Molnupiravir in global sequencing databases: open data version

This R notebook analyses signatures of molnupiravir mutagenesis using open data from INSDC. This means that the input files, extracted from the MAT, can be stored on Zenodo. The main analysis in our manuscript is based on a combination of open data version and data in the GISAID database.

The input files analysed here are a processed form of data from https://hgwdev.gi.ucsc.edu/~angie/UShER_SAR
CoV-2/

Analysis of data from mutation annotated tree

```
CtoTthreshold = 0.2
GtoAthreshold = 0.25
transitionthreshold = 0.9


red <- "#e31919"
blue1 <- "#5450f2"
threshold_branch_length <- 10
library(Biostrings)
```

```
Loading required package: BiocGenerics


Attaching package: 'BiocGenerics'

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs
```

```
The following objects are masked from 'package:base':

    anyDuplicated, aperm, append, as.data.frame, basename, cbind,
    colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
    get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
    match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
    Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
    table, tapply, union, unique, unsplit, which.max, which.min

Loading required package: S4Vectors

Loading required package: stats4


Attaching package: 'S4Vectors'

The following objects are masked from 'package:base':

    expand.grid, I, unname

Loading required package: IRanges

Loading required package: XVector

Loading required package: GenomeInfoDb


Attaching package: 'Biostrings'

The following object is masked from 'package:base':

    strsplit
```

```r
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
v ggplot2 3.4.2     v purrr   1.0.1
v tibble  3.2.1     v dplyr   1.1.2
v tidyr   1.3.0     v stringr 1.5.0
v readr   2.1.4     v forcats 1.0.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::collapse()   masks Biostrings::collapse(), IRanges::collapse()
x dplyr::combine()    masks BiocGenerics::combine()
x purrr::compact()    masks XVector::compact()
x dplyr::desc()       masks IRanges::desc()
x tidyr::expand()     masks S4Vectors::expand()
x dplyr::filter()     masks stats::filter()
x dplyr::first()      masks S4Vectors::first()
x dplyr::lag()        masks stats::lag()
x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
x purrr::reduce()     masks IRanges::reduce()
x dplyr::rename()     masks S4Vectors::rename()
x dplyr::slice()      masks XVector::slice(), IRanges::slice()
```

```
data_nodes <- read_tsv("https://zenodo.org/record/8252388/files/all_nodes.tsv.gz")
```

```
Rows: 9181422 Columns: 19
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr   (4): node_id, consensus_country, consensus_year, age
dbl  (14): num_descendants, date_length, A>C, A>G, A>T, C>A, C>G, C>T, G>A, ...
date  (1): date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_nodes <- data_nodes %>% mutate(total_muts = `A>C` + `A>G` + `A>T` + `C>A` + `C>G` + `
  mutate(
    consensus_country = recode(consensus_country,
                     "England" = "United Kingdom",
                     "Scotland" = "United Kingdom",
                     "Northern_Ireland" = "United Kingdom","Northern Ireland" = "United Ki
                     "Wales" = "United Kingdom")
  )
```

```r
data_muts <- read_tsv("https://zenodo.org/record/8252388/files/all_node_muts.tsv.gz")
```

```
Rows: 8278016 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (8): node_id, original_nt, alternative_nt, gene, original_aa, alternativ...
dbl (2): nt_index, aa_index
lgl (1): is_synonymous

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
parenthood <- read_tsv("https://zenodo.org/record/8252388/files/parenthood.tsv.gz")
```

```
Rows: 9181421 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (2): child, parent

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
find_children <- function(parenthood, parent) {
  # Find the immediate children of the parent
  children <- parenthood$child[parenthood$parent == parent]

  # Initialize a vector to store all descendants
  all_descendants <- c()

  # Loop through each child and find their descendants
  for (child in children) {
    # Add the child to the list of descendants
    all_descendants <- c(all_descendants, child)

    # Recursively find the descendants of the child
    child_descendants <- find_children(parenthood, child)

    # Add the descendants of the child to the list of all descendants
    all_descendants <- c(all_descendants, child_descendants)
  }

  return(all_descendants)
}

get_parent <- function(parenthood, node) {
  # Find the parent of the node
  parent <- parenthood$parent[parenthood$child == node]

  # If there is no parent (i.e., the node is the root), return NULL
  if (length(parent) == 0) {
    return(NULL)
  }

  return(parent)
}

data_muts <- data_muts %>% filter(gene != "ORF1a")
```

```
library(tidyverse)
library(cowplot)
data2 <- read_tsv("https://zenodo.org/record/8252388/files/public-latest.metadata.tsv.gz",
```

```
Rows: 7598460 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (2): date, country

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data3 <- data2 %>%
  select(date, country) %>%
  extract(date, "(\\d{4})", into = "year")  %>%
  mutate(
    country = recode(country,
                     "England" = "United Kingdom",
                     "Scotland" = "United Kingdom",
                     "Northern_Ireland" = "United Kingdom", "Northern Ireland" = "United K
```

```
                        "Wales" = "United Kingdom")
  )
countries_totals <- data3 %>%
  group_by(year, country) %>%
  tally() %>%
  mutate(total_genomes = n)

countries_totals
```

```
# A tibble: 316 x 4
# Groups:   year [6]
   year  country        n total_genomes
   <chr> <chr>      <int>         <int>
 1 2019  China         46            46
 2 2020  Argentina     64            64
 3 2020  Armenia       43            43
 4 2020  Australia  13299         13299
 5 2020  Bahrain      140           140
 6 2020  Bangladesh   496           496
 7 2020  Belgium        3             3
 8 2020  Belize         4             4
 9 2020  Benin         12            12
10 2020  Bhutan        40            40
# i 306 more rows
```

Temporal and geographic associations

As compared to the closed-data version of this analysis, there is less widespread data in open databases. In particular, in the open data version sequences from Australia, which provides a key signal of high use of molnupiravir, and from Canada and much of the signal from France which provide a signal for high levels of sequencing without approval of molnupiravir.

```
library(ggrepel)

tallied_big <- data_nodes %>%
  dplyr::rename(country = consensus_country, year = consensus_year) %>%
  filter(flagged, total_muts >= threshold_branch_length) %>%
  group_by(country, year) %>%
  tally() %>%
  dplyr::rename(ga_branches = n) %>%
  full_join(countries_totals) %>%
  replace_na(list("ga_branches" = 0))
```

```
Joining with `by = join_by(country, year)`
```

```
tallied <- tallied_big %>% filter(year == "2022")


# Define approved and not_approved countries
approved <- c(
  "USA", "United Kingdom", "Germany", "Denmark", "Japan", "India", "Australia", "Israel", "
  "Russia", "South Korea", "New Zealand", "Belgium", "Mauritius", "Vietnam", "Thailand", "
)
not_approved <- c(
  "France", "Canada", "Sweden", "Netherlands", "Finland", "Switzerland", "Norway", "Irelan
)

# Define usage
usage <- c(
  "Australia" = "\n(100 per 10k)",
  "United Kingdom" = "\n(5 per 10k)",
  "Japan" = "(50 per 10k)",
  "Italy" = "\n(10 per 10k)"
)

# List of years
years <- c("2020", "2021", "2022", "2023")
lightpurple <- "#c39ecd"
darkpurple <- "#77488c"
darkorange <- "#fe670a"
lightorange <- "#f1ae85"
midorange <- "#ff883c"
year_pal <- c(lightpurple, darkpurple, darkorange, lightorange)
names(year_pal) <- years



# Loop through each year
for (i in 0:length(years)) {
  # Subset data
```

```
    data_subset <- data_nodes %>%
      filter(total_muts > 20, consensus_year %in% years[0:i])

    # Define plot
    scatter <- ggplot(data_subset, aes(x = `G>A` / total_muts, y = transitions / total_muts,
      geom_point() +
      theme_bw() +
      labs(x = "G\u00adto\u00adA proportion", y = "Transition proportion", color = "Year") +
      scale_color_manual(values = year_pal) +
      theme(legend.position = "bottom") +
      scale_x_continuous(label = scales::percent) +
      scale_y_continuous(label = scales::percent) +
      coord_cartesian(xlim = c(0, 0.65), ylim = c(0, 1))

    # Save plot
    ggsave(paste0("big_scatter_big_", paste(years[0:i], collapse = "_"), ".pdf"), plot = sca
    ggsave(paste0("scatter_big_", paste(years[0:i], collapse = "_"), ".pdf"), plot = scatter
  }


  scatter <- scatter +
    annotate("rect", xmin = 0.25, xmax = 0.6, ymin = 0.6, ymax = 1.05, fill = NA, color = "#



  tallied$approved <- case_when(
    tallied$country %in% approved ~ "Available",
    tallied$country %in% not_approved ~ "Not available",
    TRUE ~ "Not identified"
  )

  country_plot_data = tallied %>% filter(country != "?", total_genomes > 500, year == "2022"

  library(knitr)


  library(knitr)
  library(kableExtra)
```

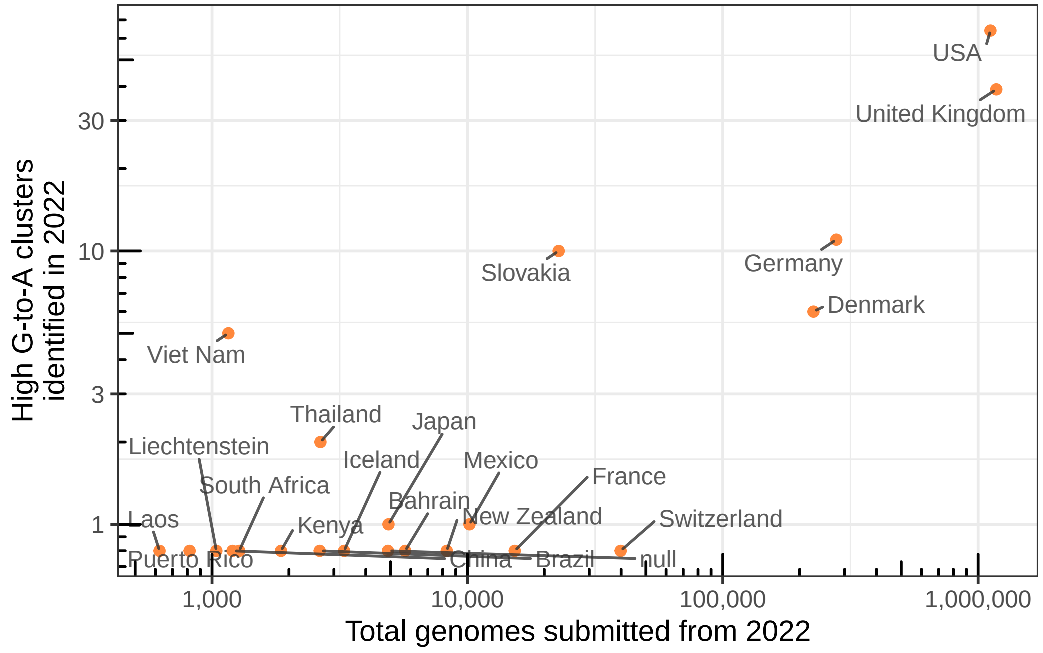
```
Attaching package: 'kableExtra'
```

```
The following object is masked from 'package:dplyr':

    group_rows
```

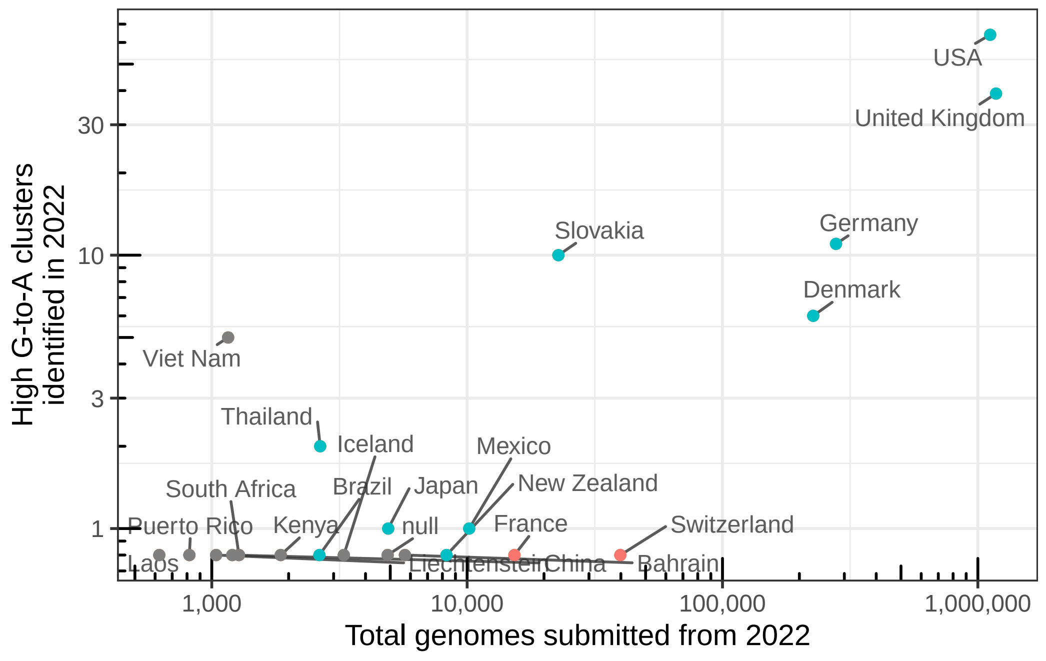
```
forlatex = country_plot_data %>% select(country, ga_branches,total_genomes) %>% arrange(-t

country_plot_data
```

```
# A tibble: 22 x 7
# Groups:   country [22]
   country        year  ga_branches       n total_genomes approved  usage
   <chr>          <chr>       <int>   <int>         <int> <fct>     <chr>
 1 Denmark        2022            6  226577        226577 Available ""
 2 Germany        2022           11  278377        278377 Available ""
 3 Japan          2022            1    4909          4909 Available "(50 per 10~
 4 Mexico         2022            1   10189         10189 Available ""
 5 Slovakia       2022           10   22775         22775 Available ""
 6 Thailand       2022            2    2658          2658 Available ""
 7 USA            2022           64 1117526       1117526 Available ""
 8 United Kingdom 2022           39 1178211       1178211 Available "\n(5 per 1~
 9 Viet Nam       2022            5    1159          1159 <NA>      ""
10 Bahrain        2022            0    5707          5707 <NA>      ""
# i 12 more rows
```

```
names(forlatex) <- c("Country", "High G-to-A branches in 2022", "Total genomes in 2022")


latex_table <- kable(forlatex, "latex", booktabs = TRUE,  linesep = "" ,
                     col.names = names(forlatex),
                     align = c('l', 'r', 'r'))
writeLines(latex_table, "countrytable.tex")


country_comp <- ggplot(
  country_plot_data,
  aes( # color = approved,
    x = total_genomes, y = ifelse(ga_branches == 0, 0.8, ga_branches), label = country
  )
) +
  geom_point(alpha = 1, color = midorange) +
  scale_x_log10(labels = scales::comma) +
  scale_y_log10() +
  geom_text_repel(alpha = 0.8, max.overlaps = 300, force = 50, min.segment.length = 0, lin
  theme_bw() +
  labs(x = "Total genomes submitted from 2022", y = "High G\u00adto\u00adA clusters\nident
  theme(legend.position = "none") +
  annotation_logticks()


country_comp
```



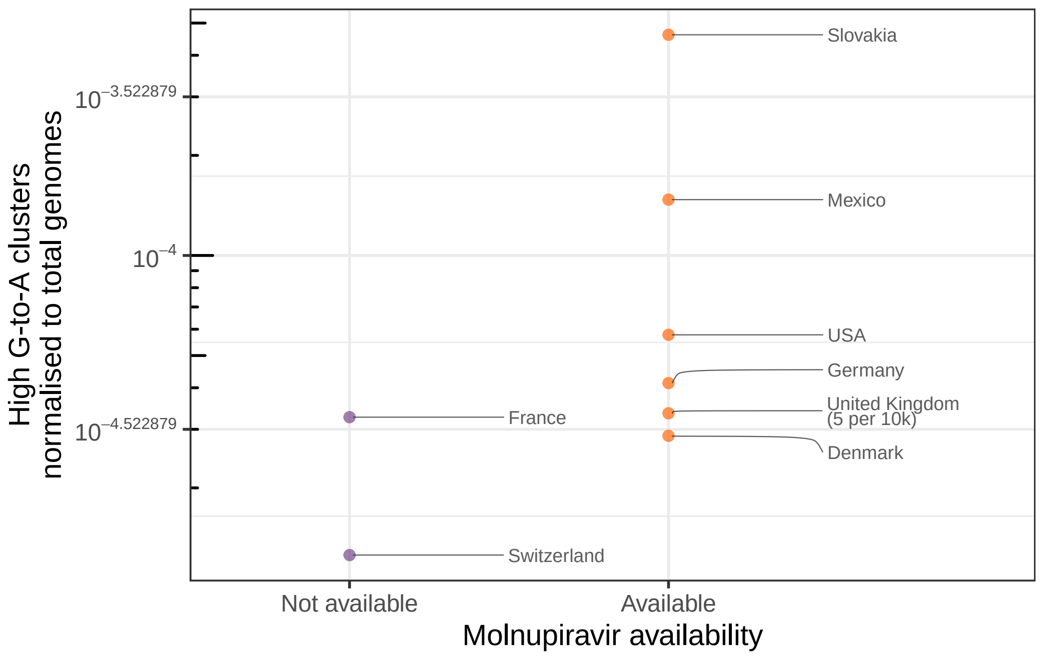
```
ggsave("country_scatter_big.pdf", width = 4, height = 3.5)

country_comp + geom_point(aes(color = approved))
```



```
recents <- data_nodes %>% filter(total_muts >= threshold_branch_length, consensus_year ==

recents$branch_type <- ifelse(recents$flagged, "High\nG\u00adto\u00adA", "Other")
recents$branch_type <- fct_relevel(recents$branch_type, "Other")



set.seed(339)
availability_dataset <- tallied %>%
  filter(country != "?", total_genomes > 10000) %>%
  mutate(usage = usage[country]) %>%
  mutate(usage = ifelse(is.na(usage), "", usage)) %>%
  mutate(approved = factor(as.character(approved), levels = c("Not available", "Available"
availability_plot <- ggplot(availability_dataset, aes(color = approved, x = approved, y =
  geom_point(alpha = 0.7) +
  scale_y_log10() +
  geom_text_repel(
    alpha = 0.8, force = 10, min.segment.length = 0, lineheight = .65, size = 2.5, color =
    # do not pull text toward the point at (0,0)
    max.time = 3,
    segment.square = TRUE,
    segment.size = 0.2,
    segment.curvature = 0.3,
    max.iter = 1e7, nudge_x = 0.5,
    max.overlaps = Inf,
    hjust = 0
  ) +
  theme_bw() +
  labs(x = "Molnupiravir availability", color = "Molnupiravir", y = "High G\u00adto\u00adA
  scale_color_manual(values = c("Not identified" = "gray", "Available" = darkorange, "Not
  theme(legend.position = "none") +
  annotation_logticks(sides = "l") +
  scale_x_discrete(
    expand = expansion(mult = c(0.5, 1.15))
  )
availability_plot <- availability_plot +
  scale_y_log10(labels = function(x) {
    expression_strs <- sapply(x, function(x_val) {
      if(is.na(x_val)){
        return(NA)
      }
      if (x_val == 0) {
        return("0")
      }
      log_val <- log10(x_val)
      paste0("10^", log_val)
    })
    parse(text = expression_strs)
  })
```

```
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```

