FINAL ASSIGNMENT

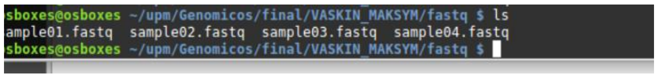
GENOMIC DATA ANALYSIS AND VISUALIZATION / MAKSYM VASKIN / UPM / MS. COMPUTATIONAL BIOLOGY

1. PRELIMINARY INSPECTION

1.1. There are 4 samples

Code:

```
ls
```



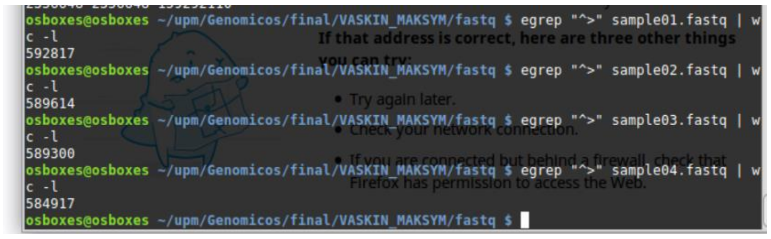
```
sboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ ls
ample01.fastq  sample02.fastq  sample03.fastq  sample04.fastq
sboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ 
```

1.2. Just by inspecting it, I can’t identify the conditions

1.3. There are 592817, 589614, 589300 and 584917 reads in samples 01, 02, 03 and 04, respectively

Code:

```
egrep”>” sample*.fastq | wc -l
```



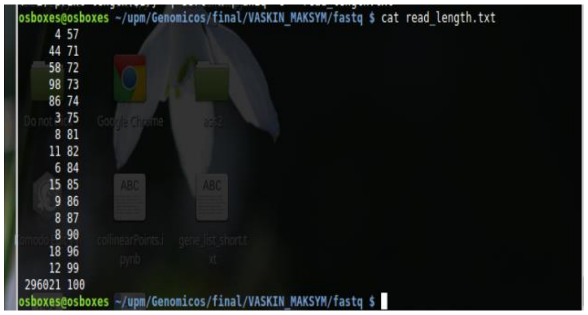
```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ egrep "^>" sample01.fastq | w
c -l
592817
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ egrep "^>" sample02.fastq | w
c -l
589614
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ egrep "^>" sample03.fastq | w
c -l
589300
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ egrep "^>" sample04.fastq | w
c -l
584917
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ 
```

1.4. They are unpaired, otherwise there would be some kind of notation in the end of the reads (/1 or /2).

1.5. They are not (but almost). Vast majority are 100 bp long

Code:

```
cat sample01.fastq | awk ‘{if(NR%4==2) print length($1)}’ | sort –n | uniq –c > read_length.txt

cat read_length.txt
```



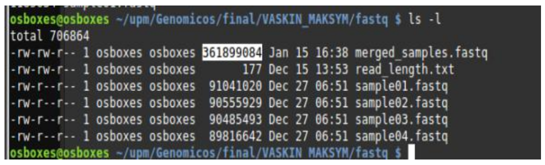
```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ cat read_length.txt
      4 57
     44 71
     58 72
     98 73
     86 74
      3 75
      8 81
     11 82
      6 84
     15 85
      9 86
      8 87
      8 90
     18 96
     12 99
 296021 100
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $
```

1.6. There is no such information so I assume they are all standard 5´-3´

2. TRANSCRIPTOME DE NOVO ASSEMBLY AND ANNOTATION

2.1.1. `cat sample01.fastq sample02.fastq sample03.fastq sample04.fastq > merged_samples.fastq`

2.1.2. It weights 36MBs



```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $ ls -l
total 706864
-rw-rw-r-- 1 osboxes osboxes 361899084 Jan 15 16:38 merged_samples.fastq
-rw-rw-r-- 1 osboxes osboxes       177 Dec 15 13:53 read_length.txt
-rw-r--r-- 1 osboxes osboxes  91041020 Dec 27 06:51 sample01.fastq
-rw-r--r-- 1 osboxes osboxes  90555929 Dec 27 06:51 sample02.fastq
-rw-r--r-- 1 osboxes osboxes  90485493 Dec 27 06:51 sample03.fastq
-rw-r--r-- 1 osboxes osboxes  89816642 Dec 27 06:51 sample04.fastq
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq $
```

2.1.3. Even though files have .fastq extention, they have FASTA format. First, a convertion to REAL FASTQ format was made with bbmap_38.34 by running

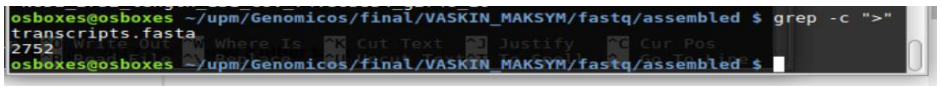
```
bbmap/reformat.sh in=merged_samples.fasta out=merged_samples1.fastq
```

Then I ran rnaspades:

```
/home/osboxes/anaconda_ete/pkgs/spades-3.13.0-0/share/spades-3.13.0-0/bin/rnaspades.py -o fastq --pe1-12 merged_samples1.fastq -t 2
```

Then I counted the contigs:

```
grep -c ">" transcripts.fasta
```



```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq/assembled $ grep -c ">"
transcripts.fasta
2752
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq/assembled $
```

There are 2752 contigs

2.1.5. No. We started from sequences of mRNA, therefore we only have the transcriptome represented in our files. The majority of ChIP-seq experiments are performed on the genome, unless for some reason we want to see only the peaks present in the genes which would make very little sense.

2.2.1First of all, I noticed there were line breaks in the output sequence that might affect the downstream applications, so just in case I ran a code transforming this:

```
>seq
ATCGATCG
ATCGATCG
```

Into this:

>seq
ATCGATCG

```
#The resulting file has linebreaks. We want to remove it without changing the FASTA format:
awk '!/^>/ { printf "%s", $0; n = "\n" }
/^>/ { print n $0; n = "" }
END { printf "%s", n }
' transcript.fasta > joinedlineoutput.fasta
```

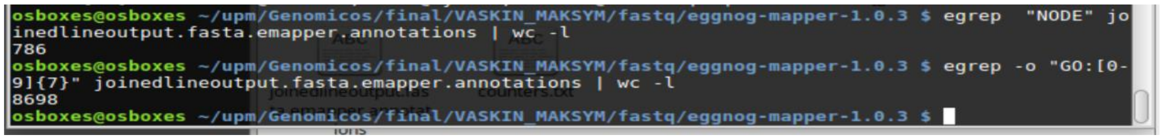
Then, I ran EGGNog mapper of Diamond mode with all other parameters left on default (http://eggnogdb.embl.de/#/app/emapper).

