**R Pipeline**

**December 2023**

Overview:

This pipeline is a series of programs to analyze genome nucleotide variants read counts and to compare the similitude within two replicates. Later, we compare two phenotypically different groups.

# Setting the pipeline:

## Required programs.

spliter.r

physmap.r

physmap.cpp

ven2x2.r

easy\_chi2.r

easy\_chi.cpp

easy\_chi\_fun.r

annotate.r

replace.r

## Required data.

Read counts results from the “varscan” program for the corresponding groups.

1 5810 C 30 27 C:27:2:36:1:20:7:0

1 5811 A 29 27 A:27:2:37:1:20:7:0

1 5812 G 30 29 G:29:2:37:1:21:8:0

1 5813 T 27 25 T:25:2:37:1:19:6:0

1 5814 G 30 27 G:27:2:37:1:19:8:0

1 5815 T 30 27 T:27:2:37:1:22:5:0

1 5816 G 30 28 G:16:2:37:1:12:4:0 C:12:2:37:1:8:4

1 5817 A 30 30 A:30:2:37:1:22:8:0

1 5818 C 29 28 C:28:2:37:1:21:7:0

1 5819 A 28 27 A:27:2:37:1:21:6:0

Genome of reference (annotation genome)

1 IG A

2 IG C

3 IG C

4 IG C

5 IG T

6 IG G

7 IG C

8 IG G

9 IG A

10 IG G

# Running the pipeline

## First steps

For this pipeline to run in a desktop environment with about 16 gb of RAM; the initial input files require some preprocessing. The read counts should be split into “three” chromosomes by the splitter.r program and the reference genome should be imported into sqlite database with the “into\_sql.r” script. Putting the file into sqlite takes ~8 hrs but is only done once.

To keep things organized create the following folder structure and move file as necessary.

data\

data\input

data\output

## Splitting the read counts (splitter.r)

#set up environment ----

rm(list = ls())

library(data.table)

#set i/o ----

## input

filePath <- "data/input/5feb\_tem\_a1.readcounts"

## output

chr1File <- "data/output/5feb\_tem\_a1\_c1\_rc.rds"

chr2File <- "data/output/5feb\_tem\_a1\_c2\_rc.rds"

chr3File <- "data/output/5feb\_tem\_a1\_c3\_rc.rds"

This program will take the read counts single file and split it into three files corresponding to the chromosomes. Notice the “rm()” line, it will remove everything that is in the R environment and start a clean run when the file is sourced.

Set the input file in the input section and pass it to filePath. Set the output files but passing the name and path to the files to the ch1File variables, etc. Hit the source button in RStudio or source it in the command line.

## Genome into sqlite db (into\_sql.r)

Coming soon!!

## Finding mono and polymorphic site (physmap.r)

#set up env ----

rm(list = ls())

library(Rcpp)

library(data.table)

sourceCpp("physmap.cpp")

#set i/o ----

inputFileName <- "data/input/5feb\_tem\_a1\_c1\_rc.rds"

outFileName <- "data/output/5feb\_tem\_a1\_c1.rds"

This program will take the read counts file and look for monomorphic and polymorphic sites. To complete the task in a reasonable time it uses parallel processing and a few C++ functions located in the “physmap.ccp” file, which is required. To run C++ code R requires the installation of the “BH” and “Rcpp” libraries. Depending on how R was setup, it could also require the installation of the [RTools package](https://cran.r-project.org/bin/windows/Rtools/rtools40.html).

Input

> head(rawData, n = 10)

chrom position refnuc depth q30\_depth refQA snp1 snp2 snp3 …

1 1 5810 C 30 27 C:27:2:36:1:20:7:0

2 1 5811 A 29 27 A:27:2:37:1:20:7:0

3 1 5812 G 30 29 G:29:2:37:1:21:8:0

4 1 5813 T 27 25 T:25:2:37:1:19:6:0

5 1 5814 G 30 27 G:27:2:37:1:19:8:0

6 1 5815 T 30 27 T:27:2:37:1:22:5:0

7 1 5816 G 30 28 G:16:2:37:1:12:4:0 C:12:2:37:1:8:4

8 1 5817 A 30 30 A:30:2:37:1:22:8:0

9 1 5818 C 29 28 C:28:2:37:1:21:7:0

10 1 5819 A 28 27 A:27:2:37:1:21:6:0

output

> head(dtChrom, n = 6)

chrom pos ref a c g t i d sumDepth

1: 1 5816 G 0 12 16 0 0 0 28

2: 1 6013 T 0 0 10 61 0 0 71

3: 1 6015 G 0 0 62 12 0 0 74

4: 1 6048 A 67 0 11 0 0 0 78

5: 1 6053 G 11 0 74 0 0 0 85

6: 1 6058 C 0 78 0 0 10 0 88

## Finding common SNPs among groups (ven2x2.r)

This program will find the SNPs that are shared between groups of tested specimens. The output file contrasts one phenotype vs another phenotype.

Input

Four files with a structure like this

> head(dtChrom, n = 6)

chrom pos ref a c g t i d sumDepth

1: 1 5816 G 0 12 16 0 0 0 28

2: 1 6013 T 0 0 10 61 0 0 71

3: 1 6015 G 0 0 62 12 0 0 74

4: 1 6048 A 67 0 11 0 0 0 78

5: 1 6053 G 11 0 74 0 0 0 85

6: 1 6058 C 0 78 0 0 10 0 88

Output

Two files one \*.rds and one \*.txt; both with the same data but different formatting. The \*.tbl was created for use with WCB4’s Fortran code.