```
  availability_plot
```

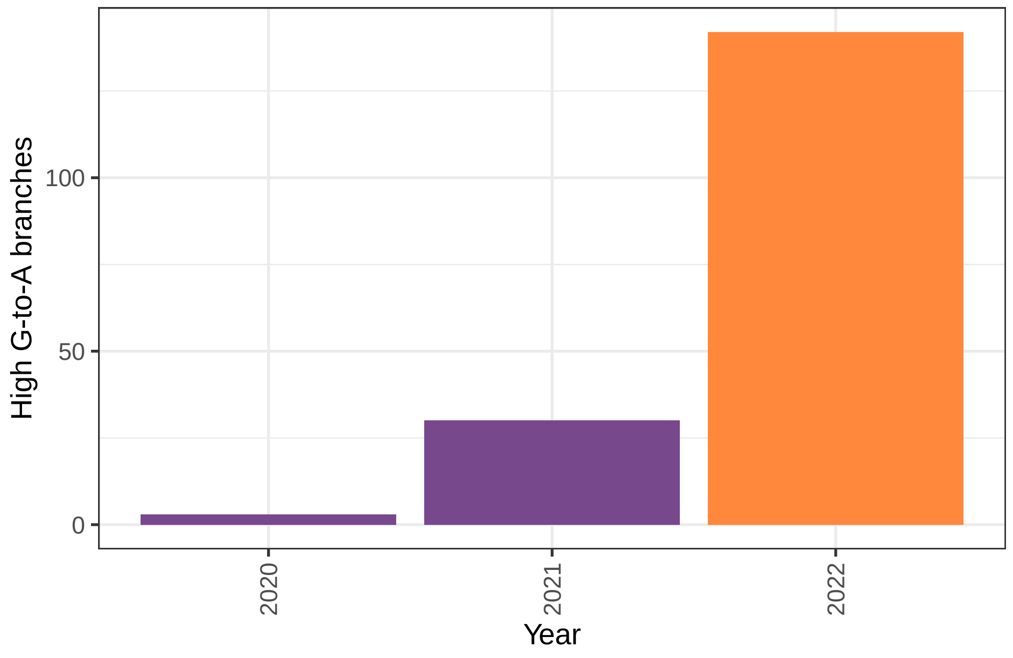


```r
t.test(log10(ga_branches + 0.5) / total_genomes ~ approved, data = availability_dataset)
```

```

	Welch Two Sample t-test

data:  log10(ga_branches + 0.5)/total_genomes by approved
t = -2.7802, df = 4.0238, p-value = 0.04948
alternative hypothesis: true difference in means between group Not available and group Availa
95 percent confidence interval:
 -5.127745e-05 -9.455042e-08
sample estimates:
mean in group Not available     mean in group Available
              -1.360382e-05                1.208218e-05
```

```r
ggsave("availability.pdf", width = 3.5, height = 3.5)

by_year <- data_nodes %>%
  filter(flagged, total_muts >= threshold_branch_length) %>%
  group_by(consensus_year) %>%
  tally()
```

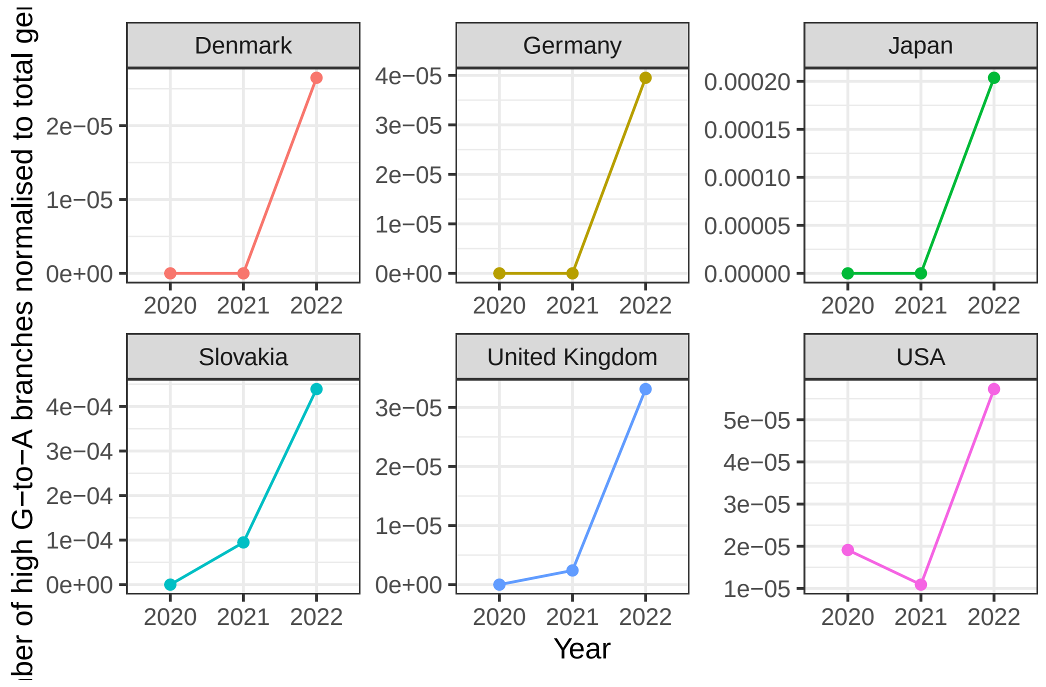
```r
by_year_plot <- ggplot(by_year %>% filter(consensus_year %in% c("2021", "2022", "2020")),
  geom_col() +
  theme_bw() +
  labs(x = "Year", y = "\nHigh G\u00adto\u00adA branches") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  scale_fill_manual(values = c(darkpurple, darkpurple, midorange)) +
  theme(legend.position = "none")
by_year_plot
```



```r
ggsave("byyearplot.pdf", width = 2, height = 3)
```

We also display data on timecourse where we normalise for total genome numbers, use a non log axis. This is particularly important for the open data.

```r
tallied_big <- tallied_big %>% mutate(p = (ga_branches) / total_genomes)

ggplot(tallied_big %>% filter(country %in% c( "United Kingdom", "USA", "Japan", "Germany",

  geom_line() +
  geom_point() +
  theme_bw() +
  facet_wrap(~country, scales = "free") +
  theme(legend.position = "none") +
  labs(y = "Number of high G-to-A branches normalised to total genomes", x = "Year")
```



```
ggsave("supp-countries_timeline.pdf", width = 7.5, height = 4.5)
```

Processing and analysis of existing genomic datasets

```
library(tidyverse)
tidyverse_conflicts()
```

```
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::collapse()        masks Biostrings::collapse(), IRanges::collapse()
x dplyr::combine()         masks BiocGenerics::combine()
x purrr::compact()         masks XVector::compact()
x dplyr::desc()            masks IRanges::desc()
x tidyr::expand()          masks S4Vectors::expand()
```

```
x dplyr::filter()          masks stats::filter()
x dplyr::first()           masks S4Vectors::first()
x kableExtra::group_rows() masks dplyr::group_rows()
x dplyr::lag()             masks stats::lag()
x ggplot2::Position()      masks BiocGenerics::Position(), base::Position()
x purrr::reduce()          masks IRanges::reduce()
x dplyr::rename()          masks S4Vectors::rename()
x dplyr::slice()           masks XVector::slice(), IRanges::slice()
```

```
nuc_genome_counts <- read_csv("./context_count.csv") %>% dplyr::rename(
  par = residue, context_before = residue_before, context_after = residue_after,
  genome_count = count
)
```

```
Rows: 64 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (3): residue_before, residue, residue_after
dbl (1): count

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
a <- read_csv("./molnupiravir_rescaled_samples.csv") %>% mutate(trial = "2", treat = "mov"
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
b <- read_csv("./MOV_rescaled_contexts_only.csv") %>% mutate(trial = "2", treat = "mov", c
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
c <- read_csv("./naive_rescaled_contexts_only.csv") %>% mutate(trial = "2", treat = "naive
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
d <- read_csv("./naive_rescaled_samples.csv") %>% mutate(trial = "2", treat = "naive", con
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
e <- read_csv("./agile_placebo_spectrum.csv") %>% mutate(trial = "1", treat = "naive", con
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
```

```
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
f <- read_csv("./agile_molnupiravir_spectrum.csv") %>% mutate(trial = "1", treat = "mov",
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
g <- read_csv("./BA.1_SBS_spectrum_Ruis.csv") %>% mutate(trial = "3", treat = "normal", co
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
long <- read_csv("./long_branch_spectrum_rescaled.csv") %>% mutate(trial = "4", treat = "l
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
specific <- read_csv("./molnupiravir_spectrum_specific.csv") %>% mutate(trial = "5", treat
```

```
Rows: 192 Columns: 2
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Substitution
dbl (1): Number_of_mutations

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
colors <- c("#3055a2", "#221f20", "#da4837", "#939598", "#3f8347", "#edb9c0", "#4a68af", "
my_levels <- c("C\u00adto\u00adA", "C\u00adto\u00adG", "C\u00adto\u00adT", "T\u00adto\u00a



combo <- bind_rows(a, b, c, d, e, f, g, long, specific) %>%
  filter(!contexts_only) %>%
  separate(Substitution, into = c("context_before", "par", "mut", "context_after"), sep =

data <- combo %>% mutate(mutation_type = factor(paste0(par, "\u00adto\u00ad", mut),
  levels = my_levels
))
```

For convenience to get the total number of each type of mutation we reverse MutTui's normalisations of context numbers.

```
totals <- data %>%
  group_by(trial) %>%
  summarise(total = sum(Number_of_mutations))


normed <- data %>%
  inner_join(totals) %>%
  mutate(Number_of_mutations = Number_of_mutations / total)
```

```
Joining with `by = join_by(trial)`
```

```
multipled <- normed %>%
  inner_join(nuc_genome_counts) %>%
  mutate(Number_of_mutations = Number_of_mutations * genome_count)
```

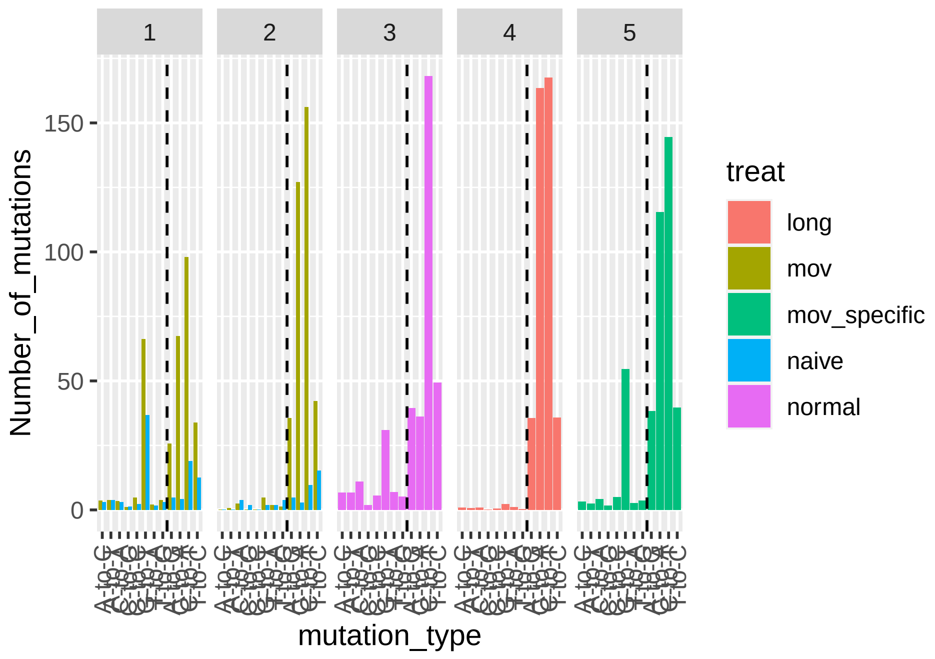
```
Joining with `by = join_by(context_before, par, context_after)`
```

```
just_class <- multipled %>%
  group_by(mutation_type, treat, trial) %>%
  summarise(Number_of_mutations = sum(Number_of_mutations))
```

```
`summarise()` has grouped output by 'mutation_type', 'treat'. You can override
using the `.groups` argument.
```

```
transversions <- c("A\u00adto\u00adC", "A\u00adto\u00adT", "C\u00adto\u00adA", "C\u00adto\

transitions <- c(
  "A\u00adto\u00adG", "G\u00adto\u00adA",
  "C\u00adto\u00adT",
  "T\u00adto\u00adC"
)


just_class <- just_class %>%
  mutate(mutation_type = fct_relevel(mutation_type,
    c(transversions, transitions),
    after = Inf
  ))
```

```
ggplot(just_class %>% filter() %>% arrange(mutation_type), aes(y = Number_of_mutations, x
  geom_col(position = "dodge") +
  facet_grid(. ~ trial) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  geom_vline(xintercept = 8.5, linetype = "dashed", color = "black")
```



```r
# Directory where your TSV files are
dir <- "./tsv_files"

# List all .tsv files in the directory
files <- list.files(path = dir, pattern = "\\.tsv$", full.names = TRUE)

# Read all files into a list of tibbles, adding the file name as a new column
big_df <- map_dfr(files, ~ read_tsv(.x, col_names = c("index", "par", "A", "C", "G", "T"))
```

```
Rows: 29812 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29694 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29617 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29624 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28827 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 25577 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28243 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
Rows: 28934 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28601 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 27536 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29625 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29398 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28785 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
```

```
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 18869 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29494 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29322 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 27603 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29686 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29849 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29664 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29348 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29836 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29796 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29638 Columns: 6
```

```
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29668 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29635 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29691 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29662 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28896 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
```

```
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29625 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29761 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29656 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28572 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29602 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29651 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29507 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28393 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 24314 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29243 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 28345 Columns: 6
-- Column specification --------------------------------------------------------
```

```
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29482 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29651 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29624 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 29663 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Rows: 27976 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (1): par
dbl (5): index, A, C, G, T
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
big_df <- big_df %>% mutate(total_depth = A + C + G + T)

big_df <- big_df %>% separate(file_name, into = c("treat", "patient", "timepoint"), sep =

long_df <- big_df %>%
  pivot_longer(
    cols = c(A, C, G, T),
    names_to = "base",
    values_to = "count"
  ) %>%
  filter(par != base, count > 0) %>%
  filter(count >= total_depth * 0.05, total_depth >= 100) %>%
  mutate(mutation_type = as.factor(paste0(par, "\u00adto\u00ad", base))) %>%
  filter(par != "N") %>%
  group_by(patient, index, par, base) %>%
  filter(row_number() == 1) # ensures we only count each mutation once



burdens <- long_df %>%
  filter(treat != "PAXLOVID") %>%
  group_by(treat, patient) %>%
  tally()

# Split mutation counts into two vectors based on treatment
naive_burden <- burdens %>%
  filter(treat == "NAIVE") %>%
  pull(n)
mov_burden <- burdens %>%
  filter(treat == "MOLNUPIRAVIR") %>%
  pull(n)

length(naive_burden)
```

```
[1] 5
```

```r
sd(naive_burden)
```

```
[1] 3.714835
```

```r
mean(naive_burden)
```

```
[1] 9.6
```

```r
length(mov_burden)
```

```
[1] 8
```

```r
sd(mov_burden)
```

```
[1] 63.19118
```

```r
mean(mov_burden)
```

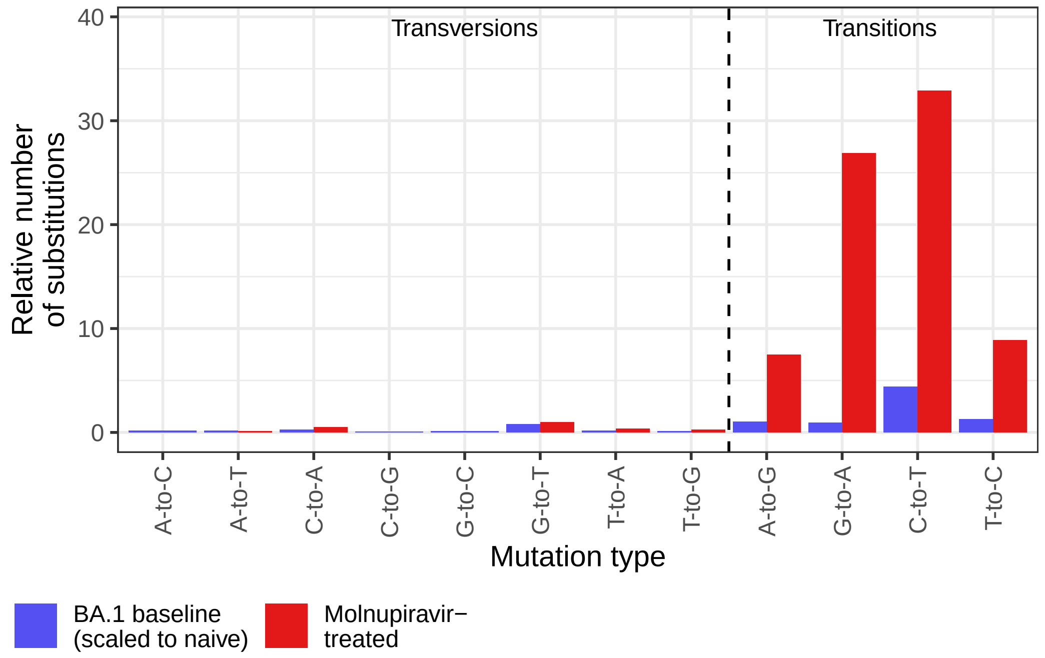
```
[1] 78.375
```

```r
n_patients_naive <- 5
n_patients_mov <- 8

ba1_basic <- just_class %>% filter(trial == 3)
ba1_normed <- ba1_basic %>% mutate(Number_of_mutations = Number_of_mutations * sum(naive_b
```

```r
lookup <- c("MOLNUPIRAVIR" = "mov", "NAIVE" = "normal")

mov_dataset <- long_df %>%
  group_by(mutation_type, treat) %>%
  tally() %>%
  filter(treat == "MOLNUPIRAVIR") %>%
  mutate(treat = "mov") %>%
  mutate(Number_of_mutations = n) %>%
  mutate(mutation_type = fct_relevel(mutation_type, c(transversions, transitions))) %>%
  mutate(Number_of_mutations = Number_of_mutations / n_patients_mov)
naive_dataset <- ba1_normed %>%
  mutate(treat = "normal") %>%
  mutate(mutation_type = fct_relevel(mutation_type, c(transversions, transitions))) %>%
  mutate(Number_of_mutations = Number_of_mutations / n_patients_naive)



relevant_dataset <- bind_rows(mov_dataset, naive_dataset)

relevant_dataset
```

```
# A tibble: 21 x 5
# Groups:   mutation_type [12]
   mutation_type treat      n Number_of_mutations trial
   <fct>         <chr>  <int>               <dbl> <chr>
 1 AtoG          mov       60               7.5   <NA>
 2 AtoT          mov        1               0.125 <NA>
 3 CtoA          mov        4               0.5   <NA>
 4 CtoT          mov      263              32.9   <NA>
 5 GtoA          mov      215              26.9   <NA>
 6 GtoT          mov        8               1     <NA>
 7 TtoA          mov        3               0.375 <NA>
 8 TtoC          mov       71               8.88  <NA>
 9 TtoG          mov        2               0.25  <NA>
10 CtoA          normal    NA               0.285 3
# i 11 more rows
```