Then, I counted the number of GO annotations by:

```
#To count the total number of annotations
egrep -o "GO:[0-9]{7}" joinedlineoutput.fasta.emapper.annotations | wc -l
```

```
#To count the number of transcripts that have annotations
egrep  "NODE" joinedlineoutput.fasta.emapper.annotations | wc –l
```

There are 8698 GO annotations distributed among 786 transcripts



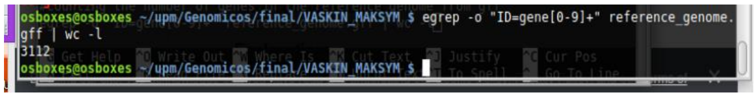
```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq/eggnog-mapper-1.0.3 $ egrep  "NODE" jo
inedlineoutput.fasta.emapper.annotations | wc -l
786
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq/eggnog-mapper-1.0.3 $ egrep -o "GO:[0-
9]{7}" joinedlineoutput.fasta.emapper.annotations | wc -l
8698
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/fastq/eggnog-mapper-1.0.3 $
```

2.2.2. Innitially, there were 2752 contigs, in emapper result we have 786, so 1966 of them lack functional annotation.

2.3.1. Counting the number of genes in the reference genome from gff

```
egrep -o "ID=gene[0-9]+" reference_genome.gff | wc –l
```

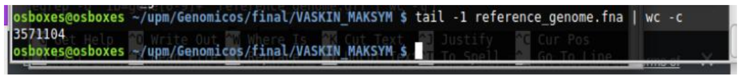
There are 3112



```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ egrep -o "ID=gene[0-9]+" reference_genome.
gff | wc -l
3112
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $
```

2.3.2. Count the size of reference genome. Technically there are two lines only, ID and sequence, so I counted the characters of the second line:

```
tail -1 reference_genome.fna | wc -c
```

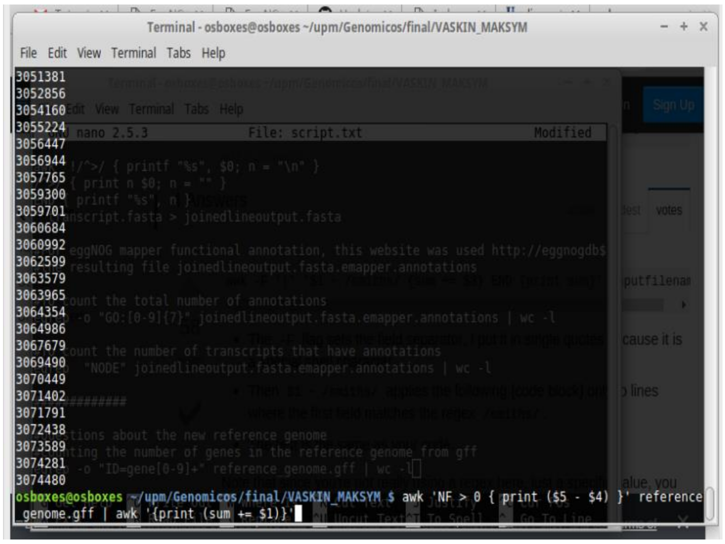


```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ tail -1 reference_genome.fna | wc -c
3571104
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $
```

Counting characters in reference genome file, ignoring the header: 3571104 nucleotides.

2.3.3. Answered in 2.3.1? The number of sequences in .fna file is 1.

2.3.4.



Terminal - osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM

```
3051381
3052856
3054160
3055224
3056447
3056944
3057765
3059300
3059701
3060684
3060992
3062599
3063579
3063965
3064354
3064986
3067679
3069490
3070449
3071402
3071791
3072438
3073589
3074281
3074480
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ awk 'NF > 0 { print ($5 - $4) }' reference
 genome.gff | awk '{print (sum += $1)}'
```

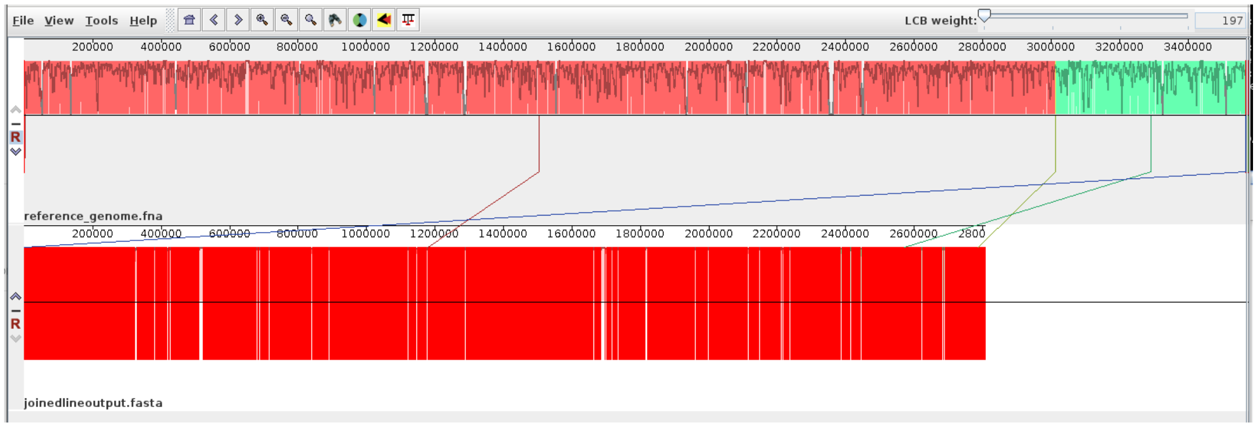
The total length of the annotated genes is 3074480

Code:

```
#Counting the summ of lenghts of all the annotated genes in gff file
awk 'NF > 0 { print ($5 - $4) }' reference_genome.gff | awk '{print (sum += $1)}'
```

2.3.5. My samples have around 58000 reads each with the average length of 100. There are 3112 genes annotated in the reference genome with a total length of 3074480. This would give a mean coverage depth of about 5800000/3074480 = 1.9. It will probably cover the large part of transcriptome but with a very low depth (around 2). This is definitely not en ough for SNP calling as even a SNP of 50/50 would have a 50% of showing no SNP in that position.

2.4.Mauve alignment



3. READ MAPPING

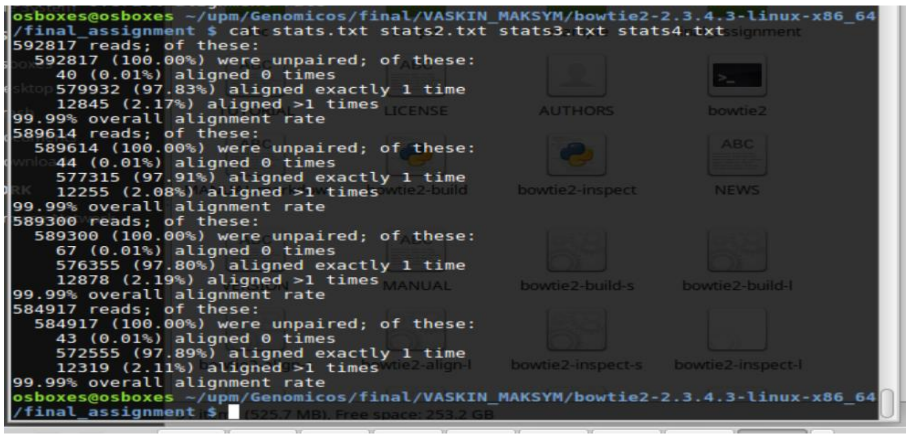
Mapping was performing with this code:

```
#Create index for reference Genome with Bowtie2-build
bowtie2-build reference_genome.fna indexes

#Mapping
bowtie2 -f -x indexes -U sample01.fastq -S aligned_sample_1 2>stats.txt
bowtie2 -f -x indexes -U sample02.fastq -S aligned_sample_2 2>stats2.txt
bowtie2 -f -x indexes -U sample03.fastq -S aligned_sample_3 2>stats3.txt
bowtie2 -f -x indexes -U sample04.fastq -S aligned_sample_4 2>stats4.txt
```

3.1.1. The vast majority of reads was aligned to the reference exactly one time in all samples. Very few (<0.01%) lacked alignment.

