MixviR_v3.3.2

Mike Sovic

2022-04-15

Contents

Introduction **1**

Quick Start **1**
Inputs . . . 1
Typical Workflow . . . 3

Full Documentation **3**
Inputs . . . 3
Primary Functions . . . 5

Cautions/Important Considerations **11**

Example Analyses **12**
Example Data . . . 12
Example Analysis 1- No Target Mutations In `call_mutations()` . . . 12
Example Analysis 2 - Target Mutations Included In `call_mutations()` . . . 18

Getting Help **19**

References **19**

Introduction

MixviR is a package designed to aid in exploring and visualizing genomic and amino-acid level variation obtained from microbial high-throughput sequencing samples, including samples that contain mixtures of genotypes. The package was originally written to detect and estimate relative frequencies of SARS-CoV-2 lineages (variants) in samples obtained from environmental sources, but can be applied to any microbial taxon.

Quick Start

Inputs

Required

1. Sample VCFs - one for each sample to be analyzed. Must contain "DP" and "AD" in FORMAT field.
2. Reference genome file (fasta; remove spaces and underscores '_' from chromosome names).
3. File (bed format) defining translated regions of the genome. 6 columns: chromosome, feature start position, feature end position, feature name, score (not used - can enter "."), and strand. Column names should not be included. Ensure that chromosome names in this file match those in the fasta reference exactly.

**Pre-constructed reference information is available for SARS-CoV-2 (Covid-19) and can be specified as an argument to `call_mutations()`. In that case, only the sample VCFs are required.

Optional

Lineage-Associated Mutations File

The lineage-associated mutations (*lineage.muts*) file (csv) provides mutations/amino acid substitutions associated with lineages of interest. Requires columns named "Gene", "Mutation", and "Lineage". Two additional columns can be included named "Chr" and "Pos". If these are included, make sure the chromosome names match those in the reference and bed files (no spaces, remove underscores). The file (including the optional columnmns) should look like...

```
Gene,Mutation,Lineage,Chr,Pos
N,R203K,Alpha,NC_045512.2,28881
ORF1a,T1001I,Alpha,NC_045512.2,NA
S,D614G,Alpha,NC_045512.2,23403
S,A570D,Alpha,NC_045512.2,23271
S,D614G,Delta,NC_045512.2,23403
S,P681R,Delta,NC_045512.2,23604
ORF1b,P314L,Delta,NC_045512.2,14408
M,I82T,Delta,NC_045512.2,NA
S,T19R,Delta,NC_045512.2,21618
S,L452R,Delta,NC_045512.2,22917
```

Figure 1: Example lineage-associated mutation file.

Location/Date File

The location/date (*dates*) file (csv) associates dates and locations with specific samples for cases where samples are obtained at various time points at one or more locations. Contains columns "SAMP_NAME", "LOCATION", and "DATE". Dates are given in *mmddyyyy* format, and the file should look like...

```
SAMP_NAME,LOCATION,DATE
AA-01.vcf,AA,10312021
AA-02.vcf,AA,11072021
AA-03.vcf,AA,11282021
AA-04.vcf,AA,12052021
AB-01.vcf,AB,10312021
AB-02.vcf,AB,11072021
AB-03.vcf,AB,11142021
AB-04.vcf,AB,11282021
AB-05.vcf,AB,12052021
```

Figure 2: Example location/date file.

Typical Workflow

Run `call_mutations()` to identify mutations in the sample(s). Required arguments:

- *sample.dir* (contains all vcfs to be analyzed; don't include any other files in this directory)
- *fasta.genome* & *bed* OR *reference*

Run `explore_mutations()` to interactively visualize results. Include one or both of the following if available:

- *lineage.muts*
- *dates*

Run `estimate_lineages()` to generate a table with summarized results for each sample. Required argument:

- *lineage.muts*

Full Documentation

Inputs

Required

There are three required inputs for running *MixviR* in its most basic form...

Sample Data

Sample data are provided as variant call format (vcf) files - one for each sample to be analyzed. These vcf files are expected to contain the "DP" and "AD" flags in the FORMAT field. *bcftools* (mpileup/call; Danecek et al 2021) and the GATK (DePristo et al 2011) offer two widely-used workflows that can generate these vcf files. Below is some example bash code I've used for generating these files (most arguments can be customized as needed - adding the FORMAT/AD AND FORMAT/DP fields is important)...

bcftools example

```
[*path to bcftools*]/bcftools mpileup -f *fasta reference* -d 4000 -q 60 -Q 30 -L 4500 /
--ff UNMAP,SECONDARY,QCFAIL /
-a FORMAT/AD,FORMAT/ADF,FORMAT/ADR,FORMAT/DP,INFO/AD,INFO/ADF,INFO/ADR /
*input.bam* | [*path to bcftools*]/bcftools call -m -A -Ov -o out_temp.vcf

[*path to bcftools*]/bcftools norm out_temp.vcf -c w -f *fasta reference* /
-m -both -Ov -o out.vcf
```

Reference Files

The other two necessary inputs define the reference information for the taxon of interest. They include...

1. A fasta-formatted reference genome file (remove spaces and underscores '_' from chromosome names).
2. An associated bed-formatted file that defines the translated regions of the genome (ORFs/genes). This bed file should be tab delimited with 6 columns: chromosome, feature start position, feature end position, feature name, score (not used - can enter "."), and strand (+/-). Column names should not be included. Ensure that the chromosome names in this file match those in the reference fasta file. An example of this file is...

```
NC_045512.2 266 13483   ORF1a   .   +
NC_045512.2 13468   21555   ORF1b   .   +
NC_045512.2 21563   25384   S   .   +
NC_045512.2 25393   26220   ORF3a   .   +
NC_045512.2 26245   26472   E   .   +
NC_045512.2 26523   27191   M   .   +
NC_045512.2 27202   27387   ORF6    .   +
NC_045512.2 27394   27759   ORF7a   .   +
NC_045512.2 27756   27887   ORF7b   .   +
NC_045512.2 27894   28259   ORF8    .   +
NC_045512.2 28274   29533   N   .   +
NC_045512.2 29558   29674   ORF10   .   +
```

Figure 3: Example features (bed) input file.

Pre-constructed reference information is available for SARS_CoV-2 (Covid-19), so only sample vcf files are necessary in that case.

Optional

There are two optional input files...

Lineage-associated mutations

The lineage-associated mutations file (*lineage.muts*) provides mutations associated with lineages/groups of interest. It requires columns named "Gene", "Mutation", and "Lineage". Two additional columns can be included named "Chr" and "Pos". If these additional columns are included, make sure the chromosome names match those in the reference and bed input files (including no spaces or underscores). The file (including the optional columnns) should look like...

```
Gene,Mutation,Lineage,Chr,Pos
N,R203K,Alpha,NC_045512.2,28881
ORF1a,T1001I,Alpha,NC_045512.2,NA
S,D614G,Alpha,NC_045512.2,23403
S,A570D,Alpha,NC_045512.2,23271
S,D614G,Delta,NC_045512.2,23403
S,P681R,Delta,NC_045512.2,23604
ORF1b,P314L,Delta,NC_045512.2,14408
M,I82T,Delta,NC_045512.2,NA
S,T19R,Delta,NC_045512.2,21618
S,L452R,Delta,NC_045512.2,22917
```

Figure 4: Example lineage-associated mutation file.

It's important that the syntax of the mutations in this file matches that used by *MixviR* - see details on naming conventions under the sections "SNP-Based Mutation Identification"," "Indel-Based Mutation Identification", and "Nonsense Mutation Identification" below. The "Chr" and "Pos" columns store the chromosome and associated genomic position giving rise to the mutation. These columns are needed if, as part of the results,

you want the program to return the sequencing depth at the position when the mutation is not observed for a sample. This information could be relevant in determining whether a target mutation not observed in a sample is absent because it doesn't occur in the sample, or because the sequencing coverage at that position is insufficient. If these columns are included, make sure the chromosome names match those in the reference genome.

Location/Date File

In cases where samples are taken from the same location at different time points, the temporal information can be included by providing a csv location/date file (*dates*) that associates the sample dates and locations with each unique sample name. The file should contain columns named "SAMP_NAME", "LOCATION", and "DATE", and should look like...

