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:

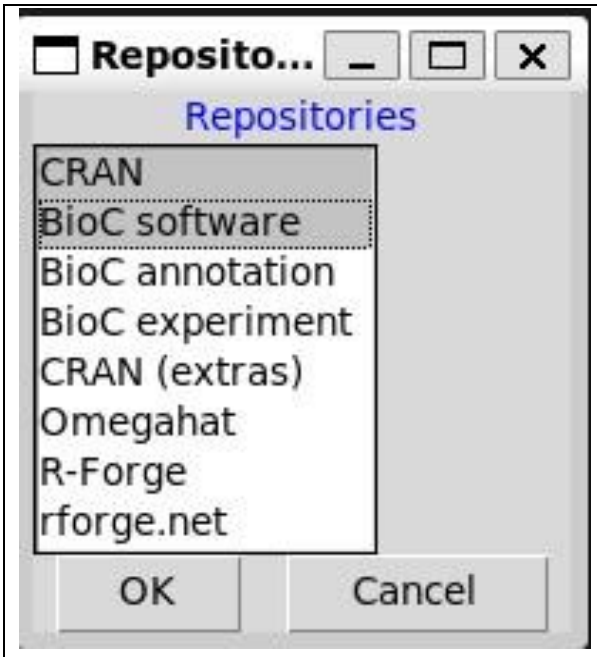
1. For users who are unfamiliar with the UNIX operating system and/or do not have access to a high performance computer cluster (HPC), we suggest running minSNPs in RStudio on a PC. RStudio is designed to facilitate the development and operation of R packages by providing a workspace that is quicker 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 HPC. 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 and commands for R Terminal can be copied and run within the terminal in RStudio, and where RStudio provides a graphical user interface (GUI) 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 an R terminal or RStudio, run the command `setRepositories()` and make sure that both **CRAN** and **BioC software** are selected. This can be confirmed with `getOption("repos")`.



Prompt in CLI

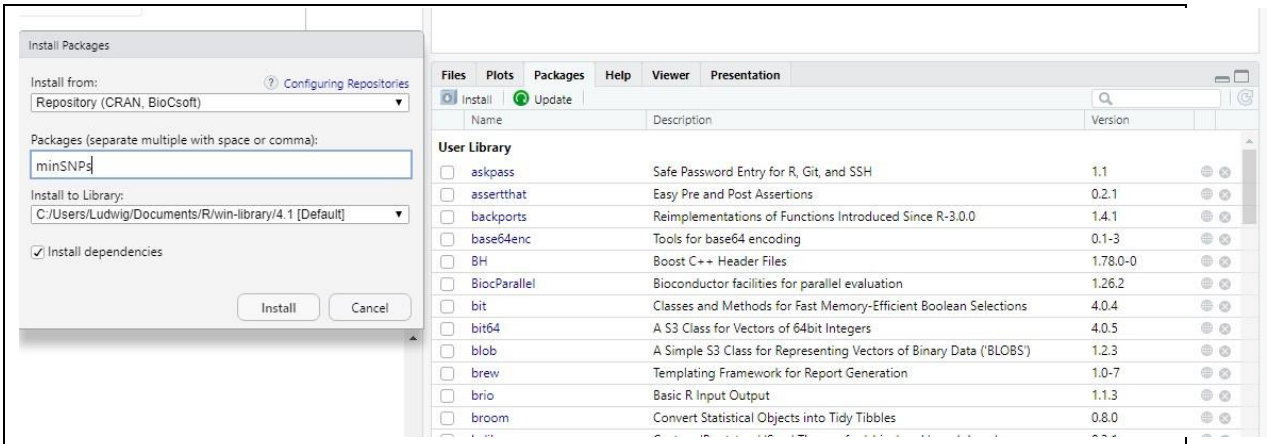
```
> setRepositories()
--- Please select repositories for use in this session ---


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 cancel
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.



Installing with GUI in RStudio

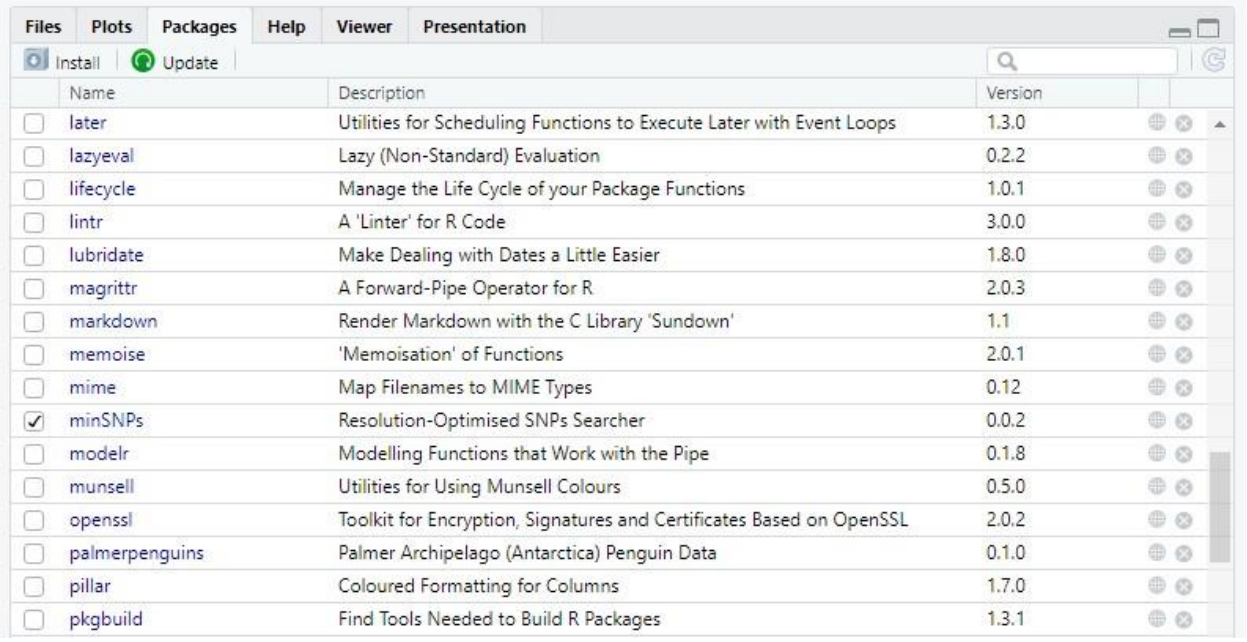
```
> install.packages("minSNPs")
Installing package into 'C:/Users/Ludwig/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/minSNPs_0.0.2.zip'
Content type 'application/zip' length 888976 bytes (868 KB)
downloaded 868 KB

