snpgenie Documentation

Damien Farrell

Nov 15, 2022

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

snipgenie is a desktop and command line tool for microbial variant calling and phylogenetic analysis from raw read data. It is primarily written to be used with bacterial isolates of MBovis but can be applied to other species. This is in early stages of development. Anyone interested in using the software is encouraged to make sugggestions on improving or adding features.

This software is written in Python and is developed with the Qt toolkit using PySide2. It was made on Ubuntu linux but is designed to also run on Windows 10 with a standalone application.

1.1 Command line tool

This tool works from the command line and via Python scripts. Unlike many other SNP calling pipelines, it is also designed to have a graphical user interface, which is in development.

1.2 Current Features

- load multiple fastq files and process together
- view fastq quality statistics
- trim reads
- align to reference
- view bam alignments
- call variants
- filter variants
- create SNP core multiple sequence alignment
- create phylogenetic tree

1.3 Links

- https://github.com/dmnfarrell/snipgenie

CHAPTER

TWO

INSTALLATION

2.1 Linux

With pip:

```
pip install -e git+https://github.com/dmnfarrell/snipgenie.git#egg=snipgenie
```

Note: You may need to use pip3 on Ubuntu to ensure you use Python 3. Use sudo if installing system-wide. Running this also requires you have git installed. The same command can be used to update to the latest version.

Install binary dependencies:

```
sudo apt install bcftools samtools bwa
```

2.2 Windows

The pip instructions will work if you have installed Python for Windows. A standalone installer will be used to deploy on windows.

CHAPTER

THREE

DEPENDENCIES

For Linux installs, you require Python 3 and the following packages. These will be installed automatically when using pip.

```
numpy
pandas
matplotlib
biopython
pyvcf
pyfaidx
pyside2 (GUI only)
toytree (GUI only)
```

Other binaries required:

```
bwa
samtools
bcftools
tabix
parallel
```

These binaries can be installed with apt in Ubuntu:

```
sudo apt install bwa samtools bcftools tabix parallel
```

If you want a tree to be built you should install RaXML, but it's optional:

```
sudo apt install raxml
```

The binaries are downloaded automatically in Windows.

CHAPTER

FOUR

FUNDING

The development of this software was largely enabled through funding by the Irish Department of Agriculture.

CHAPTER

FIVE

USAGE

The program includes both a command line and graphical interface. Both will produce the same results.

5.1 Command Line

This will run the entire process based on a set of options given at the terminal:

```
-h, --help            show this help message and exit
-i FILE, --input FILE
                      input folder(s)
-e LABELSEP, --labelsep LABELSEP
                      symbol to split the sample labels on
-r FILE, --reference FILE
                      reference genome filename
-S SPECIES, --species SPECIES
                      set the species reference genome, overrides -r
-g FILE, --genbank_file FILE
                      annotation file, optional
-t THREADS, --threads THREADS
                      cpu threads to use
-w, --overwrite       overwrite intermediate files
-T, --trim            whether to trim fastq files
-U, --unmapped        whether to save unmapped reads
-Q QUALITY, --quality QUALITY
                      right trim quality, default 25
-f FILTERS, --filters FILTERS
                      variant calling post-filters
-m MASK, --mask MASK  mask regions from a bed file
-c, --custom          apply custom filters
-p PLATFORM, --platform PLATFORM
                      sequencing platform, change to ont if using oxford nanopore
-a ALIGNER, --aligner ALIGNER
                      aligner to use, bwa, subread, bowtie or minimap2
-b, --buildtree       whether to build a phylogenetic tree, requires RaXML
-N BOOTSTRAPS, --bootstraps BOOTSTRAPS
                      number of bootstraps to build tree
-o FILE, --outdir FILE
                      Results folder
-q, --qc              Get version
-d, --dummy           Check samples but don't run
```

(continues on next page)

(continued from previous page)

```
-x, --test            Test run
-v, --version         Get version
```

Examples:

Call with your own reference fasta file:

```
snipgenie -r reference.fa -i data_files -o results
```

Use an in built species genome as reference. This will also supply an annotation file. The current options are Mbovis-AF212297, MTB-H37Rv, MAP-K10, M.smegmatis-MC2155:

```
snipgenie -S Mbovis-AF212297 -i data_files -o results
```

Provide more than one folder:

```
snipgenie -r reference.fa -i data_files1 -i data_files2 -o results
```

Provide an annotation (genbank format) for consequence calling:

```
snipgenie -r reference.fa -g reference.gb -i data_files -o results
```

Add your own filters and provide threads:

```
snipgenie -r reference.fa -i data_files -t 8 -o results` \
 -f 'QUAL>=40 && INFO/DP>=20 && MQ>40'
```

5.1.1 Aligners

You can use any one of the following aligners: bwa, subread, bowtie or minimap2. These should be present on your system, unless using the Windows version. Note that for oxford nanopore reads you should use minimap2 and specify the platform as 'ont'.

5.1.2 Mask file

You can selectively mask snp sites such as those contained in transposons or repetitive regions from being included in the output. You need to provide a bed file with the following columns: chromosome name, start and end coordinates of the regions. There is currently a built-in mask file used for M.bovis and of you select this genome as reference using the –species option it will be used automatically. Example:

```
LT708304.1     105359          106751
LT708304.1     131419          132910
LT708304.1     149570          151187
LT708304.1     306201          307872
```

5.1.3 Inputs

You can provide single folder with all the files in one place or multiple folders. Folders are searched recursively for inputs with extensions *.f*q.gz. So be careful you don't have files in the folders you don't want included. The following file structure will load both sets of files if you provide the parent folder as input. You can also provide multiple separate folders using -i as shown above.

For example if you provide -i data with the following structure:

```
data/
├── ERR1588781
│   ├── ERR1588781_1.fq.gz
│   └── ERR1588781_2.fq.gz
└── ERR1588785
    ├── ERR1588785_1.fastq.gz
    └── ERR1588785_2.fastq.gz
```

Filenames are parsed and a sample name is extracted for each pair (if paired end). This is simply done by splitting on the _ symbol. So a file called /path/13-11594_S85_L001-4_R1_001.fastq.gz will be given a sample name 13-11594. As long as the sample names are unique this is ok. If you had a file names like A_2_L001-4_R1_001, A_3_L001-4_R1_001 you should split on '-' instead. You can specify this in the labelsep option. The workflow won't run unless sample names are unique.

5.1.4 Outputs

These files will be saved to the output folder when the workflow is finished:

```
raw.bcf - unfiltered output from bcftools mpileup, not overwritten by default
calls.vcf - unfiltered variant calls
filtered.vcf.gz - filtered vcf from all variant calls
snps.vcf.gz - snps only calls, used to make the core alignment
indels.vcf.gz - indels only, made from filtered calls
core.fa - fasta alignment from core snps, can be used to make a phylogeny
core.txt - text table of core snps
csq.tsv - consequence calls (if genbank provided)
csq_indels.tsv - consequence calls for indels
csq.matrix - matrix of consequence calls
snpdist.csv - comma separated distance matrix using snps
samples.csv - summary table of samples
RAxML_bipartitions.variants - ML tree if RAxML was used, optional
tree.newick - tree with SNPs branch lengths, if RAxMl used
```

5.2 Use from Python

You can run a workflow from within Python by importing the snipgenie package and invoking the WorkFlow class. You need to provide the options in a dictionary with the same keywords as the command line. Notice in this example we are loading files from two folders.

```
from sngenie import app
args = {'threads':8, 'outdir': 'results', 'labelsep':'-',
        'input':['/my/folder/',
```

(continues on next page)

(continued from previous page)

```
                '/my/other/folder'],
        'reference': None, 'overwrite':False}
W = app.WorkFlow(**args)
st = W.setup()
W.run()
```

