

Package ‘Pv3Rs’

July 31, 2025

Title Estimate the Cause of Recurrent Vivax Malaria using Genetic Data

Version 0.0.2

Description Plot malaria parasite genetic data on two or more episodes. Compute per-person posterior probabilities that each Plasmodium vivax (Pv) recurrence is a recrudescence, relapse, or reinfection (3Rs) using per-person P. vivax genetic data on two or more episodes and a statistical model described in Taylor, Foo and White (2022) <[doi:10.1101/2022.11.23.22282669](https://doi.org/10.1101/2022.11.23.22282669)>. Plot per-recurrence posterior probabilities.

License MIT + file LICENSE

URL <https://aimeertaylor.github.io/Pv3Rs/>,
<https://github.com/aimeertaylor/Pv3Rs>

BugReports <https://github.com/aimeertaylor/Pv3Rs/issues>

Depends R (>= 3.5)

Imports dplyr, fields, grDevices, igraph, matrixStats, methods,
multicool, partitions, purrr, RColorBrewer

Suggests codetools, gtools, knitr, Matrix, plyr, R.rsp, rmarkdown,
testthat (>= 3.0.0), tictoc, utils

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Aimee Taylor [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-2337-8992>>),
Yong See Foo [aut] (ORCID: <<https://orcid.org/0000-0003-3010-9106>>),
Tymoteusz Kwiecinski [com],
Duncan Murdoch [ctb],

Mans Magnusson [ctb],
 Institut Pasteur [cph],
 European Union, Project 101110393 [fnd]

Maintainer Aimee Taylor <aimee.taylor@pasteur.fr>

Repository CRAN

Date/Publication 2025-07-31 10:10:02 UTC

Contents

compute_posterior	2
determine_MOIs	5
enumerate_RGs	6
fs_VHX_BPD	7
plot_data	8
plot_RG	10
plot_simplex	11
project2D	13
RG_to_igraph	14
sample_RG	15
ys_VHX_BPD	15
Index	17

compute_posterior	<i>Compute posterior probabilities of <i>P. vivax</i> recurrence states</i>
-------------------	---

Description

Computes per-person posterior probabilities of *P. vivax* recurrence states — recrudescence, relapse, reinfection — using per-person genetic data on two or more episodes. For usage, see **Examples** below and **Demonstrate Pv3Rs usage**. For a more complete understanding of compute_posterior output, see **Understand posterior probabilities**.

Note: The progress bar may increment non-uniformly (see **Details**); it may appear stuck when computations are ongoing.

Usage

```
compute_posterior(
  y,
  fs,
  prior = NULL,
  MOIs = NULL,
  return.RG = FALSE,
  return.logp = FALSE,
  progress.bar = TRUE
)
```

Arguments

y	List of lists encoding allelic data. The outer list contains episodes in chronological order. The inner list contains named markers per episode. Marker names must be consistent across episodes. NA indicates missing marker data; otherwise, specify a per-marker vector of distinct alleles detected (presently, compute_posterior() does not support data on the proportional abundance of detected alleles). Repeat alleles and NA entries within allelic vectors are ignored. Allele names are arbitrary, allowing for different data types, but must correspond with frequency names.
fs	List of per-marker allele frequency vectors, with names matching marker names in y. Per-marker alleles frequencies must contain one frequency per named allele, with names matching alleles in y. Per-marker frequencies must sum to one.
prior	Matrix of prior probabilities of recurrence states per episode, with rows as episodes in chronological order, and columns named "C", "L", and "I" for recrudescence, relapse and reinfection, respectively. Row names are ignored. If NULL (default), per-episode recurrence states are assumed equally likely <i>a priori</i> .
MOIs	Vector of per-episode multiplicities of infection (MOIs); because the Pv3Rs model assumes no genotyping errors, MOIs must be greater than or equal to the most parsimonious MOI estimates compatible with the data; see determine_MOIs(y). These are the estimates used when MOIs = NULL (default).
return.RG	Logical; returns the relationship graphs (default FALSE). Automatically set to TRUE if return.logp = TRUE.
return.logp	Logical; returns the log-likelihood for each relationship graph (default FALSE). Setting TRUE disables permutation symmetry optimisation and thus increases runtime, especially when MOIs are large. Does not affect the output of the posterior probabilities; for an an example of permutation symmetry, see Exploration of relationship graphs in Demonstrate Pv3Rs usage .
progress.bar	Logical; show progress bars (default TRUE). Note that the progress bar may update non-uniformly.

