Zhikai Liang's Pipeline for SNP calling (As of Sept.21st 2015)

(Modifications on “James Schnable's pipeline for processing RNA-seq data”)
zliang@huskers.unl.edu

Pre-processing the reads:

For paired end reads only:

Using Trimmomatic-0.33 to filter the data:

```
java -jar /home/zliang/software/trimmomatic-0.33.jar PE -phred33 -threads 8 mg1 mg2
temp.f.p.fq temp.f.u.fq temp.r.p.fq temp.r.u.fq LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:36
```

This will perform the following:

- Remove adapters (ILLUMINACLIP:TruSeq3-PE.fa:2:30:10)
- Remove leading low quality or N bases (below quality 3) (LEADING:3)
- Remove trailing low quality or N bases (below quality 3) (TRAILING:3)
- Scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15 (SLIDINGWINDOW:4:15)
- Drop reads below the 36 bases long (MINLEN:36)

For single end reads:

No preprocessing required

#Note: No additional quality trimming will be conducted on these reads. The alignment software used in this protocol (GSNAP) has the capacity to "soft-trim" reads during alignment, aligning only the high quality portion of the read.

Preparing to align:

You will need:

The various files and scripts included in the latest version of GMAP/GSNAP (http://research-pub.gene.com/gmap/)

Your reference genome as a FASTA file.

A GTF file describing the annotated genes in your reference genome

#Note: A GTF file is _similar to_ a GFF file, but there are crucial differences. If you can't find a GTFfile, you will have to write your own converted to turn a GFF file into a GTF file (each GFF file is a bit difference so I can't just give you a script that will work for any GFF file you throw at it).

#Note: One common source of error is differences in how chromosomes are labeled between your FASTA and GTF files. The same chromosome might be called 1 or chr1 or Chr1 or Chr01.

(Optionally) a list of known SNPs between the genotype used to generate this RNA-seq data and the genotype used to generate the reference genome.

Building the genomic index :

After installing the binaries from GMAP/GSNAP somewhere in your path, you can build the genomic index you'll be aligning reads against using gmap_build. Example command:

```
gmap_build -D ./ -d arabidopsisv10 atTair10.fasta
```

This is telling gmap to build the index of the sequences in the file "atTair10.fasta" which is stored in the current directory, to call that index "arabidopsisv10" and to store it in the current directory "./"

#Note: it is important to conduct this step on a computer with enough RAM or bad things will happen.

Aligning Reads to the Reference Genome:

This is the single most computationally intensive step and can take anywhere from minutes to days depending on read length (longer takes longer), genome size (bigger takes longer), number of processors (fewer takes longer), and single end vs paired end (paired end takes way WAY WAY WAY longer).

Example command (inputs are filtered paired-end reads):

```
gsnap -D ./ -d Zea_mays.AGPv3.22.rm --nthreads=12 -B 4 -N 1 -n 2 -Q --nofails --format=sam
temp.f.p.fq temp.r.p.fq > results.sam
```

There's a lot of stuff here, so let's walked through it from left to right.

```
-D ./
```

This is telling gsnap which directory I store my genomic index files in.

```
-d  Zea_mays.AGPv3.22.rm
```

This is telling gsnap that I want to align against an index called Zea_mays.AGPv3.22.rm

```
--nthreads=12
```

Here I tell the program how many processors to try to use for doing alignments. This particular command came from a very high powered server. Never use more threads than your computer and processing cores.

-B 4

This sets how much information GSNAP will try to load into RAM (as opposed to loading from yourhard drive as it is needed). Possible settings range from 0-5 with five loading the most data into RAM

(and being the fastest) and 0 loading the least. This lets you tune GSNAP to run faster (on high memory computers) or run at all (on computers with less ram).

If you don't enter any settings at all, GSNAP defaults to 2. If your computer doesn't have enough RAM to successfully run your alignment at _at least_ a setting of -B 2, it's best to find another computer, because aligning at slower settings is really slow.