CHAPTER

SIX

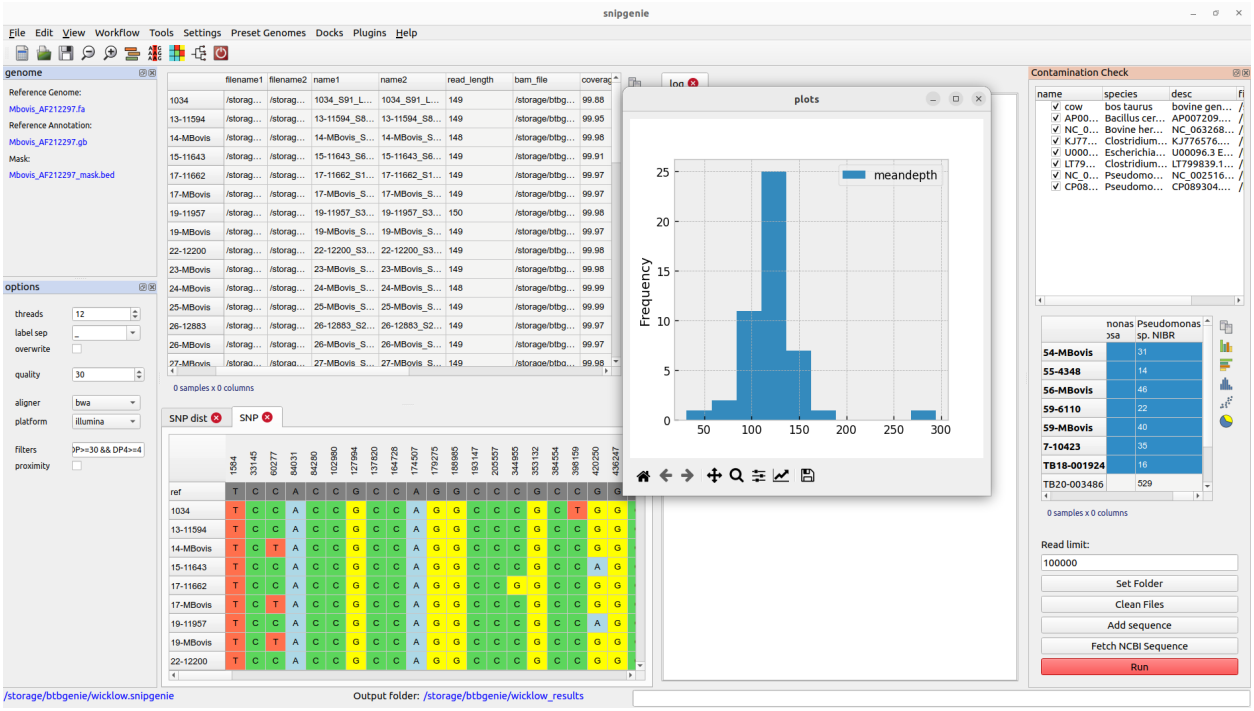
DESKTOP APPLICATION

The graphical user interface is designed for those not comfortable with the command line and includes some additional features. It requires the installation of either PyQt5 or PySide2 if using the pip install. A windows installer for this application will be available separately.

Current features include:

- Simple fastq quality analysis
- View read alignments
- Phylogenetic tree viewing
- Contamination checker

6.1 Interface



6.2 Basic workflow

- The first step is to set an output folder for the results. Choose *Settings->Set Output Folder*. Note that if this folder already contains a set of results from a previous run of the command line tool the program will attempt to load the sample table.
- Set a reference genome either by using a preset species or loading a fasta sequence you have previously identified as the one you wish to use. To load a preset use the Preset Genomes menu. The reference should currently be a single chromosome. Preset genomes have an annotation (genbank format) and sometimes a mask file (bed format) associated with them. However you can run without these.
- Save the project somewhere. It will be saved a single file with a .snipgenie extension. This only saves the loaded tables and settings and not the output results.
- Load the fastq files you wish to analyse. These can be selected individually or and entire folder added. Use *File->Add Folder* or *File->Add Fastq Files*. When the files are loaded the samples table will be updated to reflect the files and their assigned labels. See importing files.
- Once files are loaded you can being analysis. Prior to alignment you may want to check your files for contamination, though this can be done at a later stage to exclude samples causing problems (e.g. samples that have poor depth).
- The first step is align the fastq files. This is accessed from the Workflow menu. You should select the files in the table to be aligned. Just select all rows to align everything.
- After alignment you will see the table updated

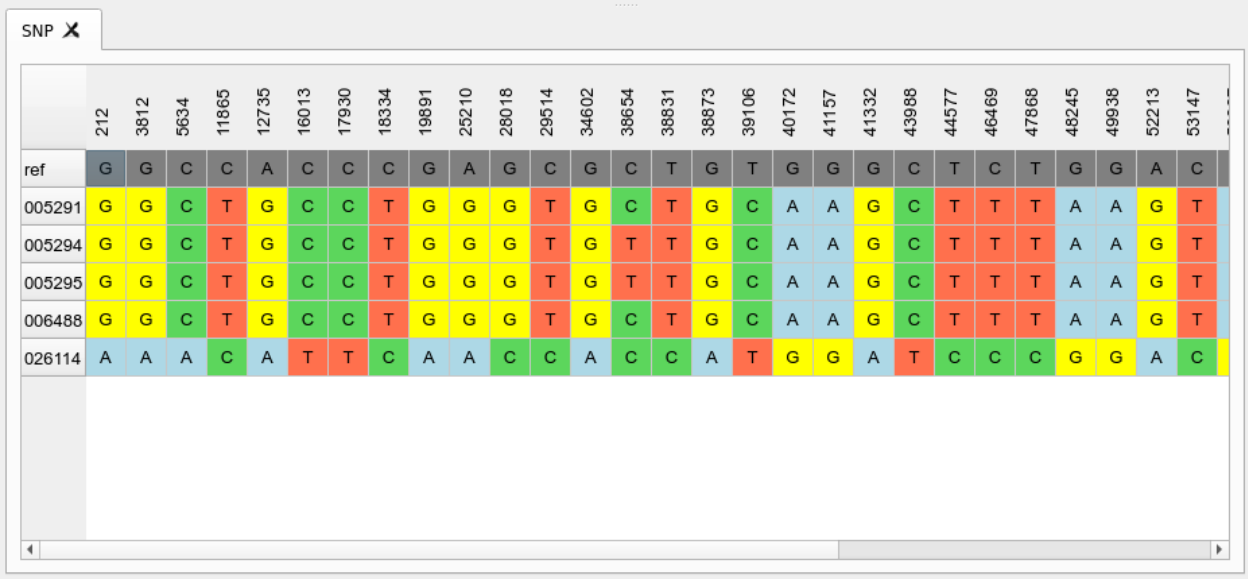
Fig. 1: Basic workflow steps.

6.3 Importing files

6.4 Viewing results

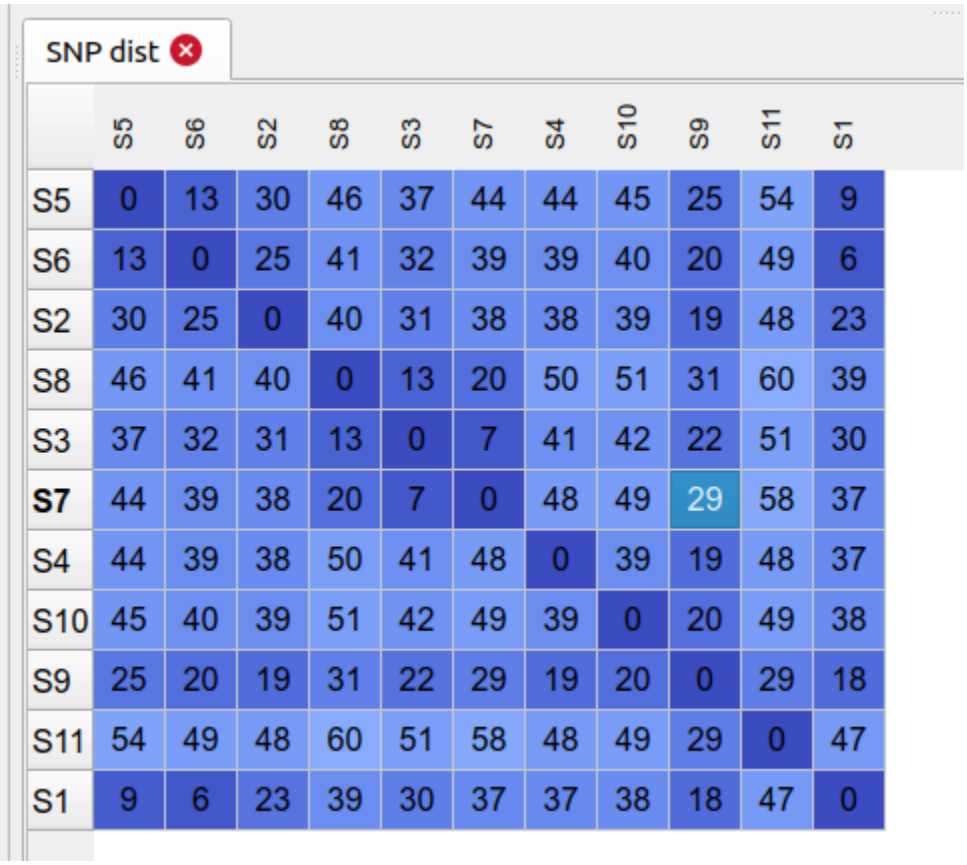
6.4.1 SNP table

This is a view of the SNPs in a table for all samples and each position. This is loaded from the core.txt file in results that is the product of filtered SNPs. This is what is used to make the final phylogeny. Below is shown the table with positions in the columns and each row is a sample. You can transpose or flip the table too.



