NCSU Fork of TASSEL for Genotyping by Sequencing

Software version: beta 0.9

Date: 2013-02-14

Legal information: This software is provided "AS IS". Use at your own discretion. It is released under the GNU Lesser GPL v 2

Contact information:

Ross Whetten (ross_whetten@ncsu.edu)

This document provides some more details than the README file found in the same directory. It is intended to help you get started and understand the basics of using the software. For ongoing limitations, known bugs, and version history, please read the README file.

This program / pipeline was built as a tool for streamlining the processing of genotyping by sequencing data. It is a fork of the TASSEL software developed and made available by the USDA-ARS Lab with Cornell's Institute for Genomic Diversity (http://www.maizegenetics.net/).

This software takes two FASTQ files representing forward and reverse sequencing of the same flowcell lane and:

1) concatenates 64-nt tags from the forward and reverse sequences into 128-nt tag sequences, using the cluster coordinates as sequence identifiers
2) logs 128-nt tag sequences and barcode combinations
3) generates a tag by taxa output file in log format
4) generates a summary file of the number of forward, reverse, and paired tags at set intervals so that the user can see trends resulting from processing the data file

Output Files

- **Barcode_ID_Info.txt** = Number of barcode combinations detected across all lanes. This is compiled after the minimum observation level is considered.
- **Tags_Totaled.txt** = Number of unique tags across all flowcell / lanes. This is compiled using any minimum observation flag is set.
- **Tags_by_Taxa.txt** = number of tags and taxa combination by lane over all flowcell / lanes. This is compiled using any minimum observation flag is set.
- **Summary_Stats.txt** = Raw printout of the number of forward, reverse, and pairs per lane over X interval. This *does not* take into account the minimum observation flag as it is a raw dump of what is happening before the minimum observation variable is implemented.
- There will be one intermediate file per pair of processed files that is named something like read1-read2.txt. These are large files and when the program has finished running, they can be deleted to recover the disk space.
- The plug-in alters the second key file and stores it as [keyfile2]_mod.key. Once processing of all data files is complete, this file can and should be deleted. If it is left within the directory it will be appended to each time the plug-in is run.

System Requirements:

The following are required regardless whether you are using a local computer or a virtual one via a cloud or high-performance computing center (such as Amazon EC2).

Hardware: Required (optional)

OS: 64-bit Linux (Ubuntu)

RAM: 16GB (32+GB)

STORAGE: A minimum size of space at least 1/4 the combined size of the files you are processing. Example: Two 40GB compressed files [80GB total], need to have access to 20GB of free disk space.

Software:

- Oracle java 1.7 (http://www.java.com; OpenJDK will not work)
- git 1.7+ (http://git-scm.com/)
- cd-hit-est (available in Debian-Med repository) for clustering of tags

File Preparations:

KeyFile Example

The key files will need to be in the following tab-delimited format:

Flowcell	Lane	Barcode	Sample	PlateName	Row	Column	Blank
C116KACXX	7	ACTCCACG	gbs	GBS1	A	1	Blank

…

FASTQ Files

FASTQ files must be named using the following convention

read#_flowcell_s_lane#_fastq.gz

Example: read1_C116JACXX_s_7_fastq.gz

Installation Instructions (Local install)

1. Verify java and git are installed by typing the following from the command prompt.

java -version

git --version

2. Make a directory where you want to store the program (example: PROGRAM_DIR)

3. Navigate to that directory from the command line

4. Execute the following command within the directory:

git clone git://github.com/conceptstailored/NCSU-GBS-Pipeline.git

5. This should have copied a directory called "NCSU-GBS-Pipeline" in your PROGRAM_DIR

6. Create an output sub-directory in your

PROGRAM_DIR/NCSU-GBS-Pipeline/TASSEL_20121011/maizegenetics

called "outputs"

7. Within PROGRAM_DIR/.../outputs create two sub-directories:

/fastq (**and copy your fastq files here**)

/FastqPairedEndTagsAndTaxa (**where program outputs will be stored**)

8. Place your key files in PROGRAM_DIR/.../outputs

** This is the bare bones approach to running only the paired end plug-in.

The TASSEL program can do many other things but requires a bit more setup that can be read about at

http://www.maizegenetics.net/tassel/docs/TasselPipelineGBS.pdf

Command:

To run the paired end plugin, execute the following command (all on one line) via command line from the **PROGRAM_DIR/.../outputs** directory. Optional arguments are in () and explained in *Getting Started Tips*. Alternatively, a bash script with the command, saved and executed in the "outputs" directory, reduces typing errors.

