genome; these smaller datasets allow the exercise to be completed more quickly and with less RAM if you want to carry out the analysis on a laptop with fewer resources.

Key Facts

RNA-Seq library construction strategies may be different for different experimental objectives:
• Differential gene expression: one sequenced “tag region” per gene is enough to estimate relative levels of
gene expression if a reference genome sequence is available to allow mapping tags uniquely to genes, but “tag
sequencing” does not give information on alternate splicing.

• Gene discovery: comprehensive coverage of transcripts is an asset to obtain complete sequences of expressed
genes in species for which a reference genome sequence is not available. Normalization methods that reduce the
difference in abundance among transcripts can work well to obtain more complete coverage of all transcripts,
but may be a problem if accurate estimates of the relative abundance of transcripts is an experimental objective.

• Alternative splicing analysis: complete coverage of exons is essential, and estimates of relative abundance are
important also. Long-read sequencing methods (PacBio and Oxford Nanopore) are becoming the method of
choice for analysis of alternative splicing events because they allow analysis of combinations of alternative
splicing events across the entire length of the mature transcript. This advantage is most important in studies of
genes in which multiple sites of alternative splicing are known, and a central question of interest is which events
occur in the same transcript.

Often researchers are interested in all aspects of transcriptome analysis – discovering new transcripts or alternate
splicing events of annotated genes, plus measuring relative abundance and detecting genetic variation – so many
RNA-Seq experiments use non-normalized libraries of cDNA synthesized by priming with random oligos, to give
relatively uniform coverage across entire transcripts. Accurate reconstruction of alternatively-spliced transcripts from
individual genes is an important part of RNA-seq data analysis if the experimental objectives include testing for
significant differences in levels of alternatively-spliced transcripts among individuals (genetic variation in splicing) or
among treatment groups (which may include developmental stages as well as environmental conditions or chemical
exposures). Software designed to test for association between genetic variants and levels of alternatively-spliced
transcripts is available (Monlong et al, 2014). Pipeline tools that combine multiple software packages into an integrated
analytical approach have been described by multiple groups (Cumbie et al, 2011 and Varet et al., 2015 are examples);
these may be worth setting up if you have a lot of data to analyze and want the added functionality of an integrated
pipeline.

Exercises

You can use the commands found in the Transcriptome_data.txt to download the required archives directly from the
command line.

1. A subset of the experimental data from Cumbie et al 2011 is saved in the AtRNAseq archive. The files c1.fq.gz,
c2.fq.gz, and c3.fq.gz contain sequence reads from three biological replicates of control samples, and files
t1.fq.gz, t2.fq.gz and t3.fq.gz contain sequence reads from three biological replicates of test samples. The
file Atchromo5.fasta.gz contains the sequence of Arabidopsis chromosome 5, and the file TAIR10.cDNA.fa.gz
contains predicted transcripts from the TAIR10 Arabidopsis genome assembly. Matrices produced by Salmon
using k-mer-based “quasi-mapping” of read counts or read TPM values are available; these can be imported into
DESeq2 or edgeR sessions and analyzed according to the vignettes for those packages (links provided above or
with direct download text).

2. A script to download the complete data from the Sequence Read Archive at NCBI is At_RNAseq.sh, and scripts
to analyze the resulting data with kallisto and either edgeR and DESeq2 are also available - these scripts are
saved in the AtRNAseq archive from the google team drive, and links are provided here for those who want to
try the analysis on other machines. A script to load output from RSEM into R for analysis is also available.

3. A fairly comprehensive discussion of RNA-seq workflow options (including different approaches to producing
tables of read counts from BAM alignment files) is available in a Bioconductor tutorial on gene-level exploratory
data analysis; a description of using biomaRt, GO, and KEGG for annotation is given in this tutorial.

