Lotus Genome v3.0 - Methods

Vikas Gupta and Stig U. Andersen

March 5, 2014

Contents

1. Introduction

This document is the detailed description of methods section for the Lotus genome article. Aim is to be so thourogh that all the steps can repeated without requiring additional information. Path of the files will be added accordingly if still exists. I will try to add the python scripts in a package but if there is any missing, you can always fetch is from the my GitHub https://github.com/vikas0633/python.

2. Gene Annotation

Primary idea was use the already available genome annotation pipelines/tools, such as PASA, MAKER, EVM and Inchworm but the annotation from the tools mentioned were not very good and so we used a custom build pipeline developed by me and Stig. I will not mention the commands used for the tools which were not used towards the final output.

2..1 Repeat Masking

RepeatScout Version 1.0.5 and RepeatMasker version open-3.3.0 was used for masking the repetitive regions of the genome. RepeatScout was used to construct denovo library from the lotus genome sequence to facilated accurate detection of novel repeat elements. These repeat library was subsequently used with RepeatMasker to mask the repeat regions.

```
### l value using python
>>> math.ceil(math.log(454435385,4)+1)
16.0

### running build_lmer_table from repeat scout
nice -n 19 build_lmer_table -sequence /u/vgupta/01_genome_annotation/01_genome/
    Ljchr0-6_pseudomol_20120830.scaf.fa -l 16 -freq lmer_Ljchr0-6
    _pseudomol_20120830.scaf.fa

### running repeatscout
nice -n 19 RepeatScout -sequence /u/vgupta/01_genome_annotation/01_genome/Ljchr0-6
    _pseudomol_20120830.scaf.fa -output output_RepeatScout_Ljchr0-6
    _pseudomol_20120830.scaf.fa -freq lmer_Ljchr0-6_pseudomol_20120830.scaf.fa -l
    16
Program duration is 5704.0 sec = 95.1 min = 1.6 hr

### filtering step-1
filter-stage-1.prl output_RepeatScout_Ljchr0-6_pseudomol_20120830.scaf.fa >
    output_filter-stage-1_RepeatScout_Ljchr0-6_pseudomol_20120830.scaf.fa

### running repeat masker
nohup nice -n 19 RepeatMasker -gff -lib output_filter-stage-1_RepeatScout_Ljchr0-6
    _pseudomol_20120830.scaf.fa /u/vgupta/01_genome_annotation/01_genome/Ljchr0-6
    _pseudomol_20120830.scaf.fa &
```

2..2 Gene model Generation

2..2.1 RNA-seq

Four pair-end RNA-seq libraries, two from each MG20 and Gifu were mapped on the genome. We ran TopHat and Cufflinks multiple times to find the best suiting parameters for mapping. TopHat v2.0.4 was used together with Bowtie v0.12.8. Tophat aligns the reads to the genome taking exon-intron boundries into consideration. Aligned reads were used to create gene models using Cufflinks v2.0.2 and many non-default parameters were used to detect all potential gene models.

```
#!/bin/csh
#PBS -l nodes=1:ppn=16
#PBS -q normal

echo "========= Job started  at `date` =========="
echo 'for only MG20 tophat cufflinks'

### get the tools from rune's directory
source /com/extra/bowtie/0.12.8/load.sh
source /com/extra/tophat/2.0.4/load.sh
source /com/extra/cufflinks/2.0.2/load.sh
source /com/extra/samtools/0.1.18/load.sh

### nodes to be used
cores=15

### data_dir
data_dir="/home/vgupta/01_genome_annotation/02_transcriptomics_data"

### work dir
work_dir="/home/vgupta/01_genome_annotation/11_tophat/04"

### log file
logfile=$work_dir"/20120917.logfile"


### reference genome
ref="/home/vgupta/01_genome_annotation/01_genome/Ljchr0-6_pseudomol_20120830.chlo.
    mito.fa"
index="/home/vgupta/01_genome_annotation/01_genome/Ljchr0-6_pseudomol_20120830.chlo
    .mito.fa"


echo 'indexing the genome' >>$logfile
### make index for the reference sequence
bowtie-build -f $ref $index
echo "indexing is finished" >>$logfile

echo 'processing first sample' >>$logfile

read1=$data_dir"/2010_02_17_Fasteris_MG20_Gifu_transcripts/100128_s_1_1_seq_GHD-1.
    txt",\
$data_dir"/2010_02_17_Fasteris_MG20_Gifu_transcripts/100128_s_2_1_seq_GHD-2.txt",\
$data_dir"/2010_03_22_Fasteris_MG20_Gifu_transcripts/100226_s_7_1_seq_GHD-1.txt",\
$data_dir"/2010_03_22_Fasteris_MG20_Gifu_transcripts/100226_s_8_1_seq_GHD-2.txt"

read2=$data_dir"/2010_02_17_Fasteris_MG20_Gifu_transcripts/100128_s_1_2_seq_GHD-1.
    txt",\
$data_dir"/2010_02_17_Fasteris_MG20_Gifu_transcripts/100128_s_2_2_seq_GHD-2.txt",\
$data_dir"/2010_03_22_Fasteris_MG20_Gifu_transcripts/100226_s_7_2_seq_GHD-1.txt",\
$data_dir"/2010_03_22_Fasteris_MG20_Gifu_transcripts/100226_s_8_2_seq_GHD-2.txt"

mkdir $work_dir"/tophat"

### print the file names
echo "Reference file: "$ref >>$logfile
echo "read 1:"$read1 >>$logfile
echo "read 2:"$read2 >>$logfile

### run tophat
tophat --bowtie1 --num-threads $cores -I 25000 -o $work_dir"/tophat" $ref $read1
    $read2
echo "tophat is done" >>$logfile

### bam_file
bam="accepted_hits.bam"
sam="accepted_hits.sam"

### convert bam to sam

samtools view $work_dir"/"$bam > $work_dir"/"$sam

### run cufflink

cufflinks --pre-mrna-fraction 0.5 --small-anchor-fraction 0.01 --min-frags-per-
    transfrag 5 --overhang-tolerance 20 --max-bundle-length 10000000 --min-intron-
    length 20 --trim-3-dropoff-frac 0.01 --max-multiread-fraction 0.99 --no-
    effective-length-correction --no-length-correction --multi-read-correct --upper
    -quartile-norm  --total-hits-norm --max-mle-iterations 10000  --max-intron-
    length 50000 --no-update-check -p $cores -o $work_dir $work_dir"/"$sam
echo "cufflinks is done" >>$logfile
```

2..2.2 De novo assembled transcripts

GMAP was used to map the assembled transcripts on the genome with maximum of 3000 base-pair as the intron length.

```
### MG20
nice -n 19 gmap -d 'Ljchr0-6_pseudomol_20120830.chlo.mito.fa' --intronlength=30000
    --nthreads=3 --format=2 /u/vgupta/01_genome_annotation/11_gmap/data/
    MG20_mRNA_illumina_denovo.fa > /u/vgupta/01_genome_annotation/11_gmap/
    MG20_mRNA_illumina_denovo.gff
```

2..2.3 Lotus Gene indices

Lotus gene indices were downloaded from `http://compbio.dfci.harvard.edu/tgi/cgi-bin/tgi/gimain.pl?gudb=l_japonicus` and were mapped back to genome using the similar approach to assembled transcripts.

```
nice -n 19 gmap -d 'Ljchr0-6_pseudomol_20120830.chlo.mito.fa' --intronlength=30000
    --nthreads=4 --format=2 LJGI.051810 > LJGI.051810.gff3
```

2..2.4 Ab-initio predictions

We have used three ab-initio predictor to find the genes that might be less expressed and not predicted by the RNA-seq approach.

2..2.4.1 Augustus

Augutus can be used as either with pre-trained parameters for a specie or it can be trained with the given a set of protein coding gene structures. As we did not have any pre-trained parameters for

Lotus japonicus, we have trained parameters using the gene structure from the RNA-seq based predictions. Augustus version 2.6.1 was used.

```
### Augustas with the training with cufflinks output

### covert to gff3 format
perl gtf2gff.pl < 05_transcripts.gtf --gff3 --printExon --out=05.gff

### converting gff3 to gb format
cd /u/vgupta/01_genome_annotation/14_augustus
perl Vikas_gff2gbSmallDNA.pl 05.gff Ljchr0-6_pseudomol_20120830.scaf.fa 20000 05.gb


### generating test set
perl /u/vgupta/01_genome_annotation/tools/augustus.2.6.1/scripts/randomSplit.pl 05.
    gb 100

### CREATE A META PARAMETERS FILE FOR YOUR SPECIES
perl /u/vgupta/01_genome_annotation/tools/augustus.2.6.1/scripts/new_species.pl --
    species=Lotus_cuff
### MAKE AN INITIAL TRAINING
# edit /u/vgupta/01_genome_annotation/tools/augustus.2.6.1/config/species/
stopCodonExcludedFromCDS true # make this 'true' if the CDS includes the stop codon
     (training and prediction)
etraining --species=Lotus_cuff 05.gb.train

### testing augustas
augustus --species=Lotus_cuff 05.gb.test

### optimise the parameters
RUN THE SCRIPT optimize_augustus.pl
perl /u/vgupta/01_genome_annotation/tools/augustus.2.6.1/scripts/optimize_augustus.
    pl --species=Lotus_cuff 05.gb.train

### testing augustas
augustus --species=Lotus_cuff 05.gb.test

### run the prediction
augustus --gff3=on --species=Lotus /u/vgupta/lotus_3.0/Ljchr0-6_pseudomol_20120830.
    chlo.mito.fa >augustus.gff3
```