Details

compute_posterior() computes posterior probabilities proportional to the likelihood multiplied by the prior. The likelihood sums over:

- ways to phase allelic data onto haploid genotypes
- graphs of relationships between haploid genotypes
- ways to partition alleles into clusters of identity-by-descent

We enumerate all possible relationship graphs between haploid genotypes, where pairs of genotypes can either be clones, siblings, or strangers. The likelihood of a sequence of recurrence states can be determined from the likelihood of all relationship graphs compatible with said sequence. More details on the enumeration of relationship graphs can be found in [enumerate_RGs](#). For each relationship graph, the model sums over all possible identity-by-descent partitions. Because some graphs are compatible with more partitions than others, the log p(Y|RG) progress bar may advance non-uniformly. We do not recommend running 'compute_posterior()' when the total genotype count (sum of MOIs) exceeds eight because there are too many relationship graphs.

Notable model assumptions and limitations:

- All siblings are regular siblings
- Recrudescence parasites derive only from the immediately preceding episode
- Recrudescence, relapse and reinfection are mutually exclusive
- Undetected alleles, genotyping errors, and *de novo* mutations are not modelled
- Population structure and various other complexities that confound molecular correction are not modelled

Value

List containing:

marg Matrix of marginal posterior probabilities for each recurrence, with rows as recurrences and columns as "C" (recrudescence), "L" (relapse), and "I" (reinfection). Each marginal probability sums over a subset of joint probabilities. For example, the marginal probability of "C" at the first of two recurrences sums over the joint probabilities of "CC", "CL", and "CI".

joint Vector of joint posterior probabilities for each recurrence state sequence; within a sequence "C", "L", and "I" are used as above.

RGs List of lists encoding relationship graphs; returned only if `return.RG = TRUE` (default FALSE), and with log-likelihoods if `return.logp = TRUE` (default FALSE). A relationship graph encoded as a list can be converted into a `igraph` object using `RG_to_igraph` and thus plotted using `plot_RG`. For more details on relationship graphs, see `enumerate_RGs`.

Examples

```
# Numerically named alleles
y <- list(enrol = list(m1 = c('3','2'), m2 = c('1','2')),
         recur1 = list(m1 = c('1','4'), m2 = c('1')),
         recur2 = list(m1 = c('1'), m2 = NA))
fs <- list(m1 = c('1' = 0.78, '2' = 0.14, '3' = 0.07, '4' = 0.01),
         m2 = c('1' = 0.27, '2' = 0.73))
compute_posterior(y, fs, progress.bar = FALSE)

# Arbitrarily named alleles, plotting per-recurrence posteriors
y <- list(enrolment = list(marker1 = c("Tinky Winky", "Dipsy"),
                             marker2 = c("Tinky Winky", "Laa-Laa", "Po")),
         recurrence = list(marker1 = "Tinky Winky",
                             marker2 = "Laa-Laa"))
fs <- list(marker1 = c("Tinky Winky" = 0.4, "Dipsy" = 0.6),
         marker2 = c("Tinky Winky" = 0.1, "Laa-Laa" = 0.1, "Po" = 0.8))
plot_simplex(p.coords = compute_posterior(y, fs, progress.bar = FALSE)$marg)

# Episode names are cosmetic: "r1_prior" is returned for "r2"
y <- list(enrol = list(m1 = NA), r2 = list(m1 = NA), r1 = list(m1 = NA))
prior <- matrix(c(0.6,0.7,0.2,0.3,0.2,0), ncol = 3,
              dimnames = list(c("r1_prior", "r2_prior"), c("C", "L", "I")))
suppressMessages(compute_posterior(y, fs = list(m1 = c(a = 1)), prior))$marg
```

```

prior

# Prior is returned when all data are missing
y_missing <- list(enrol = list(m1 = NA), recur = list(m1 = NA))
suppressMessages(compute_posterior(y_missing, fs = list(m1 = c("A" = 1))))

# Return of the prior re-weighted to the exclusion of recrudescence:
suppressMessages(compute_posterior(y_missing, fs = list(m1 = c("A" = 1)),
                                MOIs = c(1,2)))
# (Recrudescing parasites are clones of previous blood-stage parasites. The
# Pv3R model assumes no within-host de-novo mutations and perfect allele
# detection. As such, recrudescence is incompatible with an MOI increase on
# the preceding infection.)

# Beware provision of unpaired data: the prior is not necessarily returned;
# for more details, see link above to "Understand posterior estimates"
y <- list(list(m1 = c('1', '2')), list(m1 = NA))
fs <- list(m1 = c('1' = 0.5, '2' = 0.5))
suppressMessages(compute_posterior(y, fs))$marg

