# Positive selection study using Prank and CodeML

Vikas Gupta and Stig U. Andersen

January 16, 2014

## Contents

# 1. Introduction

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`.

# 2. 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
```

# 3. Installation

## 3..1 Prank

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

## 3..2 Gblocks

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

## 3..3 Codeml

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

## 3..4 pchisq

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

## 3..5 seqret

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

# 4. Data Analsis

## 4..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
```

## 4..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
```

## 4..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
```

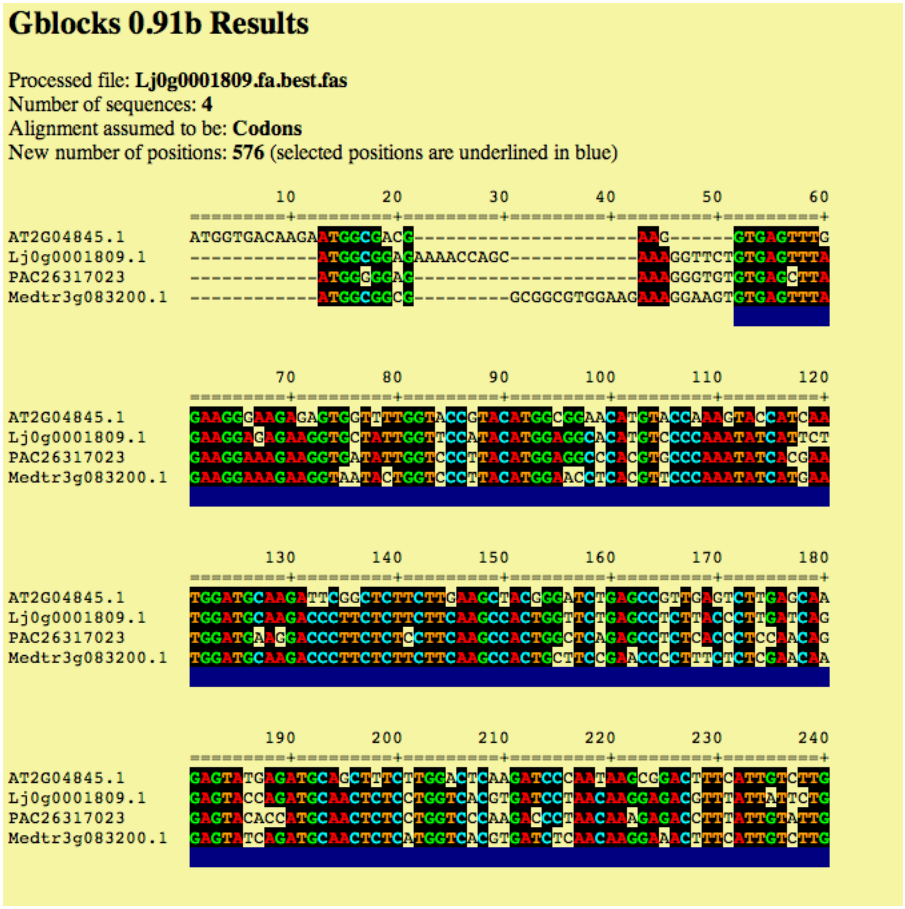


Figure 1: GblocksExample

### 4..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
```

### 4..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
```

### 4..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
```

### 4..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)
```

## 5. 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)
```