2..2.4.2 Glimmer

Glimmer gene predictor has been used with the trained parameters for the *Arabidopsis thailiana* plant. GlimmerHMM version 3.0.1 was used here to predict another set of gene models.

```
### using trained data
/u/vgupta/01_genome_annotation/tools/GlimmerHMM/trained_dir/arabidopsis

glimmerhmm_linux /u/vgupta/lotus_3.0/Ljchr0-6_pseudomol_20120830.chlo.mito.fa /u/
    vgupta/01_genome_annotation/tools/GlimmerHMM/trained_dir/arabidopsis -g >
    20121014_glimmerHMM_arabidopsis.gff3
```

2..2.4.3 GeneMark

GeneMark was the third ab-initio predictor we used for the gene models here. GeneMark does not require a pre-trained set of parameters or an user supplied gene structure for fine-tuning instead it usage a small fraction of genome(10 MB) to train the prediction parameters. We used GeneMark-ES version 2.3e.

```
# works only on genome cluster
perl /home/vgupta/01_genome_annotation/tools/gm_es_bp_linux64_v2.3e/gmes/
    vikas_gm_es.pl /home/vgupta/01_genome_annotation/01_genome/Ljchr0-6
    _pseudomol_20120830.scaf.fa

### zombie
/u/vgupta/01_genome_annotation/tools/maker/src/bin/genemark_gtf2gff3 sample.
    genemark_hmm.gtf > sample.genemark_hmm.gff3
```

2..3 Gene model selection and filtering

Six set of gene models, when put together contained a high degree of redundancy. Creating a consesus model is a non-trivial for multi-exonic genes as often the existing software mis-predict the individual exons and merging multiple exons may results into wrong set of exons for a given gene. We used RNA-seq data to find multiple wrongly annotated genes using consensus method used by EVM.

We used a heirarchial selection approach based on the confidence in the gene model evidence quality. RNA-seq based gene models were considered with the best gene strcture so these were assigned highest preriority.

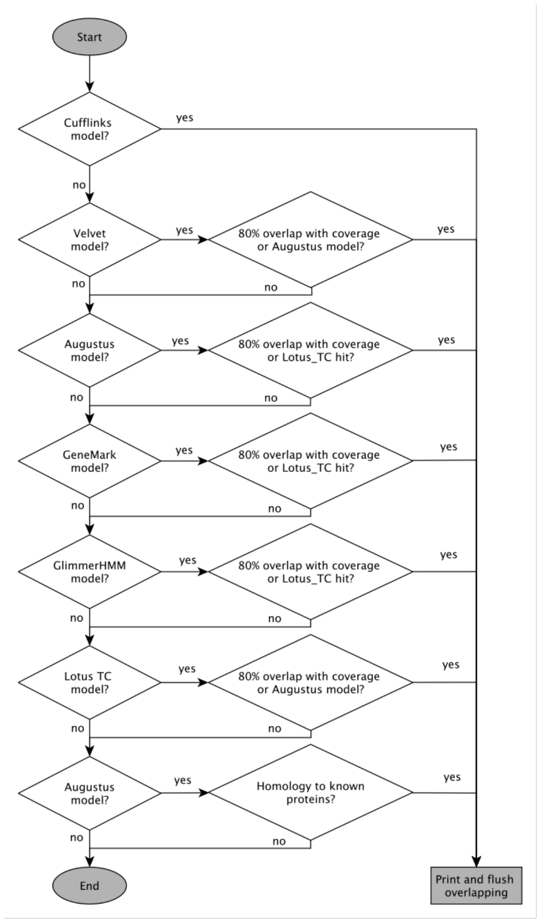

Start
Cufflinks model?
yes
no
Velvet model?
yes
80% overlap with coverage or Augustus model?
yes
no
no
Augustus model?
yes
80% overlap with coverage or Lotus_TC hit?
yes
no
no
GeneMark model?
yes
80% overlap with coverage or Lotus_TC hit?
yes
no
no
GlimmerHMM model?
yes
80% overlap with coverage or Lotus_TC hit?
yes
no
no
Lotus TC model?
yes
80% overlap with coverage or Augustus model?
yes
no
no
Augustus model?
yes
Homology to known proteins?
yes
no
no
End
Print and flush overlapping

Figure 1: Hierarchial genemodel filtering

```
### Merge files
cat 05_transcripts.evm.gff3 MG20_mRNA_illumina_denovo.gff3 LJGI.051810.updated.gff3
     augustus.EVM.gff3 accepted_hits.bam.pileup.2.updated.gtf genemark_hmm.EVM.gff3
     20121017_glimmerHMM_arabidopsis.EVM.gff3> 20121108_combined.gff3

### combine files remove overlapping
python /u/vgupta/script/python/21q_combine_GTF.py 20121108_combined.gff3.refined >
    20121116_merged_gene_models.gff3

### add known protein coding genes from NCBI
cat 20121116_merged_gene_models.gff3 20121227_conserved_proteins_mRNA_seq.gff3 >
    20130223_TAU_conserved.gff3

### sort the file
cd /u/vgupta/01_genome_annotation/21_add_conseved_genes/run2
python ~/script/python/21v_format_gff3.py -i 20130223_TAU_conserved.gff3 > 20130223
    _TAU_conserved.sorted.gff3
```

2..4 Coding potential prediction

Genemodels based on the RNA-seq/transcrptional evidences did not have a coding region predicted. Using the non-redundant comined GFF3 files we extracted all the transcript and using the TAU tool, we predicted the protein coding potential for all the genes.

```
### Run TAU
cd /u/vgupta/01_genome_annotation/28_final_gene_set/run9
genome="/u/vgupta/lotus_3.0/lj_r30.fa"
gff3="../run6/20130223_TAU_conserved.sorted.gff3"
nohup nice -n 19 python /u/vgupta/script/python/21t_tau_20130611.py -f $genome -g
    $gff3 &

SCRIPTDIR="/u/vgupta/script"

### sort the output
python "$SCRIPTDIR"/python/21v_format_gff3.py -i TAU_genemodel.gff3 > TAU_genemodel
    .gff3.sorted

### correct for UTRs
python "$SCRIPTDIR"/python/21ae_correct_UTR.py -i TAU_genemodel.gff3.sorted >
    TAU_genemodel.gff3.sorted.correctUTR

### correct for strands
python "$SCRIPTDIR"/python/21al_correct_strand.py -i TAU_genemodel.gff3.sorted.
    correctUTR > TAU_genemodel.gff3.sorted.correctUTR.correctStrand


### make files with old ids
gff3="TAU_genemodel.gff3.sorted.correctUTR.correctStrand"

### fix cufflinks for one gene- one transcript prob
python "$SCRIPTDIR"/python/21ak_remove_extra_cuff_gene.py -i $gff3 > $gff3.
    cuffFixed

### sort the GFF3 file
python "$SCRIPTDIR"/python/103_sort_gff_blocks.py -i $gff3.cuffFixed >  $gff3.
    cuffFixed.sorted

### correct phase
```

```
python "$SCRIPTDIR"/python/109_AddPhaseGFF3.py -i $gff3.cuffFixed.sorted> 20130627.
    Lj3.0.gff3.correctPhage
grep -v 'UTR' 20130627.Lj3.0.gff3.correctPhage | awk '$4<$5' > 20130627.Lj3.0.gff3.
    correctPhage.noUTR
key="20130627.Lj3.0.gff3.correctPhage.noUTR"
GFF3="20130627.Lj3.0.gff3.correctPhage.noUTR"
gffread $GFF3 -g $GENOME -w $key.exon.fa
gffread $GFF3 -g $GENOME -y $key.protein.fa
gffread $GFF3 -g $GENOME -x $key.cds.fa

### Count Ns (gap region) between the genes
python "$SCRIPTDIR"/python/21ah_count_N_between_genes.py -g  $gff3 -f $GENOME > "
    $gff3"_GeneGaps_between_genes.txt
#/u/vgupta/01_genome_annotation/26_addType/run5/20130313_lj3.0
    _GeneGaps_between_genes.txt

### add the new names
python "$SCRIPTDIR"/python/21ai_modify_gene_names.py -g $gff3 -n  /u/vgupta/01
    _genome_annotation/26_addType/run5/20130313_lj3.0_GeneGaps_between_genes.txt >
    $gff3.new_names


### get the new names from the old MySQL table
mysql -u vgupta -p gene_models < get_old_id.sql > 20130711_old_newID.txt

### rename using old ids
python "$SCRIPTDIR"/python/21ak_update_GFF3_IDsOnly.py -i $gff3.new_names -l
    20130711_old_newID.txt > $gff3.correct_names
# remember to update new ids for published proteins

### fix cufflinks for one gene- one transcript prob
python "$SCRIPTDIR"/python/21ak_remove_extra_cuff_gene.py -i $gff3.correct_names >
    $gff3.correct_names.cuffFixed

### sort the GFF3 file
python "$SCRIPTDIR"/python/103_sort_gff_blocks.py -i  $gff3.correct_names.cuffFixed
     >  20130627.Lj3.0.gff3

### correct phase
python "$SCRIPTDIR"/python/109_AddPhaseGFF3.py -i 20130627.Lj3.0.gff3 > 20130627.
    Lj3.0.gff3.correctPhage
grep -v 'UTR' 20130627.Lj3.0.gff3.correctPhage | awk '$4<$5' > 20130627.Lj3.0.gff3.
    correctPhage.noUTR
key="20130627.Lj3.0.gff3.correctPhage.noUTR"
GFF3="20130627.Lj3.0.gff3.correctPhage.noUTR"
gffread $GFF3 -g $GENOME -w $key.exon.fa
gffread $GFF3 -g $GENOME -y $key.protein.fa
gffread $GFF3 -g $GENOME -x $key.cds.fa
```