```

determine_MOIs

Determine multiplicities of infection (MOIs)

Description

Returns a MOI estimate for each episode based on allelic diversity across markers.

Usage

```
determine_MOIs(y, return.names = FALSE)
```

Arguments

<code>y</code>	List of lists encoding allelic data; see compute_posterior for more details. The outer list contains episodes in chronological order. The inner list contains named markers per episode. For each marker, one must specify an allelic vector: a set of distinct alleles detected at that marker; or NA if marker data are missing.
<code>return.names</code>	Logical; if TRUE and <code>y</code> has named episodes, episode names are returned.

Details

A true MOI is a number of genetically distinct groups of clonal parasites within an infection. Give or take *de novo* mutations, all parasites within a clonal group share the same DNA sequence, which we call a genotype. As such, MOIs are distinct parasite genotype counts. Under the Pv3Rs model assumption that there are no genotyping errors, the true MOI of an episode is greater than or equal

to the maximum distinct allele count for any marker in the data on that episode. In other words, under the assumption of no genotyping errors, maximum distinct allelic counts are the most parsimonious MOI estimates compatible with the data. By default, these MOI estimates are used by `compute_posterior`.

Value

Numeric vector containing one MOI estimate per episode, each estimate representing the maximum number of distinct alleles observed at any marker per episode.

Examples

```
y <- list(enrol = list(m1 = c("A", "B"), m2 = c("A"), m3 = c("C")),
         recur = list(m1 = c("B"), m2 = c("B", "C"), m3 = c("A", "B", "C")))
determine_MOIs(y) # returns c(2, 3)
```

enumerate_RGs

Enumerate relationship graphs (RGs)

Description

An RG is a graph over all per-person parasite genotypes (each as a vertex), with edges between clone and sibling genotypes. Valid RGs satisfy:

- Subgraphs induced by clone edges are cluster graphs.
- Subgraphs induced by clone plus sibling edges are cluster graphs.
- Clone edges only link genotypes from different episodes.

Usage

```
enumerate_RGs(MOIs, igraph = TRUE, progress.bar = TRUE)
```

Arguments

MOIs	Vector of per-episode multiplicities of infection (MOIs), i.e., numbers of per-episode genotypes / vertices.
igraph	Logical; returns RGs as igraph objects (default TRUE).
progress.bar	Logical; show progress bar (default TRUE).

Details

RGs are generated by enumerating nested set partitions under specific constraints; see [Understand graph and partition enumeration](#). Each nested set partition is an RG. Clone edges induce a cluster graph, equivalent to a partition of genotypes, with no intra-episode clones allowed. Sibling edges refine the clone partition, with no constraints (intra-episode siblings allowed). Each nested set partition is encoded as a list. Each partition is represented by a list of vectors (either `clone` or `sib`) and by a membership vector (either `clone.vec` or `sib.vec`). By default, an RG encoded as a list is converted to an igraph object.

Value

A list of RGs. If `igraph = FALSE`, each RG is a list of length four with:

clone List of vectors encoding genotypes per clonal cell.

clone.vec Numeric vector with the clonal cell membership of each genotype.

sib List of vectors encoding clonal cells per sibling cell.

sib.vec Numeric vector with the sibling cell membership of each clonal cell.

If `igraph = TRUE` (default), each RG is encoded as an `igraph` object (see [RG_to_igraph](#)).

Examples

```
graphs <- enumerate_RGs(c(2, 1, 2), progress.bar = FALSE) # nine graphs
```

`fs_VHX_BPD`*Allele frequencies computed using example Plasmodium vivax data*

Description

The posterior mean of a multinomial-Dirichlet model with uniform prior fit to data on allele prevalence in initial episodes of `ys_VHX_BPD`. Because the model is fit to allele prevalence (observed) not allele frequency (requires integrating-out unknown multiplicities of infection) it is liable to underestimate the frequencies of common alleles and overestimate those of rare but detected alleles.

