IMPROVING SNP-BASED MICROBIAL GENOTYPING SOFTWARE

Version 1 Written by: Kian Soon Hoon (s304689) Email: ldwgkshoon@gmail.com

# List of document subpart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Started | 50% completed | Signed off | Note |
| Project Plan | ✓ | ✓ | ✓ |  |
| Test Documentation | ✓ | ✓ | ✓ | All passed |
| User Manual | ✓ | ✓ | ✓ |  |
| Technical Documentation | ✓ | ✓ | ✓ | Automatically generated using devtools in R programming |
| Project Report | ✓ | ✓ | ✓ |  |

# Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| Phil Giffard | phil.giffard@menzies.edu.au | 0489468548 | Client |
| Peter Shaw | peter.shaw@cdu.00edu.au | 0489466157 | Lecturer |
| Kian Soon Hoon | ldwgkshoon@gmail.com | 0487034025 | Student |

Charles Darwin University

Project: Improving SNP-based microbial genotyping software

Table of Contents

[List of document subpart 0](#_Toc494988542)

[Stakeholders 0](#_Toc494988543)

[Project plan 4](#_Toc494988544)

[Synopsis 4](#_Toc494988545)

[Purpose: 4](#_Toc494988546)

[Scope: 4](#_Toc494988547)

[Period: 4](#_Toc494988548)

[Deliverable schedule: 4](#_Toc494988549)

[Acceptance criteria: 4](#_Toc494988550)

[The team and skills 4](#_Toc494988551)

[Milestones 4](#_Toc494988552)

[Methodology/Method 5](#_Toc494988553)

[Risks 5](#_Toc494988554)

[Tools 6](#_Toc494988555)

[Deliverables 6](#_Toc494988556)

[Project Report 7](#_Toc494988557)

[Introduction 7](#_Toc494988558)

[Requirements 8](#_Toc494988559)

[UML design 8](#_Toc494988560)

[Testing 9](#_Toc494988561)

[Unit Testing 10](#_Toc494988562)

[Regression Testing 10](#_Toc494988563)

[Integration Testing 10](#_Toc494988564)

[TDD 10](#_Toc494988565)

[Change Management 10](#_Toc494988566)

[Change request process flow 11](#_Toc494988567)

[Evaluating and authorising change requests 11](#_Toc494988568)

[Configuration Management 11](#_Toc494988569)

[Artefacts 11](#_Toc494988570)

[Backlog 11](#_Toc494988571)

[Story Card 12](#_Toc494988572)

[Continuous Integration 14](#_Toc494988573)

[System Building 14](#_Toc494988574)

[Design Pattern 14](#_Toc494988575)

[Pair Programming 14](#_Toc494988576)

[Agile Process 14](#_Toc494988577)

[Graphics User Interface Design 15](#_Toc494988578)

[Security 15](#_Toc494988579)

[Code 15](#_Toc494988580)

[Documentation 15](#_Toc494988581)

[References 15](#_Toc494988582)

[Appendix 17](#_Toc494988583)

[User Manual 17](#_Toc494988584)

[Original User Manual 18](#_Toc494988585)

[Research Plan 36](#_Toc494988586)

[Technical documentation 40](#_Toc494988587)

[Arguments 41](#_Toc494988588)

[Value 41](#_Toc494988589)

[Description 41](#_Toc494988590)

[flagAllele 41](#_Toc494988591)

[Description 41](#_Toc494988592)

[Description 42](#_Toc494988593)

[Description 42](#_Toc494988594)

[Arguments 42](#_Toc494988595)

[Value 42](#_Toc494988596)

[present.percent 42](#_Toc494988597)

[Arguments 43](#_Toc494988598)

[Value 43](#_Toc494988599)

[Arguments 43](#_Toc494988600)

[Value 43](#_Toc494988601)

[similar.percent 43](#_Toc494988602)

[Value 43](#_Toc494988603)

[Description 44](#_Toc494988604)

[Arguments 44](#_Toc494988605)

[Value 44](#_Toc494988606)

[similar.simpson 44](#_Toc494988607)

[Arguments 44](#_Toc494988608)

[Value 45](#_Toc494988609)

[Value 45](#_Toc494988610)

[usualLength 45](#_Toc494988611)

[Arguments 45](#_Toc494988612)

[Value 45](#_Toc494988613)

[Arguments 46](#_Toc494988614)

[Value 46](#_Toc494988615)

[Examples 46](#_Toc494988616)

[usualLength 47](#_Toc494988617)

[Index 0](#_Toc494988618)

[Test documentation 0](#_Toc494988619)

# Project plan

## Synopsis

### Purpose:

To design and create a software that will help the microbiologists to identify significant SNP and well as the position of those SNPs.

### Scope:

The software will make use of open source library and do the followings:

1. Input DNA sequence alignment
2. Allow user to select % or D mode
3. Identify optimised set of SNPs using either: For D mode, the greedy algorithm (i.e. Simpson’s index of diversity): for % mode: simply the maximum percentage of excluded non-target sequences in the alignment.

### Period:

The development of software for the analysis of genome-wide orthologous SNP is expected to last until 23rd September 2017 (week 10, semester 2).

### Deliverable schedule:

1. 1st release: 28th August
2. 2nd release: 4th September
3. 3rd release: 11th September
4. 4th release: 23th September

### Acceptance criteria:

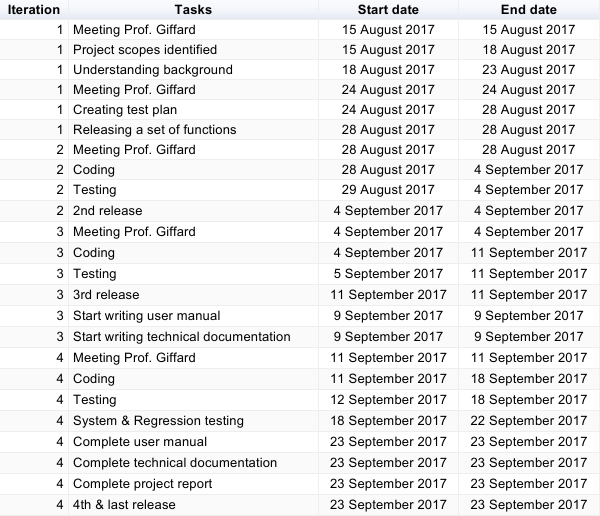
1. Software conforms to the requirements specified and accepted by client.
2. Completing the project before 23 September 2017.
3. Easy to understand documentation that provide enough details for the users.

## The team and skills

Kian Soon Hoon – Experienced in Python, C & C++.

## Milestones

1. Project scopes identified
2. Test plan
3. First release
4. Second release
5. Third release
6. Fourth/final release
7. Complete user manual
8. Complete technical documentation
9. Complete project report



\*\*\*

## Methodology/Method

The project will utilise the agile methodology (Kanban). 4 iterations are incorporated into the kanban practice. At each iteration, there are few things that have to be done, which include: (1) Meeting with client, (2) Extracting features that can be integrated in the iteration, (3) Coding the selected features using Test Driven Development, and (4) releasing the set of features.

Kanban board is used to ease the communication between team members. Asana is chosen as the online platform which consist of a Kanban board. Besides, pair programming is also used to ensure that all team members have understand the code and can modify it in future.

## Risks

A list of risks as well as the mitigation strategies has been listed in the table below.

|  |  |
| --- | --- |
| Risks | Mitigation strategies |
| Lack of knowledge in the area of microbiology | 1. Background research (Internet, journal articles). 2. Consult with Phil Giffard. 3. Consult with microbiology master/PhD student under Phil’s supervision (in the event that Phil is unavailable). |
| Lack of experience programming in R | 1. Pair programming. 2. Doing MOOC online (coursera/edx). 3. Using online tutorial resources (tutorialspoint, etc.). 4. Utilising online discussion forum (stackoverflow, etc.). 5. Youtube. |
| Technical aspects of the algorithm | 1. Utilising open source library as much as possible. 2. Consult with Peter/Phil depending on the situation. 3. Ask question in online forum. |
| Weekly physical meeting | 1. Online collaboration tools (google docs, whatsapp, etc.). 2. Whenever possible, decide on a day when everyone is free. |

## Tools

-Git repository (Accessible at <https://github.com/ludwigHoon/PRT452-BINFO->)

-Git Desktop

-R programming environment

-Asana

-Whatsapp for communication

-Email to schedule meeting & share files with client.

## Deliverables

1. Functional prototype of the improved software for SNP-based microbial typing in R programming language
2. User documentations on how to use the software
3. Test documentation
4. Project report

# Project Report

## Introduction

In clinical, public health, and research microbiology, there is frequently a need to divide bacterial species into “types”. A bacterial type may be thought of as a classification unit at a finer scale than “species”, and roughly synonymous with the term “strain”. Bacterial typing is used to trace instances and patterns of transmission and dissemination, and may be used at all scales from global to within individual buildings (e.g. hospitals), to indicate virulence or resistance properties of isolates, and in basic research to assist understanding of evolutionary history. While early typing methods were based upon reactions with standard sets of antisera (serotyping), or susceptibility to standard sets of viruses (phage typing), for decades, the great majority of bacterial typing has been performed using genetic methods. As a result, the term “genotyping” will be used in this document. Over the years, considerable ingenuity has been applied to the development of bacterial genotyping methods. Many methods have been described, with most being variations on the theme of electrophoresis of complex mixes of DNA fragments so as to yield banding pattern “fingerprints”, or sequencing of standardised gene fragment(s).

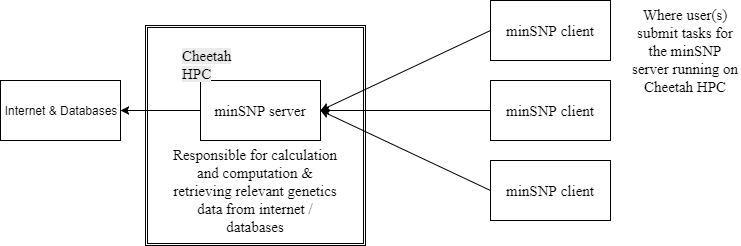
In the early 2000s, Phil Giffard led a project within the QUT node of the Cooperative Research Centre for Diagnostics. This was essentially a re-think of bacterial genotyping methodology. The innovative concept was that the volume of known comparative gene sequence data from within bacterial species was exploding. This constitutes a suburb resource to mine for sets of variable genetic locations (single nucleotide polymorphisms (SNPs)) that are optimised for use in bacterial genotyping methods with predefined performance specifications. In this way, genotyping methods that yield the needed resolving power or other information as efficiently as possible could be easily designed. An added impetus for this area of research was rapid development in technology for interrogating SNPs. The first publication outlining this approach was published in 2004. This reported the construction of the SNP-mining “Minimum SNPs” software package, and the demonstration of two bacterial genotyping methods, using allele specific PCR on a real time PCR platform.