```r
a <- ggplot(relevant_dataset, aes(y = Number_of_mutations, x = mutation_type, fill = treat
  geom_col(position = "dodge") +
  geom_vline(xintercept = 8.5, linetype = "dashed", color = "black") +
  scale_fill_manual(values = c(blue1, red), labels = c("BA.1 baseline\n(scaled to naive)",
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(fill = "") +
  annotate("text", x = 5, y = 39, label = "Transversions", size = 3) +
  labs(x = "Mutation type", y = "Relative number\nof substitutions") +
  annotate("text", x = 10.5, y = 39, label = "Transitions", size = 3) +
  theme(
    legend.position = "bottom",
    legend.justification = c(0, 1),
    legend.margin = margin(t = 0, r = 0, b = 0, l = -45, unit = "pt")
  )
a
```



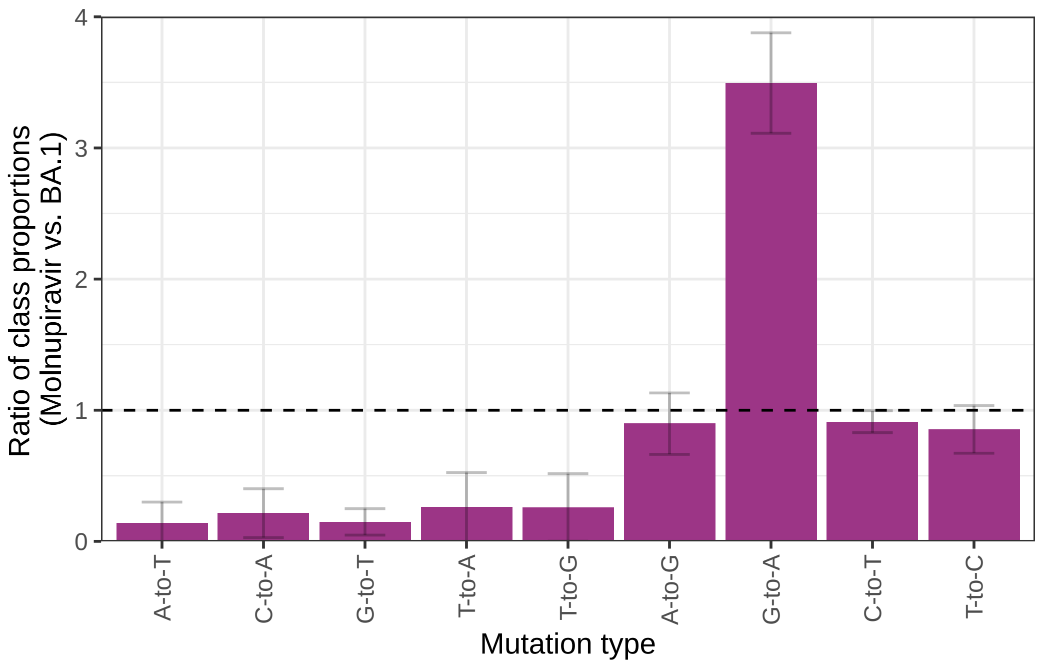
```r
naive_props <- naive_dataset %>%
  ungroup() %>%
  mutate(p = Number_of_mutations / sum(Number_of_mutations))
# The BA.1 spectrum props is based on so many mutations (hundreds of thousands) that we ca

naive_props
```

```
# A tibble: 12 x 5
   mutation_type treat  trial Number_of_mutations       p
   <fct>         <chr>  <chr>               <dbl>   <dbl>
 1 CtoA          normal 3                  0.285  0.0297
 2 CtoG          normal 3                  0.0489 0.00509
 3 CtoT          normal 3                  4.39   0.457
 4 TtoA          normal 3                  0.178  0.0185
 5 TtoC          normal 3                  1.29   0.134
 6 TtoG          normal 3                    0.135  0.0141
 7 GtoT          normal 3                    0.806  0.0840
 8 GtoC          normal 3                    0.143  0.0149
 9 GtoA          normal 3                    0.946  0.0985
10 AtoT          normal 3                    0.176  0.0183
11 AtoG          normal 3                    1.03   0.107
12 AtoC          normal 3                    0.172  0.0180
```

```
mov_for_props <- long_df %>%
  filter(treat == "MOLNUPIRAVIR") %>%
  ungroup()

resample_and_calc_ratios <- function(long_df) {
  resampled <- sample_n(mov_for_props, size = nrow(mov_for_props), replace = TRUE)
  props <- resampled %>%
    group_by(mutation_type) %>%
    tally() %>%
    mutate(p = n / sum(n))
  together <- inner_join(props, naive_props, by = "mutation_type") %>% mutate(ratio = p.x
  return(together %>% select(mutation_type, ratio))
}

bootstrap_count <- 100
bootstrap_ratios <- list()

for (i in 1:bootstrap_count) {
  bootstrap_ratios[[i]] <- resample_and_calc_ratios(long_df)
}

# Convert list to data frame
bootstrap_ratios_df <- bind_rows(bootstrap_ratios)
bootstrap_ratios_df
```

```
# A tibble: 838 x 2
   mutation_type ratio
   <fct>         <dbl>
 1 AtoG          0.831
 2 AtoT          0.348
 3 CtoA          0.161
 4 CtoT          0.872
 5 GtoA          3.64
 6 GtoT          0.247
 7 TtoA          0.172
 8 TtoC          0.843
 9 TtoG          0.340
10 AtoG          0.906
# i 828 more rows
```

```
proportions_wider <- bootstrap_ratios_df %>%
  group_by(mutation_type) %>%
  summarise(sd = sd(ratio), ratio = mean(ratio))



b <- ggplot(proportions_wider %>% mutate(mutation_type = fct_relevel(mutation_type, c(tran
  geom_col(position = "dodge", fill = "#9C3586") +
  scale_y_continuous(expand = c(0, 0)) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  labs(x = "Mutation type", y = "Ratio of class proportions   \n(Molnupiravir vs. BA.1)")
  geom_hline(yintercept = 1, linetype = "dashed", color = "black") +
  geom_errorbar(alpha = 0.25, width = 0.4) +
  coord_cartesian(ylim = c(0, 4))

b
```



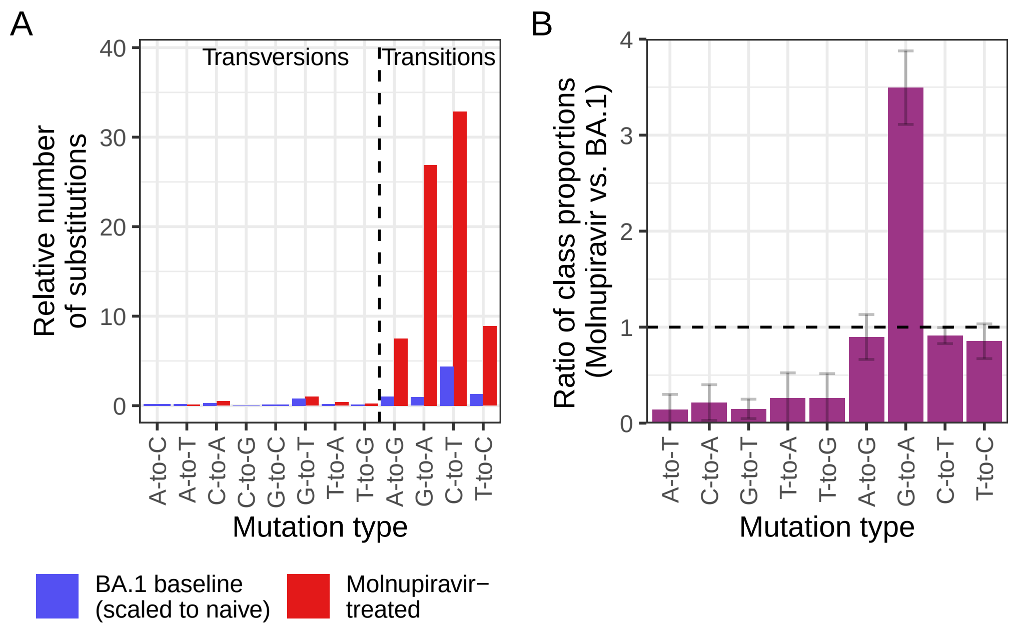
```
proportions
```

```
function (x, margin = NULL)
{
    if (length(margin))
        sweep(x, margin, marginSums(x, margin), `/`, check.margin = FALSE)
    else x/sum(x)
}
<bytecode: 0x3d1a3a630>
<environment: namespace:base>
```

```
library(patchwork)
```

```

Attaching package: 'patchwork'

The following object is masked from 'package:cowplot':

    align_plots
```

```
ab <- a + b + plot_annotation(tag_levels = "A")
ab
```



```
ggsave("a.pdf", a, width = 3, height = 3.5)
ggsave("b.pdf", b, width = 3, height = 3.5)


mov_props <- mov_for_props %>%
  group_by(mutation_type) %>%
  tally() %>%
  mutate(p = n / sum(n))


perform_sim <- function(n_sample, relevant_props) {
  # Set the number of iterations and the sample size
  n_iterations <- 10000
  15

  # Initialize a vector to hold the result of each iteration
  result <- vector(mode = "logical", length = n_iterations)


  # Run the simulation
  for (i in 1:n_iterations) {
    # Sample mutation types according to their probabilities
    sample_mutation <- sample(relevant_props$mutation_type, size = n_sample, replace = TRU

    # Calculate the proportions of each mutation type in the sample
    sample_prop <- table(sample_mutation) / n_sample

    # Calculate the transition proportion
    transition_prop <- sum(sample_prop[c("C\u00adto\u00adT", "G\u00adto\u00adA", "T\u00adt

    # Check whether the proportions meet the thresholds
    result[i] <- (sample_prop["C\u00adto\u00adT"] > CtoTthreshold & sample_prop["G\u00adto
  }

  # Calculate the proportion of iterations that meet the condition
  proportion <- sum(result) / n_iterations
  proportion
  return(proportion)
}
# Define the mutation counts to consider
mutations <- c(10,11,12,13,14, 15, 20)

# Initialize vectors to hold results
sensitivity <- numeric(length(mutations))
specificity <- numeric(length(mutations))

# Loop over each mutation count
for (i in seq_along(mutations)) {
  # Compute sensitivity and specificity
  sensitivity[i] <- perform_sim(mutations[i], mov_props)
  specificity[i] <- 1 - perform_sim(mutations[i], naive_props)
}

# Create a data frame with the results
results <- data.frame(
  Mutations = mutations,
  Sensitivity = sensitivity,
  Specificity = specificity
)

# Print the results
```

```r
print(results)
```

```
  Mutations Sensitivity Specificity
1        10      0.4709      0.9861
2        11      0.6753      0.9605
3        12      0.5578      0.9892
4        13      0.6367      0.9876
5        14      0.6955      0.9885
6        15      0.7072      0.9860
7        20      0.6310      0.9981
```

```r
library(ggpmisc)
```

```
Loading required package: ggpp


Attaching package: 'ggpp'

The following object is masked from 'package:ggplot2':

    annotate
```

```r
library(ggtext)

normed
```

```
# A tibble: 1,344 x 10
   context_before par   mut   context_after Number_of_mutations trial treat
   <chr>          <chr> <chr> <chr>                       <dbl> <chr> <chr>
 1 A              C     A     A                         0.00147 2     mov
 2 A              C     A     C                         0       2     mov
 3 A              C     A     G                         0       2     mov
 4 A              C     A     T                         0       2     mov
 5 C              C     A     A                         0       2     mov
 6 C              C     A     C                         0       2     mov
 7 C              C     A     G                         0       2     mov
 8 C              C     A     T                         0       2     mov
 9 G              C     A     A                          0       2     mov
10 G              C     A     C                          0.00317 2     mov
# i 1,334 more rows
# i 3 more variables: contexts_only <lgl>, mutation_type <fct>, total <dbl>
```

```
trial2 <- normed %>%
  filter((treat == "mov" & trial == "2")) %>%
  group_by(mutation_type) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(Number_of_mutations))
trial1 <- normed %>%
  filter((treat == "mov" & trial == "1")) %>%
  group_by(mutation_type) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(Number_of_mutations))
long <- normed %>%
  filter((trial == "4")) %>%
  select(-treat, -total, -contexts_only, -trial) %>%
  group_by(mutation_type) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(Number_of_mutations))
normal <- normed %>%
  filter((trial == "3")) %>%
  select(-treat, -total, -contexts_only, -trial) %>%
  group_by(mutation_type) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(Number_of_mutations))
normal
```

```
# A tibble: 192 x 6
# Groups:   mutation_type [12]
   context_before par   mut   context_after Number_of_mutations mutation_type
   <chr>          <chr> <chr> <chr>                       <dbl> <fct>
 1 A              C     A     A                          0.0423 CtoA
 2 A              C     A     C                          0.0618 CtoA
 3 A              C     A     G                          0.0655 CtoA
 4 A              C     A     T                          0.0737 CtoA
 5 C              C     A     A                          0.0922 CtoA
 6 C              C     A     C                          0.0506 CtoA
 7 C              C     A     G                          0.125  CtoA
 8 C              C     A     T                          0.0994 CtoA
 9 G              C     A     A                          0.0500 CtoA
10 G              C     A     C                          0.0386 CtoA
# i 182 more rows
```

```
merged <- normed %>%
  group_by(context_before, context_after, par, mut, treat, mutation_type) %>%
  summarise(Number_of_mutations = mean(Number_of_mutations)) %>%
  filter(treat == "mov")
```

```
`summarise()` has grouped output by 'context_before', 'context_after', 'par',
'mut', 'treat'. You can override using the `.groups` argument.
```

```
long_v_merged <- inner_join(long %>% rename(v1 = Number_of_mutations), merged %>% rename(v
```

```
Joining with `by = join_by(context_before, par, mut, context_after,
mutation_type)`
```

```
t1_v_merged <- inner_join(long %>% rename(v1 = Number_of_mutations), trial1 %>% rename(v2
```

```
Joining with `by = join_by(context_before, par, mut, context_after,
mutation_type)`
```

```
t2_v_merged <- inner_join(long %>% rename(v1 = Number_of_mutations), trial2 %>% rename(v2
```

```
Joining with `by = join_by(context_before, par, mut, context_after,
mutation_type)`
```

```
cosine_similarity_compute_fun <- function(data, ...) {
  force(data)

  x <- data$x
  y <- data$y

  similarity <- sum(x * y) / (sqrt(sum(x^2)) * sqrt(sum(y^2)))


  data.frame(x = 0, y = .11, label =paste0("c=",round(similarity, 3) ),color="black",hjust
}
```

```
StatCosineSimilarity <- ggproto(
  "StatCosineSimilarity",
  Stat,
  compute_group = cosine_similarity_compute_fun,
  required_aes = c("x", "y")
)

stat_cosine_similarity <- function(mapping = NULL, data = NULL, geom = "text",
                                   position = "identity", na.rm = FALSE, show.legend = NA,
                                   inherit.aes = TRUE, ...) {
  layer(
    stat = StatCosineSimilarity, data = data, mapping = mapping, geom = geom,
    position = position, show.legend = show.legend, inherit.aes = inherit.aes,
    params = list(na.rm = na.rm, ...)
  )
}


long_v_normal <- inner_join(long %>% rename(v1 = Number_of_mutations), normal %>% rename(v
```

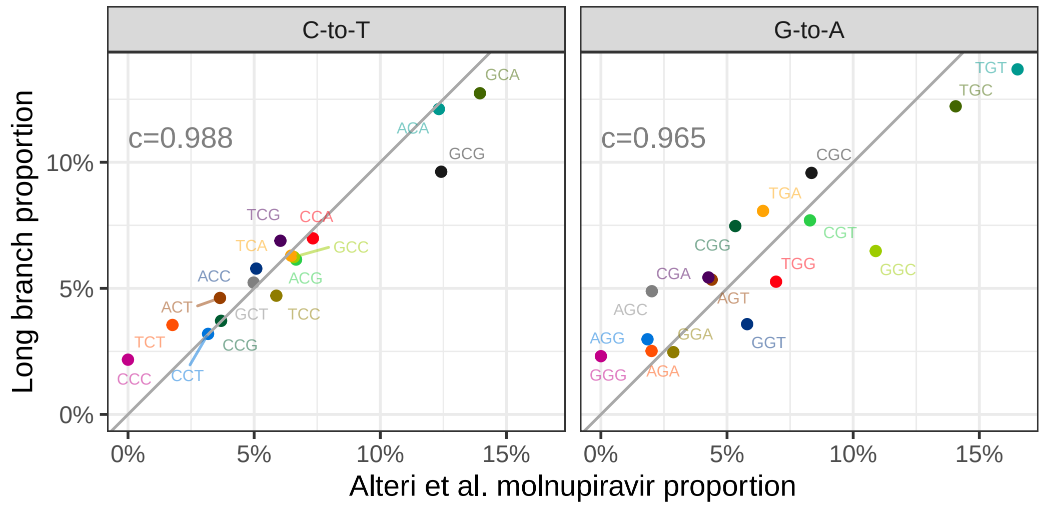
```
Joining with `by = join_by(context_before, par, mut, context_after,
mutation_type)`
```

```
oneset <- unique((t2_v_merged %>% filter(mutation_type %in% c("G\u00adto\u00adA")))$contex


library(pals)
```

```

Attaching package: 'pals'

The following object is masked from 'package:Biostrings':

    alphabet
```

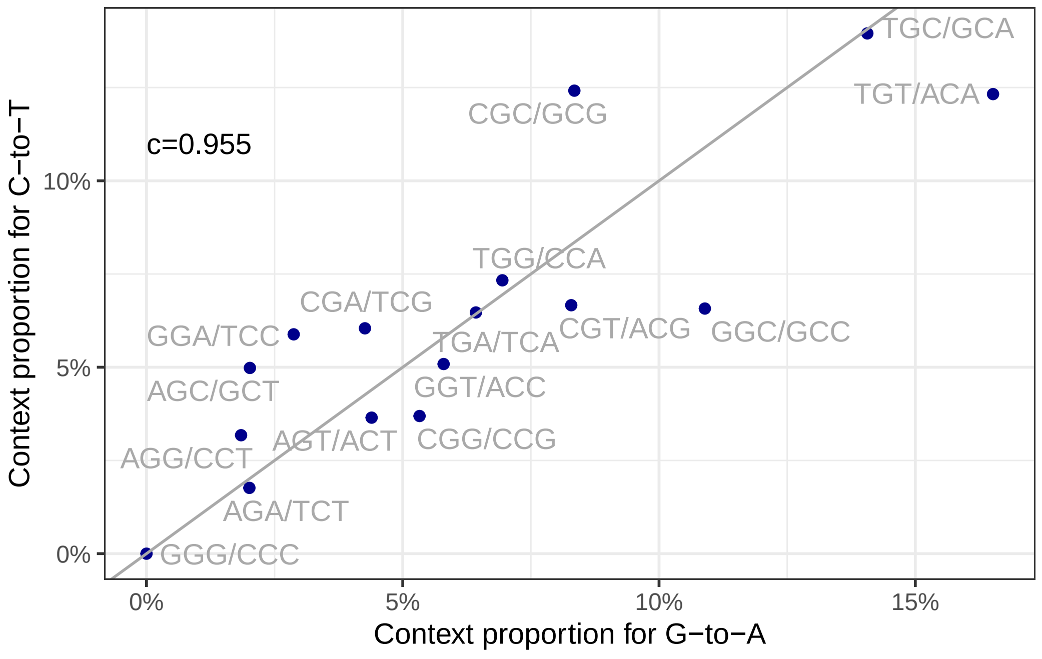
```
colors_16 <- unname(c(alphabet()[26:26], alphabet()[9], alphabet()[2:7], alphabet()[11:15]
reverse_complement <- function(context) {
  rev_nucleotide <- function(x) {
    switch(x,
      "A" = "T",
      "T" = "A",
      "C" = "G",
      "G" = "C",
      x
    )
  }
  rev_context <- sapply(strsplit(context, "")[[1]], rev_nucleotide)
  paste(rev(rev_context), collapse = "")
}

context_colors <- c()
for (i in 1:length(oneset)) {
  context <- oneset[i]
  reverse_context <- reverse_complement(context)

  if (!context %in% names(context_colors)) {
    context_colors[context] <- colors_16[i]
  }

  if (!reverse_context %in% names(context_colors)) {
    context_colors[reverse_context] <- colors_16[i]
  }
}


scatters <- ggplot(t2_v_merged %>% filter(mutation_type %in% c("G\u00adto\u00adA", "C\u00a
  geom_point() +
  labs(x = "Alteri et al. molnupiravir proportion", y = "Long branch proportion") +
  facet_wrap(~mutation_type, ncol = 2) +
  theme_bw() + stat_cosine_similarity()+
  coord_fixed(xlim = c(0, NA), ylim = c(0, NA)) +
  geom_abline(
    intercept = 0, slope = 1, # linetype = "black",
    color = "darkgray"
  ) +
  geom_text_repel(alpha = 0.5, size = 2, max.overlaps = Inf, force = 10) +
  scale_x_continuous(labels = scales::percent) +
  scale_y_continuous(labels = scales::percent) +
  scale_color_manual(values = context_colors) +
  theme(legend.position = "none")


scatters
```

```
Warning: The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```



```
start <- trial2 %>%
  mutate(context_full = paste0(context_before, par, context_after)) %>%
  mutate(rc_context = sapply(context_full, reverse_complement))
GtoA <- start %>% filter(mutation_type == "G\u00adto\u00adA")
CtoT <- start %>% filter(mutation_type == "C\u00adto\u00adT")
joint <- inner_join(GtoA, CtoT, by = c("context_full" = "rc_context"))
```

```
comp <- ggplot(joint, aes(x = Number_of_mutations.x, y = Number_of_mutations.y, label = pa
  geom_point(color = "darkblue") +
  theme_bw() +
  geom_abline(
    intercept = 0, slope = 1, # linetype = "black",
    color = "darkgray"
  ) + stat_cosine_similarity() +
  geom_text_repel(color = "darkgray") +
  scale_x_continuous(labels = scales::percent) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = "Context proportion for G-to-A", y = "Context proportion for C-to-T")

comp
```



```
names(colors) <- my_levels

other_colors <- c("A" = "#111111", "C" = "#555555", "G" = "#999999", "T" = "#cccccc")
all_colors <- c(colors, other_colors)

colors_new <- all_colors
colors_new["A\u00adto\u00adG"] <- "#5c4987"
```

```
colors_new["T\u00adto\u00adC"] <- "#5377ad"

create_scatter_plot <- function(df, x_label, file_name) {
  plot <- ggplot(df %>%
    filter(mutation_type %in% c("G\u00adto\u00adA", "C\u00adto\u00adT", "A\u00adto\u00adG"
    mutate(label = context_full), aes(x = v2, y = v1, label = label, color = mutation_type
    geom_point() +
    labs(x = x_label, y = "Long branch proportion") +
    facet_wrap(~mutation_type, ncol = 2) +
    theme_bw() +
    stat_cosine_similarity() +
    coord_fixed(xlim = c(0, NA), ylim = c(0, NA)) +
    # geom_abline(intercept = 0, slope = 1, color = "darkgray")+
    geom_text_repel(alpha = 0.5, size = 2, max.overlaps = Inf, force = 10) +
    scale_x_continuous(labels = scales::percent) +
    scale_y_continuous(labels = scales::percent) +
    scale_color_manual(values = colors_new) +
    theme(legend.position = "none") +

    geom_smooth(method = "lm", se = FALSE, color = "darkgray", fullrange = F, size = 1)



  return(plot)
}

# Call the function three times with different dataframes and labels
scatters_supplemental <- create_scatter_plot(t2_v_merged, "Alteri et al. molnupiravir prop
```