PROGRAM_DIR/.../maizegenetics/run_gbs_pairedEnd_pipeline.pl -[max_memory] -fork1 -[plugin] -i [input_directory] -k [keyfile1:keyfile2] -e [enzyme1:enzyme2] (-s [maximum_count]) (-c [minimum_detected]) -o [output_directory] -endPlugin -runfork1

Example and description:

PROGRAM_DIR/.../maizegenetics/run_gbs_pairedEnd_pipeline.pl -Xmx30g -fork1 -FastqPairedEndToTagCountPlugin -i fastq -k GBS1.key:GBS2.key -e PstI-MspI:MspI-PstI -s 10000 -c 10 -o FastqPairedEndTagsAndTaxa -endPlugin -runfork1

The example above will call on the run script, allocating 30 GB of RAM to java, set the plug-in to FastqPairedEndToTagCountPlugin, look for data files in your fastq directory, use 2 key files named GBS1.key and GBS2.key, use the PstI-MspI enzyme combination for read1 and MspI-PstI for read 2, only take a maximum of 10000 raw paired tags, and only keep any tag observed more than 10 times, writing all output to the FastqPairedEndTagsAndTaxa directory

Getting Started Tips:

1) Read through the README files before attempting to start
2) Close any other programs to free system resources
3) The program will take several hours to run, and will take longer with each lane of data that you process.
4) Optional flags that can be set in the calling command
    - -s = maximum_count = the default setting is to try and process the entire data set per lane. This can be extremely memory intensive. If the program crashes due to memory issues, you can look to the last screen output and set this flag to be a maximum number of tags to accept per lane. Using the screen output

is only a guide; you will have to play around with this value based on your system's configuration. If you have plenty of RAM (say 64 Gb for two or three lanes of data) this flag can be omitted.

- -c = minimum_detected = the default setting is to accept every tag that is detected. This may lead to a lot of singlets. Setting this argument can reduce the overall size of your outputs, and can be used to narrow your outputs to only a count you are interested in.

5) The keyfile argument flag (-k) needs to have 2 arguments even if you are using the same keyfile for both reads. The name doesn't matter so long as it matches what is in your directory, so you could name and enter file1.key:file1.key and that would be accepted.
6) Enzymes and keyfiles need to be separated by a ":".

Amazon EC2 script setup:

The following is provided to help setup an Amazon EC2 instance and run the program. It is also included in a standalone file called **pipeline_amazonEC2.sh** packaged in the directory along with this document.

```
#!/bin/bash

# A script to configure an EC2 instance for
FastqPairedEndToTagCountPlugin analysis

# The script installs the GBS software and Oracle Java 7 (a dependency)


# Use ami-a6ff46cf (CloudBioLinux Ubuntu 12.04 20121019) with 68 Gb RAM
(m2.4xlarge),

# and /mnt disk volume to have access to enough disk space for real
datasets.

# Run script from /home/ubuntu directory


### NOTE: script assumes that read1.key and read2.key files

### are packaged with this script in a tgz archive that is

### uploaded (or emailed) to the EC2 instance and unpacked

### in the /home/ubuntu directory of the EC2 instance.


# Download data from service lab website directly to EC2 #instance,
save to /mnt/NCSU-GBS
#Pipeline/TASSEL_20121011/maizegenetics/outputs/fastq

# after this script is run.


sudo chown ubuntu /mnt

cd /mnt

git clone git://github.com/conceptstailored/NCSU-GBS-Pipeline.git

mkdir NCSU-GBS-Pipeline/TASSEL_20121011/maizegenetics/outputs

mkdir NCSU-GBS-Pipeline/TASSEL_20121011/maizegenetics/outputs/fastq

mkdir NCSU-GBS-Pipeline/TASSEL_20121011/maizegenetics/outputs/PairedTBT

cd NCSU-GBS-Pipeline/TASSEL_20121011/maizegenetics/outputs
```

```
### files read1.key and read2.key are in standard TASSEL-GBS format, eg
###

#Flowcell   Lane  Barcode     Sample      PlateName   Row   Column
      Blank

#C116KACXX  7     ACTCCACG    gbs   GBS1  A     1     Blank


# Key files are in archive with this script; create symbolic links from
outputs directory to those files


ln -s ~/read1.key read1.key

ln -s ~/read2.key read2.key

chmod 744 /mnt/NCSU-GBS-
Pipeline/TASSEL_20121011/maizegenetics/run_gbs_pairedEnd_pipeline.pl


# Install Java 7 from Oracle, using package from webupd8 ppa;

# see http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-
ubuntu-via.html

sudo add-apt-repository ppa:webupd8team/java

# Requires a hard return from user to move through choices


sudo apt-get update

sudo apt-get install oracle-java7-installer

# Also requires user input to accept Oracle license terms for Java7, or

sudo echo oracle-java7-installer shared/accepted-oracle-license-v1-1
select true | sudo /usr/bin/debconf-set-selections


# Install cd-hit clustering software from Debian repository

sudo apt-get install cd-hit


# Run script from outputs directory, giving full path to script after
making it executable, and keeping only tags detected at least 100
times:

# /mnt/NCSU-GBS-
Pipeline/TASSEL_20121011/maizegenetics/run_gbs_pairedEnd_pipeline.pl -
Xmx50g -fork1 -FastqPairedEndToTagCountPlugin -i fastq -k
```

```
read1.key:read2.key -e PstI-MspI:MspI-PstI -c 100 -o PairedTBT -
endPlugin -runfork1


exit
```