Usage

```
fs_VHX_BPD
```

Format

A list of nine markers; for each marker a named vector of allele frequencies that sum to one.

Source

- <https://zenodo.org/records/3368828>
- https://github.com/aimeertaylor/Pv3Rs/blob/main/data-raw/fs_VHX_BPD.R

plot_data

Plots the data

Description

Plots allelic data as a grid of coloured rectangles.

Usage

```
plot_data(
  ys,
  fs = NULL,
  person.vert = FALSE,
  mar = c(1.5, 3.5, 1.5, 1),
  gridlines = TRUE,
  palette = RColorBrewer::brewer.pal(12, "Paired"),
  marker.annotate = TRUE,
  legend.lab = "Allele frequencies",
  legend.line = 0.2,
  legend.ylim = c(0.05, 0.2),
  cex.maj = 0.7,
  cex.min = 0.5,
  cex.text = 0.5,
  x.line = 0.2,
  y.line = 2.5
)
```

Arguments

ys	Nested list of per-person, per-episode, per-marker allelic data; see Examples and <code>compute_posterior()</code> for the expected per-person structure.
fs	A per-marker list of numeric vectors of allele frequencies. If NULL (default), only the alleles present in ys are shown in the the legend, with all per-marker alleles represented equally. Because the colour scheme is adaptive, the same allele may have different colours given different ys. If fs is specified, all alleles in fs feature in the legend with areas proportional to allele frequencies, so that common alleles occupy larger areas than rarer alleles. Specify fs to fix the allele colour scheme across plots of different ys.
person.vert	Logical. If TRUE (default), person IDs are printed vertically; otherwise, they are printed horizontally.
mar	Vector of numbers of lines of margin for the main plot; see mar entry of <code>par</code> .
gridlines	Logical. If true (default), white grid lines separating people and markers are drawn.
palette	Colour palette for alleles, see the Value section of <code>brewer.pal</code> . Generally, colours are interpolated: if a marker has d possible alleles, then the colours used

	are the $1/(d+1)$, \dots , $d/(d+1)$ quantiles of the palette to ensure that markers with different allele counts use different colours.
marker.annotate	Logical. If true (default), the names of the alleles are printed on top of their colours in the legend.
legend.lab	Label for the axis of the legend. Defaults to "Allele frequencies". Set to NA to omit the label; if so, consider adjusting legend.ylim to use more plotting space.
legend.line	Distance (in character heights) from the colour bar to the legend label (defaults to 1.5).
legend.ylim	Vector specifying the y-coordinate limits of the legend in device coordinates (between 0 and 1). Defaults to c(0.05, 0.2).
cex.maj	Numeric; font scaling of major axis labels.
cex.min	Numeric; font scaling of minor axis labels.
cex.text	Numeric; font scaling of the allele labels.
x.line	Distance between top x-axis and x-axis label, defaults to 0.2.
y.line	Distance between left y-axis and y-axis label, defaults to 2.5.

Details

This function plots alleles (colours), which are observed in different episodes (columns), on different markers (rows), with episodes grouped by person. Per-person episodes are plotted from left to right in chronological order. If multiple alleles are detected for a marker within an episode, the corresponding grid element is subdivided vertically into different colours.

By default, markers are ordered lexicographically. If `fs` is provided, markers are ordered to match the order within `fs`.

The legend depicts the alleles for each marker in the same vertical order as the main plot. The default colour scheme is adaptive, designed to visually differentiate the alleles as clearly as possible by maximizing hue contrast within a qualitative palette. Interpolation is used to make different colour palettes for markers with different numbers of possible alleles. The names of the alleles are printed on top of their colours if `marker.annotate` is set to TRUE.