A major development in very recent years has been the advent of low cost whole genome sequencing of bacteria. The work described above made extensive use of multilocus sequence typing (MLST) databases as the input data for SNP mining. MLST was developed in the UK (e.g. see http://saureus.mlst.net/), as a major and very successful initiative to standardize bacterial genotyping methods and terminology. For each bacterial species for which it has been implemented, seven standardised gene fragments are defined. These are sequenced to generate the “sequence type” (ST). The MLST database curators maintain comprehensive internet accessible information regarding sequencing variants at the loci (alleles) and the alleles found together in bacterial isolates (STs). Often several thousand STs are known. Our SNP based typing methods have been based on mining alignments of these STs. The MLST loci only cover ~0.1% of the bacterial genome. Lost cost whole genome sequence makes it possible to genotype a bacteria by whole genome sequencing and causes MLST obsolete. However, despite the decreasing cost and time needed, sequencing an entire genome still requires a substantive amount of money and time. On the other hand, the increasing data whole genome sequences available openly online provide the opportunity of identifying useful Single Nucleotide Polymorphism (SNP). Accordingly Giffard, work is shifting to SNP mining from alignments of entire genomes rather than alignments of small fragments of genomes such as MLST databases. This means that ~1000 more SNPs are available for selecting optimal combinations, resulting in much higher performance of the genotyping methods and hence called for the improvement of the existing software.

This project will scrap the old system and reverse engineer the original system in R programming environment. The project will also implement some extra features as described by Philip Giffard. An example of the extra features is the implementation of error checking for the FASTA file accepted by the program prior computation of the minimum SNP, this will ensure that the program doesn’t crash in case it has receive files with allelic profile(s) that contain deletion(s). Besides, it will also create a table to present the SNPs and the allelic profiles that they detect.

## Requirements

The original user manual describes the full functionalities of the expected program. The original user manual can be found in appendix 2. The proposed changes to the software is expected to integrate to Cheetah HPC and the requirements are included in a research plan which can be found in appendix 3. The figure below summarise the implementation of the final software.

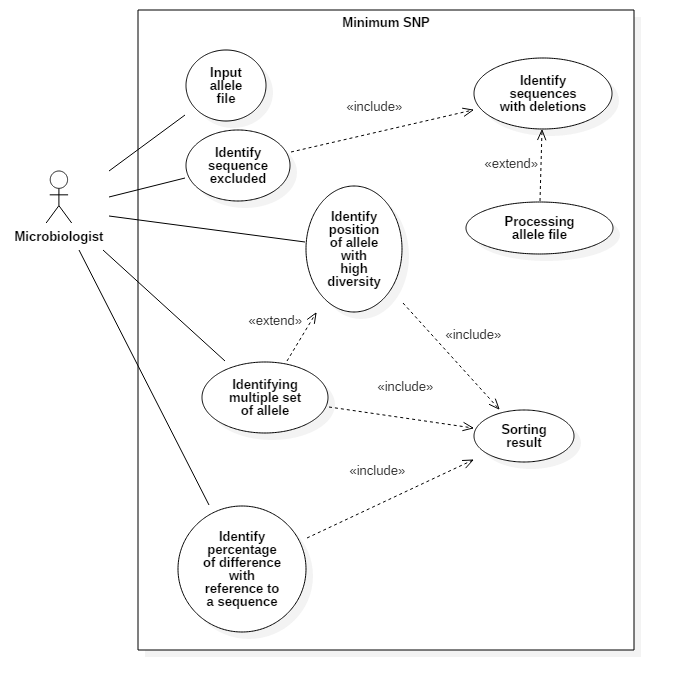


Due to time and resource constraints, the scope of this project has been limited to reverse engineering some functionalities from the Minimum SNPs software as described in Price E.P., Inman-Bamber, J., Thiruvenkataswamy, V., Huygens, F and Giffard, P.M. 2007, Computer-aided identification of polymorphism sets diagnostic for groups of bacterial and viral genetic variants. BMC Bioinformatics 8:278. The original functionalities can be found in the Minimum SNPs software user manual which is attached together in the manual folders. According to Phillip Giffard, the followings are the requirements of the project:

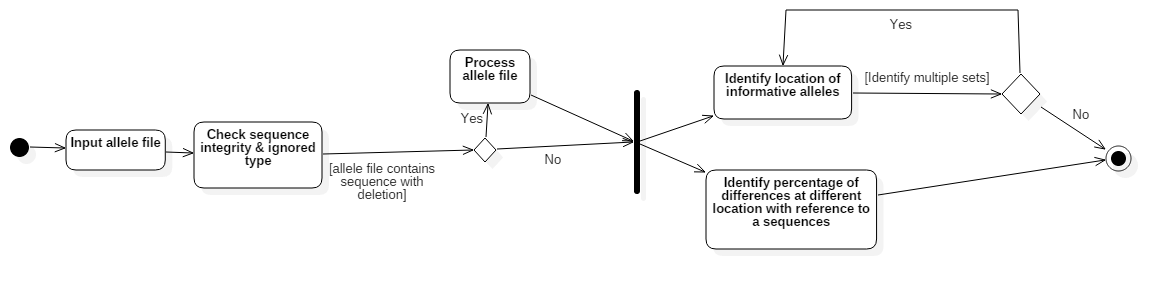
1. Reverse engineer the ‘Minimum SNPs’ software into R programming environment.
2. Improve the error checking of the new ‘Minimum SNPs’.
3. Integrate additional features into the new ‘Minimum SNPs’.

## UML design

The use case diagram below outlines how the users will interact with the new ‘Minimum SNPS’.



The activity diagram below outlines how the user will use the new ‘Minimum SNPs’ to get result(s).



## Testing

Testing is very much the essence of this project. Testing is used to ensure that the program will runs smoothly without error. Using Test Driven Development (TDD), it is ensured that the testing covers 100% of the code. The program is tested in the development platform before being committed to the GitHub repository. The testings done in the development and before releases are unit testing, regression testing and integration testing.

\*N.B. Refer to Test documentation in the appendix for list of test cases and result

### Unit Testing

Each of the functions in the package is tested. The unit tests can be found under minSNP/tests directory. The testing library used is RUnit, which is similar to Junit. The documentation for the library can be found in <https://cran.rstudio.com/web/packages/RUnit/index.html>. The result for the unit tests are obtained from the old software, whenever possible (i.e. when reverse engineering an old functionality). A 0.00016 tolerance is set for Simpson’s index as the calculation in R Programming environment is more accurate and the output from the old program can provide accuracy of only up 3 decimal points.

### Regression Testing

Every time a new function is added to the package, the whole system is run again in order to ensure that the new functionality does not affect the performance of other functions. Every time a function is modified during the development process also trigger a regression test. This is to ensure that the functions that depend on the modified functions are not affected.

### Integration Testing

After building the package, integration test is ran. This is to ensure that all the functions can work together. The integration tests can also be used for documentation and shows the users how all the functions are used. There are 3 sample cases found in Integration test. The file can be found in the minSNP/test directory. All the 3 cases make use of all the functions and demonstrate how a user might normally deal with the different functions. The only difference in the 3 cases is the file being examined. The first file contains all the sequence type with normal length; the second file contains a sequence type with deletion(s) while the third file contains multiple sequence types with deletion(s). However, regardless of the file and whether the files contain sequence type with deletion or not, the normal workflow is the same and the user should always check the file before computation.

### TDD

Unit-testings have been designed prior to the coding. The results expected are from the descriptions of the clients and the ‘Minimum SNP’ software written in Java. The software is run using the dataset and the result is used in unit testing to ensure that the calculation is correct. In situation where the result is unattainable (i.e. when it’s a new features or modification requests to the new software), description of the client is used and expected result are calculated or generated manually.

The tests can be found in the test folder and can be found on github. Moreover, the test documentation can also be found in the appendix.

## Change Management

The repository for the project can be found in <https://github.com/ludwigHoon/PRT452-BINFO->. Any issues found can be reported by the users to at the address. Users can also requests changes to the package by sending issue.

### Change request process flow



The change request can be generated by different users at the github page or by emailing Prof. Phillip Giffard. After that the change request are evaluated (method of evaluation in the following section). Authorised change request will be implemented and a subsequent update will be made available in the repository.

### Evaluating and authorising change requests

In order to evaluate and prioritize a change request, the “priority” of the change is taken into consideration.

|  |  |
| --- | --- |
| **Priority** | **Description** |
| High | *Bugs that cause the system to output wrong result; Functionality that can significantly improve the program.* |
| Medium | *Bugs that affect user experience but are functionally correct; functionality that can improve performance of the program.* |
| Low | *Minor bug & new functionalities that can improve user experience.* |

After changes and release of new version, different old version are still stored in the repository and are made available to the public.

## Configuration Management

The software is expected to be released under open source licence. MIT Licence is chosen because it allows the others to do whatever they want with the software and also absolve development team any responsibility in the event something unexpected happens. When the software is released, the source code will be made available to the general public. When the software is passed over to the maintenance team, the maintenance team can expect the support from this document and documentations. Moreover, the maintenance team should also be able to reuse the tests provided.

When the configuration is done, testing can be used to ensured that the new codes are fine by running the test, by using the command source(‘runTest.R’) after setting the directory to tests. The command will automatically run the unit tests, regression test and integration tests.

## Artefacts

Different agile artefacts have been generated, for example the backlog, some UML diagrams, documentation, etc. These artefacts can be found in the report.

### Backlog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iteration | Date | Title | Descriptions | Priority | Status |
| 1 | 28August 2017 | Detect abnormal | Detecting the allelic profiles which are ignored during computation of the minimum SNPs and informing the users about the ignored profiles. | Low | Completed |
|  | 28 August 2017 | Percentage calculation | Implementing percentage of differences at different loci with reference to a particular allelic profile | Medium | Completed |
|  | 31 August 2017 | Simpson’s index | Implementing the Simpson’s index of diversity. | Medium | Completed |
| 2 | 5 August 2017 | Table for Simpson’s | Focus on creating table for Simpson’s index and corresponding allelic profile | High | Completed |
|  | 5 August 2017 | Simpson’s search | Looking for multiple SNP that will define a particular allelic profile from the rest. Implementing multi-levels Simpson’s index. | Medium | Completed |
| 3 | 13 September 2017 | Simpson’s inclusion & exclusion | Including and/or excluding certain SNPs while calculating Simpson’s index. | High | Completed |
|  | 13 September 2017 | Simpson’s search (Multiple pairs) | Giving the option to generate multiple pairs of SNPs. This can be done by running the calculation again with the exclusion of the 1st SNP found. The rationale is that the 1st SNP with highest Simpson’s index may not return the highest index when combined with the other SNPs. | High | Completed |
| 4 | 25 September 2017 | GUI for me | Creating the GUI for the program. | Medium | On-going |

### Story Card

Story cards are used to help with the generation of requirements. It shows how the users will normally use the system and help with the coding process.