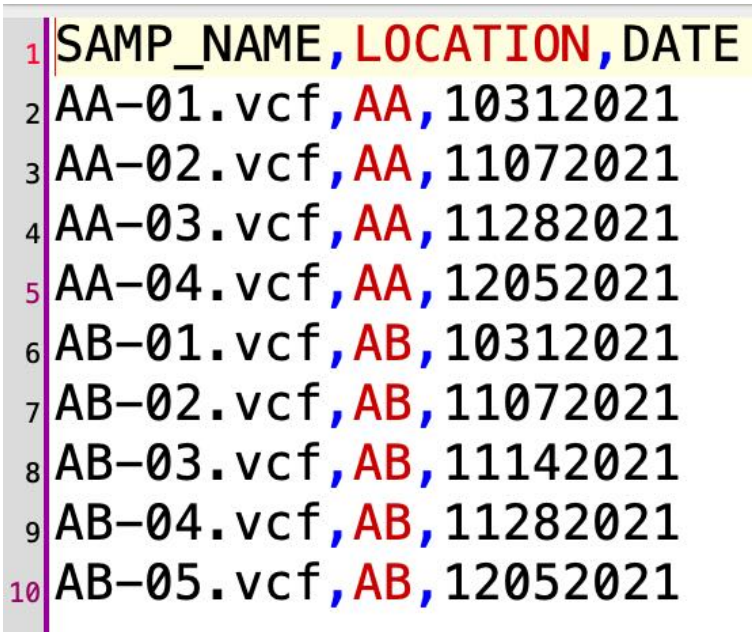


Figure 5: Example location/date file.

Primary Functions

`call_mutations()`

Overview

- **Description:** The `call_mutations()` function reads in the variant calls from the vcf file, translates the associated amino acids (for mutations within genes), and indentifies any variants relative to the reference.
- **Common Options:** Required options to `call_mutations()` include *sample.dir*, which is the path to a directory storing one or more vcf files (one for each sample to analyze). There should be no other files in this directory. Also required is information on the reference genome, which can be passed using the combination of *fasta.genome* and *bed* options. Alternatively, if working with SARS-CoV-2, the *reference* option can be set to "Wuhan" to use a pre-formatted reference. To report information (primarily sequencing depth) on all mutations of interest and not just those that are observed in the samples, the *write.all.targets* option can be set to TRUE, and the lineage-associated mutations file including the optional "Chr" and "Pos" columns (see above) must be specified with *lineage.muts*.
- **Output:** Mutation data are written to an object named *samp_mutations* that's stored in the global environment and that by default serves as the primary input for the `explore_mutations()` function.

This object can also be written to a file (see *write.mut.table*). By default, this data frame contains the columns SAMP_NAME, CHR, POS, ALT_ID, AF, DP, though a number of additional columns can be included (see "Output" section below, and *out.cols* argument to `call_mutations()`).

Methods

The `call_mutations()` function uses one or more vcf files as input, along with a *MixviR* reference object (data frame) and creates a data frame in the global environment that stores all mutations identified in the sample(s), along with a customizable set of associated information about each mutation. This data frame (table) can be written to a file, and/or used as default input to the `explore_mutations()` or `estimate_lineages()` functions.

`call_mutations()` first obtains the *MixviR* reference object (Fig 6), which is created as part of the run if the *fasta.genome* and *bed* options are provided. Alternatively, if analyzing SARS-CoV-2, the reference option can be set to "Wuhan" and a pre-constructed reference will be used. In the case of overlapping genes, positions will be duplicated in this reference object, with a separate entry for each gene the nucleotide position is associated with.

CHR	POS	REF_BASE	GENE	STRAND	REF_CODON	REF_AA	GENE_AA_POS	REF_IDENT	GENE_BASE_NUM	CODON_POSITION
NC045512.2	1	A	non-genic	.	*NA*	*NA*	*NA*	1A	*NA*	*NA*
NC045512.2	2	T	non-genic	.	*NA*	*NA*	*NA*	2T	*NA*	*NA*
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
NC045512.2	23402	G	S	+	GAT	D	614	S_D614	1840	1
NC045512.2	23403	A	S	+	GAT	D	614	S_D614	1841	2
NC045512.2	23404	T	S	+	GAT	D	614	S_D614	1842	3
NC045512.2	23405	G	S	+	GTT	V	615	S_V615	1843	1
NC045512.2	23406	T	S	+	GTT	V	615	S_V615	1844	2
NC045512.2	23407	T	S	+	GTT	V	615	S_V615	1845	3

Figure 6: Example of the initial MixviR reference object. Two non-genic positions and 6 genic positions, representing 2 amino acids, are shown.

MixviR then reads in the set of files (vcf or csv) to be analyzed. These should be the only files stored in the directory given by the *sample.dir* option. In most cases, samples will be provided in variant call format (vcf). These vcf files need to include the DP and AD flags in the FORMAT field. Relevant information from each vcf file is extracted with functionality from *vcfR* (Knaus and Grünwald, 2017). If the *write.all.targets* option will be used to report sequencing depths for genomic positions associated with a priori-defined mutations that don't occur in the sample, all positions should be included in the input vcf file(s). Otherwise, only variant positions need to be included.

MixviR loops over the set of input files, sequentially calling mutations from each and appending them to a master data frame that stores all mutations. The process of calling mutations for each sample includes several steps. The overall sequencing depths at each position in the input file are first added to the reference object (Fig. 7, column 'DP'; note that all objects shown in Figs 7-10 are temporary objects created during a *MixviR* analysis and are not directly available to the user). Depths are 'NA' for any positions not in the vcf input file.

CHR	POS	REF_BASE	GENE	STRAND	REF_CODON	REF_AA	GENE_AA_POS	REF_IDENT	GENE_BASE_NUM	CODON_POSITION	DP
NC045512.2	1	A	non-genic	.	NA	NA	NA	1A	NA	NA	NA
NC045512.2	2	T	non-genic	.	NA	NA	NA	2T	NA	NA	NA
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
NC045512.2	23402	G	S	+	GAT	D	614	S_D614	1840	1	3680
NC045512.2	23403	A	S	+	GAT	D	614	S_D614	1841	2	3666
NC045512.2	23404	T	S	+	GAT	D	614	S_D614	1842	3	3424
NC045512.2	23405	G	S	+	GTT	V	615	S_V615	1843	1	3751
NC045512.2	23406	T	S	+	GTT	V	615	S_V615	1844	2	3463
NC045512.2	23407	T	S	+	GTT	V	615	S_V615	1845	3	3543

Figure 7: MixviR reference object with sequencing depths for the sample being analyzed added in the DP column. NA in this column indicates the position was not included in the input vcf file.

Variable sites in the sample (sites with an entry in the ALT column of the input file) are then extracted, the frequency of the ALT allele (AF) is calculated for each, and the set of variants is filtered to retain those that exceed the value of *min.alt.freq*. This filtering step is intended to remove low-frequency sequencing noise and/or PCR artifacts from the data. Deletions are represented as strings >1 bp in the REF column (Fig 8; position 11282 is a 9-bp deletion) and insertions are represented as strings >1 bp in the ALT column (Fig 8; 1 bp insertion at position 19983).

CHR	POS	REF	ALT	ALT_COUNT	DP	AF	chr_pos
NC_045512.2	10029	C	T	2728	2730	0.99926740	NC_045512.2_10029
NC_045512.2	11282	AGTTTGTCTG	A	1714	3966	0.43217347	NC_045512.2_11282
NC_045512.2	11537	A	G	2152	3542	0.60756635	NC_045512.2_11537
NC_045512.2	19983	C	CT	349	3919	0.08905333	NC_045512.2_19983
NC_045512.2	21618	C	G	548	3620	0.15138122	NC_045512.2_21618

Figure 8: Example set of variants from a sample being analyzed after filtering for the default minimum allele frequency of 1%. Deletions have length >1bp in the REF column, while insertions have length >1bp in the ALT column. Variants are merged with the reference to create a sample genome that is translated to identify amino acid changes.

Next, *MixviR* checks the set of sample variants to determine if there are any genomic positions where more than one unique variant occurs in the (potentially mixed) sample. If not, mutations are characterized in two steps, first by identifying those based on single nucleotide polymorphism (SNP) variation, and subsequently by identifying mutations arising from insertions or deletions. Each of these steps is described below. In the event one or more positions have multiple mutations, the duplicated sites are first removed and stored. Mutations are initially called from the first set of variants (all variants without duplicated positions and the first variant at each duplicated site), and the process is iterated on the stored set of duplicated variants until no duplicated sites remain.

