BrepPhylo: Phylogenetic analysis

Joseph Ng

2021-06-21

1 Background: Reconstructing B cell lineages

Immunoglobulins produced by B cells are constantly modified in order to optimise their function. One aspect of their function is to recognise antigens the immune system encounters during an infection; this is achieved with the '*variable region*' of immunoglobulins, where mutations can be introduced to enhance the affinity and specificity towards the given target, during a process called *affinity maturation*. Immunoglobulin sequences can be clustered by observing similarities of the variable region to obtain *clonotypes*[1]: typically, all sequences within a clonotype have similar Complementarity-determining region 3 (CDR3) and therefore should all belong to one 'lineage' and presumably target one specific part of an antigen. It is however also interesting to consider differences between sequences that come from the same clonotype, as this knowledge allow us to learn how this lineage of B cells optimises its variable region throughout its response to the immune challenge. This typically involves the reconstruction of 'lineage trees' that connect every observation in the clonotype with its germline 'root' of the tree. Analysis of such lineage tree would reveal details of the maturation process of this group of B cells in the context of the immune response.

Another aspect of immunoglobulin function is to initiate a cascade of events in order to orchestrate an effective response towards the immune challenge, e.g. by allowing the B cell to communicate with other cell types in the immune system. This aspect of function depends on the '*constant region*' of the immunoglobulin molecule. A lineage of B cells can 'switch' their constant regions to trigger different downstream effects; this process is known as Class-Switch Recombination (CSR). CSR is uni-directional: mechanistically it involves excisions of DNA segments from the genome, therefore it is an irreversible process with a restricted pool of options to switch to, depending on the constant region gene used at the beginning of this process. The physical arrangement of the genes encoding different constant regions thus determines whether a switch is possible. Traditionally the constant region is less well-sampled in sequencing studies of immunoglobulin repertoires; conventional lineage reconstruction methods do not consider the constant region. Knowledge of CSR, however, could be used to evaluate the accuracy of the reconstructed lineage trees, as well as to infer the dynamics and interplay between CSR and affinity maturation.

The `BrepPhylo` package contains a set of functions written for the purpose of performing lineage reconstruction of a large number of B cell clonotypes obtained via high-throughput sequencing of immunoglobulin repertoires. The aim is to provide 'wrappers' around popular, existing tools for lineage reconstruction, detect Class-Switch Recombination (CSR) events from lineage trees, and generate easy-to-interpret output to analyse these lineages.

In this vignette, we first demonstrate case studies where a few ($n = 3$) clonotypes of interest are analysed for lineage reconstruction and showcase the graphical and analytical output of the `BrepPhylo` package. We then show how to wrap these functionalities into a pipeline to apply to a large number of clonotypes. This yields an automatic analysis of all lineages obtained from sequencing immunoglobulin repertoire which could be incorporated into general pipelines of repertoire data processing and mining.

[1] This can be done in various ways. We have established protocols which is available on the BRepertoire web-server. Users can nevertheless use their preferred clonotype clustering method - our input formats are flexible and expect only a input table with clone identifiers as one of the columns.

2 Installation

To install the BrepPhylo package, in R do:

```
require(devtools)
install_github("Fraternalilab/BrepPhylo", ref = "main", dependencies = TRUE)
```

The above should install BrepPhylo itself as well as its dependencies. You might encounter issues related to the package sf which is the dependency of one of the packages we import. Following these instructions from sf to resolve these issues.

BrepPhylo also uses phylogenetics packages PHYLIP and IgPhyML. Please follow the respective documentation pages to install these packages if you wish to use them through BrepPhylo functions. We included the relevant PHYLIP executable (dnapars) we used in BrepPhylo in the package so you don't have to install this by yourself; this executable should work on MacOS and Linux systems. See examples below for instructions on how to source this executable.

3 Lineage analysis of a small number of B cell clonotypes

3.1 Input data

BrepPhylo assumes a table has been generated to hold sequences and annotations of immunoglobulins observed from sequencing studies of B-cell repertoire. This could typically be obtained by, e.g. the IMGT/HighV-QUEST [1] webserver. We offer flexibility so that this input can be in the standardised AIRR format, or in any similar data table format with custom columns. HighV-QUEST output can be readily analysed as spreadsheets, or by using dedicated tools such as BRepertoire.

Below are additional annotations **required** on top of the IMGT/HighV-Quest output:

1. **Clonotype clustering**: we assume that clustering of clonotypes has already been performed on sequences. We use our protocols which have been made available on BRepertoire. Equivalently, users can use clonotype clustering procedures of their choice, such as those implemented in Change-O, or the `cellranger vdj` pipeline (if your repertoire data originate from a single-cell experiment using the 10X Genomics protocols) etc. The minimum requirement is an additional column detailing the identifier of clonotypes each sequence is mapped to. Here in our example, unique identifiers are given in column `CloneID` (observations with the same integer number are considered part of the same clone).
2. **Constant region**: we also require that the constant region of each sequence be assigned, down to the level where different antibody 'subclasses' can be distinguished (e.g. IgG1, IgG2, ... - rather than only 'IgG'). In our examples we have the columns `Class` and `Subclass`. These annotations are used for both labelling purposes (in graphical representations of lineage trees), and for analysing CSR events.