|  |  |
| --- | --- |
| Story name (ID) | Detect abnormal (1) |
| As a | Microbiology researcher |
| I want to | Input FASTA file for analysis and be informed if there are allelic profiles that contains deletion and are ignored from computation |
| So that | I know what are the allelic profiles that have been excluded and make my work easier |
| Importance | Low |
| Estimated time | 2 days |

|  |  |
| --- | --- |
| Story name (ID) | Percentage Calculation (2) |
| As a | Microbiology researcher |
| I want to | Calculate the percentage of differences at different location for an allelic profile |
| So that | I can find the location(s) of the SNPs with the highest percentage of difference |
| Importance | Medium |
| Estimated time | 2 days |

|  |  |
| --- | --- |
| Story name (ID) | Simpson’s Index (3) |
| As a | Microbiology researcher |
| I want to | Calculate the Simpson’s index of the SNP at a location |
| So that | I know how useful the SNP at that location is |
| Importance | Medium |
| Estimated time | 2 days |

|  |  |
| --- | --- |
| Story name (ID) | Table for Simpson’s index (4) |
| As a | Microbiology researcher |
| I want to | To know see the sequence type in a table |
| So that | I can know how the sequence types are categorised given any number of SNPs |
| Importance | High |
| Estimated time | 3 days |

|  |  |
| --- | --- |
| Story name (ID) | Simpson’s search (5) |
| As a | Microbiology researcher |
| I want to | Find the SNP that returns the highest Simpson’s index at each level |
| So that | I can find the most useful SNPs to categorise the sequence types |
| Importance | High |
| Estimated time | 4 days |

|  |  |
| --- | --- |
| Story name (ID) | Simpson’s search inclusion & exclusion (6) |
| As a | Microbiology researcher |
| I want to | Include and exclude certain SNPs in the Simpson’s search |
| So that | I am able to see how the SNP performs under certain situation |
| Importance | High |
| Estimated time | 3 days |

|  |  |
| --- | --- |
| Story name (ID) | Simpson’s multiple search (7) |
| As a | Microbiology researcher |
| I want to | Have multiple SNP returned at different levels |
| So that | I am convinced that the SNPs I found is really better than the rest, as different SNP perform differently when combined with another SNP. |
| Importance | High |
| Estimated time | 5 days |

|  |  |
| --- | --- |
| Story name (ID) | GUI for me (8) |
| As a | Microbiology researcher |
| I want to | Use the graphical user interface instead of using the function and typing all the things in |
| So that | It’s easier for me to use the software |
| Importance | Medium |
| Estimated time | 7 days |

## Continuous Integration

Continuous integration is a necessary part of this project. Every time a new functios become available or when a bug is discovered and debugged, the codes are checked into the github. The different functions are integrated and the package is built every week on Saturday. Besides that, Travis-CI is also used as the continuous integration tool which will automatically build and test the R package.

## System Building

The output of the project is a package in R programming environment. The package is built using devtools provided together in R programming environment. The command build() is used to generate a tar file. The tar file generated can then be checked using check() to ensure that all necessary files are in the tar.gz and that the format of the package conform to the standard. The package is then ready to be installed in another machine.

## Design Pattern

Wrapper pattern is used in the coding. According to Warnholz 2015, Wrapper function is used to add functionality to a function. This is used to simplify works by utilising existing functions to develop a more complicated function.

Cache pattern is another design pattern used. According to Warnholz 2015, Cache pattern is used when the result from a previous computation is cached for performances reason. This pattern is used to control branching in the calculation of multi-level Simpson’s index and is important as the calculation can be expensive.

## Pair Programming

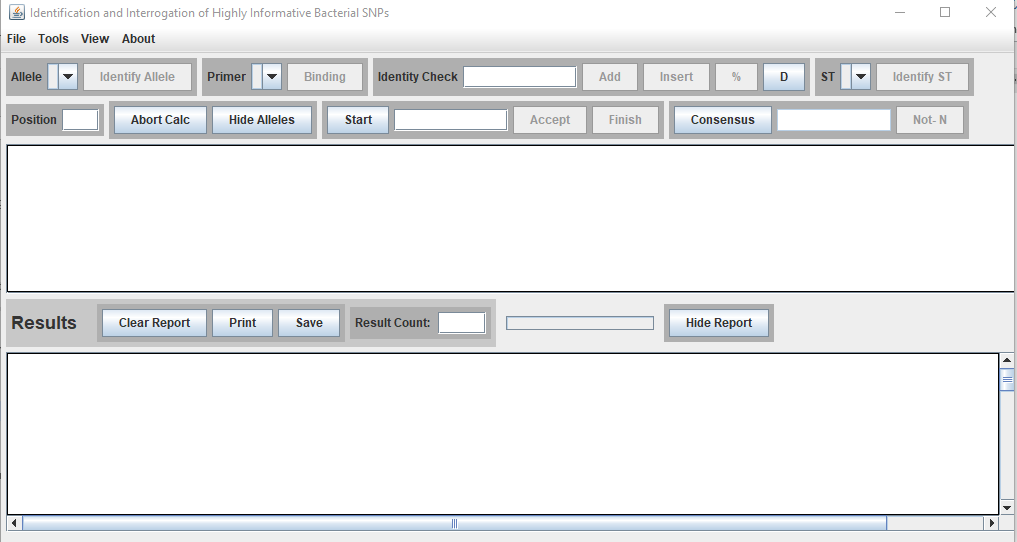
Pair programming was originally planned to be used in order to reduce errors in code and to make it easier when different people needs to modify the code. However, it is not used as a technique in the end because the number of member in the group has been reduced to one.

## Agile Process

As part of the agile process, there have been frequent releases. Moreover, the client was met at least once a week and client also gets to decide what features are prioritised first. Inputs from the client is put as the top priority. Moreover, different team members also work on different part of the project together and different tools that support the agile process are used.

## Graphics User Interface Design

The new software is expected to look similar to the original minSNP software. The figure below shows the original graphics user interface.



## Security

Security is not a major part of the project as it is designed to be used by individuals and is not connected to the Internet or contains any sensitive information. However, it is important that the data imported by the users maintains its integrity throughout the usage of the program. This is because the integrity of the data will affect the result from the computation. Hence system testing and integration testing are focused on checking that the data passed between functions are in their right format and are integrated.

## Code

The code is available in the GitHub repository. Please refer to [https://github.com/ludwigHoon/PRT452-BINFO-](https://github.com/ludwigHoon/PRT452-BINFO-%20) for the codes. Some snippets are presented here for some discussion.

## Documentation

The user documentation can be found attached together with the software.

The technical documentation are generated with roxygen2. It helps the users with the usage of the software. The technical documentation can be found in the manual folders in the software package. Alternatively, it can be accessed in the R programming environment by typing help(<function>). Please refer to the github repository for the documentations.

# References

Warnholz 2015, *Design Patterns in R*, https://wahani.github.io/2015/12/Design-Patterns-In-R/.

# Appendix

## User Manual

User Manual

Minimum SNPs

Table of Contents

[Getting started 1](file:///C:\Users\acer\Documents\GitHub\PRT452-BINFO-\minSNP\man\documents\User%20Manual.docx#_Toc494444921)

[Importing file 1](file:///C:\Users\acer\Documents\GitHub\PRT452-BINFO-\minSNP\man\documents\User%20Manual.docx#_Toc494444922)

[Processing the allelic profiles and identify profiles to be excluded 1](file:///C:\Users\acer\Documents\GitHub\PRT452-BINFO-\minSNP\man\documents\User%20Manual.docx#_Toc494444923)

[Using % mode 2](file:///C:\Users\acer\Documents\GitHub\PRT452-BINFO-\minSNP\man\documents\User%20Manual.docx#_Toc494444924)

[Using D mode 2](file:///C:\Users\acer\Documents\GitHub\PRT452-BINFO-\minSNP\man\documents\User%20Manual.docx#_Toc494444925)

**Getting started**

To get started, first download the package from <https://github.com/ludwigHoon/PRT452-BINFO->. After that, import the library in the R-programming environment by the command: library(minSNP).

**Importing file**

The first thing that a user probably wants to do first is to import the fasta file which contains a list of allelic profiles into the programming environment. This can be achieved in the following steps:

1. Change directory to the location of the fasta file in the R-programming environment (option under files).
2. Import SeqinR library, by the command: library(seqinr).
3. Read the fasta file, by using the command: VariableA<-read.fasta(‘filename’).

**Processing the allelic profiles and identify profiles to be excluded**

Before doing any other operation, it is prudent for the user to first process the allelic profiles, in order to make sure that the file is correct and that the allelic profiles don’t contain any allelic with deletion as the algorithms in the package does not take into account of the SNPs with deletion and may return incorrect result or fail to execute. In order to process the allelic profiles, the following steps can be taken:

1. After reading the allelic profiles file and setting that in a VariableA, flagAllele function can be used to return a list of allelic profiles that is shorter than the others. I.e. with command: flagAllele(VariableA), the flagged allelic profiles will be shown if there is any.
2. The function processAllele can then be used to processed the allelic profiles and ignore those allelic profiles which is shorter. I.e. with the command: VariableA <- processAllele(VariableA). The VaraibleA contains only those allelic profiles with normal length.

**Using % mode**

In order to calculate the percentage of difference of a SNP of an allelic profile as compared to the others, similar.percent function can be used. The command steps are as followed:

1. Read the fasta file that contain the list of allelic profiles to be analysed (i.e. the allelic profiles are now in VariableA) into the R-programming environment.
2. Use the function similar.percent to calculate the percentage and assign the result to a variable. I.e. with the command: Result\_Variable <- similar.percent(VariableA, ‘Targeted\_Allelic\_Profile’).
3. Pass the result variable to present.percent to process the result so that it shows only (1) predefined number of results, and (2) results with the minimum percentage of difference. I.e. with the command: present.percent(Result\_Variable, <minimum percent>, <number of result>).

**Using D mode**

In order to find the pair of SNPs that can be used to type the different sequence type, along with the Simpson’s index, similar.simpson function can be used. The command steps are as followed:

1. Read the fasta file that contain the list of allelic profiles to be analysed (i.e. the allelic profiles are now in VariableA) into the R-programming environment.
2. Use the function similar.simpson to find the top SNPs pair and assign the result to a variable. I.e. with the command: Result<- similar.simpson (VariableA, <level>, NULL, <included>, <excluded>), where level is number of SNPs returned (e.g. 2, returns 2 SNPs, 3 return 3); included is the position(s) of SNP that is forced into consideration; excluded is the position(s) of SNP that is forced out of consideration.

\*Note: If there are more than 1 positions to be included/excluded, <included>/<excluded> can be replaced with a vector of numeric, i.e. c(position1, position2,…..).

1. The result can be viewed by typing the result variable into the environment. Alternatively, what the users might want to see is the sequence types that are defined by the specific SNPs selected. In this case, the user can pass the result variable into present.simpson. I.e. with the command: present.simpson(VariableA, Result).

## Original User Manual

**Minimum SNPs**

**Version 2043**

**User Manual**

For assistance or additional information, contact Phil Giffard at p.giffard@qut.edu.au

Largely written by John Bamber, November 2006.

Last updated by Phil Giffard, September 2007

**Contents**

1. Definitions. Definitions of terminology, background information, references. P3

1. Getting Started. System and software requirements, how to start the program P6

1. Creating a mega-alignment. How to down load and concatenate MLST data. P7

1. Obtaining and loading a pre-concatenated MLST database. P10

1. Deriving highly informative sets of SNPs. P11

1. Working backwards method. How to go from SNP profile to sequences. P16

1. Tools and options. These functions facilitate efficient and effect searches. P29
2. **Definitions:**

***1.1 Minimum SNPs***

**Minimum SNPs is proprietary software for deriving discriminatory sets of SNPs from DNA sequence alignments, as described in**:

Price E.P., Inman-Bamber, J., Thiruvenkataswamy, V., Huygens, F and Giffard, P.M. 2007. Computer-aided identification of polymorphism sets diagnostic for groups of bacterial and viral genetic variants. BMC Bioinformatics 8:278.

