Package 'jellyfisher'

August 21, 2026

Title Visualize Spatiotemporal Tumor Evolution with Jellyfish Plots

Version 1.1.2

Description Generates interactive Jellyfish plots to visualize spatiotemporal tumor evolution by integrating sample and phylogenetic trees into a unified plot. This approach provides an intuitive way to analyze tumor heterogeneity and evolution over time and across anatomical locations. The Jellyfish plot visualization design was first introduced by Lahtinen, Lavikka, et al. (2023, <doi:10.1016/j.ccell.2023.04.017>).
This package also supports visualizing ClonEvol results, a tool developed by Dang, et al. (2017, <doi:10.1093/annonc/mdx517>), for analyzing clonal evolution from multi-sample sequencing data. The 'clonevol' package is not available on CRAN but can be installed from its GitHub repository (<https://github.com/hdng/clonevol>).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

URL https://github.com/HautaniemiLab/jellyfisher, https://hautaniemilab.github.io/jellyfisher/

BugReports https://github.com/HautaniemiLab/jellyfisher/issues

Imports htmlwidgets, dplyr, stringr

Suggests knitr, rmarkdown, clonevol, testthat (>= 3.0.0)

Depends R (>= 4.1.0)

VignetteBuilder knitr

biocViews Visualization, Phylogenetics, Software, Spatial

Config/testthat/edition 3

NeedsCompilation no

Author Kari Lavikka [cph, aut, cre] (ORCID: <https://orcid.org/0000-0002-4163-4945>)

Maintainer Kari Lavikka <kari@karilavikka.fi>

Repository CRAN

Date/Publication 2026-08-21 08:50:02 UTC

Contents

add_inferred_sample 2
extract_tables_from_clonevol 3
jellyfisher 4
jellyfisher-shiny 7
jellyfisher_example_tables 8
select_patients 9
set_parents 10
set_ranks 11
validate_tables 12

Index **13**

`add_inferred_sample` *Insert an a hypothetical inferred sample into the sample tree and set it as the parent of specified samples.*

Description

Insert an a hypothetical inferred sample into the sample tree and set it as the parent of specified samples.

Usage

```
add_inferred_sample(
  tables,
  name,
  rank = NULL,
  samples = character(),
  parent_sample = NULL,
  display_name = "Inferred"
)
```

Arguments

`tables`	A list of tables (samples, phylogeny, compositions, ranks)
`name`	The name of the inferred sample to add
`rank`	The rank of the inferred sample
`samples`	A character vector of samples to set the inferred sample as their parent
`parent_sample`	The parent sample of the inferred sample. If NULL, the inferred root is used.
`display_name`	The display name of the inferred sample (default: "Inferred")

Value

A list of tables with a newly added inferred sample and updated parent relationships

Examples

```
jellyfisher_example_tables |>
  select_patients("EOC153") |>
  add_inferred_sample("EOC153_Inf",
                      2,
                      c("EOC153_iPer1_DNA4",
                        "EOC153_iOme1_DNA4",
                        "EOC153_iOvaR1_DNA1")) |>
  jellyfisher()
```

---

extract_tables_from_clonevol

Extract Data for Jellyfisher from ClonEvol Results

Description

ClonEvol infers clonal evolution from multi-sample cancer sequencing data and generates phylogenetic models of tumor evolution. This function extracts data frames from a ClonEvol object that can be used to create a Jellyfish plot.

Usage

```
extract_tables_from_clonevol(y, model = 1)
```

Arguments

y	A ClonEvol object
model	The model to extract. Defaults to 1

Details

Note: ClonEvol reports clonal prevalences as confidence intervals. This function extracts the mean values and uses them as the prevalence values.

For more details about ClonEvol, including the installation instructions, visit its GitHub repository or read the publication by Dang et al. (2017, doi:10.1093/annonc/mdx517).