3.2 Load data

```
# load the necessary packages
library( BrepPhylo )

# load the example input data frame
input <- read.csv( system.file( "extdata/input.csv",
                                package = "BrepPhylo" ),
                   stringsAsFactors = FALSE )

# select only the columns that are needed
input <- input[ , c( "PatientID", "Class", "Subclass",
                     "CloneID", "Seq_ID", "UseAsRef",
                     "V.GENE.and.allele", "V_gapped" ) ]
# 'V_gapped' is the gapped VDJ DNA sequence

# generate an output folder to store results running this vignette
```

```
outputFolder <- path.expand( "~/Desktop/BrepPhylo-example" )
dir.create( outputFolder, showWarnings = FALSE )
```

```
# print first rows (without the sequence)
tibble::as_tibble( head( input ) )
#> # A tibble: 6 x 8
#>   PatientID Class Subclass CloneID Seq_ID   UseAsRef V.GENE.and.alle~ V_gapped
#>   <chr>     <chr> <chr>      <int> <chr>    <lgl>    <chr>            <chr>
#> 1 CV233     G     IgG1           3 CV233D0~ TRUE     IGHV1-69*06 F    caggtgcag~
#> 2 CV233     G     IgG1           3 CV233D0~ NA       IGHV1-69*06 F    caggtgcag~
#> 3 CV233     G     IgG1           3 CV233D0~ NA       IGHV1-69*06 F    caggtgcag~
#> 4 CV233     G     IgG1           3 CV233D0~ NA       IGHV1-69*06 F    caggtgcag~
#> 5 CV233     G     IgG1           3 CV233D0~ NA       IGHV1-69*06 F    caggtgcag~
#> 6 CV233     G     IgG1           3 CV233D0~ NA       IGHV1-69*06 F    caggtgcag~

# print (sub-)classes of clone 3 of patient CV233
print( unique( input[ input[ , "CloneID" ] == 3 & input[, "PatientID"] == "CV233",
                      "Subclass" ] ) )
#> [1] "IgG1" "IgM"  "IgG2"
```

In this example, it is evident that the members of clone 3 of patient CV233 show three sub-classes, namely "IgM", "IgG1" and "IgG2".

3.3 Construct phylogenetic trees and PDF alignments for selected clones

Function `cloneLineage()` takes a clone and constructs a lineage tree to show phylogenetic relationships between sequences and the annotated germline. This problem has been explored extensively in the literature and a variety of methods exist to reconstruct lineage trees. At the moment this function supports three methods:

- **'simple'**: a neighbour-joining (NJ) tree. Internally a distance matrix between sequences is constructed for the NJ algorithm to provide a solution. This can be viewed as a 'quick-and-dirty' way of lineage reconstruction; the output tree represents only one of the many possible solutions, and may not conform to biological assumptions inherent in the development of immunoglobulin lineages.
- **'dnapars'**: a maximum parsimony tree constructed using the `dnapars` program in the popular phylogenetics package `PHYLIP`. **This is the default method used in the cloneLineage function.** In our experience this is superior to the 'simple' NJ method above in terms of accuracy, without compromising too much on efficiency (such that it is still feasible to apply to a large number of clonotypes).
- **'igphyml'**: a maximum likelihood tree constructed using the IgPhyML program [2]. This implements codon-specific substitution model aware of mutational preferences in somatic hypermutation, so theoretically this is best suited to be applied to immunoglobulin sequences. However the run-time of IgPhyML is significantly longer than the other two modes (see Appendix of this vignette), rendering this not too feasible to generate lineage trees for a large number of clonotypes. (*Note:* this is kept as an legacy option. Active development and debugging has stopped for runs using the `igphyml` mode.)

3.3.1 Sequence format and germline database

We support either ungapped or gapped (according to the IMGT nomenclature) version of V-gene sequences provided by IMGT/V-Quest. Tree inference requires identification of a germline sequence as root. We offer flexibility depending on the data you are working with:

- By default, our package provides databases of *both* gapped and ungapped germline sequence for the following species: `Homo_sapiens`, `Bos_taurus`, `Gallus_gallus`, `Mus_musculus`, `Oryctolagus_cuniculus`, `Sus_scrofa`. You can specify your species in the parameter `species` in calling `cloneLineage()`.
- If you work on other species, and/or you would like to use a germline sequence set of your choice, you can pass the FASTA file containing such sequences using the parameter `germlineSet` in calling `cloneLineage()`. If you opt for this option, remember the germline sequence set **must** match V-gene

sequences in the submitted data frame in terms of gapping. If e.g. the sequences are IMGT-gapped but the germline sequence set is not, the function will return an error message. IMGT germline sequences can easily be downloaded from the IMGT website (IMGT/Gene-DB for flexible selection of sequence sets, or IMGT reference directory for IMGT-gapped sequences for each gene locus of a given species).
- Please note that the parameter `species` needs to match the naming system in the germline sequence set. If you use the default in-built germlines, please refer to the first point for allowed species names. If you use your own germline sequence set, sequences downloaded from IMGT are named by the binomial nomenclature (i.e. "Homo sapiens", "Mus musculus" etc. - notice space instead of "_") - this should be specified in the `species` argument.

Germline sequence will be selected for each clone automatically by taking the allele annotated for most of the sequences in the given clone. If gapped sequences are submitted, sequences will be processed obeying the IMGT gaps. If ungapped sequences are submitted, ClustalOmega will be called to generate a multiple sequence alignment (MSA) of the given clone with the assigned germline, prior to feeding these to the tree construction program of your choice.

3.3.2 Expected output

The `cloneLineage()` function generates the following files in a specified folder, where "ID" is the clone ID and "X" denotes the parts of the alignment (in case it exceeds more than maximum of 80 sequences):

- One folder called "clone_ID" for every clone considered, including:
- File "clone_ID.csv", holding the observations comprising the clone (i.e. an excerpt of the input data frame).
- File "clone_ID.fasta", holding the multiple sequence alignment (including the chosen germline sequence) of the clone.
- File "clone_ID.log", a comprehensive log file describing the steps of the process.
- File "clone_ID.png", a PNG of the resulting tree.
- File "clone_ID.tree", the generated tree in Newick format.
- File "clone_ID_IgPhyML_fit.stats", holding the output of the fitting step in the tree generation using function `IgPhyML` (only if used).
- File(s) "clone_ID_part_X.pdf", reporting the multiple sequence alignment in PDF format (optional).
- A folder called "plots", which holds a copy of all the tree pictures generated.
- File "colour_code.csv", which holds the colour code used in the pictures and alignments.
- File "parameters.log", holding the parameter values for the call of function `cloneLineage()`.

```
# here for illustrative purposes we select 3 clones of different sizes from patient CV233
vecClones <- c( "1284",  # 715 members
                "1548",  # 109 members
                "4279" ) #  22 members
selected_input <- input[ which( input$PatientID == "CV233" ), ]

# generate an output folder on the desktop
outputFolder <- path.expand( "~/Desktop/BrepPhylo-example/cloneLineage" )
dir.create( outputFolder, showWarnings = FALSE )

dnapars_executable <- system.file( "exe/dnapars", package = "BrepPhylo" )

# run the tree reconstruction
cloneLineage( input = selected_input,
              outputFolder = outputFolder,
              species = "Homo_sapiens",
              whichClones = vecClones,
              colourSpecification = list( colours = list( "M" = "#FF0000",
```

```
                                                        "D" = "#FF00FF",
                                                        "IgA1" = "#9999CF",
                                                        "IgA2" = "#000088",
                                                        "IgG1" = "#326600",
                                                        "IgG2" = "#58B200",
                                                        "IgG3" = "#8BFF19",
                                                        "IgG4" = "#CBFF99" ),
                                          coloursColumn = "Subclass",
                                          germlineColour = "#AAAAAA" ),
              sequenceColumn = "V_gapped",
              cloneIdColumn = "CloneID",
              germlineIdColumn = "V.GENE.and.allele",
              labelColumn = "Seq_ID",
              plotType = "phylogram",
              treeConstruction = list( type = "dnapars",
                                       parameters = list( "executable" = dnapars_executable ) ),
              makeAlignmentPDFConstruction = list( "makeAlignmentPDF" = TRUE ) )
#> Clone 1284 (715 members) has been completed.
#> Clone 1548 (109 members) has been completed.
#> Clone 4279 (22 members) has been completed.
#> A total of 0 clones (of 3) have been skipped.
```