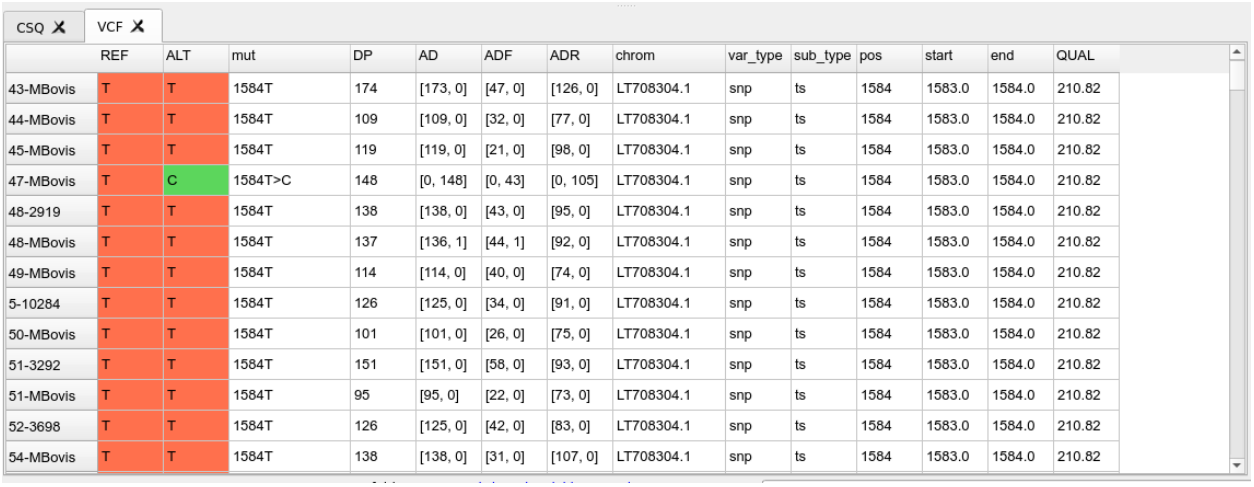
6.4.2 SNP distance table

This shows the distance matrix of samples derived from the core SNP alignment above. The samples can be sorted by their order in the phylogeny stored in the project.



6.4.3 VCF table

This lets you view the content of vcf files that are the product of variant calls. Normally useful for debugging errors that might be occurring or checking on the depth and quality values for a position/site. By default the filtered SNPs vcf will be displayed but you can load other vcf/bcf files from the file system.



CSQ VCF

	REF	ALT	mut	DP	AD	ADF	ADR	chrom	var_type	sub_type	pos	start	end	QUAL
43-MBovis	T	T	1584T	174	[173, 0]	[47, 0]	[126, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
44-MBovis	T	T	1584T	109	[109, 0]	[32, 0]	[77, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
45-MBovis	T	T	1584T	119	[119, 0]	[21, 0]	[98, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
47-MBovis	T	C	1584T>C	148	[0, 148]	[0, 43]	[0, 105]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
48-2919	T	T	1584T	138	[138, 0]	[43, 0]	[95, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
48-MBovis	T	T	1584T	137	[136, 1]	[44, 1]	[92, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
49-MBovis	T	T	1584T	114	[114, 0]	[40, 0]	[74, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
5-10284	T	T	1584T	126	[125, 0]	[34, 0]	[91, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
50-MBovis	T	T	1584T	101	[101, 0]	[26, 0]	[75, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
51-3292	T	T	1584T	151	[151, 0]	[58, 0]	[93, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
51-MBovis	T	T	1584T	95	[95, 0]	[22, 0]	[73, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
52-3698	T	T	1584T	126	[125, 0]	[42, 0]	[83, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82
54-MBovis	T	T	1584T	138	[138, 0]	[31, 0]	[107, 0]	LT708304.1	snp	ts	1584	1583.0	1584.0	210.82

6.4.4 CSQ table

This is a table showing the contents of the csq.tsv file that is calculated as from the *consequence calling* step. This is only present if we have provided a genbank annotation file when running the variant calling. This shows the effects of each identified SNP in terms of their amino acid changes in the annotated proteins along the genome.

6.5 Plotting from the tables

Some tables will allow you to make simple plots from numerical data.

6.6 Adding sample metadata

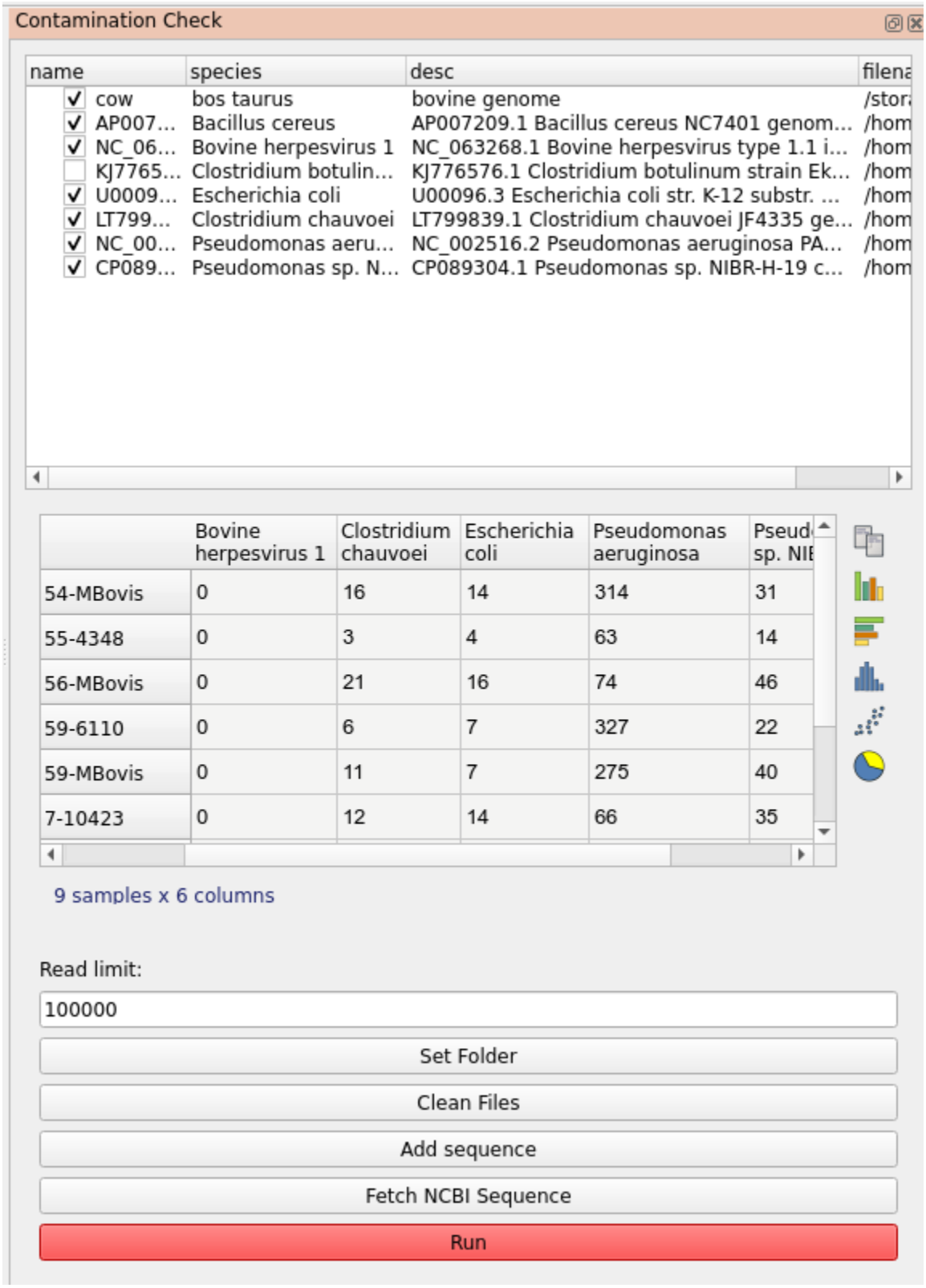
6.7 Checking for contamination

The program includes a plugin tool called Contamination Check for this purpose. It allows you to check for the presence of possible contaminants by aligning a subset of the reads against arbitrary genomes that you can select. A number of bacterial genomes are provided by default but you will often want to customise this. There are two ways to add a sequence:

- Find the sequence you want, download it and add from the local file system using the 'Add Sequence' button.
- You can download directly from NCBI using an accession number. To make it easier to find the sequence you want there is a link to the nucleotide search page accessible via the help menu. This will open the page inside the application in a browser tab.