-N 1

Look for novel splicing

-n 2

Only report reads with two or fewer “best” alignments in the genome. It's quite ok to increase this setting, but be prepared for the side of your output file to grow substantially larger if you do (since GSNAP has to report a separate alignment for each location a read maps to, a small number of reads which align at many locations add up quickly).

-Q

If a read aligns equally well to more than two locations – or however many you specificied with -n above -- don't report it at all (the alternative is to report two of the 3-infinite equally good positions at random).

--nofails

Don't write out information on reads which fail to align at all. This saves space by making the resulting alignment file smaller.

--format=sam

Report alignments in the standardized SAM format instead of gsnap's native alignment format.

temp.f.p.fq temp.r.p.fq

The name of the files containing the sequencing reads to be aligned against the genome.

results.sam

The name of the file to store alignment data in.

Converting alignment data to a useful format:

These steps require that you have downloaded and installed the samtools package available from: http://samtools.sourceforge.net/

The plus side of the SAM (Sequence alignment map) format is that it is basically human readable – ifyou're one of the rare humans who has spent a long enough time looking at it to understand what you're seeing. The downside of SAM is that the files are huge (sometimes bigger than the raw reads file you started with) and it's not easy for a computer to find just the parts of the file it is interested in.

So the next step is to convert SAM to sorted and (optionally) indexed BAM. BAM is the binary version of SAM, not at all human readable, but much more compact and much easier/faster for computer programs to access.

Example commands (these shouldn't need much explaining):

```
samtools view -bS results.sam > results.bam

samtools sort results.bam results-sorted

samtools index results-sorted.bam
```

At the end of this process you have a sorted binary alignment file (results-sorted.bam) and – if youindexed it – an index file that tells a computer wherein that file to look for the reads that map to a specific part of the genome called “results-sorted.bam.bai”

Calling SNP:

Create a new folder called “Bam_files”, and move all of files containing “sorted.bam” and “sorted.bam.bai” to this folder.

Running in-house python script below to call SNP

```
import os

import subprocess as sp

import sys

# conducting samtools mpileup to call snp and bcftools convert its output to bcf files (regardless of splicing variants)

# Note: You should create a folder called “SNP_list” before you run this code, because it may generate a thousand hundred of bcf files, otherwise your current directory will be covered by them totally

def call_SNPs(mylist):
    mycount = mylist[0]
    mychr = mylist[1]
    mystart = mylist[2]
    mystop = mylist[3]
    fh = open("SNP_list/temp.var.raw.{0}.bcf".format(mycount),'w')
    snp_call = sp.Popen('samtools mpileup -I -F 0.01 -r {2}:{3}-{4} -uf {0} {1} | bcftools call -
mv -Vindels -Ob'.format(fasta_reference,"
".join(blist),mychr,mystart,mystop),shell=True,stdout=fh)
    snp_call.wait()
    fh.close()
# You can use exon regions extracted corresponding fasta file you used, in order to speed up
the calling step
ef = open("exon_region.txt","r")


initial = []
for n in ef:
        initial.append(n.rstrip().split(','))
temp = []
for x in range(1,11):
        temp.append(str(x))
good_chromosomes = set(temp)
ef.close()
snp_call_list = []
for m in initial:
        if not m[1] in good_chromosomes: continue
        m = map(int,m)
        snp_call_list.append(m)
bam_folder = 'Bam_files'
fasta_reference = '/home/zliang/storage/maize_10/Zea_mays.AGPv3.22.dna_rm.genome.fa'
myfiles = os.listdir("./{0}".format(bam_folder))
blist = []
for x in myfiles:
        if not x[-3:] == 'bam':continue
        blist.append(bam_folder + "/" + x)
# using multiple processors in oder to speed up the calling step
from multiprocessing import Pool
p = Pool(18)
p.map(call_SNPs,snp_call_list)
```