2..4.1 Functional annotation

Blastp

All the Lotus protein coding genes were annotated usig Blast veersion 2.2.26 against the non-redundant gene set.

```
cd /home/vgupta/01_genome_annotation/28_FinalSet/01_proteins
perl ~/script/perl/fasta_splitter.pl -n-parts-sequence 1000 20130802_Lj30_proteins.
    fa
source /com/extra/BLAST/2.2.26/load.sh
```

```
file_dir="/home/vgupta/01_genome_annotation/28_FinalSet/01_proteins"
db="/home/vgupta/80_blastDatabases/nr"
out="/home/vgupta/01_genome_annotation/28_FinalSet/01_proteins/blastout"
for f in *.part* ;  do qx -q normal -n 1 -c 7 -v "blastp  -max_target_seqs=100 -
    num_threads=6 -db $db -query $file_dir/$f -outfmt 6 -out $out/$f.blastout" |
    qsub -l walltime=5:00:00 -N "qsub.script".$f; done
```

IPRScan
Protein coding genes were scan for known domains using IPRScan version 4.8.

```
#!/bin/bash
#PBS -l nodes=1:ppn=1
#PBS -q normal

cd /home/vgupta/01_genome_annotation/25_InterProScan
run_no=01
logfile=`date "+20%y%m%d_%H%M"`'.logfile'.$run_no

iprscan -cli -verbose -i Ljr_cds_protein.Ljr3.0.20130102.refined.part-"$run_no".fa
    -o Ljr_cds_protein.Ljr3.0.20130102.refined.part-"$run_no".fa.out -format raw -
    goterms -iprlookup
```

3. Lotus accession resequencing

3..1 Variant calling and filtering

We have analysed a polymorphic variations within the Lotus population. MG20, Gifu together Burttii together with other 28 japanese Lotus accessions were analyzed using the following GATK workflow.

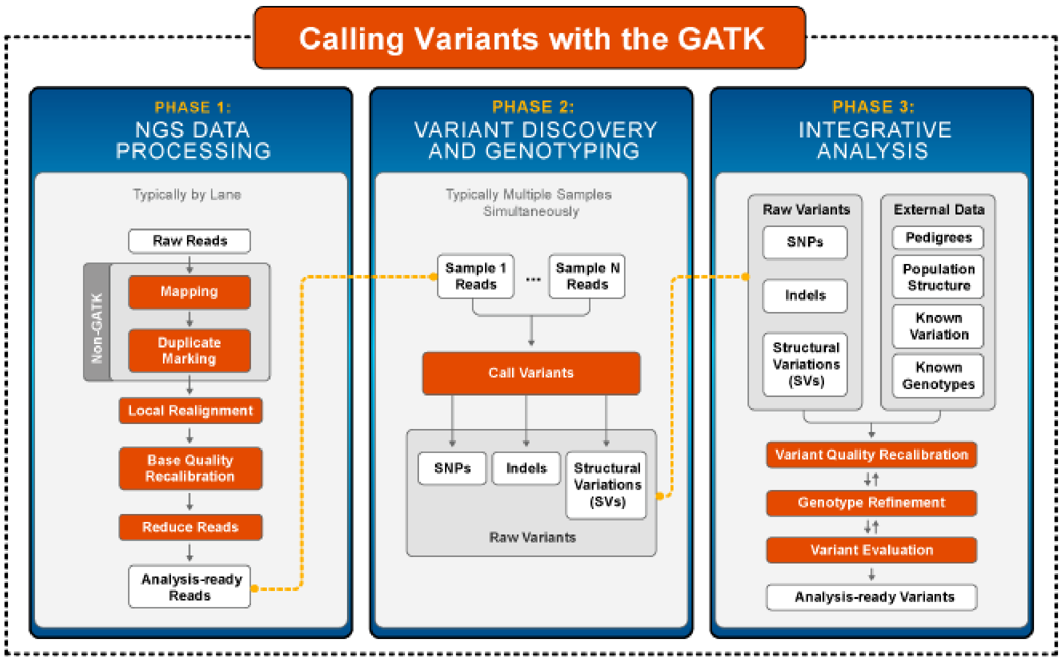


Figure 2: Three sections of GATK pipeline

We have divided work here into four major points:
1. Initial read mapping
2. Local realignment around indels
3. Base quality score recalibration
4. SNP and indel detection

3..1.1 Fastq Mapping

Fastq files were mapped to the Lotus genome 3.0 using the BWA version-0.7.5a-r405 with default parameters and with appropriate insert size for the pair-end libraries. GATK pipeline does not support files without read groups so these were added whenever necessary with Picard *AddOrReplaceReadgroup* function. All the mapped files are placed at genome.au.dk. `/home/vgupta/LotusGenome/100_data/01_Jin_BamFile/02_withReadGroup`.

```
source /com/extra/FastQC/0.10.1/load.sh
source /com/extra/samtools/0.1.18/load.sh
python ~/script/python/115_MapFastq.py \
-f /home/vgupta/LotusGenome/100_data/20130801_Japan_sequencing/data1 \
-x fastq \
-r /home/vgupta/LotusGenome/ljr30/lj_r30.fa
```

3..1.2 Duplicate Marking

Duplicates were filtered using Picard toolkit version 1.96. Two things to remember when using Picard for duplicate filtering:
1. Set *VALIDATION_STRINGENCY=LENIENT* otherwise Picard will not accept umapped read positions.
2. Use *TMP_DIR* variable to a folder where you have sufficient space.

3..1.3 Realiging the reads

Duplicate filtered reads are mapped back on the genome using RealignerTargetCreator and IndelRealigner commands from the GATK. All the fastq files must follow the sanger quality encoding. IndelRealigner locally aligns the reads to minimize the mismatches. Also many reads are misaligned due to presence of insertions and deletions, leading to many false discoveries of SNPs.

3..1.4 Unified genotyper

Realigned bam files are parsed through the unified genotyper to procude a primary list of snp and indel list. -glm BOTH option must be used to predict indels together with SNPs. A higher degree of calling confidence cut-off can be used to insure minimal false positives. Unified genotyper uses a Bayesian genotype likelihood model to find genotypes and allele frequency in a population of N samples. It provides a posterior probability of there being a segregating variant allele at each locus as well as for the genotype of each sample.

3..1.5 Base Recalibrator

In this step, base quality scores are recalibrated. Given a set of high confidence SNPs from the earlier step, program calculates an empirical probability of error given the particular covariates seen at this site, where p(error) = num mismatches / num observations.

```
### read group using python script
qx -q normal -n 1 -c 16 -v "python ~/script/python/117_addReadGroup.py --cores 15 -
    i /home/vgupta/LotusGenome/100_data/01_Jin_BamFile/temp/ -CN CARB -PL ILLUMINA
    -DS Burtii_20130605.bam -DT 20130822 -PI 0" | qsub -N qsub.script

source /com/extra/samtools/0.1.19/load.sh
source /com/extra/picard/1.74/load.sh
source /com/extra/GATK/2.6-4/load.sh
python ~/script/python/116_runGATK.py \
-b Burtii_20130605.bam, Gifu_20130609.bam, mg004.bam, mg010.bam, mg012.bam, mg019.
    bam, mg023.bam, mg036.bam, mg049.bam, mg051.bam, mg062.bam, mg072.bam, mg073.
    bam, mg077.bam, mg080.bam, mg082.bam, mg083.bam, mg086.bam, mg089.bam, mg093.
    bam, mg095.bam, mg097.bam, mg101.bam, mg107.bam, mg109.bam, mg112.bam,
    MG20_genomic_20130609.bam \
-r /home/vgupta/LotusGenome/ljr30/lj_r30.fa \
-p /com/extra/picard/1.74/jar-bin \
-g /com/extra/GATK/2.6-4/jar-bin \
-t 10 \
s
```

4. Analysis of phylogenetic relationships and LD

4..1 Phylogenetics

A phylogenetic tree was created based on the polymorphic positions among all the accession except Burttii. Phylogenetic distances were based on the genotypic difference. There are three possible genotypes 0/0 referece allele, 0/1 heterozygous allele and 1/1 alternative allele, these three cases were assigned as 0, 0.5 and 1 unit distance, respectively.

```
cd /home/vgupta/LotusGenome/04_Phylogenetics
## genome
cd  /home/vgupta/LotusGenome/03_VariantStig/02_withoutBurttii
python ~/script/python/21bc_GenotypicDistance.py -i 20130905.snp.vcf.markers.snps.
    inbreed.ref.NoBurttiiGifu
```

4..2 LD

A final list of markers was created by filtering the GATK output against previously known marker list. Most of the polymorphic variation was contributed by Burtti which have higher evolutionary distance compare to other accessions. Polymorphic variations were removed if caused only by Burttii in the calculation of LD to increase high quality SNPs, we also removed chromosome 0 as the genomic fragments on this chromosome are not in correct order.