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 of minSNPs. 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 is provided below the syntax.

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



RStudio's GUI to import package to environment

Part A. The basic functions

User's thoughts / Tips
Before any of the following functions may be used, minSNPs package needs to be loaded after installation. This needs to be done in the beginning of each analysis with: `library(minSNPs)`.

1. Sequence data input.

minSNPs derives resolution optimised 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 an R terminal, the tab key can be used to show suggestions or autocomplete the file path.

User's thoughts / Tips

The above function can be simplified if the working directory is defined first.
 	
You can check which directory is set as your working directory using the R function getwd().
 	
If you wish to change your working directory, 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 file extension.
 	
This will load the file "Staph_genomes.fasta.txt" from the working directory into the R environment, 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.
 	
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 previous example.

User's thoughts / Tips

In some instances, there may be a warning message similar to below:
 	
`Warning message:`
`In readLines(file) : incomplete final line found on 'Mu50.fasta'`
 	
This is usually safe to ignore, the warning serves to notice the user that R cannot find newline character at the end of the file.

2. Dealing with alignments containing non-standard symbols or indels.

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

The function performs two different tasks, i.e., (1) remove sequences within the alignment which differ length from most other sequences, and (2) ignores for analysis purposes, positions in the alignment where at least one of the sequences contains a non-standard symbol at that position.

The first task is non-modifiable. To analyse a matrix while retaining any indels represented by "-" characters, make sure that all the sequences within the alignment have the same length, i.e., for all sequences without insertions, use "-" at the inserted positions. Likewise for deletions, use "-" to represent the deleted bases in the relevant sequences.

The second task can be modified. By default, all positions in the alignment for which at least one sequence has a non-standard symbol, including failed sequencing ("N") and ambiguity codes, are ignored for analysis purposes, but the accepted symbols can be modified. Similarly, positions with a "-" are ignored by default, however they can be accepted either by adding the symbol 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 acceptable character or to ignore positions where at least one of the sequences contains a "-" at that position. By default, any position containing a "-" is ignored for analysis purposes (i.e., the default is dash_ignore=TRUE).
- accepted_char: This specifies the list of accepted character, by default, this is c("A","C","T","G"), but can be modified.