> head(pol, n=5)

pos chrom1 ref1 a1 c1 g1 t1 i1 d1 sumDepth1 chrom2 ref2 a2 c2 g2 …

1: 5951 1 G 0 0 52 0 0 0 52 1 G 0 0 49

2: 5995 1 A 64 0 0 0 0 0 64 1 A 59 0 1

3: 6013 1 T 0 0 10 61 0 0 71 1 T 0 0 3

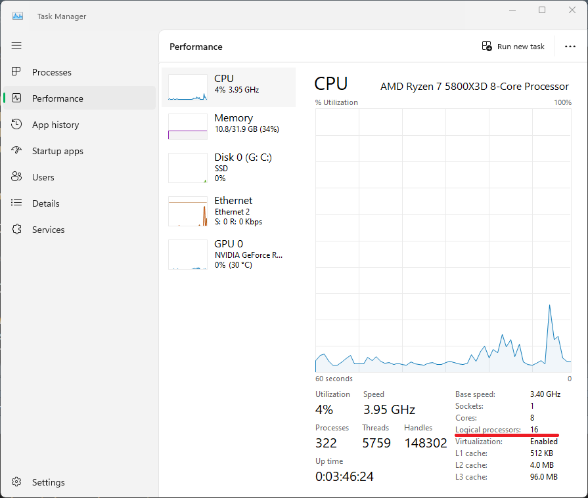
4: 6015 1 G 0 0 62 12 0 0 74 1 G 0 0 69

5: 6048 1 A 67 0 11 0 0 0 78 1 A 83 0 3

# Finding significant associations (easy\_chi2.r)

Performs a chi square analysis to test for significant difference between alive and dead groups.

To speed up the calculations parallel processing was used, to find out how many processors are available look under Logical processors in the “Task Manager”. To keep system stability give the “numberOfProcesors” variable at maximum the number of processor minus 2.



For the “rejectThreshold” variable select an appropriate value (read Benajamini et al )

Input

> head(pol, n=5)

pos chrom1 ref1 a1 c1 g1 t1 i1 d1 sumDepth1 chrom2 ref2 a2 c2 g2 …

1: 5951 1 G 0 0 52 0 0 0 52 1 G 0 0 49

2: 5995 1 A 64 0 0 0 0 0 64 1 A 59 0 1

3: 6013 1 T 0 0 10 61 0 0 71 1 T 0 0 3

4: 6015 1 G 0 0 62 12 0 0 74 1 G 0 0 69

5: 6048 1 A 67 0 11 0 0 0 78 1 A 83 0 3

Output

Text file

The following is the transposed results table of the χ2 results

SNPID 47120 180470 196288 197405 203626 215642

MUTATION AT\_\_\_\_ GA\_\_\_\_ GA\_\_\_\_ GT\_\_\_\_ CG\_\_\_\_ CI\_\_\_\_

FREQ(ALIVE) 0.72068 0 0 0.34802 0.86376 0.78282

FREQ(DEAD) 0.93333 0.23397 0.16077 0.11757 0.61864 0.42947

LOD 4.15 6.58 4.54 3.94 4.12 6.5

HET(ALL) 0.33425 0.12099 0.14924 0.42324 0.37612 0.46935

HET(ALIVE) 0.40095 0 0 0.48481 0.30839 0.35503

HET(DEAD) 0.16037 0.33143 0.26664 0.20031 0.46537 0.49514

FREQ(A) 56 13 16 0 0 0

FREQ(C) 0 0 0 0 56 117

FREQ(G) 0 188 181 238 167 0

FREQ(T) 208 0 0 104 0 0

FREQ(I) 0 0 0 0 0 194

FREQ(D) 0 0 0 0 0 0

CHISQ(ALL) 0.42708 0 0 5.72489 10.13573 1.025

CHISQ(ALIVE) 7.14286 12.6612 1.67668 3.22781 2.09185 9.27838

CHISQ(DEAD) 15.80444 26.49728 17.48237 14.86702 15.6614 26.14947

DF(ALL) 1 0 0 1 1 1

DF(ALIVE) 1 1 1 1 1 1

DF(DEAD) 1 1 1 1 1 1

2\* 2\* 1\* 1\* 2\*

R Binaries

The two resulting binary files contain very similar information. The “raw” file contains all the SNPs while the not “raw” file contains only the SNPs that passed the Benjamini threshold.

**8. Annotate**

This program adds gene information to the SNPs using the annotation file.

Input data

VCP\_AvD\_L5\_C1.chi

13ch/Chrom1\_1.trn

VCP\_AvD\_L5\_C1\_1.ant

Then, run chromosome 1 for the second part Chrom1\_2.trn until finishing chromosome 1

**8. Replace**

This program adds the residue number, the type of substitution, the reference and alternative aminoacid. The data it is still by SNP per chromosome (several files per chromosome)

Input data

VCP\_AvD.c1\_1.ant

output

VCP\_AvD.c1\_1.replace

Then, run to Chrom1\_2.ant until finishing chromosome 1

**9. PerGene**

This program calculates the mean LOD for the top 95% of the SNPs at a gene basis.

Requires the first LOC# in the replace file. Use the following command

awk '/LOC/ {print}' VCP\_AvD.c1\_1.replace | head

Paste OC

Input data

VCP\_AvD.c1\_1.replace

Scratch1

Scratch2

VCP\_AvD.c1\_1.pergene

Then, run Chrom1\_2.replace until finishing chromosome 1