4. BAM alignment files are not the only way to estimate the number of transcripts from each gene detected in
an RNA-seq dataset; an alternative approach is to create a k-mer hash table of the transcripts that might be
detected, then use that table to analyze the filtered and trimmed reads themselves to estimate the count of reads
from each transcript, and therefore the counts for each transcript detected. Software tools to carry out this type
of transcript-count estimation include Sailfish, Salmon, Kallisto, and HTSeq. The AtRNAseq archive from
the google team drive contains a file called TAIR10.cDNA.fa.gz that contains 41,671 sequences of putative transcripts predicted from gene models in the TAIR 10 version of the Arabidopsis genome assembly. This file was downloaded from www.arabidopsis.org, from the Download/Sequences/TAIR10blastsets directory as a file called TAIR10_cdna_20101214_updated.

5. The “Tuxedo” package of programs (Bowtie2, Tophat, Cufflinks) provide splice-aware read alignment, transcript reconstruction, and estimation of transcript abundance. The latest versions of Bowtie2, Tophat, and Cufflinks are available as compiled executables, and those version can read and write gzipped files. Simply download and unpack the archives for each program, then create a symbolic link between the program and the /usr/local/bin directory.

6. A complete tutorial for analysis of RNA-seq data using Tophat and Cufflinks is available in Trapnell et al (2012); this can be used as a guide to carry out analysis of the control and test datasets used for the RNA-seq exercise described above.

7. An older tutorial for Gene Ontology (GO) Term Enrichment from RNAseq analysis. The tutorial from a Cold Spring Harbor Plant Biology short course contains information on the overview of the GO term enrichment and notes on the sampling procedure used to shrink the dataset from the full NCBI record. For more information the Gene Ontology page has links to the annotation tables of various organisms. Additionally, a vignette for the goseq package for GO Term Enrichment using v3.4 of Bioconductor is also available.

8. An exercise in evaluating the influence of total read depth on the sensitivity and precision of detecting genes using RNA-seq data is available. The download.bamfiles.sh script will download a set of BAM files from Google Drive and run the Hisat2 reference-guided transcript assembler on the files, then compare the GTF files output from the Hisat2 runs with the TAIR10 Arabidopsis reference genome assembly annotation to assess the sensitivity and precision of detecting annotated genes from RNA-seq data. These data are from the study of Marquez et al, 2012 - the BAM files contain only reads aligning to chromosome 2 of the Arabidopsis genome, and were subsampled to represent 20%, 40%, 60%, and 80% of all the available data, as well as the complete set of reads aligned to chr2. A set of questions to answer about the output from the experiment is also available.

9. An exercise on annotation of assembled transcripts (either reference-guided or de-novo assembled) with TransDecoder is available. The annotation.sh script file picks up where Exercise 8 ends, and assumes that the GTF file produced by Stringtie reference-guided assembly of RNA-seq reads is available. Some editing of the script will be necessary to make sure the file names and paths are correct for the files you are using for the annotation exercise. The text file Trinotate_Bioconda_install.txt has information on how to install the complete Trinotate annotation pipeline using Bioconda - this may work on the VCL image, but will be most useful on the HPC because it has more resources available to actually process large datasets.

Additional Resources

- Choice of library size normalization and statistical methods for differential gene expression analysis in balanced two-group comparisons for RNA-seq studies. Li et al, BMC Genomics 21:75, 2020. These authors compare different normalization methods and statistical tests for sensitivity and specificity in analysis of simulated RNA-seq datasets, where the correct answer is known, and report that different methods give optimal results depending on the experimental design.

- CHESS: a new human gene catalog curated from thousands of large-scale RNA sequencing experiments reveals extensive transcriptional noise. Pertea et al, Genome Biol 19:208, 2018. These authors report the discovery of 224 novel protein-coding genes and 116,156 novel transcripts in the human genome, in additional to millions of transcripts they hypothesize are transcriptional ‘noise’. See also the Research Highlight by W.F. Doolittle in the same issue of the journal, which discusses the definition of gene function as “honed by natural selection in order to contribute to organismal fitness”, and the alternative perspective that suggests that transcribed regions of the genome must have a function because they are transcribed.