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. The default value for dash_ignore is TRUE. If this argument is omitted, all positions containing a "-" are ignored for analysis purposes. The below example assumes "read_sequences" is the assigned variable from read_fasta of a file containing multiple sequences of varied length: `Staph_alignment <- process_allele(seqc = read_sequences)`

The above removes those sequences with a length different from the majority of the sequences. It also ignores for analysis purposes any positions in the alignment where at least one sequence contains a symbol other than "A", "C", "T", or "G".

If "read_sequences" is the assigned variable from read_fasta of a file containing multiple sequences of varied length and indels that the user wishes to consider in the analysis:

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

The above will remove those sequences with a length different from the majority of the sequences and ignore for analysis purposes those positions that contain a symbol other than "A", "C", "T", "G", or "-".

minSNPs can be used to analyse RNA data if desired. The below example assumes "read_sequences" is the assigned variable from read_fasta of 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 a length different from a majority of the sequences and ignores for analysis purposes those positions that contain a symbol other than "A", "U", "C", or "G".

Note

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

3. Derivation of resolution optimised SNP sets.

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

minSNPs can derive optimised SNP sets according to two different metrics of resolving power, "percent" metric and "simpson" metric. An explanation of each is provided below.

The **syntax** for deriving optimised 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 percent mode only>")
```

Arguments

- seqc: The value should be replaced with the user-assigned variable from the the "process_allele" or the "read_fasta" function.
- metric: The options "percent" and "simpson" specify the metric that will be used. The default is "simpson" but it is good practice to always specify the metric. You can only specify one option at a time; minSNPs cannot simultaneously perform SNP set derivations using both metrics.
    - The "percent" metric yields SNP sets that are optimised to discriminate a subset of the sequences in the alignment (the "group of interest (goi)) from all other sequences in the alignment. Sensitivity is constrained to 100%, and the SNPs sets are optimised with respect to specificity which is expressed as a decimal between 0.0 and 1.0.
    - The "simpson" metric yields SNP sets that are optimised to organise sequences into different groups. The SNPs are selected based on maximum cumulative Simpson's index of diversity. Similar to "percent" metric, the diversity index is expressed as a decimal between 0.0 and 1.0.

  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 input 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 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 optimised 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 sequence names in the input

FASTA file, and must be an exact match. Ten sets of SNPs consisting of a maximum of 5 SNPs 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_optimised_SNPs(Staph_SNP_matrix,
metric="percent", number_of_result=10, max_depth=5, goi=c("seq12",
"seq38", "seq50"))
```

is equivalent to the previous example.

A hypothetical example of a script to search in "simpson" mode is:

```
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 optimised with respect to the Simpsons index of diversity. Ten sets of SNPs consisting of a maximum of 8 SNPs will be derived. The results will be assigned to a variable with the user selected name "Staph_D_SNPs".

4. 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">)
```

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 in R ("") or to a tsv file ("tsv").
- file_name: This specifies the filename if the output format is tsv. The file extension is not required.

User's thoughts / Tips
A hypothetical example of using the function in a script to save the results to a tab-delimited file is: `output_result(result = Staph_SNPs_seqs12_38_50, view = "tsv", file_name = "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 required. 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), the arguments view and file_name can be omitted.

The output shows the selected SNPs, the specificity/index of diversity achieved as well as the allelic profiles and how the sequences are grouped. If the selected metric is "percent", the residuals (sequences that the SNP sets cannot discriminate from the goi) are also shown. For an example of output, see tutorial section.

5. Additional functions and features

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

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

  This can be specified as an additional argument for the **"find_optimised_snps"** function. Either or both parameters can be used at the same time. However, the same positions cannot be included in both arguments, otherwise an 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 resulting 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 "percent" 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 must be excluded from 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 resulting SNP sets will include SNPs 1, 2, 3 and up to an additional 5 SNPs, and none of the SNP sets will include SNPs 4, 5, 6. The

actual number of SNPs returned will differ between sets, because the function will stop searching once the SNP set achieves a 100% specificity (expressed as 1.0, i.e., the SNP set completely distinguishes all members of the group of interest from all other sequences).