Code:

```
#Visualizing statistics of mappings
cat stats.txt stats2.txt stats3.txt stats4.txt
```

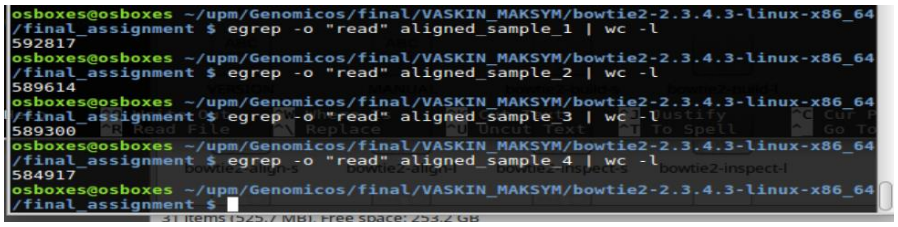


```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $ cat stats.txt stats2.txt stats3.txt stats4.txt
592817 reads; of these:
  592817 (100.00%) were unpaired; of these:
    40 (0.01%) aligned 0 times
    579932 (97.83%) aligned exactly 1 time
    12845 (2.17%) aligned >1 times
99.99% overall alignment rate
589614 reads; of these:
  589614 (100.00%) were unpaired; of these:
    44 (0.01%) aligned 0 times
    577315 (97.91%) aligned exactly 1 time
    12255 (2.08%) aligned >1 times
99.99% overall alignment rate
589300 reads; of these:
  589300 (100.00%) were unpaired; of these:
    67 (0.01%) aligned 0 times
    576355 (97.80%) aligned exactly 1 time
    12878 (2.19%) aligned >1 times
99.99% overall alignment rate
584917 reads; of these:
  584917 (100.00%) were unpaired; of these:
    43 (0.01%) aligned 0 times
    572555 (97.89%) aligned exactly 1 time
    12319 (2.11%) aligned >1 times
99.99% overall alignment rate
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $
```

3.1.2. There was 5922817, 589614, 58300 and 584917 reads in sample 1, 2, 3 and 4, respectively.

Code:

```
#To see the number of the reads
egrep -o "read" aligned_sample_1 | wc -l
egrep -o "read" aligned_sample_2 | wc -l
egrep -o "read" aligned_sample_3 | wc -l
egrep -o "read" aligned_sample_4 | wc -l
```



```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $ egrep -o "read" aligned_sample_1 | wc -l
592817
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $ egrep -o "read" aligned_sample_2 | wc -l
589614
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $ egrep -o "read" aligned_sample_3 | wc -l
589300
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $ egrep -o "read" aligned_sample_4 | wc -l
584917
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM/bowtie2-2.3.4.3-linux-x86_64
/final_assignment $
```

3.1.3. I don't think we could do the analysis of copy number variance because even if the transcript is present once, it will map to all the places in the reference genome.

4. VARIANT CALLING

The code for variant calling was:

```
# Convert to BAM and sort them
for i in 1 2 3 4
do
    samtools view -bS final_assignment/aligned_sample_$i >
aligned_sample_try2_$i.bam
    samtools sort aligned_sample_try2_$i.bam sorted_sample_try2_$i
done
# Merge the already sorted bams and create the index
samtools merge -r -h final_assignment/aligned_sample_1 merged_bam_try3
sorted_sample_try2_*
samtools index merged_bam_try3 merged_bai_try3
# Stats
samtools flagstat merged_bam_try3
# Mpileup
samtools mpileup -g -f reference_genome.fna merged_bam_try3 >
raw_mpileup_try3
# variant calling
bcftools view -bvcg raw_mpileup_try3 > snp_try3.bcf

# Results in .tsv with some columns and no header
bcftools view snp_try3.bcf | grep -v "^#" | cut -f 1,2,4,5,6,8 | sed
's#DP=\([0-9][0-9]*\).*#\1#' > calls_try3.tsv
```

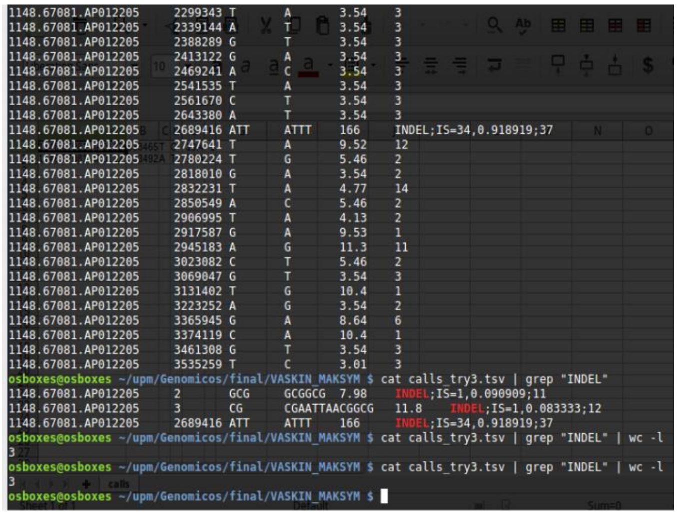
4.1.1. The command used to count the number of entries was:

```
#Counting the number of entries
cat calls_try3.tsv | wc -l
```

It gives us 74 entries

Too see hom many of them were INDELS, I used:

```
#Counting the number of INDELs (last column has an indication of whether
the polymorphism is an INDEL)
cat calls_try3.tsv | grep "INDEL" | wc -l
```



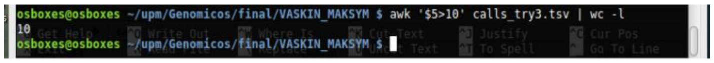
```
1148.67081.AP012205     2299343 T       A       3.54    3
1148.67081.AP012205     2339144 A       T       3.54    3
1148.67081.AP012205     2388289 G       T       3.54    3
1148.67081.AP012205     2413122 G       A       3.54    3
1148.67081.AP012205     2469241 A       C       3.54    3
1148.67081.AP012205     2541535 T       A       3.54    3
1148.67081.AP012205     2561670 C       T       3.54    3
1148.67081.AP012205     2643380 A       T       3.54    3
1148.67081.AP012205     2689416 ATT     ATTT    166     INDEL;IS=34,0.918919;37
1148.67081.AP012205     2747641 T       A       9.52    12
1148.67081.AP012205     2780224 T       G       5.46    2
1148.67081.AP012205     2818010 G       A       3.54    2
1148.67081.AP012205     2832231 T       A       4.77    14
1148.67081.AP012205     2850549 A       C       5.46    2
1148.67081.AP012205     2906995 T       A       4.13    2
1148.67081.AP012205     2917587 G       A       9.53    1
1148.67081.AP012205     2945183 A       G       11.3    11
1148.67081.AP012205     3023082 C       T       5.46    2
1148.67081.AP012205     3069047 G       T       3.54    3
1148.67081.AP012205     3131402 T       G       10.4    1
1148.67081.AP012205     3223252 A       G       3.54    2
1148.67081.AP012205     3365945 G       A       8.64    6
1148.67081.AP012205     3374119 C       A       10.4    1
1148.67081.AP012205     3461308 G       T       3.54    3
1148.67081.AP012205     3535259 T       C       3.01    3
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ cat calls_try3.tsv | grep "INDEL"
1148.67081.AP012205     2       GCG     GCGGCG  7.98    INDEL;IS=1,0.090909;11
1148.67081.AP012205     3       CG      CGAATTAACGGCG   11.8    INDEL;IS=1,0.083333;12
1148.67081.AP012205     2689416 ATT     ATTT    166     INDEL;IS=34,0.918919;37
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ cat calls_try3.tsv | grep "INDEL" | wc -l
3
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ cat calls_try3.tsv | grep "INDEL" | wc -l
3
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $
```

There are 3 insertions (no deletions though). The other 71 are SNPs.