One can see there is a variability in terms of how close each sequence is to the root (depicted bottom left of the tree) of the tree, i.e. the germline sequence. This distance signifies the similarity of each sequence to germline: a larger distance means a lower similarity, implying more mutations could be observed in this sequence.

The function `getGermlineDistance` calculates the distance-from-germline for all observations in a lineage:

```
# for clone 1548
tree <- "~/Desktop/BrepPhylo-example/cloneLineage/clone_1548/clone_1548.tree"
germline_distance <- getGermlineDistance( tree )
tibble::as_tibble( head( germline_distance ) )
#> # A tibble: 6 x 2
#>   SeqID                                        distFromGermline
#>   <chr>                                                   <dbl>
#> 1 CV233D0_G1_10768612_TTTCATCTGTTATGAT_RD105              0.106
#> 2 CV233D0_G1_10279961_TTTTGTTCTTTTGTTT_RD43               0.0563
#> 3 CV233D0_G1_10365099_TGCGTTCGCATAGGGT_RD17               0.0563
#> 4 CV233D0_G1_10516151_TAACATTTGGTCCTGT_RD89               0.0563
#> 5 CV233D0_G1_11020664_TTTTTTGTGTTGGCAT_RD35               0.0563
#> 6 CV233D0_G1_10558247_TGCCATCATTTTGTTT_RD12               0.0563
```

We find that these distance-from-germline calculations can be used to summarise the overall germline likeness of a repertoire (see below), which is a useful metric to assess the relevance of germinal centre reaction in shaping the development of this lineage.

