Daniel Burkhardt, Summer 2014

Comparing Alignment Software

1. Introduction
2. Prerequisite Software
3. Downloading Reference Genome from Ensembl
4. Downloading FastQ files from the Short Read Archive (SRA)
5. Aligning using a variety of tools
6. Creating bedgraphs
7. Comparing bedgraphs

Finding SNPs in a gene pathway

1. Introduction
2. Searching for a pathway in Plantcyc and downloading genes
3. Exporting gene locations from Gramene
4. Searching for SNPs in VCF files using R
5. Using the Variant Effect Predictor (VEP) on found SNPs

Introduction

This manual is written to serve as a guide to replicate the analysis performed by Daniel Burkhardt during the summer undergraduate research program during the summer of 2014. Specifically it focuses on two different goals: the use of four alignment programs to map sequenced genomes to a reference genome, and the search for sites of single nucleotide variations in the genomes of multiple individuals. This guide will also include notes for how one might improve the analysis performed in the summer of 2014.

Requisite software:

Guides for installation can be found in the websites for the programs used and may vary with time.

|  |  |  |
| --- | --- | --- |
| Program | Website | Notes |
| **Utilities** | | |
| Bedtools2 | https://github.com/arq5x/bedtools2 | Don’t trust the online documentation!\* |
| Coreutils | http://www.gnu.org/software/coreutils/ | Comes with most linux distributions |
| GNU Grep | ftp://ftp.gnu.org/gnu/grep/ |  |
| FastQC | http://www.bioinformatics.babraham.ac.uk/projects/fastqc/ | Good for checking FastQ format version |
| Picard | http://picard.sourceforge.net/ | Use if samtools doesn’t work |
| SAMtools | http://samtools.sourceforge.net/ | Get used to using this |
| SRAtoolkit | http://www.ncbi.nlm.nih.gov/Traces/sra/?view=software |  |
| Various Scripts | https://github.com/warelab/misc |  |
| **Aligners** | | |
| Bowtie2 | http://bowtie-bio.sourceforge.net/bowtie2/index.shtml | Quick, good documentation |
| BWA | http://bio-bwa.sourceforge.net/ |  |
| NovoAlign | http://www.novocraft.com/main/index.php | Weird licensing system |
| SOAP2 | http://soap.genomics.org.cn/soapaligner.html | Aligner of choice for BGI |

Step 1: Downloading Reference Genome from Ensembl

Select “Download DNA Sequence (FASTA) on the EnsemblPlants website:

<http://plants.ensembl.org/Sorghum_bicolor/Info/Index>

I used:

ftp://ftp.ensemblgenomes.org/pub/plants/release-19/fasta/sorghum\_bicolor/dna/Sorghum\_bicolor.Sorbi1.19.dna.toplevel.fa.gz

Step 2: Downloading FastQ files from the Short Read Archive

Download SRA archive file <file.sra> from website: <http://www.ncbi.nlm.nih.gov/sra>

Unpack archive files using FastQ dump from the SRAtoolkit

/bin/fastq-dump [options] <path> [<path>...]

The National Center for Biotechnology Information’s Short Read Archive is a public online repository for short-read sequencing reads. Reads are accessed by their accession number (matches regular expression ‘SRR[0-9]+’).

Often there are multiple sequencing runs for an organism. For example, SRR999019 and SRR998974 both correspond to sequencing of the SC56-14E Sorghum cultivar. Both runs will be listed under the same experiment. These runs should be merged before or after aligning because they represent reads originating from the same genome.

I used:

SRR633293 SRR771638 SRR771849 SRR998950 SRR998996 SRR998961 SRR999007 SRR998967 SRR999013 SRR999016 SRR999017 SRR998974 SRR999019

Step 3: Aligning with a variety of software tools

Each aligner has it’s own set of options, and I won’t go through them here. All work in essentially the same way: first, each creates an index or hash of the reference genome before mapping reads to the reference genome. Each program also has it’s own algorithm for mapping and scoring mapped reads. It would be worthwhile to review: \*\*\***REFERENCE TO NGS SEQUENCING PAPER** I used default parameters for each aligner while making sure to specify paired-end for paired-end FASTQ files