SNP-Based Mutation Identification

For mutations based on SNP variation, the full set of variants identified in the sample is filtered to retain just SNP-based variants (single nucleotide changes, no indels). These are then joined with the reference object, replacing the reference base with any alternate alleles observed in the sample to create a "sample genome" (in other words, the reference genome with SNP variants substituted in to their respective position). This updated sample genome is then translated using functionality from *Biostrings* (Pagès et al 2019) to get the

sample amino acids. Mutations resulting in amino acid substitutions are retained and named in the form "S_D614G", which would indicate a substitution of 'G' for the original 'D' at amino acid position 614 of the 'S' gene. Subsequently, all SNP variants that don't result in an amino acid change, including synonymous changes or variants outside of genes, are identified and named in the form Chr1_500A->T, which would represent a mutation from 'A' to 'T' at nucleotide position 500 of Chromosome 1. The example below (Fig 9) shows one SNP in a non-genic region, one synonymous and one non-synonymous SNP-based mutation in the 'ORF1a' gene, and one synonymous and one non-synonymous SNP-based mutation in the 'S' gene.

CHR	POS	REF_BASE	GENE	STRAND	REF_CODON	REF_AA	GENE_AA_POS	REF_IDENT	REF	ALT	AF	ALT_COUNT	samp_codon	samp_AA	samp_identity	DP
NC045512.2	241	C	non-genic	.	NA	NA	NA	241C	C	T	0.9975021	2396	NA	NA	NC045512.2_241C->T	2402
NC045512.2	2470	C	ORF1a	+	CCC	A	735	ORF1a_A735	C	T	0.7949913	1365	CCT	A	NC045512.2_2470C->T	1717
NC045512.2	2832	A	ORF1a	+	AAG	K	856	ORF1a_K856	A	G	0.7212714	885	AGG	R	ORF1a_K856R	1227
NC045512.2	22912	T	S	+	AAT	N	450	S_N450	T	C	0.2817109	191	AAC	N	NC045512.2_22912T->C	678
NC045512.2	23403	A	S	+	GAT	D	614	S_D614	A	G	1.0000000	3666	GGT	G	S_D614G	3666

Figure 9: Example set of SNP-based mutations. Synonymous and non-synonymous mutations in two different genes, and a mutation in a non-genic region, are included.

Indel-Based Mutation Identification

Indel-based mutation identification is performed separately for insertions and deletions, and for each, separately for in-frame and frameshift indels (defined as indels for which the number of nucleotides gained or lost is an even multiple of 3 or not, respectively, regardless of whether the event occurred in a gene). *MixviR* uses the following rules for naming indels...

In-frame deletions

In-frame deletions (even multiples of 3bp) are indicated with 'del'.

Examples: S_del144/144 (genic); ORF1a_del3675/3677 (genic); NC-045512.2_del23121/3bp (non-genic)

- *Genic:* Deletions starting at a first codon position of a gene are named starting with the first AA deleted and ending with the last. Those that start at a 2nd or 3rd codon position are named with the downstream (3')-most AA affected in the case of deletion of a single AA, or, if multiple AA's are deleted, naming starts with the AA associated with the first full codon deleted, and extends to the last codon/AA affected. Below is an example of the latter scenario from the S gene of SARS-CoV-2, in which 6 bases (AGTTCA) are deleted from positions 22029-22034 (beginning at codon position 2 of amino acid 156 of the S gene).

  The original sequence, broken into its codons and starting at position 22025, is...

  Ref Sequence: AGT GAG TTC AGA GTT

  Ref Amino Acids: S E F R V

  Ref Amino Acid Position: 155 156 157 158 159

  The corresponding deleted sequence is...

  Deletion Sequence: AGT GGA GTT

  Deletion Amino Acids: S G V

  There is some ambiguity regarding how to name this event. It could be named "S_del156/157", with a resulting substitution of "R158G". Alternatively, it could be named "S_del157/158" with a resulting substitution of "E156G". As described above, *MixviR* numbers the deletion beginning with the amino acid corresponding to the first full codon deleted - in this case "S_del157/158", and it does not call the substitution, as both the deletion and substitution represent just one mutational (evolutionary) event.

- *Non-genic:* In-frame deletions occurring outside of genes are named with the chromosome, position (nucleotide) along the chromosome of the first deleted base, and the length of the deletion in bp.

Out-of-frame deletions

Out-of-frame deletions (not even multiples of 3bp deleted) are indicated with 'Fdel'.

Examples: ORF1a_Fdel2454/7bp (genic); NC-045512.2_del23121/2bp (non-genic)

- *Genic:* Named with the amino acid position within the gene where the first base is deleted. This is followed by the length of the deletion in bp.
- *Non-genic:* Named with the chromosome, position (nucleotide) along the chromosome of the first deleted base, and the length of the deletion in bp.

In-frame insertions

In-frame insertions (even multiples of 3bp) are indicated with 'ins'.

Examples: S_ins214/216 (genic); NC-045512.2_ins23121/3bp (non-genic)

- *Genic:* Named with the amino acid position within the gene where the first base is inserted, followed by the (new) position of the last amino acid inserted.
- *Non-genic:* Named with the chromosome, position (nucleotide) along the chromosome of the first inserted base, and the length of the insertion in bp.

Out-of-frame insertions

Examples: S_Fins649/1bp (genic); NC-045512.2_Fins23121/1bp (non-genic)

Out-of-frame insertions (not even multiples of 3bp deleted) are indicated with 'Fins'.

- *Genic:* Named with the amino acid position within the gene where the first base is inserted. This is followed by the length of the insertion in bp.
- *Non-genic:* Named with the chromosome, position (nucleotide) along the chromosome of the first inserted base, and the length of the insertion in bp.

Indel examples are given in Figure 10, which shows a one-bp deletion within *ORF1a* at position 4947 of chromosome NC045512.2, a deletion of amino acid positions 3674-3676 in the *ORF1a* gene, a 1bp insertion within amino acid position 2173 of *ORF1b*, and a deletion of amino acid position 212 of the *S* gene.

CHR	POS	REF_BASE	GENE	STRAND	REF_CODON	REF_AA	GENE_AA_POS	REF_IDENT	REF	ALT	AF	ALT_COUNT	samp_codon	samp_AA	samp_identity	DP
NC045512.2	4947	C	ORF1a	+	TCT	S	1561	ORF1a_S1561	CT	C	0.06193548	240	NA	NA	ORF1a_Fdel1561/1bp	3875
NC045512.2	11282	A	ORF1a	+	AGT	S	3674	ORF1a_S3673	AGTTTGTCTG	A	0.43217347	1714	NA	NA	ORF1a_del3674/3676	3966
NC045512.2	19983	C	ORF1b	+	GTC	V	2173	ORF1b_V2172	C	CT	0.08905333	349	NA	NA	ORF1b_Fins2173/1bp	3919
NC045512.2	22193	A	S	+	AAT	N	212	S_N211	AATT	A	0.28109772	1096	NA	NA	S_del212/212	3899

Figure 10: Example insertions and deletions called from a sample during a MixviR anaysis. Deletions are indicated with 'del' in the name, and insertions with 'ins' - see samp_identity column for examples.

Nonsense Mutation Identification

Nonsense mutations resulting in a premature stop codon are designated with an asterisk (i.e. ORF1a_R718*).

Output

The full set of mutations identified (SNP and indel-based) are joined to create a data frame that serves as the output returned by `call_mutations()`. Any of the following columns can be included in this data frame (defined by the *out.cols* option):

- CHR: Chromosome.
- POS: Genomic position along the chromosome.
- GENE: Gene, if site is associated with a gene, otherwise listed as "non-genic".
- STRAND: If part of a gene, strand the gene is on (+, -, .).
- REF_CODON: If in a gene, reference codon the position is part of.

- REF_AA: If in a gene, identity of the reference amino acid the position is associated with.
- GENE_AA_POSITION: If in a gene, amino acid position within the gene the nucleotide position is associated with.
- REF: Reference nucleotide, or if representing a deletion, the reference nucleotide plus deleted bases.
- ALT: Alternate nucleotide, or if representing an insertion, the reference nucleotide plus inserted bases.
- AF: Frequency of the alternate allele (mutation) in the sample. Calculated by dividing ALT_COUNT by DP.
- ALT_COUNT: number of sequencing reads associated with the alternate (mutant) allele in the sample.
- SAMP_CODON: If part of a gene, the codon arising from the alternate (mutant) allele in the sample, otherwise 'NA'.
- SAMP_AA: If part of a gene, the amino acid arising from the alternate (mutant) allele in the sample, otherwise 'NA'.
- ALT_ID: Identifier of the mutation in the sample.
- DP: Total sequencing depth at the genomic position associated with the mutation.
- SAMP_NAME: Sample name, which is the name of the input file, unless the name.sep option is used in call_mutations() to retain just a portion of the name.
- TYPE: The type of mutation identified. One of SNP-Syn (synonymous SNP within a gene), SNP-Nsyn (non-synonymous SNP within a gene), "SNP-Nongenic" (SNP outside of a genic region), Indel-Genic (insertion or deletion within a gene), Indel-Nongenic (insertion or deletion outside of a genic region), or "Unobserved Target", which will only appear for the set of target mutations that don't appear in the sample if the *write.all.targets* option is set to TRUE.

