minSNPs User Manual

Table of contents

Introduction

minSNPs is written in R. The convention is that programs written in this language are termed "packages".

We suggest two different user interfaces:

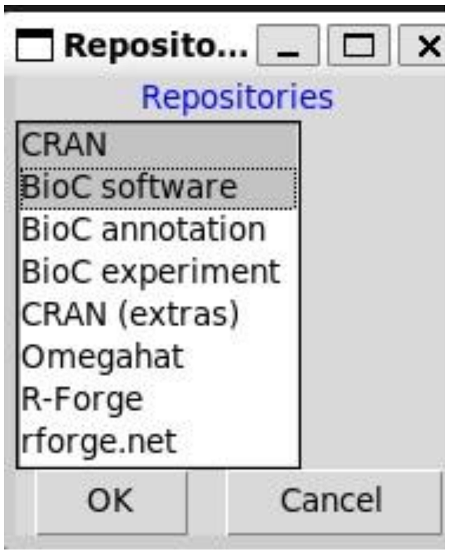
1. For users who are unfamiliar with the UNIX operating system and/or do not have access to a computer cluster, we suggest running minSNPs in RStudio on a PC. RStudio is designed to facilitate the development and operation of R packages by providing a workspace that is quicker and easier to use than a purely command line interface. Although computational capacity will limit the scale and speed of analyses, a typical PC is powerful enough for many useful applications of minSNPs. RStudio Desktop can be accessed at https://www.rstudio.com/products/rstudio/
2. For users who are familiar with the UNIX environment, or who can access appropriate support, we suggest operating minSNPs using a standard command line interface, on a UNIX-based high performance computer cluster. R base Binary will need to be installed. Running R in this environment creates an "R Terminal".

The operations for both interfaces are very similar. In general, all operations & commands for R Terminal can be copied and run within the terminal in RStudio, and where RStudio provides a UI to simplify the user experience, the following sections provide additional screenshots.

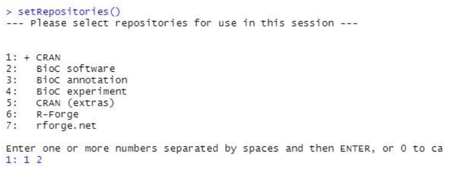
Accessing & Installing minSNPs

minSNPs is available in the CRAN repository of R packages. The R environment makes it straightforward to load packages directly from CRAN i.e., it is unnecessary to download the code onto a local machine prior to installation.

1. In either a R terminal or RStudio, run the command `setRepositories()` and make sure that both **CRAN** and **BioC software** are selected; can be confirmed with `getOption("repos")`.

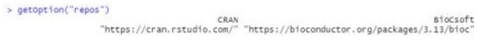


Prompt in CLI



```
> setRepositories()
--- Please select repositories for use in this session ---


1: + CRAN
2:   BioC software
3:   BioC annotation
4:   BioC experiment
5:   CRAN (extras)
6:   R-Forge
7:   rforge.net

Enter one or more numbers separated by spaces and then ENTER, or 0 to ca
1: 1 2
```

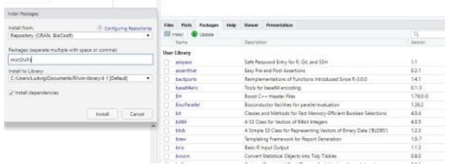
Prompt in RStudio



```
> getOption("repos")
                                CRAN                                              BioCsoft
            "https://cran.rstudio.com/" "https://bioconductor.org/packages/3.13/bioc"
```

Correct outcome

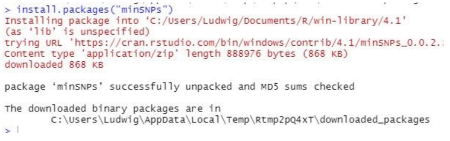
2. Run the command `install.packages("minSNPs")` or use the GUI in Rstudio to install `minSNPs`, the package will be downloaded from CRAN and installed,.



```
Install Packages
Install from:                                   Configuring Repositories
Repository (CRAN, BioCsoft)
Packages (separate multiple with space or comma):
minSNPs
Install to Library:
C:/Users/Ludwig/Documents/R/win-library/4.1 [Default]
☑ Install dependencies
                                                Install   Cancel
```

```
Files  Plots  Packages  Help  Viewer  Presentation
Install  Update
     Name         Description                                                      Version
User Library
☐   askpass       Safe Password Entry for R, Git, and SSH                          1.1
☐   assertthat    Easy Pre and Post Assertions                                     0.2.1
☐   backports     Reimplementations of Functions Introduced Since R-3.0.0          1.4.1
☐   base64enc     Tools for base64 encoding                                        0.1-3
☐   BH            Boost C++ Header Files                                           1.78.0-0
☐   BiocParallel  Bioconductor facilities for parallel evaluation                  1.26.2
☐   bit           Classes and Methods for Fast Memory-Efficient Boolean Selections 4.0.4
☐   bit64         A S3 Class for Vectors of 64bit Integers                         4.0.5
☐   blob          A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS') 1.2.3
☐   brew          Templating Framework for Report Generation                       1.0-7
☐   brio          Basic R Input Output                                             1.1.3
☐   broom         Convert Statistical Objects into Tidy Tibbles                    0.8.0
```

Installing with GUI in RStudio



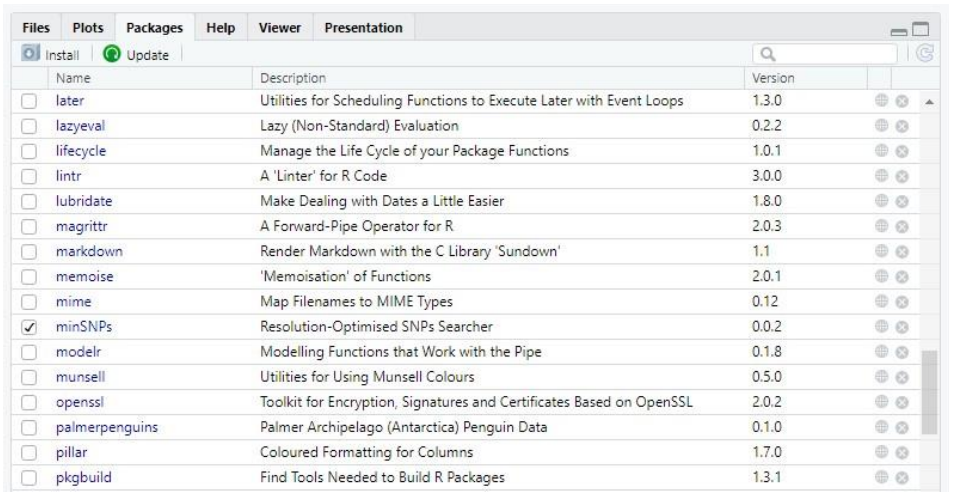
```
> install.packages("minSNPs")
Installing package into 'C:/Users/Ludwig/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/minSNPs_0.0.2.
Content type 'application/zip' length 888976 bytes (868 KB)
downloaded 868 KB

package 'minSNPs' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Ludwig\AppData\Local\Temp\Rtmp2pQ4xT\downloaded_packages
> |
```

Successful installation message

Using minSNPs

This section provides the syntax for the basic functions, all the text enclosed by arrow brackets and highlighted (e.g., `<user selected name for variable>`) are to be replaced by user with a suitable value as described, further explanation for each argument will be provided below the syntax.

Before using any of the following functions, first import minSNPs package into the environment. This can be done with `library(minSNPs)` in R terminal or by ticking the box in RStudio.



	Name	Description	Version
☐	later	Utilities for Scheduling Functions to Execute Later with Event Loops	1.3.0
☐	lazyeval	Lazy (Non-Standard) Evaluation	0.2.2
☐	lifecycle	Manage the Life Cycle of your Package Functions	1.0.1
☐	lintr	A 'Linter' for R Code	3.0.0
☐	lubridate	Make Dealing with Dates a Little Easier	1.8.0
☐	magrittr	A Forward-Pipe Operator for R	2.0.3
☐	markdown	Render Markdown with the C Library 'Sundown'	1.1
☐	memoise	'Memoisation' of Functions	2.0.1
☐	mime	Map Filenames to MIME Types	0.12
✓	minSNPs	Resolution-Optimised SNPs Searcher	0.0.2
☐	modelr	Modelling Functions that Work with the Pipe	0.1.8
☐	munsell	Utilities for Using Munsell Colours	0.5.0
☐	openssl	Toolkit for Encryption, Signatures and Certificates Based on OpenSSL	2.0.2
☐	palmerpenguins	Palmer Archipelago (Antarctica) Penguin Data	0.1.0
☐	pillar	Coloured Formatting for Columns	1.7.0
☐	pkgbuild	Find Tools Needed to Build R Packages	1.3.1

RStudio's GUI to import package to environment

Part A. The basic functions

1. Sequence data input.

minSNPs derives resolution optimized sets of SNPs from files that are in the FASTA format, and are text files. No other sequence or sequence alignment formats can be analysed. minSNPs will input and analyse any text file in FASTA format, irrespective of the symbols, so a SNP is simply any position in the alignment that is variable.