When you add references the top list will be updated, you can deselect if you don't want to include them in the search. By default the output is written to the 'contam' folder in your results folder. Use *Clean Files* to remove the temporary bam files if they are taking too much space. Once you have the references to check against, select the samples in the main table you want to check and click run in the plugin dialog.

Results are displayed as a separate table in the plugin dialog. This will show the number of reads mapped to each reference. This gives a rough idea of the composition of our sample and whether it contains what we expect.



6.8 Create test data

This plugin allows you to simulate a set of read data based on a pre-defined phylogeny and reference genome. The purpose of this is to test the workflow against known data. It also provides a useful dataset to practice with.

6.9 File renaming

Batch file renaming

CHAPTER

SEVEN

M.BOVIS ANALYSIS

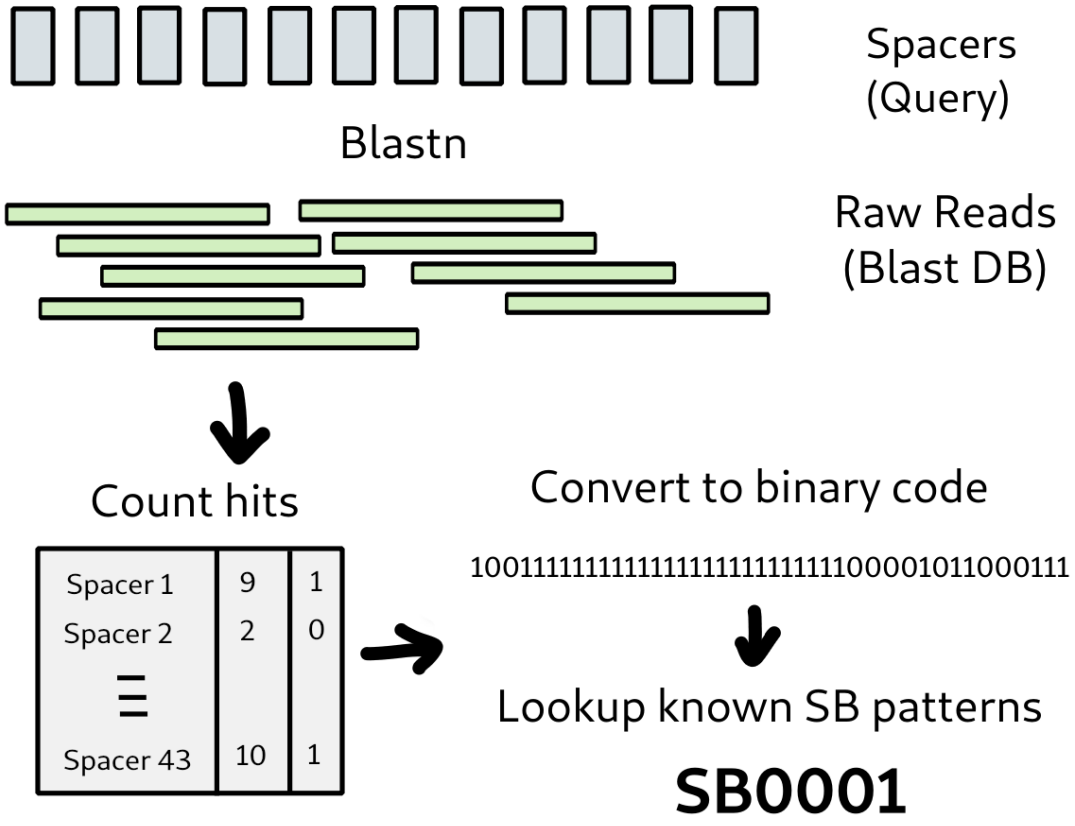
This tool was initially written for analysis of M.bovis isolates. It can however be used for any other bacterial or viral species for which this kind of workflow is appropriate. The tools documented here are specific to M.bovis.

7.1 Spoligotyping

Spoligotyping (spacer oligonucleotide typing) is a widely used genotyping method for M. tb (Mycobacterium tuberculosis species), which exploits the genetic diversity in the direct repeat (DR) locus in Mtb genome. Each DR region consists of several copies of the 36 bp DR sequence, which are interspersed with 34 bp to 41 bp non-repetitive spacers. A set of 43 unique spacer sequences is used to classify Mtb strains based on their presence or absence. This a molecular method traditionally conducted using a PCR-based or other method. Whole genome sequence is a far more sensitive method of phylogenetic identification of strains and makes other typing methods redundant if you have the whole sequence. It may be useful however to relate the sequence back to the spoligotype in some circumstances. It is not hard to calculate the spoligotype by analysis of the raw sequence reads (or an assembly).

About the method

7.2 Regions of difference


Spacers
(Query)
Blastn
Raw Reads
(Blast DB)
Count hits
Spacer 1
Spacer 2
Spacer 43
9
2
10
1
0
1
Convert to binary code
10011111111111111111111111111000010110000111
Lookup known SB patterns
SB0001

CHAPTER

EIGHT

FAQ

The run was stopped during execution, can it be resumed?

Yes, by default the program won't overwrite intermediate files when re-run. So just run it again. Make sure there are no old tmp.****.bam files in the mapped folder if an alignment got interrupted.

My sample files are not being parsed properly.

This may be because your sample names are unusual. The program extracts the unique sample names from the files by using the '_' symbol as delimeter. If your names differ you can supply a different delimeter with the labelsep option.

I added new files and tried to re-run but it failed.

This is because the samples don't match the previous variant call output. You might see a different number of samples warning. By default the results of mpileup are not overwritten as this is the slowest step. You should first delete the file raw.bcf in the output folder and run again.

I have more than 1000 samples and the bcftools mpileup step fails.

This is likely due to the limit on the number of files that can be opened at the same time. You can increase this limit on Linux using ulimit -n 2000 or whatever value you need up to 9999. Note that for many samples this step could take several days to run.

CHAPTER

NINE

SNIPGENIE

9.1 snipgenie package

9.1.1 Submodules

9.1.2 snipgenie.gui module

snipgenie GUI. Created Jan 2020 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

class snipgenie.gui.**App**(*filenames=[]*, *project=None*)

Bases: QMainWindow

GUI Application using PySide2 widgets

about()

add_dock(*widget*, *name*)

Add a dock widget

add_file(*filter='Fasta Files(*.fa *.fna *.fasta)'*, *path=None*)

Add a file to the config folders

add_gc_mean(*progress_callback*)

Get mean GC to indicate contamination

add_mapping_stats(*progress_callback*)

get mapping stats for all files and add to table

add_mask(*filename=None*)

Add mask bed file

add_mean_depth(*progress_callback*)

find mean depth for bam file

add_plugin_dock(*plugin*)

Add plugin as dock widget

add_read_lengths(*progress_callback*)

Get read lengths

add_recent_file(*fname*)

Add file to recent if not present

align_files(*progress_callback*)

Run gene annotation for input files. progress_callback: signal for indicating progress in gui

alignment_completed()

Alignment/calling completed

calling_completed()

check_contamination()

Blast to common contaminant sequences

check_fastq_table()

Update samples file to reflect table

check_files()

Check input files exist

check_heterozygosity()

Plot heterozygosity for each sample

check_missing_files()

Check folders for missing files

check_output_folder()

Check if we have an output dir

clean_up()

Clean up intermediate files

clear_plugins()

remove all open plugins

clear_tabs()

Clear tabbed panes

closeEvent(*event=None*)

Close main window

close_right_tab(*index*)

Close right tab

close_tab(*index*)

Close current tab

create_menu()

Create the menu bar for the application.

create_tool_bar()