This final data frame is named *samp_mutations*, and is written to the global environment. It can also be written to a file if *write.mut.table* is set to TRUE. By default, the columns included in *samp_mutations* are SAMP_NAME, CHR, POS, GENE, ALT_ID, AF, and DP. This *samp_mutations* object serves as default input to the `explore_mutations()` and `estimate_lineages()` functions, and must include at least the set of default columns to run either of these subsequent functions.

`explore_mutations()`

Summary

The `explore_mutations()` function opens an interactive RShiny dashboard that allows you to explore the data. The dashboard will have from 1 to 4 tabs at the top, depending on what combination of optional input files (if any) are provided...

Available Tabs

- **Lineages Present:** Available if the lineage/mutation file is provided with the *lineage.muts* option. The top plot represents the proportion of "lineage-characteristic mutations" present in a sample. These "lineage-characteristic mutations" are the set of mutations from the *lineage.muts* file that occur only in the selected lineage. In other words, *MixviR* looks through the set of mutations provided and removes any that are shared by more than one lineage. The remaining mutations are used to generate the plots in this tab. The "Presence Threshold" slider on the left allows the user to set a threshold for the proportion of such mutations required to consider a lineage as present in the sample. For each lineage identified as present (exceeding the threshold), the frequences of the lineage-characteristic mutations that occur in the sample are used to estimate a relative frequency of the lineage in the sample - these estimated relative frequencies are shown in the bottom plot, and can be generated with either the median or mean values (specific metric can be selected in left panel). Note that while biologically the sums of these estimated proportions can't exceed 1, there is no constraint with the way the raw estimates are calculated, and occasionaly the bars will exceed 1. When this happens, it may mean that at least one mutation that was identified and used as a "lineage-characteristic" mutation based on the provided list is, in reality, shared among two or more lineages. In cases where the sum exceeds 1, the default behavior is to proportionally scale the estimates down to sum to 1 (controlled by the "scale" option in the left panel). Additional details associated with each plot are provided as tooltips by hovering over plot features/regions.

- **New Mutations:** Present if a "location/dates" file is provided with the *dates* option. Table that lists all mutations first observed (across the entire dataset) on or after the selected date. So, if the mutation "S_D614G" was not observed in any sample before 6/20/2021, was observed in a single sample on that date, and then continued to be observed in multiple samples after that date, that mutation would show up in the *New Mutations* table for the date 6/20/21, and for any date prior to that, but would not show up if 6/21/21 or any later date is selected. If the *lineage.muts* option is defined, a column is added to the table that includes all lineages the mutation is associated with.
- **Mutation Frequencies:** Present if a location/dates file is provided with the *dates* option. Plots show estimated frequencies of a specific mutation(s) over time for one or more samples. Mutations should be entered in the form "S_D614G" for a substitution, or, for indels, "S_del144/144" for single amino-acid mutations, or "S_del143/145" for a multi-amino acid deletion. Multiple mutations can be entered by separating them with a comma. Multiple samples can be selected and will be distinguished by color on the plots. If more than one mutation is entered, each mutation is displayed as a separate facet on the plot. If in doubt about the naming convention of mutations, check the *samp_mutations* object or the mutation names in other tabs.
- **View Mutations:** Lists all mutations observed for the selected sample. If the *lineage.muts* option is defined, a column is added to the table that includes all lineages the mutation is associated with. If *write.all.targets* was set to TRUE in `call_mutations()`, all mutations in the *lineage.muts* file are included, even if no associated reads were observed (sequencing depths at underlying genomic positions are reported, while the reported observed frequency will be zero). This table is searchable, and can be sorted by columns (SHIFT + click to sort on more than one column).

`estimate_lineages()`

Summary

The `estimate_lineages()` function is meant to output the data from the "Variants Present" plots of `explore_mutations()` in table form. You can choose to write data for all lineages analyzed or just the lineages identified as present in the sample (default) based on the *presence.thresh* threshold. The tables are printed to the screen and can also be written to a file.

Cautions/Important Considerations

- Remove spaces and underscores (_) from the chromosome names given to *MixviR*. This applies to the *reference* and *bed* files associated with `call_mutations()`, and also the *lineage.muts* file if the optional "Chr" column is included. Also make sure the chromosome names match across these files.
- The *min.alt.freq* argument is set to 0.01 by default. It may be tempting to drop this to try to detect lineages at extremely low frequencies in a sample. This should be done with caution, as sequencing noise included in the vcf file can start to interfere with the translation of true amino acids in the sample. For example, consider a situation where a codon at positions 100-102 in the reference is CCC (amino acid P), and a sample has a true mutation to ACC (amino acid T) at a frequency of 75%. Now imagine that at position 101 (the 2nd codon position) there is sequencing/PCR noise of C->G with frequency 0.0005 (0.05%). If the *min.alt.freq* threshold is less than 0.0005, both mutations will be considered in the translation, and the sample codon will be called AGC (amino acid S) instead of the actual ACC (amino acid T). Fortunately, sequencing and PCR noise, at least in our experience, typically occurs well below the 0.01 frequency, and this seems to be a reasonable default choice. While some minor adjusting to this value is probably safe if necessary, extreme values in either direction will likely lead to unreasonable/unreliable results.
- Some amino acid substitutions have/require multiple underlying genomic mutations to their respective codon. In these cases, *MixviR* is expected to call the mutation correctly, but the relevant frequencies and associated sequencing depths will be obtained from the variant with the highest sequencing depth (if not identical, these are typically very similar).

- It may be tempting to try to include mutation data for lots of closely-related lineages in the lineage/mutation (*lineage.muts*) file to try to distinguish among lineages at a very fine scale. One example of this in SARS-CoV-2 is the Delta variant, which as of the time this vignette was written had >100 sublineages defined that were designated AY.1, AY.2, AY.3, etc... Note that any mutations that show up more than once in this file are removed for the analysis, and so including many closely-related lineages/groups will likely result in having few or no mutations to use for the analysis. The estimates of lineages present in the "Lineages Present" tab of the RShiny dashboard is highly dependent on having mutations that are highly-informative/diagnostic for each lineage.
- For the purpose of data visualization in the `explore_mutations()` Shiny dashboard, it's important that the mutation designations assigned by `call_mutations()` match those in the optional input files. If you're unsure, check the names in the ALT_ID column of *samp_mutations* against those in your files.
- The color palette used to generate plots in the RShiny dashboard (palette: "Paired") has 12 colors. Therefore, this is the maximum number of lineages that can be plotted.

Example Analyses

Example Data

MixviR comes with a set of example data files that can be used to test out the program. These include vcf files representing SARS-CoV-2 environmental samples obtained from a single location on three different dates, a "lineage.muts" file (example_lineage_muts.csv) containing a subset of four Sars-Cov2 variants to evaluate, a "location/dates" file (example_location_date.csv), and a "samp_mutations.tsv" file that stores the output of a `call_mutations()` run for these three samples with default settings.

You can use the `system.file()` function to see the location of the raw files. For example, the path to the directory containing the example vcf files is given by `system.file("extdata", "vcfs", package = "MixviR")` and the path to the lineage.muts file by `system.file("extdata", "vcfs", "example_lineage_muts.csv", package = "MixviR")`

Example Analysis 1- No Target Mutations In `call_mutations()`

Step 1: `call_mutations()`

Your first step with *MixviR* will typically be to run the `call_mutations()` function, which identifies all the mutations in the input datasets (vcf files). We'll point this function to the directory storing the three example vcf files, and since these data are for SARS-CoV-2, we can use the pre-formatted "Wuhan" reference. These two pieces of information (location of input files and reference info) are all that's required to make it run. In this case though, I'll also clean up the sample names by trimming off all the text after the "_" in each input file name with the *name.sep* option...

```
call_mutations(sample.dir = system.file("extdata", "vcfs", package = "MixviR"),
               reference = "Wuhan",
               name.sep = "_")
```

Running this creates a new object named *samp_mutations*, which stores all the mutations observed in each sample. In this case, there are a total of 316 mutations identified. The *samp_mutations* data frame looks like...