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

It is possible to define the arrays for "goi", "included_positions", and "excluded_positions" using a variable to simplify 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 the **"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 positions to be included>,
excluded_positions=<list of positions to be excluded>,
bp=<parallelization>)
```

Arguments

- bp: This specifies whether to parallelise the search. By default, the function is not parallelised and run with only a single core. To enable parallelisation, specify this optional argument with `BiocParallel::MulticoreParam()`. That will correctly detect the number of available cores in PC and make use of the available cores. However, to parallelise the search when using an HPC that makes use of a 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) minus 2.

User's thoughts / Tips

minSNPs will run on a PC with no issues, defaulting to using only 1 core and no changes to the "find_optimised_snps" are needed. The example below will not work using a 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())
```

It is not necessary to specify the number of workers when using a non-Windows 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 parallelisation depends on BiocParallel, for more information, further information can be found here.

Part B. Tutorial

The following sections are a demonstration making use of three sample FASTA files:

- Chlamydia_1.fasta
- Chlamydia_2.fasta
- Chlamydia_mapped.fasta

You can download the files and follow along.

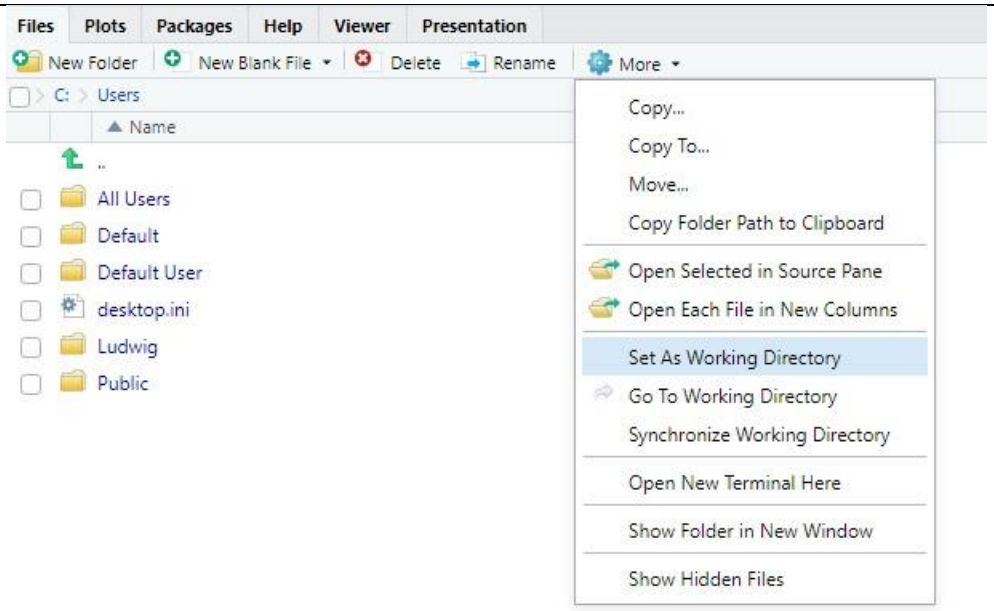
These steps are always needed in any analysis:

1. Import the installed minSNPs package in the R environment using the `library("minSNPs")` function.

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

minSNPs outputs the loaded version when imported successfully

2. Set the working directory to where the files are located using the `setwd("<directory>")` function.



RStudio provides a GUI to set the working directory

1. Reading in and cleaning an orthologous SNP matrix in FASTA format

a. Use the function `read_fasta("<fasta_file>")` to read an orthologous SNP matrix into an R variable.

b. 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 in the file Chlamydia_1.fasta and assigned it to a variable called `chlamydia_1`.
- We processed the matrix and:
    - removed two sequences, `A_D213` and `Ba_Aus25`, due to these sequences having a different length from the other sequences.
    - identified two positions, 22 and 24, to be ignored in subsequent analysis due to the presence of non-standard characters (ie characters other than "A", "T", "C", or "G") at those positions in at least one of the sequences.

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 in the file Chlamydia_2.fasta and assigned it to a variable called `chlamydia_2`.
- We processed the matrix and:
    - removed one sequence, `A_D213,` due to it having the same sequence name as an earlier sequence.
    - removed three sequences, `A_D213`, `Ia_SotoGIa3,` and `D_SotoGD1`, due to these sequences having a different length to the other sequences.
    - Identified four positions, 1, 2, 3, and 4, to be ignored in subsequent analysis due to the presence of non-standard characters (i.e., characters other than "A", "T", "C", or "G") at those positions in at least one of the sequences.

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?