**Earlier versions of the software are described in:**

Robertson, G.A., Thiruvenkatswamy, V, Shilling, H., Price E.P., Huygens, F., Henkens, F.A. and Giffard, P.M. 2004. indentifcation and interrogation of highly informative single nucleotide polymorphism sets defined by bacterial multilocus sequence typing databases. J. Med. Microbiol. **53:**35-45.

Price, E.P., Thirvenkatswamy, V., Mickan, L., Unicomb, L., Rios, R.E., Huygens, F., and Giffard, P.M. 2006. Genotyping of *Campylobacter jejuni* using seven single-nucleotide polymorphisms in combination with *flaA* short variable region sequencing. J. med. Microbiol. **55:**1061-1070.

**Additional applications of the software are described in:**

Stephens, A.J., Huygens, F., Inman-Bamber, J., Price, E.P., Nimmo, G.R., Schooneveldt, J, Minckhof, W and Giffard, P.M. 2006. Methicillin-resistant *Staphylococcus aureus* genotyping using a small set of polymorphisms. J. Med. Microbiol. **55:**43-51.

Price, E.P., Huygens, F and Giffard, P.M. 2006. Fingerprinting of *Campylobacter jejuni* using resolution-optimized binary gene targets derived from comparative genome hybridization studies. Appl. Env. Microbiol. **72:** 7793-7803.

Huygens, F., Inman-Bamber, J., Nimmo, G.R., Munckhof, W., Schooneveldt, J., Harrison, B., McMahon, J.A and Giffard, P.M. 2006. *Staphylococcus aureus* genotyping using novel real-time PCR formats. J. Clin. Microbiol. **44:**3712-3719

Stephens A.J., Huygens F., and Giffard, P.M. 2007. Systematic derivation of marker sets for Staphylococcal Cassette Chromosome *mec* typing. Antimicrob. Agents Chemother. **51:**2954-2964.

The software was developed by researchers at the Queensland University of Technology

(Brisbane, Queensland, Australia) node of the Cooperative Research Centre for Diagnostics, with input from researchers at the University of Newcastle, Newcastle, New South Wales, Australia.

Minimum SNPs can only be obtained from www.ihbi.qut.edu.au/research/cells\_tissue/phil\_giffard/

***1.2 SNP***

A single nucleotide polymorphism is a variation in the genetic code at a specific point on the DNA. In principle, SNPs could be bi-, tri-, or tetra-allelic polymorphisms. However, tri- allelic and tetra- allelic SNPs are rare and so SNPs are sometimes simply referred to as bi- allelic markers. However, we have identified two tri-allelic SNP that have proved very useful in genotyping *Staphylococcus aureus*.

***1.3 MLST***

Multi Locus Sequence Typing:

*“The original MLST web software was developed by Man-Suen Chan (Oxford University) and this version has been developed by David Aanensen (Imperial College) who is funded by The Wellcome Trust*

*Multilocus sequence typing (****MLST****) is an unambiguous procedure for characterising isolates of bacterial species using the sequences of internal fragments of seven housekeeping genes. Approx. 450-500 bp internal fragments of each gene are used, as these can be accurately sequenced on both strands using an automated DNA sequencer. For each house-keeping gene, the different sequences present within a bacterial species are assigned as distinct* ***alleles*** *and, for each isolate, the alleles at each of the seven loci define the* ***allelic profile*** *or* ***sequence type*** *(****ST****).*

*Each isolate of a species is therefore unambiguously characterised by a series of seven integers which correspond to the alleles at the seven house-keeping loci.*

*In MLST the number of nucleotide differences between alleles is ignored and sequences are given different allele numbers whether they differ at a single nucleotide site or at many sites. The rationale is that a single genetic event resulting in a new allele can occur by a point mutation (altering only a single nucleotide site), or by a recombinational replacement (that will often change multiple sites) - weighting according to the number of nucleotide differences between alleles would imply that the latter allele was more distantly-related to the original allele than the former, which would be true if all nucleotide changes occurred by mutation, but not if the changes occurred by a recombinational replacement.*

*Most bacterial species have sufficient variation within house-keeping genes to provide many alleles per locus, allowing billions of distinct allelic profiles to be distinguished using seven house-keeping loci. For example, an average of 30 alleles per locus allows about 20 billion genotypes to be resolved.*

*MLST is based on the well established principles of multilocus enzyme electrophoresis, but differs in that it assigns alleles at multiple house-keeping loci directly by DNA sequencing, rather than indirectly via the electrophoretic mobility of their gene products.”* (Extracted from<http://www.mlst.net/misc/further.asp>on November 1, 2006)

***1.4 Locus/loci:***

The loci are internal fragments of seven house-keeping genes (450-500bp) common to all isolates within a species. More broadly speaking, a locus is position on a chromosome of a gene or other chromosome marker; also, the DNA at that position.

***1.5 Allele***

For each housekeeping gene, alleles are the different sequences present within a bacterial species. More broadly speaking, an allele is any one of a number of alternative forms of the same gene occupying a given locus (position) on a chromosome.

***1.6 Allelic profile:***

The alleles at each of the seven loci (housekeeping genes) in a particular isolate. More broadly speaking, the allelic profile is the combination of alleles that any one individual possesses.

***1.7 ST:***

Sequence Type:

Sequence type is a number given to represent a unique allelic profile  analogous to allelic profile.

***1.8 Method***

There are four basic applications, or methods, that we have developed for use with Minimum SNPs. Below is a brief description of each.

**1.8.1 Percent Method:**

What set of SNPs should be tested to differentiate a single known sequence type from all other sequence types available on the MLST database?

**1.8.2 “D” method:**

What set of SNPs should be tested to differentiate an unknown sequence type from any other sequence type in the MLST database?

**1.8.3 Not-N method:**

What set of SNPs should be tested to identify a defined group or complex of sequence types such that a false negative result cannot occur?

**1.8.4 Working backwards method:**

Which sequence types available on the MLST database share a defined set of SNP alleles?

**2 Getting Started**

Minimum SNPs uses the Java Runtime Environment so you will need this software installed on your computer before you begin. You can download the latest version of Java from<http://www.java.com/en/download/windows_ie.jsp>. If you experience problems using the latest version of Java you may wish to try using an older version. Java 2 Runtime Environment, SE v1.4.2\_01 was used for the creation of this manual.

There are many versions of Minimum SNPs that were released at various staged in it’s evolution. Be ware that some older versions have serious bugs that will make some of the applications described in this manual erroneous. The version of Minimum SNPs used at the time of writing this manual was MLST2043\_\_\_Not\_N.

Once you have unzipped the parent file (MLST2043\_\_\_Not\_N) you will notice that a new folder has been created called “classes”. There are many files within the “classes” folder but there is only one executable Jar file called “MLST”. When you open the “MLST” file make sure that you open with Java (“javaw”). The following icon should appear if the file is ready to be opened with Java.



Mlst.jar

**Important note**

At the time the early versions of Minimum SNPs were written, it was not possible to download concatenated databases from MLST web sites. As a consequence the software contains its own concatenation facility. (In this manual, a concatenated MLST database is termed a “mega-alignment”). Although the concatenation function in Minimum SNPs is completely functional, it is somewhat inconvenient and counter-intuitive to use, with the major issue being that the concatenated data cannot be stored and must be re-assembled from the allele sequence and ST profile data every time the software is used. (Although this only takes a few minutes, it is quite annoying). In consequence, we recommend that analyses be carried out on pre-concatenated databases where possible. This allows analyses to be carried out using much fewer key-strokes and in an inherently simpler fashion. The only disadvantage is that while the “on board” concatenation function keeps track of where each locus starts and stops, and defines SNPs in terms of location within MLST loci, this obviously cannot be done with pre-concatenated data. In this case, the output SNPs are defined only of terms of their location within the concatenated data.

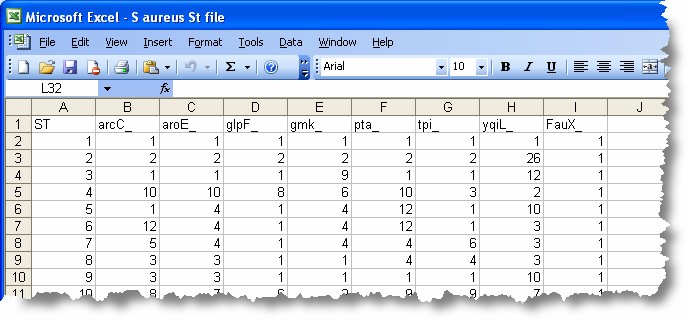
This manual is currently structured around using the on-board concatenation function. However, sections that detail how to obtain and use pre-concatenated MLST data have been added throughout.

1. **Creating a Mega-Alignment:**

**Note: if you intend to use pre-concatenated MLST data, you do not need to create a mega-alignment. Go to Section 4.**

**Minimum SNPs concatenates MLST data by processing the individual allele sequence alignments, with reference to a file of MLST allele profiles. This section describes how to download these data, ensure that they are in the correct format, and assemble a mega-alignment i.e. a concatenated MLST database.**

* 1. ***Obtaining and preparing a file of MLST allele profiles :***



Sequence types and allelic profiles can be downloaded from the Multi Locus Sequence Typing website ([http://www.mlst.net/)](http://www.mlst.net/). Follow the link to “Databases” and select the database for the organism of interest. Depending on the host website, allelic profiles may be downloaded in tab-delimited form, comma-delimited form or as a csv file. The sequence type and allelic profile list must then be converted to a csv file with column headers “ST” for column **A** and the loci names for columns **B** through **H.** The software has a bug which results in the last locus (column **H**) not being included in the mega-alignment. To circumvent this problem, simply insert a fake locus into column **I**. Column **I** then becomes the last column and is not included in the analysis.

**NB:** The names used to identify the loci must be exactly the same as those used in the allele files.

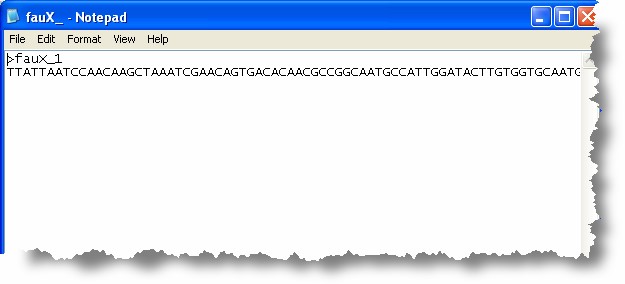
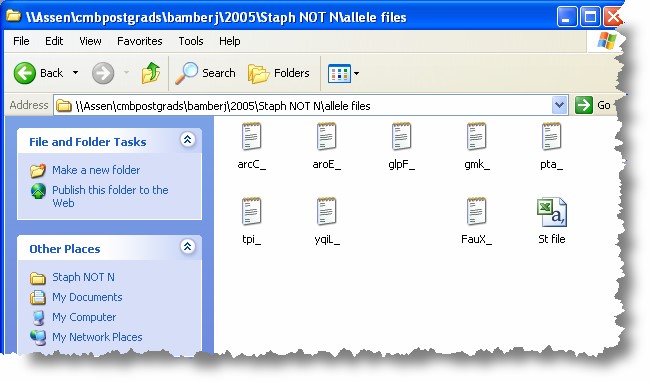
* 1. ***Obtaining and preparing allele sequence alignment files:***

Allele sequences are also downloaded from the MLST web site (http://www.mlst.net). Once again, follow the link to “Databases” and select the database for the organism of interest. Allele sequences can be downloaded in FASTA format form all host websites. Some downloaded allele sequences will have paragraph endings (¶) at regular intervals thought each sequence while others will have no paragraph endings within the sequence.