File input uses the minSNPs function "read_fasta". The **syntax** is:

```
<user selected name for variable> <- read_fasta(file=<"path to FASTA
file on your computer network">)
```

Arguments

- file: the value must include the file extension of the file, in R terminal, the tab key can be used to show suggestion or autocomplete the file path.

User's thoughts / Tips
The above function can be simplified if the working directory is defined first. This can be done using the R function setwd.
The syntax is `setwd("path to your working directory")`.
In R, paths to files or directories are defined using forward slashes.
A hypothetical example of the functions to set a working directory and load a FASTA file is:
`setwd("C:/my_alignments/Staph_aureus")`
`Staph_SNP_matrix <- read_fasta(file = "Staph_genomes.fasta.txt")`
Note, the name of the file to be loaded must contain the extension.
This will load the file "Staph_genomes.fasta.txt" from the working directory into minSNPs, and give it the variable name "Staph_SNP_matrix". This variable name is how this file will be referred to in all minSNPs analysis commands.
Because all minSNPs analyses incorporate the variable name of the FASTA file, multiple files can be input and analysed in a single R session.
Note that it is a convention that the name of the first argument in a function does not need to be specified, e.g.
`Staph_SNP_matrix <- read_fasta("Staph_genomes.fasta.txt")`
is equivalent to the example before.

Commented [PG1]: originally ;left out underscore

Commented [PG2]: as above

2. Derivation of resolution optimized SNP sets.

The minSNPs function for initiating an analysis to derive optimized SNP sets is "find_optimised_snps"

minSNPs can derive optimized SNP sets according to two different metrics of resolving power, "percent mode" and "simpson", see explanation for difference below.

The **syntax** for deriving optimized SNP sets is:

```
<user selected name for results variable > <-
find_optimised_snps(seqc=<variable name of FASTA file to be
analysed>, metric=<"percent" or "simpson">, number_of_result=<number
of SNP sets the user requires>, max_depth=<maximum number of SNPs
within each set>, goi=<names of sequences in group of interest, for
"% mode only">)
```

Arguments

- seqc: The value should be replaced with the user-assigned variable from the "read_fasta" function.
- metric: The options "percent" and "simpson" specify the metric that will be used. The default is "simpson" but it is probably good practice to always specify the metric. Both options may not be specified at one time; minSNPs cannot simultaneously perform SNP set derivations using both metrics.

  This is the metric used to evaluate the resolving power of the SNP sets, the two modes are:

  - The "percent" metric yields SNP sets that are optimized to discriminate a subset of the sequences in the alignment (the "group of interest) from all other sequences in the alignment. Sensitivity is constrained to 1.0 and the SNPs sets optimised with respect to specificity.
  - The "simpson" metric yields SNP sets that are optimized to organize sequences into different groups. The SNPs are selected based on maximum cumulative Simpson's index of diversity.

  When the "percent" metric is specified, a subset of the sequences in the alignment must be specified as the goi (see "percent mode" above). The names of the sequences in the FASTA file are used to define the goi.
- number_of_result: This specifies the number of SNP sets that the analysis will yield.
- max_depth: This specifies the maximum number of SNPs in each set.
- goi: This specifies the names of the sequences in the FASTA file are used to define the group of interest for "percent mode" analysis, must be specified if metric="percent", and otherwise ignored.

- The progressive generation of required number of results (i.e. SNP sets) is reported in the R console as the program runs.

User's thoughts / Tips
A hypothetical example of a script to search in D mode: `Staph_D_SNPs <- find_optimised_SNPs(seqc = Staph_SNP_matrix, metric="Simpson", number_of_result=10, max_depth=8)` minSNPs will derive SNP sets from the FASTA format variable "Staph_SNP_matrix", with all sets optimized with respect to the Simpsons Index of Diversity. Ten sets of

SNPs with maximum size 8 will be derived. The results will be assigned to a variable with the user selected name "Staph_D_SNPs".

A hypothetical example of a script to search using percent as the metric:

```
Staph_SNPs_seqs12_38_50 <- find_optimised_SNPs(seqc =
Staph_SNP_matrix, metric="percent", number_of_result=10, max_depth=5,
goi=c("seq12", "seq38", "seq50"))
```

minSNPs will derive SNP sets from the variable "Staph_SNP_matrix", with all sets optimized with respect to the power to discriminate sequences "seq12", "seq38" and "seq50" from all other sequences in the alignment. "c()" is an R way to create an array. The sequence names inside the "c()" are from sequence names in the FASTA file, and must be a perfect match. Ten sets of SNPs with maximum size 5 will be derived. The results will be assigned to a variable with the user selected name " Staph_SNPs_seqs12_38_50".

Remember that the name of the first argument in a function does not need to be specified, e.g.

```
Staph_SNPs_seqs12_38_50 <- find_optimi/sed_SNPs(Staph_SNP_matrix,
metric="percent", number_of_result=10, max_depth=5, goi=c("seq12",
"seq38", "seq50"))
```

is equivalent to the previous example.

3. Output of the results

To output the result in RStudio or an R terminal, the syntax is:

```
    output_result(result=<result variable>, view=<"" or "tsv">,
    file_name=<"filename to save as">)
```

Commented [PG3]: The file name for the tsv file needs to be in inverted commas – the way this is written suggest not – there are inverted commas around "tsv" but not "filename to save as".

Also using tsv or csv seems interchangeable – I guess that does not matter?

Finally, you have tp put the tsv file extension in the file name if you want Microsoft to think it is a tsv file – but I guess it does not matter because it should load into Excel anyway

Arguments

- result: The value should be replaced with the user-assigned variable from the "find_optimised_snps" function.
- view: This specifies whether to output the result to console ("") or to tsv ("tsv").
- file_name: This specifies the filename if the output format is tsv, file extension is not required.

User's thoughts / Tips

A hypothetical example of using the function in a script to save result to a tab-delimited file is:

```
output_result(result = Staph_SNPs_seqs12_38_50, view = "tsv",
"SNPs_seqs12_38_50")
```

This will generate a version of the output from "Staph_SNPs_seqs12_38_50" in tab-delimited file format, which is saved to the working directory, with the user specified name "SNPs_seqs12_38_50.tsv". Note, the "tsv" file name extension is not specified.

Tab delimited files can be imported into Microsoft Excel.

To view the result output for "Staph_SNPs_seqs12_38_50" in RStudio or an R terminal (whichever is being used), view or file_name can be omitted.

4. Additional functions and features

These features are functions/additional arguments to the 3 previously described functions.

- *Forced inclusion and exclusion of alignment positions into/from the derived SNP sets*

This can be specified as additional argument for **"find_optimised_snps"** function. Either or both parameters can be used at the same time. However, the positions cannot overlap, otherwise, error will be returned.

Note

SNPs specified in "included_positions" don't count towards "max_depth", i.e., if there are 2 SNPs forced to be included in the result set and "max_depth" is 2, the final result SNP sets may have up to 2 additional SNPs.

```
<user selected name for results variable > <-
find_optimised_snps(seqc=<variable name of FASTA file to be
analysed>, metric=<"percent" or "simpson">,
number_of_result=<number of SNP sets the user requires>,
max_depth=<maximum number of SNPs within each set>, goi=<names of
sequences in group of interest, for "% mode only">,
included_positions=<list of position to be included>,
excluded_positions=<list of positions to be excluded>)
```

Arguments

- included_positions: Positions of SNPs that must be included in the SNP sets.
- excluded_positions: Positions of SNPs that are excluded in the SNP sets.

User's thoughts / Tips

Extending from the earlier hypothetical example in "find_optimised_snps",

```
Staph_SNPs_seqs12_38_50 <- find_optimised_SNPs(seqc =
Staph_SNP_matrix, metric="percent", number_of_result=10,
max_depth=5, goi=c("seq12", "seq38", "seq50"),
included_positions=c(1,2,3), excluded_positions=c(4,5,6))
```

All 10 of the result SNP sets will have SNPs 1, 2, 3 in it and up to additional 5 SNPs, and none of the SNP sets will have SNPs 4, 5, 6. The actual number of SNPs returned will defer set by set, the function will stop searching once the SNP set achieved 1 in metric (completely distinguished all group of interest).Optional exclusion of positions with non-standard symbols

User's thoughts / Tips – Streamlining "goi", "included_positions" and "excluded_positions" argument

With a large number of goi, included_positions or excluded_positions, the function can be extremely long and unruly, it is possible to specifies all the above to a variable before calling the function, e.g.,

```
my_goi <- c("seq12", "seq38", "seq50", "seq55", "seq56",
"seq65", "seq70", "seq71", "seq72", "seq75", "seq85", "seq95")

interested_positions <- c(1,2,3,4,5,6,7,8,9,10)

unwanted_positions <- c(30,40,50,60,70,80,90,100)

Staph_SNPs_seqs12_38_50 <- find_optimised_SNPs(seqc =
Staph_SNP_matrix, metric="percent", number_of_result=10,
max_depth=5, goi=my_goi,
included_positions=interested_positions,
excluded_positions=unwanted_positions)
```

- *Enabling parallel processing*

This can be specified as additional argument for **"find_optimised_snps"** function.

```
<user selected name for results variable > <-
find_optimised_snps(seqc=<variable name of FASTA file to be
analysed>, metric=<"percent" or "simpson">,
number_of_result=<number of SNP sets the user requires>,
max_depth=<maximum number of SNPs within each set>, goi=<names of
sequences in group of interest, for "% mode only">,
included_positions=<list of position to be included>,
excluded_positions=<list of positions to be excluded>,
bp=<parallelization>)
```

Arguments

- bp: This specifies whether to parallelize the search. By default, the function is not parallelized and run with only a single core. To enable parallelization, substitute the value for this with `BiocParallel::MulticoreParam()`. That will correctly detect the number of available cores in PC and make use of the available cores. However, to parallelize the search in a HPC that make use of queue system, the value for this argument should be substituted with

`BiocParallel::MulticoreParam(workers=<X>),` X being the number of cores to use, this is typically (number of assigned cores for the run in the HPC) – 2.