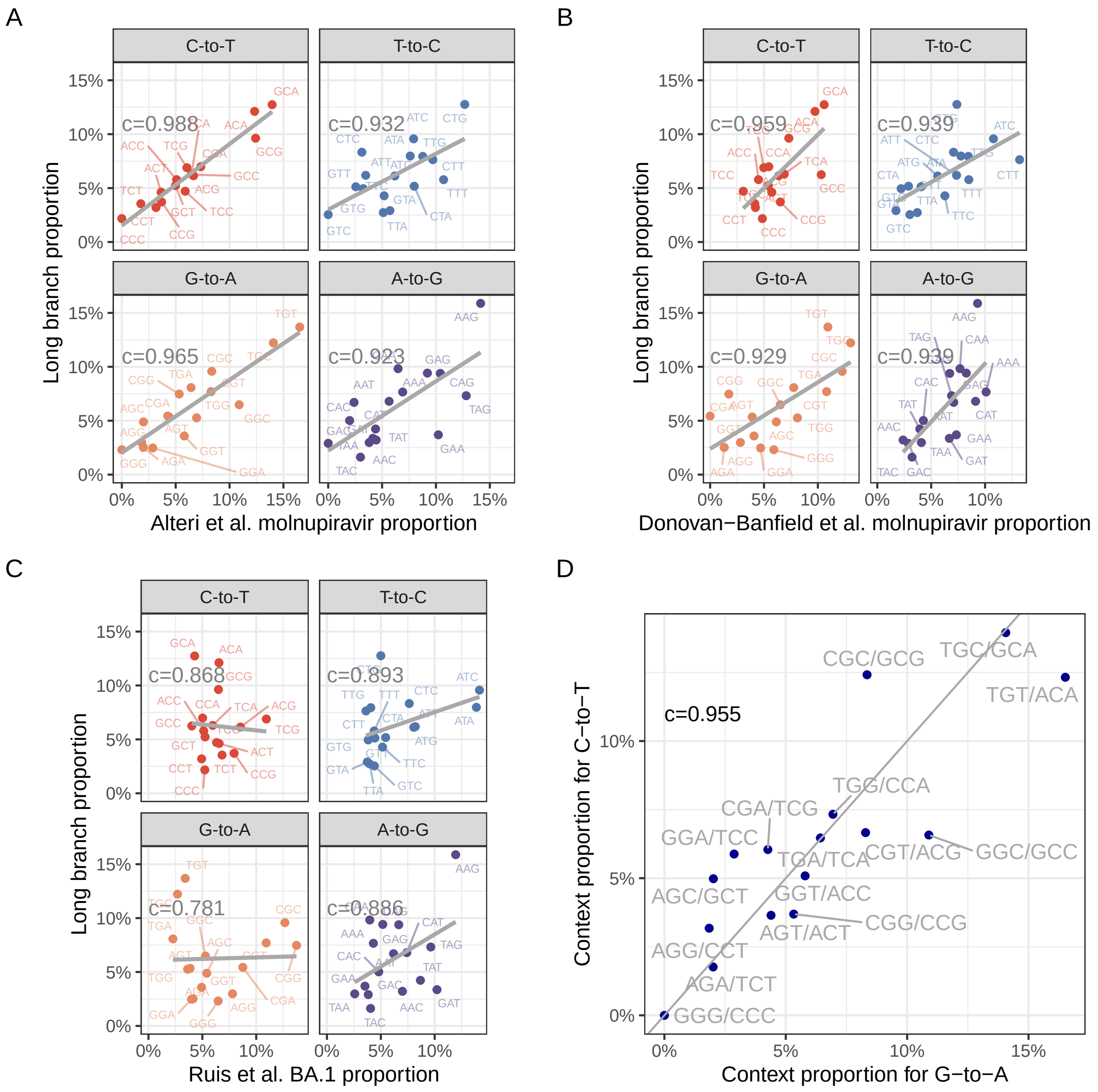
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
scatters_normal <- create_scatter_plot(long_v_normal, "Ruis et al. BA.1 proportion", "scat
scatters_supplemental2 <- create_scatter_plot(t1_v_merged, "Donovan-Banfield et al. molnup

scatters_supplemental + scatters_supplemental2 + scatters_normal + comp + plot_annotation(
```

```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: Duplicated aesthetics after name standardisation: colour

`geom_smooth()` using formula = 'y ~ x'

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
```

```
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?


Warning: Duplicated aesthetics after name standardisation: colour


`geom_smooth()` using formula = 'y ~ x'


Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?


Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?


Warning: Duplicated aesthetics after name standardisation: colour
```



```
ggsave("supplemental_scatters.pdf")
```

```
Saving 8 x 8 in image
`geom_smooth()` using formula = 'y ~ x'

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
```

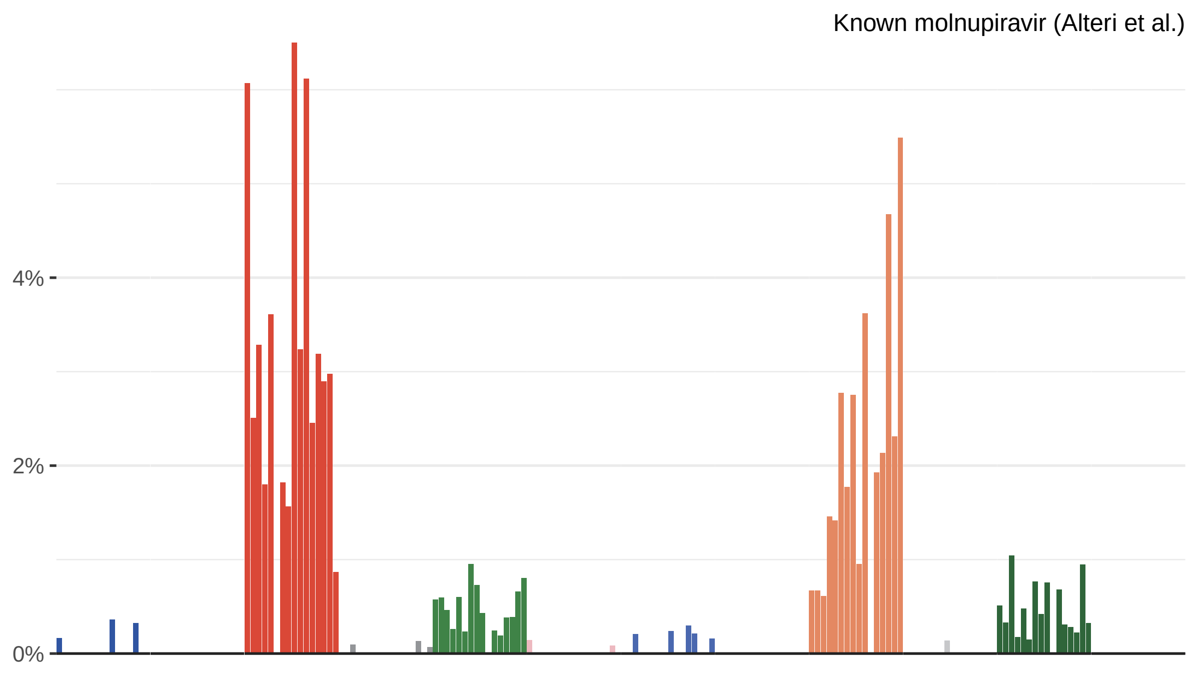
```
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: Duplicated aesthetics after name standardisation: colour

`geom_smooth()` using formula = 'y ~ x'

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
```

```
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: Duplicated aesthetics after name standardisation: colour

`geom_smooth()` using formula = 'y ~ x'

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: label
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?

Warning: Duplicated aesthetics after name standardisation: colour
```

```
plot_spectrum <- function(data, globalmax = 0, limit = 0.1, extra_axis = FALSE, title = ""
  if (!globalmax) {
    globalmax <- max(data$Number_of_mutations)
  }
  my_levels <- sort(unique(paste0(data$context_before, data$context_after)))

  data$level <- factor(paste0(data$context_before, data$context_after), levels = my_levels
```

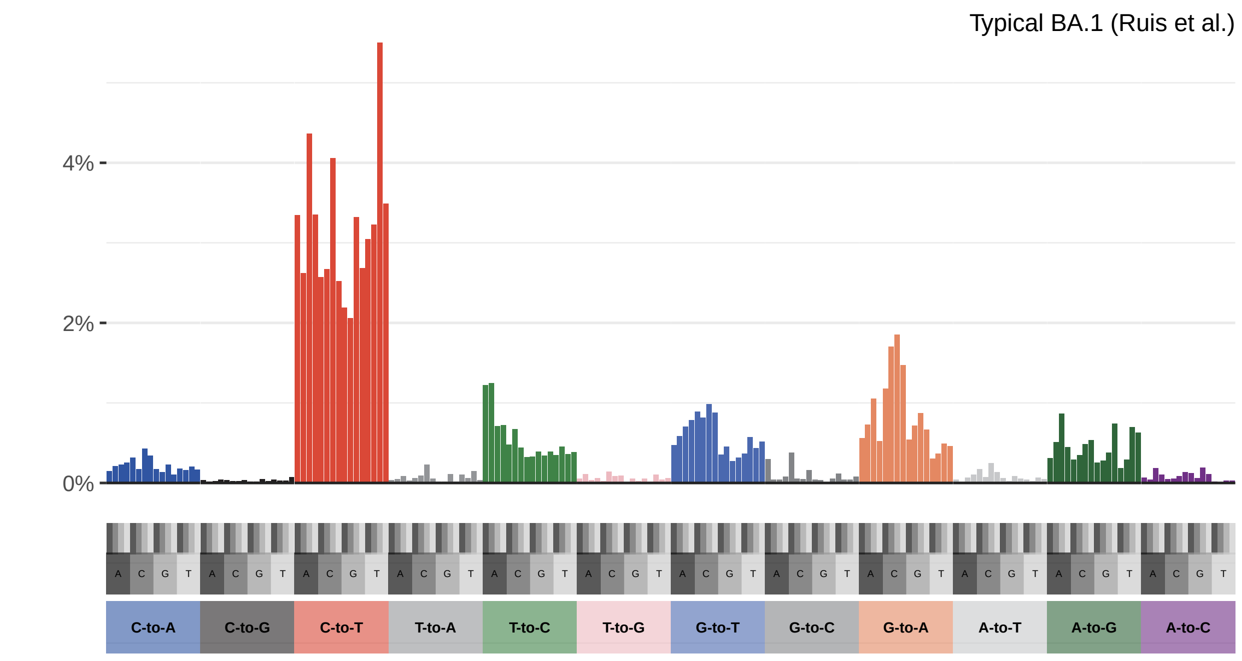
```r
data$levelno <- as.numeric(data$level)

precedings <- data %>%
  group_by(mutation_type, context_before) %>%
  summarise(levelno = mean(levelno))

offset <- 0.05

facet_style_labels <- data %>%
  group_by(mutation_type) %>%
  tally() %>%
  mutate(x = mean(data$levelno), y = -0.13 * globalmax - offset * globalmax)


p <- ggplot(data, aes(x = levelno, y = `Number_of_mutations`, fill = mutation_type)) +
  facet_wrap(~mutation_type, nrow = 1, strip.position = "top") +
  theme_bw() +
  geom_col() +
  theme(panel.spacing = unit(0, "lines"), panel.border = element_blank()) +
  geom_bar(stat = "identity") +
  theme( # remove the vertical grid lines
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank()
    # explicitly set the horizontal lines (or they will disappear too)
    # panel.grid.major.y = element_line( size=.2, color="black" )
  ) +
  theme(legend.position = "none") +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank()
  ) +
  scale_x_continuous(expand = c(0, 0)) +
  theme(
    strip.background = element_blank(),
    strip.text.x = element_blank()
  ) +
  scale_fill_manual(values = all_colors) +
  scale_y_continuous(labels = scales::percent, breaks = c(0, 0.02, 0.04), limits = c(NA,
  labs(y = " ", title = title) +
  theme(plot.title = element_text(margin = margin(t = 0, b = -10), size = 10, hjust = 1)
    geom_hline(yintercept = 0, color = "#222222")

  if (extra_axis) {
    p <- p + geom_rect(data = data, aes(xmin = levelno - 0.5, xmax = levelno + 0.5, ymin =

      geom_tile(data = precedings, aes(x = levelno, y = -.09 * .7 * globalmax - globalmax
      geom_text(data = precedings, aes(x = levelno, y = -.09 * .7 * globalmax - globalmax
      geom_tile(data = facet_style_labels, aes(label = mutation_type, fill = mutation_type
      geom_text(data = facet_style_labels, aes(label = mutation_type, label = mutation_typ
  }
  print(p)
  return(p)
}


trial2 <- normed %>%
  filter((treat == "mov" & trial == "2")) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(`Number_of_mutations`))
ba1 <- normed %>%
  filter((trial == "3")) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(`Number_of_mutations`))

long <- normed %>%
  filter((trial == "4")) %>%
  mutate(Number_of_mutations = Number_of_mutations / sum(`Number_of_mutations`))


p_t2 <- plot_spectrum(trial2, 0.1, 0.065, FALSE, "Known molnupiravir (Alteri et al.)")
```

```
`summarise()` has grouped output by 'mutation_type'. You can override using the
`.groups` argument.
```



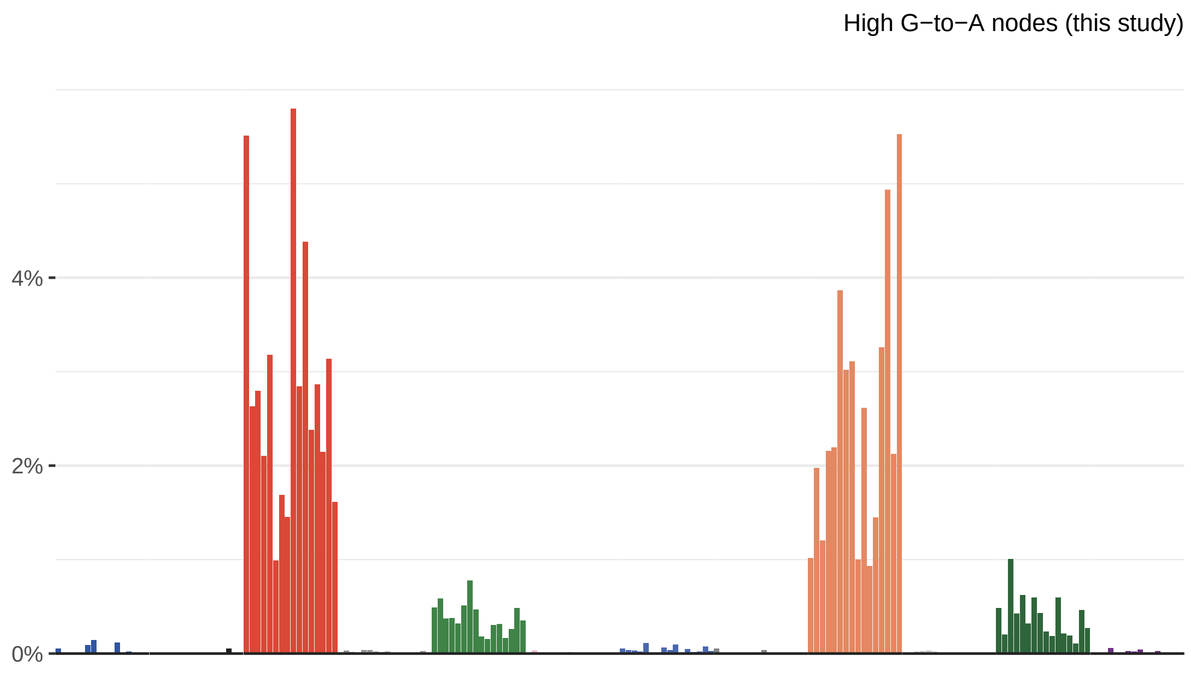
```r
p_ba1 <- plot_spectrum(ba1, 0.1, 0.055, TRUE, "Typical BA.1 (Ruis et al.)")
```

```
`summarise()` has grouped output by 'mutation_type'. You can override using the
`.groups` argument.

Warning in geom_tile(data = facet_style_labels, aes(label = mutation_type, :
Ignoring unknown aesthetics: label

Warning: Duplicated aesthetics after name standardisation: label
Duplicated aesthetics after name standardisation: label
```

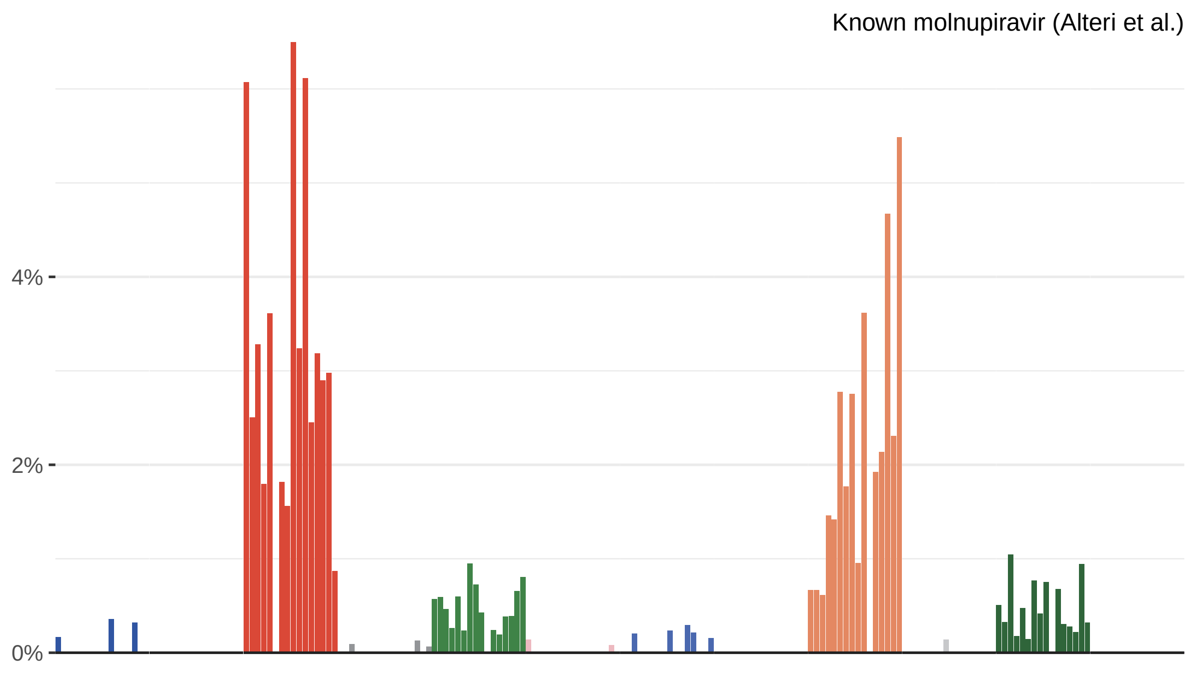


```
p_long <- plot_spectrum(long, 0.1, 0.065, FALSE, "High G-to-A nodes (this study)")
```