Create main toolbar

csq_viewer()
Show CSQ table - output of bcftools csq

discover_plugins()
Discover available plugins

fastq_quality_report()
Make fastq quality report as pdf

get_fasta_reads()
Get a sample of reads for blasting

get_right_tabs()

get_selected()
Get selected rows of fastq table

get_tab_indices(*tab_widget*, *tab_name*)

get_tab_names()

get_tabs()

import_results_folder(*path*)
Import previously made results

load_fastq_files_dialog()
Load fastq files

load_fastq_folder_dialog()
Load fastq folder

load_fastq_table(*filenames*)
Append/Load fasta inputs into table

load_plugin(*plugin*)
Instantiate the plugin and show widget or run it

load_preset_genome(*seqname*, *gbfile*, *mask*, *ask*)

load_presets_menu(*ask=True*)
Add preset genomes to menu

load_project(*filename=None*)
Load project

load_project_dialog()
Load project

load_settings()
Load GUI settings

load_test()
Load test_files

make_phylo_tree(*progress_callback=None*, *method='raxml'*)
Make phylogenetic tree

mapping_stats(*row*)
Summary of a single fastq file

merge_meta_data()
Add sample meta data by merging with file table

missing_sites(*progress_callback=None*)
Find missing sites in each sample - useful for quality control

new_project(*ask=False*)
Clear all loaded inputs and results

online_documentation(*event=None*)
Open the online documentation

phylogeny_completed()

plot_dist_matrix()

preferences()
Preferences dialog

processing_completed()
Generic process completed

progress_fn(*msg*)

quality_summary(*row*)
Summary of a single sample, both files

quit()

rd_analysis(*progress_callback*)
Run RD analysis for MTBC species

rd_analysis_completed()
RD analysis completed

read_distributon(*row*)
get read length distribution

redirect_stdout()
redirect stdout

run()
Run all steps

run_threaded_process(*process*, *on_complete*)
Execute a function in the background with a worker

run_trimming(*progress_callback*)
Run quality and adapter trimming

sample_details(*row*)

save_plugin_data()
Save data for any plugins that need it

save_project()
Save project

save_project_dialog()
Save as project

save_settings()
Save GUI settings

set_annotation(*filename=None*)

set_mask(*filename*)

set_output_folder()
Set the output folder

set_reference(*filename=None*, *ask=True*)
Reset the reference sequence

set_style(*style='default'*)
Change interface style.

setup_gui()
Add all GUI elements

setup_paths()
Set paths to important files in proj folder

show_bam_viewer(*row*)
Show simple alignment view for a bam file

show_blast_url()

show_browser_tab(*link*, *name*)
Show web page in a tab

show_error_log()
Show log file contents

show_info(*msg*, *color=None*)

show_map()

show_nucldb_url()

show_phylogeny()
Show current tree

show_plugin(*plugin*)
Show plugin in dock or as window

show_recent_files()
Populate recent files menu

show_ref_annotation()
Show annotation in table

show_snpdist()
Show SNP distance matrix

show_variants()
Show the stored results from variant calling as tables

snp_alignment(*progress_callback=None*)
Make snp matrix from variant positions

snp_typing(*progress_callback*)
SNP typing for M.bovis

snp_viewer()
Show SNP table - output of core.txt

start_logging()
Error logging

staticMetaObject = <PySide2.QtCore.QMetaObject object>

tree_viewer()
Show tree viewer

update_labels()

update_mask()

update_plugin_menu()
Update plugins

update_ref_genome()
Update the ref genome labels

update_table(*new*)
Update table with changed rows

variant_calling(*progress_callback=None*)
Run variant calling for available bam files.

vcf_viewer()
Show VCF table

zoom_in()

zoom_out()

class snipgenie.gui.**AppOptions**(*parent=None*)
Bases: *BaseOptions*
Class to provide a dialog for global plot options

class snipgenie.gui.**Communicate**
Bases: QObject

newproj

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.gui.**StdoutRedirect**(**param*)
Bases: QObject

printOccur

start()

staticMetaObject = <PySide2.QtCore.QMetaObject object>

stop()

write(*s*, *color='black'*)

class snipgenie.gui.**Worker**(*fn*, **args*, ***kwargs*)

Bases: QRunnable

Worker thread for running background tasks.

run(*self*) → None

class snipgenie.gui.**WorkerSignals**

Bases: QObject

Defines the signals available from a running worker thread. Supported signals are: finished

No data

error

tuple (exctype, value, traceback.format_exc())

result

object data returned from processing, anything

error

finished

progress

result

staticMetaObject = <PySide2.QtCore.QMetaObject object>

snipgenie.gui.**main**()

Run the application

9.1.3 snipgenie.widgets module

Qt widgets for snpgenie. Created Jan 2020 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

class snipgenie.widgets.**BaseOptions**(*parent=None*, *opts={}*, *groups={}*)

Bases: object

Class to generate widget dialog for dict of options

apply()

applyOptions()

Set the plot kwd arguments from the widgets

increment(*key*, *inc*)

Increase the value of a widget

setWidgetValue(*key*, *value*)

Set a widget value

showDialog(*parent*, *wrap=2*, *section_wrap=2*, *style=None*)

Auto create tk vars, widgets for corresponding options and and return the frame

updateWidgets(*kwds*)

class snipgenie.widgets.**BasicDialog**(*parent*, *table*, *title=None*)

Bases: QDialog

Qdialog for table operations interfaces

apply()

Override this

close(*self*) → bool

copy_to_clipboard()

Copy result to clipboard

createButtons(*parent*)

createWidgets()

Create widgets - override this

staticMetaObject = <PySide2.QtCore.QMetaObject object>

update()

Update the original table

class snipgenie.widgets.**BrowserViewer**(*parent=None*)

Bases: QDialog

Browser widget

add_widgets()

Add widgets

load_page(*url*)

navigate_to_url()

method called by the line edit when return key is pressed getting url and converting it to QUrl object

staticMetaObject = <PySide2.QtCore.QMetaObject object>

update_urlbar(*q*)

method for updating url this method is called by the QWebEngineView object

zoom()

class snipgenie.widgets.**ColorButton**(**args*, *color=None*, ***kwargs*)

Bases: QPushButton

Custom Qt Widget to show a chosen color.

Left-clicking the button shows the color-chooser, while right-clicking resets the color to None (no-color).

color()

colorChanged

mousePressEvent(*self*, *e: PySide2.QtGui.QMouseEvent*) → None

onColorPicker()

Show color-picker dialog to select color. Qt will use the native dialog by default.

setColor(*color*)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**DynamicDialog**(*parent=None*, *options={}*, *groups=None*, *title='Dialog'*)

Bases: QDialog

Dynamic form using baseoptions

get_values()

Get the widget values

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**Editor**(*parent=None*, *fontsize=12*, ***kwargs*)

Bases: QTextEdit

contextMenuEvent(*self*, *e: PySide2.QtGui.QContextMenuEvent*) → None

insert(*txt*)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

zoom(*delta*)

class snipgenie.widgets.**FileViewer**(*parent=None*, *filename=None*)

Bases: QDialog

Sequence records features viewer

show_records(*recs*, *format='genbank'*)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**GraphicalBamViewer**(*parent=None*, *filename=None*)

Bases: QDialog

Alignment viewer with pylab

add_widgets()

Add widgets

load_data(*bam_file*, *ref_file*, *gb_file=None*, *vcf_file=None*)

Load reference seq and get contig/chrom names

redraw(*xstart=1*, *xend=2000*)

Plot the features

set_chrom(*chrom*)

Set the selected record which also updates the plot

staticMetaObject = <PySide2.QtCore.QMetaObject object>

update_chrom(*chrom=None*)

Update after chromosome selection changed

value_changed()

Callback for widgets

zoom_in()

Zoom in

zoom_out()

Zoom out

class snipgenie.widgets.**MergeDialog**(*parent*, *table*, *df2*, *title='Merge Tables'*)

Bases: *BasicDialog*

Dialog to melt table

apply()

Do the operation

createWidgets()

Create widgets

staticMetaObject = <PySide2.QtCore.QMetaObject object>

updateColumns()

class snipgenie.widgets.**MultipleInputDialog**(*parent*, *options=None*, *title='Input'*, *width=400*, *height=200*)

Bases: QDialog

Qdialog with multiple inputs

accept(*self*) → None

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**PlainTextEditor**(*parent=None*, ***kwargs*)

Bases: QPlainTextEdit

contextMenuEvent(*self*, *e: PySide2.QtGui.QContextMenuEvent*) → None

staticMetaObject = <PySide2.QtCore.QMetaObject object>

zoom(*delta*)

class snipgenie.widgets.**PlotViewer**(*parent=None*)

Bases: QWidget

matplotlib plots widget

clear()
Clear plot

create_figure(*fig=None*)
Create canvas and figure

redraw()

set_figure(*fig*)
Set the figure

staticMetaObject = <PySide2.QtCore.QMetaObject object>

zoom(*zoomin=True*)
Zoom in/out to plot by changing size of elements

class snipgenie.widgets.**PreferencesDialog**(*parent*, *options={}*)
Bases: QDialog

Preferences dialog from config parser options

apply()
Apply options to current table

createButtons(*parent*)

createWidgets(*options*)
create widgets

reset()
Reset to defaults

setDefaults()
Populate default kwds dict

staticMetaObject = <PySide2.QtCore.QMetaObject object>

updateWidgets(*kwds=None*)
Update widgets from stored or supplied kwds

class snipgenie.widgets.**SimpleBamViewer**(*parent=None*, *filename=None*)
Bases: QDialog