User's thoughts / Tips

minSNPs will run on Windows machine with no problem, defaulting to using only 1 core and no changes to the "find_optimised_snps" is needed. The example below will not work for Windows operating system, further information can be found here.
 	
The example below is for machines running on Linux operating system. Further extending from the previous hypothetical example in "find_optimised_snps",
 	
`Staph_SNPs_seqs12_38_50 <- find_optimised_SNPs(seqc = Staph_SNP_matrix, metric="percent", number_of_result=10, max_depth=5, goi=c("seq12", "seq38", "seq50"), included_positions=c(1,2,3), excluded_positions=(4,5,6), bp=BiocParallel::MulticoreParam())`
 	
Not specifying number of workers is fine in Linux Desktop PC, the function will automatically detect the number of available cores.
 	
In an HPC, the number of workers must be specified to the requested cores, since not all the detected cores will be assigned to the job.
 	
`Staph_SNPs_seqs12_38_50 <- find_optimised_SNPs(seqc = Staph_SNP_matrix, metric="percent", number_of_result=10, max_depth=5, goi=c("seq12", "seq38", "seq50"), included_positions=c(1,2,3), excluded_positions=(4,5,6), bp=BiocParallel::MulticoreParam(workers = 8))`
 	
The above call will run the search with 8 cores, the parallelization depends on BiocParallel, for more information, further information can be found here.

- *Dealing with alignments containing non-standard symbols or indels*

This is done with "process_allele" function, before using the "find_optimised_snps".

The function performs 2 different functions, i.e., (1) removes sequences within the alignment that has length different from most other sequences, and (2) removes positions of SNPs where at least one of the sequences contain a non-standard symbol at that position.

The first function is non-modifiable, to analyse a matrix with indels, make sure that all the sequences within the alignment have the same length, i.e., for all sequences without insertion, use "-" at the inserted positions, likewise for deletion, use "-" to represent the deleted bases in all the sequences.

The second function can be modified, by default, all non-standard symbols including failed sequencing ("N"), and ambiguity codes are removed, but these can be adjusted. Similarly, "-" are by default removed, it can be accepted, either by adding it to the accepted_char argument, or by setting dash_ignore to FALSE.

The syntax is:

```
<user selected name> <-process_allele(seqc=<user variable for
read fasta file>, dash_ignore=<TRUE or FALSE>,
accepted_char=<list of accepted symbols>)
```

Arguments

- seqc: This should be replaced by the variable from "read_fasta".
- dash_ignore: This specifies whether to treat "-" as another type or just remove position where at least one of the sequences contain a "-", by default, any position containing a "-" is removed.
- accepted_char: This specifies the list of accepted character, by default, this is c("A","C","T","G"), but can be changed.

User's thoughts / Tips

While this is non-essential, users are encouraged to use the function prior to running any minSNPs analysis to ensure that the input data are clean.

Assumes "read_sequences" is the assigned variable from read_fasta from a file containing multiple sequences of varied length,

```
Staph_alignment <- process_allele(seqc = read_sequences)
```

The above removes those sequences with length different from a majority of the sequences and those positions that contain a symbol other than "A", "C", "T" and "G".

If "read_sequences" is the assigned variable from read_fasta from a file containing multiple sequences of varied length, and contain indels,

```
Staph_alignment <- process_allele(seqc = read_sequences,
dash_ignore = FALSE)
```

The above removes those sequences with length different from a majority of the sequences and those positions that contain a symbol other than "-", "A", "C", "T" and "G".

minSNPs can be used to analyse RNA data if desired, assumes "read_sequences" is the assigned variable from read_fasta from a file containing RNA sequences instead of DNA sequences,

```
Staph_RNA_alignment <- process_allele(seqc = read_sequences,
accepted_char = c("A", "U", "C", "G"))
```

The above removes those sequences with length different from a majority of the sequences and those positions that contain a symbol other than "A", "U", "C" and "G".

Note

Any of the positions excluded by the "process_allele" function will be automatically excluded in any minSNPs analysis, it is user's responsibility to ensure that none of the excluded positions are used in the included_position argument. If any of the positions excluded by the function is also in the included_position, an error will pop up and minSNPs analysis will not run.

Part B. Tutorial

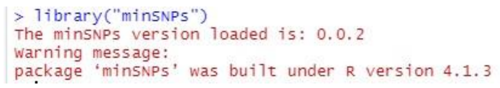
The following sections are demo making use of 3 sample FASTA files:

1. Chlamydia_1.fasta
2. Chlamydia_2.fasta
3. Chlamydia_mapped.fasta

You can download the files and follow along.

These steps are always needed in any analysis:

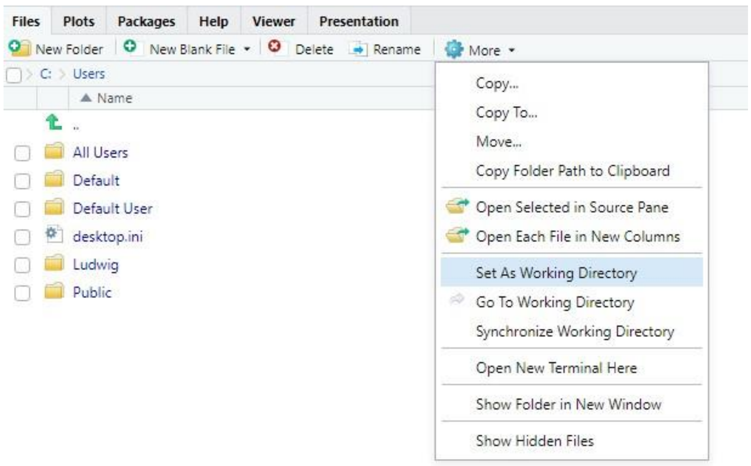
4. Import minSNPs package in R environment with command `library("minSNPs")`.



```
> library("minSNPs")
The minSNPs version loaded is: 0.0.2
Warning message:
package ‘minSNPs’ was built under R version 4.1.3
```

minSNPs output loaded version when imported successfully

5. Setting the working directory to where the files are with `setwd("<directory>")`.



RStudio provide GUI to set working directory

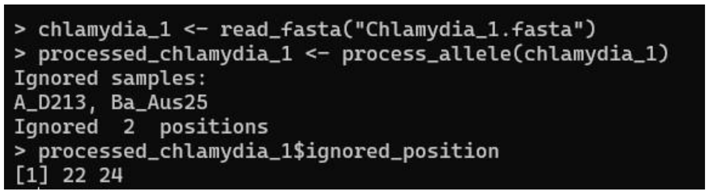
1. Reading & Cleaning orthologous SNP matrix in FASTA format

1. Use the function `read_fasta("<fasta_file>")` to read an orthologous SNP Matrix.
2. Use the function `process_allele(<read variable>)` to preprocess the matrix, see function reference for options.

Reading and processing Chlamydia_1

Code:

```
chlamydia_1 <- read_fasta("Chlamydia_1.fasta")
processed_chlamydia_1 <- process_allele(chlamydia_1)
processed_chlamydia_1$ignored_position
```



```
> chlamydia_1 <- read_fasta("Chlamydia_1.fasta")
> processed_chlamydia_1 <- process_allele(chlamydia_1)
Ignored samples:
A_D213, Ba_Aus25
Ignored  2  positions
> processed_chlamydia_1$ignored_position
[1] 22 24
```

Output

What happened?

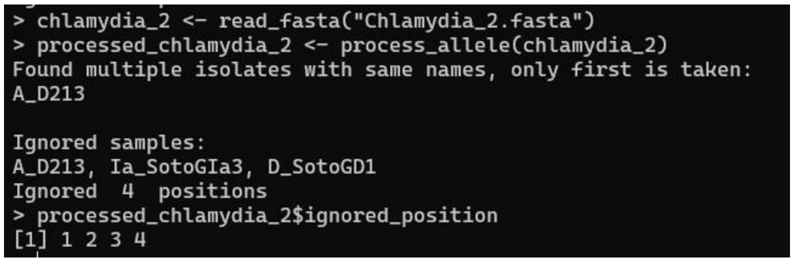
- We read Chlamydia_1.fasta and assigned it to a variable called `chlamydia_1`.
- We processed the matrix, and:

- removed 2 sequences, `A_D213` and `Ba_Aus25`, this is due to sequence with length different from others.
- ignored 2 positions, 22 and 24, this can be due to non-standard character or dash.