We used vcftools version 0.1.9 was used to calculate the linkage disequilibrium within the window of 500,000 base-pairs. A total of 200,000 randomly selected snp-pairs were plotted using R.

```
### Without Burttii
source /com/extra/vcftools/0.1.9/load.sh
awk '$1!="chr0"' /home/vgupta/LotusGenome/03_VariantStig/02_withoutBurttii
    /20130905.snp.vcf.markers.snps.inbreed.ref.NoBurttiiGifu > /home/vgupta/
    LotusGenome/03_VariantStig/02_withoutBurttii/20130905.snp.vcf.markers.snps.
    inbreed.ref.NoBurttiiGifu.nochr0
file="/home/vgupta/LotusGenome/03_VariantStig/02_withoutBurttii/20130905.snp.vcf.
    markers.snps.inbreed.ref.NoBurttiiGifu.nochr0"
vcftools --vcf $file --geno-r2 --ld-window-bp 500000 --out 20131219_500kb_LD

### 31 individuals
cat 20131219_500kb_LD.geno.ld | awk '$4>30' | awk '{FS="\t";OFS="\t";}{print $1,$3-
    $2,$5}' > 20131219_500kb_LD.geno.ld.chr1-6.31

 shuf 20131219_500kb_LD.geno.ld.chr1-6.31| perl -pe '$_ =~ s/chr1/chr/' \
| perl -pe '$_ =~ s/chr2/chr/' \
| perl -pe '$_ =~ s/chr3/chr/' \
| perl -pe '$_ =~ s/chr4/chr/' \
| perl -pe '$_ =~ s/chr5/chr/' \
| perl -pe '$_ =~ s/chr6/chr/' | head -n 500000 > 20131219_500kb_LD.geno.ld.chr1
    -6.31.500000

### R code
setwd('~/Desktop/03_Lotus_annotation/2013_week50/')
infile="20131219_500kb_LD.geno.ld.chr1-6.31.w100.s100"
pdf(paste0(infile,'.pdf'),height=10,width=20)
d<-read.table(infile)
names(d)<-c("chr","pos","R2")

infile2='20131219_500kb_LD.geno.ld.chr1-6.31.1000000'
d2<-read.table(infile2)
names(d2)<-c("chr","pos","R2")


library(ggplot2)
require(mgcv)
d2 <- d2[1:200000,]
ggplot(d, aes(x=pos, y=R2, col=chr)) + geom_point(data=d2, col='gray60',size=1) +
    theme_classic() + xlim(0,200000)+ stat_smooth(method = "loess", formula = y ~ x
    , colour="#CC0000", size = 3,  span = 0.001, se = FALSE)

dev.off()
```

5. snpEffect

Final set of markers from the GATK analysis of 31 accessions were subject to annotation process. Aim was to assign the genic region category and potential effect of the polumorphic variation on the protein if SNP is in protein coding region. We used SNPeff 3.1m to analyze the distribution of variants across exon, intron, coding region and intergenic space as well as based on the Lotus 3.0 protein coding regions, variants were also defined as sysnonymous and non-synonymous.

```
dir="/u/vgupta/01_genome_annotation/tools/snpEff"
data_dir="/u/vgupta/01_genome_annotation/28_final_gene_set/run6"
cd /u/vgupta/01_genome_annotation/tools/snpEff/data/lj3.0
cp $data_dir/Lj3.0.gff3.refined ./Lj3.0.gff3
mv Lj3.0.gff3 genes.gff
### database built
cd $dir
java -jar snpEff.jar build -gff3 -v lj3.0

### snps
cd /u/vgupta/08_snpEff/01_GATKsnps
file="20130905.snp.vcf.markers.snps.inbreed.ref"
java -Xmx4g -jar /u/vgupta/01_genome_annotation/tools/snpEff/snpEff.jar eff -c /u/
    vgupta/01_genome_annotation/tools/snpEff/snpEff.config -v lj3.0 $file > $file.
    snpEff
mv snpEff_summary.html 20130923_snp.snpEff_summary.html
mv snpEff_genes.txt 20130923_snp.snpEff_genes.txt

### indels
cd /u/vgupta/08_snpEff/01_GATKsnps
file="20130905.snp.vcf.markers.indels.inbreed.ref"
java -Xmx4g -jar /u/vgupta/01_genome_annotation/tools/snpEff/snpEff.jar eff -c /u/
    vgupta/01_genome_annotation/tools/snpEff/snpEff.config -v lj3.0 $file > $file.
    snpEff
mv snpEff_summary.html 20130923_indel.snpEff_summary.html
mv snpEff_genes.txt 20130923_indel.snpEff_genes.txt
```

6. Prank based alignments

This document is a quick description for the positive selection test pipeline. Method is very similar to suggested by Victor Albert and we have used the codeML control and test parameters supplemented by his lab. I will attach maximum information but if anything missing you can always fetch script from the my GitHub `https://github.com/vikas0633/python`.

6..1 Input data

We have downloaded three cds fasta files for Arabidopsis, Glycine max and Medicago.

Lotus CDS fasta file is at:
˜/vgupta/lotus_3.0/20130802_Lj30_CDS.fa

Ortholog groups were extrated from results provided by Vic are at:
˜/vgupta/01_genome_annotation/32_prank/01_PositiveSelectionCandidate/20131213_orthoGroups.lotus.txt

```
### Arabidosis
wget ftp://ftp.arabidopsis.org/home/tair/Sequences/blast_datasets/TAIR10_blastsets/
    TAIR10_cds_20101214_updated
### Glycin max
wget ftp://ftp.plantgdb.org/download/Genomes/GmGDB/Gmax_109_cds.fa.gz
### Medicago
```

```
wget ftp://ftp.jcvi.org/pub/data/m_truncatula/Mt4.0/Annotation/Mt4.0v1/Mt4.0
    v1_GenesCDSSeq_20130731_1800.fasta
```

6..2 Installation

6..2.1 Prank

http://code.google.com/p/prank-msa/wiki/PRANK

6..2.2 Gblocks

http://bioweb2.pasteur.fr/docs/gblocks/Installation

6..2.3 Codeml

http://abacus.gene.ucl.ac.uk/software/pamlX-1.1-x11-x86$_6$4.*tgz*

6..2.4 pchisq

http://stat.ethz.ch/R-manual/R-patched/library/stats/html/Chisquare.html

6..2.5 seqret

http://emboss.open-bio.org/rel/dev/apps/seqret.html

6..3 Data Analsis

6..3.1 Spliting fasta

First step is to create individual fasta file where each contains homologous sequences for four species and it done using custom python script.

```
### grep orthogroup sequences
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/01_fasta
python ~/script/python/21bf_ortho2fasta.py -i ../20130103_orthoGroups.lotus.txt -f
    /array/users/vgupta/01_genome_annotation/32_prank/01_PositiveSelectionCandidate
    /01_fasta/02_cds/Lj_Gm_Mt_At.cds.fa
```