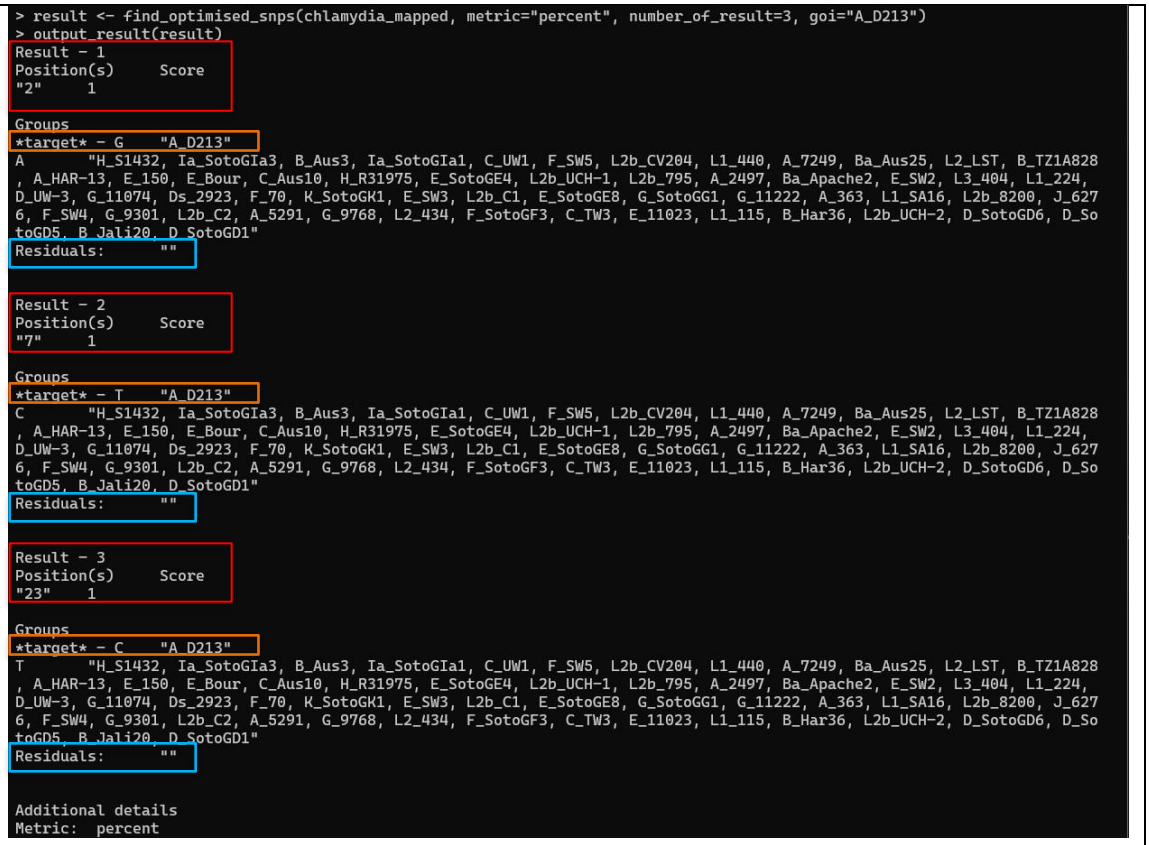
- We read in the file Chlamydia_mapped.fasta and assigned it to a variable called `chlamydia_mapped`.
- We processed the matrix and did not find any anomalies.