This progress with result in a SAM or BAM file. BAM files are preferable because they take up considerably less space. I would also suggest sorting the SAM / BAM file because sorting increases the compression efficiency (~10%). Sorting can be done with the Swiss-army knife of SAM / BAM files: Samtools. Samtools has a variety of tools you might want to use. Unmerged runs should be merged at this point. The following commands will convert a SAM file (aln.sam) to a sorted BAM file (aln.sorted.bam)

cat in1.sam in2.sam > aln.sam

Samtools view –uS aln.sam | samtools sort – aln.sorted

#samtools automatically attaches .bam to the output prefix

#the –u flag signals uncompressed bam output –S signals SAM input the ‘-‘ after the pipe signals that input will come from stdin

Step 4: Creating bedgraphs

At this point, one can perform a variety of analysis on the aligned files. I chose to compare areas of overlap and the unique regions between different programs. I first created bedgraph files, which record the coverage (in reads) over a range of bases. This is done in bedtools. From the manual:

The bedtools utilities are a swiss-army knife of tools for a wide-range of genomics analysis tasks. The most widely-used tools enable genome arithmetic: that is, set theory on the genome…quite sophisticated analyses can be conducted by combining multiple bedtools operations on the UNIX command line.

1 18 19 5

1 19 20 8

1 20 21 15

1 21 22 20

1 22 23 27

1 23 24 33

1 24 25 36

1 25 26 43

1 26 27 50

1 27 28 56

Bedgraph files are large, but usually about 50% smaller than the bam files they come from. The columns are chromosome, start, end, and coverage in that order. The genomecov program creates bedgraph files from bam files. I found that genomecov doesn’t like writing bedgraph files for entire alignments—it seems to stop after getting through one chromosome. To combat this, I wrote a script called ‘createBedGraphsByChromosome.sh’ to do each chromosome individually and then concatenates them at the end. It can be found on the github page. Note this script is the one used by me to create the bedgraphs for downstream analysis.

**Example bedgraph file**

The basic syntax:

bedtools genomecov -bg -ibam aln.bam -g genome.index

#genome index file can be made using samtools faidx reference.fa ; Only the first two columns are necessary.

Once you have created bedgraph files for as many aligners as you are interested in, bedtools intersect will allow you to find areas of intersection.

bedtools intersect -sorted -a bowtie2.bg -b bwa.bg > bowtie2.bwa.union.bg

Note: The -sorted flag *dramatically* increases the speed of intersections. At this point the .union.bed file contains all the sections of the genome where the genome overlaps, but notice that it doesn’t contain any information about the genome. This can be done using samtools view and picard samtofasta.

Unique sections for each aligner can be made using wig\_subtract.pl found on the github.

wig\_subtract.pl bowtie2.bg bwa.bg | wig\_subtract.pl /dev/stdin novo.bg | wig\_subtract.pl /dev/stdin soap2.bg > bowtie2.unique\_regions.bg

At this point you should have bedgraph files for the overlapping regions between aligners and bedgraph files for the regions of the genome mapped to by each aligner. This can be done for as many aligners as desired.

Step 5: Comparing bedgraphs

One could do many things with these bedgraphs. I chose to calculate the percentage of the genome mapped by each aligner and the average depth of mapping coverage by each aligner. There are again multiple ways to consider these problems, but using awk is probably the easiest.

Awk is an extremely versatile program which processes files line-by-line. This makes it lightweight and easily applicable to large files. Tutorials can be found by googling “awk tutorials”

To calculate the percentage of the genome mapped in a bedfile using awk is straightforward

awk ‘-F "\t" 'BEGIN{area=0; avgcov=0;}{region = $3-$2;};area += region}END{print area/738540932 #738540932 is the length of the reference genome

The average coverage is also straightforward to calculate.

awk -F "\t" 'BEGIN{avgcov=0;}{avgcov += $4\*($3-$2)}END{print avgcov/738540932}'

The following will calculate both of these for an arbitrary list of bedgraph called FILES and write to a single output

for f in $FILES ; do awk -F "\t" 'BEGIN{sumdiff=0; avgcov=0;}{if ($4 < 101){ diff = $3-$2; avgcov += $4\*diff};sumdiff += diff}END{print FILENAME "\t" sumdiff/738540932 "\t" avgcov/738540932}' $f >> unique.stats & done

There are of course, more interesting things to be done. This paper has a whole slew of statistics one could calculate in order to compare different aligners:

How do alignment programs perform on sequencing data with varying qualities and from repetitive regions? Yu et al. 2012. [10.1186/1756-0381-5-6](http://dx.doi.org/10.1186%2F1756-0381-5-6" \t "pmc_ext)

Finding SNPs in a gene pathway

I will not cover SNP calling in this tutorial, as I am not qualified to give advice. There are multiple SNP calling programs and each has their own pros and cons. See: doi:10.1038/nrg2986

I will assume that you already have a VCF file containing SNP information. Most of this analysis will take place in R. I would also recommend using the R.studio IDE, but it is not necessary.