• Systematic evaluation of spliced alignment programs for RNA-seq data. Engström et al, Nature Methods 10:1185-1191, 2013. This paper reports results of comparisons of several different splice-aware alignment programs, and concludes that none of the programs tested is optimal by all criteria. The STAR alignment program (Dobin et al, 2013; see next reference) ranks highly by most measures, though, and is recommended for use by the Broad Institute as part of their Best Practices pipeline for variant discovery in RNA-Seq experiments.
• Molecular indexing enables quantitative targeted RNA sequencing and reveals poor efficiencies in standard library preparations. Fu et al, PNAS 111:1891–1896, 2014
• Robust adjustment of sequence tag abundance. Baumann & Doerge, Bioinformatics 30(5):601-605, 2014


1.1.14 Genome or Chromatin Structural Analysis: Chromatin immunoprecipitation, DNase hypersensitivity, 3-D conformation

Global Overview

High-throughput sequencing has been widely adopted as a method of collecting data about events that occur in genomes. Examples of events that occur in genomes include three-dimensional folding to pack genomic DNA efficiently into compact spaces, DNA modifications such as methylation or glycosylation, and interactions between DNA and regulatory or structural proteins. A reference genome sequence assembly is required for analysis of these types of genomic events, because the key question is one of quantitative differences among genomic locations - which sites in the genome show higher frequencies of events, and which show lower frequencies? This emphasis on quantitative analysis of the results means that experimental design is a critical part of this type of experiment. The goal should be to avoid confounding technical sources of variation (lane-to-lane variation on the sequencer, sample-to-sample variation in library preparation, etc) with biological variation of interest (differences between genotypes, treatment groups, developmental stages, or other biological factors).

A variety of specific file formats have been created for storage of data related to genomic location of different types of events. The University of California at Santa Cruz genome browser site includes a FAQ list that describes many of these specialized file formats. Binary Alignment and Mapping (BAM), Browser Extensible Data (BED), General Feature Format (GFF), and Variant Call Format (VCF) are widely-used formats; some of the other formats described at the UCSC site are used primarily by human genome researchers.
Objective

The objective of this session is to introduce methods for analysis of genomic events, using high-throughput sequencing primarily as a means of detecting specific fragments of DNA from a known reference genome. The DNA sequence itself is not the primary goal of the experiment in this case; it is simply a means of identifying specific regions of the genome that meet specific experimental criteria. These criteria can include

- covalent modification of the DNA itself, such as methylation or other modifications of nucleotide residues in DNA, or covalent modification of DNA binding proteins such as methylation or acetylation of histones
- modification of the arrangement of nucleosomes along the DNA or the physical packing of the DNA-histone complexes into higher-order chromatin
- interactions between specific DNA regions and DNA-binding proteins such as transcriptional activators or repressors
- interactions between regions of chromatin that are physically distant from each other on chromosomes but interact in three-dimensional space within nuclei.