2. minSNPs using the percent metric

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)
```



Output

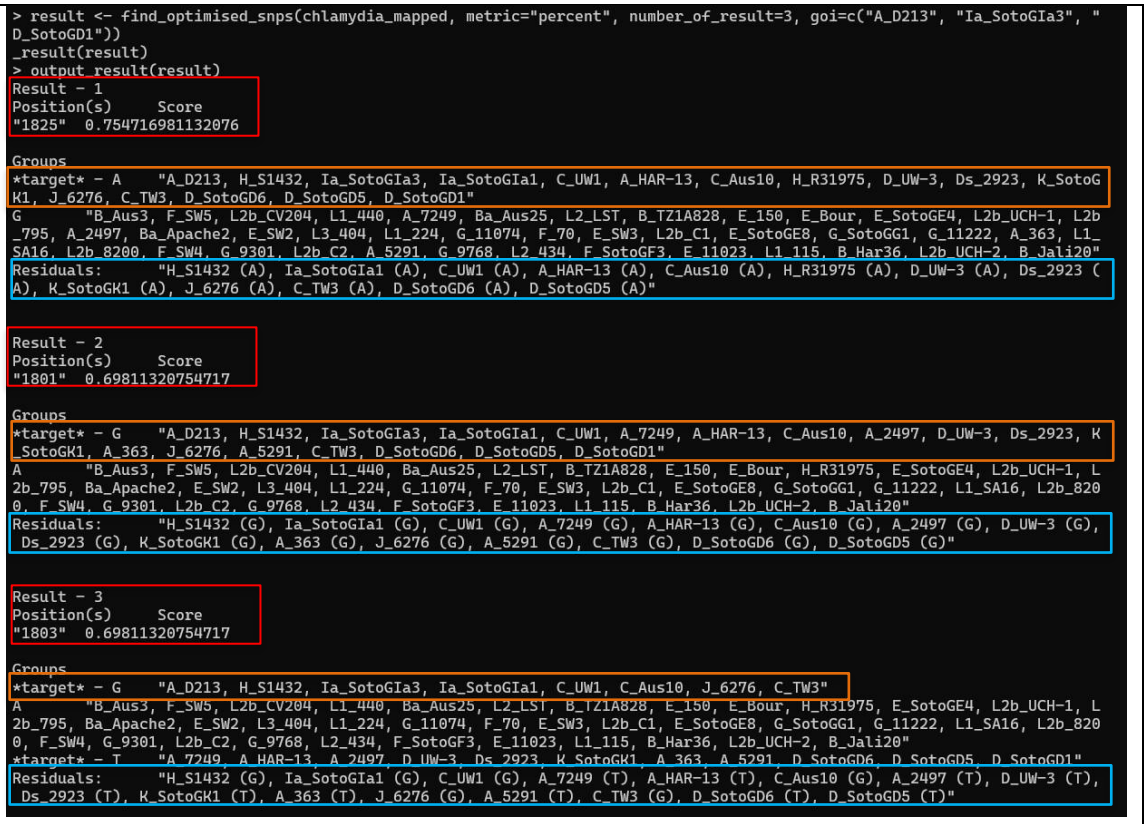
What happened?

- We identified three sets of SNPs that discriminate `A_D213` with 100% sensitivity `(red boxes)`.
- Each of the three SNPs on its own completely discriminate `A_D213`, from all other sequences (i.e., the specificity "Score" is 1.0). The target allele is shown (orange boxes).
- There are no residuals (blue boxes) in any of the SNP sets, i.e., there is no other sequences within the matrix that has the same allelic profile as those sequences in the goi.

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

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)
```



Output

What happened?

- We identified three sets of SNPs that discriminate `A_D213`, `Ia_SotoGIa3`, `D_SotoGD1` (red boxes).
- None of the three sets discriminate the goi from all of the other sequences (ie the specificity "Score" is <1.0). The selected SNPs and how they would group the sequences are shown (target allele in orange boxes, residual in blue boxes). Sequences which possess the same SNP at the identified position as the goi are listed as "residuals".

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

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="percent", number_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_SotoG
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_SotoGE
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_976
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 three sets of SNPs that discriminate `A_D213`, `Ia_SotoGIa3`, `D_SotoGD1`, with each SNP set containing up to 5 SNPs.
- SNP sets 1 and 3 completely discriminate the goi from all the other sequences (ie the specificity "Score" is 1.0) with 5 and 4 SNPs respectively, while SNP set 2 does not completely discriminate the goi from all the other sequences (ie the specificity "Score" is <1.0). The cumulative specificity "Score" is shown as each additional SNP is added. The selected SNPs and how it would group the sequences are shown. The layout of the result is similar to the output above.

3. minSNPs using the simpson metric

Identifying SNPs that maximise the Simpson's index of diversity:

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 three sets of SNPs that maximise the Simpson's index of diversity. Each SNP set can have up to 5 SNPs, only terminating before 5 SNPs are identified if a Simpson's index of diversity of 1.0 has already been achieved (red boxes).
- The selected SNPs and how the SNPs would group the sequences are shown (orange boxes).
- The layout of the result is consistent for all analysis using the simpson metric.

Identifying SNPs that maximise the Simpson's index of diversity while excluding specific SNP positions:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_depth=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"  0.7318181818181818
"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"
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 up to 5 SNPs that has the highest Simpson's index of diversity, but does not include SNP positions 1988 or 8241.