Most of the columns here are fairly self-explanatory. Each mutation identified is listed in the "ALT_ID" column. "AF" gives the estimated frequency of that mutation in the sample, which is simply the "ALT_COUNT"/"DP". The "POS" column provides the genomic position of the mutation along the chromosome. By default, this data frame serves as input for the `explore_mutations()` function, which we'll run next.

	SAMP_NAME	CHR	POS	GENE	ALT_ID	AF	DP
1	Sample1	NC-045512.2	241	non-genic	NC-045512.2_241C->T	1.00000000	692
2	Sample1	NC-045512.2	774	ORF1a	ORF1a_T170I	0.08057515	3686
3	Sample1	NC-045512.2	913	ORF1a	NC-045512.2_913C->T	0.99967969	3122
4	Sample1	NC-045512.2	1010	ORF1a	ORF1a_Y249H	0.04138771	3286
5	Sample1	NC-045512.2	1861	ORF1a	NC-045512.2_1861T->C	0.29987945	3318
6	Sample1	NC-045512.2	2060	ORF1a	ORF1a_A599T	0.02526670	3562
7	Sample1	NC-045512.2	2110	ORF1a	NC-045512.2_2110C->T	0.61747270	3205
8	Sample1	NC-045512.2	2417	ORF1a	ORF1a_R718*	0.07692308	13
9	Sample1	NC-045512.2	2758	ORF1a	NC-045512.2_2758A->T	0.02500000	40
10	Sample1	NC-045512.2	3037	ORF1a	NC-045512.2_3037C->T	0.99804688	1024
11	Sample1	NC-045512.2	3267	ORF1a	ORF1a_T1001I	0.54729730	3256
12	Sample1	NC-045512.2	3369	ORF1a	ORF1a_T1035I	0.25206738	3265
13	Sample1	NC-045512.2	3777	ORF1a	ORF1a_T1171I	0.37323944	142
14	Sample1	NC-045512.2	4197	ORF1a	ORF1a_E1311G	0.20281400	2914
15	Sample1	NC-045512.2	4936	ORF1a	ORF1a_K1557N	0.43884892	834
16	Sample1	NC-045512.2	5388	ORF1a	ORF1a_A1708D	0.63773485	3779
17	Sample1	NC-045512.2	5388	ORF1a	ORF1a_A1708G	0.36226515	3779
18	Sample1	NC-045512.2	5766	ORF1a	ORF1a_G1834A	0.01035582	3766

Figure 11: The samp_mutations data frame is the primary output of the call_mutations() function.

Step 2: `explore_mutations()`

Basic Run Running the `explore_mutations()` function opens an RShiny dashboard in a separate window. The number of tabs available in the window depends on the number of options (if any) passed to the function.

```
explore_mutations()
```

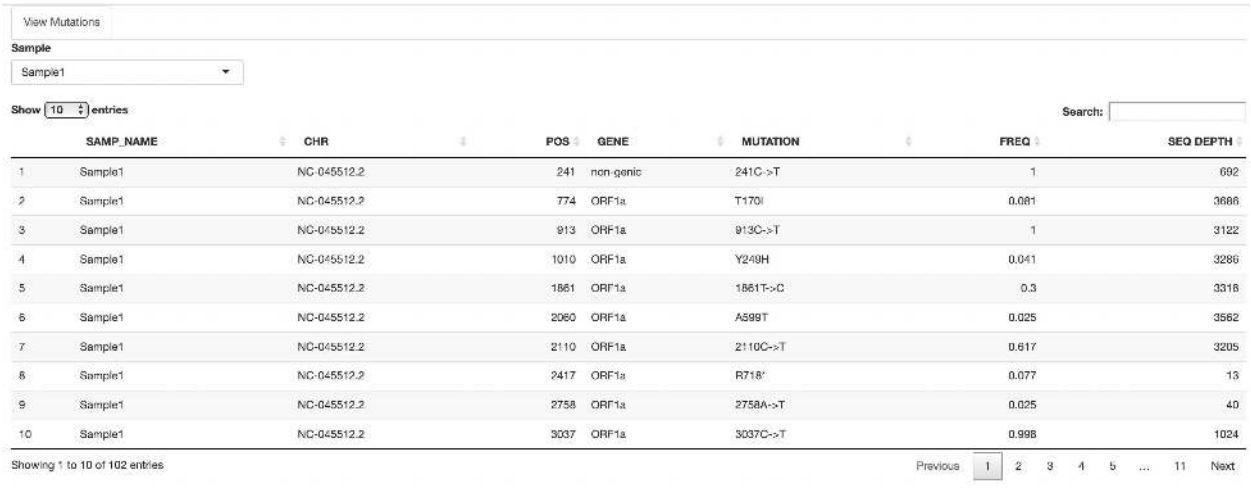


Figure 12: 'View Mutations' RShiny tab.

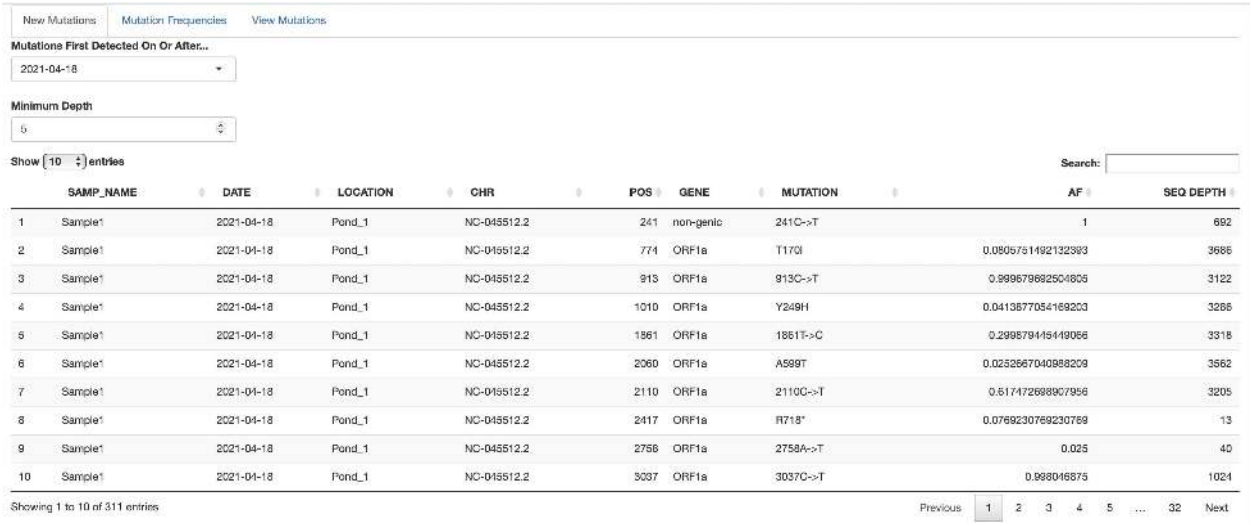
Here we ran `explore_mutations()` in its most basic form and got just a single tab named "View Mutations", which is just a slightly reformatted version of the *samp_mutations* data frame. There are options to select the sample you want to view (drop-down box in top left), and also to filter results by searching for specific text (i.e. a gene or mutation name). Each column is also sortable with the arrows at the top of the column (shift+click allows sorting by multiple columns).

Now lets try adding one option to `explore_mutations()`, passing it a location/dates file with the *dates* option...

```
explore_mutations(dates = system.file("extdata",
                                      "example_location_date.csv",
                                      package = "MixviR"))
```

With Dates This time we have three tabs available in the RShiny window: New Mutations, Mutation Frequencies, and View Mutations.

The New Mutations tab allows you to select a date and view the set of mutations that were observed for the first time (across the entire dataset) on or after that date.