Description

The methods used for chromatin analysis are all similar in concept, although the chromatin characteristics assayed differ along with the experimental methods used to assay them. Tsompana and Buck (2014) provide an overview of methods for analysis of chromatin “accessibility”, or structural differences that result in differing sensitivity of the DNA in chromatin to modification by various types of proteins. Chromatin immunoprecipitation and sequencing (ChIP-seq) represents another class of experimental techniques, and relies on immunoprecipitation of DNA fragments as a means of enriching DNA sequences. This technique can be used to study any characteristic of chromatin for which a specific antibody is available, ranging from covalent modification of DNA nucleotides (e.g. cytosine methylation) or ubiquitous DNA-binding proteins (e.g. histone acetylation) to the presence of specific DNA-binding proteins (e.g. a transcription factor bound to a transcription promoter region). ChIP-seq, and other methods for analysis of chromatin structure, result in detection of a pool of sequences enriched for sites that meet a set of experimental criteria. The ratio of “signal”, or sequences that meet the experimental criteria, to “noise”, or random fragments of genomic DNA that do not meet the experimental criteria, determines the sensitivity and specificity of the experiment. The ENCODE consortium published guidelines for analysis of ChIP-seq data that provide a good overview of potential sources of technical as well as biological variation (Landt et al, 2012), and a publication by Thomas et al (2017) compared peak-calling algorithms in several software packages and identified strengths and weaknesses of different tools. The MACS2 package performed well in these comparisons - instructions for installing that package in a Python virtual environment on a VCL instance are available. A recently-developed procedure called “cleavage under targets and release using nuclease” (CUT&RUN) uses an antibody against a protein of interest, followed by proteinA-coupled micrococcal nuclease treatment to cut DNA on either side of the bound protein, and is reported to have high sensitivity and low background (Skene et al, 2018). The level of background from this approach is so low that software designed for analysis of ChIP-seq data is not optimal for CUT&RUN data, so new analytical approaches have been developed (Meers et al, 2019).

One advance that has had major impacts on high-throughput sequencing library preparation is the application of a bacterial transposase enzyme to fragment DNA and attach sequencing adapters to the fragments (Adey et al, 2010). This approach works well for construction of libraries from genomic DNA, but is particularly well-suited to experiments with very small quantities of DNA (sub-picogram amounts, Picelli et al, 2014), and has been adapted for use in experiments to explore chromatin structure (Buenrostro et al, 2013; Gabdank et al., 2016).

Eukaryotic chromosomes exist in vivo as tightly-packed complexes of DNA and proteins, and high-throughput sequencing methods for exploring the three-dimensional structure of those complexes were developed several years ago (Lieberman-Aiden, et al., 2009). Few genome assemblies are completely finished, without gaps or errors, and the three-dimensional interactions of chromatin within nuclei has proven informative in evaluating the accuracy of genome assemblies and determining the phase of linkage of sequence variants at long distances within individual eukaryotic chromosomes or individual genomes in metagenomic studies (Burton et al, 2013; Kaplan and Dekker, 2013; Selvaraj
et al., 2013). A 2017 publication (Dudchenko et al., 2017) described a new approach to using 3-dimensional contact data for building chromosome-scale scaffolds of genome assemblies, and demonstrated the method on both a well-studied human genome and on genomes of two mosquito species. This method has been used to improve the contiguity of the barley genome assembly (Mascher et al., 2017).

**Key Facts**

The analysis of genomic events must consider the nature of expected events, as well as sources of technical and biological variation. A key question is whether the event of interest occurs at discrete sites (such as a transcription factor binding to gene promoter sequences) or varies across broad regions (such as histone methylation or acetylation). The number and size of regions at which events occur has a significant impact on the signal-to-noise ratio of the experiment, and therefore a strong influence on the number of sequencing reads required to reach adequate coverage to allow meaningful statistical analysis of potential differences. Analyzing genomic events as reported by a collection of DNA sequence reads typically involves ‘normalization’ or adjustment of the data to account for differences in the numbers of reads obtained from different samples. This step is critical because the objective of the experiment is to identify differences in the frequency of detection of different regions of the genome under different experimental conditions, and estimates of the frequency of detection can be biased if the number of reads obtained differs under different experimental conditions. If the objective is to compare different kinds of events, such as binding of transcription factors relative to regions of histone acetylation, the challenge for normalization becomes greater, because the nature of the events is so different.

**Exercises**

1. A brief introduction to chromatin immunoprecipitation-sequencing experiments is presented in the IntroToChIPseq.pdf document.