4.1.2. There are 10 entries with a quality score of >10

Code:

```
#Counting the entries with a quality score >10 (qualities are on 5th
column)
awk '$5>10' calls_try3.tsv | wc -l
```

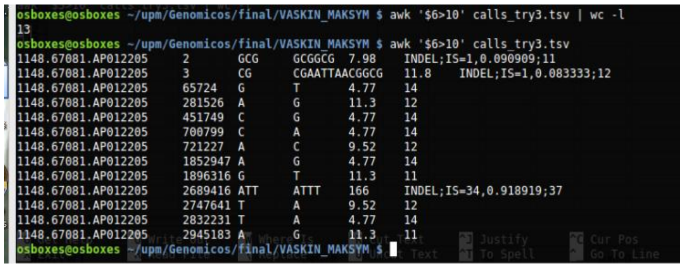


```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ awk '$5>10' calls_try3.tsv | wc -l
10
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $
```

4.1.3. Same line with depth of coverage>10… there are 13

Code:

```
#Same but for depth of coverage >10
awk '$5>10' calls_try3.tsv | wc -l
```



```
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ awk '$6>10' calls_try3.tsv | wc -l
13
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $ awk '$6>10' calls_try3.tsv
1148.67081.AP012205     2       GCG     GCGGCG  7.98    INDEL;IS=1,0.090909;11
1148.67081.AP012205     3       CG      CGAATTAACGGCG   11.8    INDEL;IS=1,0.083333;12
1148.67081.AP012205     65724   G       T       4.77    14
1148.67081.AP012205     281526  A       G       11.3    12
1148.67081.AP012205     451749  C       G       4.77    14
1148.67081.AP012205     700799  C       A       4.77    14
1148.67081.AP012205     721227  A       C       9.52    12
1148.67081.AP012205     1852947 A       G       4.77    14
1148.67081.AP012205     1896316 G       T       11.3    11
1148.67081.AP012205     2689416 ATT     ATTT    166     INDEL;IS=34,0.918919;37
1148.67081.AP012205     2747641 T       A       9.52    12
1148.67081.AP012205     2832231 T       A       4.77    14
1148.67081.AP012205     2945183 A       G       11.3    11
osboxes@osboxes ~/upm/Genomicos/final/VASKIN_MAKSYM $
```

4.1.4. To do this I used the following commands:

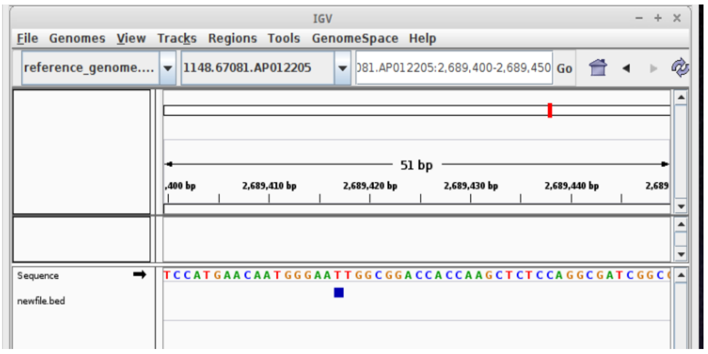
```
#Find the SNP with the best qualit
sort -k5 -n -r calls_try3.tsv | head -1
#Its an insertion ATT>ATTT at the position 2689416

#To find the gene, I used a filter to capture those genes that begin
before and finish after the INDEL site
awk '$4<2689416 && $5>2689416' reference_genome.gff

#Result:
#1148.67081.AP012205  RefSeq gene    2689351 2690262 .      +       .
       ID=gene2518;Name=SYNGTS_RS12550;gbkey=Gene;gene_biotype=protein_cod
ing;locus_tag=SYNGTS_RS12550;old_locus_tag=SYNGTS_2418
```

416 - 351 = 65. The conclusion is ATT begins on the 3rd base of the codon, therefore in the protein TTN becomes TTT, it has a 50% of being neutral but the problem is the frameshift. Since this frame shift is at a beginning of the protein (the 21st aminoacid, it has a great probability of being damaging).

4.1.5. For this, what I did was: cut the first two columns in the .tsv file and create a new file with only those two columns as bed file. This is the minimal information needed to visualize. The more correct thing to to would also to add a another column with (value_of_2nd_column + length(3rdcolumn_of_tsv)). But for the purpose of visualization of the position on the genome it's not needed.

Code:

```
#Creating a bed file with minimal information to upload to IGV. A more
correct
#approach would have been adding a 3rd column where it would be equal to
the
#value of 2nd column + the lenght of the 3rd column of tsv file. But for
simple
#visualization, its not needed.
cat calls_try3.tsv | cut -f 1,2 > newfile.bed
```



```
TCCATGAACAATGGGAATTGGCGGACCACCAAGCTCTCCAGGCGATCGGC
```

5. DIFFERENTIAL EXPRESSION ANALYSIS

The code used to display the number of reads mapping to each GFF feature was:

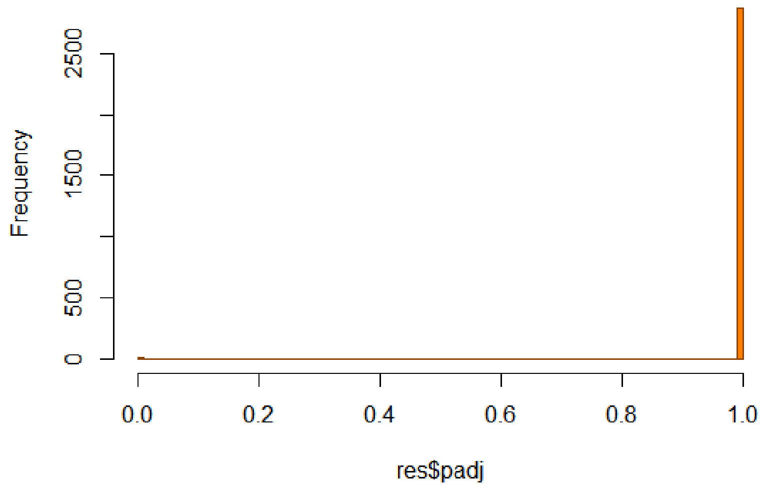
```
#HTSeq Count to display the number of reads mapping to each GFF feature
for i in 1 2 3 4; do htseq-count -t gene -i old_locus_tag
final_assignment/aligned_sample_$i reference_genome.gff; done
```

And then to jon files, I used:

```
#Join the 4 files
join htseq_countfile_1 htseq_countfile_2 > htseq_count_12
join htseq_count_12 htseq_countfile_3 > htseq_count_123
join htseq_count_123 htseq_countfile_4 > htseq_count_all
```

All the R code I used for this part is in DES.R file provided.

5.1.1. The vast majority of genes have a p-value for differential expression analysis close to one, meaning the vast majority of genes are not expressed differently in the two growing conditions.