Either version of the download can be used with Minimum SNPs.

**NB:** The names used for the loci in FASTA format allele sequences must be exactly the same as those used in the ST file above.

Hint: use the find and replace function in



Microsoft Word to change the allele sequences into the correct format.

* 1. ***Loading Allele Files:***

Loading the Allele files into Minimum SNPs follows the same procedure for all four methods (applications). As described in Section 3.1, a false locus should be the last allele file to be loaded into Minimum SNPs. The last allele will not be included in the MegaAlignment, but will act as a bookend or a full stop, in a manner of speaking.

* + - Open Minimum SNPs.
    - Click “File” on the menu bar.
    - Select the option “Load Allele File” from the dropdown menu.

o A new window opens which allows you to brows for the location of you allele files.

* + - Open the first allele file.

o The sequences with which the allele files are loaded does not matter at this point. However, I usually load the false allele last for consistency, as you will soon see.

* + - Repeat the steps above until all allele files have been loaded.

o You can keep track of where you are up to in loading allele files by the title given to the Minimum SNPs window.

Once you have loaded all the allele files, individual alleles can be viewed in task pane.

A locus is selected buy

* + - Click “File” on the menu bar.
    - Select the option “Alleles” from the dropdown menu.
    - From the second dropdown menu select the locus that you would like to use.

o Notice that the incorrect terminology has been used here. The option “Alleles” on the “File” dropdown menu should really read “Loci”.

* + - You will notice that each allele of the selected locus can be viewed by selecting from the drop down menu adjacent to the word “Allele” in the task bar.

* 1. ***Assembling Alleles and Loading ST File:***

Once all the alleles have been loaded into Minimum SNPs, they need to be assembled into a Mega-Alignment.

This is done by sequentially ‘joining’ the loci together followed by loading the sequence type file which of which alleles from each locus to include in the sequence of a particular sequence type. It is important to note that the loci do not need to be joined in any particular order except for the final locus, which is a false locus, acting as a bookend.

* + - Change the mode form “%” to “D” by clicking the “D” button on the task bar.
    - Click “File” on the menu bar.
    - Select the option “Alleles” from the dropdown menu.
    - From the second dropdown menu select the locus that you would to add fists.

o I usually work from top to bottom as they are listed in the drop down menu – the order in which they were loaded..

* + - On the task bar click “Add”
    - On the task bar click “Start”
    - On the task bar click “Accept”
    - Click “File” on the menu bar.
    - Select the option “Alleles” from the dropdown menu.
    - From the second dropdown menu select the locus that you would to add next.
    - Click “Add”
    - Click “Accept”
    - Repeat the above 5 steps until all loci have been joined (added) including the false locus.

o Once again, you can keep track of where you are up to in joining loci together (adding loci) by the title given to the Minimum SNPs window.

* On the task bar click “Finish”
* At this point a new dialog box opens enabling you to locate and open the sequence time file (called a strain file).
* Once you have opened the sequence type file the word “Mega-alignment” will appear in the title if the process has been successfully completed. If you do not see the word “Mega-Alignment”, then there has been an error in either the data within your allele or ST files or in the sequence of loading and adding alleles in Minimum SNPs.

This mega-alignment is now ready for analysis using the %, D or Not-N modes.

1. **Obtaining and loading a pre-concatenated MLST database**

Most MLST sites now provide the option of downloading concatenated data. The early versions of Minimum SNPs were designed to analyse data from individual MLST loci. As a consequence of this, Minimum SNPs can deal with a pre-concatenated MLST database as if it were, in effect, a single giant MLST locus, and each sequence type is an allele. This approach is very easy and straightforward.

* 1. *Downloading a pre-concatenated MLST database*

* + - Download and save the concatenated MLST data in FASTA format. In order for the program to run, all the sequences must be same length and composed only of G’s, A’s, T’s and C’s. We have found that sequences of the incorrect length are indeed present in some concatenated databases. These must be removed or “doctored” so they are the correct length.

* 1. *Loading pre-concatentated data into Minimum SNPs*

* + - Open Minimum SNPs.
    - Click “File” on the menu bar.
    - Select the option “Load Allele File” from the dropdown menu.

o A new window opens which allows you to browse for the location of your concatenated data file.

* + - Select the file that is the concatenated data in FASTA format.

The file will then load. If the file is large (e.g. thousands of STs), the loading process may take 20 seconds or more.

The concatenated MLST data is now ready for analysis using the %, D or Not-N modes.

1. **Deriving highly informative sets of SNPs.**

Minimum SNPs assembles highly informative sets of SNPs by carrying out an empirical search i.e. by assessing the informative powers of thousands of different SNP combinations. Three different algorithms – the %, D and Not-N modes – are available for measuring the informative powers of candidate SNP sets. For complete details see Robertson et al (2004), Price et al (2006), Price et al (2007).

***5. 1. % Mode***

This mode is used to identify SNP set(s) diagnostic for a single user-specified ST. It is called the % mode because the informative power is stated as the % of STs that are discriminated from the user-specified ST by the SNP(s).

To obtain and load data files, see Section 3.

***5.1.1. Using a mega-alignment (An MLST database that has been concatenated using the Minimum SNPs on-board concatenation facility)***

* + - * Change the mode from “D” to “%” by clicking the “D” button on the task bar.
      * Select the ST that you are interested from the dropdown menu adjacent to the word “Allele” in the top right corner of the task bar.
      * Select percent.
      * Click “Identify Allele” in the task bar.

The results pane will display the following information: o The loci included in the analysis o The ST that has been queried

* + - * + The concatenated sequence of that sequence type (according to the order in which the loci were joined together (added).
        + The identification constraints – see Tools and Options
        + The SNP’s returned by the Minimum SNP’s analysis and the corresponding confidence of identifying the sequence type queried from all other sequence types in the database using those SNP’s.
      * Click “Clear Report” on the results bar to start over OR;
      * Save and Print your Results

***5.1.2. Using a pre-concatenated MLST database downloaded from an MLST web site.***

* + - * Select the sequence type of interest in the “allele” drop-down menu.

* + - * Ensure that the “%” mode has been selected (top right hand corner). The symbol appears ghosted when it is selected.

* + - * Click “identify allele”.

The results pane will display the following information: o The ST that has been queried (labelled as an allele). o The sequence of that ST.

* + - * + The identification constraints – see Tools and Options
        + The SNP’s returned by the Minimum SNP’s analysis and the corresponding confidence of identifying the sequence type queried from all other sequence types in the database using those SNP’s.
      * Click “Clear Report” on the results bar to start over OR;
      * Save and Print your Results

***5. 2 “D” Method:***

This is used to identify SNP sets that are optimised towards discriminating all STs from all STs. This is accomplished by assessing the informative power of candidate SNP(s) by calculating the Simpsons Index of Diversity (*D*) with respect to the input sequence alignment. In this context, *D* is the probability that any two sequences selected at random from the alignment (without replacement), will be discriminated by the SNP set under test.

To obtain and load data files, see Section 3.

***5.2.1.*** ***Using a mega-alignment (An MLST database that has been concatenated using the Minimum SNPs on-board concatenation facility)***

* + - * Make sure that the mode selected in the task bar is “D” (The symbol appears ghosted when it is selected).
      * Click “Identify Allele” in the task bar. You may need to wait some time at this point. **Note: although it is possible to select an ST in the drop down menu, this has no effect on the results, because the D algorithm does not identify SNPs with reference to any particular ST.**

The results pane will display the following information: o The loci included in the analysis

o The identification constraints – see Tools and Options o The SNP’s returned by the Minimum SNP’s analysis and the corresponding Simpson’s index of diversity associated when the MLST database is assessed with the resulting SNPs.

* + - * Click “Clear Report” on the results bar to start over OR;
      * Save and Print your Results

***5.2.2. Using a pre-concatenated MLST database downloaded from an MLST web site.***

* + - * Ensure that the “D” mode has been selected (top right hand corner). The symbol appears ghosted when it is selected.

* + - * Click “identify allele”.

**Note: although it is possible to select an “allele” (actually an ST) in the drop down menu, this has no effect on the results, because the D algorithm does not identify SNPs with reference to any particular ST.**

The results pane will display the following information: o The identification constraints – see Tools and Options o The SNP’s returned by the Minimum SNP’s analysis and the corresponding Simpson’s index of diversity associated when the MLST database is assessed with the resulting SNPs.

* + - * Click “Clear Report” on the results bar to start over OR;
      * Save and Print your Results
  1. ***Not-N Method: (Note – the key strokes for this method are very different from the % and D modes)***

* + 1. ***Using a mega-alignment (An MLST database that has been concatenated using the Minimum SNPs on-board concatenation facility)***

* + - * Change the mode form “D” to “%” by clicking the “D” button on the task bar.
      * Click “Consensus” in the task bar.
      * In the data box adjacent to the “Consensus” button, type (or paste) the list of sequence types (ST’s) that you want to identify as a group.
        + The list of ST’s must have the format:

ST x1, ST x2, …,ST xn

Where x is the MLST sequence type and n the nth sequence type in the series.

Make sure there is a space after every “ST” and a space after every “,” and no other spaces.

* + - * Click “Not-N” in the task bar.

The results pane will display the following information: o The loci included in the analysis

* + - * + The identification constraints – see Tools and Options
        + The results pane is as for the % and D modes, with the exception that the selected group of sequences is listed, and the SNPs are defined in “NOT” mode, i.e. “NOT – AGT” means “C”, and “NOT – GC” means “A or T”.

* + - * Click “Clear Report” on the results bar to start over OR;
      * Save and Print your Results

* + 1. ***Using a pre-concatenated MLST database downloaded from an MLST web site.***

* + - * (A quirk of the software is that it does not matter which of the D or % buttons is selected.)

* + - * Click on the “consensus” button

* + - * In the data box adjacent to the “Consensus” button, type (or paste) the list of sequence types that you wish to identify as a group.
        + **Important:** **these must be typed EXACTLY as they are listed in the “allele” drop down menu. This is case sensitive, and spaces must be included if they are within the name. In addition, the “>” symbol that denotes sequence names in the FASTA format must also be included. The sequence names must be separated by commas, and there MUST be a space after each comma. If you are doing many Not-N analyses, it is helpful to use an input file in which the sequence names (in the Fasta format) are very short e.g. just “>x” where x is the ST number. You can then type the sequence as e.g. “>5, >28, >79, >766” in the consensus box.**

* + - * Click the Not-N button.

* + - * The results pane is as for the % and D modes, with the exception that the selected group of sequences is listed, and the SNPs are defined in “NOT” mode, i.e. “NOT – AGT” means “C”, and “NOT – GC” means “A or T”.