	SAMP_NAME	DATE	LOCATION	CHR	POS	GENE	MUTATION	AF	SEQ DEPTH
1	Sample1	2021-04-18	Pond_1	NC-045512.2	241	non-genic	241C->T	1	692
2	Sample1	2021-04-18	Pond_1	NC-045512.2	774	ORF1a	T170I	0.0805751492132393	3686
3	Sample1	2021-04-18	Pond_1	NC-045512.2	913	ORF1a	913C->T	0.999579692504805	3122
4	Sample1	2021-04-18	Pond_1	NC-045512.2	1010	ORF1a	Y249H	0.0413577054169203	3266
5	Sample1	2021-04-18	Pond_1	NC-045512.2	1561	ORF1a	1561T->C	0.299579415449056	3318
6	Sample1	2021-04-18	Pond_1	NC-045512.2	2060	ORF1a	A599T	0.0252667040988209	3562
7	Sample1	2021-04-18	Pond_1	NC-045512.2	2110	ORF1a	2110C->T	0.617472698907956	3205
8	Sample1	2021-04-18	Pond_1	NC-045512.2	2417	ORF1a	R718*	0.0769230769230769	13
9	Sample1	2021-04-18	Pond_1	NC-045512.2	2758	ORF1a	2758A->T	0.025	40
10	Sample1	2021-04-18	Pond_1	NC-045512.2	3037	ORF1a	3037C->T	0.998046875	1024

Showing 1 to 10 of 311 entries
Previous 1 2 3 4 5 ... 32 Next

Figure 13: 'New Mutations' RShiny tab. The displayed mutations were first observed on or after the selected date.

The Mutation Frequencies tab allows you to enter one or more mutation names (comma separated if more than one), and the frequencies of these mutations are plotted for each date available in the dataset for the selected location. Multiple mutations are plotted as separate facets, while multiple locations can also be selected and are distinguished by color on each facet.

Providing a lineage/mutations file with the *lineage.muts* option in addition to a locations/dates file results in a fourth tab (and also some additional information included in the New Mutations and View Mutations tabs from above).

```
explore_mutations(dates = system.file("extdata",
                                      "example_location_date.csv",
                                      package = "MixviR"),
                  lineage.muts = system.file("extdata",
                                             "example_lineage_muts.csv",
                                             package = "MixviR"))
```

With Dates + Lineages The fourth tab is named "Variants Present", and provides two plots...

In this example, B.1.1.7 and P.1 have been selected for analysis. When a lineage is selected, *MixviR* filters the lineage/mutations file for mutations that are unique to that lineage. The number of these identified for each lineage is indicated in parentheses in the legend for the plot - in this case, there are 15 mutations unique to

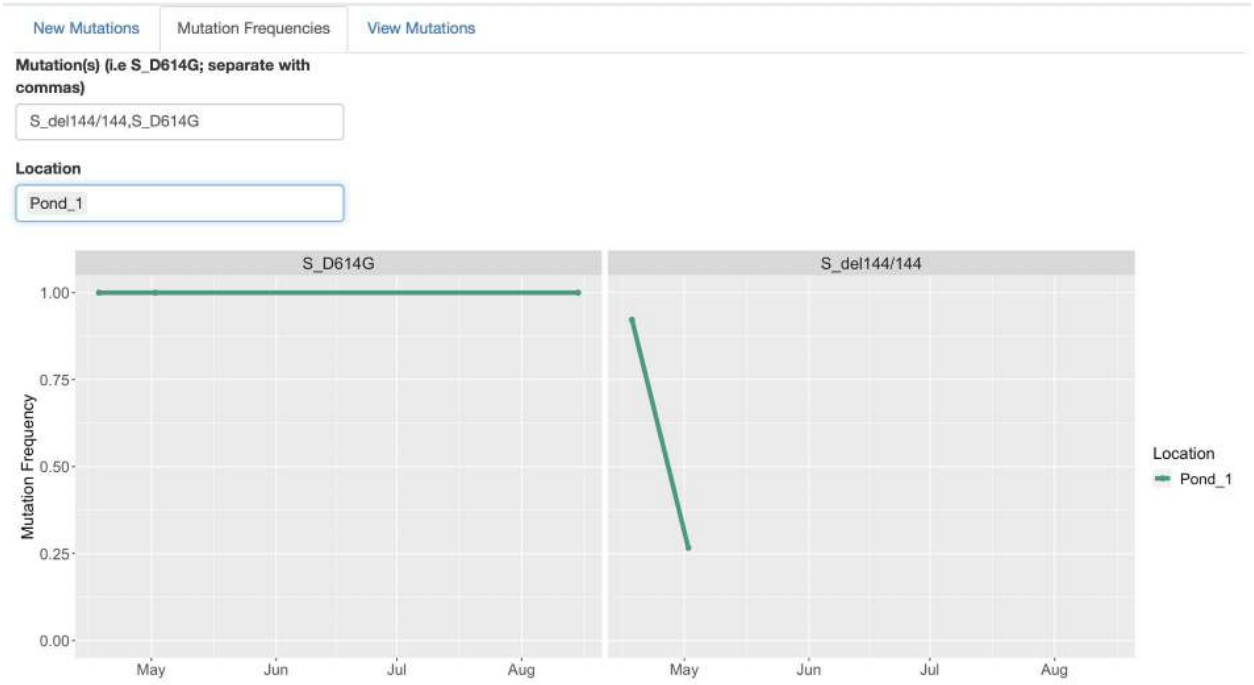


Figure 14: 'Mutation Frequencies' RShiny tab displaying frequencies of specific mutations over time.

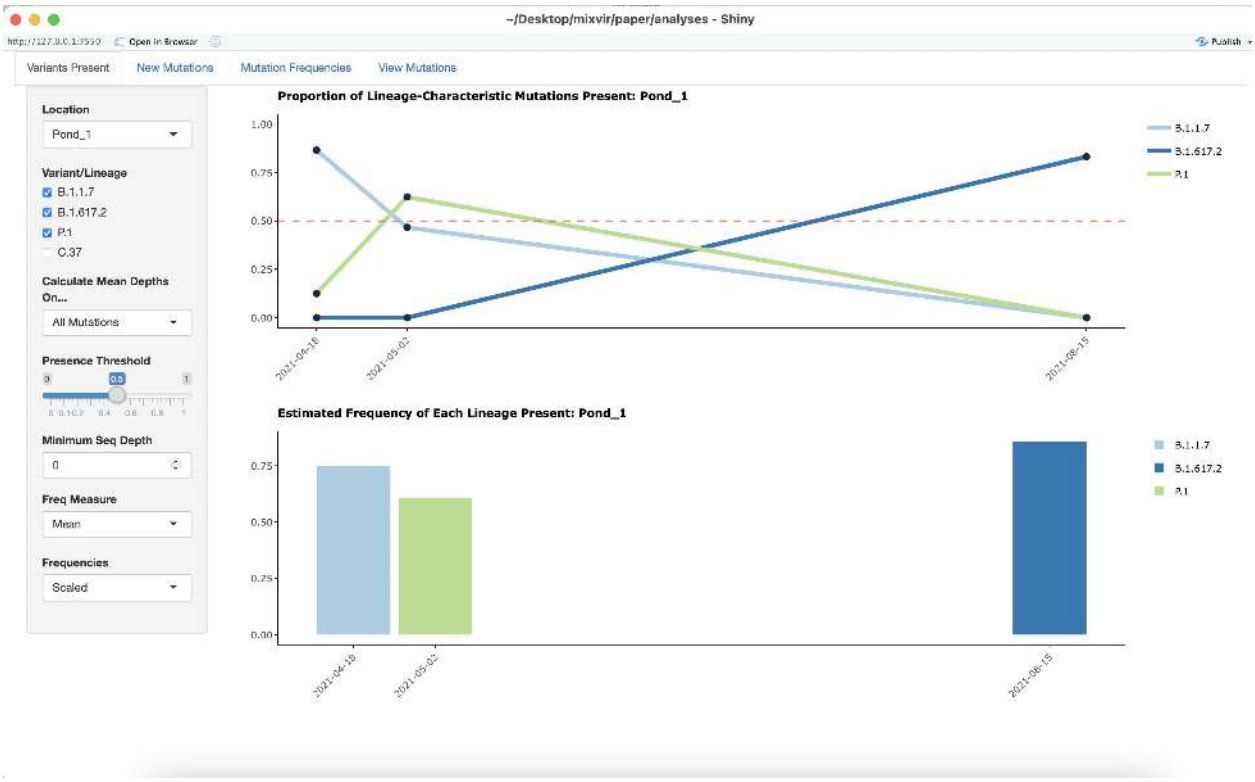


Figure 15: 'Variants Present' RShiny tab identifying lineages present in samples and estimating their frequencies.