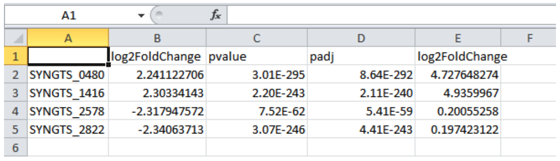


5.1.2. I extracted the indexes of genes with padj<0.01, then saved the values of those indexes selecting the relevant columns (pvalue, padj and log2foldchange) in a new data frame. Then, I calculated fold-change from log2fold change and appended the results as a new column to the dataframe. Finally, I wrote the output to a csv.

```
#Counting the number of genes where Differential Expression between groups had a p-value <0.01
true_ones<-res$padj<0.01
which(true_ones, arr.ind = FALSE, useNames = TRUE)

#Retreiving the genes that are differentially expressed
```

```
answer512<-res[c(428,1264, 2312, 2526),c(2,5,6)]
answer512df<-as.data.frame(answer512)
answer512df

#Calculating foldchange from log2foldchange
foldchanges<-c(2^answer512df[1])
foldchanges
answer_final<-cbind(answer512df,foldchanges)

#Write in a csv
write.csv(answer_final, file="answer_final.csv")
```

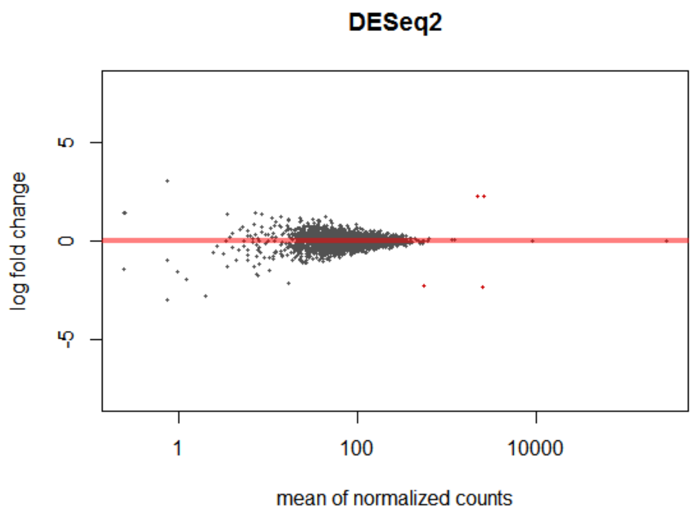
This is the resulting CSV:



	log2FoldChange	pvalue	padj	log2FoldChange
SYNGTS_0480	2.241122706	3.01E-295	8.64E-292	4.727648274
SYNGTS_1416	2.30334143	2.20E-243	2.11E-240	4.9359967
SYNGTS_2578	-2.317947572	7.52E-62	5.41E-59	0.20055258
SYNGTS_2822	-2.34063713	3.07E-246	4.41E-243	0.197423122

There are 2 genes that are repressed and two that are overexpressed when switching from dark to light.

5.1.3. Yes. The p-values are extremely low and the read counts of the genes that yielded the statistically significant differential expression are very high (graph below).



6. SEARCHING FOR CLUSTERS THAT CONTAIN TARGET GENES:

This was the process used to build trees and find Orthologues:

First, I concatenated the proteome FASTA file of the target organism with the proteomes of the organisms I want to compare (all in one file):

```
cat ref_proteomes/*faa > concatenated_proteomes.faa
cat reference_proteome.faa >> concatenated_proteomes.faa
```

Then, using mmseqs2 to create clusters of proteins (belonging to the same family)

```
# Create a database with all sequences
./mmseqs createdb concatenated_proteomes.faa concatenated_proteomes.db
# identify clusters by sequence similarity (with min coverage set to 10%)
./mmseqs cluster concatenated_proteomes.db
concatenated_proteomes.clusters tmp/ -c 0.1
```

Then, I converted each cluster of proteins into a separate FASTA file. For that, I use the split_clusters.py script. But to use it, first I need to:

```
#1)Create a database of clusters with mmseqscreate seq file db
./mmseqs createseqfiledb concatenated_proteomes.db
concatenated_proteomes.clusters clusters.seq
#2)convert them into fasta format with mmseqs result2flat
./mmseqs result2flat concatenated_proteomes.db concatenated_proteomes.db
clusters.seq clusters.fasta
```

Finally splitting clusters

```
mkdir splitted_clusters
python split_clusters.py clusters.fasta splitted_clusters/
```

From all the clusters created (a lot), identify only those that contain the genes found differentially expressed in light VS in dark (see DES.R script for how they are identified). These genes are:

```
#SYNGTS_0480
#SYNGTS_1416
#SYNGTS_2578
#SYNGTS_2822
```

The command I used to find them:

```
grep -E "SYNGTS_0480|SYNGTS_1416|SYNGTS_2578|SYNGTS_2822"
splitted_clusters/*
```

Output is presented below. There are 2 clusters containing 2 target genes each.

```
#splitted_clusters/Arabidopsis_thaliana.AT2G27070.1.fa:>Synechocystis_sp.S
YNGTS_2822
```

```
#splitted_clusters/Arabidopsis_thaliana.AT2G27070.1.fa:>Synechocystis_sp.S
YNGTS_2578
#splitted_clusters/Cyanothece_sp.Cyan7425_3552.fa:>Synechocystis_sp.SYNGTS
_0480
#splitted_clusters/Cyanothece_sp.Cyan7425_3552.fa:>Synechocystis_sp.SYNGTS
_1416
```

Building a philogenetic tree for these 2 clusters.This is done with ete3 and using python2, therefore after installation and running ete3:

```
#now it can be done
mkdir fasta_trees
ete3 build -w standard_fasttree -a
splitted_clusters/Arabidopsis_thaliana.AT2G27070.1.fa -o
fasta_trees/fasta1.tree --clearall --compress -t 0.5 --launch-time 1 --
noimg
ete3 build -w standard_fasttree -a
splitted_clusters/Cyanothece_sp.Cyan7425_3552.fa -o fasta2.tree --clearall
--compress -t 0.5 --launch-time 1 --noimg
```

Lastly, I build the two trees (images below), queried for duplication events and queried for orthologous genes in Arabidopsis by traversing the tree and finding speciation events. The complete python2 script is in the jupyter notebook provided.

6.1. The complete list of genes in Arabidopsis orthologous to any of target genes are:

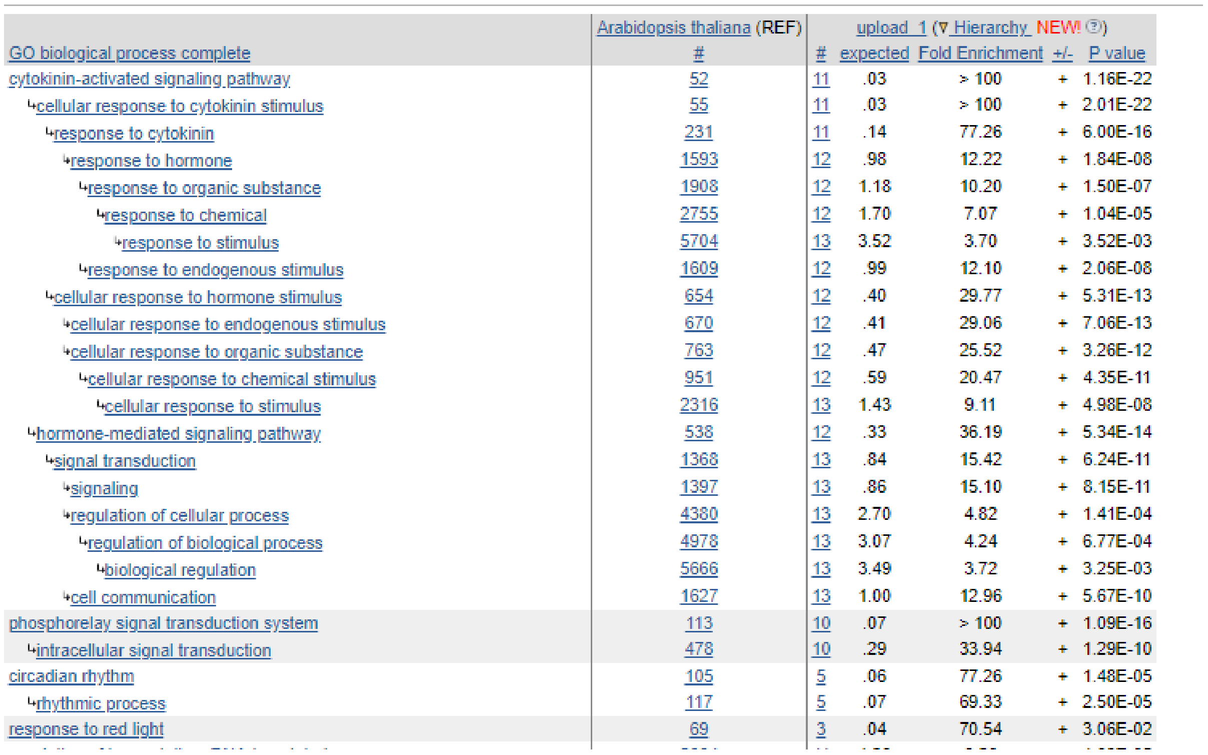
First tree (short one)

```
Arabidopsis_thaliana.AT3G57040.1
Arabidopsis_thaliana.AT2G07440.1
Arabidopsis_thaliana.AT2G41310.1
Arabidopsis_thaliana.AT3G56380.1
Arabidopsis_thaliana.AT2G40670.2
Arabidopsis_thaliana.AT1G10470.1
Arabidopsis_thaliana.AT1G59940.1
Arabidopsis_thaliana.AT1G74890.1
Arabidopsis_thaliana.AT1G19050.1
Arabidopsis_thaliana.AT3G48100.1
Arabidopsis_thaliana.AT5G62920.1
Arabidopsis_thaliana.AT1G68210.1
Arabidopsis_thaliana.AT5G62120.1
Arabidopsis_thaliana.AT5G07210.1
Arabidopsis_thaliana.AT2G27070.1
```