Reading and processing Chlamydia_2

Code:

```
chlamydia_2 <- read_fasta("Chlamydia_2.fasta")
processed_chlamydia_2 <- process_allele(chlamydia_2)
processed_chlamydia_2$ignored_position
```



```
> chlamydia_2 <- read_fasta("Chlamydia_2.fasta")
> processed_chlamydia_2 <- process_allele(chlamydia_2)
Found multiple isolates with same names, only first is taken:
A_D213

Ignored samples:
A_D213, Ia_SotoGIa3, D_SotoGD1
Ignored  4  positions
> processed_chlamydia_2$ignored_position
[1] 1 2 3 4
```

Output

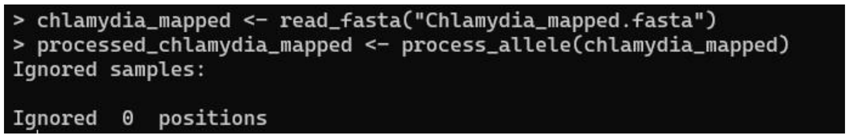
What happened?

- We read Chlamydia_2.fasta and assigned it to a variable called `chlamydia_2`.
- We processed the matrix, and:
    - removed 1 or sequences `A_D213` because of the same sequence name.
    - removed 3 sequences, `A_D213`, `Ia_SotoGIa3` and `D_SotoGD1`, this is due to sequence with length different from others.
    - ignored 4 positions, 1, 2, 3, 4, this can be due to non-standard character or dash.

Reading and processing Chlamydia_mapped

Code:

```
chlamydia_mapped <- read_fasta("Chlamydia_mapped.fasta")
processed_chlamydia_mapped <- process_allele(chlamydia_mapped)
```



```
> chlamydia_mapped <- read_fasta("Chlamydia_mapped.fasta")
> processed_chlamydia_mapped <- process_allele(chlamydia_mapped)
Ignored samples:

Ignored  0  positions
```

Output

What happened?

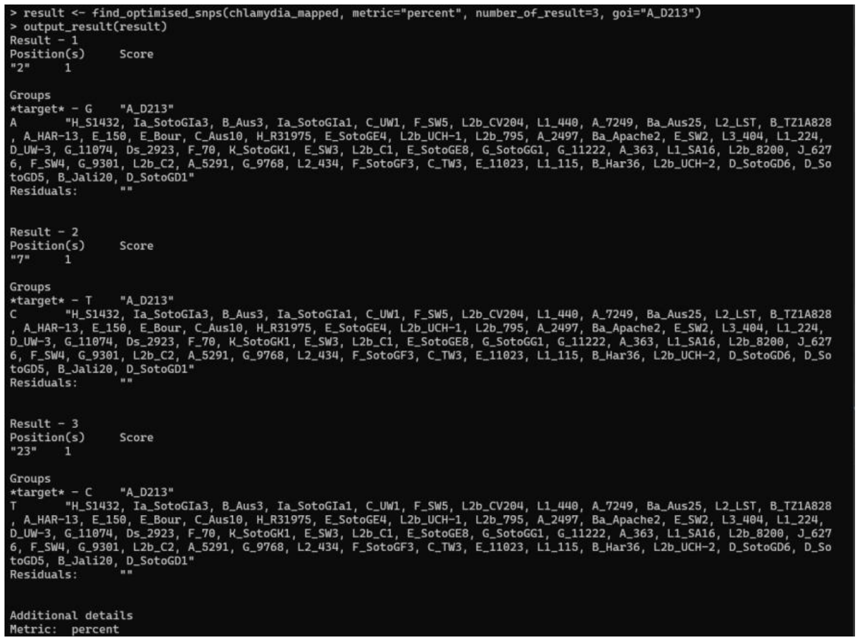
- We read Chlamydia_mapped.fasta and assigned it to a variable called `chlamydia_mapped`.
- We processed the matrix, but did not find any anomaly.

2. minSNPs in %-mode

Identify SNPs discriminating a single sequence (A_D213):

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="percent", numbe
r_of_result=3, goi="A_D213")
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="percent", number_of_result=3, goi="A_D213")
> output_result(result)
Result - 1
Position(s)     Score
"2"     1

Groups
*target* - G    "A_D213"
A       "H_S1432, Ia_SotoGIa3, B_Aus3, Ia_SotoGIa1, C_UW1, F_SW5, L2b_CV204, L1_440, A_7249, Ba_Aus25, L2_LST, B_TZ1A828
, A_HAR-13, E_150, E_Bour, C_Aus10, H_R31975, E_SotoGE4, L2b_UCH-1, L2b_795, A_2497, Ba_Apache2, E_SW2, L3_404, L1_224,
D_UW-3, G_11074, Ds_2923, F_70, K_SotoGK1, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, A_363, L1_SA16, L2b_8200, J_627
6, F_SW4, G_9301, L2b_C2, A_5291, G_9768, L2_434, F_SotoGF3, C_TW3, E_11023, L1_115, B_Har36, L2b_UCH-2, D_SotoGD6, D_So
toGD5, B_Jali20, D_SotoGD1"
Residuals:      ""


Result - 2
Position(s)     Score
"7"     1

Groups
*target* - T    "A_D213"
C       "H_S1432, Ia_SotoGIa3, B_Aus3, Ia_SotoGIa1, C_UW1, F_SW5, L2b_CV204, L1_440, A_7249, Ba_Aus25, L2_LST, B_TZ1A828
, A_HAR-13, E_150, E_Bour, C_Aus10, H_R31975, E_SotoGE4, L2b_UCH-1, L2b_795, A_2497, Ba_Apache2, E_SW2, L3_404, L1_224,
D_UW-3, G_11074, Ds_2923, F_70, K_SotoGK1, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, A_363, L1_SA16, L2b_8200, J_627
6, F_SW4, G_9301, L2b_C2, A_5291, G_9768, L2_434, F_SotoGF3, C_TW3, E_11023, L1_115, B_Har36, L2b_UCH-2, D_SotoGD6, D_So
toGD5, B_Jali20, D_SotoGD1"
Residuals:      ""


Result - 3
Position(s)     Score
"23"    1

Groups
*target* - C    "A_D213"
T       "H_S1432, Ia_SotoGIa3, B_Aus3, Ia_SotoGIa1, C_UW1, F_SW5, L2b_CV204, L1_440, A_7249, Ba_Aus25, L2_LST, B_TZ1A828
, A_HAR-13, E_150, E_Bour, C_Aus10, H_R31975, E_SotoGE4, L2b_UCH-1, L2b_795, A_2497, Ba_Apache2, E_SW2, L3_404, L1_224,
D_UW-3, G_11074, Ds_2923, F_70, K_SotoGK1, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, A_363, L1_SA16, L2b_8200, J_627
6, F_SW4, G_9301, L2b_C2, A_5291, G_9768, L2_434, F_SotoGF3, C_TW3, E_11023, L1_115, B_Har36, L2b_UCH-2, D_SotoGD6, D_So
toGD5, B_Jali20, D_SotoGD1"
Residuals:      ""