**Helpful hint.** It is common for Not-N analyses to fail to yield highly resolving sets of SNPs

– such sets may simply not exist. Therefore, it is important to not set the “confidence” level (under the “tools” drop down menu, “allele options” menu item) too high. If it is too high, the program will simply provide no results. It is prudent to start with the confidence at about 50% and work up from there.

***6. Working Backwards Method:***

“Working backwards” is calculating the STs defined by a particular user-defined SNP profile. The methods for doing this in Minimum SNPs are a little counter-intuitive, as is explained below:

* + 1. The software cannot work backwards using a mega-alignment that has been constructed using the on-board concatenation function. However, it can work backwards using the separate locus-specific alignments that are used to construct mega-alignments. Although this is fully functional, it is not very convenient. In particular, it requires that the software be re-started, and the sequence alignments reloaded.
    2. Working backwards using a pre-concatenated MLST database (i.e. a single alignment) is extremely quick and easy. The ease of working backwards is a major argument in favour of using pre-concatenated data.
    3. This mode does not operate by the user actually inputting the SNP profile. Rather, it operates as follows: if the program is given one or more SNPs and a particular ST, it will determine the SNP profile defined by that ST, and also determine which STs have the same SNP profile.

***6.1 Working backwards if you are using separate locus specific sequence alignments and an ST allele profiles file i.e. you usually work using the on-board “mega-alignment” concatenation function***

If you have already created a Mega-Alignment you will need to close Minimum SNPs and start over as working backwards does not make use of a Mega-Alignment.

* + - Load allele files
    - Change the mode form “D” to “%” by clicking the “D” button on the task bar.
    - Click “File” on the menu bar.
    - Select the option “Alleles” from the dropdown menu.
    - From the second dropdown menu select a locus that you would like to work with.
    - In the data box adjacent to “Identity Check” type the position of the SNP within the selected locus that you would like add to the SNP profile you are characterising. o If there is more than one position within that particular locus you must query both at once. Separate the SNP positions with a coma. Do not use any spaces. Example:

 36,241,243

* + - Click “Add” in the task bar.
    - Click “Start” in the task bar.
    - Click “Insert” in the task bar.
    - Check that the polymorphs presented in the data box adjacent to the “Start” button represent the profile that you are characterising. If they do not, modify the data by replacing the letter for the nucleotide at the appropriate position with the desired polymorph. Do not change the format of the data in the box at all.
    - Click “Accept” in the task bar.
    - Click “File” on the menu bar.
    - Select the option “Alleles” from the dropdown menu.
    - From the second dropdown menu select the next locus that you would like to work with.
    - In the data box adjacent to “Identity Check” type the position of the SNP within the second locus that you would like add to the SNP profile you are characterising.
    - Click “Add” in the task bar.
    - Click “Insert” in the task bar.
    - Check that the polymorphs presented in the data box adjacent to the “Start” button represent the profile that you are characterising
    - Click “Accept” in the task bar.
    - Repeat the above process for all loci included in the profile.
    - Click “Finish” on the menu bar.
    - At this point a new dialog box opens enabling you to locate and open the sequence time file (called a strain file).
    - Once you have opened the sequence type file the process is complete and the results will be presented in the results pane.

The results pane will display the following information:

o A list of alleles that share the same profile at each selected locus o The indistinguishable STs based on the SNPs that have been included in the profile.

* + - Click “Clear Report” on the results bar to start over OR;
    - Save and Print your Results

***6.2 Working backwards* u*sing a pre-concatenated MLST database downloaded from an MLST web site.***

* + Type the SNP positions in the “identity check” box. These must be separated by commas (in this instance, do NOT put a space next to the comma).
  + Select the ST that defines the SNP profile you are interested in, using the “allele” drop down menu
  + Click “add”
  + The results pane contains the SNP profile of the selected ST, and the other STs that share the same profile.
  + Click “Clear Report” on the results bar to start over OR;
  + Save and Print your Results

**7. Tools and Options**

The “allele options” page under the “tools” menu is very useful. The functions of these options are listed below

1. **1 Number of results**

Minimum SNPs assembles SNP sets by first finding the most informative SNP and calling that SNP1, and then finding the SNP that is most informative in combination with the SNP1, and calling that SNP2, and so forth. If there is a “tie” (i.e. the same score – a draw) between one or more SNPs, it uses that to define alternative SNP sets. In effect, each of the tied alternatives seeds new pathways of SNP set assembly. The “number of results” option simply sets an upper limit on the number of SNP sets that are listed in the results. Nearly all ties occur late in the process of SNP set assembly, so that if the the SNP set is kept small using e.g. the Confidence option, then often only one result will be returned.

* 1. **Paragraph width**

Affects the format of the results.

* 1. **Inclusions and Exclusions**

This option is extremely useful. It allows the user to tell the program to ignore SNPs

(exclusions), or to include SNPs in the output set, irrespective of resolving power

(inclusions). This allows SNP sets to be easily adapted and optimised. For example, a SNP that is technically difficult to interrogate can easily be replaced by the next most informative SNP, without the necessity for redesigning the whole SNP set. It is important to note that when a SNP is forced to be included in the set using the “include” function, its informative power is used in the derivation of other SNPs to be added to the set. “Included” SNPs can therefore be used to “seed” SNP sets. The “include” function is also a very easy way of determining the resolving power of SNP sets that have been identified by “Minimum SNPs”-independent methods.

* 1. **Time out (seconds)**

This also can be very important, although primarily for the trouble is causes, rather than the benefits. In essence, if it is set too low, then the program will terminate before finishing its calculation, and will return a result of “no results found”. It usually makes sense to set it to a high value – in particular if you are doing D mode searches on large databases. These searches can take up to several hours, especially on older computers (although more usually it is a few minutes at most). % and not-N searches should only take a few seconds.

1. **5 Confidence (1-100)**

This option is very important. It is the maximum resolving power that is aimed for in % and not-N searches. It is important to have this set at an appropriate value. This is because Minimum SNPs does not have good error-handling functions. If this value is set higher than can ever be reached – and this can happen if the not-N mode is used, or there are identical sequences in the input file, then the program will never finish its calculation, and will simply proceed until it times out, and returns no result. An efficient and low stress approach to doing searches is to start with the confidence set low, and work up. E.g. start with 95 for % searches, and 50 for not-N searches.

* 1. **Simpsons Index (0-1)**

This is the D value analog of the “confidence” option above. This option is even more important, because it can take a large number of SNPs to reach a D value of 1.0 with most MLST databases. As the calculation time increases exponentially with the number of SNPs, it can be impractical in terms of processing time to reach D = 1.0. Once again, it is prudent (and efficient) to begin with a low D (e.g. 0.9), and work up from there to see how high it is practical to achieve. A search that is limited by D = 0.9 will generally only take a few seconds, even on very large data sets.

* 1. **Search Depth**

We do not use this option very often, but it can be useful. This limits the size of the SNP set that is returned. It apparently over-rides the limits set by the Confidence or Simpsons Index options above, so e.g. if the Confidence is set to 99%, and the Search Depth to 2, then it may return a SNP set that does not reach 99%. Also (somewhat confusingly), the search depth is incorrect by 1, so that a search depth of 1 will actually allow the return of a two member SNP set, and so forth. It is most useful as a time out or infinite loop error avoidance strategy, although similar aims can be achieved with judicious use of the Confidence and Simpsons Index options.

* 1. **Number of loci**

This sets an upper limit on the number of loci that can be used in assembling megaalignments. There is no problem with having it set higher than the number of loci used – in particular having it set to “7” (the number of loci in MLST schemes) causes no problem in using pre-concatenated MLST datasets in the form of single alignments. Therefore, it is better to simply set it to 7 and then leave it alone.

## Research Plan

**A cohesive bioinformatics pipeline to support design and operation of SNP-based microbial genotyping methods.**

**Aims and significance**

The aim is to construct a bioinformatics software package that supports the knowledge-based design and use of highly efficient bacterial genotyping methods . The package will:

* Incorporate the functionality of previously published (by CI Giffard) software packages “Minimum SNPs9”, and “HRM Type4”.
* Enhance important aspects of the functionality and useability of the existing software, primarily to improve the ability to predict and interpret the results of single nucleotide polymorphism (SNP)-based bacterial genotyping methods.
* Interface with bioinformatics software pipelines installed on Cheetah (The CDU-Menzies high performance bioinformatics computer) that derive matrices of genome-wide orthologous SNPs from intraspecific comparisons of multiple bacterial genomes. Such matrices will serve as input for the proposed package.

***Background***

*Bacterial typing.* In clinical, public health, and research microbiology, there is frequently a need to divide bacterial species into “types”. A bacterial type may be thought of as a classification unit at a finer scale than “species”, and roughly synonymous with the term “strain”. Bacterial typing is used to trace instances and patterns of transmission and dissemination, and may be used at all scales from global to within individual buildings (e.g. hospitals), to indicate virulence or resistance properties of isolates, and in basic research to assist understanding of evolutionary history. While early typing methods were based upon reactions with standard sets of antisera (serotyping), or susceptibility to standard sets of viruses (phage typing), for decades, the great majority of bacterial typing has been performed using genetic methods. As a result, the term “genotyping” will be used in this document. Over the years, considerable ingenuity has been applied to the development of bacterial genotyping methods. Many methods have been described, with most being variations on the theme of electrophoresis of complex mixes of DNA fragments so as to yield banding pattern “fingerprints”, or sequencing of standardised gene fragment(s).

*History of project.* The genesis of this proposal goes back more than a decade. In the early 2000s, Phil Giffard led a project within the QUT node of the Cooperative Research Centre for Diagnostics. This was essentially a re-think of bacterial genotyping methodology. The innovative concept was that the volume of known comparative gene sequence data from within bacterial species was exploding. This constitutes a suburb resource to mine for sets of variable genetic locations (single nucleotide polymorphisms (SNPs)) that are optimised for use in bacterial genotyping methods with predefined performance specifications. In this way, genotyping methods that yield the needed resolving power or other information as efficiently as possible could be easily designed. An added impetus for this area of research was rapid development in technology for interrogating SNPs. The first publication outlining this approach was published in 20049. This reported the construction of the SNP-mining “Minimum SNPs” software package, and the demonstration of two bacterial genotyping methods, using allele specific PCR on a real time PCR platform. This was followed by demonstration of new and improved genotyping methods e.g. 2, 3, 5, 7, 10.

Concurrently with this, CI Giffard pioneered the application of high resolution melting analysis (HRMA) to bacterial genotyping 6, 11. This method is based upon the controlled melting of an amplified DNA fragment in the presence of a dye that fluoresces when bound to double stranded nucleic acid. The relationship between fluorescence and temperature (the melting curve) is information rich; in effect a spectrum. While it is impossible to infer a DNA sequence *de novo* from a melting curve, discrimination of multiple sequence variants (alleles) can be demonstrated. The innovation within the research led by CI Giffard was demonstration as to just how effective HRMA is at resolving multiple alleles, and the application of this to bacterial genotyping.