6..3.2 Creating alignments

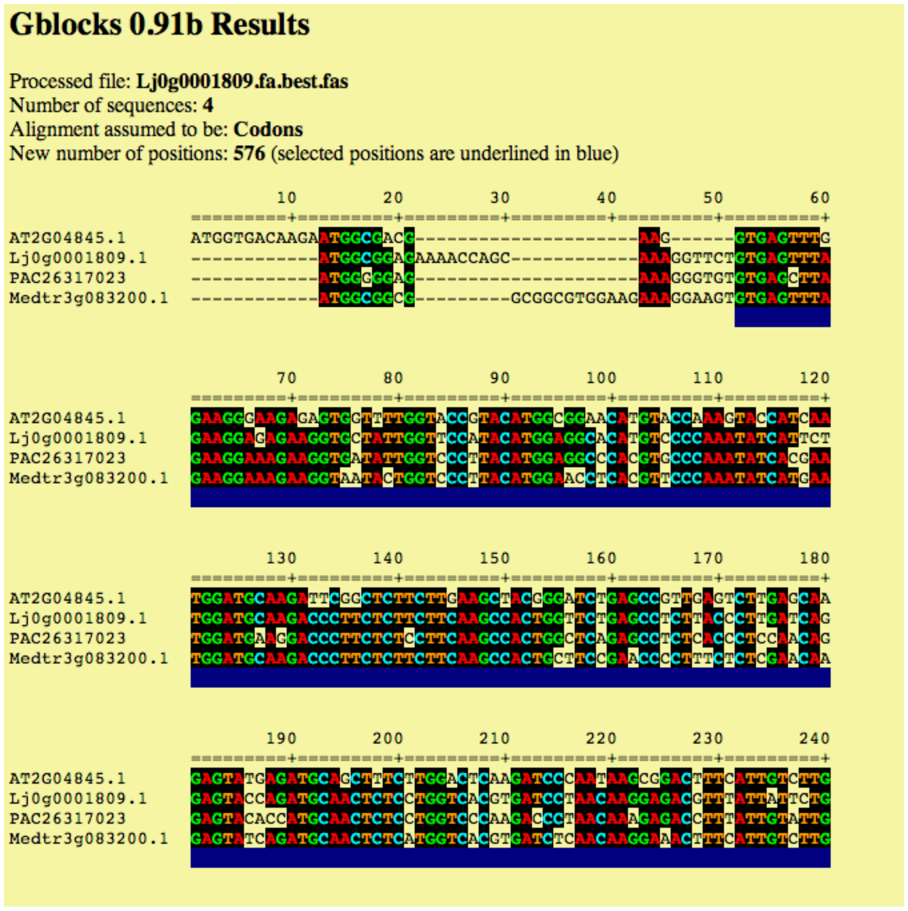
While running the Prank remember to used -codon option for codon based alignments.

```
### run prank
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/01_fasta
for file in Lj*.fa
do
prank -d=$file -o=$file -showall -codon -F
done
```

6..3.3 Filtering alignments

Again remember to use -t=c for the codon based filtering. An example output from the Gblocks is shown in the figure 1, blue region is cosidered as good alignment region.

```
### run Gblocks
for file in *.best.fas
do
Gblocks $file -t=c -d=y
done
```



Gblocks 0.91b Results

Processed file: **Lj0g0001809.fa.best.fas**
Number of sequences: **4**
Alignment assumed to be: **Codons**
New number of positions: **576** (selected positions are underlined in blue)

```
                          10        20        30        40        50        60
                 =========+=========+=========+=========+=========+=========+
AT2G04845.1      ATGGTGACAAGAATGGCGACG---------------------AAG------GTGAGTTTG
Lj0g0001809.1    ------------ATGGCGGAGAAAACCAGC------------AAAGGTTCTGTGAGTTTA
PAC26317023      ------------ATGGGGGAG---------------------AAAGGGTGTGTGAGCTTA
Medtr3g083200.1  ------------ATGGCGGCG---------GCGGCGTGGAAGAAAGGAAGTGTGAGTTTA


                          70        80        90       100       110       120
                 =========+=========+=========+=========+=========+=========+
AT2G04845.1      GAAGGGAAGAGAGTGGTTTTGGTACCGTACATGGCGGAACATGTACCAAAGTACCATCAA
Lj0g0001809.1    GAAGGAGAGAAGGTGCTATTGGTTCCATACATGGAGGCACATGTCCCCAAATATCATTCT
PAC26317023      GAAGGAAAGAAGGTGATATTGGTCCCTTACATGGAGGCCCACGTGCCCAAATATCACGAA
Medtr3g083200.1  GAAGGAAAGAAGGTAATACTGGTCCCTTACATGGAACCTCACGTTCCCAAATATCATGAA


                         130       140       150       160       170       180
                 =========+=========+=========+=========+=========+=========+
AT2G04845.1      TGGATGCAAGATTCGGCTCTTCTTGAAGCTACGGGATCTGAGCCGTTGAGTCTTGAGCAA
Lj0g0001809.1    TGGATGCAAGACCCTTCTCTTCTTCAAGCCACTGGTTCTGAGCCTCTTACCCTTGATCAG
PAC26317023      TGGATGAAGGACCCTTCTCTCCTTCAAGCCACTGGCTCAGAGCCTCTCACCCTCCAACAG
Medtr3g083200.1  TGGATGCAAGACCCTTCTCTTCTTCAAGCCACTGCTTCCGAACCCCTTTCTCTCGAACAA


                         190       200       210       220       230       240
                 =========+=========+=========+=========+=========+=========+
AT2G04845.1      GAGTATGAGATGCAGCTTTCTTGGACTCAAGATCCCAATAAGCGGACTTTCATTGTCTTG
Lj0g0001809.1    GAGTACCAGATGCAACTCTCCTGGTCACGTGATCCTAACAAGGAGACGTTTATTATTCTG
PAC26317023      GAGTACACCATGCAACTCTCCTGGTCCCAAGACCCTAACAAAGAGACCTTTATTGTATTG
Medtr3g083200.1  GAGTATCAGATGCAACTCTCATGGTCACGTGATCTCAACAAGGAAACTTTCATTGTCTTG
```

Figure 3: GblocksExample

6..3.4 Data munging

Gblocks outputs genbank format files and CodeML requires Phylip format file so a bit data format transformation is done using below code. Also I have added 1 to all Lotus branches using sed regular expession.

```
## gb to fas
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/04_gb
for file in *-gb
do
seqret -sequence $file -outseq $file.fas
done

cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/04_gb
## fas to phy
for file in *.fas
do
perl ~/script/perl/MFAtoPHY.pl $file
done

### add the # in the end file
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/03_dnd
for file in *.best.dnd
do
sed -e 's/\(Lj[a-zA-Z0-9]*\.[a-zA-Z0-9]\:[a-zA-Z0-9]*\.[a-zA-Z0-9]*\)/\1 #1/' $file
    > $file.1
done
```

6..3.5 Running CodeML

CodeML runs only one alignement at a time and each time it requires a parameter file with individual file path. Follwing code loops over files one after another, replacing file paths in the parameter files and running CodeML.

```
### Control
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/06
    _model_bs_ctrl
for file in /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates
    /05_phy/*.phy
do
echo $file
id=`echo $file | awk -F"/" '{print $NF}' | awk -F"." '{print $1}'`
seqfile=$file
treefile="/array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/03
    _dnd/"$id".fa.best.dnd.1"
outfile=$id".out"
awk -v var="$seqfile" ' {split ($0, arr, "="); if ($0~/seqfile/) print arr[1]"="
    var; else print $0};' 06_PS_control.txt | awk -v var="$treefile" ' {split ($0,
    arr, "="); if ($0~/treefile/) print arr[1]"=" var; else print $0};'| awk -v var
    ="$outfile" ' {split ($0, arr, "="); if ($0~/outfile/) print arr[1]"=" var;
    else print $0};' > temp.txt
codeml temp.txt
done

### Test
cd /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/07
    _model_ps_test
for file in /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates
    /05_phy/*.phy
```

```
do
echo $file
id=`echo $file | awk -F"/" '{print $NF}' | awk -F"." '{print $1}'`
seqfile=$file
treefile="/array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/03
    _dnd/"$id".fa.best.dnd.1"
outfile=$id".out"
awk -v var="$seqfile" ' {split ($0, arr, "="); if ($0~/seqfile/) print arr[1]"="
    var; else print $0};' 07_model_ps_test.txt | awk -v var="$treefile" ' {split (
    $0, arr, "="); if ($0~/treefile/) print arr[1]"=" var; else print $0};'| awk -v
     var="$outfile" ' {split ($0, arr, "="); if ($0~/outfile/) print arr[1]"=" var;
     else print $0};' > temp.txt
codeml temp.txt
done
```

6..3.6 Parsing CodeML output

To do a loglikelihood test, we needed the likelihood values from the control and test CodeML output files and use a chi-square test with one degree of freedom to test the significance.

```
### fetch the likelihood values
for file in /array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates
    /06_model_bs_ctrl/*.out;
do
id=`echo $file | awk -F"/" '{print $NF}' | awk -F"." '{print $1}'`
file="/array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/06
    _model_bs_ctrl"/"$id".out
Ctrl_lnL=`grep lnL $file | awk '{split($0, a, " "); print a[5]}'`
file="/array/users/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates/07
    _model_ps_test"/"$id".out
test_lnL=`grep lnL $file | awk '{split($0, a, " "); print a[5]}'`
echo -n $id,$Ctrl_lnL,$test_lnL;
echo ;
done
```

6..3.7 Chi-square test

Chi-square test was done using a R fucntion called pchiseq.
`http://www.ndsu.edu/pubweb/~mcclean/plsc431/mendel/mendel4.htm`

```
d <- read.table('/Volumes/vgupta/01_genome_annotation/32_prank/02_AllGeneCandidates
    /08_PS_compare/20140106_lnL.txt', sep = ',')
diff_df = 1
colnames(d) <- c("LjID", "Control_lnL","Test_lnL")

d$diff.2 <- 2*abs(d$Test_lnL - d$Control_lnL)
d$p_value <- 1 - pchisq( d$diff.2 , df =  diff_df)

write.table(d, file='/Volumes/vgupta/01_genome_annotation/32_prank/02
    _AllGeneCandidates/08_PS_compare/20140106_lnL.p_value', sep="\t", row.names =F,
     quote = F)
```