Value

None

Examples

```
oldpar <- par(no.readonly = TRUE) # Store user's options before plotting

# Plot example Plasmodium vivax data set:
mar <- c(2, 3.5, 1.5, 1) # extra vertical margin for vertical person labels
plot_data(ys = ys_VHX_BPD, person.vert = TRUE, mar = mar, legend.lab = NA)
plot_data(ys = ys_VHX_BPD, person.vert = TRUE, mar = mar, legend.lab = NA,
          fs = fs_VHX_BPD)
plot_data(ys = ys_VHX_BPD, person.vert = TRUE, mar = mar, legend.lab = NA,
          fs = fs_VHX_BPD, marker.annotate = FALSE)
```

```
# Demonstrating the adaptive nature of the colour scheme:
ys <- ys_VHX_BPD["VHX_52"] # A single person
plot_data(ys, fs = fs_VHX_BPD, marker.annotate = FALSE) # Colours match above
plot_data(ys) # Colours and the legend adapt to alleles detected in VHX_52

par(oldpar) # Restore user's options
```

plot_RG

Plot a relationship graph (RG)

Description

This function is a wrapper around `plot.igraph`, written to group parasite genotypes by episode both spatially and using vertex colour (specifically, parasite genotypes within episodes are vertically distributed with some horizontal jitter when `layout.by.group = TRUE` (default), and `equicolored`), and to ensure clone and sibling edges are plotted using different line types.

Usage

```
plot_RG(
  RG,
  layout.by.group = TRUE,
  vertex.palette = "Set2",
  edge.lty = c(sibling = "dashed", clone = "solid"),
  edge.col = c(sibling = "black", clone = "black"),
  edge.width = 1.5,
  ...
)
```

Arguments

RG	igraph object encoding an RG; see RG_to_igraph .
layout.by.group	Logical; if TRUE (default) overrides the default layout of <code>plot.igraph</code> so that vertices that represent parasite genotypes from different episodes are distributed horizontally and vertices that represent genotypes within episodes are distributed vertically.
vertex.palette	A character string specifying an RColorBrewer palette. Overrides the default palette of <code>plot.igraph</code> .
edge.lty	Named vector of edge line types corresponding to different relationships.
edge.col	Named vector of edge colours corresponding to different relationships.
edge.width	Overrides the default edge.width of <code>plot.igraph</code> .
...	Additional arguments to pass to <code>plot.igraph</code> , e.g., <code>edge.curved</code> .

Value

None

Provenance

This function was adapted from plot_Vivax_model at https://github.com/jwatowatson/RecurrentVivax/blob/master/Genetic_Model/iGraph_functions.R.

Examples

```
RGs <- enumerate_RGs(c(2, 1, 1), progress.bar = FALSE)
oldpar <- par(no.readonly = TRUE) # record user's options
par(mfrow = c(3, 4), mar = c(0.1, 0.1, 0.1, 0.1))
for (i in 12:23) {
  plot_RG(RGs[[i]],
    edge.col = c(sibling = "gray", clone = "black"),
    edge.lty = c(sibling = "dotted", clone = "solid"),
    edge.curved = 0.1)
  box()
}
par(oldpar) # restore user's options
```

plot_simplex

*Plots a 2D simplex***Description**

Plots a 2D simplex (a triangle with unit sides centered at the origin) onto which per-recurrence posterior probabilities of recrudescence, relapse, reinfection (or any other probability triplet summing to one) can be projected; see [project2D\(\)](#) and **Examples** below.

Usage

```
plot_simplex(
  v.labels = c("Recrudescence", "Relapse", "Reinfection"),
  v.cutoff = 0.5,
  v.colours = c("yellow", "purple", "red"),
  plot.tri = TRUE,
  p.coords = NULL,
  p.labels = rownames(p.coords),
  p.labels.pos = 3,
  p.labels.cex = 1,
  ...
)
```

Arguments

v.labels	Vertex labels anticlockwise from top (default: "Recrudescence", "Relapse", "Reinfection"). If NULL, vertices are not labelled.
v.cutoff	Number between 0.5 and 1 that separates lower vs higher probability regions. Use with caution for recrudescence and reinfection classification; see Understand posterior probabilities .

v.colours	Vertex colours anticlockwise from top.
plot.tri	Logical; draws the triangular boundary if TRUE (default).
p.coords	Matrix of 3D simplex coordinates (e.g., per-recurrence probabilities of recrudescence, relapse and reinfection), one vector of 3D coordinates per row, each row is projected onto 2D coordinates using <code>project2D()</code> and plotted as a single simplex point using <code>graphics::points()</code> . If the user provides a vector encoding a probability triplet summing to one, it is converted to a matrix with one row.
p.labels	Labels of points in p.coords (default row names of p.coords) No labels if NA.
p.labels.pos	Position of p.labels: 1 = below, 2 = left, 3 = above (default) and 4 = right. Can be a single value or a vector.
p.labels.cex	Size expansion of p.labels passed to <code>text</code> .
...	Additional parameters passed to <code>graphics::points()</code> .

