

Package ‘timescape’

February 2, 2026

Title Patient Clonal Timescapes

Version 1.35.0

Description TimeScape is an automated tool for navigating temporal clonal evolution data. The key attributes of this implementation involve the enumeration of clones, their evolutionary relationships and their shifting dynamics over time. TimeScape requires two inputs: (i) the clonal phylogeny and (ii) the clonal prevalences. Optionally, TimeScape accepts a data table of targeted mutations observed in each clone and their allele prevalences over time. The output is the TimeScape plot showing clonal prevalence vertically, time horizontally, and the plot height optionally encoding tumour volume during tumour-shrinking events. At each sampling time point (denoted by a faint white line), the height of each clone accurately reflects its proportionate prevalence. These prevalences form the anchors for bezier curves that visually represent the dynamic transitions between time points.

Depends R (>= 3.3)

Imports htmlwidgets (>= 0.5), jsonlite (>= 0.9.19), stringr (>= 1.0.0), dplyr (>= 0.4.3), gtools (>= 3.5.0)

biocViews Visualization, BiomedicalInformatics

License GPL-3

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/timescape>

git_branch devel

git_last_commit a04609a

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-02-01

Author Maia Smith [aut, cre]

Maintainer Maia Smith <maiaannesmith@gmail.com>

Contents

timescape	2
timescapeOutput	4

Index	9
--------------	----------

timescape	<i>TimeScape</i>
-----------	------------------

Description

timescape is a tool for visualizing temporal clonal evolution data.

Usage

```
timescape(clonal_prev, tree_edges, mutations = "NA", clone_colours = "NA",
  xaxis_title = "Time Point", yaxis_title = "Clonal Prevalence",
  phylogeny_title = "Clonal Phylogeny", alpha = 50,
  genotype_position = "stack", perturbations = "NA", sort = FALSE,
  show_warnings = TRUE, width = 900, height = NULL)
```

Arguments

clonal_prev	data.frame Clonal prevalence. Required columns are: timepoint: character() time point. Time points will be alphanumerically sorted in the view. clone_id: character() clone id. clonal_prev: numeric() clonal prevalence.
tree_edges	data.frame Tree edges of a rooted tree. Required columns are: source: character() source node id. target: character() target node id.
mutations	data.frame (Optional) Mutations occurring at each clone. Required columns are: chrom: character() chromosome number. coord: numeric() coordinate of mutation on chromosome. clone_id: character() clone id. timepoint: character() time point. VAF: numeric() variant allele frequency of the mutation in the corresponding timepoint. Any additional field will be shown in the mutation table.
clone_colours	data.frame Clone ids and their corresponding colours. Required columns are: clone_id: character() clone id. colour: character() the corresponding Hex colour for each clone id.

<code>xaxis_title</code>	<code>character()</code> (Optional) x-axis title. Default is "Time Point".
<code>yaxis_title</code>	<code>character()</code> (Optional) y-axis title. Default is "Clonal Prevalence".
<code>phylogeny_title</code>	<code>character()</code> (Optional) Legend phylogeny title. Default is "Clonal Phylogeny".
<code>alpha</code>	<code>numeric()</code> (Optional) Alpha value for clonal sweeps, range [0, 100].
<code>genotype_position</code>	<code>character()</code> (Optional) How to position the genotypes from ["centre", "stack", "space"]. <ol style="list-style-type: none"> 1. centre: genotypes are centred with respect to their ancestors. 2. stack: genotypes are stacked such that nongenotype is split at any time point. 3. space: genotypes are stacked but with a bit of spacing at the bottom.
<code>perturbations</code>	<code>data.frame</code> (Optional) Any perturbations that occurred between two time points. Required columns are: pert_name: <code>character()</code> the perturbation name. prev_tp: <code>character()</code> the time point (as labelled in clonal prevalence data) BEFORE perturbation.
<code>sort</code>	<code>logical()</code> (Optional) Whether (TRUE) or not (FALSE) to vertically sort the genotypes by their emergence values (descending). Default is FALSE. Note that genotype sorting will always retain the phylogenetic hierarchy, and this parameter will only affect the ordering of siblings.
<code>show_warnings</code>	<code>logical()</code> (Optional) Whether or not to show any warnings. Default is TRUE.
<code>width</code>	<code>numeric()</code> (Optional) Width of the plot. Minimum width is 450.
<code>height</code>	<code>numeric()</code> (Optional) Height of the plot. Minimum height with and without mutations is 500 and 260, respectively.