6..4 Comparing with Vic's list

P-values from the Prank and Muscle alignment based resutls has been loaded into a MySQL database. There were a total 281 candidate significant in both list.

```
### Make MySQL table
CREATE TABLE `20140106_PS_comp`  (Lj30_ID VARCHAR(100), Prank FLOAT, Vic FLOAT);
LOAD DATA LOCAL INFILE '/array/users/vgupta/01_genome_annotation/32_prank/02
    _AllGeneCandidates/08_PS_compare/20140106_lnL.comp.txt' INTO TABLE  `20140106
    _PS_comp`;
CREATE INDEX `20140106_PS_comp.index` ON `20140106_PS_comp` (Lj30_ID);


mysql> select count(*) from 20140106_PS_comp WHERE Prank<0.01  AND Vic<0.01 ;
+----------+
| count(*) |
+----------+
|      281 |
+----------+
1 row in set (0.01 sec)
```

7. Python Script

Listing 1: Map Fastq

```
#----------------------------------------------------------+
#                                                           |
# 115_MapFastq.py - Script to Map Fastq files               |
#                                                           |
#----------------------------------------------------------+
#                                                           |
# AUTHOR: Vikas Gupta                                       |
# CONTACT: vikas0633@gmail.com                              |
# STARTED: 09/06/2013                                       |
# UPDATED: 09/06/2013                                       |
#                                                           |
# DESCRIPTION:                                              |
#                                                           |
# LICENSE:                                                  |
#  GNU General Public License, Version 3                    |
#  http://www.gnu.org/licenses/gpl.html                     |
#                                                           |
#----------------------------------------------------------+

# Example:
# python ~/script/python/115_MapFastq.py -i 02_Stegodyphous_cdna.refined.fa.orf.
    tr_longest_frame


### import modules
import os, sys, getopt, re, glob


### global variables
global ref, folder, extension, fastqc, mapper, threads, seed_length, mismatches,
    single, index, compress_extension, uncompress

### make a logfile
import datetime
now = datetime.datetime.now()
o = open(str(now.strftime("%Y-%m-%d_%H%M."))+'logfile','w')



### write logfile

def logfile(infile):
    o.write("Program used: \t\t%s" % "115_MapFastq.py"+'\n')
    o.write("Program was run at: \t%s" % str(now.strftime("%Y-%m-%d_%H%M"))+'\n')
    o.write("Infile used: \t\t%s" % infile+'\n')


def help():
    print '''
            python 115_MapFastq.py
                                    -r <ref> [reference sequence]
                                    -f <folder> [folder1, folder2, .., folderN]
                                    -x <extension> [default: fastq]
                                    -q <fastqc> [#runs fastq rather than mapping]
                                    -p <mapper> [default: bwa]
                                    -t <threads> [default: 6, numbers of core to be used]
                                    -l <seed_length> [default: 28, seed length to be used in mapping]
                                    -m <mismatches> [default: 2, mismatches allowed in the seed]
                                    -s <single> [default: Pair End alignments]
                                    -i <index> [Option to create index for Reference Sequence]
                                    -u <uncompress> [default extention: bz2]

            Fastq pair must be specified with "*_R1_*" and "*_R2_*"
            '''
    sys.exit(2)

### main argument to

def options(argv):
    global ref, folder, extension, fastqc, mapper, threads, seed_length, mismatches, single, index, compress_extension, uncompress
    ref = ''
    folder = ''
    extension = 'fastq'
    fastqc = False
    mapper='bwa'
    threads = 6
    seed_length = 28
    mismatches = 2
    single = False
    index = False
    compress_extension = 'bz2'
    uncompress = False

    try:
        opts, args = getopt.getopt(argv,"hr:f:x:qp:t:l:m:s:iu:",["ref=","folder=","extension=","fastqc=","mapper=","threads=","seed_length=","mismatches=","single=","index=","uncompress="])
    except getopt.GetoptError:
        help()
    for opt, arg in opts:
        if opt == '-h':
            help()
        elif opt in ("-r", "--ref"):
            ref = arg
        elif opt in ("-f", "--folder"):
            folder = arg
        elif opt in ("-x", "--extension"):
            extension = arg
        elif opt in ("-q", "--fastqc"):
            fastqc = True
        elif opt in ("-p", "--mapper"):
            mapper = arg
        elif opt in ("-t", "--threads"):
            threads = arg
        elif opt in ("-l", "--seed_length"):
            seed_length = arg
        elif opt in ("-m", "--mismatches"):
            mismatches = arg
        elif opt in ("-s", "--single"):
            single = True
        elif opt in ("-i", "--index"):
            index = True
        elif opt in ("-u", "--uncompress"):
            uncompress = True
            compress_extension = arg

    logfile(ref)

    return

def Uncompress(file):

    if compress_extension == 'bz2':
        os.system('bzip2 -d --keep --verbose ' + file)

def files():
    print 'Files to processed'
    file_list = []
    for f in folder.split(','):
        f = f.strip()
        if uncompress == True:
            for file in glob.glob(os.path.join(f, '*'+compress_extension)):
                Uncompress(file)
                file_list.append('.'.join(file.split('.')[:-1]))
        else:
            for file in glob.glob(os.path.join(f, '*'+extension)):
                file_list.append(file)

        print file

    return file_list

def FastQC(file_list):
    if fastqc == True:
        for file in file_list:
            os.system('Running FastQC for '+file)
            os.system('fastqc '+file)

def Index():
    if index == True:
        os.system('nice -n 19 samtools faidx '+ ref)
        os.system('nice -n 19 bwa index -a bwtsw '+ ref)
    if not os.path.isfile(ref + '.fai'):
        os.system('nice -n 19 samtools faidx '+ ref)


def AlignReads(file_list):
    for file in file_list:
        if mapper == 'bwa':
            if not os.path.isfile(file+'.sai'):
                os.system('nice -n 19 bwa aln -t '+ str(threads) +' -l ' +str(
                    seed_length)+ ' ' + ref + ' ' + file + ' > ' + file+'.sai')

def MapReads(file_list):
    for file in file_list:
        if mapper == 'bwa':
            if single == False:
                if re.search('_R1_', file):
                        read1 = file
                        read2 = file.replace('_R1_','_R2_')
                        rg = file.strip().split('/')[-1].strip()[:6].strip()
                        rg = '"@RG\tID:'+rg+'\tSM:'+rg+'\tPL:illumina\tLB:lib1\tPU:unit
                            "'

                        os.system('nice -n 19 bwa sampe -P '+ ' -r ' + rg +' '+ ref +'
                            '+ read1+".sai " + read2+".sai " +\
                                read1 +' '+read2 +' | nice -n 19 samtools view -bt '+
                                    ref+'.fai -| nice -n 19 samtools sort - '+ \
                                read1+'_sorted')

if __name__ == "__main__":

    options(sys.argv[1:])


    ### print the files to be process and return the file path as list
    file_list = files()

    ### fastqc
    FastQC(file_list)

    ### index reference
    Index()

    ### align the reads
    AlignReads(file_list)

    ### map the files
    MapReads(file_list)

    ### close the logfile
    o.close()
```

Listing 2: GATK pipeline

```
#------------------------------------------------------------+
#                                                            |
# 116_runGATK.py - script to run GATK analysis               |
#                                                            |
#------------------------------------------------------------+
#                                                            |
# AUTHOR: Vikas Gupta                                        |
# CONTACT: vikas0633@gmail.com                               |
# STARTED: 09/06/2013                                        |
# UPDATED: 09/06/2013                                        |
#                                                            |
# DESCRIPTION:                                               |
#                                                            |
# LICENSE:                                                   |
#  GNU General Public License, Version 3                     |
#  http://www.gnu.org/licenses/gpl.html                      |
#                                                            |
#------------------------------------------------------------+

# Example:
# python ~/script/python/116_runGATK.py -i 02_Stegodyphous_cdna.refined.fa.orf.
    tr_longest_frame