```
`summarise()` has grouped output by 'mutation_type'. You can override using the
`.groups` argument.
```

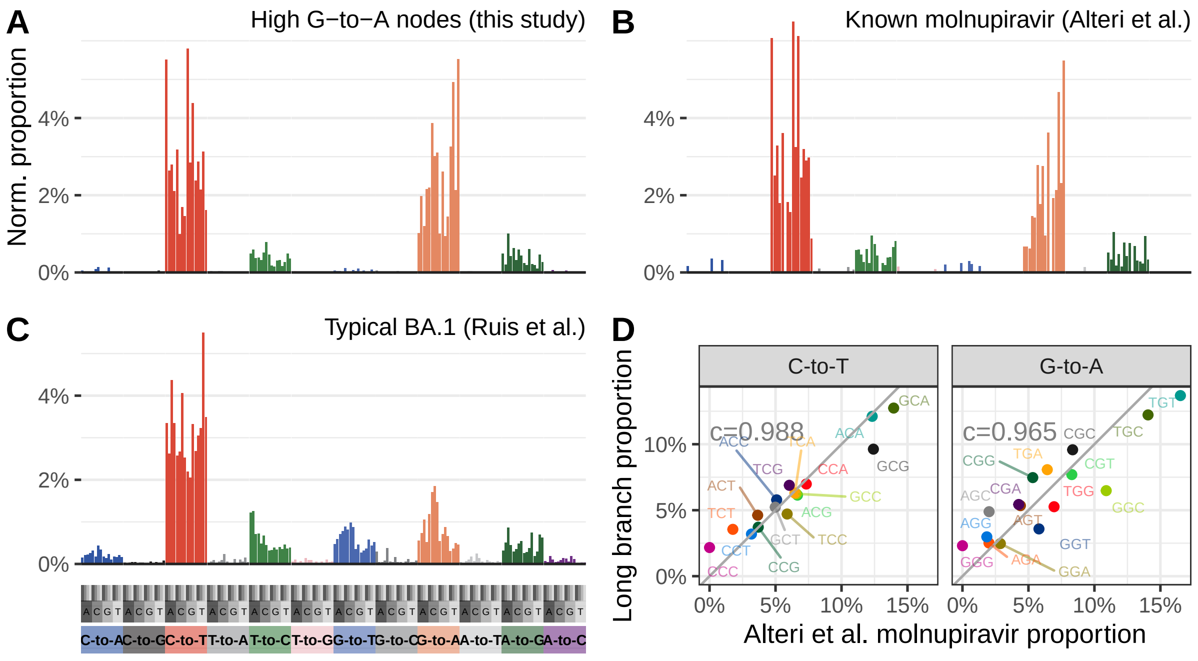

High G−to−A nodes (this study)
4%
2%
0%

```
p_t2
```


Known molnupiravir (Alteri et al.)
4%
2%
0%

```
stacked <- (p_ba1 / p_t2 / p_long)


plot_grid(p_long + labs(y = "Norm. proportion"), p_t2, p_ba1, (scatters), labels = c("A",
```

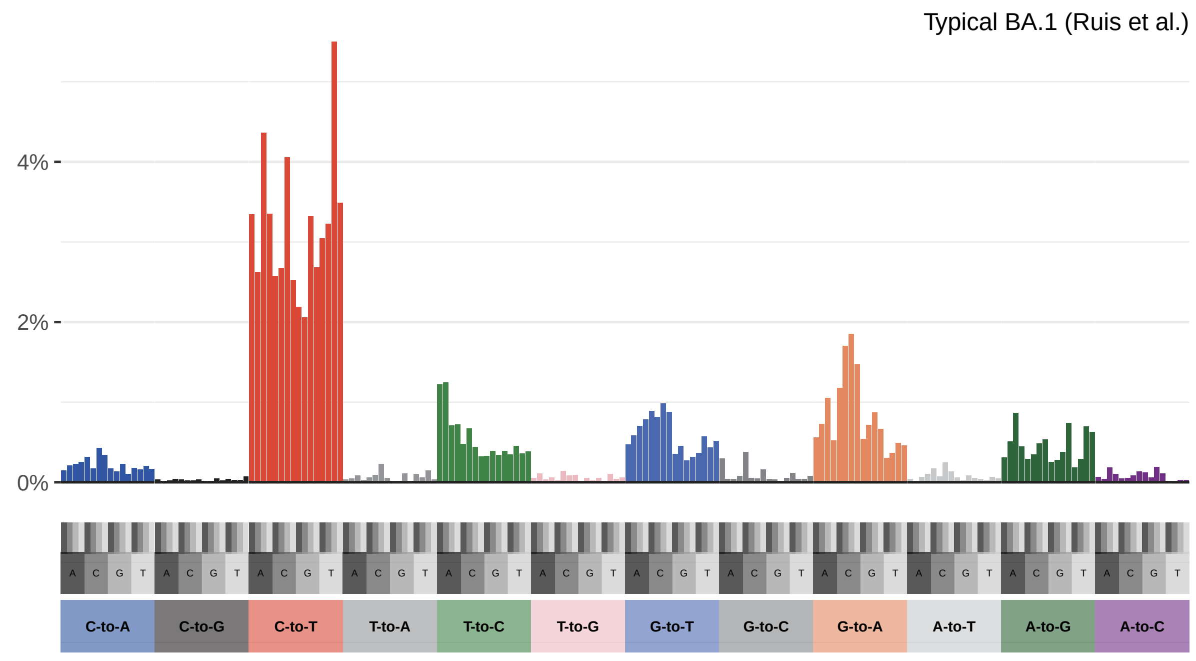
```
Warning: Duplicated aesthetics after name standardisation: label

Warning: The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```



```
ggsave("t2vlong.pdf", width = 8, height = 4)
```

```
# plot_grid(p_long +labs(y="Norm. proportion") , p_t2  , p_ba1 , (scatters) , rel_heights

ggsave("t2vlong-present.pdf", width = 8, height = 4)
ggsave("spectra.pdf", width = 8, height = 4)

p_ba1
```

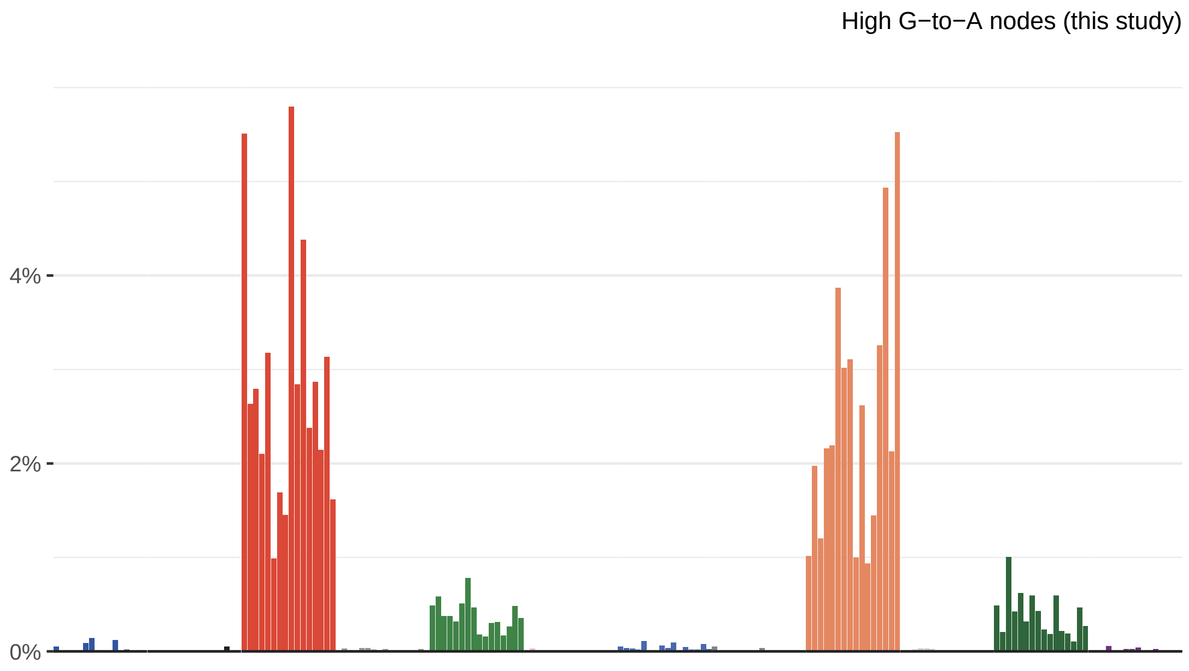
```
Warning: Duplicated aesthetics after name standardisation: label
```



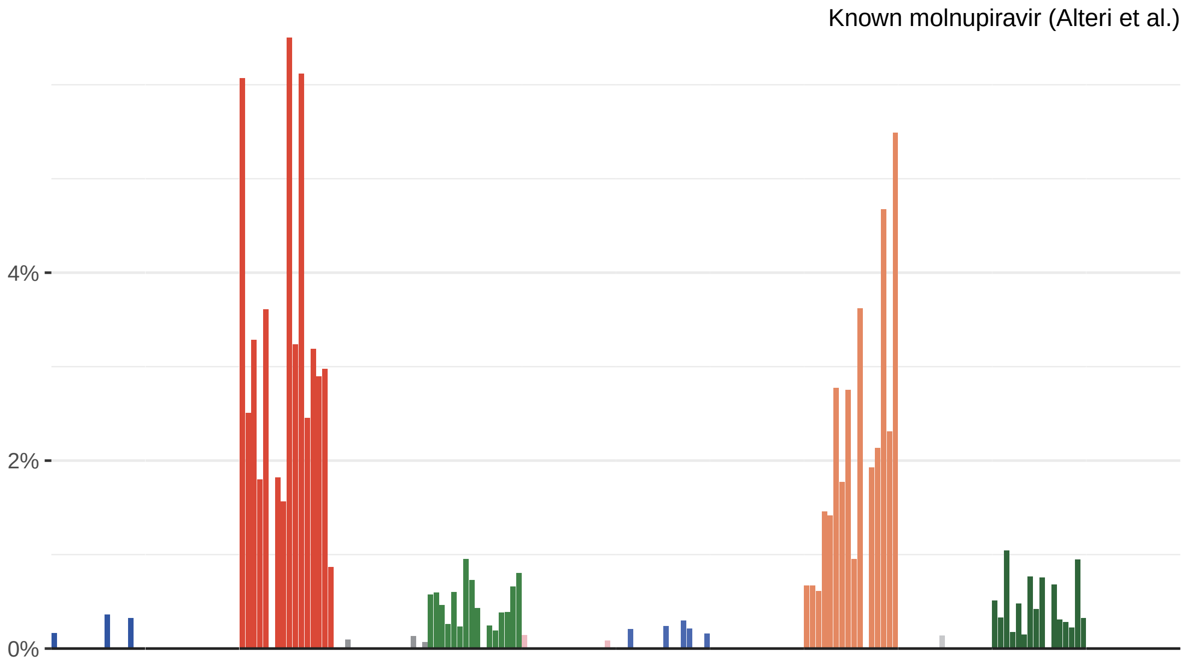
```
ggsave("p_ba1.pdf", width = 0.5 * 10, height = 0.5 * 4.5)
```

```
Warning: Duplicated aesthetics after name standardisation: label
```

```
p_long
```

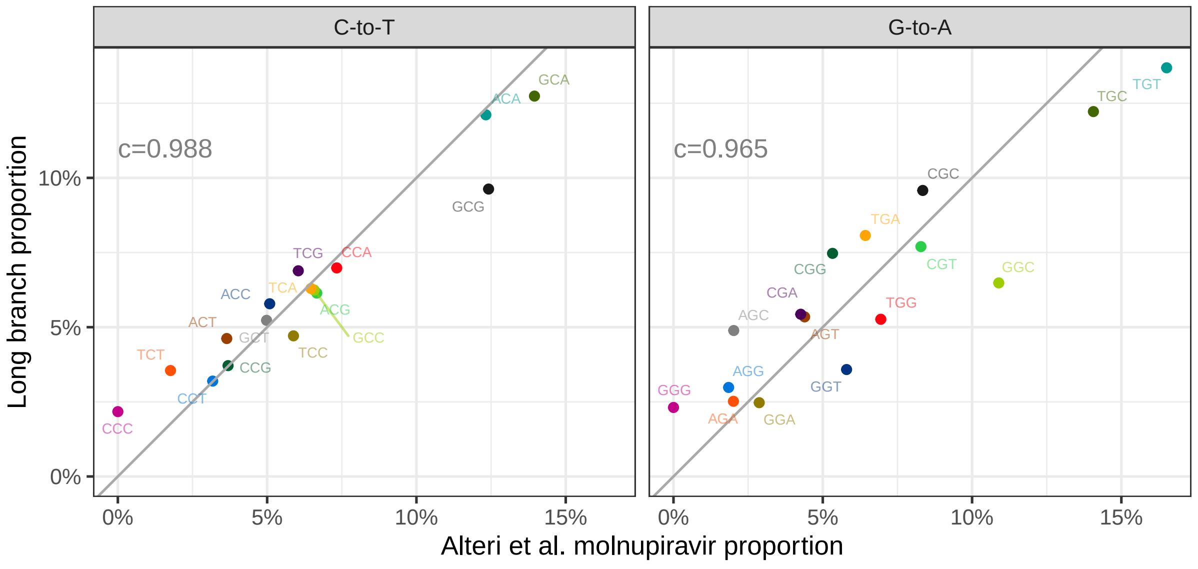


```
ggsave("p_long.pdf", width = 0.5 * 10, height = 0.5 * 4.5)

p_t2
```



```r
ggsave("p_t2.pdf", width = 0.5 * 10, height = 0.5 * 4.5)


scatters
```

```
Warning: The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```



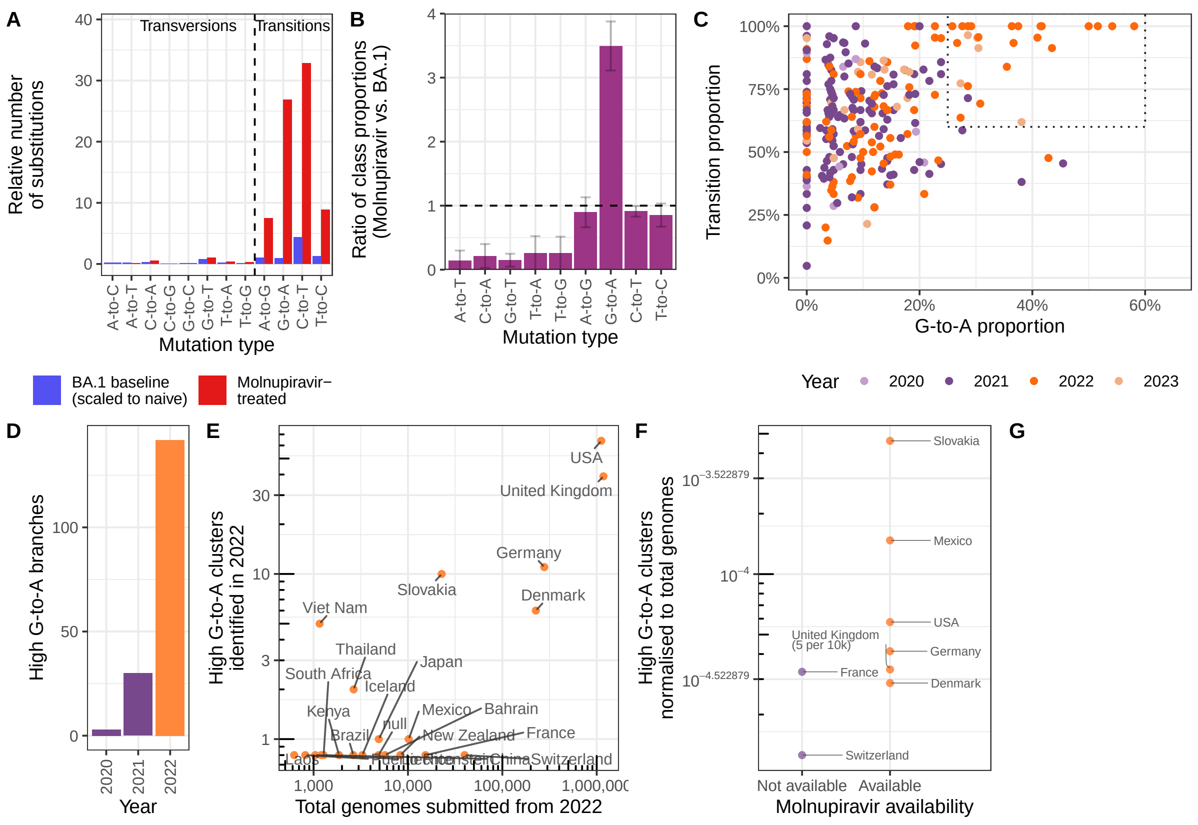
```
ggsave("scatters_small.pdf", width = 0.5 * 10, height = 0.5 * 4.5)
```

```
Warning: The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
The following aesthetics were dropped during statistical transformation: colour
i This can happen when ggplot fails to infer the correct grouping structure in
  the data.
i Did you forget to specify a `group` aesthetic or to convert a numerical
  variable into a factor?
```

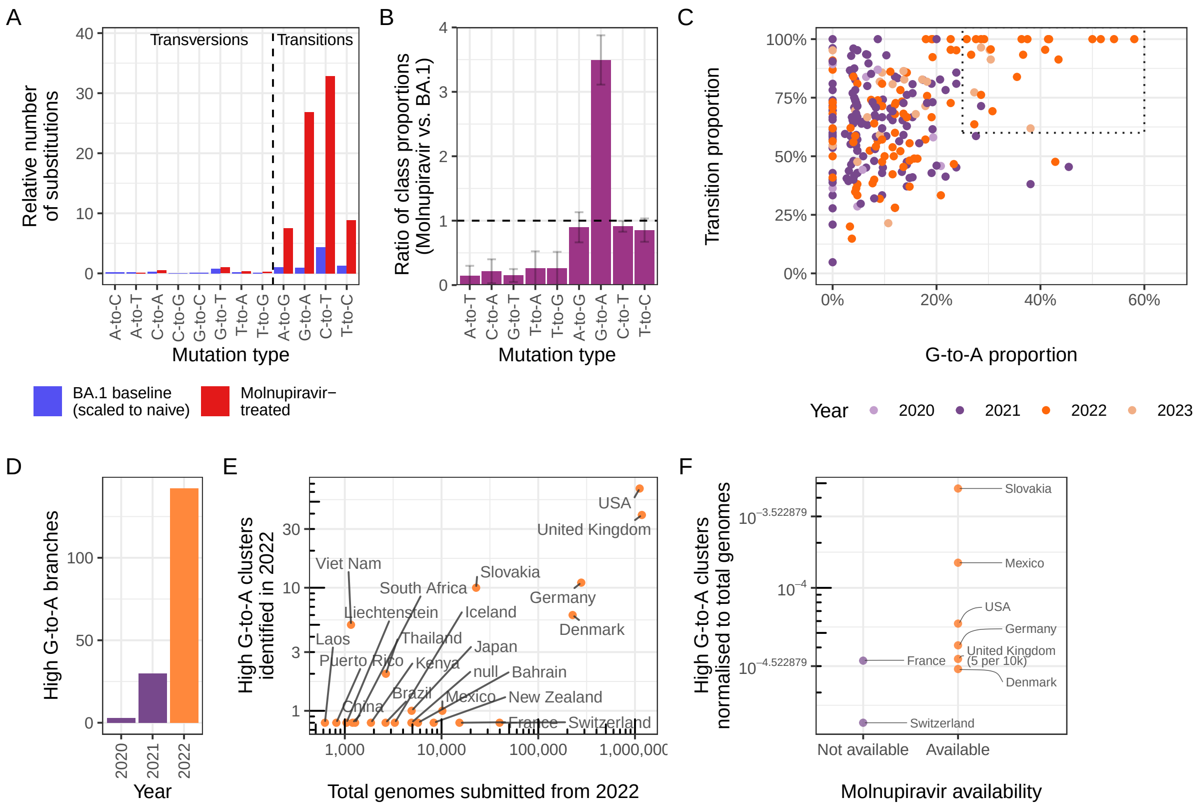
```
toprow <- plot_grid(a, b + labs(caption = "\n\n"), scatter, labels = c("A", "B", "C"), lab
bottomrow <- plot_grid(by_year_plot, country_comp, availability_plot, "", labels = c("D",
```

```
Warning in as_grob.default(plot): Cannot convert object of class character into
a grob.
```

