Shazam: Mutation analysis

Susanna Marquez & Julian Q. Zhou

2025-12-26

Contents

Basic mutational load calculations are provided by the `observedMutations` function. `observedMutations` provides multiple options to control how mutations are calculated. Mutations can be calculated as either counts or frequencies, divided into replacement (R) and silent (S) mutations, and subset into FWR and CDR specific mutations. Additionally, alternative mutational definitions may be considered based on the physicochemical properties of translated codons.

Example data

A small example AIRR Rearrangement database is included in the `alakazam` package. Analyzing mutations requires the following fields (columns) to be present in the table:

- `sequence_alignment`
- `germline_alignment_d_mask`

```
# Import required packages
library(alakazam)
library(dplyr)
library(ggplot2)
library(shazam)

# Load and subset example data
data(ExampleDb, package="alakazam")
db <- subset(ExampleDb, c_call %in% c("IGHA", "IGHG") & sample_id == "+7d")
```

Calculate the counts and frequencies of mutations over the entire sequence

When calling `observedMutations` with `regionDefinition=NULL`, the entire input sequence (`sequenceColumn`) is compared to the germline sequence (`germlineColumn`) to identify R and S mutations. If `frequency=TRUE`, the number of mutations is expressed as the frequency of mutations over the total number of positions that are non-N in both the input and the germline sequences.

In the example below, the counts (`frequency=FALSE`) and frequencies (`frequency=TRUE`) of R and S mutations are calculated separately. New columns containing mutation counts are appended to the input data.frame with names in the form `mu_count_<Region>_<R/S>`. Mutation frequencies appear in new columns named `mu_freq_<Region>_<R/S>`.

```
# Calculate R and S mutation counts
db_obs <- observedMutations(db, sequenceColumn="sequence_alignment",
                            germlineColumn="germline_alignment_d_mask",
                            regionDefinition=NULL,
                            frequency=FALSE,
                            nproc=1)
# Show new mutation count columns
db_obs %>%
  select(sequence_id, starts_with("mu_count_")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 3
##   sequence_id    mu_count_seq_r mu_count_seq_s
##   <chr>                   <dbl>          <dbl>
## 1 GN5SHBT07FUXY8              0              0
## 2 GN5SHBT05JMPI5              8              2
## 3 GN5SHBT08H4LPP              8              2
## 4 GN5SHBT05JGND3              0              0
```

```
# Calculate R and S mutation frequencies
db_obs <- observedMutations(db_obs, sequenceColumn="sequence_alignment",
                            germlineColumn="germline_alignment_d_mask",
                            regionDefinition=NULL,
                            frequency=TRUE,
                            nproc=1)
# Show new mutation frequency columns
db_obs %>%
  select(sequence_id, starts_with("mu_freq_")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 3
##   sequence_id    mu_freq_seq_r mu_freq_seq_s
##   <chr>                  <dbl>         <dbl>
## 1 GN5SHBT07FUXY8        0             0
## 2 GN5SHBT05JMPI5        0.0237        0.00592
## 3 GN5SHBT08H4LPP        0.0236        0.00590
## 4 GN5SHBT05JGND3        0             0
```

Specifying the `combine=TRUE` argument will aggregate all mutation columns into a single value.