2. An exercise using Galaxy, an on-line sequence analysis resource with a graphical user interface, is described in the Tutorial_ChIPseqOnGalaxy_2020.pdf document. This uses aligned reads in BED format, which is not a file format directly produced by most read alignment programs. Instead, the BAM files that are the direct output of alignment programs must be converted into BED files using a program designed for that conversion, such as BEDtools. The sample datasets used in this exercise are a subset of the data described by Ross-Innes et al. (2012); reviewing that publication will give you an overview of how the ChIP-seq analyses conducted in the exercise fit into the overall biology of the experimental system and the biological conclusions drawn by the authors. The public Galaxy server can be slow or unavailable, depending on network traffic, local computational load, and hardware maintenance issues, but offers a free portal to a wide variety of NGS analysis software tools. As an alternative, the data required to complete the ChIP-seq tutorial has been saved in the archive chipseq. A script file, chip.script.sh, contains commands to execute the same set of analyses as the Galaxy tutorial, but using software installed on the VCL machine image rather than the Galaxy server. Execution of this script should require only entering the path to the script at a terminal prompt; the script will create a directory called chipseq in the home directory that contains output files. Running the script is not as informative as working through the step-by-step exercise in the Tutorial_ChIPseqOnGalaxy.pdf document, but reading the tutorial document and executing the script commands one at a time would be a good compromise offering the speed of local computation and the step-by-step approach of the Galaxy tutorial.

3. The BED file format is commonly used to describe the locations at which peaks of sequence reads mark likely locations of immuno-reactive complexes (DNA modifications or protein binding), and the bedtools program is a powerful tool for analysis and manipulation of such files. Bedtools can be installed from the Ubuntu software repository, and a tutorial with guided exercises using bedtools and DNaseI-hypersensitivity data from Maurano et al. (2012) is available. The Integrative Genome Viewer is a tool for visualizing genomic data, and provides a graphic interface for exploration of experimental results; the bedtools tutorial includes some visualization steps as examples. A video-guided tutorial (2 hours, ~130mb) is available for download from google drive with this link along with the text document from the session here. There is also bedtools.tutorial.sh which is a script to download the IGV and quickly get started with some ChIPseq exercises. A zoom-recorded class session
from March 27th, 2020 with transcribed audio captioning is available here or by download from google drive (transcript).

4. Evaluating data for potential bias and other quality issues is an important part of all experiments. Meyer and Liu (2014) discuss approaches for identifying and dealing with bias in sequencing datasets, and Diaz et al (2012) describe a software package (CHip-seq ANalytics and Confidence Estimation, or CHANCE) designed to help experimenters analyze sequencing data.

Additional Resources

• De novo assembly of the Aedes aegypti genome using Hi-C yields chromosome-length scaffolds. Dudchenko et al, Science 356:92-95, 2017

• A comprehensive comparison of tools for differential ChIP-seq analysis. Steinhauser et al., Briefings in Bioinformatics 1:14, 2016.


• Important biological information uncovered in previously-unaligned reads from ChIP-seq experiments. Ouma et al, Scientific Reports 5:8635, 2015 – This article highlights the differences among alignment programs in ability to map reads with multiple mismatches to the corresponding genomic region, and reports that the SHRiMP alignment program has the highest sensitivity of the programs compared. Using this program with reads that fail to align to the target genome using Bowtie, the authors were able to recover additional useful data and identify additional target protein binding sites.

• CHANCE: comprehensive software for quality control and validation of ChIP-seq data. Diaz et al, Genome Biology 13:R98, 2012

• The lab home page for the program Model-based Analysis of Chip-Seq data (MACS), and the Github software download page.

• The home page for the Galaxy workspace development team

• Chip-seq analysis in R (CSAR): an R package for the statistical detection of protein-bound genomic regions. Muño et al, Plant Methods 7:11, 2011


• The hitchhiker’s guide to Hi-C analysis. Lajoie et al., Methods 72: 65 75, 2015.