Value

A named list with three data frames: samples, phylogeny, and compositions

Examples

```
if (requireNamespace("clonevol", quietly = TRUE)) {
  # Run ClonEvol with its example data
  # (refer to ClonEvol documentation for details)
  data <- clonevol::aml1
  y <- clonevol::infer.clonal.models(
    variants = data$variants,
    cluster.col.name = "cluster",
    vaf.col.names = data$params$vaf.col.names,
    subclonal.test = "bootstrap",
    subclonal.test.model = "non-parametric",
    num.boots = 1000,
    founding.cluster = 1,
    cluster.center = "mean",
    ignore.clusters = NULL,
    min.cluster.vaf = 0.01,
    sum.p = 0.05,
    alpha = 0.05
  )
  # Make branch lengths available
  y <- clonevol::convert.consensus.tree.clone.to.branch(y)

  # Extract data and plot the results
  extract_tables_from_clonevol(y, model = 1) |>
    jellyfisher()
} else {
  message(
   "Please install the clonevol package from GitHub: devtools::install_github('hdng/clonevol')"
  )
}
```

`jellyfisher` *Creates a Jellyfish plot*

Description

Creates a Jellyfish plot from samples, a phylogeny, and subclonal compositions.

Usage

```
jellyfisher(
  tables,
  options = list(),
  controls = "closed",
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`tables` A named list of data frames. The list must contain the following elements:

`samples` A data frame with sample data. The expected columns are:

sample specifies the unique identifier for each sample. (string)

displayName allows for specifying a custom name for each sample. If the column is omitted, the `sample` column is used as the display name. (string, optional)

rank specifies the position of each sample in the Jellyfish plot. For example, different stages of a disease can be ranked in chronological order: diagnosis (1), interval (2), and relapse (3). The zeroth rank is reserved for the root of the sample tree. Ranks can be any integer, and unused ranks are automatically excluded from the plot. If the `rank` column is (integer)

parent identifies the parent sample for each entry. Samples without a specified parent are treated as children of an imaginary root sample. (string)

`phylogeny` A data frame with phylogeny data. The expected columns are:

subclone specifies subclone IDs, which can be any string. (string)

parent designates the parent subclone. The subclone without a parent is considered the root of the phylogeny. (string)

color specifies the color for the subclone. If the column is omitted, colors will be generated automatically. (string, optional)

branchLength specifies the length of the branch leading to the subclone. The length may be based on, for example, the number of unique mutations in the subclone. The branch length is shown in the Jellyfish plot's legend as a bar chart. It is also used when generating a phylogeny-aware color scheme. (number)

`compositions` A data frame with subclonal compositions. The expected columns are:

sample specifies the sample ID. (string)

subclone specifies the subclone ID. (string)

clonalPrevalence specifies the clonal prevalence of the subclone in the sample. The clonal prevalence is the proportion of the subclone in the sample. The clonal prevalences in a sample must sum to 1. (number)

`ranks` An optional data frame with ranks. The expected columns are:

rank specifies the rank number. The zeroth rank is reserved for the inferred root of the sample tree. However, you are free to define a title for it. (integer)

title specifies the title for the rank. (string)

`options` A named list of options to configure the plot. Available options:

crossingWeight Weight for tentacle bundles between two pairs of samples crossing each other. Defaults to `10`.

pathLengthWeight Weight for the total length of the paths (tentacle bundles) connecting samples. Defaults to `2`.

orderMismatchWeight Weight for the mismatch in the order of samples. The order is based on the "phylogenetic center of mass" computed from the subclonal compositions. Defaults to `2`.

bundleMismatchWeight Weight for the mismatch in the placement of bundles. The "optimal" placement is based on the subclonal compositions, but such placement may produce excessively long tentacle bundles. Defaults to `3`.

divergenceWeight Weight for the sum of divergences between adjacent samples. Defaults to `4`.

bellTipShape The shape of the bell tip. 0 is a sharp tip, 1 is a blunt tip. Defaults to `0.1`.

bellTipSpread How much to spread nested bell tips. 0 is no spread, 1 is full spread. Defaults to `0.5`.

bellStrokeWidth The width of strokes in the bell. Defaults to `1`.

bellStrokeDarkening How much the stroke color of the bells is darkened. Defaults to `0.6`.

bellPlateauPos Where the bell has fully appeared and the plateau starts. Defaults to `0.7`.

sampleHeight Height of real sample nodes Defaults to `110`.

sampleWidth Width of sample nodes Defaults to `90`.

inferredSampleHeight Height of inferred sample nodes Defaults to `120`.

gapHeight Height of gaps between samples. Gaps are routes for tentacle bundles. Defaults to `60`.

sampleSpacing Vertical space between samples Defaults to `60`.

columnSpacing Horizontal space between columns Defaults to `90`.

tentacleWidth Width of tentacles in pixels Defaults to `2`.

tentacleSpacing Space between tentacles in a bundle, in pixels Defaults to `5`.

inOutCPDistance Relative distance of tentacle control points from the edge of the sample node Defaults to `0.3`.

bundleCPDistance Relative distance of tentacle bundle's control points. The higher the value, the longer the individual tentacles stay together before diverging. Defaults to `0.6`.

sampleFontSize Font size for sample labels Defaults to `12`.

showLegend Whether to show the legend Defaults to `TRUE`.

phylogenyColorScheme Whether to use a color scheme based on phylogeny Defaults to `TRUE`.

phylogenyHueOffset Offset for the hue of the phylogeny color scheme. If the automatically generated hues are not to your liking, you can adjust the hue offset to get a different color scheme. Defaults to `0`.

phylogenyLightnessStart Lightness at the root of the phylogeny Defaults to `0.94`.

phylogenyLightnessEnd Lightness at the most divergent leaf of the phylogeny Defaults to `0.68`.

phylogenyChromaStart Chroma at the root of the phylogeny Defaults to `0.025`.

phylogenyChromaEnd Chroma at the most divergent leaf of the phylogeny Defaults to `0.21`.

sampleTakenGuide Type of the "sample taken" guide. `"none"` for no guides, `"line"` for a faint dashed line in all samples, `"text"` same as line, but with a text label in one of the samples. `"text-all"` same as text, but with a text label in all samples. Defaults to `"text"`.

showRankTitles Whether to show rank titles above the samples (if provided). Defaults to `TRUE`.

normalsAtPhylogenyRoot Whether the root of the phylogenetic tree contains normal cells. If true, no tentacles will be drawn for the root clone and its color will be white if phylogenyColorScheme is used. Defaults to `FALSE`.

`controls`	An optional parameter to set the initial state of the controls. Can be "open", "closed", or "hidden".
`width`	The width of the widget
`height`	The height of the widget
`elementId`	An optional element ID for the widget

Details

The format of the data frames is described with examples in Jellyfish documentation: https://github.com/HautaniemiLab/jellyf
ov-file#input-data

Value

A Jellyfish plot HTML widget

Examples