Additional details
Metric:  percent
```

Output

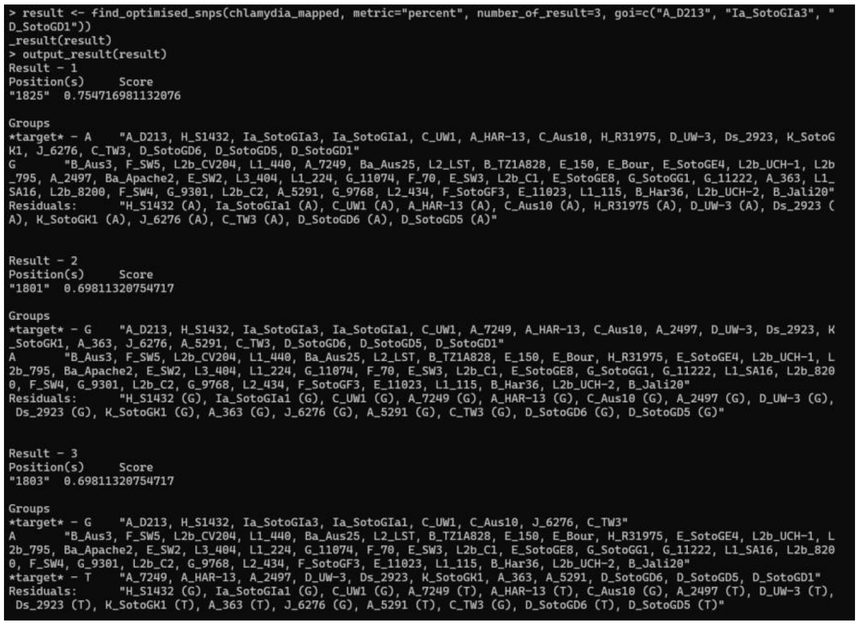
What happened?

- We identified 3 set of SNP that discriminate against `A_D213`.
- All the 3 SNPs on its own completely discriminate against `A_D213`, and the target allele is shown.

Identify SNPs discriminating multiple sequences (A_D213, Ia_SotoGIa3, D_SotoGD1):

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="percent", numbe
r_of_result=3, goi=c("A_D213", "Ia_SotoGIa3", "D_SotoGD1"))
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="percent", number_of_result=3, goi=c("A_D213", "Ia_SotoGIa3", "
D_SotoGD1"))
_result(result)
> output_result(result)
Result - 1
Position(s)     Score
"1825"  0.754716981132076

Groups
*target* - A    "A_D213, H_S1432, Ia_SotoGIa3, Ia_SotoGIa1, C_UW1, A_HAR-13, C_Aus10, H_R31975, D_UW-3, Ds_2923, K_SotoG
K1, J_6276, C_TW3, D_SotoGD6, D_SotoGD5, D_SotoGD1"
G       "B_Aus3, F_SW5, L2b_CV204, L1_440, A_7249, Ba_Aus25, L2_LST, B_TZ1A828, E_150, E_Bour, E_SotoGE4, L2b_UCH-1, L2b
_795, A_2497, Ba_Apache2, E_SW2, L3_404, L1_224, G_11074, F_70, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, A_363, L1_
SA16, L2b_8200, F_SW4, G_9301, L2b_C2, A_5291, G_9768, L2_434, F_SotoGF3, E_11023, L1_115, B_Har36, L2b_UCH-2, B_Jali20"
Residuals:      "H_S1432 (A), Ia_SotoGIa1 (A), C_UW1 (A), A_HAR-13 (A), C_Aus10 (A), H_R31975 (A), D_UW-3 (A), Ds_2923 (
A), K_SotoGK1 (A), J_6276 (A), C_TW3 (A), D_SotoGD6 (A), D_SotoGD5 (A)"


Result - 2
Position(s)     Score
"1801"  0.69811320754717

Groups
*target* - G    "A_D213, H_S1432, Ia_SotoGIa3, Ia_SotoGIa1, C_UW1, A_7249, A_HAR-13, C_Aus10, A_2497, D_UW-3, Ds_2923, K
_SotoGK1, A_363, J_6276, A_5291, C_TW3, D_SotoGD6, D_SotoGD5, D_SotoGD1"
A       "B_Aus3, F_SW5, L2b_CV204, L1_440, Ba_Aus25, L2_LST, B_TZ1A828, E_150, E_Bour, H_R31975, E_SotoGE4, L2b_UCH-1, L
2b_795, Ba_Apache2, E_SW2, L3_404, L1_224, G_11074, F_70, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, L1_SA16, L2b_820
0, F_SW4, G_9301, L2b_C2, G_9768, L2_434, F_SotoGF3, E_11023, L1_115, B_Har36, L2b_UCH-2, B_Jali20"
Residuals:      "H_S1432 (G), Ia_SotoGIa1 (G), C_UW1 (G), A_7249 (G), A_HAR-13 (G), C_Aus10 (G), A_2497 (G), D_UW-3 (G),
 Ds_2923 (G), K_SotoGK1 (G), A_363 (G), J_6276 (G), A_5291 (G), C_TW3 (G), D_SotoGD6 (G), D_SotoGD5 (G)"


Result - 3
Position(s)     Score
"1803"  0.69811320754717

Groups
*target* - G    "A_D213, H_S1432, Ia_SotoGIa3, Ia_SotoGIa1, C_UW1, C_Aus10, J_6276, C_TW3"
A       "B_Aus3, F_SW5, L2b_CV204, L1_440, Ba_Aus25, L2_LST, B_TZ1A828, E_150, E_Bour, H_R31975, E_SotoGE4, L2b_UCH-1, L
2b_795, Ba_Apache2, E_SW2, L3_404, L1_224, G_11074, F_70, E_SW3, L2b_C1, E_SotoGE8, G_SotoGG1, G_11222, L1_SA16, L2b_820
0, F_SW4, G_9301, L2b_C2, G_9768, L2_434, F_SotoGF3, E_11023, L1_115, B_Har36, L2b_UCH-2, B_Jali20"
*target* - T    "A_7249, A_HAR-13, A_2497, D_UW-3, Ds_2923, K_SotoGK1, A_363, A_5291, D_SotoGD6, D_SotoGD5, D_SotoGD1"
Residuals:      "H_S1432 (G), Ia_SotoGIa1 (G), C_UW1 (G), A_7249 (T), A_HAR-13 (T), C_Aus10 (G), A_2497 (T), D_UW-3 (T),
 Ds_2923 (T), K_SotoGK1 (T), A_363 (T), J_6276 (G), A_5291 (T), C_TW3 (G), D_SotoGD6 (T), D_SotoGD5 (T)"
```

Output

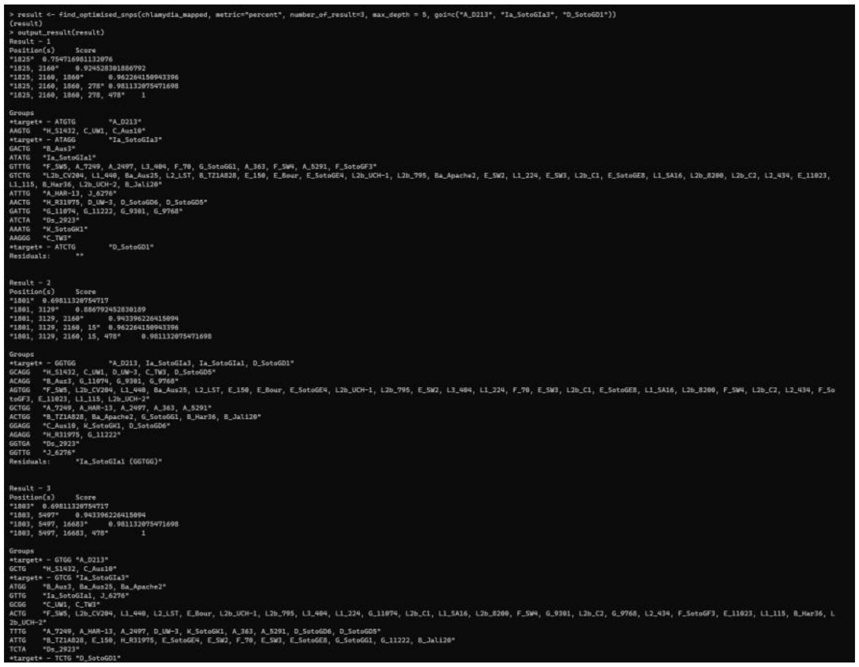
What happened?

- We identified 3 sets of SNP that discriminate against `A_D213`, `Ia_SotoGIa3`, `D_SotoGD1`.
- None of the 3 sets completely discriminate against all the sequences, the selected SNPs and how it would group the sequences are shown.

Identify SNPs discriminating multiple sequences (A_D213, Ia_SotoGIa3, D_SotoGD1) II:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="percent", numbe
r_of_result=3, max_depth = 5, goi=c("A_D213", "Ia_SotoGIa3", "D_SotoGD1
"))
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="percent", number_of_result=3, max_depth = 5, goi=c("A_D213", "Ia_SotoGIa3", "D_SotoGD1"))
(result)
> output_result(result)
Result - 1
Position(s)     Score
"1825"  0.754716981132076
"1825, 2160"    0.924528301886792
"1825, 2160, 1860"      0.962264150943396
"1825, 2160, 1860, 278" 0.981132075471698
"1825, 2160, 1860, 278, 478"    1

Groups
*target* - ATGTG        "A_D213"
AAGTG   "H_S1432, C_UW1, C_Aus10"
*target* - ATAGG        "Ia_SotoGIa3"
GACTG   "B_Aus3"
ATATG   "Ia_SotoGIa1"
GTTTG   "F_SW5, A_7249, A_2497, L3_404, F_70, G_SotoGG1, A_363, F_SW4, A_5291, F_SotoGF3"
GTCTG   "L2b_CV204, L1_440, Ba_Aus25, L2_LST, B_TZ1A828, E_150, E_Bour, E_SotoGE4, L2b_UCH-1, L2b_795, Ba_Apache2, E_SW2, L1_224, E_SW3, L2b_C1, E_SotoGE8, L1_SA16, L2b_8200, L2b_C2, L2_434, E_11023,
L1_115, B_Har36, L2b_UCH-2, B_Jali20"
ATTTG   "A_HAR-13, J_6276"
AACTG   "H_R31975, D_UW-3, D_SotoGD6, D_SotoGD5"
GATTG   "G_11074, G_11222, G_9301, G_9768"
ATCTA   "Ds_2923"
AAATG   "K_SotoGK1"
AAGGG   "C_TW3"
*target* - ATCTG        "D_SotoGD1"
Residuals:      ""


Result - 2
Position(s)     Score
"1801"  0.69811320754717
"1801, 3129"    0.886792452830189
"1801, 3129, 2160"      0.943396226415094
"1801, 3129, 2160, 15"  0.962264150943396
"1801, 3129, 2160, 15, 478"     0.981132075471698