Value

None

Examples

```
# Plot 2D simplex
plot_simplex(p.coords = diag(3),
             p.labels = c("(1,0,0)", "(0,1,0)", "(0,0,1)"),
             p.labels.pos = c(1,3,3))

# =====
# Given data on an enrollment episode and a recurrence,
# compute the posterior probabilities of the 3Rs and plot the deviation of the
# posterior from the prior
# =====

# Some data:
y <- list(list(m1 = c('a', 'b'), m2 = c('c', 'd')), # Enrollment episode
          list(m1 = c('a'), m2 = c('c'))) # Recurrent episode

# Some allele frequencies:
fs <- list(m1 = setNames(c(0.4, 0.6), c('a', 'b')),
           m2 = setNames(c(0.2, 0.8), c('c', 'd')))

# A vector of prior probabilities:
prior <- array(c(0.2, 0.3, 0.5), dim = c(1,3),
              dimnames = list(NULL, c("C", "L", "I")))

# Compute posterior probabilities
post <- compute_posterior(y, fs, prior, progress.bar = FALSE)

# Plot simplex with the prior and posterior
plot_simplex(p.coords = rbind(prior, post$marg),
             p.labels = c("Prior", "Posterior"),
             pch = 20)
```

```
# Add the deviation between the prior and posterior: requires obtaining 2D
# coordinates manually
xy_prior <- project2D(as.vector(prior))
xy_post <- project2D(as.vector(post$marg))
arrows(x0 = xy_prior["x"], x1 = xy_post["x"],
       y0 = xy_prior["y"], y1 = xy_post["y"], length = 0.1)
```

project2D

Project 3D probability coordinates onto 2D simplex coordinates

Description

Project three probabilities that sum to one (e.g., per-recurrence probabilities of recrudescence, relapse and reinfection) onto the coordinates of a 2D simplex centred at the origin (i.e., a triangle centred at (0,0) with unit-length sides).

Usage

```
project2D(v)
```

Arguments

`v` A numeric vector of three numbers in zero to one that sum to one.

Details

The top, left, and right vertices of the 2D simplex correspond with the first, second and third entries of `v`, respectively. Each probability is proportional to the distance from the point on the simplex to the side opposite the corresponding probability; see **Examples** below and [plot_simplex\(\)](#) for more details.

Value

A numeric vector of two coordinates that can be used to plot the probability vector `v` on the origin-centred 2D simplex.

Examples

```
probabilities_of_v1_v2_v3 <- c(0.75,0.20,0.05)
coordinates <- project2D(v = probabilities_of_v1_v2_v3)

# Plot probability vector on 2D simplex
plot_simplex(v.labels = c("v1", "v2", "v3"))
points(x = coordinates[1], y = coordinates[2], pch = 20)

# Plot the distances that represent probabilities
# get vertices, get points on edges by orthogonal projection, plot arrows
v <- apply(matrix(c(1,0,0,0,1,0,0,0,1), nrow = 3), 1, project2D)
p3 <- v[,1] + sum((coordinates - v[,1]) * (v[,2] - v[,1])) * (v[,2] - v[,1])
```

```

p1 <- v[,2] + sum((coordinates - v[,2]) * (v[,3] - v[,2])) * (v[,3] - v[,2])
p2 <- v[,3] + sum((coordinates - v[,3]) * (v[,1] - v[,3])) * (v[,1] - v[,3])
arrows(x0 = coordinates[1], y0 = coordinates[2], x1 = p1[1], y1 = p1[2], length = 0.1)
arrows(x0 = coordinates[1], y0 = coordinates[2], x1 = p2[1], y1 = p2[2], length = 0.1)
arrows(x0 = coordinates[1], y0 = coordinates[2], x1 = p3[1], y1 = p3[2], length = 0.1)

```

RG_to_igraph	<i>Converts a relationship graph (RG) encoded as a list to an igraph object</i>
--------------	---

Description

Converts an RG encoded as a list to an igraph object, which requires more memory allocation but can be plotted using [plot_RG](#).