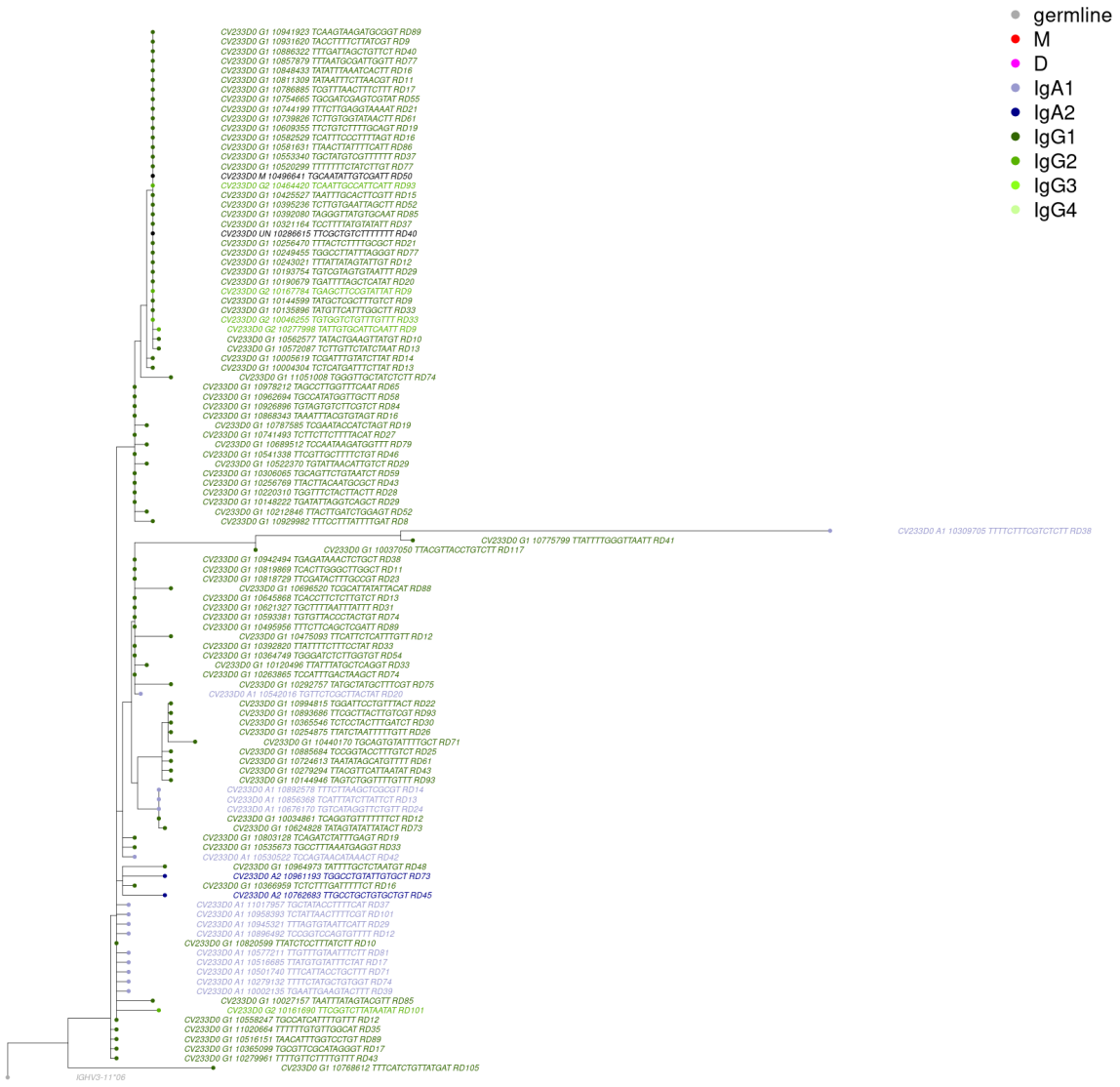


Figure 1: Clone 1548, 109 members, lineage tree reconstructed using dnapars

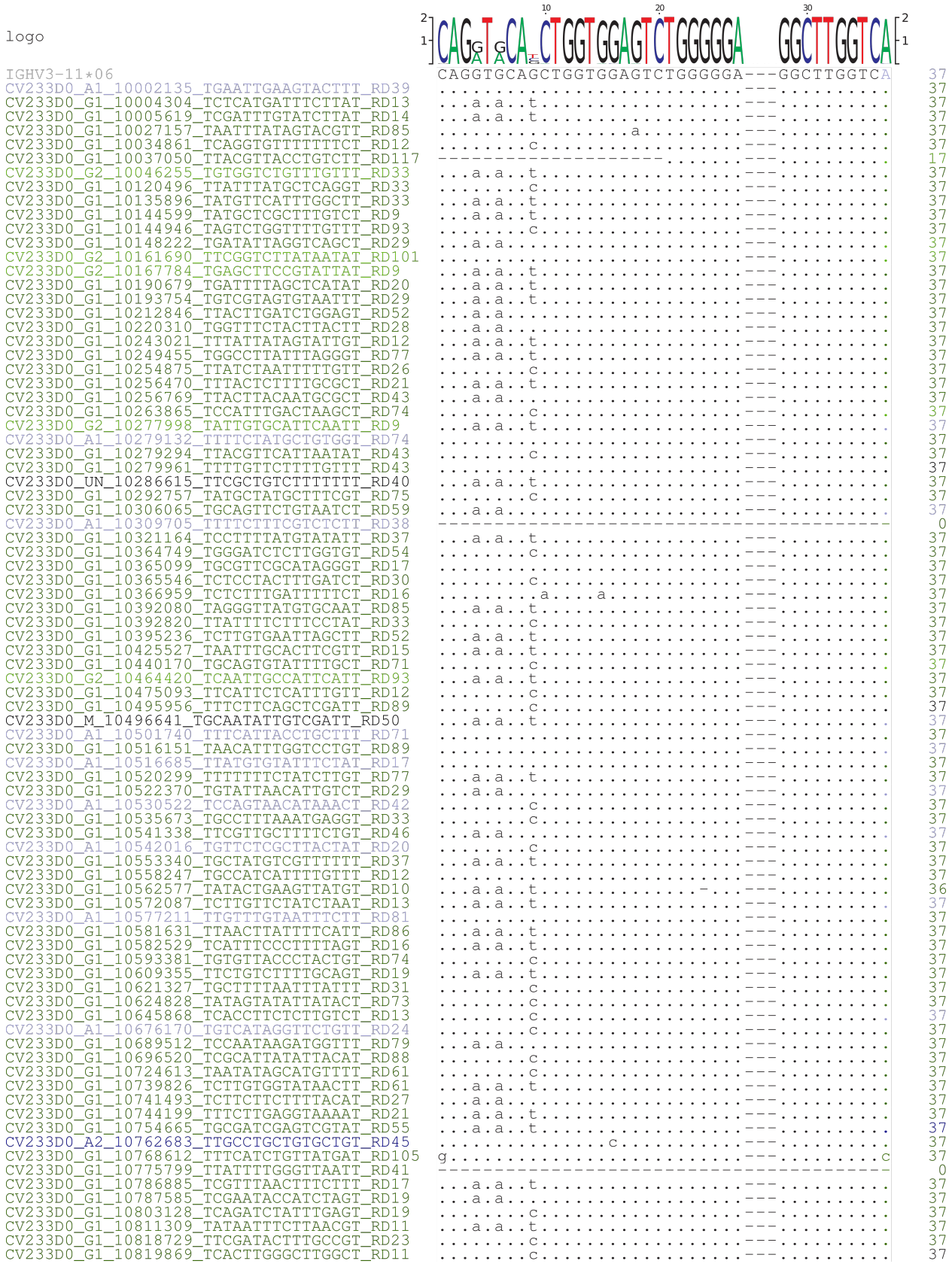


Figure 2: Clone 1548, 109 members, sequence alignment (excerpt)

3.4 Construct an arborescence tree for a B cell clone

The phylogenetic tree built above does not allow direct inference of CSR events, as the lineage reconstruction step considers only the variable region. However, with the knowledge of constant region for each sequence, we can prune the reconstructed tree and identify edges in the tree where sequences of different constant regions are connected. We provide the function `cloneArborescence`, which consider the phylogenetic tree as a directed network, removes edges in the network which violates the physical ordering of immunoglobulin constant region genes in the genome, and generates a directed minimum spanning tree rooted at the germline sequence; such tree is technically known as an 'arborescence tree'. Using Edmonds' algorithm, a tree is built satisfying the following requirements: (i) the destination of each edge uses a constant region genes that is either identical or downstream of the constant region gene for the source of the edge, (ii) the germline is set to be the true root node, (iii) all (directed) edges lead away from the root and (iv) a node has only one parent node. Although this only provides estimation of CSR (since we do not directly reconstruct 'hidden' internal nodes of the tree), this approach provides a straight-forward method to evaluate the number of class-switching events; the lengths ('distances') of edges in the tree also allows an estimation of the mutational level in the variable region at which the CSR event takes place.