Sequence records features viewer using dna_features_viewer

add_widgets()
Add widgets

find_gene()
Go to selected gene if annotation present

goto()

load_data(*bam_file*, *ref_file*, *gb_file=None*, *vcf_file=None*)
Load reference seq and get contig/chrom names

next_page()

prev_page()

redraw(*xstart=1*)
Plot the features

set_chrom(*chrom*)
Set the selected record which also updates the plot

staticMetaObject = <PySide2.QtCore.QMetaObject object>

update_chrom(*chrom=None*)
Update after chromosome selection changed

value_changed()
Callback for widgets

zoom_in()
Zoom in

zoom_out()
Zoom out

class snipgenie.widgets.**TableViewer**(*parent=None*, *dataframe=None*, ***kwargs*)
Bases: QDialog

View row of data in table

setDataFrame(*dataframe*)

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**TextViewer**(*parent=None*, *text=''*, *width=200*, *height=400*, *title='Text'*)
Bases: QDialog

Plain text viewer

add_widgets()
Add widgets

staticMetaObject = <PySide2.QtCore.QMetaObject object>

class snipgenie.widgets.**ToolBar**(*table*, *parent=None*)
Bases: QWidget

Toolbar class

addButton(*name*, *function*, *icon*)

createButtons()

staticMetaObject = <PySide2.QtCore.QMetaObject object>

snipgenie.widgets.**addToolBarItems**(*toolbar*, *parent*, *items*)
Populate toolbar from dict of items

snipgenie.widgets.**dialogFromOptions**(*parent*, *opts*, *sections=None*, *wrap=2*, *section_wrap=4*, *style=None*)
Get Qt widgets dialog from a dictionary of options. :param opts: options dictionary :param sections: :param section_wrap: how many sections in one row :param style: stylesheet css if required

snipgenie.widgets.**getWidgetValues**(*widgets*)
Get values back from a set of widgets

snipgenie.widgets.**setWidgetValues**(*widgets*, *values*)
Set values for a set of widgets from a dict

9.1.4 snipgenie.tools module

Various methods for bacterial genomics. Created Nov 2019 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

snipgenie.tools.**bam_to_fastq**(*filename*)

bam to fastq using samtools

snipgenie.tools.**batch_iterator**(*iterator*, *batch_size*)

Returns lists of length batch_size.

This can be used on any iterator, for example to batch up SeqRecord objects from Bio.SeqIO.parse(...), or to batch Alignment objects from Bio.AlignIO.parse(...), or simply lines from a file handle.

This is a generator function, and it returns lists of the entries from the supplied iterator. Each list will have batch_size entries, although the final list may be shorter.

snipgenie.tools.**blast_fasta**(*database*, *filename*, ***kwargs*)

Blast a fasta file

snipgenie.tools.**blast_sequences**(*database*, *seqs*, *labels=None*, ***kwargs*)

Blast a set of sequences to a local or remote blast database :param database: local or remote blast db name

'nr', 'refseq_protein', 'pdb', 'swissprot' are valide remote dbs

Parameters

- **seqs** – sequences to query, list of strings or Bio.SeqRecords
- **labels** – list of id names for sequences, optional but recommended

Returns

pandas dataframe with top blast results

snipgenie.tools.**checkDict**(*d*)

Check a dict recursively for non serializable types

snipgenie.tools.**clustal_alignment**(*filename=None*, *seqs=None*, *command='clustalw'*)

Align 2 sequences with clustal

snipgenie.tools.**concat_seqrecords**(*recs*)

Join seqrecords together

snipgenie.tools.**core_alignment_from_vcf**(*vcf_file*, *callback=None*, *uninformative=False*, *missing=False*, *omit=None*)

Get core SNP site calls as sequences from a multi sample vcf file. :param vcf_file: multi-sample vcf (e.g. produced by app.variant_calling) :param uninformative: whether to include uninformative sites :param missing: whether to include sites with one or more missing samples (ie. no coverage) :param omit: list of samples to exclude if required

snipgenie.tools.**dataframe_to_fasta**(*df*, *seqkey='translation'*, *idkey='locus_tag'*, *descrkey='description'*, *outfile='out.faa'*)

Genbank features to fasta file

snipgenie.tools.**diffseqs**(*seq1*, *seq2*)

Diff two sequences

snipgenie.tools.**fasta_to_dataframe**(*infile*, *header_sep=None*, *key='name'*, *seqkey='sequence'*)

Get fasta proteins into dataframe

snipgenie.tools.**fastq_quality_report**(*filename*, *figsize=(7, 5)*, ***kwargs*)

Fastq quality plots

snipgenie.tools.**fastq_random_seqs**(*filename*, *size=50*)

Random sequences from fastq file. Requires pyfastx. Creates a fastq index which will be a large file.

snipgenie.tools.**fastq_to_dataframe**(*filename*, *size=5000*)

Convert fastq to dataframe. size: limit to the first reads of total size, use None to get all reads Returns: dataframe with reads

snipgenie.tools.**fastq_to_fasta**(*filename*, *out*, *size=1000*)

Convert fastq to fasta size: limit to the first reads of total size

snipgenie.tools.**fastq_to_rec**(*filename*, *size=50*)

Get reads from a fastq file :param size: limit

Returns: biopython seqrecords

snipgenie.tools.**features_summary**(*df*)

SeqFeatures dataframe summary

snipgenie.tools.**fetch_sra_reads**(*df*, *path*)

Download a set of reads from SRA using dataframe with runs

snipgenie.tools.**genbank_to_dataframe**(*infile*, *cds=False*)

Get genome records from a genbank file into a dataframe returns a dataframe with a row for each cds/entry

snipgenie.tools.**get_attributes**(*obj*)

Get non hidden and built-in type object attributes that can be persisted

snipgenie.tools.**get_blast_results**(*filename*)

Get blast results into dataframe. Assumes column names from local_blast method. :returns: dataframe

snipgenie.tools.**get_chrom**(*filename*)

Get chromosome name from fasta file

snipgenie.tools.**get_cmd**(*cmd*)

Get windows version of a command if required

snipgenie.tools.**get_fasta_length**(*filename*)

Get length of reference sequence

snipgenie.tools.**get_fastq_info**(*filename*)

snipgenie.tools.**get_fastq_read_lengths**(*filename*)

Return fastq read lengths

snipgenie.tools.**get_fastq_size**(*filename*)

Return fastq number of reads

snipgenie.tools.**get_gc**(*filename*, *limit=10000.0*)

snipgenie.tools.**get_mean_depth**(*bam_file*, *chrom=None*, *start=None*, *end=None*, *how='mean'*)

Get mean depth from bam file

snipgenie.tools.**get_sb_number**(*binary_str*)

Get SB number from binary pattern usinf database reference

snipgenie.tools.**get_snp_matrix**(*df*)

SNP matrix from multi sample vcf dataframe

snipgenie.tools.**get_spoligotype**(*filename*, *reads_limit=3000000*, *threshold=2*, *threads=4*)

Get mtb spoligotype from WGS reads

snipgenie.tools.**get_spoligotypes**(*samples*, *spo=None*)

Get spoligotypes for multiple M.bovis strains

snipgenie.tools.**get_subsample_reads**(*filename*, *outpath*, *reads=10000*)

Sub-sample a fastq file with first n reads. :param filename: input fastq.gz file :param outpath: output directory to save new file :param reads: how many reads to sample from start

snipgenie.tools.**get_unique_snps**(*names*, *df*, *present=True*)

Get snps unique to one or more samples from a SNP matrix. :param name: name of sample(s) :param df: snp matrix from app.get_aa_snp_matrix(csq) :param present: whether snp should be present/absent

snipgenie.tools.**get_vcf_samples**(*filename*)

Get list of samples in a vcf/bcf

snipgenie.tools.**gff_bcftools_format**(*in_file*, *out_file*)

Convert a genbank file to a GFF format that can be used in bcftools csq. see https://github.com/samtools/bcftools/blob/develop/doc/bcftools.txt#L1066-L1098. :param in_file: genbank file :param out_file: name of GFF file

snipgenie.tools.**gff_to_records**(*gff_file*)

Get features from gff file

snipgenie.tools.**gunzip**(*infile*, *outfile*)

Gunzip a file

snipgenie.tools.**kraken**(*file1*, *file2=''*, *dbname='STANDARD16'*, *threads=4*)

Run kraken2 on single/paired end fastq files

snipgenie.tools.**local_blast**(*database*, *query*, *output=None*, *maxseqs=50*, *evalue=0.001*, *compress=False*, *cmd='blastn'*, *threads=4*, *show_cmd=False*, ***kwargs*)