Usage

```
RG_to_igraph(RG, MOIs)
```

Arguments

RG	List encoding an RG; see Value of enumerate_RGs when <code>igraph = FALSE</code> .
MOIs	Vector of per-episode multiplicities of infection (MOIs), i.e., numbers of per-episode genotypes / vertices; adds to the graph an attribute that is used by plot_RG to group genotypes / vertices by episode.

Value

A weighted graph whose edge weights 1 and 0.5 encode clonal and sibling relationships, respectively.

Examples

```

MOIs <- c(3,2)
set.seed(6)
RG_as_list <- sample_RG(MOIs, igraph = FALSE)
RG_as_igraph <- RG_to_igraph(RG_as_list, MOIs)

# RG encoded as a list requires less memory allocation
utils::object.size(RG_as_list)
utils::object.size(RG_as_igraph)

# RG encoded as an igraph object can be plotted using plot_RG() and
# manipulated using igraph functions
plot_RG(RG_as_igraph, margin = rep(0,4), vertex.label = NA)

# Edge weights 1 and 0.5 encode clonal and sibling relationships
igraph::E(RG_as_igraph)$weight

```

```
# Vertex attribute group encodes episode membership
igraph::V(RG_as_igraph)$group
```

sample_RG	<i>Sample a relationship graph (RG)</i>
-----------	---

Description

Uses the techniques in [enumerate_RGs](#) to sample a single RG uniformly. All clonal partitions are generated, each weighted by its number of consistent sibling partitions. A clonal partition is sampled proportional to its weight, then a consistent sibling partition is drawn uniformly. The resulting nested partition represents the RG; see [enumerate_RGs](#) for details.

Usage

```
sample_RG(MOIs, igrph = TRUE)
```

Arguments

MOIs	Vector of per-episode multiplicities of infection (MOIs), i.e., numbers of per-episode genotypes / vertices.
igrph	Logical; if TRUE (default), returns the RG as an igraph object.

Value

An RG encoded either as an igraph object (default), or as a list; see [enumerate_RGs](#) for details.

Examples

```
set.seed(1)
RG <- sample_RG(c(3, 2))
plot_RG(RG, vertex.label = NA)
```

ys_VHX_BPD	<i>Example Plasmodium vivax data</i>
------------	--------------------------------------

Description

Previously-published microsatellite data on *P. vivax* parasites extracted from study participants enrolled in the Best Primaquine Dose (BPD) and Vivax History (VHX) trials; see Taylor & Watson et al. 2019 ([doi:10.1038/s4146701913412x](https://doi.org/10.1038/s4146701913412x)) for more details of the genetic data; for more details of the VHX and BPD trials, see Chu et al. 2018a ([doi:10.1093/cid/ciy319](https://doi.org/10.1093/cid/ciy319)) and Chu et al. 2018b ([doi:10.1093/cid/ciy735](https://doi.org/10.1093/cid/ciy735)).

Usage

ys_VHX_BPD

Format

A list of 217 study participants; for each study participant, a list of one or more episodes; for each episode, a list of three or more microsatellite markers; for each marker, a vector of observed alleles (repeat lengths). For example:

BPD_103 Study participant identifier: study participant 103 in the BPD trial

BPD_103_1 Episode identifier: episode one of study participant 103 in the BPD trial

PV.3.27 Marker identifier: *P. vivax* 3.27

18 Allele identifier: 18 repeat lengths

Source

- <https://zenodo.org/records/3368828>
- https://github.com/aimeertaylor/Pv3Rs/blob/main/data-raw/ys_VHX_BPD.R

Index

* datasets

fs_VHX_BPD, [7](#)
ys_VHX_BPD, [15](#)

brewer.pal, [8](#)

compute_posterior, [2](#), [5](#), [6](#)
compute_posterior(), [8](#)

determine_MOIs, [5](#)

enumerate_RGs, [3](#), [4](#), [6](#), [14](#), [15](#)

fs_VHX_BPD, [7](#)

graphics::points(), [12](#)

par, [8](#)

plot.igraph, [10](#)

plot_data, [8](#)

plot_RG, [4](#), [10](#), [14](#)

plot_simplex, [11](#)

plot_simplex(), [13](#)

project2D, [13](#)

project2D(), [11](#), [12](#)

RG_to_igraph, [4](#), [7](#), [10](#), [14](#)

sample_RG, [15](#)

text, [12](#)

ys_VHX_BPD, [7](#), [15](#)