```
plot_grid(toprow, bottomrow, ncol = 1)
```

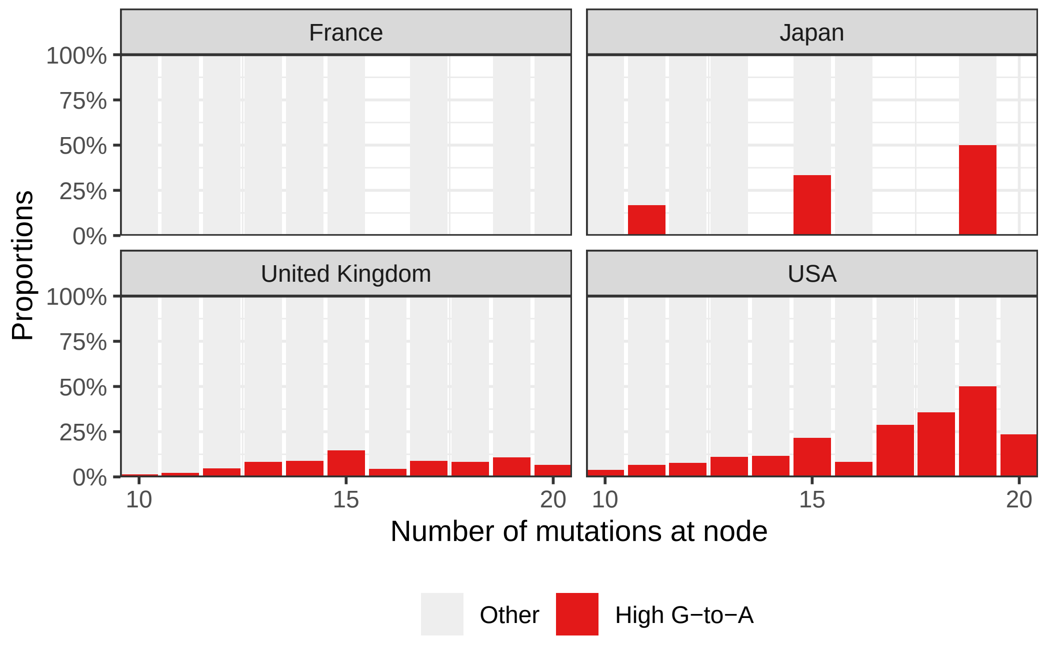


```r
ggsave("plot.pdf", width = 9.5, height = 6.5)


library(patchwork)


layout <- "
AAABBBBCCCCCCC
DDEEEEEFFFFFGG
"
a + b + scatter + by_year_plot + country_comp + availability_plot + plot_spacer()  + plot_
```



```r
ggsave("patchwork_version.pdf", width = 9.5, height = 6.5)
ggsave("patchworkcombo.pdf", width = 9.5, height = 6.5)
```

```r
countries <- c("Australia", "United Kingdom", "Japan", "France", "England", "USA")
proportions_of_long_branches <- ggplot(data_nodes %>% filter(total_muts > 9, total_muts <
  geom_bar(position = "fill") +
  facet_wrap(~consensus_country) +
  theme_bw() +
  scale_y_continuous(labels = scales::percent, expand = c(0, 0)) +
  labs(x = "Number of mutations at node", y = "Proportions") +
  scale_fill_manual(labels = c("Other", "High G-to-A"), values = c("#eeeeee", red)) +
  scale_x_continuous(expand = c(0, 0), breaks = c(10, 15, 20)) +
  labs(fill = "") +
  theme(legend.position = "bottom")
proportions_of_long_branches
```



```r
ggsave("plotter.pdf", width = 5, height = 4)
```

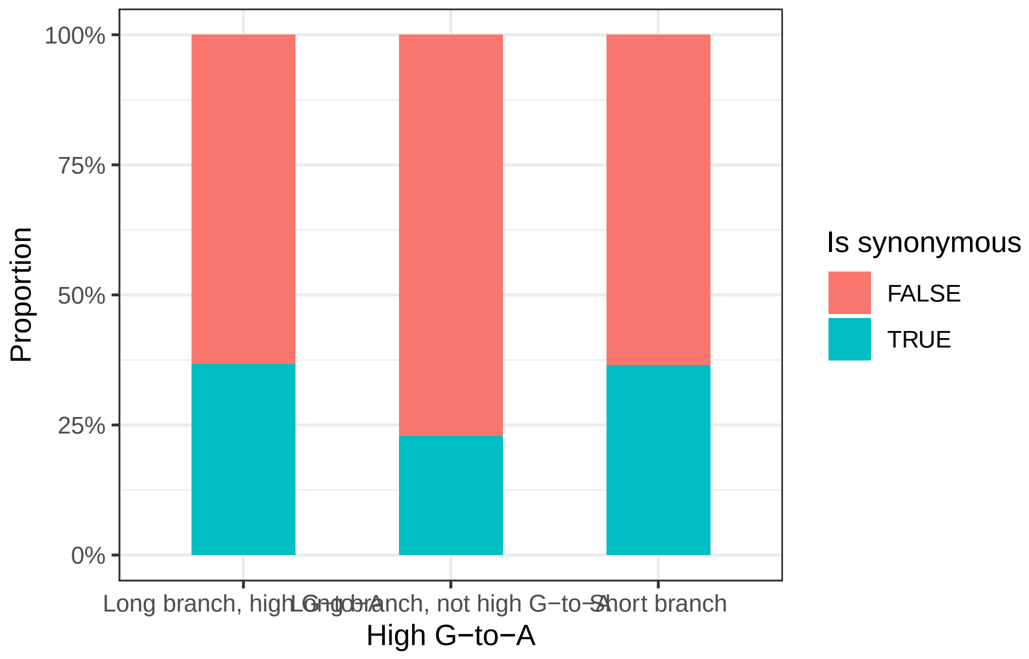
```r
library(Biostrings)
data("BLOSUM62")
bl62 <- as.data.frame(as.table(BLOSUM62))

colnames(bl62) <- c("original_aa", "alternative_aa", "bl62_score")


all <- inner_join(data_nodes, data_muts %>% right_join(bl62), by = "node_id")
```

```
Joining with `by = join_by(original_aa, alternative_aa)`
```

```r
adjustment_factor <- 3.24



dnds_stuff <- all %>% filter(gene=="S") %>%
  mutate(branch_type = case_when(
    total_muts >= threshold_branch_length & flagged ~ "Long branch, high G-to-A",
    total_muts >= threshold_branch_length & !flagged ~ "Long branch, not high G-to-A",
    TRUE ~ "Short branch"
  )) %>%
  group_by(branch_type, is_synonymous) %>%
  tally() %>%
  group_by(branch_type) %>%
  mutate(p = n / sum(n), ratio = n / (sum(n) - n), dnds = ratio / adjustment_factor) %>%
  rowwise() %>%
  mutate(confidence_interval = list(binom.test(n, n/p)$conf.int))  %>%
  mutate(
         lower = confidence_interval[1],
         upper = confidence_interval[2])

dnds_stuff
```

```
# A tibble: 6 x 9
# Rowwise:  branch_type
  branch_type  is_synonymous      n     p ratio   dnds confidence_interval lower
  <chr>        <lgl>          <int> <dbl> <dbl>  <dbl> <list>              <dbl>
1 Long branch~ FALSE            273 0.633 1.73  0.533  <dbl [2]>           0.586
2 Long branch~ TRUE             158 0.367 0.579 0.179  <dbl [2]>           0.321
3 Long branch~ FALSE          15421 0.772 3.39  1.04   <dbl [2]>           0.766
4 Long branch~ TRUE            4555 0.228 0.295 0.0912 <dbl [2]>           0.222
5 Short branch FALSE         495892 0.635 1.74  0.537  <dbl [2]>           0.634
6 Short branch TRUE          285150 0.365 0.575 0.177  <dbl [2]>           0.364
# i 1 more variable: upper <dbl>
```

```r
dnds_stuff %>% ggplot(aes(y = p, fill = is_synonymous, x = branch_type)) +
  geom_col(width = 0.5) +
  scale_y_continuous(label = scales::percent) +
  theme_bw() +
  labs(fill = "Is synonymous", x = "High G-to-A", y = "Proportion")
```



```
ggsave("plot.png", width = 4, height = 3)

library(gt)
dnds_stuff = dnds_stuff %>% dplyr::filter(!is_synonymous) %>% mutate(total_n = n/p)
# Extract the relevant data for "Long branch, high G-to-A"
long_high <- dnds_stuff %>%
  filter(branch_type == "Long branch, high G-to-A", !is_synonymous)

# Extract the relevant data for "Long branch, not high G-to-A"
long_not_high <- dnds_stuff %>%
  filter(branch_type == "Long branch, not high G-to-A", !is_synonymous)

# Extract the relevant data for "Short branch"
short_branch <- dnds_stuff %>%
  filter(branch_type == "Short branch", !is_synonymous)

# Conduct the proportion test between "Long branch, high G-to-A" and "Long branch, not hig
test1 <- prop.test(x = c(long_high$n, long_not_high$n),
                   n = c(long_high$total_n, long_not_high$total_n),
                   alternative = "two.sided")

# Conduct the proportion test between "Long branch, high G-to-A" and "Short branch"
```

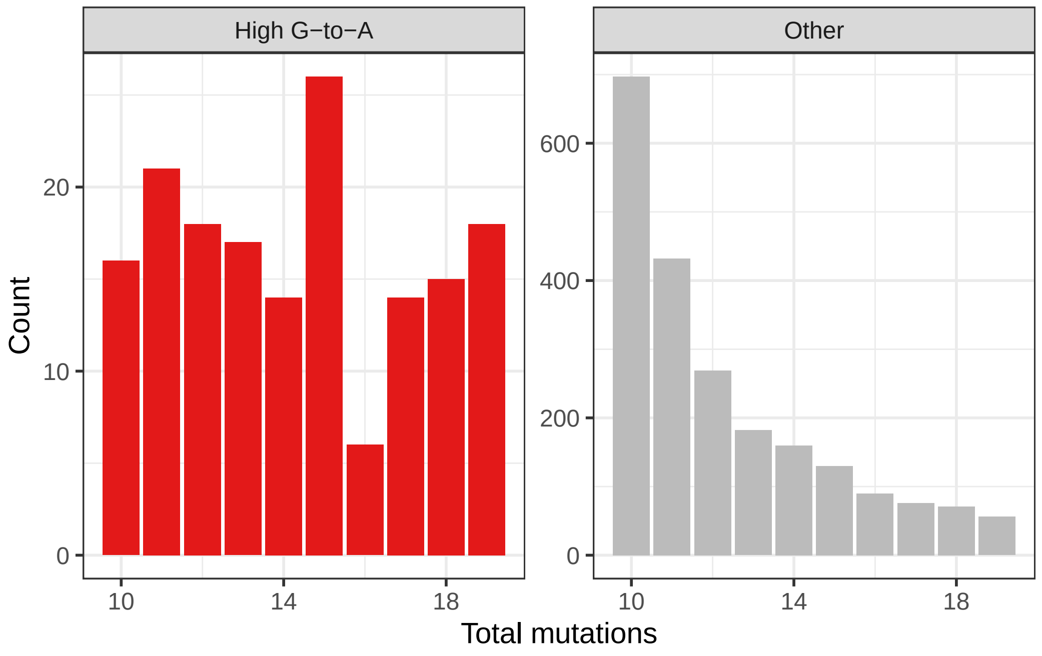
```r
test2 <- prop.test(x = c(long_high$n, short_branch$n),
                   n = c(long_high$total_n, short_branch$total_n),
                   alternative = "two.sided")

# Print the results
print(test1)
```

```

	2-sample test for equality of proportions with continuity correction

data:  c(long_high$n, long_not_high$n) out of c(long_high$total_n, long_not_high$total_n)
X-squared = 44.832, df = 1, p-value = 2.147e-11
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.1856141 -0.0915173
sample estimates:
   prop 1    prop 2 
0.6334107 0.7719764 
```

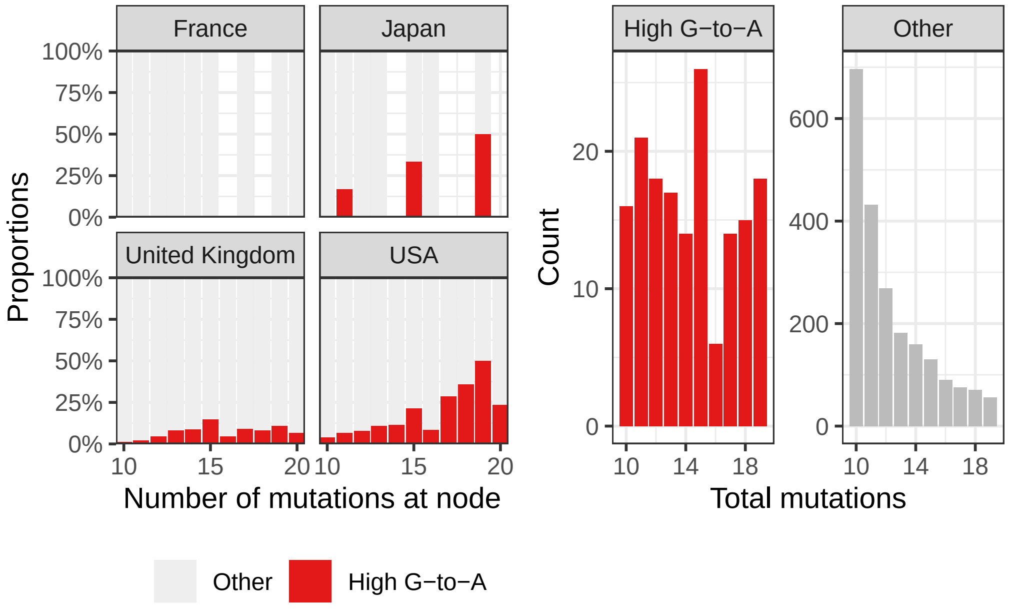
```r
print(test2)
```

```

	2-sample test for equality of proportions with continuity correction

data:  c(long_high$n, short_branch$n) out of c(long_high$total_n, short_branch$total_n)
X-squared = 0.00021406, df = 1, p-value = 0.9883
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.04816612  0.04516587
sample estimates:
   prop 1    prop 2 
0.6334107 0.6349108 
```

```r
for_plot <- data_nodes %>%
  filter(consensus_year %in% c("2022", "2023"), total_muts > 9, total_muts < 20) %>%
  mutate(new = ifelse(flagged, "High G-to-A", "Other"))

distributions <- ggplot(for_plot, aes(x = total_muts, fill = flagged)) +
  geom_bar() +
  facet_wrap(~new, scales = "free_y") +
  theme_bw() +
  scale_x_continuous(breaks = c(10, 14, 18)) +
  scale_fill_manual(values = c("#bbbbbb", red)) +
  labs(x = "Total mutations", y = "Count") +
  theme(legend.position = "none")
distributions
```



```
proportions_of_long_branches + distributions
```



```r
final_df <- tibble()

many_descendants <- data_nodes %>%
  filter(total_muts > 9, flagged, num_descendants > 1)

# Loop through every single_node in many_descendants
for (i in 1:nrow(many_descendants)) {
  single_node <- many_descendants$node_id[i]
  children <- find_children(parenthood, single_node)
  children <- children[!grepl("^node_", children)]


  # Create a temporary tibble for the current node
  cluster_df <- tibble(node_id = children, cluster = single_node)


  # bind the current tibble with the final one
  final_df <- bind_rows(final_df, cluster_df)
}

single_descendants <- data_nodes %>%
  filter(total_muts > 9, flagged, num_descendants == 1)
```

```r
single_df <- tibble(node_id = single_descendants$node_id, cluster = single_descendants$nod

final_df <- bind_rows(final_df, single_df)

final_df2 <- final_df %>%
  separate_wider_delim(node_id, delim = "|", names = c("name", "epi", "date"), cols_remove
  separate_wider_delim(name, delim = "/", names = c("country", "name2", "year"), cols_remo


final_df2 %>% filter(country == "England")
```

```
# A tibble: 73 x 8
   country name2         year  name                   epi   date  node_id cluster
   <chr>   <chr>         <chr> <chr>                  <chr> <chr> <chr>   <chr>
 1 England PHEC-X304X519 2021  England/PHEC-X304X51~ OX81~ 2021~ Englan~ node_2~
 2 England PHEC-X304X519 2021  England/PHEC-X304X51~ 2021~ <NA>  Englan~ node_2~
 3 England HSLL-1AF5265  2021  England/HSLL-1AF5265~ OU54~ 2021~ Englan~ node_5~
 4 England HSLL-1BBA08F  2021  England/HSLL-1BBA08F~ OU58~ 2021~ Englan~ node_5~
 5 England PHEC-YYFCTBO  2022  England/PHEC-YYFCTBO~ 2022~ <NA>  Englan~ node_8~
 6 England PHEC-YYFCTBO  2022  England/PHEC-YYFCTBO~ OX85~ 2022~ Englan~ node_8~
 7 England PHEC-YYRSCKX  2022  England/PHEC-YYRSCKX~ 2022~ <NA>  Englan~ node_8~
 8 England PHEC-YYRSCKX  2022  England/PHEC-YYRSCKX~ OX95~ 2022~ Englan~ node_8~
 9 England PHEC-YYEKP64  2023  England/PHEC-YYEKP64~ 2023~ <NA>  Englan~ node_9~
10 England PHEC-YYEKP64  2023  England/PHEC-YYEKP64~ OX76~ 2023~ Englan~ node_9~
# i 63 more rows
```

```r
write_csv(final_df2, "associated.csv")
```

```r
library(ggtree)

format_mutation_counts <- function(node_data) {
  # Extract mutation count columns
  mutation_cols <- c("A>C", "A>G", "A>T", "C>A", "C>G", "C>T", "G>A", "G>C", "G>T", "T>A",

  # Create a named vector of mutation counts
  mutation_counts <- sapply(mutation_cols, function(x) node_data[[x]])
  names(mutation_counts) <- mutation_cols

  # Remove zeros
  mutation_counts <- mutation_counts[mutation_counts > 0]
  # Sort in descending order
  mutation_counts <- sort(mutation_counts, decreasing = TRUE)

  # Format as a string
  mutation_str <- paste(names(mutation_counts), mutation_counts, sep = ": ", collapse = ",
  mutation_str <- gsub(">", "\u00adto\u00ad", mutation_str)

  return(mutation_str)
}

prune_and_plot <- function(node_id, parent, node_data) {
  mutation_title <- format_mutation_counts(node_data)
  print(node_id)

  # Create directories if they do not exist
  if (!dir.exists("data")) {
    dir.create("data")
  }

  if (!dir.exists("trees")) {
    dir.create("trees")
  }


  gotree_command <- paste0("~/Dropbox/new_mov_data/gotree_arm64_darwin subtree -i ~/Dropbo

  print(gotree_command)

  # Execute the system call
  system(gotree_command)

  # Read the newick file
  tree <- read.tree(paste0("data/pruned_", node_id, ".nwk"))


  get_node_index <- function(tree, node_name) {
    for (i in 1:length(tree$node.label)) {
      if (tree$node.label[i] == node_name) {
        return(i + ape::Ntip(tree)) # Return the index of the node
      }
    }
    return(NULL) # Return NULL if no node with that name was found
  }

  node_index <- get_node_index(tree, node_id)


  # Plot the tree using ggtree
  ggtree_plot <- ggtree(tree, aes( # color=node==node_index
  )) +
    geom_tiplab(size = 3, aes(label = label)) + # Add tip labels
    geom_point2(aes(subset = !is.na(num_tips)), color = "#4561de") + # Add points to visua
    coord_cartesian(clip = "off") +
    theme_tree2(plot.margin = margin(6, 290, 6, 6)) +
    theme(legend.position = "none") + #+scale_color_manual(values = c("TRUE" = "darkblue",
    geom_text(aes(x = branch, label = ifelse(node == node_index, mutation_title, "")),
      size = 3,
      vjust = -.4, color = "firebrick"
    ) #+ggtitle(node_id)

  # Calculate the number of tips
  num_tips <- ape::Ntip(tree)

  # Calculate a reasonable height for the plot
  # Adjust this calculation as needed
  plot_height <- max(1.5, num_tips / 5)

  # Save the plot to a pdf
  # ggsave(filename = paste0("trees/node_", node_id, ".pdf"), plot = ggtree_plot, height =

  return(list(ggtree_plot, plot_height))
}

filtered_nodes <- data_nodes %>%
  filter(total_muts > 9, flagged, num_descendants > 2) %>%
  arrange(desc(num_descendants))

filtered_nodes

library(patchwork)

plots_list <- list()
```

```r
heights_list <- c()
total_height <- 0
plot_number <- 1

pdf("trees/combined_plots.pdf", height = 11.7, width = 8.3) # Create a PDF file

for (i in 1:nrow(filtered_nodes)) {
  print(i)

  listed <- prune_and_plot(filtered_nodes$node_id[i], get_parent(parenthood, filtered_node

  ggtree_plot <- listed[[1]]
  plot_height <- listed[[2]]