Groups
*target* - GGTGG        "A_D213, Ia_SotoGIa3, Ia_SotoGIa1, D_SotoGD1"
GCAGG   "H_S1432, C_UW1, D_UW-3, C_TW3, D_SotoGD5"
ACAGG   "B_Aus3, G_11074, G_9301, G_9768"
AGTGG   "F_SW5, L2b_CV204, L1_440, Ba_Aus25, L2_LST, E_150, E_Bour, E_SotoGE4, L2b_UCH-1, L2b_795, E_SW2, L3_404, L1_224, F_70, E_SW3, L2b_C1, E_SotoGE8, L1_SA16, L2b_8200, F_SW4, L2b_C2, L2_434, F_So
toGF3, E_11023, L1_115, L2b_UCH-2"
GCTGG   "A_7249, A_HAR-13, A_2497, A_363, A_5291"
ACTGG   "B_TZ1A828, Ba_Apache2, G_SotoGG1, B_Har36, B_Jali20"
GGAGG   "C_Aus10, K_SotoGK1, D_SotoGD6"
AGAGG   "H_R31975, G_11222"
GGTGA   "Ds_2923"
GGTTG   "J_6276"
Residuals:      "Ia_SotoGIa1 (GGTGG)"


Result - 3
Position(s)     Score
"1803"  0.69811320754717
"1803, 5497"    0.943396226415094
"1803, 5497, 16683"     0.981132075471698
"1803, 5497, 16683, 478"        1

Groups
*target* - GTGG "A_D213"
GCTG    "H_S1432, C_Aus10"
*target* - GTCG "Ia_SotoGIa3"
ATGG    "B_Aus3, Ba_Aus25, Ba_Apache2"
GTTG    "Ia_SotoGIa1, J_6276"
GCGG    "C_UW1, C_TW3"
ACTG    "F_SW5, L2b_CV204, L1_440, L2_LST, E_Bour, L2b_UCH-1, L2b_795, L3_404, L1_224, G_11074, L2b_C1, L1_SA16, L2b_8200, F_SW4, G_9301, L2b_C2, G_9768, L2_434, F_SotoGF3, E_11023, L1_115, B_Har36, L
2b_UCH-2"
TTTG    "A_7249, A_HAR-13, A_2497, D_UW-3, K_SotoGK1, A_363, A_5291, D_SotoGD6, D_SotoGD5"
ATTG    "B_TZ1A828, E_150, H_R31975, E_SotoGE4, E_SW2, F_70, E_SW3, E_SotoGE8, G_SotoGG1, G_11222, B_Jali20"
TCTA    "Ds_2923"
*target* - TCTG "D_SotoGD1"
```

Output

What happened?

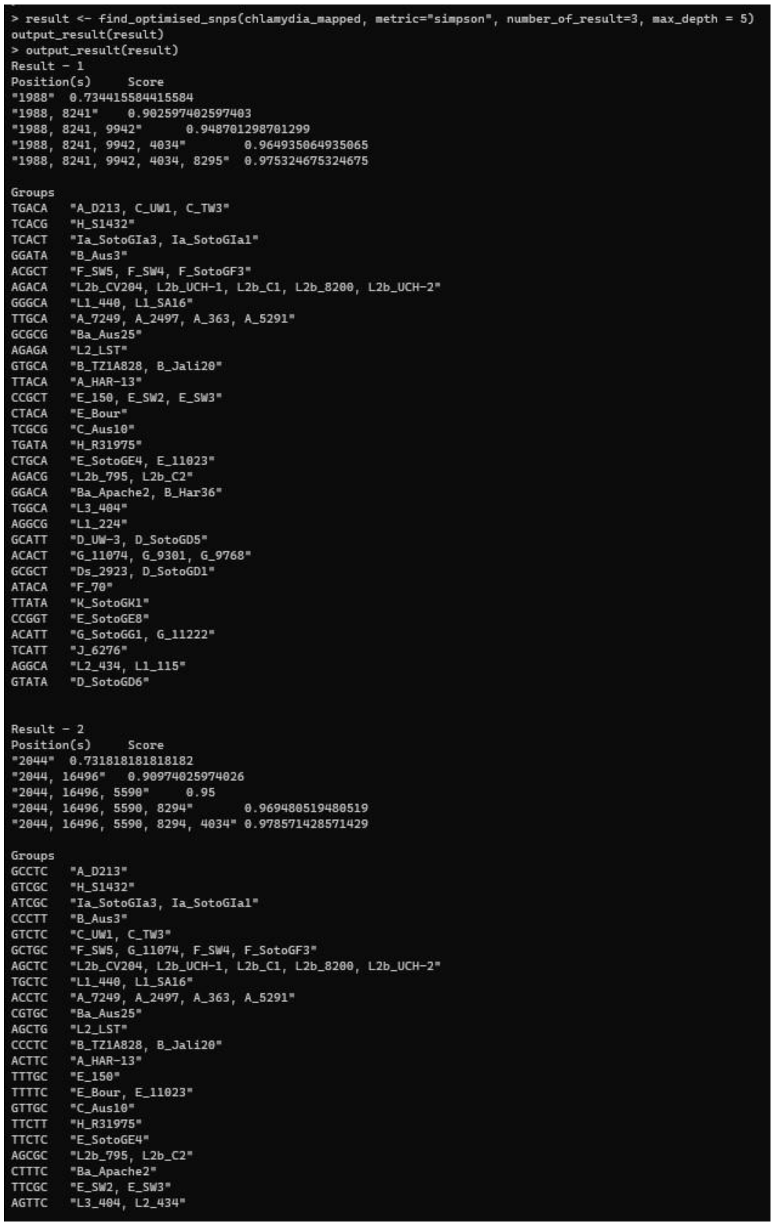
- We identified 3 sets of SNP that discriminate against `A_D213`, `Ia_SotoGIa3`, `D_SotoGD1`, each set can have up to 5 SNPs.
- Set 1 and 3 completely discriminate against all the sequences with (5 and 4 SNPs respectively), while set 2 still fail to completely discriminate against all the 3 isolates; the selected SNPs and how it would group the sequences are shown.

2. minSNPs in D-mode

Running D-mode analysis as-is:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", numbe
r_of_result=3, max_depth = 5)
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="simpson", number_of_result=3, max_depth = 5)
output_result(result)
> output_result(result)
Result - 1
Position(s)     Score
"1988"  0.734415584415584
"1988, 8241"    0.902597402597403
"1988, 8241, 9942"      0.948701298701299
"1988, 8241, 9942, 4034"        0.964935064935065
"1988, 8241, 9942, 4034, 8295"  0.975324675324675

Groups
TGACA   "A_D213, C_UW1, C_TW3"
TCACG   "H_S1432"
TCACT   "Ia_SotoGIa3, Ia_SotoGIa1"
GGATA   "B_Aus3"
ACGCT   "F_SW5, F_SW4, F_SotoGF3"
AGACA   "L2b_CV204, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
GGGCA   "L1_440, L1_SA16"
TTGCA   "A_7249, A_2497, A_363, A_5291"
GCGCG   "Ba_Aus25"
AGAGA   "L2_LST"
GTGCA   "B_TZ1A828, B_Jali20"
TTACA   "A_HAR-13"
CCGCT   "E_150, E_SW2, E_SW3"
CTACA   "E_Bour"
TCGCG   "C_Aus10"
TGATA   "H_R31975"
CTGCA   "E_SotoGE4, E_11023"
AGACG   "L2b_795, L2b_C2"
GGACA   "Ba_Apache2, B_Har36"
TGGCA   "L3_404"
AGGCG   "L1_224"
GCATT   "D_UW-3, D_SotoGD5"
ACACT   "G_11074, G_9301, G_9768"
GCGCT   "Ds_2923, D_SotoGD1"
ATACA   "F_70"
TTATA   "K_SotoGK1"
CCGGT   "E_SotoGE8"
ACATT   "G_SotoGG1, G_11222"
TCATT   "J_6276"
AGGCA   "L2_434, L1_115"
GTATA   "D_SotoGD6"


Result - 2
Position(s)     Score
"2044"  0.731818181818182
"2044, 16496"   0.90974025974026
"2044, 16496, 5590"     0.95
"2044, 16496, 5590, 8294"       0.969480519480519
"2044, 16496, 5590, 8294, 4034" 0.978571428571429

Groups
GCCTC   "A_D213"
GTCGC   "H_S1432"
ATCGC   "Ia_SotoGIa3, Ia_SotoGIa1"
CCCTT   "B_Aus3"
GTCTC   "C_UW1, C_TW3"
GCTGC   "F_SW5, G_11074, F_SW4, F_SotoGF3"
AGCTC   "L2b_CV204, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
TGCTC   "L1_440, L1_SA16"
ACCTC   "A_7249, A_2497, A_363, A_5291"
CGTGC   "Ba_Aus25"
AGCTG   "L2_LST"
CCCTC   "B_TZ1A828, B_Jali20"
ACTTC   "A_HAR-13"
TTTGC   "E_150"
TTTTC   "E_Bour, E_11023"
GTTGC   "C_Aus10"
TTCTT   "H_R31975"
TTCTC   "E_SotoGE4"
AGCGC   "L2b_795, L2b_C2"
CTTTC   "Ba_Apache2"
TTCGC   "E_SW2, E_SW3"
AGTTC   "L3_404, L2_434"
```