B.1.1.7 and 16 unique to P.1. *MixviR* then checks to see how many of these were detected (at any frequency > the *min.alt.freq* set in `call_mutations()`) for each sample and plots this proportion. The red horizontal dashed line is a reference line showing the threshold proportion of mutations necessary to consider the lineage "present" in the sample. By default this is set to 0.5, but it can be adjusted with the slider on the left. For any lineages identified as present in a sample, the frequencies of the characteristic mutations present in the sample are averaged over to estimate the proportion of that lineage in the sample, which is plotted in the bottom plot.

Notice in Figure 15 that B.1.1.7 on the 2021-05-02 date had just under 50% of the lineage-characteristic mutations in the sample. Based on the default threshold of 0.5, only P.1 is called as present in that sample (and therefore plotted in the bottom plot). However, if the threshold is lowered to 0.40, then both P.1 and B.1.1.7 are called as present for the 2021-05-02 sample, and frequencies are reported for both as a stacked bar in the lower plot...

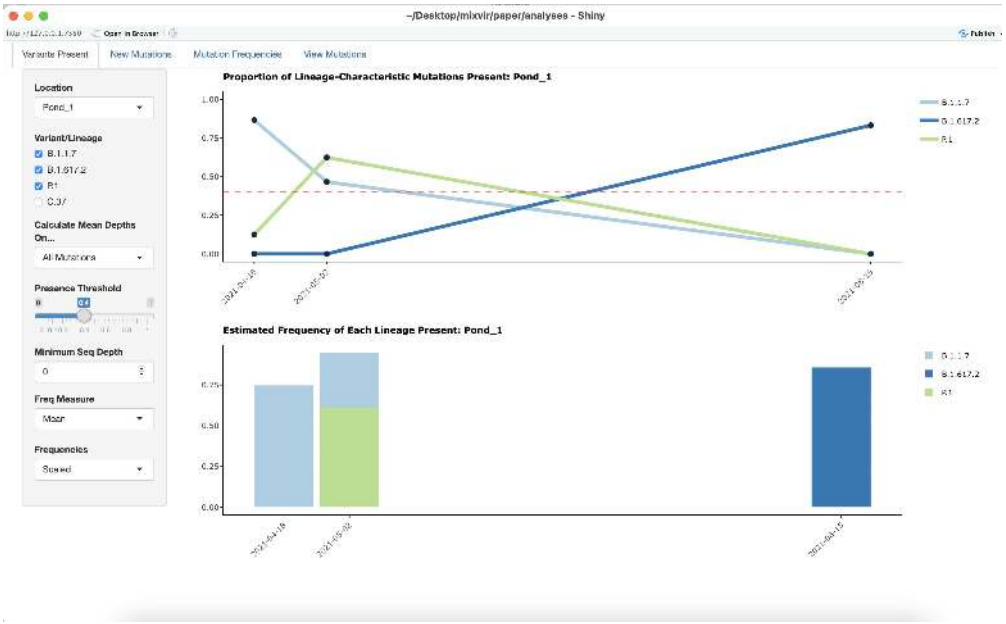


Figure 16: 'Variants Present' RShiny tab after adjusting the threshold for calling a lineage as present, and identifying a mixture of lineages in one sample.

Tooltips are also available if you hover over features on the plot (Figs. 17, 18).

Elements in the tooltip on the top plot...

- Lineage: The lineage analyzed
- Total Lineage-Characteristic Muts: The number of mutations the Proportion Present is based on. These are mutations in the *lineage.muts* file that are associated with only the target lineage (mutations not shared with any other lineages in the file).
- Proportion Present: The proportion of the lineage-characteristic mutations present in the sample (at a frequency greater than *min.alt.freq* from `call_mutations()`.
- Avg Seq Depth: The average sequencing depth for genomic positions underlying the mutations. The default is to calculate this from all mutations identified in the sample. Alternatively, it can be calculated based on just the lineage-characteristic mutations observed by setting the "Calculate Mean Depths On" drop-down box to "Lineage-Characteristic Mutations Only".
- Avg Seq Based On: The number of mutations used to calculate the average sequencing depth.

Tooltips on the bottom plot provide identities of the target mutations.

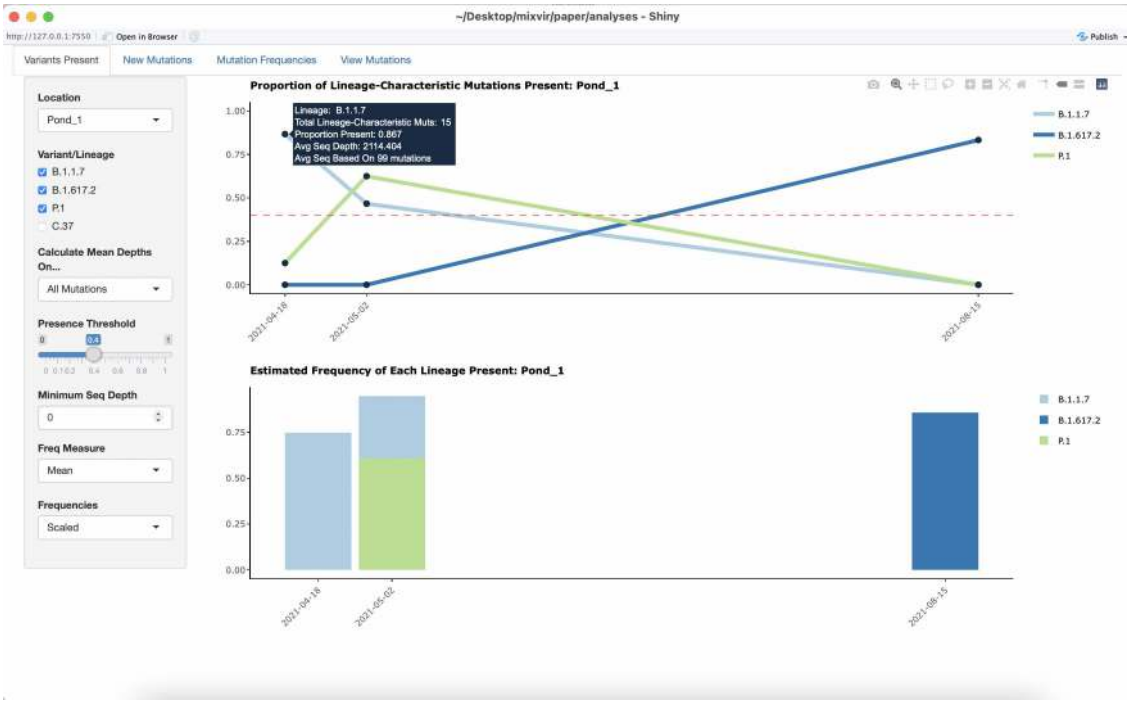


Figure 17: Example of a tooltip on Lineages Present plot.

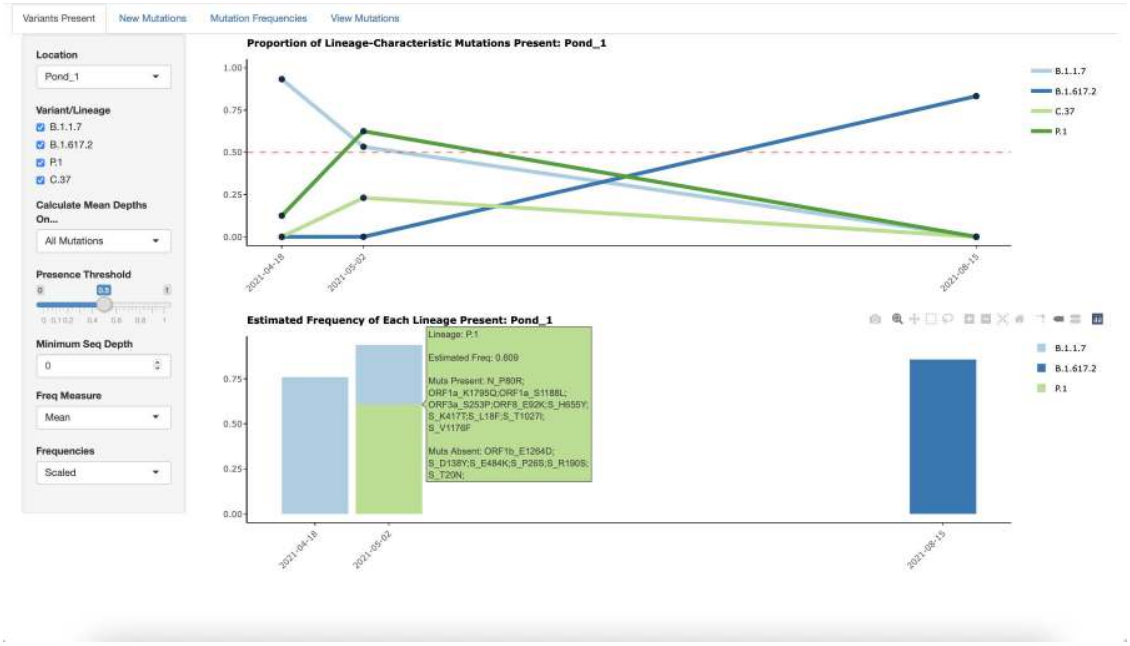


Figure 18: Example tooltop on lineage frequency plot.