In 2008 CI Giffard moved to the Menzies School of Health Research. In his first few years at Menzies, a major research activity was combining the SNP-based genotyping with HRMA i.e. the platform technology for SNP interrogation was changed from allele specific PCR to HRMA. The advantage of this was that SNPs identified using Minimum SNPs often have neighbouring “bystander” SNPs, resulting in the amplified fragment containing multiple SNPs that define multiple alleles. Resolution of all those alleles by HRMA very significantly increases the resolving power of the relevant genotyping method. However, this innovation makes data analysis complex. To underpin this, another software package, “HRMtype” was written, primarily by Steven Tong 4. This program converts known alleles into predicted HRMA-resolved alleles on the basis of predicted melting point, and converts multilocus sequence information into predicted multilocus HRMA genotypes. This genotyping format has been termed “minim typing” and has been applied to four bacterial species1, 4, 8, 12. It is used extensively at Menzies.

A major development in very recent years has been the advent of low cost whole genome sequencing of bacteria. The work described above made extensive use of multilocus sequence typing (MLST) databases as the input data for SNP mining. MLST was developed in the UK (e.g. see <http://saureus.mlst.net/>) , as a major and very successful initiative to standardize bacterial genotyping methods and terminology. For each bacterial species for which it has been implemented, seven standardised gene fragments are defined. These are sequenced to generate the “sequence type” (ST). The MLST database curators maintain comprehensive internet accessible information regarding sequencing variants at the loci (alleles) and the alleles found together in bacterial isolates (STs). Often several thousand STs are known. Our SNP based typing methods have been based on mining alignments of these STs. The MLST loci only cover ~0.1% of the bacterial genome. The advent of low cost whole genome sequencing has made MLST somewhat obsolete, because it is now possible to genotype bacteria by whole genome sequencing. It is still quite expensive (>$200 per isolate), but that is comparable in cost to MLST, and the whole genome provides ~1000 time more resolving power. Accordingly CI Giffard’s work is shifting to SNP mining from alignments of entire genomes rather than alignments of small fragments of genomes such as MLST databases. This means that ~1000 more SNPs are available for selecting optimal combinations, resulting in much higher performance of the genotyping methods. A prototype *Chlamydia trachomatis* genotyping method based on SNPs extracted from whole genome alignments is in the late stages of validation (unpublished data). This method entails HRMA of just two gene fragments, but resolves *C. trachomatis* into 15 major evolutionary lineages. The method was designed using Minimum SNPs to analyse a matrix of genome-wide orthologous SNPs, and is integral to a pending NHMRC project grant application.

A limiting factor in the development of these methods, and their adoption by others is that the critical software packages are very user unfriendly, and have major gaps in their functionality:

1. Minimum SNPs and HRMType are written in completely different languages (Java and Stata “do file’ respectively), and do not interface.
2. Both programs are very sensitive regarding input format, and preparation of input files is very time consuming.
3. Neither program comprehensively supports the analysis of genotype assays results, with Minimum SNPs being particularly poor in this regard.
4. HRMType is very user-unfriendly with respect to accommodating experimentally determined variation from predicted HRMA alleles.
5. There are extremely effective bioinformatics pipeline installed in Cheetah, yielding matrices of SNP variation from multiple genomes. Minimum SNPs cannot at all easily be made to automatically interface with these pipelines.

**Research plan, methods and techniques.**

The research pan is straightforward. A computer program will be written that:

* Replicates the functionality of the “Minimum SNPs” and “HRMType” programs
* Provides greatly improved ease of use
* Provides greatly improved capacity for the prediction and interpretation of genotype assay results
* Interfaces with bioinformatics pipelines on Cheetah that yield matrices of genome-wide orthologous SNPs.

The current intention is that the software will be written in the “R” language. This is a very widely used language that is particularly suitable for statistical and graphical tasks. R is becoming the software engineering *lingua franca* for collaboration between Menzies and CDU IT researchers.

The primary performance specification is that the software must:

* Be easily useable
* Form the basis of a high quality publication
* To be able to be downloaded and readily used by other researchers in this field.

The outputs will be:

* The software itself
* Comprehensive documentation/user manual
* The software loaded into the UNIX environment on Cheetah, and interfaced with other bioinformatics pipelines that yield genome wide orthologous SNP matrices.
* The software in a form that it can be downloaded via the WWW, and run in a Microsoft Windows environment on a PC. This will involve an appropriate materials transfer agreement, probably in the form of a click-wrap licence.

**Participants’ roles**

CI Giffard: Project coordination, making use of his deep knowledge of the functionality and user experience required.

CI Shaw: Deep knowledge of software engineering, particularly in R language, emerging interest in bioinformatics applications and research. CI Shaw will provide close supervision of the software engineering.

Participant Kleinecke: Mariana Kleinecke is currently employed at 50% FTE by Menzies to be the administrator and software manager for Cheetah. She does not currently have other employment so has additional time available. The intention is that Mariana will be employed on a casual basis to carry out the software engineering, with extensive guidance from CI Shaw.

Participant Holt: Deb Holt is a colleague of CI Giffard at Menzies, with extensive experience with bioinformatics, including “Minimum SNPs”, “HRMType”, and genome assembly and comparison pipelines on Cheetah. Her expertise and input will facilitate all aspects of the project.

Participant Harris: Tegan Harris is also employed at Menzies, and has similar expertise and interests as Deb Holt, including being a highly experienced “power user” of Cheetah pipelines. Tegan’s perspective will be of great value.

**Justification of budget.**

The budget of $10,000 will provide 157 hours of causal employment of Mariana Kleinecke at CDU Academic level A7. It is judged that this will be sufficient to achieve a working package that meets the specifications above, provided there is also significant input from the investigators and other participants.

**Timetable**

It is anticipated that the software will be working, and ready for description in a publication, by the end of 2017. A publication will be prepared in the first half of 2018.

**References**

1 P. Andersson, S. Y. Tong, J. M. Bell, J. D. Turnidge, and P. M. Giffard, 'Minim Typing--a Rapid and Low Cost Mlst Based Typing Tool for *Klebsiella Pneumoniae*', *PLoS One,* 7 (2012), e33530.

2 E. Honsa, T. Fricke, A. J. Stephens, D. Ko, F. Kong, G. L. Gilbert, F. Huygens, and P. M. Giffard, 'Assignment of *Streptococcus agalactiae* Isolates to Clonal Complexes Using a Small Set of Single Nucleotide Polymorphisms', *BMC Microbiol,* 8 (2008), 140.

3 F. Huygens, J. Inman-Bamber, G. R. Nimmo, W. Munckhof, J. Schooneveldt, B. Harrison, J. A. McMahon, and P. M. Giffard, '*Staphylococcus ureus* Genotyping Using Novel Real-Time Pcr Formats', *J Clin Microbiol,* 44 (2006), 3712-9.

4 R. A. Lilliebridge, S. Y. Tong, P. M. Giffard, and D. C. Holt, 'The Utility of High-Resolution Melting Analysis of Snp Nucleated Pcr Amplicons--an Mlst Based *Staphylococcus aureus* Typing Scheme', *PLoS One,* 6 (2011), e19749.

5 E. P. Price, J. Inman-Bamber, V. Thiruvenkataswamy, F. Huygens, and P. M. Giffard, 'Computer-Aided Identification of Polymorphism Sets Diagnostic for Groups of Bacterial and Viral Genetic Variants', *BMC Bioinformatics,* 8 (2007), 278.

6 E. P. Price, H. Smith, F. Huygens, and P. M. Giffard, 'High-Resolution DNA Melt Curve Analysis of the Clustered, Regularly Interspaced Short-Palindromic-Repeat Locus of *Campylobacter Jejuni*', *Appl Environ Microbiol,* 73 (2007), 3431-6.

7 E. P. Price, V. Thiruvenkataswamy, L. Mickan, L. Unicomb, R. E. Rios, F. Huygens, and P. M. Giffard, 'Genotyping of *Campylobacter Jejuni* Using Seven Single-Nucleotide Polymorphisms in Combination with Flaa Short Variable Region Sequencing', *J Med Microbiol,* 55 (2006), 1061-70.

8 L. J. Richardson, S. Y. Tong, R. J. Towers, F. Huygens, K. McGregor, P. K. Fagan, B. J. Currie, J. R. Carapetis, and P. M. Giffard, 'Preliminary Validation of a Novel High-Resolution Melt-Based Typing Method Based on the Multilocus Sequence Typing Scheme of *Streptococcus pyogenes*', *Clin Microbiol Infect,* 17 (2011), 1426-34.

9 G. A. Robertson, V. Thiruvenkataswamy, H. Shilling, E. P. Price, F. Huygens, F. A. Henskens, and P. M. Giffard, 'Identification and Interrogation of Highly Informative Single Nucleotide Polymorphism Sets Defined by Bacterial Multilocus Sequence Typing Databases', *J Med Microbiol,* 53 (2004), 35-45.

10 A. J. Stephens, F. Huygens, J. Inman-Bamber, E. P. Price, G. R. Nimmo, J. Schooneveldt, W. Munckhof, and P. M. Giffard, 'Methicillin-Resistant *Staphylococcus aureus* Genotyping Using a Small Set of Polymorphisms', *J Med Microbiol,* 55 (2006), 43-51.

11 A. J. Stephens, J. Inman-Bamber, P. M. Giffard, and F. Huygens, 'High-Resolution Melting Analysis of the Spa Repeat Region of *Staphylococcus aureus*', *Clin Chem,* 54 (2008), 432-6.

12 S. Y. Tong, S. Xie, L. J. Richardson, S. A. Ballard, F. Dakh, E. A. Grabsch, M. L. Grayson, B. P. Howden, P. D. Johnson, and P. M. Giffard, 'High-Resolution Melting Genotyping of *Enterococcus faecium* Based on Multilocus Sequence Typing Derived Single Nucleotide Polymorphisms', *PLoS One,* 6 (2011), e29189.

## Technical documentation

R documentation

of ‘minSNP’

October 5, 2017

branch.simpson branch.simpson *is used to calculate 1 or more simpson index and list the position with highest index.*

Description branch.simpson is used to calculate 1 or more simpson index and list the position with highest index.

Usage branch.simpson(seq, level = 1, included = NULL, excluded = NULL,

numRes = 1)

# Arguments

seq is fastaDNA object to analysed. level is the number of positions.

included is the included position for analysed, these will force the computation to compute the simpson’s index at the position no matter what

excluded is the positions that are excluded from computation numRes is the number of result to be returned.

# Value

Will returns simpson’s index and the position.

Chlamydia\_1 *Normal Fasta sample file Chlamydia sequence type with 1 deletion at 1*

# Description

Normal Fasta sample file Chlamydia sequence type with 1 deletion at 1

1

## flagAllele

Chlamydia\_2 *Normal Fasta sample file Chlamydia sequence type with multiple deletions at 1, 3, 6*

# Description

Normal Fasta sample file Chlamydia sequence type with multiple deletions at 1, 3, 6

Chlamydia\_mapped *Normal Fasta sample file Chlamydia sequence types with no deletion*

# Description

Normal Fasta sample file Chlamydia sequence types with no deletion

flagAllele flagAllele *is used to find out a list of allelic profiles that has been flagged and will not be included in computation of minimum SNPs.*

# Description

flagAllele is used to find out a list of allelic profiles that has been flagged and will not be included in computation of minimum SNPs.

Usage flagAllele(seq)

# Arguments

seq a list of SeqFastadna. To keep it simple, use read.fasta from seqinr to import the

fasta file.

# Value

Will return a list of ignored allelic profiles.

## present.percent

present.percent present.percent *is used to find present and filter the similarity calculated using* similar.percent*.*

Description present.percent is used to find present and filter the similarity calculated using similar.percent.