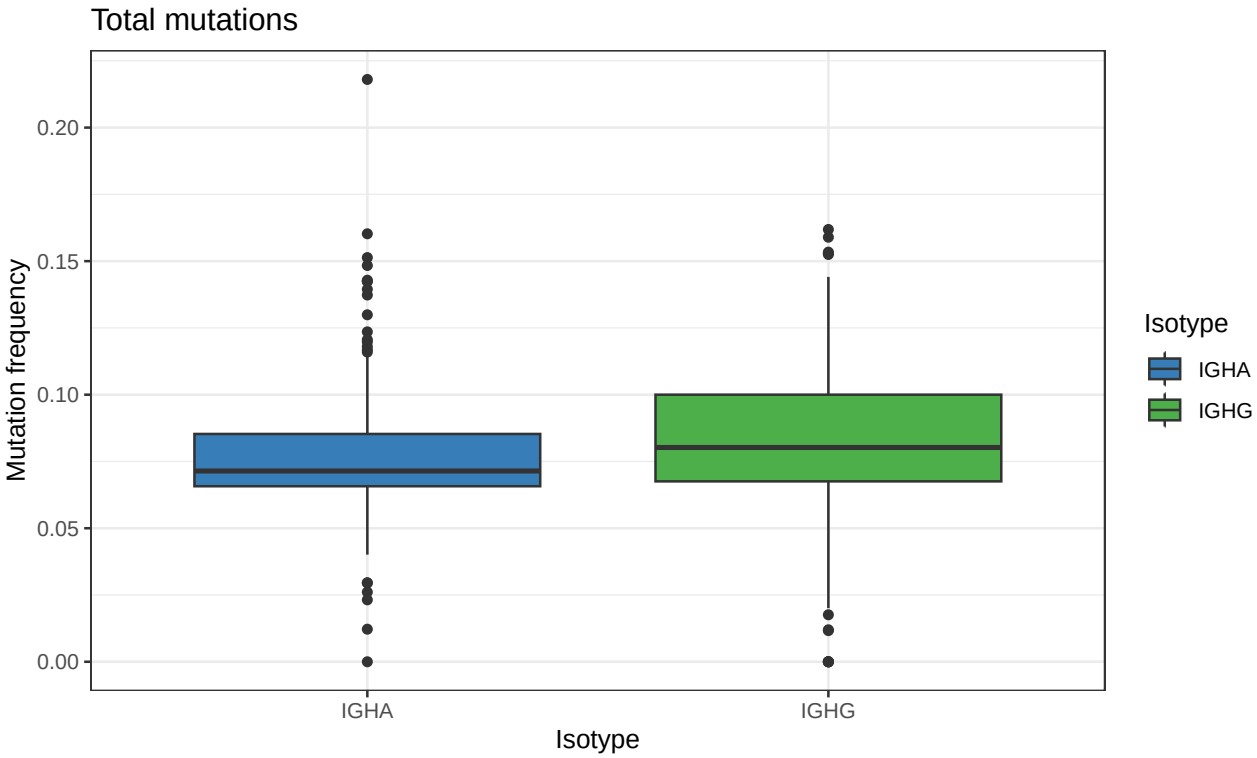
```
# Calculate combined R and S mutation frequencies
db_obs <- observedMutations(db, sequenceColumn="sequence_alignment",
                            germlineColumn="germline_alignment_d_mask",
                            regionDefinition=NULL,
                            frequency=TRUE,
                            combine=TRUE,
```

```
                                nproc=1)
# Show new mutation frequency columns
db_obs %>%
  select(sequence_id, starts_with("mu_freq")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 2
##   sequence_id    mu_freq
##   <chr>            <dbl>
## 1 GN5SHBT07FUXY8  0
## 2 GN5SHBT05JMPI5  0.0296
## 3 GN5SHBT08H4LPP  0.0295
## 4 GN5SHBT05JGND3  0
```

We can plot the mutation frequencies and explore differences between samples or isotypes.

```
g1 <- ggplot(db_obs, aes(x=c_call, y=mu_freq, fill=c_call)) +
        geom_boxplot() +
        labs(title="Total mutations",
             x="Isotype", y="Mutation frequency") +
        scale_fill_manual(name="Isotype", values=IG_COLORS, limits=force) +
        theme_bw()
plot(g1)
```



Calculate mutations within subregions

To restrict the mutational analysis to a particular area in the sequence, the `regionDefinition` argument needs to be assigned a `RegionDefinition` object, which simply defines the subregion

boundaries of the Ig sequence. For convenience, `shazam` provides a set of such objects, for which an overview is provided via `?IMGT_SCHEMES`. Each of these objects cover the IMGT numbered V segment up to nucleotide position 312. Different objects treat regions within the V segment with varying granularity:

- `IMGT_V_BY_CODONS`: treats each codon, from codon 1 to codon 104, as a distinct region;
- `IMGT_V_BY_REGIONS`: defines regions to be CDR1, CDR2, FWR1, FWR2 and FWR3;
- `IMGT_V`: defines regions to be either CDR or FWR;
- `IMGT_V_BY_SEGMENTS`: provides no subdivisions and treats the entire V segment as a single region.
- `IMGT_VDJ`: All regions, including CDR3 and FWR4, grouped as either CDR or FWR. This `RegionDefinition` is initially empty, and one is created on the fly for each set of clonally related sequences.
- `IMGT_VDJ_BY_REGIONS`: CDR1, CDR2, CDR3, FWR1, FWR, FWR3 and FWR4 regions treated as individual regions. This `RegionDefinition` is initially empty, and one is created on the fly for each set of clonally related sequences.