Step 1: Retrieving Gene Locations

Information about protein pathways can be found on http://www.plantcyc.org/. Using the search bar at the top of the page, search for the name of the molecule of interest. Under the heading “Pathways,” select the pathway of interest. This will bring you to the pathway page. On the tool bar on the right, select “Download Genes.” This will give you a text file of all of the genes in all of the organisms whose genomes are in the PlantCyc database. Retrieving the genes IDs from this text file can be done using grep by finding a regular expression that matches the genes for the organism of interest. For sorghum:

cat AllGenesInAnthocyaninPtwy.tab | grep -oP 'Sb[0-9]{2}g[0-9]{6}' | sort | uniq > geneIDs.txt

This will give you a long list of geneIDs. To find the gene locations, head over to <http://gramene.org/>.

In Gramene, select “Gramene Mart” and choose “Plant genes” for the database and your organism of interest for dataset. On the left hand side of the page click on “Filters” and then restrict query to ”Gene.” Check the “ID List Limit” checkbox and either paste your list or browse for your file of genes.

Next select “Attributes” on the left side of the page. All you need to select for this part is “Gene Stable ID” “Chromosome” “Gene Start” and “Gene End.” (In that order; the order selected affects output)

Finally, at the top of the page, hit “Results” and browse your results. To facilitate downstream analysis, you will want to convert this into bed format (chromosome, start, end, info)

cat genes.txt | awk 'BEGIN { FS="\t" ; } {print $2 FS $3-1000 FS $4+1000 FS $1}'

Step 2: Finding SNPs within genes

At this point I assume that you have a VCF containing SNPs and a tab delimited file with columns containing chromosome, gene start, gene end, and geneID in that order. Bedtools window will return features (like SNPs) in file -a that have overlaps in file -b *along with upstream and downstream regions specified using -w.*

bedtools window -w 5000 -a SNPs.vcf -b genelocations.bed

This will return a VCF file containing only SNPs that are within your genes, or upstream or downstream by 5000bp. (I used 2000bp in my study).

Step 3: Finding SNPs within genes that are shared by different individuals

This step is slightly more complicated that the previous two, but mostly because I chose to do it in R. I would suggest that anyone reading this should do this with bash using awk and grep.

Essentially one only cares about the 1/1 or 0/1 or 0/0 flag under the column for each individual at this step. Even better, since these are inbred lines, it is probably safer to only consider 1/1 or 0/0 calls as 0/1 phenotype may be an artifact of sequencing errors.

For my research, I was only interested in sites of variation that occurred in all stay-green sorghum lines and in none of the wildtype lines I was studying. To do this, I first loaded the VCF file (with header chopped off) into R

snps <- read.delim(“path/to/file/snps.vcf”, header=FALSE)

Then I found a subset of the snps dataframe using subset() I used a regex matcher to find rows with 1/1 in all columns belonging to stay-green lines and 0/0 in all wild-type lines.

inStayGreen <- subset(snps, grepl("1/1", SC35) & grepl("1/1", SC56) & grepl("0/0", RTx7000) & grepl("0/0", SC1708))

This should return a subset of the “snps” VCF file with SNPs within the area defined by bedtools window in the previous step.

Step 4: Predicting SNP effects.

Once again, we return to Gramene. The Variant Effect Predictor (VEP) takes sites of variation and returns one of a number of predicted effects. This includes missense variation, nonsense variation, intron variation, etc. The VEP can be found here: <http://ensembl.gramene.org/tools.html>

The common format is chr start end ref\_allele/alt\_allele strant (e.g. 1 881907 881906 -/C +).

Select the proper reference genome, and format the subset VCF file to match the VEP format and the VEP will take care of the rest.

At this point, one could take a few options. Promoter motif prediction can be done using MEME suite, Bioprospector, etc. Care should be taken to ensure that motifs found are valid and not an artifact of noise in the sequences.