  # Check if adding the new plot will exceed the page size
  if ((total_height + plot_height) >= 16) { # A4 height in inches
    # Save the existing plots
    if (length(plots_list) > 0) {
      combined_plot <- wrap_plots(plots_list) +
        plot_layout(heights = heights_list / total_height) # Normalize to make it relative
      print(combined_plot)
      ggsave(filename = paste0("trees/combined_", plot_number, ".pdf"), plot = combined_pl

      plot_number <- plot_number + 1 # Increment plot_number
    }
    # Reset the list and total height
    plots_list <- list()
    heights_list <- c()
    total_height <- 0
  }


  if (plot_height < 15 * 5) {
    # Add the new plot
    plots_list[[length(plots_list) + 1]] <- ggtree_plot
    heights_list <- c(heights_list, plot_height)
    total_height <- total_height + plot_height
  }
}

# After the loop, save any remaining plots
```

```r
if (length(plots_list) > 0) {
  combined_plot <- wrap_plots(plots_list) +
    plot_layout(heights = heights_list / total_height) # Normalize to make it relative
  ggsave(filename = paste0("trees/combined_", plot_number, ".pdf"), plot = combined_plot,
}

dev.off()
```

```r
# Function to read FASTA file and convert to a tibble
read_fasta_to_tibble <- function(file_path) {
  # Load the fasta file
  fasta_data <- readDNAStringSet(file_path)

  # Get sequence from the first (and possibly only) sequence in the fasta file
  sequence <- as.character(fasta_data[[1]])
  residues <- strsplit(sequence, "")[[1]]
  # Create a tibble with residue and index
  tibble(
    index = seq_along(residues),
    residue = residues
  )
}

ref_tib <- read_fasta_to_tibble("ref.fa.fasta") %>% mutate(context_before = lag(residue),
ref_tib
```

```
# A tibble: 29,903 x 4
   index residue context_before context_after
   <int> <chr>   <chr>          <chr>
 1     1 A       <NA>           T
 2     2 T       A              T
 3     3 T       T              A
 4     4 A       T              A
 5     5 A       A              A
 6     6 A       A              G
 7     7 G       A              G
 8     8 G       G              T
 9     9 T       G              T
10    10 T       T              T
# i 29,893 more rows
```

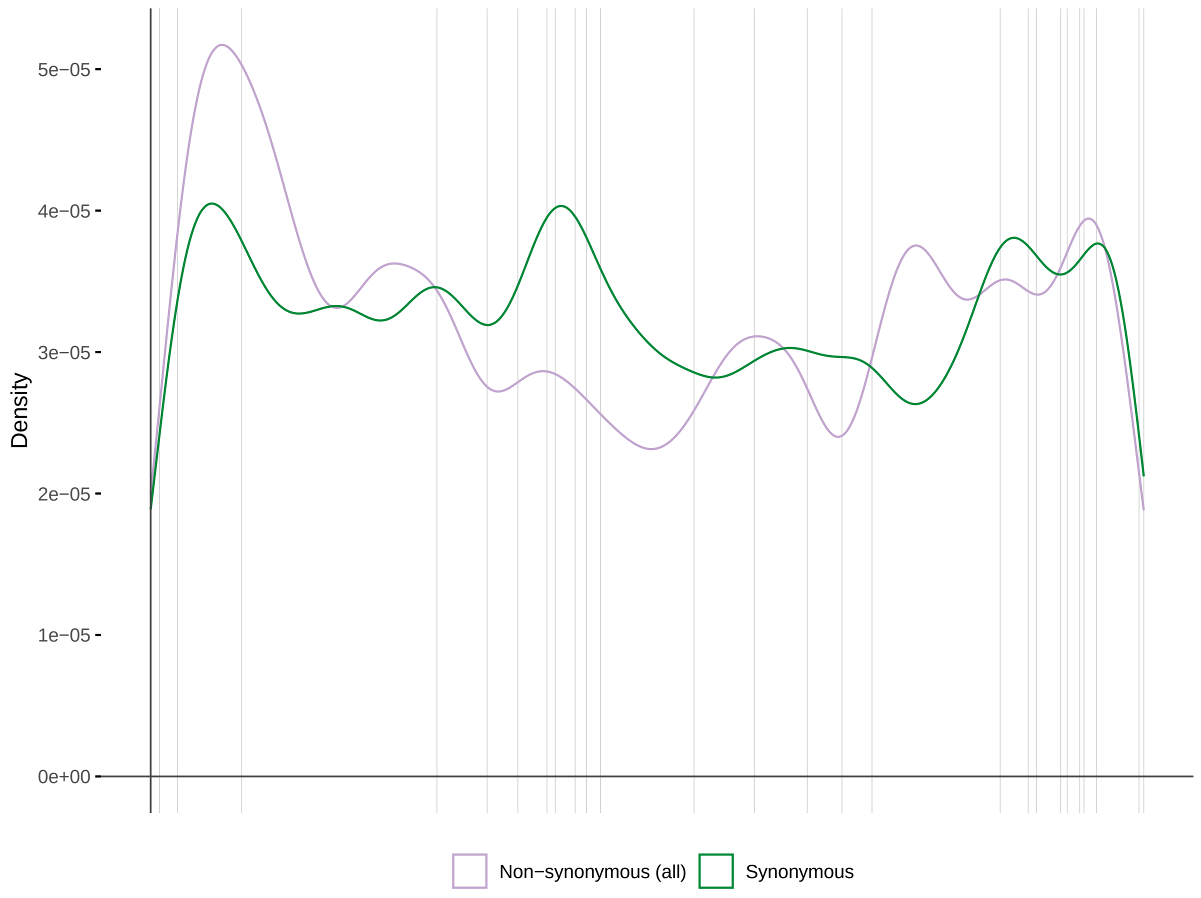
```r
library(gggenes)
library(tidyverse)

# Read data
hu1 <- read_tsv("./hu1.tsv")
```

```
Rows: 38 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: "\t"
chr (2): feature_name, feature_type
dbl (2): start, end

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

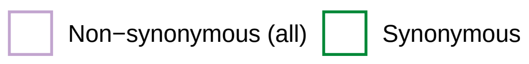
```r
# Define unique end_points
end_points <- unique(hu1$end)

# Define a function to generate the vertical line
generate_vline <- function(end_points) {
  geom_vline(
    xintercept = end_points # , linetype = "dashed"
    , color = "lightgray", size = .2
  )
}

# Define common theme
common_theme <- theme(
  axis.ticks = element_line(color = "black"),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()
)

# Define filtered hu1
filtered_hu1 <- hu1 %>% filter(feature_type %in% c("CDS", "mat_peptide"))

# hu1_plot
hu1_plot <- ggplot(filtered_hu1, aes(xmin = start, xmax = end, y = feature_type, fill = fe
  generate_vline(end_points) +
  scale_fill_manual(values = c("#fbe4bc", "#dff3f8")) +
  labs(x = "Nucleotide position", y = "Feature", fill = "Type") +
  theme_minimal() +
  geom_gene_arrow() +
  geom_gene_label() +
  common_theme +
  labs(y = "") +
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank()) +
  theme(plot.margin = margin(t = 0, r = 5, l = 5, b = 0)) +
  xlim(c(0, NA))


# Define myset
myset <- all %>%
  mutate(blcut = cut(bl62_score, 3)) %>%
  filter(total_muts > 10, flagged) %>%
  mutate(mut_type = case_when(
    (alternative_aa == "*") & (original_aa != "*") ~ "STOP",
    # bl62_score < -0 ~ "Negative BLOSUM",
    is_synonymous ~ "Synonymous",
    TRUE ~ "Non-synonymous (all)"
  )) %>%
  filter(mut_type != "STOP")

start_nsp14_codon <- 5926
end_nsp14_codon <- 6452


myset <- myset %>%
  mutate(
    is_nsp14 = ifelse(gene == "ORF1ab" & aa_index >= start_nsp14_codon & aa_index <= end_n
    nsp14_index = ifelse(is_nsp14, aa_index - start_nsp14_codon + 1, NA)
  )

nsp14_muts <- myset %>%
  filter(is_nsp14, !is_synonymous) %>%
  group_by(aa_string, nsp14_index) %>%
  tally() %>%
  arrange(-n)

ggplot(nsp14_muts %>% filter(n > 4), aes(x = nsp14_index)) +
  geom_density(bw = 50)
```

```
fullmyset <- bind_rows(myset)

my_colors <- c(
  "STOP" = "#D55E00",
  "Synonymous" = "#008837",
  "Non-synonymous (all)" = "#c2a5cf",
  "Non-synonymous (site recurrent 4+ times)" = "#7b3294"
)

density_plot <- ggplot(fullmyset, aes(x = nt_index, color = mut_type, group = mut_type)) +
  generate_vline(end_points) +
  geom_density(bw = 900) +
  theme_minimal() +
  common_theme +
  theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    legend.position = "bottom", # change position to top, bottom, left, right or c(x, y) f
    legend.direction = "horizontal"
  ) +
  geom_hline(yintercept = 0, color = "#444444", size = 0.4) +
  geom_vline(xintercept = 0, color = "#444444", size = 0.4) +
  scale_color_manual(values = my_colors) +
  labs(y = "Density", color = "") +
  theme(plot.margin = margin(t = 5, r = 5, l = 5, b = 0))
```

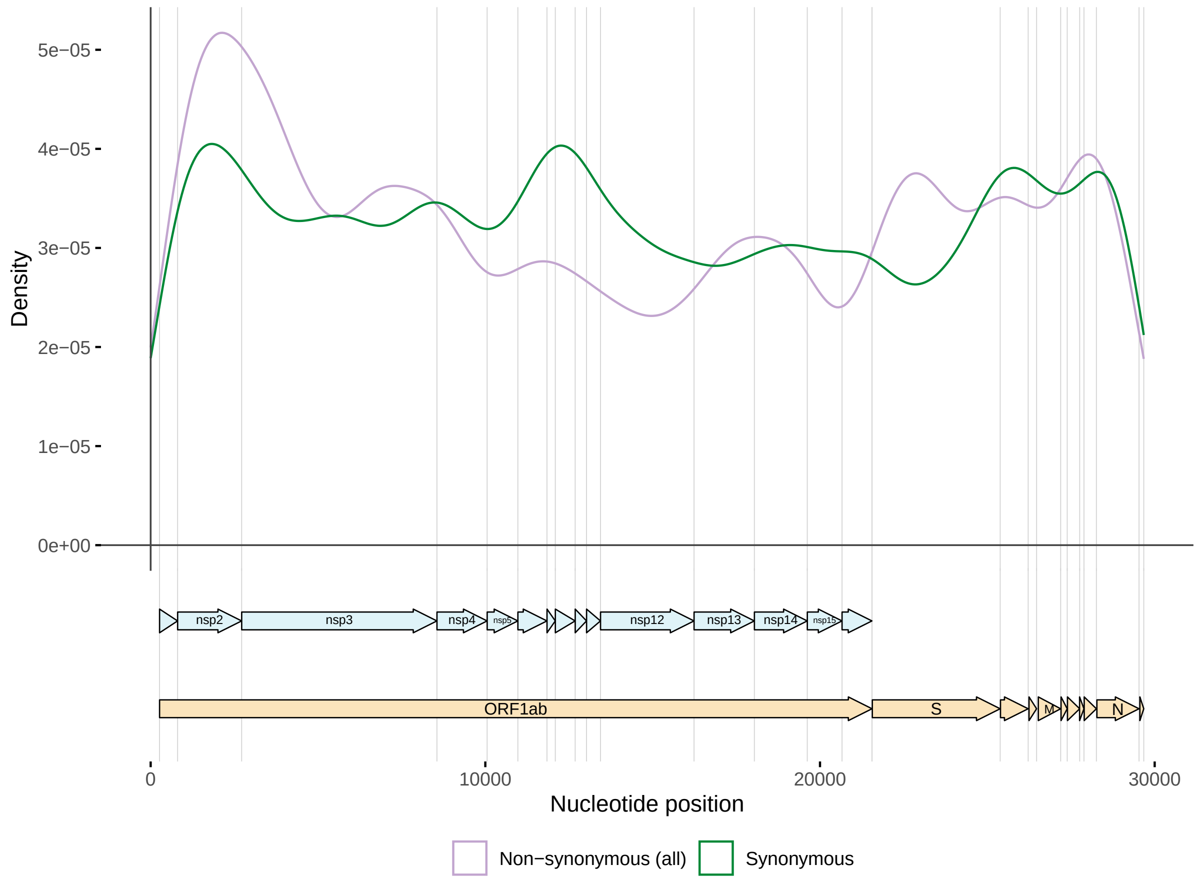
```
density_plot
```



```
# Legends
legends <- plot_grid(get_legend(density_plot))
legends
```



```
# Final plot
final_plot <- plot_grid(density_plot + theme(legend.position = "none"), NULL, hu1_plot + t

final_plot
```



```
ggsave("genome.pdf", width = 10, height = 4)
```

```r
myset$nt_mut = paste0(myset$original_nt,myset$nt_index, myset$alternative_nt)

fortable <- myset %>%
  filter(!is_synonymous) %>%
  group_by(original_aa, alternative_aa, gene, aa_index, mutation_type,nt_mut) %>%
  tally() %>%
  mutate(mut_types = paste0(mutation_type, ":", n)) %>%
  mutate(nt_muts=paste0(nt_mut, ":", n)) %>%
  group_by(original_aa, alternative_aa, gene, aa_index) %>%
  summarise(n = sum(n), types = paste(mut_types, collapse = ", "),nt_muts =  paste(nt_muts
  arrange(-n) %>%
  filter(gene == "S") %>%
  mutate(mut_format = paste0("S:", original_aa, aa_index, alternative_aa)) %>%
  mutate(type = substr(types, 1, 3)) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'original_aa', 'alternative_aa', 'gene'.
You can override using the `.groups` argument.
```

```r
fortable <- fortable %>%
  mutate(index = as.numeric(str_extract(nt_muts, "\\d+"))) %>% inner_join(ref_tib)
```

```
Joining with `by = join_by(index)`
```

```r
fortable <- fortable %>%
  mutate(context = paste0(context_before, substr(nt_muts, 1, 1), context_after))


library(gridExtra)
```

```

Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
The following object is masked from 'package:BiocGenerics':

    combine
```

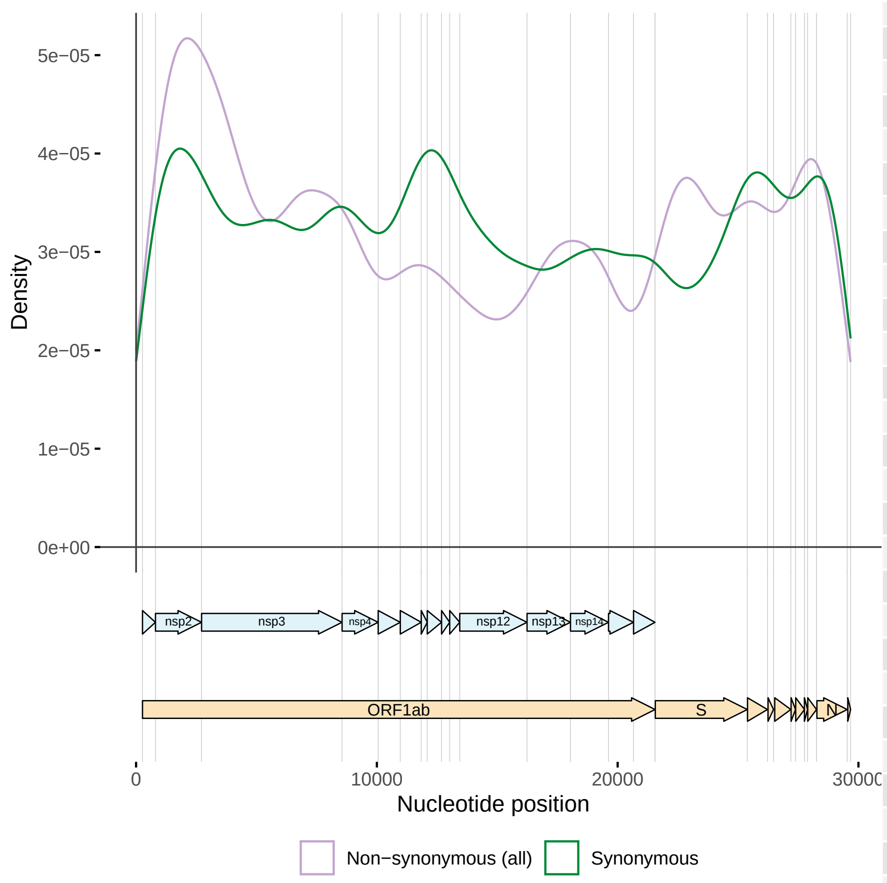
```
table_theme <- ttheme_default(
  core = list(fg_params = list(cex = 0.6)), # font size for table body
  colhead = list(fg_params = list(cex = 0.6)), # font size for column headers
  rowhead = list(fg_params = list(cex = 0.6)) # font size for row headers
)


fortable
```

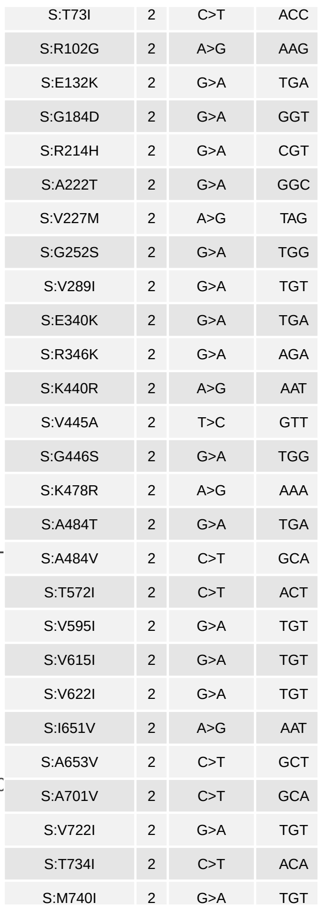
```
# A tibble: 210 x 14
   original_aa alternative_aa gene  aa_index     n types nt_muts   mut_format
   <chr>       <chr>          <chr>    <dbl> <int> <chr> <chr>     <chr>
 1 A           T              S         1070     3 G>A:3 G24770A:3 S:A1070T
 2 D           N              S          574     3 G>A:3 G23282A:3 S:D574N
 3 P           L              S            9     3 C>T:3 C21588T:3 S:P9L
 4 P           L              S         1162     3 C>T:3 C25047T:3 S:P1162L
 5 A           T              S          222     2 G>A:2 G22226A:2 S:A222T
 6 A           T              S          484     2 G>A:2 G23012A:2 S:A484T
 7 A           V              S          484     2 C>T:2 C23013T:2 S:A484V
 8 A           V              S          653     2 C>T:2 C23520T:2 S:A653V
 9 A           V              S          701     2 C>T:2 C23664T:2 S:A701V
10 E           K              S          132     2 G>A:2 G21956A:2 S:E132K
# i 200 more rows
# i 6 more variables: type <chr>, index <dbl>, residue <chr>,
#   context_before <chr>, context_after <chr>, context <chr>
```

```
# Convert the fortable data frame to a table grob
table_grob <- tableGrob(fortable %>% filter(n > 1) %>% arrange(-n, aa_index) %>% select(mu

fortable %>% filter(n > 1)  %>% group_by(context) %>% tally() %>% arrange(-n)
```

```
# A tibble: 20 x 2
   context     n
   <chr>   <int>
 1 TGT         9
 2 TGA         4
 3 AAT         2
 4 ACA         2
 5 CCA         2
 6 GCA         2
 7 TGG         2
 8 AAA         1
 9 AAG         1
10 ACC         1
11 ACT         1
12 AGA         1
13 CGT         1
14 GCT         1
15 GGC         1
16 GGT         1
17 GTT         1
18 TAG         1
19 TGC         1
20 TTT         1
```

```r
grid.arrange(final_plot, table_grob, ncol = 2, widths = c(3, 1))
```