When supplying one of these objects to `regionDefinition`, and with `combined=FALSE`, the resultant mutation counts/frequencies will be tabulated in a way consistent with the granularity of the object's region definition. For example,

- With `IMGT_V_BY_REGIONS`, mutation frequencies will be reported in columns `mu_freq_cdr1_r`, `mu_freq_cdr1_s`, `mu_freq_cdr2_r`, `mu_freq_cdr2_s`, `mu_freq_fwr1_r`, `mu_freq_fwr1_s`, `mu_freq_fwr2_r`, `mu_freq_fwr2_s`, `mu_freq_fwr3_r`, and `mu_freq_fwr3_s`.
- With `IMGT_V`, mutation frequencies will be reported in columns `mu_freq_cdr_r`, `mu_freq_cdr_s`, `mu_freq_fwr_r`, and `mu_freq_fwr_s`.
- With `IMGT_V_BY_SEGMENTS`, mutation frequencies will be reported in columns `mu_freq_v_r`, and `mu_freq_v_s`.

In the following example, we will explore the mutation frequency in the V-segment using two of the region definitions.

```
# Calculate R and S mutation counts for individual CDRs and FWRs
db_obs_v <- observedMutations(db, sequenceColumn="sequence_alignment",
                              germlineColumn="germline_alignment_d_mask",
                              regionDefinition=IMGT_V_BY_REGIONS,
                              frequency=FALSE,
                              nproc=1)
# Show new FWR mutation columns
db_obs_v %>%
  select(sequence_id, starts_with("mu_count_fwr")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 7
##   sequence_id    mu_count_fwr1_r mu_count_fwr1_s mu_count_fwr2_r mu_count_fwr2_s
##   <chr>                    <dbl>           <dbl>           <dbl>           <dbl>
## 1 GN5SHBT07FUXY8               0               0               0               0
## 2 GN5SHBT05JMPI5               1               0               0               0
## 3 GN5SHBT08H4LPP               1               0               0               0
## 4 GN5SHBT05JGND3               0               0               0               0
## # i 2 more variables: mu_count_fwr3_r <dbl>, mu_count_fwr3_s <dbl>
```

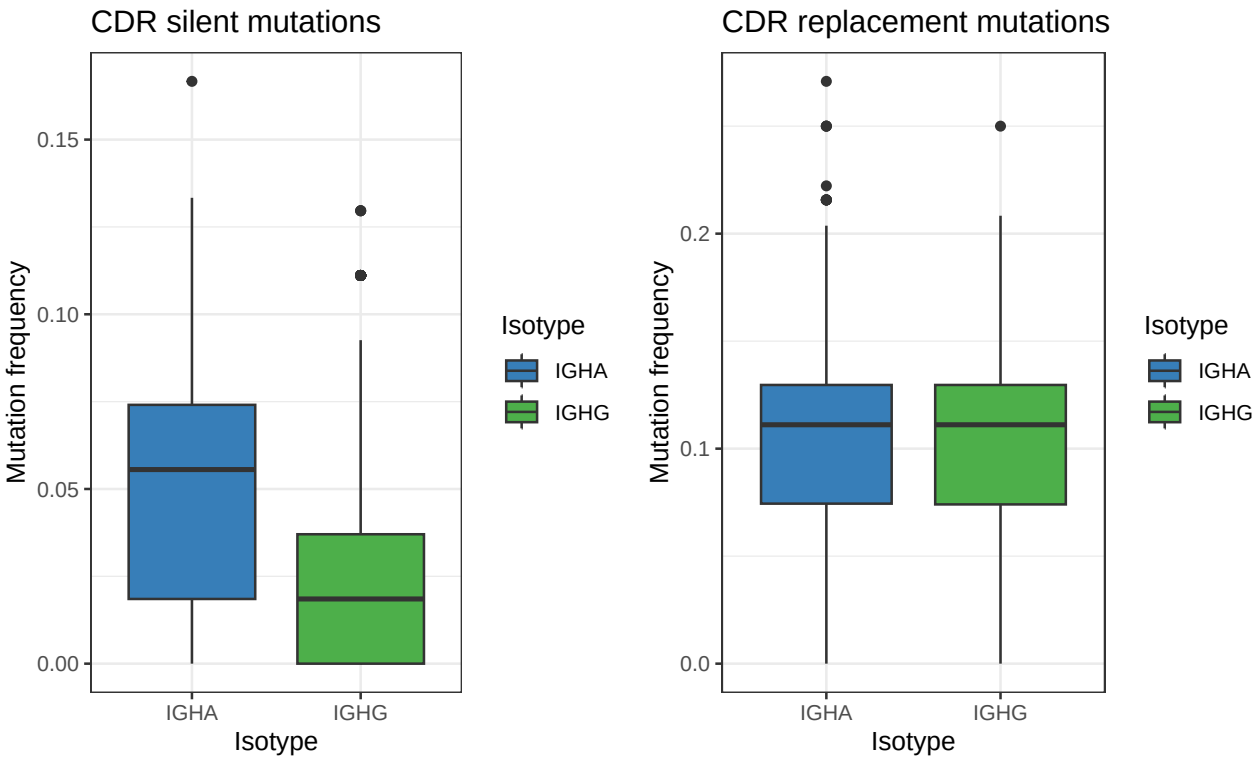
```
# Calculate aggregate CDR and FWR V-segment R and S mutation frequencies
db_obs_v <- observedMutations(db_obs_v, sequenceColumn="sequence_alignment",
                              germlineColumn="germline_alignment_d_mask",
                              regionDefinition=IMGT_V,
                              frequency=TRUE,
                              nproc=1)
# Show new CDR and FWR mutation frequency columns
db_obs_v %>%
  select(sequence_id, starts_with("mu_freq_")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 5
##   sequence_id    mu_freq_cdr_r mu_freq_cdr_s mu_freq_fwr_r mu_freq_fwr_s
##   <chr>                  <dbl>         <dbl>         <dbl>         <dbl>
## 1 GN5SHBT07FUXY8             0             0        0             0
## 2 GN5SHBT05JMPI5             0             0        0.0251        0.00418
## 3 GN5SHBT08H4LPP             0             0        0.025         0.00417
## 4 GN5SHBT05JGND3             0             0        0             0
```