The `cloneArborescence` function generates the following output in a specified output folder, where "ID" is the clone ID:

- File "clone_ID.arbo", a data frame with the edges of the arborescence tree and their lengths.
- File "clone_ID_arbo.png", a graphical representation of the arborescence tree.

```
# generate an output folder
outputFolder <- path.expand( "~/Desktop/BrepPhylo-example/arborescence" )
dir.create( outputFolder, showWarnings = FALSE )

# execute it for clone 1548
# read in the clone and the tree
clone <- read.csv(
  path.expand( "~/Desktop/BrepPhylo-example/cloneLineage/clone_1548/clone_1548.csv" )
)
tree <- "~/Desktop/BrepPhylo-example/cloneLineage/clone_1548/clone_1548.tree"

# calculate the arborescence for clone 1548
arborescence <- cloneArborescence( clone = clone,
                                   cloneID = "1548",
                                   columnSeqID = "Seq_ID",
                                   columnSubclass = "Subclass",
                                   tree = tree,
                                   outputFolder = outputFolder )
```

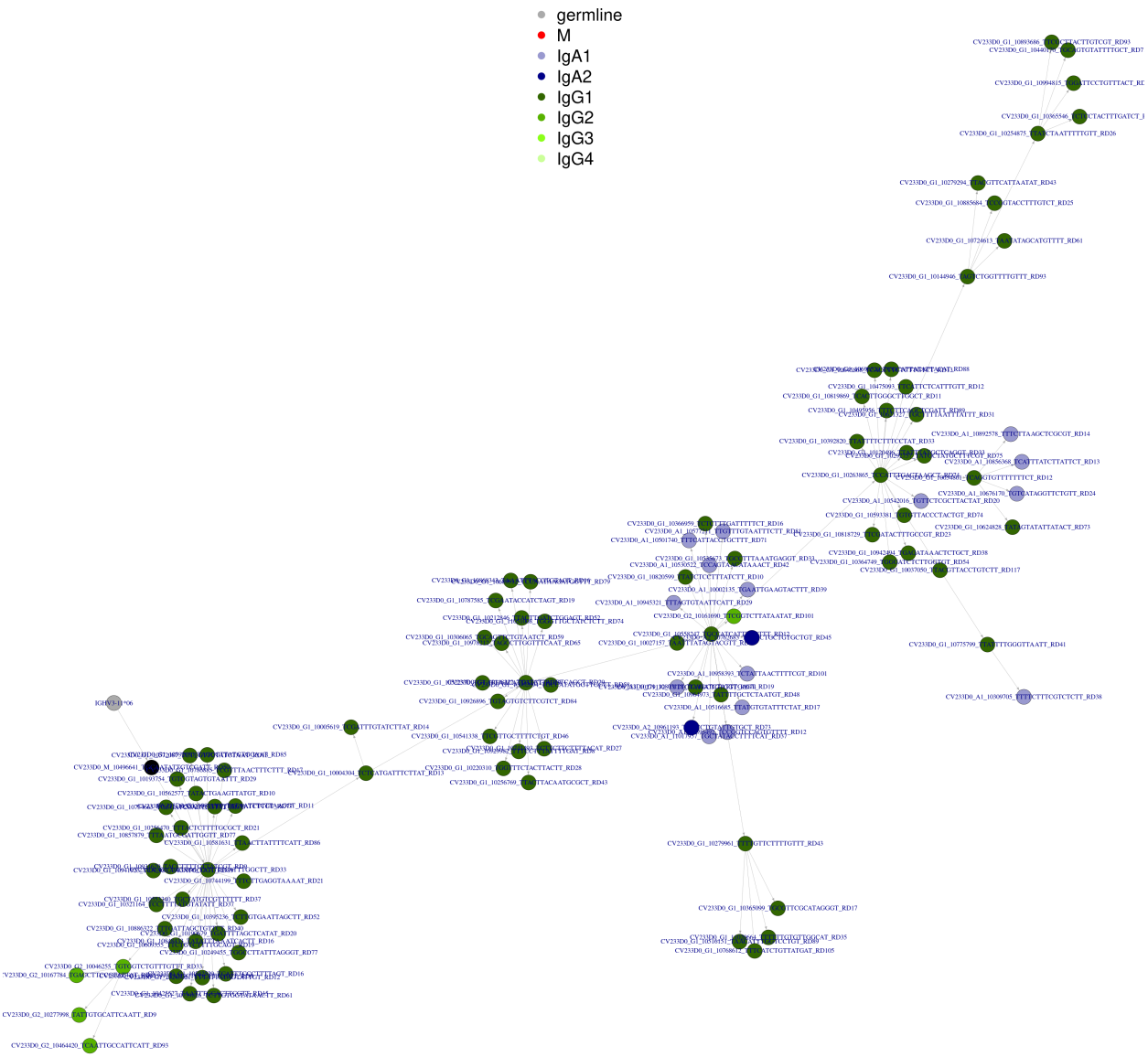


Figure 3: Clone 188, 53 members, arborescence tree (simple model)

This arborescence lineage can also be visualised in a plot by ordering the nodes (i.e. sequences) by their distance from the root, groupng them by their ancestors in the graph, and plotting the subclass distribution across the distance-from-root axis:

```
arborescence$lineage_plot + cowplot::theme_cowplot()
```

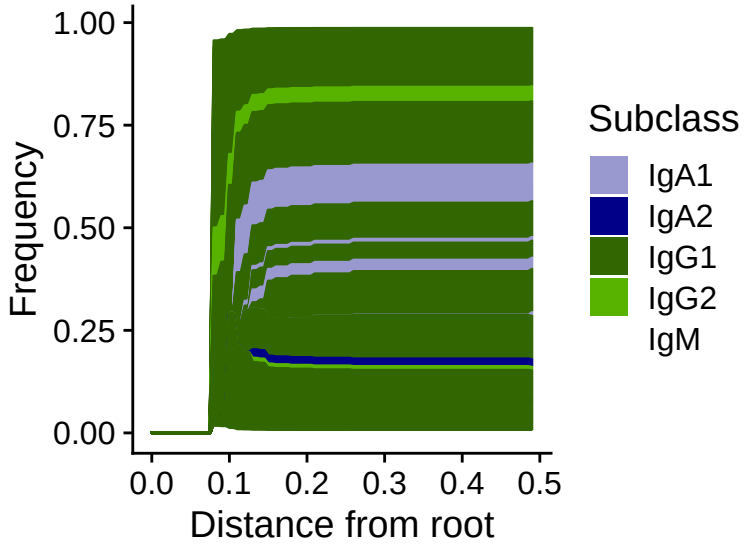


Figure 4: Clone 1548 arborescence tree visualised as a 'lineage plot' highlighting emergence of sequences of different isotypes.