• The Variant Call Format and VCFtools. Danecek et al, Bioinformatics 27: 2156 - 2158, 2011
1.1.15 Awk, Sed, and Bash: Command-line file Editing and Processing

Global Overview

The process of collecting, managing, and analyzing high-throughput sequencing data requires the ability to handle large text files, sometimes 10 Gb or more in size. Command-line tools optimized for efficiency and speed are often the best for simple tasks of summarizing data, modifying file formats, and automating repetitive tasks. The four steps of computational thinking are directly applicable in this context - break a complex problem down into simple steps, look for patterns and similarities among steps, then find a general solution for each class of steps and put those solutions together into an algorithm or pipeline that will accomplish the desired task. For repetitive tasks, loops are a useful tool that allow a script to carry out the same sequence of commands over and over until a desired endpoint is reached.

Objective

Awk, sed, and bash are command-line tools that provide tremendous power for managing and manipulating text files of arbitrary size, from very small to extremely large. The objective of this class session is to give course participants experience in using these tools to carry out specific tasks, and experience in searching on-line resources to find out how to accomplish specific objectives with these tools.

Description

The bash shell is the default command-line user interface in the Lubuntu 16.04 Linux system used for the course. A shell script is simply a text file that contains a series of commands recognized by the bash shell, which allows users to create standard workflows and use them over and over with different input files. This is a powerful tool to automate routine or repetitive tasks in data management or analysis, and learning some basic skills can make these tasks much easier. Awk is a scripting language that is particularly well-suited to handling tabular data in text files, such as SAM alignment files or VCF files of DNA sequence variant data. Sed is a “stream editor”, a program that allows manipulation of text files one or two lines at a time, as the text passes through a series of piped commands. The capabilities of these three tools overlap, and many tasks can be accomplished using any of them, but each has its own particular advantages for specific types of problems. Handling multiple files is made easier using file globbing, as described in the FileGlobbing.pdf document, while the RegularExpressions.pdf file has provides an overview of regular expressions, a more general and powerful tool for pattern matching in text files. A powerpoint presentation from a previous year’s lecture is available with this link.

Key Facts

Sequence data analysis often requires the ability to examine and modify the contents of text files, and this is exactly the purpose for which awk and sed were designed. Combining these tools with command-line utilities such as cut, sort, uniq, grep, and other shell functions provides powerful capabilities for summarizing or re-formatting data files. The “modulo” operator (%) in awk, for example, is well-suited to the challenge of working with sequence files in FASTA or FASTQ format, where specific information is found in a particular line within each group of two (for FASTA) or four (for FASTQ) lines. The bioawk version of awk removes the need for this trick by allowing the user to specify that the input file format is ‘fastx’, meaning either FASTA or FASTQ, and the program then assigns the variables $name, $seq, and $quality to the appropriate files in the input file. Another specialized version of awk is vawk, which is designed for manipulation of VCF files containing data on the locations of SNPs and other sequence variants as well...
as which alleles of those variants are detected in a set of samples. Both of these programs are installed in the VCL machine image, so you can compare them and decide for yourself which you prefer. A sample VCF file is available here for use with bioawk and vawk; the official format specification for the Variant Call Format is available on the Github website for VCFtools.

**Exercises**

`awk_sed_bash.txt` has the list of links for the data files needed for the following exercises.

1. **Bash and awk exercises.** Writing and executing loops is a key skill to learn in programming, because this makes completion of repetitive tasks much easier. The bash shell also provides a wide variety of tools to manage system functions, maintain software, and track system resources. Awk allows use of both conditional statements and loops to process and manipulate text files, and can carry out many text-processing activities commonly done using spreadsheet programs in a Windows environment.

2. **Exercises using find, sed, bioawk, and bash to find and modify files.**

3. **Handy tips for bash, awk and sed -** these are examples I have saved from my own applications of these tools. You may find some of these tips useful, but these lists are by no means complete, so feel free to add additional information and keep your own list of the most useful tricks for each of these tools.

4. **An online resource called Linux Command Line Exercises for NGS Data Processing** is mostly about awk.