Plot a comparison between CDR silent and replacement mutations.

```
g2 <- ggplot(db_obs_v, aes(x=c_call, y=mu_freq_cdr_s, fill=c_call)) +
        geom_boxplot() +
        labs(title="CDR silent mutations",
             x="Isotype", y="Mutation frequency") +
        scale_fill_manual(name="Isotype", values=IG_COLORS, limits=force) +
        theme_bw()
g3 <- ggplot(db_obs_v, aes(x=c_call, y=mu_freq_cdr_r, fill=c_call)) +
        geom_boxplot() +
        labs(title="CDR replacement mutations",
             x="Isotype", y="Mutation frequency") +
        scale_fill_manual(name="Isotype", values=IG_COLORS, limits=force) +
        theme_bw()

alakazam::gridPlot(g2, g3, ncol=2)
```



Use amino acid physicochemical properties to define mutations

By default, replacement and silent mutations are determined by exact amino acid identity; this can be changed by setting the `mutationDefinition` argument. For convenience, `shazam` provides a set of `MutationDefinition` objects defining changes in amino acid charge, hydrophobicity, polarity and volume.

In the following example, replacement mutations are defined as amino acid changes that lead to a change in charge (`mutationDefinition=CHARGE_MUTATIONS`). Mutations that do not alter the charge classification of a translated codon will be considered silent mutations.

```
# Calculate charge mutation frequency for the full sequence
db_obs_ch <- observedMutations(db, sequenceColumn="sequence_alignment",
                               germlineColumn="germline_alignment_d_mask",
                               regionDefinition=NULL,
                               mutationDefinition=CHARGE_MUTATIONS,
                               frequency=TRUE,
                               nproc=1)
# Show new charge mutation frequency columns
db_obs_ch %>%
  select(sequence_id, starts_with("mu_freq_")) %>%
  head(n=4)
```

```
## # A tibble: 4 x 3
##   sequence_id    mu_freq_seq_r mu_freq_seq_s
##   <chr>                  <dbl>         <dbl>
## 1 GN5SHBT07FUXY8       0              0
## 2 GN5SHBT05JMPI5       0.00296        0.0266
```

```
## 3 GN5SHBT08H4LPP       0.00295         0.0265
## 4 GN5SHBT05JGND3       0               0
```

We can make a plot to visualize if mutations that change the sequence charge are more frequent in one isotype.

```
g4 <- ggplot(db_obs_ch, aes(x=c_call, y=mu_freq_seq_r, fill=c_call)) +
        geom_boxplot() +
        labs(title="Charge replacement mutations",
             x="Isotype", y="Mutation frequency") +
        scale_fill_manual(name="Isotype", values=IG_COLORS, limits=force) +
        theme_bw()
```

```
plot(g4)
```