Output

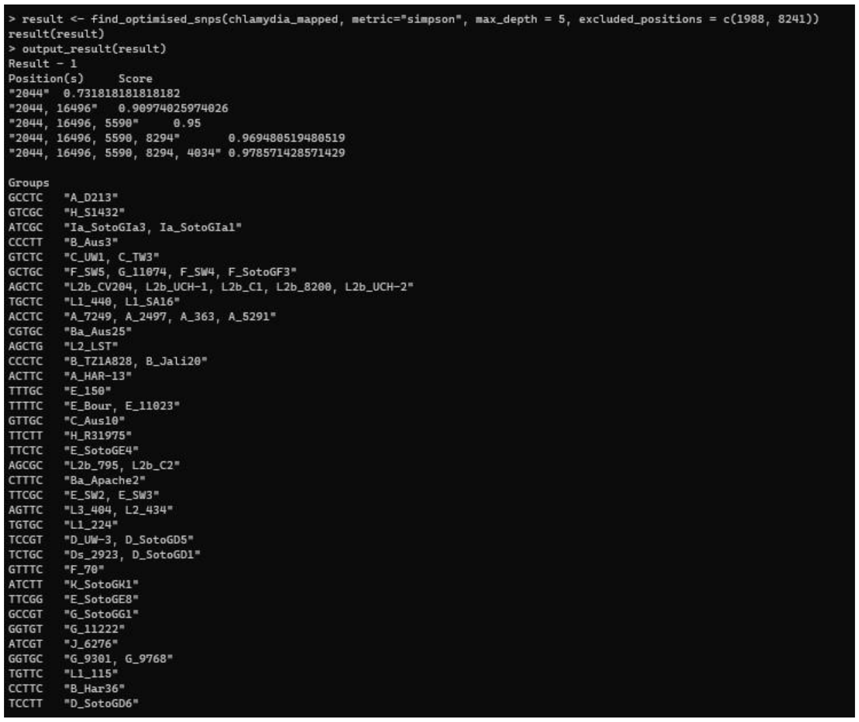
What happened?

- We identified 3 sets of SNP that have the highest Simpson score, each set can have up to 5 SNPs, only terminating if the Simpson is 1.
- The selected SNPs and how the SNPs would group the sequences are shown.

Exclude specific SNPs:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_d
epth = 5, excluded_positions = c(1988, 8241))
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_depth = 5, excluded_positions = c(1988, 8241))
result(result)
> output_result(result)
Result - 1
Position(s)     Score
"2044"  0.731818181818182
"2044, 16496"   0.909740259740260
"2044, 16496, 5590"     0.95
"2044, 16496, 5590, 8294"       0.969480519480519
"2044, 16496, 5590, 8294, 4034" 0.978571428571429

Groups
GCCTC   "A_D213"
GTCGC   "H_S1432"
ATCGC   "Ia_SotoGIa3, Ia_SotoGIa1"
CCCTT   "B_Aus3"
GTCTC   "C_UW1, C_TW3"
GCTGC   "F_SW5, G_11074, F_SW4, F_SotoGF3"
AGCTC   "L2b_CV204, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
TGCTC   "L1_440, L1_SA16"
ACCTC   "A_7249, A_2497, A_363, A_5291"
CGTGC   "Ba_Aus25"
AGCTG   "L2_LST"
CCCTC   "B_TZ1A828, B_Jali20"
ACTTC   "A_HAR-13"
TTTGC   "E_150"
TTTTC   "E_Bour, E_11023"
GTTGC   "C_Aus10"
TTCTT   "H_R31975"
TTCTC   "E_SotoGE4"
AGCGC   "L2b_795, L2b_C2"
CTTTC   "Ba_Apache2"
TTCGC   "E_SW2, E_SW3"
AGTTC   "L3_404, L2_434"
TGTGC   "L1_224"
TCCGT   "D_UW-3, D_SotoGD5"
TCTGC   "Ds_2923, D_SotoGD1"
GTTTC   "F_70"
ATCTT   "K_SotoGK1"
TTCGG   "E_SotoGE8"
GCCGT   "G_SotoGG1"
GGTGT   "G_11222"
ATCGT   "J_6276"
GGTGC   "G_9301, G_9768"
TGTTC   "L1_115"
CCTTC   "B_Har36"
TCCTT   "D_SotoGD6"
```

Output

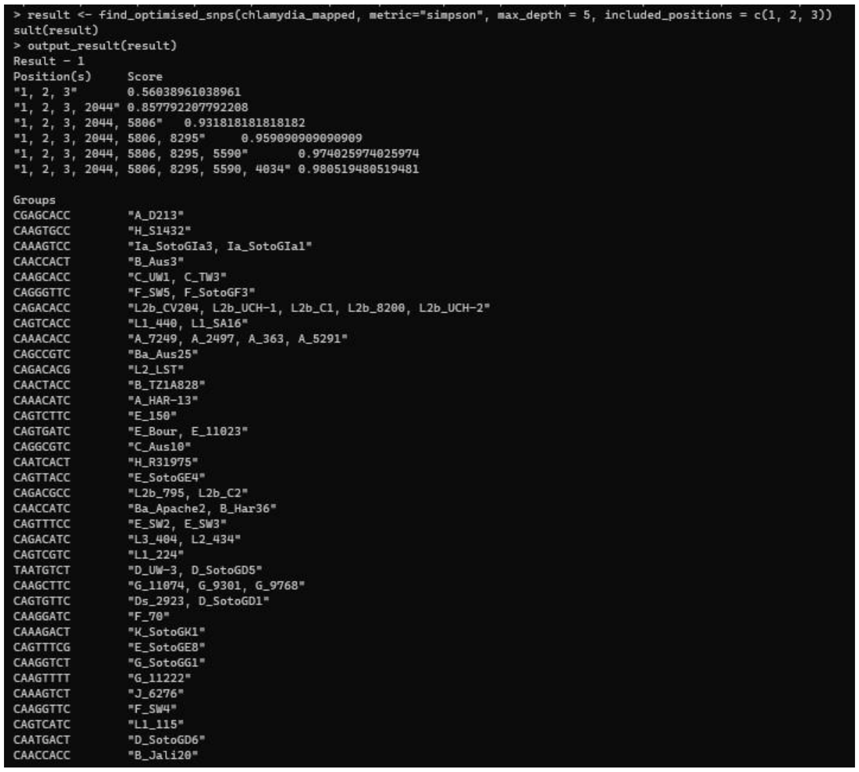
What happened?

- We identified a set of maximum 5 SNPs that has the highest Simpson score, the set will not include SNPs at 1988, and 8241.
- The selected SNPs and how the SNPs would group the sequences are shown.

Include specific SNPs:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_d
epth = 5, included_positions = c(1, 2, 3))
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_depth = 5, included_positions = c(1, 2, 3))
sult(result)
> output_result(result)
Result - 1
Position(s)     Score
"1, 2, 3"       0.56038961038961
"1, 2, 3, 2044" 0.857792207792208
"1, 2, 3, 2044, 5806"   0.931818181818182
"1, 2, 3, 2044, 5806, 8295"     0.959090909090909
"1, 2, 3, 2044, 5806, 8295, 5590"       0.974025974025974
"1, 2, 3, 2044, 5806, 8295, 5590, 4034" 0.980519480519481

Groups
CGAGCACC        "A_D213"
CAAGTGCC        "H_S1432"
CAAAGTCC        "Ia_SotoGIa3, Ia_SotoGIa1"
CAACCACT        "B_Aus3"
CAAGCACC        "C_UW1, C_TW3"
CAGGGTTC        "F_SW5, F_SotoGF3"
CAGACACC        "L2b_CV204, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
CAGTCACC        "L1_440, L1_SA16"
CAAACACC        "A_7249, A_2497, A_363, A_5291"
CAGCCGTC        "Ba_Aus25"
CAGACACG        "L2_LST"
CAACTACC        "B_TZ1A828"
CAAACATC        "A_HAR-13"
CAGTCTTC        "E_150"
CAGTGATC        "E_Bour, E_11023"
CAGGCGTC        "C_Aus10"
CAATCACT        "H_R31975"
CAGTTACC        "E_SotoGE4"
CAGACGCC        "L2b_795, L2b_C2"
CAACCATC        "Ba_Apache2, B_Har36"
CAGTTTCC        "E_SW2, E_SW3"
CAGACATC        "L3_404, L2_434"
CAGTCGTC        "L1_224"
TAATGTCT        "D_UW-3, D_SotoGD5"
CAAGCTTC        "G_11074, G_9301, G_9768"
CAGTGTTC        "Ds_2923, D_SotoGD1"
CAAGGATC        "F_70"
CAAAGACT        "K_SotoGK1"
CAGTTTCG        "E_SotoGE8"
CAAGGTCT        "G_SotoGG1"
CAAGTTTT        "G_11222"
CAAAGTCT        "J_6276"
CAAGGTTC        "F_SW4"
CAGTCATC        "L1_115"
CAATGACT        "D_SotoGD6"
CAACCACC        "B_Jali20"
```