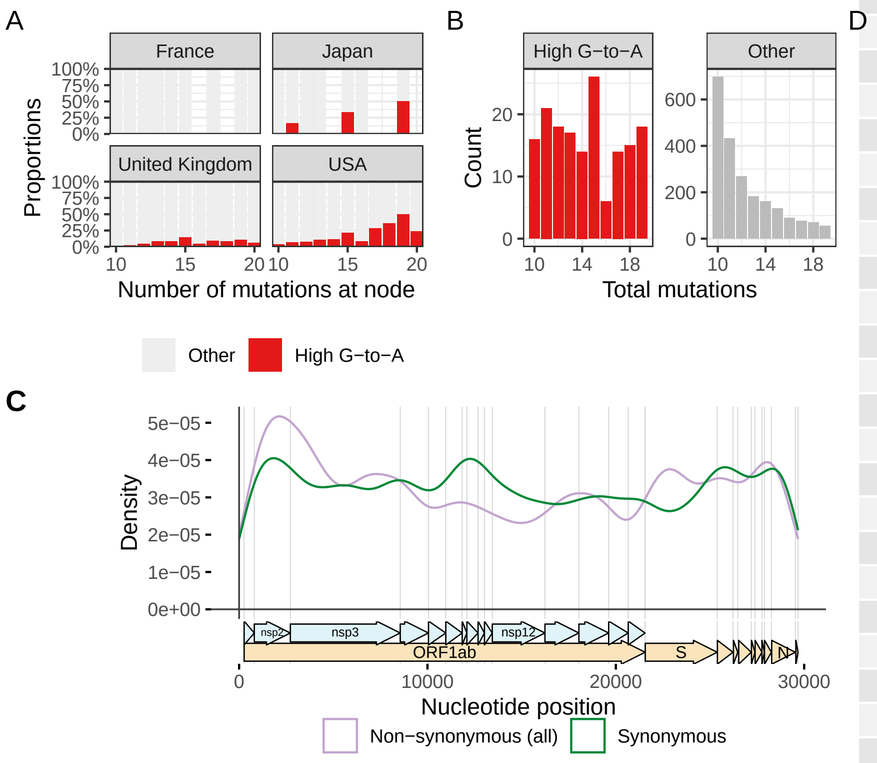




S:T73I	2	C>T	ACC
S:R102G	2	A>G	AAG
S:E132K	2	G>A	TGA
S:G184D	2	G>A	GGT
S:R214H	2	G>A	CGT
S:A222T	2	G>A	GGC
S:V227M	2	A>G	TAG
S:G252S	2	G>A	TGG
S:V289I	2	G>A	TGT
S:E340K	2	G>A	TGA
S:R346K	2	G>A	AGA
S:K440R	2	A>G	AAT
S:V445A	2	T>C	GTT
S:G446S	2	G>A	TGG
S:K478R	2	A>G	AAA
S:A484T	2	G>A	TGA
S:A484V	2	C>T	GCA
S:T572I	2	C>T	ACT
S:V595I	2	G>A	TGT
S:V615I	2	G>A	TGT
S:V622I	2	G>A	TGT
S:I651V	2	A>G	AAT
S:A653V	2	C>T	GCT
S:A701V	2	C>T	GCA
S:V722I	2	G>A	TGT
S:T734I	2	C>T	ACA
S:M740I	2	G>A	TGT

```r
library(ggplotify)
table_plot <- as.ggplot(table_grob)

# Arrange the plot and table using patchwork
final_figure <-
  proportions_of_long_branches + distributions +
  final_plot + table_plot +
  plot_layout(ncol = 2, widths = c(3, 1))
```

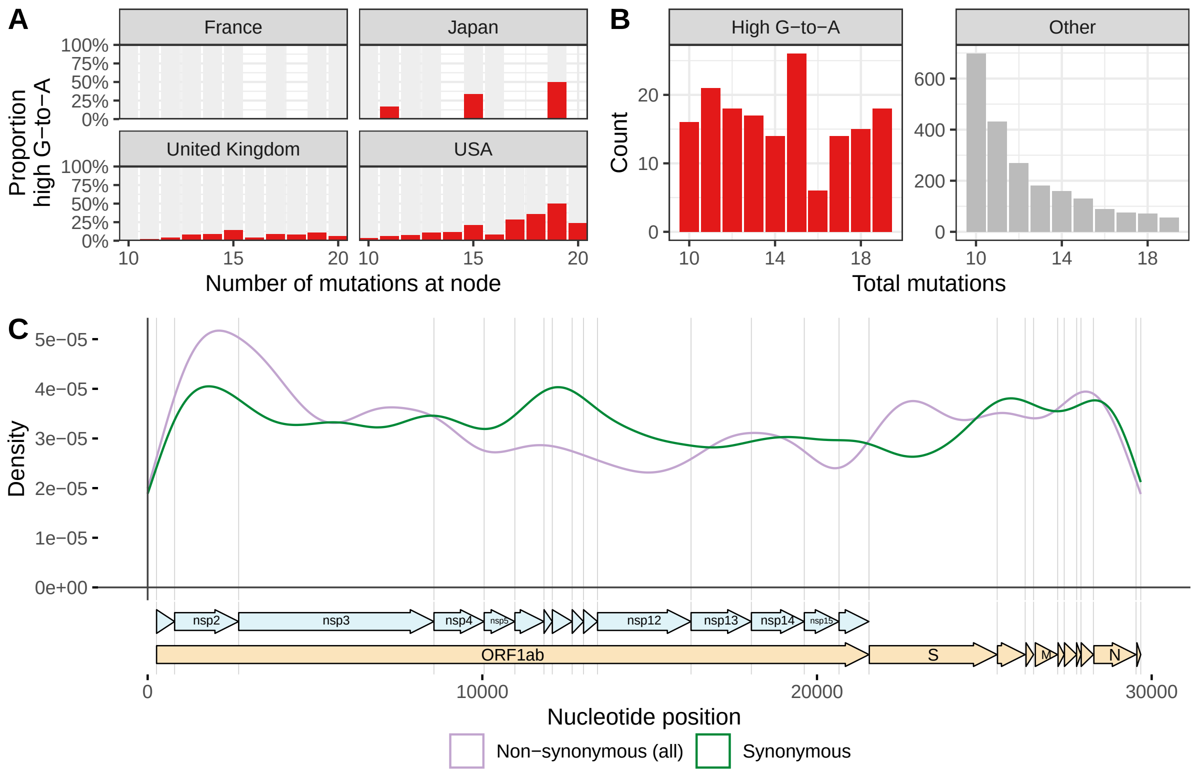
```r
layout <- "
AABBDD
CCCCDD
CCCCDD
"
proportions_of_long_branches + distributions +
  final_plot + table_plot +
  plot_layout(design = layout) + plot_annotation(tag_levels = "A")
```



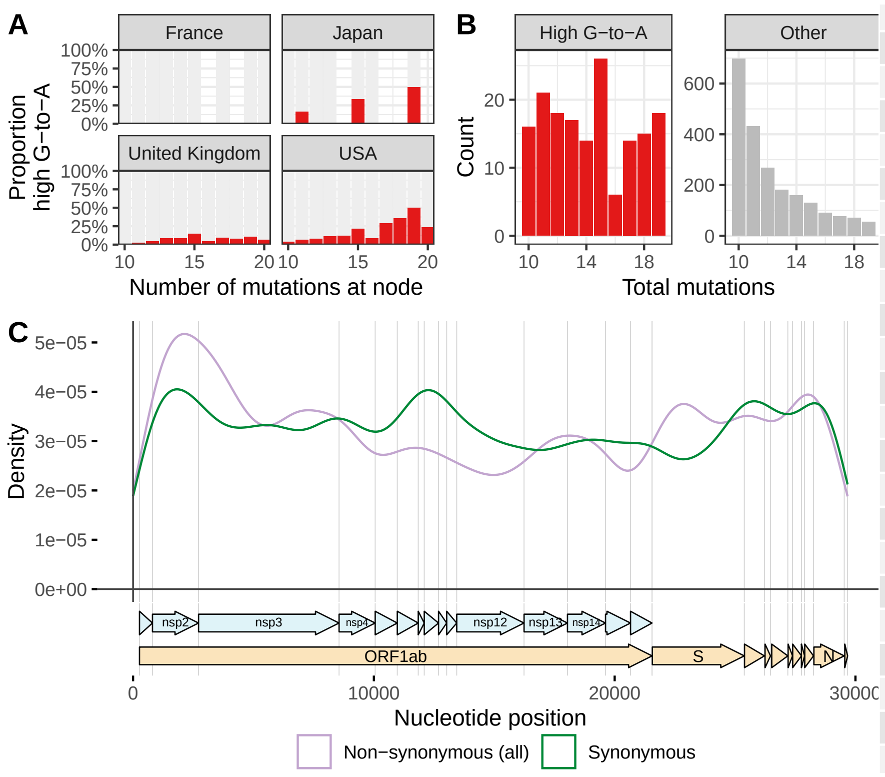
D

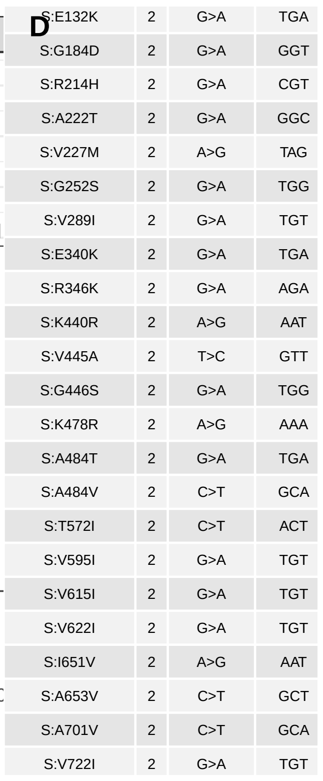
S:R102G	2	A>G	AAG
S:E132K	2	G>A	TGA
S:G184D	2	G>A	GGT
S:R214H	2	G>A	CGT
S:A222T	2	G>A	GGC
S:V227M	2	A>G	TAG
S:G252S	2	G>A	TGG
S:V289I	2	G>A	TGT
S:E340K	2	G>A	TGA
S:R346K	2	G>A	AGA
S:K440R	2	A>G	AAT
S:V445A	2	T>C	GTT
S:G446S	2	G>A	TGG
S:K478R	2	A>G	AAA
S:A484T	2	G>A	TGA
S:A484V	2	C>T	GCA
S:T572I	2	C>T	ACT
S:V595I	2	G>A	TGT
S:V615I	2	G>A	TGT
S:V622I	2	G>A	TGT
S:I651V	2	A>G	AAT
S:A653V	2	C>T	GCT
S:A701V	2	C>T	GCA

```r
a <- plot_grid(proportions_of_long_branches + theme(legend.position = "none") + labs(y = "
b <- plot_grid(a, final_plot, ncol = 1, labels = c("", "C"), rel_heights = c(0.4, 0.6))
b
```



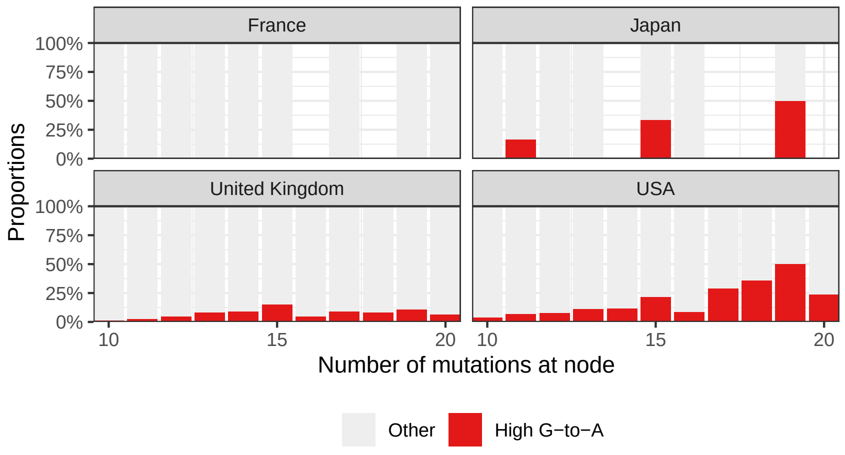
```r
c <- plot_grid(b, table_plot, labels = c("", "D"), rel_widths = c(0.75, 0.25))

c
```

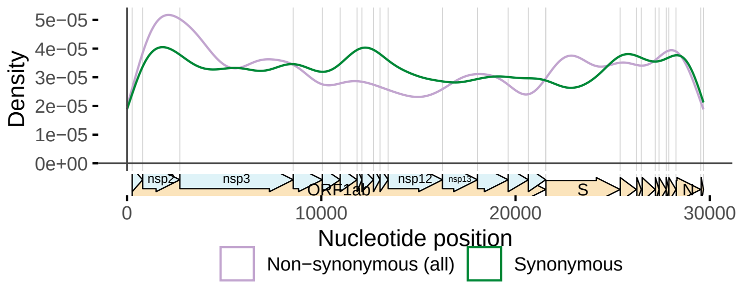


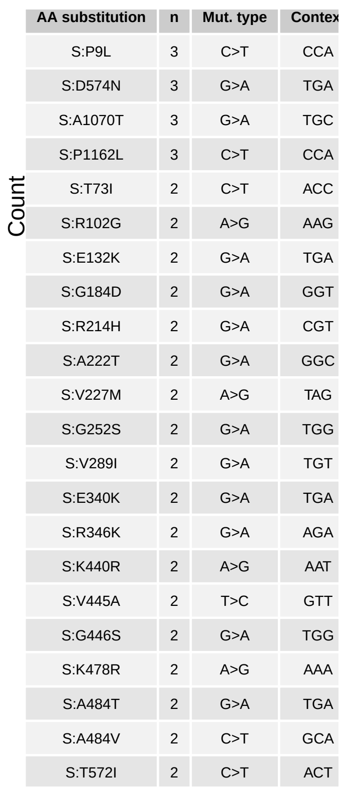


S:E132K	2	G>A	TGA
S:G184D	2	G>A	GGT
S:R214H	2	G>A	CGT
S:A222T	2	G>A	GGC
S:V227M	2	A>G	TAG
S:G252S	2	G>A	TGG
S:V289I	2	G>A	TGT
S:E340K	2	G>A	TGA
S:R346K	2	G>A	AGA
S:K440R	2	A>G	AAT
S:V445A	2	T>C	GTT
S:G446S	2	G>A	TGG
S:K478R	2	A>G	AAA
S:A484T	2	G>A	TGA
S:A484V	2	C>T	GCA
S:T572I	2	C>T	ACT
S:V595I	2	G>A	TGT
S:V615I	2	G>A	TGT
S:V622I	2	G>A	TGT
S:I651V	2	A>G	AAT
S:A653V	2	C>T	GCT
S:A701V	2	C>T	GCA
S:V722I	2	G>A	TGT

```r
ggsave("figtt.pdf", width = 9, height = 5.15)

# Print the final figure
print(final_figure)
```







AA substitution	n	Mut. type	Contex
S:P9L	3	C>T	CCA
S:D574N	3	G>A	TGA
S:A1070T	3	G>A	TGC
S:P1162L	3	C>T	CCA
S:T73I	2	C>T	ACC
S:R102G	2	A>G	AAG
S:E132K	2	G>A	TGA
S:G184D	2	G>A	GGT
S:R214H	2	G>A	CGT
S:A222T	2	G>A	GGC
S:V227M	2	A>G	TAG
S:G252S	2	G>A	TGG
S:V289I	2	G>A	TGT
S:E340K	2	G>A	TGA
S:R346K	2	G>A	AGA
S:K440R	2	A>G	AAT
S:V445A	2	T>C	GTT
S:G446S	2	G>A	TGG
S:K478R	2	A>G	AAA
S:A484T	2	G>A	TGA
S:A484V	2	C>T	GCA
S:T572I	2	C>T	ACT

```
nsp14_muts
```

```
# A tibble: 73 x 3
# Groups:   aa_string [73]
   aa_string      nsp14_index     n
   <chr>                <dbl> <int>
 1 ORF1ab:V6362I          437     4
 2 ORF1ab:S6428L          503     3
 3 ORF1ab:T6449I          524     3
 4 ORF1ab:V6026I          101     3
 5 ORF1ab:A6044V          119     2
 6 ORF1ab:A6319T          394     2
 7 ORF1ab:A6396T          471     2
 8 ORF1ab:D6357N          432     2
 9 ORF1ab:M6240I          315     2
10 ORF1ab:P6354L          429     2
# i 63 more rows
```

```r
nsp14_muts$nsp14_index[1:30]
```

```
 [1] 437 503 524 101 119 394 471 432 315 429  31   4  85 119 138 281 307 353  48
[20] 291 415 449  36 453  26 228 373 427 455  55
```

```r
data_nodes %>%
  filter(flagged) %>%
  filter(total_muts >= 10) %>%
  arrange(-num_descendants)
```

```
# A tibble: 224 x 23
   node_id      num_descendants consensus_country consensus_year date
   <chr>                  <dbl> <chr>             <chr>          <date>
 1 node_1548417               6 United Kingdom    2022           2022-02-20
 2 node_836114                4 Slovakia          2021           2021-05-29
 3 node_1524319               4 United Kingdom    2022           2022-02-20
 4 node_605176                3 Estonia           2021           2021-02-27
 5 node_654605                3 USA               2021           2021-05-18
 6 node_882486                3 United Kingdom    2022           2022-04-09
 7 node_902283                3 Germany           2022           2022-04-26
 8 node_1138270               3 United Kingdom    2022           2022-12-15
 9 node_1199137               3 United Kingdom    ?              2022-10-31
10 node_1287135               3 Denmark           2022           2022-05-07
# i 214 more rows
# i 18 more variables: date_length <dbl>, age <chr>, `A>C` <dbl>, `A>G` <dbl>,
#   `A>T` <dbl>, `C>A` <dbl>, `C>G` <dbl>, `C>T` <dbl>, `G>A` <dbl>,
#   `G>C` <dbl>, `G>T` <dbl>, `T>A` <dbl>, `T>C` <dbl>, `T>G` <dbl>,
#   total_muts <dbl>, transitions <dbl>, transversions <dbl>, flagged <lgl>
```

```r
mutations_in_highly_mutated_seq = "A543G, G1068A, G1186A, G1264A, T1370C, G1743A, A2497G,


mutations_in_highly_mutated_seq = str_replace_all(mutations_in_highly_mutated_seq, "nt:",
# Split the string by commas, and then extract the initial nucleotide, index, and final nu
mutations_tibble <- str_split(mutations_in_highly_mutated_seq, ",\\s*") %>%
  unlist() %>%
  tibble(mutation = .)  %>%
  mutate(
    par = str_extract(mutation, "^[A-Z]"),
    index = str_extract(mutation, "[0-9]+"),
    mut = str_extract(mutation, "[A-Z]$")
  ) %>%
  select(-mutation) %>% mutate(index=as.numeric(index)) %>% inner_join(ref_tib)
```

```
Joining with `by = join_by(index)`
```

```
of_interest = mutations_tibble %>% group_by(par,context_before,context_after,mut) %>% tall


unnormalise <- function(df){


inner_join(df,nuc_genome_counts) %>% mutate(spectrum_value = spectrum_value * genome_count
}

model1 = long %>% mutate(type=paste0(par,mut)) %>% rename(spectrum_value = Number_of_mutat
```

```
Joining with `by = join_by(context_before, par, context_after)`
```

```
model2 = ba1 %>% mutate(type=paste0(par,mut)) %>% rename(spectrum_value = Number_of_mutati
```

```
Joining with `by = join_by(context_before, par, context_after)`
```

```
types_of_interest = c("GA","CT","AG","TC")

library(nnet)
library(BayesFactor)
```

```
Loading required package: coda
Loading required package: Matrix

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack

The following object is masked from 'package:S4Vectors':

    expand

************
Welcome to BayesFactor 0.9.12-4.4. If you have questions, please contact Richard Morey (richa

Type BFManual() to open the manual.
************
```

```
join_everything = full_join(model1,model2,by=c("par","mut","type","context_before","contex



bfs = c()


for (mytype in types_of_interest) {

  filtered = join_everything %>% filter(type==mytype)
  prob1 = dmultinom(filtered$n, size = sum(filtered$n), prob = filtered$spectrum_value_1,
  prob2 = dmultinom(filtered$n, size = sum(filtered$n), prob = filtered$spectrum_value_2,
  bf = prob1/prob2
  bfs[mytype] = bf

}

bfs
```

```
          GA           CT           AG           TC
35017.651240  9636.017471    52.565716     1.227803
```

```
prod(bfs)
```

```
[1] 21777892240
```