Blast a local database. :param database: local blast db name :param query: sequences to query, list of strings or Bio.SeqRecords

Returns

pandas dataframe with top blast results

snipgenie.tools.**make_blast_database**(*filename*, *dbtype='nucl'*)

Create a blast db from fasta file

snipgenie.tools.**move_files**(*files*, *path*)

snipgenie.tools.**normpdf**(*x*, *mean*, *sd*)

Normal distribution function

snipgenie.tools.**pdf_qc_reports**(*filenames*, *outfile='qc_report.pdf'*)

Save pdf reports of fastq file quality info

snipgenie.tools.**plot_fastq_gc_content**(*filename*, *ax=None*, *limit=50000*)

Plot fastq gc conent

snipgenie.tools.**plot_fastq_qualities**(*filename*, *ax=None*, *limit=10000*)

Plot fastq qualities for illumina reads.

snipgenie.tools.**records_to_dataframe**(*records*, *cds=False*, *nucl_seq=False*)

Get features from a biopython seq record object into a dataframe :param features: Bio SeqFeatures :param returns: a dataframe with a row for each cds/entry.

snipgenie.tools.**remote_blast**(*db*, *query*, *maxseqs=50*, *evalue=0.001*, ***kwargs*)

Remote blastp. :param query: fasta file with sequence to blast :param db: database to use - nr, refseq_protein, pdb, swissprot

snipgenie.tools.**resource_path**(*relative_path*)

Get absolute path to resource, works for dev and for PyInstaller

snipgenie.tools.**samtools_coverage**(*bam_file*)

Get coverage/depth stats from bam file

snipgenie.tools.**samtools_depth**(*bam_file*, *chrom=None*, *start=None*, *end=None*)

Get depth from bam file

snipgenie.tools.**samtools_flagstat**(*filename*)

Parse samtools flagstat output into dictionary

snipgenie.tools.**samtools_tview**(*bam_file*, *chrom*, *pos*, *width=200*, *ref=''*, *display='T'*)

View bam alignment with samtools

snipgenie.tools.**set_attributes**(*obj*, *data*)

Set attributes from a dict. Used for restoring settings in tables

snipgenie.tools.**snp_dist_matrix**(*aln*)

Get pairwise snps distances from biopython Multiple Sequence Alignment object. returns: pandas dataframe

snipgenie.tools.**trim_reads**(*filename1*, *filename2*, *outpath*, *quality=20*, *method='cutadapt'*, *threads=4*)

Trim adapters using cutadapt

snipgenie.tools.**trim_reads_default**(*filename*, *outfile*, *right_quality=35*)

Trim adapters - built in method

snipgenie.tools.**vcf_to_dataframe**(*vcf_file*)

Convert a multi sample vcf to dataframe. Records each samples FORMAT fields. :param vcf_file: input multi sample vcf

Returns: pandas DataFrame

9.1.5 snipgenie.app module

snipgenie methods for cmd line tool. Created Nov 2019 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warroanty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

class snipgenie.app.**Logger**(*logfile='log.dat'*)

Bases: `object`

flush()

write(*message*)

class snipgenie.app.**WorkFlow**(***kwargs*)

Bases: `object`

Class for implementing a prediction workflow from a set of options

run()

Run workflow

setup()

Setup main parameters

snipgenie.app.**align_reads**(*df*, *idx*, *outdir='mapped'*, *callback=None*, *aligner='bwa'*, *platform='illumina'*, *unmapped=None*, ***kwargs*)

Align multiple files. Requires a dataframe with a 'sample' column to indicate paired files grouping. If a trimmed column is present these files will align_reads instead of the raw ones. :param df: dataframe with sample names and filenames :param idx: index name :param outdir: output folder :param unmapped_dir: folder for unmapped files if required

snipgenie.app.**blast_contaminants**(*filename*, *limit=2000*, *random=False*, *pident=98*, *qcovs=90*)

Blast reads to contaminants database Returns: percentages of reads assigned to each species.

snipgenie.app.**check_platform**()

See if we are running in Windows

snipgenie.app.**check_samples_aligned**(*samples*, *outdir*)

Check how many samples already aligned

snipgenie.app.**check_samples_unique**(*samples*)

Check that sample names are unique

snipgenie.app.**clean_bam_files**(*samples*, *path*, *remove=False*)

Check if any bams in output not in samples and remove. Not used in workflow.

snipgenie.app.**copy_ref_genomes**()

Copy default ref genome files to config dir

snipgenie.app.**csq_call**(*ref*, *gff_file*, *vcf_file*, *csqout*)

Consequence calling

snipgenie.app.**fetch_binaries**()

Get windows binaries – windows only

snipgenie.app.**fetch_contam_file**()

Get contam sequences

snipgenie.app.**get_aa_snp_matrix**(*df*)

Get presence/absence matrix from csq calls table

snipgenie.app.**get_files_from_paths**(*paths*, *ext='*.f*q.gz'*, *filter_list=None*)

Get files in multiple paths. :param ext: wildcard for file types to parse eg. *.f*q.gz] :param filter_list: list of labels that should be present in the filenames, optional

snipgenie.app.**get_pivoted_samples**(*df*)

Get pivoted samples by pair, returns a table with one sample per row and filenames in separate columns.

snipgenie.app.**get_samples**(*filenames*, *sep='-'*, *index=0*)

Get sample pairs from list of files, usually fastq. This returns a dataframe of unique sample labels for the input and tries to recognise the paired files. :param sep: separator to split name on :param index: placement of label in split list, default 0

snipgenie.app.**get_samples_from_bam**(*filenames*, *sep='-'*, *index=0*)

Samples from bam files

snipgenie.app.**main**()

Run the application

snipgenie.app.**mapping_stats**(*samples*)

Get stats on mapping of samples

snipgenie.app.**mask_filter**(*vcf_file*, *mask_file*, *overwrite=False*, *outdir=None*)

Remove any masked sites using a bed file, overwrites input

snipgenie.app.**mpileup**(*bam_file*, *ref*, *out*, *overwrite=False*)

Run bcftools for single file.

snipgenie.app.**mpileup_multiprocess**(*bam_files*, *ref*, *outpath*, *threads=4*, *callback=None*)

Run mpileup in parallel over multiple files and make separate bcfs. Assumes alignment to a bacterial reference with a single chromosome.

snipgenie.app.**mpileup_parallel**(*bam_files*, *ref*, *outpath*, *threads=4*, *callback=None*, *tempdir=None*)

Run mpileup in over multiple regions with GNU parallel on linux or rush on Windows Separate bcf files are then joined together. Assumes alignment to a bacterial reference with a single chromosome.

snipgenie.app.**mpileup_region**(*region*, *out*, *bam_files*, *callback=None*)

Run bcftools for single region.

snipgenie.app.**overwrite_vcf**(*vcf_file*, *sites*, *outdir=None*)

Make a new vcf with subset of sites

snipgenie.app.**read_csq_file**(*filename*)

Read csq tsv outpt file into dataframe

snipgenie.app.**relabel_vcfheader**(*vcf_file*, *sample_file*)

Re-label samples in vcf header

snipgenie.app.**run_bamfiles**(*bam_files*, *ref*, *gff_file=None*, *mask=None*, *outdir='.'*, *threads=4*, *sep='_'*, *labelindex=0*, *samples=None*, ***kwargs*)

Run workflow with bam files from a previous sets of alignments. We can arbitrarily combine results from multiple other runs this way. kwargs are passed to variant_calling method. Should write a samples.txt file in the outdir if vcf header is to be relabelled. :param samples: dataframe of sample names, if not provided try to get from bam files

snipgenie.app.**site_proximity_filter**(*vcf_file*, *dist=10*, *overwrite=False*, *outdir=None*)

Remove any pairs of sites within dist of each other. :param vcf_file: input vcf file with positions to filter :param dist: distance threshold :param overwrite: whether to overwrite the vcf

snipgenie.app.**test_run**()

Test run

snipgenie.app.**trim_files**(*df*, *outpath*, *overwrite=False*, *threads=4*, *quality=30*)

Batch trim fastq files

snipgenie.app.**variant_calling**(*bam_files*, *ref*, *outpath*, *relabel=True*, *threads=4*, *callback=None*, *overwrite=False*, *filters=None*, *gff_file=None*, *mask=None*, *tempdir=None*, *custom_filters=False*, ***kwargs*)

Call variants with bcftools

snipgenie.app.**worker**(*args*)

snipgenie.app.**write_samples**(*df*, *path*)