- The selected SNPs and how the SNPs would group the sequences are shown.

Identifying SNPs that maximise the Simpson's index of diversity while forcing the inclusion of specific SNPs:

Code:

```
result <- find_optimised_snps(chlamydia_mapped, metric="simpson", max_depth=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

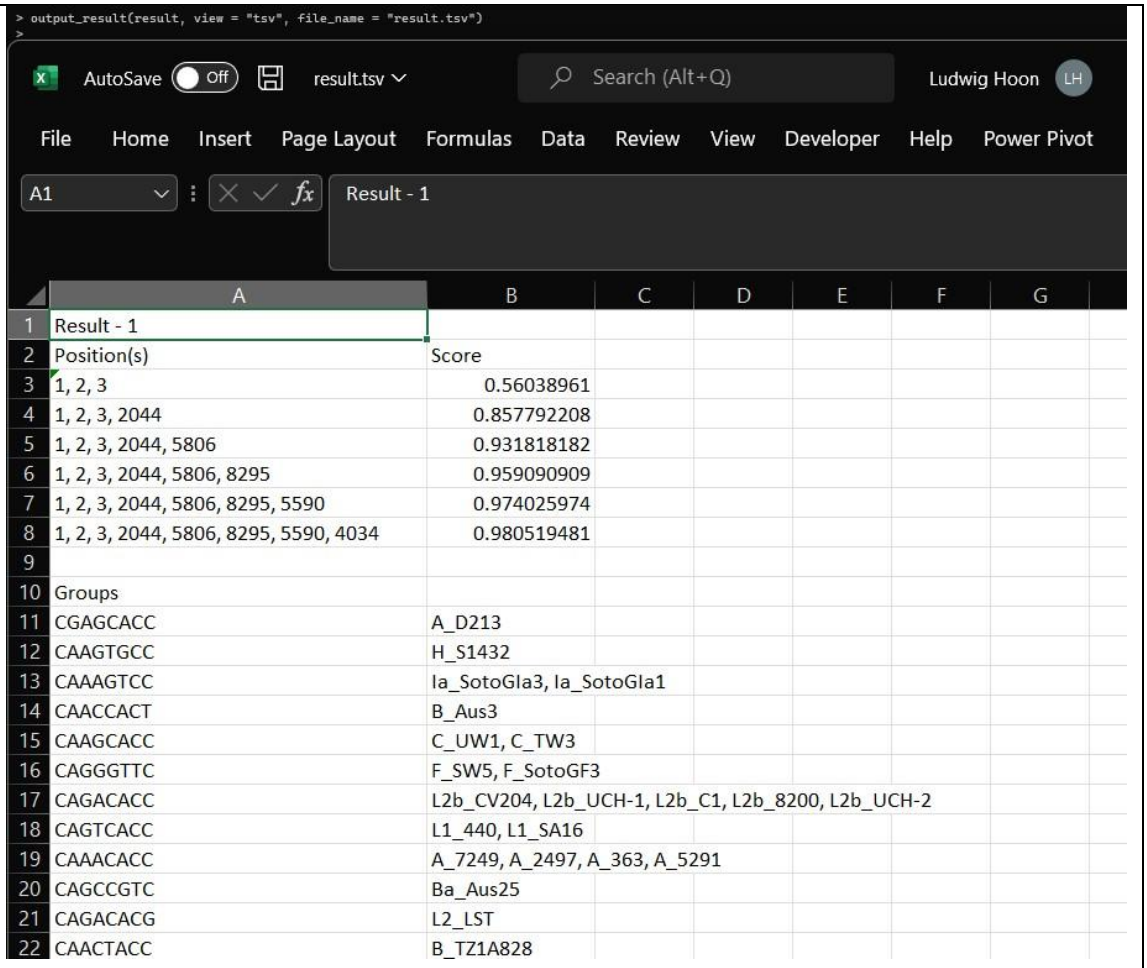
What happened?

- We identified a set of SNPs that gives the highest Simpson's index of diversity, which includes SNP positions 1, 2, and 3, and up to additional 5 SNPs.
- The selected SNPs and how the SNPs would group the sequences are shown.

4. Save result to a TSV file

Code:

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



	A	B
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_SotoGIa3, Ia_SotoGIa1
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 in a file called `result.tsv`, which can be opened in excel, with tab as the delimiter.

5. Parallelising 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
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 using four cores, by adding the bp argument.

Other

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