Usage present.percent(result, percent = 100, number = 100)

# Arguments

|  |  |
| --- | --- |
| result | the result from similar.percent. |
| percent | minimum percentage to be included |
| number | number of results to be displayed |

# Value

Will return the a list of SNPs (as specified) that can be used and the associated percentage at the particular location.

present.simpson present.simpson *is used to present the result from the calculation of simpson’s index.*

Description present.simpson is used to present the result from the calculation of simpson’s index.

Usage present.simpson(seq, result)

# Arguments

|  |  |
| --- | --- |
| seq | is fastaDNA object to analysed. |
| result | is the result from branch.simpson. |

# Value

Will returns the presentation of the result.

## similar.percent

processAllele processAllele *is used to returned the processed allelic profiles.*

Description processAllele is used to returned the processed allelic profiles.

Usage processAllele(seq)

Arguments seq a list of SeqFastadna. To keep it simple, use read.fasta from seqinr to import the

fasta file.

# Value

Will return the processed allelic profiles.

result *Result file for validation Result from old software*

# Description

Result file for validation Result from old software

similar.percent similar.percent *is used to find the calculate the percentage of sim-*

*ilarity at alleles.*

Description similar.percent is used to find the calculate the percentage of similarity at alleles.

Usage similar.percent(seq, ref)

# Arguments

|  |  |
| --- | --- |
| seq | a list of SeqFastadna. To keep it simple, use read.fasta from seqinr to import the fasta file. |
| ref | the specific allele to be identified. |

# Value

Will return the a list of SNPs that can be used

## similar.simpson

similar.simpson similar.simpson *is used to calculate the simpson index and list the position with highest index.*

Description similar.simpson is used to calculate the simpson index and list the position with highest index.

Usage similar.simpson(seq, level = 1, included = NULL, excluded = NULL)

# Arguments

|  |  |
| --- | --- |
| seq | is fastaDNA object to analysed. |
| level | is the number of positions. |
| included | is the included position for analysed, these will force the computation to compute the simpson’s index at the position no matter what |
| excluded | is the positions that are excluded from computation |

# Value

Will returns simpson’s index and the position.

simpson.calculate simpson.calculate *is used to calculate the simpson’s index given a*

*pattern.*

Description simpson.calculate is used to calculate the simpson’s index given a pattern.

Usage simpson.calculate(pattern, N)

Arguments pattern is a pattern, can be a vector or a list.

N is the total number of entities that are in the pattern.

# Value

Will returns the simpson’s index for the pattern.

## usualLength

simpson.pattern simpson.pattern *is used to generate pattern for calculation at a later*

*stage.*

Description simpson.pattern is used to generate pattern for calculation at a later stage.

Usage simpson.pattern(seq, position, appended = NULL)

# Arguments

seq is fastaDNA object to analysed. position is the position of the sequences used to generate pattern. appended is the pattern that the current operation will appended onto

# Value

Will returns the generated pattern.

usualLength usualLength *is used to find out the length of the sequence (W/O dele-*

*tion).*

Description usualLength is used to find out the length of the sequence (W/O deletion).

Usage usualLength(seq)

# Arguments

seq a list of SeqFastadna. To keep it simple, use read.fasta from seqinr to import the

fasta file.

# Value

Will return the maximum length of all the allelic profiles.

*usualLength*

# Examples

sample.case1 <-function(){

#Read the file

Chlamydia <- read.fasta(file='../data/Chlamydia\_mapped.txt')

#STEP 1. Process the file

Chlamydia <- processAllele(Chlamydia)

untempered <-read.fasta(file='../data/Chlamydia\_mapped.txt')

#Since Chlamydia is normal checkIdentical(Chlamydia, untempered)

#PERCENT MODE result=similar.percent(Chlamydia, 'A\_D213') present=present.percent(result, 98, 100)

#All result should have percent higher or equal to 98 for (a in 1:100){

checkTrue(present[[a]]$percent>=98) }

#SIMPSON MODE result=branch.simpson(Chlamydia, level=1, numRes=3) output=present.simpson(Chlamydia, result)

#Should have 3 results checkTrue(length(output)==3)

checkEquals(output[[1]]$'Index', 0.7344, tolerance=0.00016) Description=paste('At position:', '1988', sep='-') checkEquals(output[[1]]$'Description', Description)

checkEquals(output[[2]]$'Index', 0.7318, tolerance=0.00016) Description=paste('At position:', '2044', sep='-') checkEquals(output[[2]]$'Description', Description)

checkEquals(output[[3]]$'Index', 0.7266, tolerance=0.00016) Description=paste('At position:', '2034', sep='-') checkEquals(output[[3]]$'Description', Description)

}

sample.case2 <-function(){

#Read the file

Chlamydia <- read.fasta(file='../data/Chlamydia\_1.txt')

#STEP 1. Process the file

Chlamydia <- processAllele(Chlamydia)

#Since 1 sequence has deletion and is ignored checkEquals(length(Chlamydia), 55)

#PERCENT MODE

result=similar.percent(Chlamydia, 'A\_D213') present=present.percent(result, 98, 100)

#Should have no result because A\_D213 is ignored

checkEquals(result, NULL) checkEquals(present, list())

#PERCENT MODE result=similar.percent(Chlamydia, 'H\_S1432') present=present.percent(result, 98, 100)

#Check result checkEquals(length(present), 100) checkEquals(present[[1]]$'position', 171) checkEquals(present[[1]]$'percent', 100)

#SIMPSON MODE

result=branch.simpson(Chlamydia, level=1, numRes=3) output=present.simpson(Chlamydia, result)

#Should have 3 results checkTrue(length(output)==3)

checkEquals(output[[1]]$'Index', 0.7347, tolerance=0.00016) Description=paste('At position:', '1988', sep='-') checkEquals(output[[1]]$'Description', Description)

checkEquals(output[[2]]$'Index', 0.7306, tolerance=0.00016) Description=paste('At position:', '2044', sep='-') checkEquals(output[[2]]$'Description', Description)

checkEquals(output[[3]]$'Index', 0.7199, tolerance=0.00016) Description=paste('At position:', '2034', sep='-') checkEquals(output[[3]]$'Description', Description)

}

sample.case3 <-function(){

#Read the file

Chlamydia <- read.fasta(file='../data/Chlamydia\_2.txt')

#STEP 1. Process the file

Chlamydia <- processAllele(Chlamydia)

#Since 1 sequence has deletion and is ignored checkEquals(length(Chlamydia), 53)

#PERCENT MODE result=similar.percent(Chlamydia, 'A\_D213') present=present.percent(result, 98, 100)

#Should have no result because A\_D213 is ignored

checkEquals(result, NULL) checkEquals(present, list())

#PERCENT MODE

result=similar.percent(Chlamydia, 'H\_S1432') present=present.percent(result, 98, 100)

#Check result checkEquals(length(present), 100) checkEquals(present[[1]]$'position', 171) *usualLength* checkEquals(present[[1]]$'percent', 100)

#SIMPSON MODE result=branch.simpson(Chlamydia, level=1, numRes=3) output=present.simpson(Chlamydia, result)

#Should have 3 results checkTrue(length(output)==3)

checkEquals(output[[1]]$'Index', 0.7343, tolerance=0.00016) Description=paste('At position:', '2044', sep='-') checkEquals(output[[1]]$'Description', Description)

checkEquals(output[[2]]$'Index', 0.7329, tolerance=0.00016) Description=paste('At position:', '1988', sep='-') checkEquals(output[[2]]$'Description', Description)

checkEquals(output[[3]]$'Index', 0.7263, tolerance=0.00016) Description=paste('At position:', '2034', sep='-') checkEquals(output[[3]]$'Description', Description)

}

test.deactivation <- function()

{

DEACTIVATED('Deactivating integration test function')

}

# Index

branch.simpson, 1

Chlamydia\_1, 1

Chlamydia\_2, 2

Chlamydia\_mapped, 2 flagAllele, 2

present.percent, 3 present.simpson, 3 processAllele, 4 result, 4

similar.percent, 4 similar.simpson, 5 simpson.calculate, 5 simpson.pattern, 6 usualLength, 6

10

## Test documentation

**Unit Testing output**

<testsuites errors="0" failures="0" tests="13">

<testsuite errors="0" failures="0" name="minSNP RUnit Tests" tests="13">

<testcase name="test.deactivation">

<skipped/>

</testcase>

<testcase name="test.flag" time="0.38"/>

<testcase name="test.processAllele" time="0.35"/>

<testcase name="test.setUp" time="0.36"/>

<testcase name="test.usualLength" time="0.36"/>

<testcase name="test.deactivation">

<skipped/>

</testcase>

<testcase name="test.present.percent" time="3.42"/>

<testcase name="test.setUp" time="0.14"/>

<testcase name="test.similar.percent" time="3.23"/>

<testcase name="test.branch.simpson" time="174.12"/>

<testcase name="test.deactivation">

<skipped/>

</testcase>

<testcase name="test.present.simpson" time="103.47"/>

<testcase name="test.setUp" time="0.14"/>

<testcase name="test.similar.simpson" time="104.33"/>

<testcase name="test.simpson.calculate" time="0"/>

<testcase name="test.simpson.pattern" time="0.12"/>

</testsuite>

</testsuites>

**TEST 1: unit1.R**

**PASSED: TRUE**

What have been tested:

Check that the function returns the normal length correctly for all 3 different files

When the sequence within the test file are all of the same length

When there is a sequence within the test file that is not of the same length

When there are multiple sequences within test file that are not of the same length

Check that processed Allele returns correctly

Processed allelic profiles have removed the profiles with deletion only removed correct profiles

**TEST 2: unit2.R**

**PASSED: TRUE**

What have been tested:

Return NULL when the reference allele can't be found

Returning percentage at every SNP

Check the range of percentage (within 0 and 100)

Check the results of 352 SNPs (result from original minSNP software)

Check number of result expected are correct

Expect no result when the parameter is wrong

**TEST 3: unit3.R**

**PASSED: TRUE**

What have been tested:

Check simpson's index with normal pattern

Check simpson's index at extreme

Generate pattern from sequences when it's just 1 level

Appending pattern to result from 1st level to 2nd level

Comparing simpson's with result from minSNP at position 1

Comparing simpson's with result from minSNP at position 2

Check the expected result, ensuring that in level 3, index is higher, and position is not the same as in level 1 and 2

Check the expected result, ensuring that in level 4, index is higher, and position is not the same as in level 1, 2, and 3

Check the 3 results, 2nd result should ignore the first result at level 1 & 3rd result should ignore both the 1st and 2nd at level 1

Check that the pair of position returned by the 3 results are correct

Confirm the number of result

Check the result at 1st level for the 3 outputs

**Integration test: integrated.R**

**PASSED: TRUE**

Output

<testsuites errors="0" failures="0" tests="3">

<testsuite errors="0" failures="0" name="minSNP Integration Tests" tests="3">

<testcase name="sample.case1" time="31.83"/>

<testcase name="sample.case2" time="28.58"/>

<testcase name="sample.case3" time="28.95"/>

</testsuite>

</testsuites>

Case 1: None of the sequence types with no deletion

Case 2: One of the sequence types with deletion

Case 3: Multiple sequence types with deletions