Write out sample names only using dataframe from get_samples

9.1.6 snipgenie.aligners module

Aligner methods for bacterial genomics. Created Nov 2019 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

snipgenie.aligners.**bowtie_align**(*file1*, *file2*, *idx*, *out*, *unmapped=None*, *threads=2*, *overwrite=False*, *verbose=True*, *options=''*)

Map reads using bowtie

snipgenie.aligners.**build_bowtie_index**(*fastafile*, *path=None*)

Build a bowtie index :param fastafile: file input :param path: folder to place index files

snipgenie.aligners.**build_bwa_index**(*fastafile*, *path=None*, *show_cmd=True*, *overwrite=True*)

Build a bwa index

snipgenie.aligners.**build_subread_index**(*fastafile*)

Build an index for subread

snipgenie.aligners.**bwa_align**(*file1*, *file2*, *idx*, *out*, *threads=4*, *overwrite=False*, *options=''*, *filter=None*, *unmapped=None*)

Align reads to a reference with bwa. :param file1: fastq files :param file2: fastq files :param idx: bwa index name :param out: output bam file name :param options: extra command line options e.g. -k INT for seed length :param unmapped: path to file for unmapped reads if required

snipgenie.aligners.**minimap2_align**(*file1*, *file2*, *idx*, *out*, *platform='illumina'*, *threads=4*, *overwrite=False*)

Align illumina/ONT reads with minimap2

snipgenie.aligners.**subread_align**(*file1*, *file2*, *idx*, *out*, *threads=2*, *overwrite=False*, *verbose=True*)

Align reads with subread

9.1.7 snipgenie.plotting module

Plotting methods for snpgenie Created Jan 2020 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

snipgenie.plotting.**create_grid**(*gdf=None*, *bounds=None*, *n_cells=10*, *overlap=False*, *crs='EPSG:29902'*)

Create square grid that covers a geodataframe area or a fixed boundary with x-y coords returns: a GeoDataFrame of grid polygons

snipgenie.plotting.**create_hex_grid**(*gdf=None*, *bounds=None*, *n_cells=10*, *overlap=False*, *crs='EPSG:29902'*)

Hexagonal grid over geometry. See https://sabrinadchan.github.io/data-blog/building-a-hexagonal-cartogram.html

snipgenie.plotting.**display_igv**(*url='http://localhost:8888/files/'*, *ref_fasta=''*, *bams=[]*, *gff_file=None*, *vcf_file=None*)

Display IGV tracks in jupyter, requires the igv_jupyterlab package. Example usage:

bams = {'24':'results/mapped/24-MBovis.bam'} igv=display_igv(url='http://localhost:8888/files/', ref_fasta='Mbovis_AF212297.fa',

gff_file='results/Mbovis_AF212297.gb.gff', vcf_file='results/filtered.vcf.gz', bams=bams)

snipgenie.plotting.**draw_pie**(*vals*, *xpos*, *ypos*, *colors*, *size=500*, *ax=None*)

Draw a pie at a specific position on an mpl axis. Used to draw spatial pie charts on maps. :param vals: values for pie :param xpos: x coord :param ypos: y coord :param colors: colors of values :param size: size of pie chart

snipgenie.plotting.**gen_colors**(*cmap*, *n*, *reverse=False*)

Generates n distinct color from a given colormap. :param cmap: The name of the colormap you want to use.

Refer https://matplotlib.org/stable/tutorials/colors/colormaps.html to choose Suggestions: For Metallicity in Astrophysics: Use coolwarm, bwr, seismic in reverse For distinct objects: Use gnuplot, brg, jet,turbo.

Parameters

- **n** (*int*) – Number of colors you want from the cmap you entered.
- **reverse** (*bool*) – False by default. Set it to True if you want the cmap result to be reversed.

Returns
A list with hex values of colors.

Return type
colorlist(list)

Taken from the mycolorpy package by binodbhttr see also https://matplotlib.org/stable/tutorials/colors/colormaps.html

snipgenie.plotting.**get_bam_aln**(*bam_file*, *chr*, *start*, *end*, *group=False*)

Get all aligned reads from a sorted bam file for within the given coords

snipgenie.plotting.**get_chrom_from_bam**(*bam_file*)

Get first sequence name in a bam file

snipgenie.plotting.**get_color_mapping**(*df*, *col*, *cmap=None*, *seed=1*)

Get random color map for categorcical dataframe column

snipgenie.plotting.**get_coverage**(*bam_file*, *chr*, *start*, *end*)

Get coverage from bam file at specified region

snipgenie.plotting.**get_fasta_length**(*filename*, *key=None*)

Get length of reference sequence

snipgenie.plotting.**get_fasta_names**(*filename*)

Get names of fasta sequences

snipgenie.plotting.**get_fasta_sequence**(*filename*, *start*, *end*, *key=0*)

Get chunk of indexed fasta sequence at start/end points

snipgenie.plotting.**heatmap**(*df*, *cmap='gist_gray_r'*, *w=15*, *h=5*, *ax=None*)

Plot dataframe matrix

snipgenie.plotting.**make_legend**(*fig*, *colormap*, *loc=(1.05, 0.6)*, *title=''*, *fontsize=12*)

Make a figure legend wth provided color mapping

snipgenie.plotting.**plot_bam_alignment**(*bam_file*, *chr*, *xstart*, *xend*, *ystart=0*, *yend=100*, *rect_height=0.6*, *fill_color='gray'*, *ax=None*)

bam alignments plotter. :param bam_file: name of a sorted bam file :param start: start of range to show :param end: end of range

snipgenie.plotting.**plot_coverage**(*df*, *plot_width=800*, *plot_height=60*, *xaxis=True*, *ax=None*)

Plot a bam coverage dataframe returned from get_coverage :param df: dataframe of coverage data (from get_coverage) :param plot_width: width of plot :param xaxis: plot the x-axis ticks and labels

snipgenie.plotting.**plot_features**(*rec*, *ax*, *rows=3*, *xstart=0*, *xend=30000*)

snipgenie.plotting.**random_colors**(*n=10*, *seed=1*)

Generate random hex colors as list of length n.

snipgenie.plotting.**show_colors**(*colors*)

display a list of colors

9.1.8 snipgenie.trees module

Tree methods for bacterial phylogenetics, mostly using ete3. Created Nov 2019 Copyright (C) Damien Farrell

snipgenie.trees.**biopython_draw_tree**(*filename*)

snipgenie.trees.**color_leaves**(*t*, *colors*, *color_bg=False*)

snipgenie.trees.**colors_from_labels**(*df*, *name*, *group*)

Colors from dataframe columns for use with an ete3 tree drawing

snipgenie.trees.**convert_branch_lengths**(*treefile*, *outfile*, *snps*)

snipgenie.trees.**create_tree**(*filename=None*, *tree=None*, *ref=None*, *labelmap=None*, *colormap=None*, *color_bg=False*, *format=1*)

Draw a tree

snipgenie.trees.**delete_nodes**(*t*, *names*)

snipgenie.trees.**format_nodes**(*t*)

snipgenie.trees.**get_clusters**(*tree*)

Get snp clusters from newick tree using TreeCluster.py

snipgenie.trees.**get_colormap**(*values*)

snipgenie.trees.**remove_nodes**(*tree*, *names*)

snipgenie.trees.**remove_tiplabels**(*t*)

snipgenie.trees.**run_RAXML**(*infile*, *name='variants'*, *threads=8*, *bootstraps=100*, *outpath='.'*)

Run Raxml pthreads. :returns: name of .tree file.

snipgenie.trees.**run_fasttree**(*infile*, *outpath*, *bootstraps=100*)

Run fasttree

snipgenie.trees.**run_treecluster**(*f*, *threshold*, *method='max_clade'*)

Run treecluster on a newick tree. Clustering Method (options: avg_clade, length, length_clade, max, max_clade, med_clade, root_dist, single_linkage_clade) (default: max_clade)

see https://github.com/niemasd/TreeCluster

snipgenie.trees.**set_nodesize**(*t*, *size=12*)

Change the node size

snipgenie.trees.**set_tiplabels**(*t*, *labelmap*)

snipgenie.trees.**toytree_draw**(*tre*, *meta*, *labelcol*, *colorcol*)

Draw colored tree with toytree

9.1.9 snipgenie.simulate module

Simulate reads Created Sep 2022 Copyright (C) Damien Farrell

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

`snipgenie.simulate.`**`artificial_fastq_generator`**(*ref*, *outfile*, *cmp=100*)

Generate reads from reference

`snipgenie.simulate.`**`generate_fastqs`**(*infile*, *outpath*, *reads=100000.0*, *overwrite=False*)

Make multiple fastqs

`snipgenie.simulate.`**`run_phastsim`**(*path*, *ref*, *newick*)

Run phastsim

9.1.10 Module contents

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z