With this plot we can easily trace class-switching events in the clone (narrow "tips" from left to right):

- G1 → G2 switch at the beginning of the lineage;
- Followed by G1 → A1;
- Later in the lineage switches to A2 and G2 limited to a small number of observations.

The class-switching events are listed in the `csr_events` element in the `cloneArborescence()` output:

```
tibble::as_tibble( head( arborescence$csr_events ) )
#> # A tibble: 6 x 5
#>   startLabel           endLabel           startIsotype endIsotype distFromGermline
#>   <chr>                <chr>              <fct>        <fct>                 <dbl>
#> 1 CV233D0_M_10496641~ CV233D0_G1_10135~ IgM          IgG1                  0.075
#> 2 CV233D0_G1_1013589~ CV233D0_G2_10046~ IgG1         IgG2                  0.075
#> 3 CV233D0_G1_1055824~ CV233D0_A2_10961~ IgG1         IgA2                  0.112
#> 4 CV233D0_G1_1055824~ CV233D0_A2_10762~ IgG1         IgA2                  0.112
#> 5 CV233D0_G1_1055824~ CV233D0_A1_10530~ IgG1         IgA1                  0.105
#> 6 CV233D0_G1_1055824~ CV233D0_A1_11017~ IgG1         IgA1                  0.103
```

All of these could be traced on the lineage plot above. Note that the `distFromGermline` refers to the point this event is estimated to occur - this can be a proxy of the mutational level of the V-domain at which CSR occurs.

4 Batch lineage reconstruction analysis of a large number of clonotypes

While the above showcases the detailed analysis on lineages possible using the `BrepPhylo` functionalities, often we aim to apply this sort of analyses on a large number of clonotypes, e.g. all clones sampled in the immunoglobulin repertoire of an individual. Here some of the outputs shown above (e.g. PDF alignments, graphical representation of trees) are less important; we are instead interested in quick and efficient reconstruction of lineages for these clonotypes, and easy-to-interpret metrics which can be quickly estimated from trees.

The function `doBatchCloneAnalysis` is designed for this purpose. This function is a wrapper for the `cloneLineage` and `cloneArborescence` functions explained above, to apply it over all clonotypes present in an input data frame. The output of `doBatchCloneAnalysis` include:

- in the given output folder, all identical output as `cloneLineage`.
- a subfolder named 'arborescence' containing all identical output as `cloneArborescence`.
- in R, `doBatchCloneAnalysis` returns a nested list: each element of the list represents one clonotype, itself a list containing two data frames:
- `distances`: the output from `getGermlineDistance` of the calculated lineage tree.
- `csr_events`: the list of CSR events output from `cloneArborescence`.

```
# To illustrate use all sequences from patient CV325
selected_input <- input[ which( input$PatientID == "CV325" ), ]

# generate output folder
outputFolder <- path.expand( "~/Desktop/BrepPhylo-example/batchAnalysis" )
dir.create( outputFolder, showWarnings = FALSE )

batch_results <- doBatchCloneAnalysis(
  selected_input,
  outputFolder = outputFolder,
  species = "Homo_sapiens",
  sequence_column = "V_gapped",
  IGHVgeneandallele_column = "V.GENE.and.allele",
  plotFormat = "pdf",
  phyloTreeType = "dnapars",
  phyloTreeOptions = list( "executable" = dnapars_executable ),
  useTempDir = FALSE
)
# see above: doBatchCloneAnalysis returns a list in R which can be useful for
# further analysis. We can save it in a R data file, so we can load these
# separately later on (with readRDS()), without re-running the analysis.
saveRDS(batch_results, paste0(outputFolder, '/CV325_batch_analysis.rds'))
```

The `doBatchCloneAnalysis` function is very efficient: for this example considering 8420 sequences from 252 clonotypes, it takes ≈ 5 minutes to perform lineage analyses in one batch. This function can also be easily wrapped in e.g. a `for` loop to process clonotypes from multiple repertoires. **Note:** That said, the computing time scales exponentially as the clone size increases, so be prepared to wait for hours/days if you have much more sequences to analyse than what is demonstrated here!

4.1 Assessing Germline Likeness of a repertoire

```
# We have included in the package the pre-computed germline distances of
# sequences in the 'input' data frame, here use this as an example
distFromGermline <- system.file( "extdata/input_GermlineDistances.csv", package = "BrepPhylo")
distFromGermline <- read.csv(distFromGermline)

# this summarise for each clone the median germline distance
# and place each clone in the distribution of this median-distance over all clones
germlineDistance_summary <- summariseGermlineDistance(
```

```
  distFromGermline, dist_column = "distFromGermline",
  cloneID_column = "CloneID",
  summarise_variables = c( "PatientID", "CloneID" )
)

# we can plot this as a curve to show the overall distribution
library(ggplot2)
ggplot(germlineDistance_summary, aes(x = dist_median, y = clone_order,
                                     group = PatientID, colour = PatientID)) +
  geom_line() + scale_colour_discrete(name = "") + cowplot::theme_cowplot() +
  xlab("median distance from germline") + ylab("clone percentile")
```

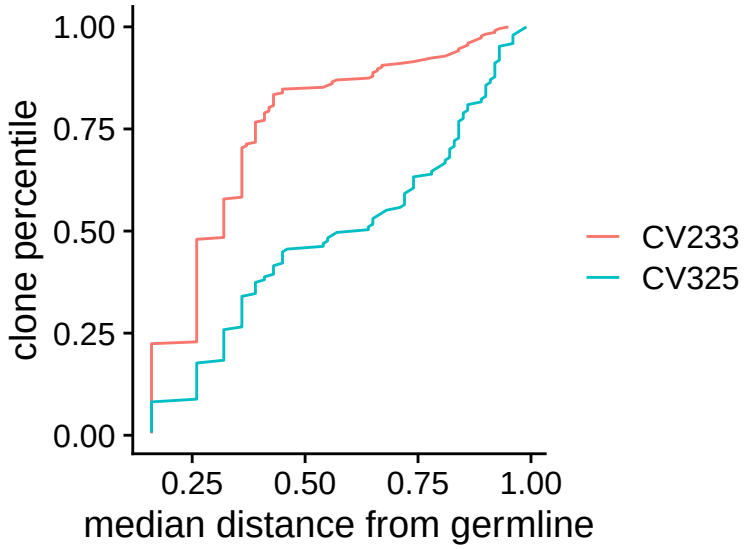


Figure 5: Distribution of distance-from-germline of clonotypes observed in the repertoires of two patients.