5. One feature of the bash shell mentioned in the list of handy bash shell tricks is parameter expansion, which offers a range of tools for modifying the values of variables. One example of the utility of these tools is processing a set of FASTQ sequence files - suppose there are samples named S001 to S150, so the sequencing center splits the reads into 150 files named S001.fq.gz to S150.fq.gz. If all these files are saved in a directory, a bash loop can be used to align them to a reference genome, but simply using the input filename as the base for the output alignment file will result in files named S001.fq.gz.bam to S150.fq.gz.bam, in which the “fq.gz” no longer serves a meaningful role. For a variable called $file, parameter expansions such as `${file%%.*}` can be used to retrieve specific parts of the string of filenames and extensions. The `${file#}` and `${file##}` constructs remove matched patterns from the left end of the string stored in the $file variable, while the `${file%}` and `${file%%}` constructs remove matched patterns from the right end of the stored string. Examples make this somewhat more clear, but the best way to see how it works is to practice (for example on the files saved in the AtRNAseq archive).

6. Another useful tool in bash is process substitution, the ability to nest commands inside other commands to combine outputs from different files and commands into a single process. For example, to compare column 2 from one multi-column tabular file to column 3 from a different tabular file and report differences between them:

```
diff <(cut -f2 file1) <(cut -f3 file2)
```

7. **Basic walkthrough of AWK** from chapter 1 of “The AWK Programming Language” with datamash examples as well.

8. **Datamash examples and exercises** is a text file with example problems for practice with Datamash as well as a comment on how to deal with non-uniform filenames.

**Additional Resources**


- The [GNU awk manual](http://www.gnu.org) and [sed manual](http://www.gnu.org) are available on the [www.gnu.org](http://www.gnu.org) website.

- A scan of [The AWK Programming Language](#), a textbook written by the creators of the AWK programing language.
The site panix.com has information on several aspects of the Unix or Linux command-line interface: sed, grep, and bash scripting.

Datamash main page with links to helpful examples and one-liners. A html Datamash manual is also available.

Bruce Barnett’s Unix tutorials page at grymoire.com includes tutorials on awk, sed, grep, and regular expressions, and links to Unix and Linux-related books.

The IBM developerWorks site has a three-part series on awk.

The blog TheUnixSchool has a page with example awk and sed commands to accomplish specific tasks, as well as a grep search function to find previous postings on any topic of interest (look on the right side of the page, below the “join us on RSS Twitter Facebook Google+” box).

The LinuxCommand.org website contains tutorials called Learning the Shell and Writing Shell Scripts that provide a good introduction to shell commands and strategies for writing scripts to combine individual commands into a coherent and efficient workflow. There is also a link to a book called The Linux Command Line which can be downloaded as a PDF.

A quick guide to organizing computational biology projects. Noble, PLoS Computational Biology 5:1000425, 2009 This paper offers a suggested organizational plan for keeping track of data from different experiments and projects in a structured set of directories and files. It is focused on bioinformatics students, so it emphasizes source code and programs more than experimental data or field notes, but the general strategy is applicable to many disciplines.


1.1.16 CLC Genomics Workbench

Objective

To demo CLC Genomics Workbench, proprietary bioinformatic software, useful for visualization and data processing.

Exercises

Working with real data.

1. Download the three files: human_g1k_v37.fasta, reads.tumour.fastq, reads.normal.fastq Note: If you don’t want to split the fastq files yourself, only download the Human glk v37 fasta and then download the split_r1_r2.zip file below.

2. Either follow along with Loading data to split the fastq files from compiled read 1 and 2 to separate files or download the split_r1_r2.zip with the files already split.

3. Zoom video is available for stream or by download from gdrive (transcript) for the resequencing analysis using tracks tutorial. A second zoom video recording is available with this link (or by download, transcript) which finishes the resequencing tutorial and continues into the de-novo genome assembly tutorial.