```

```
### import modules
import os,sys,getopt, re


### global variables
global bams, ref, picard, gatk, threads, variant, sort_bams, tmp_dir

### make a logfile
import datetime
now = datetime.datetime.now()
o = open(str(now.strftime("%Y-%m-%d_%H%M."))+'logfile','w')



### write logfile

def logfile(infile):
    o.write("Program used: \t\t%s" % "116_runGATK.py"+'\n')
    o.write("Program was run at: \t%s" % str(now.strftime("%Y-%m-%d_%H%M"))+'\n')
    o.write("Infile used: \t\t%s" % infile+'\n')


def help():
    print '''
            python 116_runGATK.py
                                -b <bams> [One bam file per sample seperated by
                                    commas]
                                -r <ref> [Reference sequence]
                                -p <picard> [Path to picard folder MarkDuplicates.
                                    jar]
                                -g <gatk> [Path to GATK folder containing
                                    GenomeAnalysisTK.jar]
                                -t <threads> [Number of threads to be used]


            '''
    sys.exit(2)

### main argument to

def options(argv):

    global bams, ref, picard, gatk, threads, variant, sort_bams, tmp_dir

    bams = ''
    ref = ''
    picard = ''
    gatk = ''
    threads = str(1)
    variant = ''
    sort_bams = False
    tmp_dir="/home/vgupta/temp"

    try:
        opts, args = getopt.getopt(argv,"hb:r:p:g:t:v:s",["bams=","ref=","picard=",
            "gatk=","threads=","variant=", "sort_bams="])
    except getopt.GetoptError:
        help()
    for opt, arg in opts:
        if opt == '-h':
            help()
        elif opt in ("-b", "--bams"):
            bams = arg
        elif opt in ("-r", "--ref"):
            ref = arg
        elif opt in ("-p", "--picard"):
            picard = arg
        elif opt in ("-g", "--gatk"):
            gatk = arg
        elif opt in ("-t", "--threads"):
            threads = str(arg)
        elif opt in ("-v", "--variant"):
            variant = arg
        elif opt in ("-s", "--sort"):
            sort_bams = True


    logfile(bams)

def Index():
    if not os.path.isfile(ref + '.fai'):
        os.system('nice -n 19 samtools faidx '+ ref)


def files():
    files = []
    print bams
    for file in bams.split(','):
        print file
        files.append(file.strip())
    return files

def sortBams(file_list):
    if sort_bams == True:
        file_list_bams = []
        for file in file_list:
            os.system('samtools sort '+file+' '+file+'_sorted')
            file_list_bams.append(file+'_sorted.bam')
        return file_list_bams

    return file_list

def MarkDuplicates(file_list):
    for file in file_list:
        print 'Marking duplicates for', file
        os.system('java -Xmx50g -jar '+picard+'/MarkDuplicates.jar
            VALIDATION_STRINGENCY=LENIENT TMP_DIR='+tmp_dir +' INPUT='+ file + '
            OUTPUT=' + file+'.dedup.bam' +' METRICS_FILE='+ file +'.dups')


def ReAlign(file_list):
    for file in file_list:
        print 'Realigning', file
        os.system('java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK
            .jar --fix_misencoded_quality_scores -fixMisencodedQuals -U -T
            RealignerTargetCreator '+' -I ' + file+'.dedup.bam' +' -nt '+ threads +
            ' -R ' + ref +' -o '+ file+'.intervals')
        os.system('java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK
            .jar --fix_misencoded_quality_scores -fixMisencodedQuals -U -T
            IndelRealigner ' + ' -targetIntervals ' + file+'.intervals '+ ' -I ' +
            file+'.dedup.bam' +' -R ' + ref \
                    + ' -o ' + file+'.realigned.bam')


def UnifiedGenotyper(file_list):
    if variant == '':
        in_string = ''
        ### make input string
        for file in file_list:
            os.system('samtools index '+file)
            in_string += ' -I '+file
        print 'Running UnifiedGenotyper'
        '''
        ### for sample
        os.system(' java -jar '+gatk+'/GenomeAnalysisTK.jar '\
        + ' -R ' + ref \
        + ' -T  UnifiedGenotyper '\
        + in_string \
        + ' -nt ' + threads \
        + ' -o snps.90.raw.vcf ' \
        + '-stand_call_conf 20 ' \
        + '-stand_emit_conf 10.0 '\
        + '-dcov 2 ')

        '''
        os.system(' java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/
            GenomeAnalysisTK.jar '\
        + ' -R ' + ref \
        + ' -T  UnifiedGenotyper -glm BOTH '\
        + in_string \
        + ' -nt ' + threads \
        + ' -o '+ file +'.90.vcf ' \
        + '-stand_call_conf 90 ' \
        + '-stand_emit_conf 10.0 '\
        + '-dcov 200 ')


def recal(file_list):
    global variant
    if variant == '':
        variant = file_list[-1] +'.90.vcf'
    for file in file_list:
        print 'Running BaseRecalibrator for ', file
        os.system('java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK
            .jar -U -T BaseRecalibrator -rf BadCigar ' + ' -knownSites ' +variant +
             ' -I ' + file+'.realigned.bam '+ ' -R ' + ref \
              + ' -o ' + file+'.recal.table')
        os.system('java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK
            .jar -T PrintReads -R '+ref +' -I '+ file+'.realigned.bam  -L 20 '+ ' -
            BQSR '+file+'.recal.table' +' -o '+ file+ '.recal_reads.bam' )

def ReduceReads(file_list):
    for file in file_list:
        print 'Running ReduceReads for ', file
        os.system('java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK
            .jar -U -T ReduceReads -rf BadCigar -I ' + file+ '.recal_reads.bam'+ '
            -R ' + ref \
              + ' -o ' + file+'.reduced.bam')

def ReUnifiedGenotyper(file_list):
    print 'Running ReUnifiedGenotyper'
    in_string = ''
    ### make input string
    for file in file_list:
        os.system('samtools index '+ file+'.reduced.bam')
        in_string += ' -I '+file+'.reduced.bam'

    os.system(' java -jar -Djava.io.tmpdir='+tmp_dir+' '+gatk+'/GenomeAnalysisTK.
        jar '\
    + ' -R ' + ref \
    + ' -T  UnifiedGenotyper -glm BOTH'\
    + in_string \
    + ' -nt ' + threads \
    + ' -o snps.raw.vcf ')

def BuildErrorModelWithVQSR(file , var):
    os.system('java -jar '+gatk+'/GenomeAnalysisTK.jar '\
    + ' -T VariantRecalibrator ' \
    + ' -R '+ ref \
    + ' -input '+ file \
    + ' -recalFile output.recal ' \
    + ' -tranchesFile output.tranches ' \
    + ' -nt ' + threads \
    + ' -mode ' + var)

if __name__ == "__main__":

    options(sys.argv[1:])

    ### check if index exits
    Index()


    ### return the list of the bam files
    file_list = files()

    ### sort Bams file
    file_list = sortBams(file_list)

    ### mark duplicates
    MarkDuplicates(file_list)

    ### realign the reads
    ReAlign(file_list)

    ### call UnifiedGenotyper to make a primary list of variants
    UnifiedGenotyper(file_list)

    ### Baserecalibration
    recal(file_list)

    ### reducing BAM files
    ReduceReads(file_list)

    ### Run UnifiedGenotyper
    ReUnifiedGenotyper(file_list)


    ### BuildErrorModelWithVQSR
    #BuildErrorModelWithVQSR('snps.raw.vcf', 'SNP')
    #BuildErrorModelWithVQSR('snps.raw.vcf', 'INDEL')

    ### close the logfile
    o.close()
```

Listing 3: vcfParser

```
#------------------------------------------------------------+
#                                                            |
# 119_vcfParser.py - script to parse vcf format file        |
#                                                            |
#------------------------------------------------------------+
#                                                            |
# AUTHOR: Vikas Gupta                                        |
# CONTACT: vikas0633@gmail.com                               |
# STARTED: 09/06/2013                                        |
# UPDATED: 09/06/2013                                        |
#                                                            |
# DESCRIPTION:                                               |
# Short script to convert and copy the wheat BACs            |
# Run this in the parent dir that the HEX* dirs exist        |
#                                                            |
# LICENSE:                                                   |
#  GNU General Public License, Version 3                     |
#  http://www.gnu.org/licenses/gpl.html                      |
#                                                            |
#------------------------------------------------------------+

# Example:
# python ~/Desktop/script/python/119_vcfParser.py -i snp.90.PhredQual_5000.vcf


### import modules
import os,sys,getopt, re, classVCF


### global variables
global ifile, HEADER

### make a logfile
import datetime
now = datetime.datetime.now()
o = open(str(now.strftime("%Y-%m-%d_%H%M."))+'logfile','w')



### write logfile

def logfile(infile):
    o.write("Program used: \t\t%s" % "100b_fasta2flat.py"+'\n')
    o.write("Program was run at: \t%s" % str(now.strftime("%Y-%m-%d_%H%M"))+'\n')
    o.write("Infile used: \t\t%s" % infile+'\n')


def help():
    print '''
            python 100b_fasta2flat.py -i <ifile>
            '''
    sys.exit(2)

### main argument to

def options(argv):
    global ifile
    ifile = ''
    try:
        opts, args = getopt.getopt(argv,"hi:",["ifile="])
    except getopt.GetoptError:
        help()
    for opt, arg in opts:
        if opt == '-h':
            help()
        elif opt in ("-i", "--ifile"):
            ifile = arg

    logfile(ifile)

### check if file empty
def empty_file(infile):
    if os.stat(infile).st_size==0:
        sys.exit('File is empty')


def parseFile(ifile):
    o = open(ifile+'.MG20filtered','w')
    global HEADER
    count = 0
    for line in open(ifile,'r'):
        if len(line) > 1 and not line.startswith('##'):
            line = line.strip('\n')
            if line.startswith('#CHROM'):
                o.write(line+'\n')
                HEADER = line
                samples_het = []
                samples_homo = []
                sample_names = line.split('\t')[9:]
                samples_len = len(line.split('\t')) -9
                for i in range(samples_len):
                    samples_het.append(0)
                    samples_homo.append(0)
            else:
                obj = classVCF.VCF(line)
                genotypes = obj.genotypes()
                ### check if the MG20 is 0/0 reference Homozygous
                if obj.genotype(2) == '0/0':
                    o.write(line+'\n')
                    count += 1
                    genotypes = obj.genotypes()
                    for i in range(len(genotypes)):
                        if obj.genotype(i) =='0/1' or obj.genotype(i) =='1/0':
                            samples_het[i] += 1
                        elif obj.genotype(i) =='0/0' or obj.genotype(i) =='1/1':
                            samples_homo[i] += 1
    print 'Marksers used: ',count
    print 'Sample\tHetCount\tHomoCount\tHetPer\tHomoPer'

    for i in range(len(sample_names)):
        total = int(samples_het[i]) + int(samples_homo[i])
        Het_per = float(samples_het[i])/total
        Homo_per = float(samples_homo[i])/total
        print sample_names[i] + '\t' + str(samples_het[i]) + '\t' + str(
            samples_homo[i]) + '\t' + str(Het_per) + '\t' + str(Homo_per)
    o.close()

if __name__ == "__main__":

    options(sys.argv[1:])
    empty_file(ifile)

    parseFile(ifile)


    ### close the logfile
    o.close()
```