Second tree (long one)

```
Arabidopsis_thaliana.AT1G66340.1
Arabidopsis_thaliana.AT5G10720.1
```

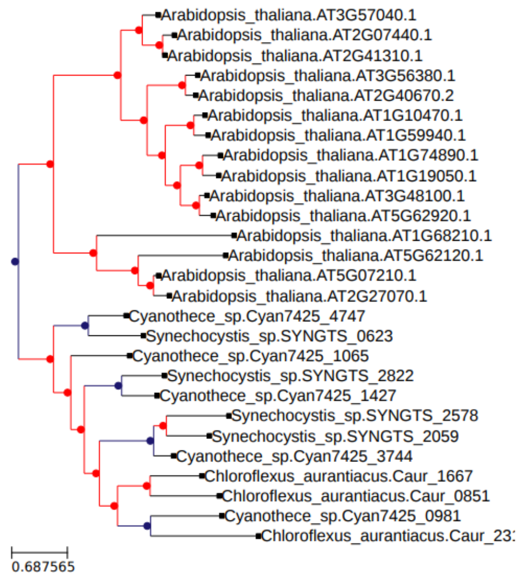
6.2. Instead of introducing the genes individually into the Uniprot database, I opted to introduce the list to GO terms enrichment (http://www.geneontology.org/) which outputs the overall picture og gene list functionality. The top biological functions reported were: cytokine-activated signaling pathway, phosphorelay signal transduction system, circadian rhythm and response to red light.



	Arabidopsis thaliana (REF)	upload_1 (▽ Hierarchy NEW! ⓘ)				
GO biological process complete	#	#	expected	Fold Enrichment	+/-	P value
cytokinin-activated signaling pathway	52	11	.03	> 100	+	1.16E-22
↳cellular response to cytokinin stimulus	55	11	.03	> 100	+	2.01E-22
↳response to cytokinin	231	11	.14	77.26	+	6.00E-16
↳response to hormone	1593	12	.98	12.22	+	1.84E-08
↳response to organic substance	1908	12	1.18	10.20	+	1.50E-07
↳response to chemical	2755	12	1.70	7.07	+	1.04E-05
↳response to stimulus	5704	13	3.52	3.70	+	3.52E-03
↳response to endogenous stimulus	1609	12	.99	12.10	+	2.06E-08
↳cellular response to hormone stimulus	654	12	.40	29.77	+	5.31E-13
↳cellular response to endogenous stimulus	670	12	.41	29.06	+	7.06E-13
↳cellular response to organic substance	763	12	.47	25.52	+	3.26E-12
↳cellular response to chemical stimulus	951	12	.59	20.47	+	4.35E-11
↳cellular response to stimulus	2316	13	1.43	9.11	+	4.98E-08
↳hormone-mediated signaling pathway	538	12	.33	36.19	+	5.34E-14
↳signal transduction	1368	13	.84	15.42	+	6.24E-11
↳signaling	1397	13	.86	15.10	+	8.15E-11
↳regulation of cellular process	4380	13	2.70	4.82	+	1.41E-04
↳regulation of biological process	4978	13	3.07	4.24	+	6.77E-04
↳biological regulation	5666	13	3.49	3.72	+	3.25E-03
↳cell communication	1627	13	1.00	12.96	+	5.67E-10
phosphorelay signal transduction system	113	10	.07	> 100	+	1.09E-16
↳intracellular signal transduction	478	10	.29	33.94	+	1.29E-10
circadian rhythm	105	5	.06	77.26	+	1.48E-05
↳rhythmic process	117	5	.07	69.33	+	2.50E-05
response to red light	69	3	.04	70.54	+	3.06E-02

6.3. One of the trees is way too big. I will attach a high quality image of them with the submission.

6.4. There are, as seen by the duplication events in this tree:


Arabidopsis_thaliana.AT3G57040.1
Arabidopsis_thaliana.AT2G07440.1
Arabidopsis_thaliana.AT2G41310.1
Arabidopsis_thaliana.AT3G56380.1
Arabidopsis_thaliana.AT2G40670.2
Arabidopsis_thaliana.AT1G10470.1
Arabidopsis_thaliana.AT1G59940.1
Arabidopsis_thaliana.AT1G74890.1
Arabidopsis_thaliana.AT1G19050.1
Arabidopsis_thaliana.AT3G48100.1
Arabidopsis_thaliana.AT5G62920.1
Arabidopsis_thaliana.AT1G68210.1
Arabidopsis_thaliana.AT5G62120.1
Arabidopsis_thaliana.AT5G07210.1
Arabidopsis_thaliana.AT2G27070.1
Cyanothece_sp.Cyan7425_4747
Synechocystis_sp.SYNGTS_0623
Cyanothece_sp.Cyan7425_1065
Synechocystis_sp.SYNGTS_2822
Cyanothece_sp.Cyan7425_1427
Synechocystis_sp.SYNGTS_2578
Synechocystis_sp.SYNGTS_2059
Cyanothece_sp.Cyan7425_3744
Chloroflexus_aurantiacus.Caur_1667
Chloroflexus_aurantiacus.Caur_0851
Cyanothece_sp.Cyan7425_0981
Chloroflexus_aurantiacus.Caur_23
0.687565

6.5. Judging by the tree topology, the closest organism to our query is Cyanothece.

7. CELL-TYPE FUNCTIONS AND BIOMARKER PREDICTION

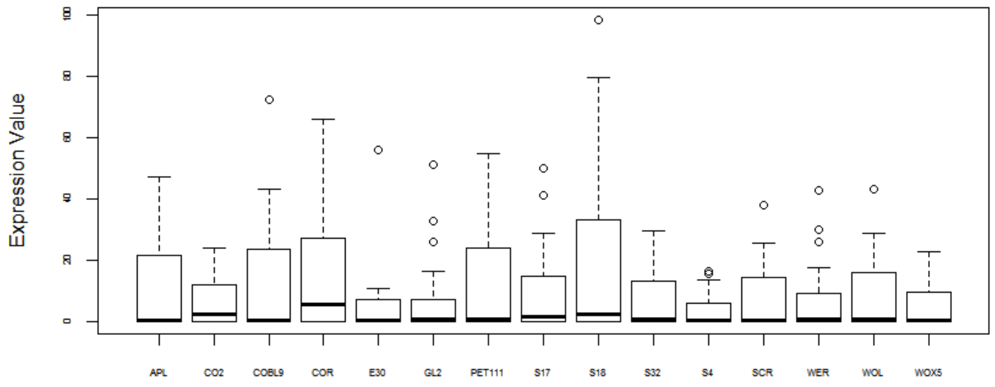
I loaded the .tsv file with data and extracted the genes I'm interested in.