Here, each clone is summarised and ranked by its median distance-from-germline metric. The rank is shown on the y-axis and the actual median distance-from-germline on the x-axis. This curve shows the overall level of mutation (and inversely, similarity to the germline) over all clonotypes in a repertoire. Comparing these two curves, we can conclude that patient CV233 overall has a higher similarity to germline (i.e. fewer mutations accumulated) compared to patient CV325.

This phenomenon can be summarised as a metric: by quantifying the Area-under-curve (AUC) this metric quantifies what we call 'Germline Likeness', i.e. for this given repertoire, how much are the clonotypes similar to germline?

```
# set up a table containing metadata with which you want to
# obtain a summary of Germline Likeness
# here simply set up one column with PatientID
patient_details <- data.frame( PatientID = unique( distFromGermline[, c("PatientID") ] ) )

germline_likeness <- getGermlineLikeness(
  germlineDistance_summary,
  metadata_table = patient_details,
  summarise_variables = "PatientID"
)
#> Warning in `[<-.factor`(`*tmp*`, list, value = ""): invalid factor level, NA
#> generated

#> Warning in `[<-.factor`(`*tmp*`, list, value = ""): invalid factor level, NA
#> generated
tibble::as_tibble( germline_likeness )
#> # A tibble: 2 x 2
```

```
#>   PatientID GermlineLikeness
#>   <fct>                <dbl>
#> 1 CV233                0.598
#> 2 CV325                0.410
```

The higher this metric, the closer the clonotypes in general are to their corresponding germline. One can calculate the Germline Likeness metric over repertoires of many different donors and perform statistical comparisons on it to evaluate differences in mutational levels in repertoires sampled from different disease conditions.

4.2 Summarising CSR events in (groups of) repertoire data

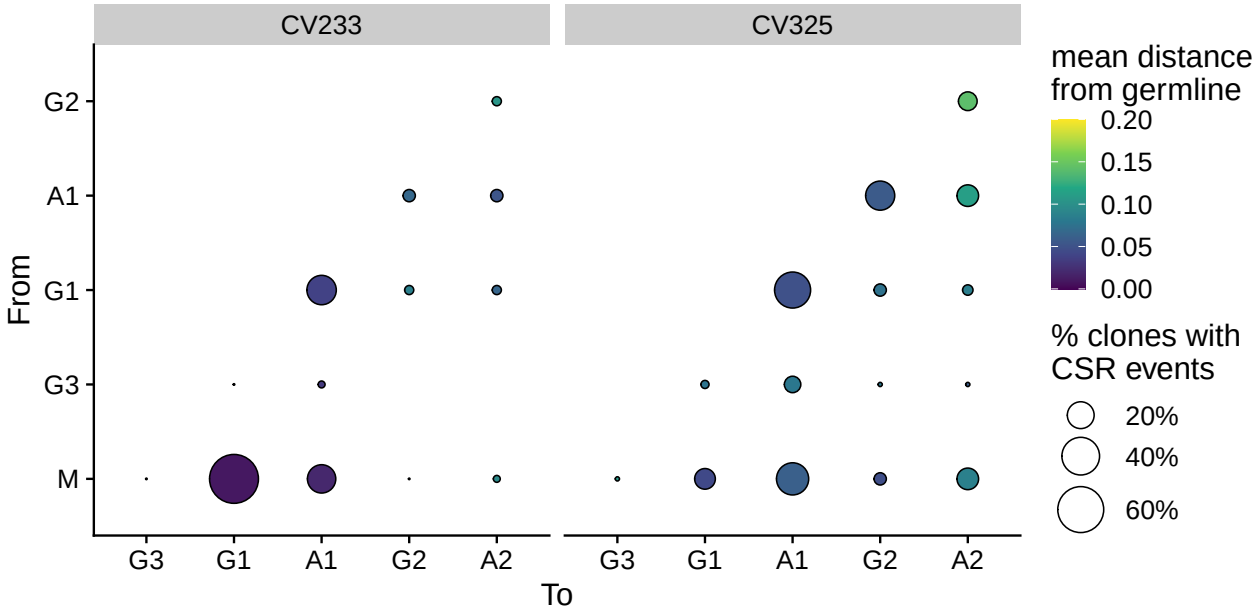
We have seen in the above that `BrepPhylo` is capable of analysing CSR events for a given lineage. `doBatchCloneAnalysis` gives detailed lists of CSR events observed in each lineage of the given batch of clones. We can further process such list and produce graphical summary of the distribution of CSR events, both in terms of the frequency of such events (broken down by the start- and end-points of the switches), and the distance-from-germline at which such switches occur.

```
# we provide doBatchCloneAnalysis results for data from the
# two individuals included here in this vignette
cv233 <- system.file( "extdata/CV233_batch_analysis.rds", package = "BrepPhylo")
cv325 <- system.file( "extdata/CV325_batch_analysis.rds", package = "BrepPhylo")
cv233 <- readRDS( cv233 )
cv325 <- readRDS( cv325 )

# the getSummaryFromBatch function flattens the output to two
# elements: a table containing  distance-from-germline (demonstrated
# above) and another table of all CSR events observed in lineages
cv233 <- getSummaryFromBatch( cv233 )
cv233 <- cv233$csr_events
cv325 <- getSummaryFromBatch( cv325 )
cv325 <- cv325$csr_events
# just add an extra column with PatientID
cv233$PatientID <- "CV233"
cv325$PatientID <- "CV325"
# we can combine the two tables
csr_details <- rbind( cv233, cv325 )
csr_summary <- summariseCSR( csr_details,
                             dist_column = "distFromGermline",
                             cloneID_column = "CloneID",
                             summarise_variables = "PatientID")

# next we can plot a graphical summary
# first set the order of the isotypes so that they reflect
# the actual physical order on the genome (hence the
# order possible in CSR)
csr_summary$startIsotype <- factor(csr_summary$startIsotype,
                                   levels = c("IgM", "IgG3", "IgG1", "IgA1",
                                              "IgG2", "IgG4", "IgE", "IgA2"),
                                   labels = c("M", "G3", "G1", "A1", "G2",
                                              "G4", "E", "A2"))
csr_summary$endIsotype <- factor(csr_summary$endIsotype,
                                 levels = c("IgM", "IgG3", "IgG1", "IgA1",
                                            "IgG2", "IgG4", "IgE", "IgA2"),
                                 labels = c("M", "G3", "G1", "A1", "G2",
                                            "G4", "E", "A2"))

# plot it; it uses ggplot2 so you can extend this with ggplot2 functions
# here separate into different panels using the column 'PatientID'
library(ggplot2)
plotCSRsummary( csr_summary ) + facet_wrap(~ PatientID)
```