4. To download the genome sequences for the Denovo assembly tutorial use the following links SRR396639_1.fastq.gz, SRR396639_2.fastq.gz, SRR396640_1.fastq.gz, and SRR396640_2.fastq.gz. Please note that SRR396639_1/2 is a mate-pair library while SRR396640_1/2 are standard paired-end sequences.

Resources

Installation guide
1.1.17 HPC and LSF

Objective

The objective of this section is to work through NCSU’s High-Performance Cluster (HPC) quick-start tutorial to gain familiarity with the NC State HPC, and use other reference videos to review and practice cluster job submission using LSF. Singularity containers are an important asset for working on the HPC, because they allow preservation of exactly the same working environment for the duration of a project, and can be made available to other scientists in pursuit of the goal of reproducible research. Unlike Docker containers, which require root access to run, Singularity containers run with the same permissions of the user who launches the container, but can still be deployed on computing clusters to take advantage of the hardware resources available there.

Exercises

Using NC State University’s High-Performance Computing (HPC) Service:

1. Go to the quick-start guide page on the HPC website.

2. Either watch and follow along with the video guide or work through the text version at the bottom of the same page.

3. A Zoom video recording of the class session working through the Quick-Start tutorial is available with this link (or by video download & transcript download).

4. A text file is available with an overview of commands used to set up a Conda environment on the HPC, and an example LSF job script used to carry out a series of commands using software installed through Conda. A Zoom video recording of the class session working through conda setup and LSF job submission is available with this link (or by video download & transcript download).

5. A text file is available with an outline of steps required to set up a Conda environment with Trinity and Transdecoder, do de novo transcriptome assembly, and carry out functional annotation. Sorting out the details of how to do these steps is left as an exercise. A Zoom video recording of the class session working through these steps is available with this link (or by video download & transcript download).

Final Project

The Final Project is optional, but highly recommended to complete. The goal of this project (should you choose to accept it), is to utilize the HPC and LSF scripting to:

1. Do a de novo transcriptome assembly with yeast RNA-seq data.
2. Identify open reading frames, and annotate those ORFs using results of protein similarity searches, protein domain analysis, and various other tools.

The software to be used includes the Trinity assembler, the TransDecoder suite for identification of ORFs, and the Trinotate suite for annotation. You will need to install the necessary tools to a Conda environment on the HPC. The text file linked to the HPC and LSF section under section 6 of Exercises has some notes on how to set up and run the Conda environment to carry out the Trinity assembly as an example. The diamond (protein similarity search) and Pfam (protein domain search) databases are available at /share/bit815s20/databases. The RNA-seq data are in /share/bit815s20/yeast/RNAdata - look for the six files [rnaC1, rnaC2, rnaC3]_1.fastq.gz, _2.fastq.gz]. The TransDecoder and Trinotate pipelines will be similar, but (of course) using commands specific to those software packages.

More detailed information on how to structure the commands for Trinity, TransDecoder, and Trinotate is available at the respective websites for those software packages. The ability to find the information you need to understand how to use software is an important skill to practice, as new software and new methods emerge all the time. However, we are available to answer questions, either via the Piazza forum or during the MWF 8:30 to 10:20 class periods.

Resources

NCSU OIT HPC main page
NCSU HPC LSF guide
NCSU HPC conda guide - Conda is the preferred method for installing software on the HPC, because it provides a way to manage dependencies and version requirements. Conda was originally written primarily for managing Python virtual environments, but has been extended to include a variety of software not related to Python. See the Bioconda site for more information about the enormous variety (> 7000 different bioinformatics programs) available for installation through the bioconda channel of Conda.
NCSU Bioinformatic Users Group (deBUG) ReadtheDoc site.
Singularity v3.5 documentation from Sylabs.io
Using Singularity on the NIH HPC is documentation for the NIH cluster, but has lots of useful advice and links with more information
Using Singularity on the NC State BRC cluster is specific to the BRC cluster, which uses SLURM rather than LSF. This also has good advice and links.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search