Listing 4: classVCF

```


import re

### split line
def split_line(line):
    return line.strip().split('\t')

class VCF:
    def __init__(self, line):

        tokens = split_line(line)
        self.CHROM = tokens[0]
        self.POS = tokens[1]
        self.ID = tokens[2]
        self.REF = tokens[3]
        self.ALT = tokens[4]
        self.QUAL = tokens[5]
        self.FILTER = tokens[6]
        self.INFO = tokens[7]
        self.FORMAT = tokens[8]

        self.GENOTYPE = []

        for i in tokens[9:]:
            self.GENOTYPE.append(i)


    def __str__(self):
        return self.CHROM+'\t'+self.POS

    def chroms(self):
        return self.CHROM

    def poss(self):
        return self.POS

    def ids(self):
        return self.ID

    def refs(self):
        return self.REF

    def alts(self):
        return self.ALT

    def quals(self):
        return self.QUAL

    def filters(self):
        return self.FILTER

    def infos(self):
        return self.INFO

    def formats(self):
        return self.FORMAT

    def genotypes(self):
        return self.GENOTYPE

    def depth(self):
        match = re.search(r'DP=.+;',self.INFO)
        if match:
            return match.group().split(';')[0].replace('DP=','')
        else:
            return 0

    def genotype(self, i):
        if len(self.GENOTYPE[i].split(':')) > 1:
            return self.GENOTYPE[i].split(':')[0]
        else:
            return 'NONE'

    def genotypeDepth(self, i):
        if len(self.GENOTYPE[i].split(':')) > 1:
            if len(self.GENOTYPE[i].split(':')) > self.FORMAT.split(':').index('DP') and self.genotype(i) != './.':
                return self.GENOTYPE[i].split(':')[self.FORMAT.split(':').index('DP')]
            else:
                return 'NONE'
        else:
            return 0

    def genotypeQual(self, i):
        if len(self.GENOTYPE[i].split(':')) > 1:
            if len(self.GENOTYPE[i].split(':')) > self.FORMAT.split(':').index('GQ') and self.genotype(i) != './.':
                return self.GENOTYPE[i].split(':')[self.FORMAT.split(':').index('GQ')]
            else:
                return 0
        else:
            return 0

    def genotypeDepthSUM(self):
        geno_sum = 0
        for i in self.GENOTYPE:
            if len(i.split(':')) > 1:
                if len(self.GENOTYPE[i].split(':')) > self.FORMAT.split(':').index('DP'):
                    geno_sum += int(i.split(':')[self.FORMAT.split(':').index('DP')])
        return geno_sum

    def genotypeCalls(self):
        geno_call = 0
        for i in self.GENOTYPE:
            if len(i.split(':')) > 1:
                geno_call += 1
        return geno_call

    def genotypeCallsHete(self):
        geno_call_hete = 0
        for i in self.GENOTYPE:
            if len(i.split(':')) > 1:
                if i.split(':')[0] == '0/1' or i.split(':')[0] == '1/0':
                    geno_call_hete += 1
        return geno_call_hete

    def genotypeCallsHomo(self):
        geno_call_homo = 0
        for i in self.GENOTYPE:
            if len(i.split(':')) > 1:
                if i.split(':')[0] == '0/0' or i.split(':')[0] == '1/1':
                    geno_call_homo += 1
        return geno_call_homo

    def InbreedingCoeffs(self):
        match = re.search(r'InbreedingCoeff=.+;',self.INFO)
        if match:
            return match.group().split(';')[0].replace('InbreedingCoeff=','')
        else:
            return 0

    def HaplotypeScores(self):
        match = re.search(r'HaplotypeScore=.+;',self.INFO)
        if match:
            return match.group().split(';')[0].replace('HaplotypeScore=','')
        else:
            return 0

    def variants(self):
        if self.ALT == '.':
            return 0
        else:
            return 1
```

Listing 5: GenotypicDistance

```
#-----------------------------------------------------------+
#                                                           |
# 119_vcfParser.py - script to parse vcf format file       |
#                                                           |
#-----------------------------------------------------------+
#                                                            |
# AUTHOR: Vikas Gupta                                        |
# CONTACT: vikas0633@gmail.com                               |
# STARTED: 09/06/2013                                        |
# UPDATED: 09/06/2013                                        |
#                                                            |
# DESCRIPTION:                                               |
# Short script to convert and copy the wheat BACs            |
# Run this in the parent dir that the HEX* dirs exist        |
#                                                            |
# LICENSE:                                                   |
#  GNU General Public License, Version 3                     |
#  http://www.gnu.org/licenses/gpl.html                      |
#                                                            |
#------------------------------------------------------------+

# Example:
# python ~/script/python/100b_fasta2flat.py -i 02_Stegodyphous_cdna.refined.fa.orf.
   tr_longest_frame


### import modules
import os,sys,getopt, re, classVCF, time


### global variables
global ifile

### make a logfile
import datetime
now = datetime.datetime.now()
o = open(str(now.strftime("%Y-%m-%d_%H%M."))+'logfile','w')



### write logfile

def logfile(infile):
    o.write("Program used: \t\t%s" % "100b_fasta2flat.py"+'\n')
    o.write("Program was run at: \t%s" % str(now.strftime("%Y-%m-%d_%H%M"))+'\n')
    o.write("Infile used: \t\t%s" % infile+'\n')


def help():
    print '''
            python 100b_fasta2flat.py -i <ifile>
            '''
    sys.exit(2)

### main argument to

def options(argv):
    global ifile
    ifile = ''
    try:
        opts, args = getopt.getopt(argv,"hi:",["ifile="])
    except getopt.GetoptError:
        help()
    for opt, arg in opts:
        if opt == '-h':
            help()
        elif opt in ("-i", "--ifile"):
            ifile = arg

    logfile(ifile)


def calc_dist(g1, g2):
    dist = 0

    if g1 == '0/0' and g2 == '0/1':
        dist += 0.5
    elif g1 == '0/1' and g2 == '0/0':
        dist += 0.5
    elif g1 == '0/1' and g2 == '1/1':
        dist += 0.5
    elif g1 == '1/1' and g2 == '0/1':
        dist += 0.5

    elif g1 == '0/0' and g2 == '1/1':
        dist += 1
    elif g1 == '1/1' and g2 == '0/0':
        dist += 1

    return dist



def printOut(dist_mat, genotypes):
    o = open(ifile+'.dist','w')
    o.write(str(len(genotypes)))
    for i in range(len(genotypes)):
        o.write('\n'+genotypes[i])
        for j in range(len(genotypes)):
            o.write('\t'+str(dist_mat[i,j]))
    o.close()

def parse():
    count = 0
    then = time.time()
    for line in open(ifile, 'r'):
        if len(line) > 0 and not line.startswith('##'):
            line = line.strip()
            obj = classVCF.VCF(line)


            count += 1
            if count%10000 == 0:
                diff = time.time() - then
                minutes, seconds = int(diff)/60, diff % 60
                print 'Number of markers processed: ', '{:9,.0f}'.format(count)
                print('Time taken Min:Sec ==> ' + str(minutes) + ':' + str(round(
                    seconds,2)))

            if line.startswith('#'):
                genotypes = obj.genotypes()
                g_count = len(genotypes)
                dist_mat = {}
                for i in range(g_count):
                    for j in range(g_count):
                        dist_mat[i,j] = 0
            else:
                for i in range(g_count):
                    for j in range(g_count):
                        geno1 = obj.genotype(i)
                        geno2 = obj.genotype(j)
                        dist_mat[i,j] += calc_dist(geno1, geno2)

    printOut(dist_mat, genotypes)

if __name__ == "__main__":

    options(sys.argv[1:])

    ### parse vcf
    parse()


    ### close the logfile
    o.close()
```