Output

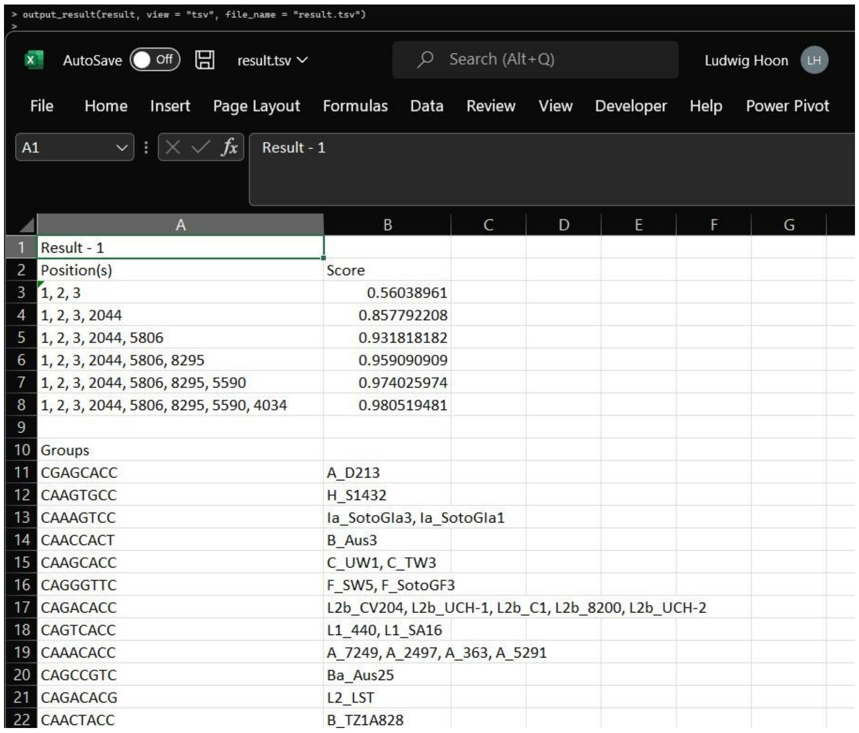
What happened?

- We identified a set of SNPs that has the highest Simpson score, the set will include SNPs at 1, 2, 3, and up to additional 5 SNPs.
- The selected SNPs and how the SNPs would group the sequences are shown.

3. Save result to TSV

Code:

```
output_result(result, view = "tsv", file_name = "result.tsv")
```



```
> output_result(result, view = "tsv", file_name = "result.tsv")
>
```

AutoSave Off
result.tsv
Search (Alt+Q)
Ludwig Hoon
File Home Insert Page Layout Formulas Data Review View Developer Help Power Pivot
A1
Result - 1

	A	B	C	D	E	F	G
1	Result - 1						
2	Position(s)	Score					
3	1, 2, 3	0.56038961					
4	1, 2, 3, 2044	0.857792208					
5	1, 2, 3, 2044, 5806	0.931818182					
6	1, 2, 3, 2044, 5806, 8295	0.959090909					
7	1, 2, 3, 2044, 5806, 8295, 5590	0.974025974					
8	1, 2, 3, 2044, 5806, 8295, 5590, 4034	0.980519481					
9							
10	Groups						
11	CGAGCACC	A_D213					
12	CAAGTGCC	H_S1432					
13	CAAAGTCC	Ia_SotoGla3, Ia_SotoGla1					
14	CAACCACT	B_Aus3					
15	CAAGCACC	C_UW1, C_TW3					
16	CAGGGTTC	F_SW5, F_SotoGF3					
17	CAGACACC	L2b_CV204, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2					
18	CAGTCACC	L1_440, L1_SA16					
19	CAAACACC	A_7249, A_2497, A_363, A_5291					
20	CAGCCGTC	Ba_Aus25					
21	CAGACACG	L2_LST					
22	CAACTACC	B_TZ1A828					

Output

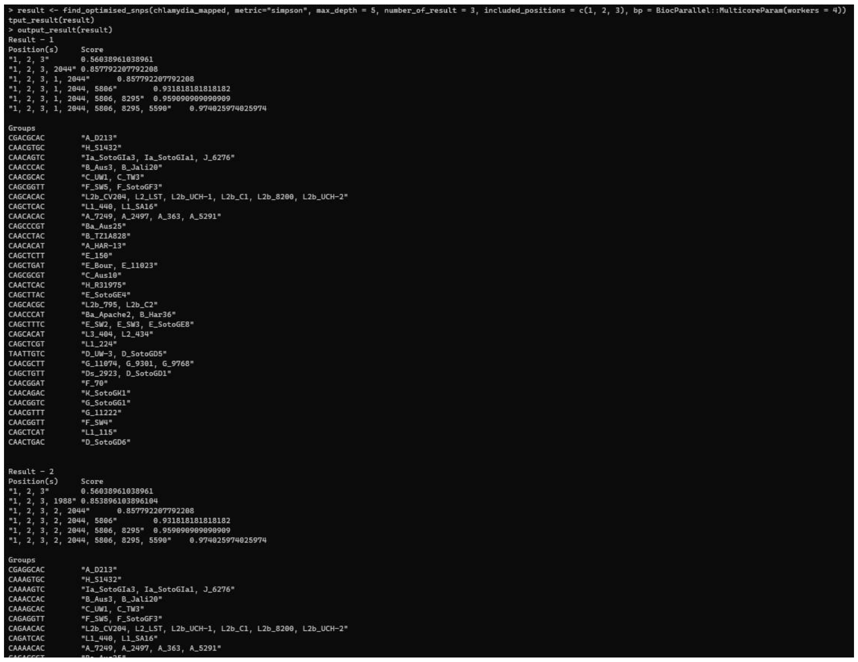
What happened?

- The result is saved at a file called `result.tsv`, which can be opened in excel, with tab as delimiter.

4. Parallelizing runs

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_d
epth = 5, number_of_result = 3, included_positions = c(1, 2, 3), bp = B
iocParallel::MulticoreParam(workers = 4, ))
output_result(result)
```



```
> result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_depth = 5, number_of_result = 3, included_positions = c(1, 2, 3), bp = BiocParallel::MulticoreParam(workers = 4))
tput_result(result)
> output_result(result)
Result - 1
Position(s)     Score
"1, 2, 3"       0.56038961038961
"1, 2, 3, 2044" 0.857792207792208
"1, 2, 3, 1, 2044"      0.857792207792208
"1, 2, 3, 1, 2044, 5806"        0.931818181818182
"1, 2, 3, 1, 2044, 5806, 8295"  0.959090909090909
"1, 2, 3, 1, 2044, 5806, 8295, 5590"    0.974025974025974

Groups
CGACGCAC        "A_D213"
CAACGTGC        "H_S1432"
CAACAGTC        "Ia_SotoGIa3, Ia_SotoGIa1, J_6276"
CAACCCAC        "B_Aus3, B_Jali20"
CAACGCAC        "C_UW1, C_TW3"
CAGCGGTT        "F_SW5, F_SotoGF3"
CAGCACAC        "L2b_CV204, L2_LST, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
CAGCTCAC        "L1_440, L1_SA16"
CAACACAC        "A_7249, A_2497, A_363, A_5291"
CAGCCCGT        "Ba_Aus25"
CAACCTAC        "B_TZ1A828"
CAACACAT        "A_HAR-13"
CAGCTCTT        "E_150"
CAGCTGAT        "E_Bour, E_11023"
CAGCGCGT        "C_Aus10"
CAACTCAC        "H_R31975"
CAGCTTAC        "E_SotoGE4"
CAGCACGC        "L2b_795, L2b_C2"
CAACCCAT        "Ba_Apache2, B_Har36"
CAGCTTTC        "E_SW2, E_SW3, E_SotoGE8"
CAGCACAT        "L3_404, L2_434"
CAGCTCGT        "L1_224"
TAATTGTC        "D_UW-3, D_SotoGD5"
CAACGCTT        "G_11074, G_9301, G_9768"
CAGCTGTT        "Ds_2923, D_SotoGD1"
CAACGGAT        "F_70"
CAACAGAC        "K_SotoGK1"
CAACGGTC        "G_SotoGG1"
CAACGTTT        "G_11222"
CAACGGTT        "F_SW4"
CAGCTCAT        "L1_115"
CAACTGAC        "D_SotoGD6"


Result - 2
Position(s)     Score
"1, 2, 3"       0.56038961038961
"1, 2, 3, 1988" 0.853896103896104
"1, 2, 3, 2, 2044"      0.857792207792208
"1, 2, 3, 2, 2044, 5806"        0.931818181818182
"1, 2, 3, 2, 2044, 5806, 8295"  0.959090909090909
"1, 2, 3, 2, 2044, 5806, 8295, 5590"    0.974025974025974

Groups
CGAGGCAC        "A_D213"
CAAAGTGC        "H_S1432"
CAAAAGTC        "Ia_SotoGIa3, Ia_SotoGIa1, J_6276"
CAAACCAC        "B_Aus3, B_Jali20"
CAAAGCAC        "C_UW1, C_TW3"
CAGAGGTT        "F_SW5, F_SotoGF3"
CAGAACAC        "L2b_CV204, L2_LST, L2b_UCH-1, L2b_C1, L2b_8200, L2b_UCH-2"
CAGATCAC        "L1_440, L1_SA16"
CAAAACAC        "A_7249, A_2497, A_363, A_5291"
```

Output

What happened?

- The is similar to analysis before, except that we parallelised with 4 cores, by adding the bp argument.

Others

1. Functions documentation can be found at: https://ludwighoon.github.io/minSNPs/reference/index.html
2. Cheat sheet for R