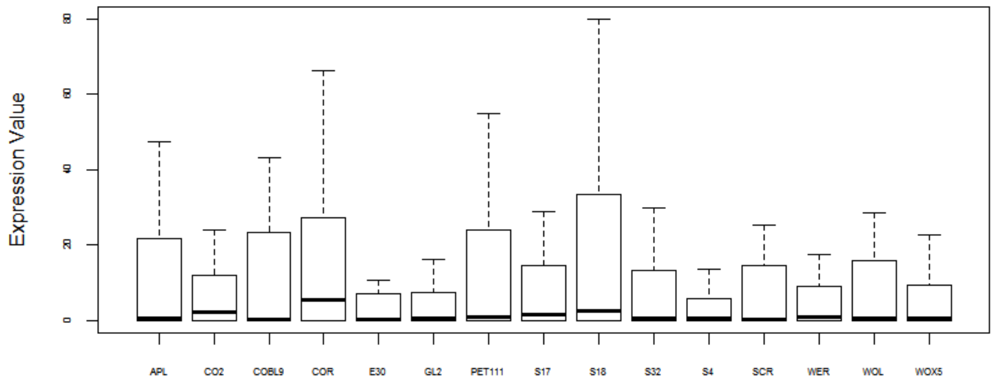
Code:

```
setwd("C:/Users/Maksym/Desktop/UPM/Genomic Data Analysis/Final")
#Read the table containing Arabidopsis RPKMs
pre_mydata <- read.csv("TableCellType.csv", header = TRUE, sep = ",", dec = ".", row.names = 1)
```

Then I made two boxplot (one raw and one without outliers)
Code:

```
boxplot (mydata, ylab="Expression Value", cex.axis=0.5)
boxplot (mydata, outline=FALSE, ylab="Expression Value", cex.axis=0.5)
```



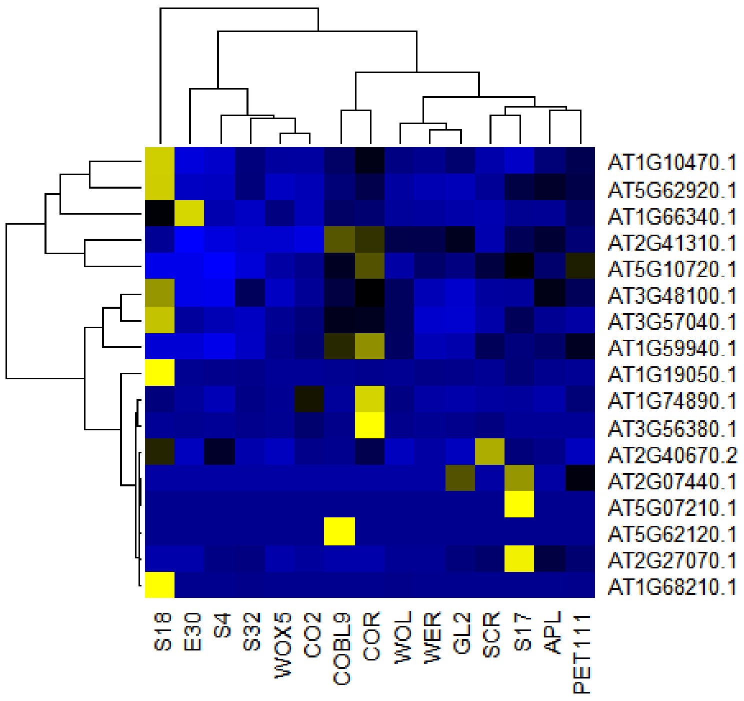


Then I built heatmap

Code:

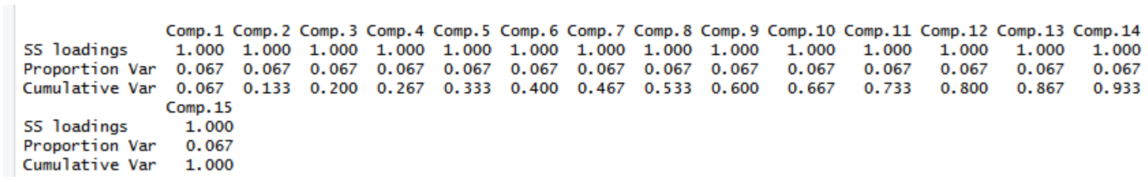
```
#Build a Heatmap of genes (requires data stored as matrix)
marcadoresmatrix <-as.matrix(mydata)
library(RColorBrewer)
```

```
newcolors<-colorRampPalette(colors=c("blue","black","yellow"))(256)
heatmap(marcadoresmatrix,scale="row",col =newcolors, cluster_rows=TRUE)
```



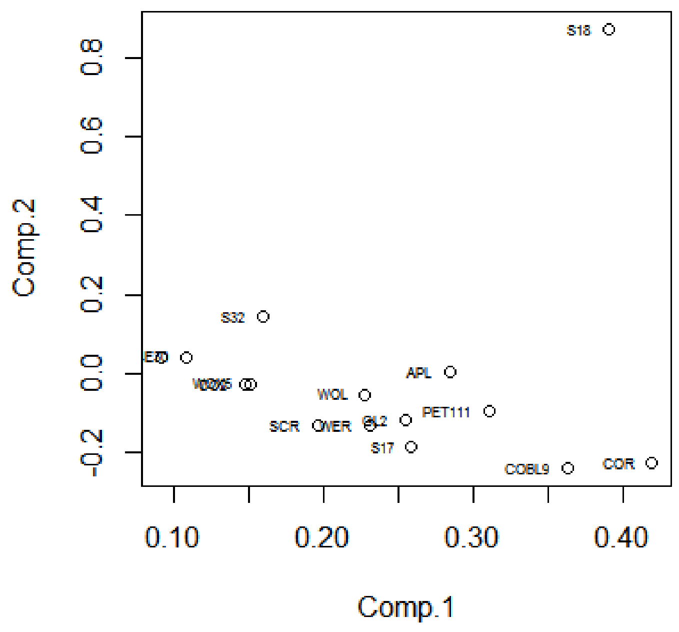
Performing PCA and extracting/plotting loadings.
Code:

```
PCA<-princomp(mydata,cor=FALSE,scores=TRUE)
PCA$loadings
plot(PCA$loadings)
text(PCA$loadings,labels=rownames(PCA$loadings),pos=2,cex=0.5)
```