```
# Plot the bundled example data
jellyfisher(jellyfisher_example_tables,
            options = list(
              sampleHeight = 70,
              sampleTakenGuide = "none",
              showLegend = FALSE
            ))
```

`jellyfisher-shiny` *Shiny bindings for jellyfisher*

Description

Output and render functions for using jellyfisher within Shiny applications and interactive Rmd documents.

Usage

```
jellyfisherOutput(outputId, width = "100%", height = "400px")

renderJellyfisher(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a jellyfisher
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

jellyfisherOutput() A shiny.tag object that can be included in a Shiny UI definition.

renderJellyfisher() A shiny.render.function that returns an htmlwidget object for use in a Shiny server function.

---

jellyfisher_example_tables

Jellyfisher example data

Description

Example data based on the following publication: Lahtinen A, Lavikka K, Virtanen A, et al. Evolutionary states and trajectories characterized by distinct pathways stratify patients with ovarian high grade serous carcinoma. *Cancer Cell*. 2023;41(6):1103-1117.e12. doi:10.1016/j.ccell.2023.04.017

Format

jellyfisher_example_tables:

A named list of data frames containing the following tables:

samples A data frame with sample data. Columns are:

- **sample** specifies the unique identifier for each sample. (string)
- **displayName** allows for specifying a custom name for each sample. If the column is omitted, the sample column is used as the display name. (string, optional)
- **rank** specifies the position of each sample in the Jellyfish plot. For example, different stages of a disease can be ranked in chronological order: diagnosis (1), interval (2), and relapse (3). The zeroth rank is reserved for the root of the sample tree. Ranks can be any integer, and unused ranks are automatically excluded from the plot. If the rank column is (integer)
- **parent** identifies the parent sample for each entry. Samples without a specified parent are treated as children of an imaginary root sample. (string)

phylogeny A data frame with phylogeny data. Columns are:

- **subclone** specifies subclone IDs, which can be any string. (string)
- **parent** designates the parent subclone. The subclone without a parent is considered the root of the phylogeny. (string)

color specifies the color for the subclone. If the column is omitted, colors will be generated automatically. (string, optional)

branchLength specifies the length of the branch leading to the subclone. The length may be based on, for example, the number of unique mutations in the subclone. The branch length is shown in the Jellyfish plot's legend as a bar chart. It is also used when generating a phylogeny-aware color scheme. (number)

`compositions` A data frame with subclonal compositions. Columns are:

sample specifies the sample ID. (string)

subclone specifies the subclone ID. (string)

clonalPrevalence specifies the clonal prevalence of the subclone in the sample. The clonal prevalence is the proportion of the subclone in the sample. The clonal prevalences in a sample must sum to 1. (number)

`ranks` An optional data frame with ranks. Columns are:

rank specifies the rank number. The zeroth rank is reserved for the inferred root of the sample tree. However, you are free to define a title for it. (integer)

title specifies the title for the rank. (string)

Source

https://github.com/HautaniemiLab/jellyfish/tree/main/data

`select_patients` *Filter Jellyfish tables by patient*

Description

Given a list of tables, filter them by patient.

Usage

```
select_patients(tables, patient)
```

Arguments

`tables`	A list of tables (samples, phylogeny, compositions, ranks)
`patient`	The patient or patients to filter by

Value

A list of tables filtered by patient

Examples

```
jellyfisher_example_tables |>
  select_patients("EOC809")
```

`set_parents` *Set parents for samples*

Description

Given a list of jellyfish input tables and a named list of parents for each sample, set the parent for each sample.

Usage

```
set_parents(tables, parents, unset_missing = FALSE)
```

Arguments

`tables`	A list of tables (samples, phylogeny, compositions, ranks)
`parents`	A named list of parents for each sample. Keys are the samples and values are their new parents
`unset_missing`	If TRUE, unset the parent for samples that are not in the parent list

Details

By default, all samples that have no explicit parent are children of the *inferred root* sample. You can customize the parent-child relationships by modifying the `parent` column in the `samples` data frame before plotting.

You can also modify the relationships using the `set_parents` function.

For example, if you have three samples, A, B, and C, they will have the following relationships by default:

```
   Root
  / | \
 A  B  C
```

With the explicit parents, you can customize the relationships:

```
tables |>
  set_parents(list(
    # The parent of C is B
    C = "B"
  ) |>
  jellyfisher()
```

```
   Root
  /  \
 A    B
       \
        C
```

Value

A list of tables with parents set for each sample

Examples

```
jellyfisher_example_tables |>
  select_patients("EOC809") |>
  set_parents(list("EOC809_r1Bow1_DNA1" = "EOC809_p2Per1_cO_DNA2")) |>
  jellyfisher()
```

---

set_ranks *Set ranks for samples*

Description

Given a list of jellyfish input tables and a named list of ranks for each sample, set the rank for each sample.

Usage

```
set_ranks(tables, ranks, default = 1)
```

Arguments

tables	A list of tables (samples, phylogeny, compositions, ranks)
ranks	A named list of ranks for each sample
default	The default rank to use when a sample is not in the rank list (default: 1)

Value

A list of tables with ranks set for each sample

Examples

```
jellyfisher_example_tables |>
  select_patients("EOC809") |>
  set_ranks(list("EOC809_r1Bow1_DNA1" = 2, "EOC809_p2Per1_cO_DNA2" = 2),
    default = 1
  ) |>
  jellyfisher()
```

`validate_tables` *Validate Jellyfish tables*

Description

Superficially validate that the tables are in the correct format.

Usage

```
validate_tables(tables)
```

Arguments

`tables`	A list of tables (samples, phylogeny, compositions, ranks)

Index

∗ **datasets**
jellyfisher_example_tables, 8

add_inferred_sample, 2

extract_tables_from_clonevol, 3

jellyfisher, 4
jellyfisher-shiny, 7
jellyfisher_example_tables, 8
jellyfisherOutput *(*jellyfisher-shiny*)*, 7

renderJellyfisher *(*jellyfisher-shiny*)*, 7

select_patients, 9
set_parents, 10
set_ranks, 11

validate_tables, 12