This graphical summary shows both the amount (the size of bubbles) of CSR and the distance-from-germline (colour in the heat-scale fill of the bubbles) at which CSR occurs, separate by subclasses. We can see here that a large amount of switches to IgM to IgG1 occurs in patient CV233, at a very close distance from the germline (i.e. very few mutations have been accumulated at the time of the switch). In the other patient there appear to have a higher mutational rate, and more switches to IgG2. This graphical summary allows us to interpret the complex CSR dynamics in a simple bubble plot.

5 Appendix

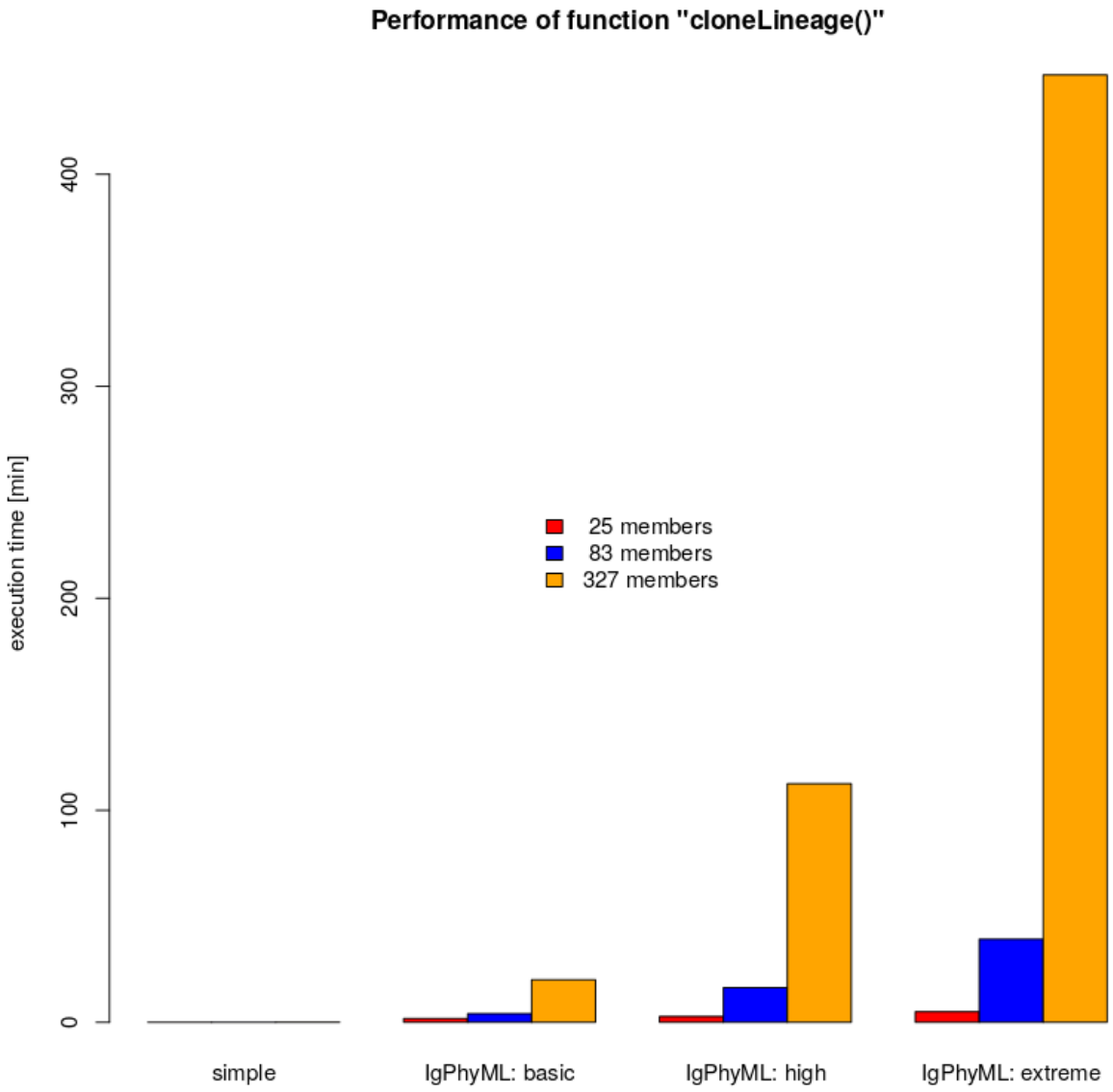


Figure 6: Performance benchmark of `cloneLineage()` run on three clones of different sizes: Computing time on a 4GB iMac (model circa 2014)

Bibliography

1. Brochet,X., Lefranc,M.-P. and Giudicelli,V. (2008) IMGT/V-QUEST: The highly customized and integrated system for IG and TR standardized V-J and V-D-J sequence analysis. *Nucleic Acids Research*, **36**, W503–W508.

2. Hoehn,K.B., Lunter,G. and Pybus,O.G. (2017) A Phylogenetic Codon Substitution Model for Antibody Lineages. *Genetics*, **206**, 417–427.