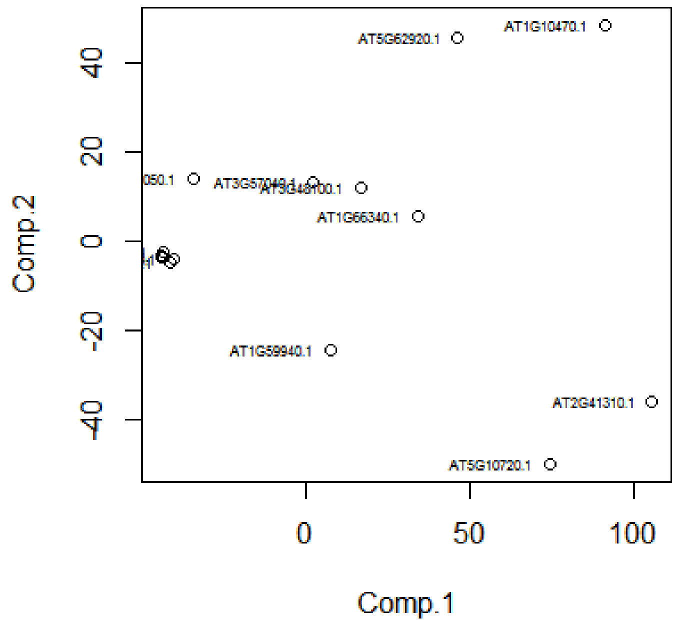
```
               Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9 Comp.10 Comp.11 Comp.12 Comp.13 Comp.14
SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000   1.000   1.000   1.000   1.000   1.000
Proportion Var  0.067  0.067  0.067  0.067  0.067  0.067  0.067  0.067  0.067   0.067   0.067   0.067   0.067   0.067
Cumulative Var  0.067  0.133  0.200  0.267  0.333  0.400  0.467  0.533  0.600   0.667   0.733   0.800   0.867   0.933
               Comp.15
SS loadings      1.000
Proportion Var   0.067
Cumulative Var   1.000
```

Each component contributes exactly 6.7% to explain the variability. The cell type S18 is clearly the best separated one in our PCA.



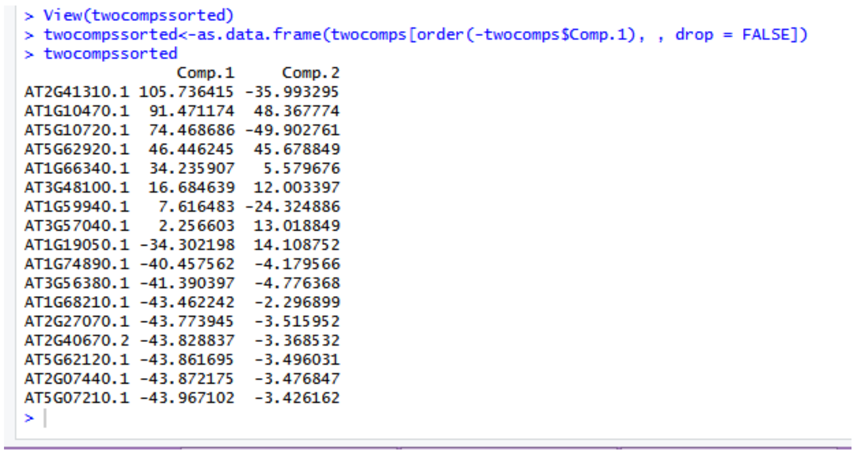
Then I also plotted the scores:

```
PCA$scores
plot(PCA$scores)
text(PCA$scores,labels=rownames(PCA$scores),pos=2,cex=0.5)
```



And I checked how much does each gene contribute for each of the two components:

```
#Check the scores for the two components
twocomps = as.data.frame(PCA$scores[,1:2])
twocompssorted<-as.data.frame(twocomps[order(-twocomps$Comp.1), , drop = FALSE])
twocompssorted
```



```
> View(twocompssorted)
> twocompssorted<-as.data.frame(twocomps[order(-twocomps$Comp.1), , drop = FALSE])
> twocompssorted
               Comp.1     Comp.2
AT2G41310.1 105.736415 -35.993295
AT1G10470.1  91.471174  48.367774
AT5G10720.1  74.468686 -49.902761
AT5G62920.1  46.446245  45.678849
AT1G66340.1  34.235907   5.579676
AT3G48100.1  16.684639  12.003397
AT1G59940.1   7.616483 -24.324886
AT3G57040.1   2.256603  13.018849
AT1G19050.1 -34.302198  14.108752
AT1G74890.1 -40.457562  -4.179566
AT3G56380.1 -41.390397  -4.776368
AT1G68210.1 -43.462242  -2.296899
AT2G27070.1 -43.773945  -3.515952
AT2G40670.2 -43.828837  -3.368532
AT5G62120.1 -43.861695  -3.496031
AT2G07440.1 -43.872175  -3.476847
AT5G07210.1 -43.967102  -3.426162
> 
```

The genes better separating the cell types are those that have the most negative value on one or positive value on one of the axis: ATG41310.1, AT1G10470 and AT5G10720.1

7.1. It seems like many of them are specifically overexpressed in S18 cell type (by looking at heatmap).

7.2. They definitely separate S18 very well (by looking at loading plot). Also they separate COR quite well.

7.3. Yes. Half of them are heavily overexpressed in S18.

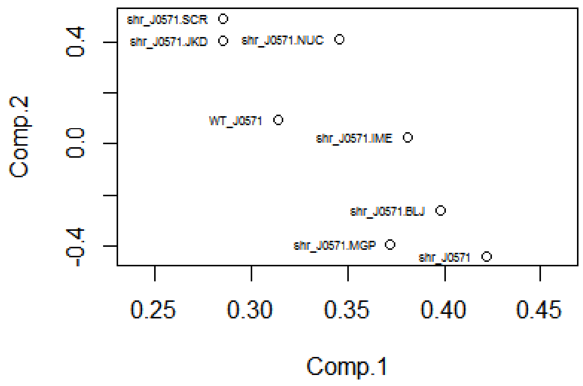
7.4. They seem to be silenced in most cell types and few of them expressed in particular tissue in different combination.

7.5.All of them could, specially those who are heavily repressed in all but one tisse (for example AT5G62120). But not for stem cells (WOX5).

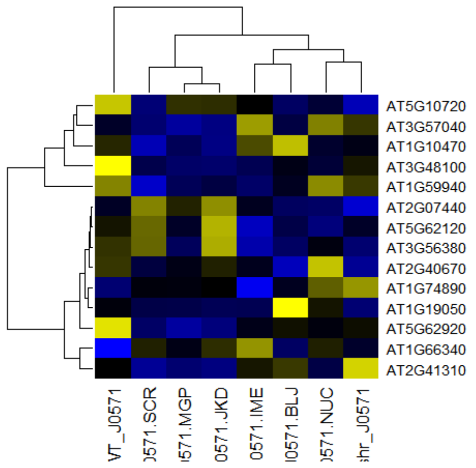
7.6. Yes, as said before some genes are not expressed in any but 1 cell type.

7.7. No.

For the next questions, I retrieved data of stem cell function disruption/complementation analysis and picked the orthologues as subset and saw if any of their functions is lost as a result of the mutation (3 of them were not in the table). By constructing a PCA, I could see that the complements that revert stem function the most are: SRC, JKD and NUC.



My logic was: the gene is responsible for stem cell function, it will be active in WT condition, reduced in shr_J0571 condition and expressed again in shr_J0571,SCR condition. I produced a heat-map of the orthologous genes and looked for the genes meeting this criteria: AT5G62120, AT3G56380



7.8. They might be but it will be inconsistent.

7.9. Maybe only the genes : AT5G62120, AT3G56380

7.10. On the heatmap, AT2G41310 is very repressed in WT and SCR but overexpressed in the mutant, meaning it is switched off in stem-cells. This might be a cancer biomarker.