Example Analysis 2 - Target Mutations Included In `call_mutations()`

This analysis is similar to the one above, but the *write.all.targets* option is used to report sequencing depths for all mutations associated with lineages of interest, even if the mutation isn't observed in the sample. This requires that the optional columns "Chr" and "Pos" are included in the *lineage.muts* file.

```
call_mutations(sample.dir = system.file("extdata",
                                        "vcfs",
                                        package = "MixviR"),
               reference = "Wuhan",
               name.sep = "_",
               write.all.targets = TRUE,
               lineage.muts = system.file("extdata",
                                          "example_lineage_muts.csv",
                                          package = "MixviR"))
```

In comparison to the `call_mutations()` run in example 1 above, which reported 316 mutations, this time 393 are reported. The extra mutations are those that were listed in the *lineage.muts* file but not identified in the sample. See rows 103-116 for Sample 1 below, which all have an observed frequency of zero, though the sequencing depth associated with the genomic positions underlying the mutations is > 0...

	SAMP_NAME	CHR	POS	GENE	ALT_ID	AF	DP
96	Sample1	NC-045512.2	28881	N	N_R203K	1.00000000	1333
97	Sample1	NC-045512.2	28882	N	N_R203K	1.00000000	1189
98	Sample1	NC-045512.2	28883	N	N_G204R	1.00000000	1376
99	Sample1	NC-045512.2	28977	N	N_S235F	1.00000000	1720
100	Sample1	NC-045512.2	29051	N	N_Fins260/1bp	0.21444639	4001
101	Sample1	NC-045512.2	29690	non-genic	NC-045512.2_29690G->T	0.17143659	3564
102	Sample1	NC-045512.2	29724	non-genic	NC-045512.2_29724C->T	0.16533865	3012
103	Sample1	NC-045512.2	23604	S	S_P681R	0.00000000	1422
104	Sample1	NC-045512.2	21618	S	S_T19R	0.00000000	142
105	Sample1	NC-045512.2	22995	S	S_T478K	0.00000000	159
106	Sample1	NC-045512.2	22917	S	S_L452R	0.00000000	143
107	Sample1	NC-045512.2	24410	S	S_D950N	0.00000000	3558
108	Sample1	NC-045512.2	22028	S	S_del157/158	0.00000000	2984
109	Sample1	NC-045512.2	10029	ORF1a	ORF1a_T3255I	0.00000000	3658
110	Sample1	NC-045512.2	23525	S	S_H655Y	0.00000000	1075
111	Sample1	NC-045512.2	25088	S	S_V1176F	0.00000000	3556
112	Sample1	NC-045512.2	21614	S	S_L18F	0.00000000	141
113	Sample1	NC-045512.2	21638	S	S_P26S	0.00000000	171
114	Sample1	NC-045512.2	21621	S	S_T20N	0.00000000	150
115	Sample1	NC-045512.2	24642	S	S_T1027I	0.00000000	3680
116	Sample1	NC-045512.2	22812	S	S_K417T	0.00000000	691

Figure 19: Example of the samp_mutations data frame when the write.all.muts option is used to include sequencing depths for mutations not observed in the sample.

If target mutations were included in the `call_mutations()` run, this can be indicated when running `explore_mutations()` by setting *all.target.muts* to TRUE, and an additional piece of information (Proportion Exceeding Seq Depth Threshold) is added to the tooltips on the "Variants Present" plot (Fig 20). When mutations are not identified in a sample, this measure can help provide some insight into whether they are missing because they truly don't occur in the sample, or because the sequencing didn't adequately cover their associated genomic positions.

The individual mutations, along with their sequencing depth, can also be explored in the "View Mutations" tab.

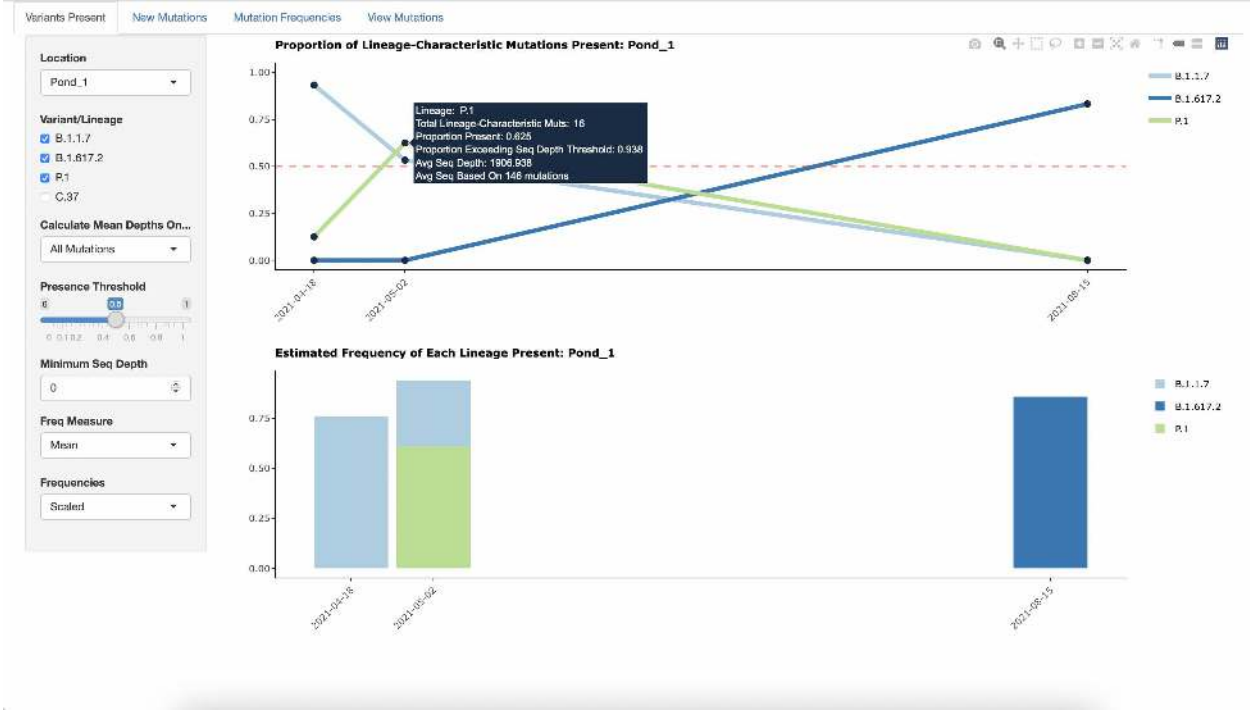


Figure 20: Example tooltip in explore_mutations() run with all.target.muts set to TRUE.

Getting Help

While you're welcome to email the authors directly if you have questions about, or problems with *MixviR* that aren't addressed in this vignette, we encourage you to instead visit the MixviR Google Group, and post your issue if it hasn't yet been addressed there.

References

Danecek P, Bonfield JK, et al. Twelve years of SAMtools and BCFtools. Gigascience (2021) 10(2):giab008

DePristo M, Banks E, Poplin R, Garimella K, Maguire J, Hartl C, Philippakis A, del Angel G, Rivas MA, Hanna M, McKenna A, Fennell T, Kernytsky A, Sivachenko A, Cibulskis K, Gabriel S, Altshuler D, Daly M. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nat Genet, 43:491-498.

Knaus BJ, Grünwald NJ (2017). "VCFR: a package to manipulate and visualize variant call format data in R." *Molecular Ecology Resources*, *17*(1), 44-53.

H. Pagès, P. Aboyoun, R. Gentleman and S. DebRoy (2019). Biostrings: Efficient manipulation of biological strings. R package version 2.52.0.