Details

Interactive components:

1. Mouseover any clone to view its (i) clone ID and (ii) clonal prevalence at each time point.
2. Click the view switch button to switch from the traditional timescape view to the clonal trajectory view, where each clone changes prevalence on its own track.
3. Click the download buttons to download a PNG or SVG of the view.

Value

None

Examples

```
# EXAMPLE 1 - Acute myeloid leukemia patient, Ding et al., 2012

# genotype tree edges
tree_edges <- read.csv(system.file("extdata", "AML_tree_edges.csv",
  package = "timescape"))
```

```

# clonal prevalences
clonal_prev <- read.csv(system.file("extdata", "AML_clonal_prev.csv",
  package = "timescape"))

# targeted mutations
mutations <- read.csv(system.file("extdata", "AML_mutations.csv",
  package = "timescape"))

# perturbations
perturbations <- data.frame( pert_name = c("Chemotherapy"),
  prev_tp = c("Diagnosis"))

# run timescape
timescape(clonal_prev = clonal_prev, tree_edges = tree_edges,
  perturbations = perturbations, mutations = mutations)

# EXAMPLE 2 - Patient 7, McPherson and Roth et al., 2016

# genotype tree edges
tree_edges <- read.csv(system.file("extdata", "px7_tree_edges.csv",
  package = "timescape"))

# clonal prevalences
clonal_prev <- read.csv(system.file("extdata", "px7_clonal_prev.csv",
  package = "timescape"))

# clone colours
clone_colours <- data.frame(clone_id = c("A", "B", "C", "D", "E"),
  colour = c("d0ced0", "2CD0AB", "FFD94B",
    "FD8EE5", "F8766D"))

# run timescape
timescape(clonal_prev = clonal_prev, tree_edges = tree_edges,
  clone_colours = clone_colours, height=260, alpha=15)

```

timescapeOutput

Widget output function for use in Shiny

Description

Widget output function for use in Shiny

Widget render function for use in Shiny

Function to process the user data

Function to check minimum dimensions

Function to check required inputs are present

check alpha value input is correct

check clonal_prev parameter data

check tree_edges parameter data
check genotype_position parameter
check clone_colours parameter
check perturbations parameter
get mutation data
function to replace spaces with underscores in all data frames & keep maps of original names to space-replaced names

Usage

```
timescapeOutput(outputId, width = "100%", height = "400px")  
  
renderTimescape(expr, env = parent.frame(), quoted = FALSE)  
  
processUserData(clonal_prev, tree_edges, mutations, clone_colours, xaxis_title,  
  yaxis_title, phylogeny_title, alpha, genotype_position, perturbations, sort,  
  show_warnings, width, height)  
  
checkMinDims(mutations, height, width)  
  
checkRequiredInputs(clonal_prev, tree_edges)  
  
checkAlpha(alpha)  
  
checkClonalPrev(clonal_prev)  
  
checkTreeEdges(tree_edges)  
  
checkGtypePositioning(genotype_position)  
  
checkCloneColours(clone_colours)  
  
checkPerts(perturbations)  
  
getMutationsData(mutations, tree_edges, clonal_prev)  
  
replaceSpaces(clonal_prev, tree_edges, clone_colours, mutation_info, mutations,  
  mutation_prevalences)
```

Arguments

outputId	– id of output
width	– width of output
height	– height of output
expr	– expression for Shiny
env	– environment for Shiny

quoted	– default is FALSE
clonal_prev	– data frame of Clonal prevalence. Note: timepoints will be alphanumerically sorted in the view. Format: columns are (1) character() "timepoint" - time point (2) character() "clone_id" - clone id (3) numeric() "clonal_prev" - clonal prevalence.
tree_edges	– data frame of Tree edges of a rooted tree. Format: columns are (1) character() "source" - source node id (2) character() "target" - target node id.
mutations	– data frame (Optional) of Mutations occurring at each clone. Any additional field will be shown in the mutation table. Format: columns are (1) character() "chrom" - chromosome number (2) numeric() "coord" - coordinate of mutation on chromosome (3) character() "clone_id" - clone id (4) character() "timepoint" - time point (5) numeric() "VAF" - variant allele frequency of the mutation in the corresponding timepoint.
clone_colours	– data frame (Optional) of Clone ids and their corresponding colours Format: columns are (1) character() "clone_id" - the clone ids (2) character() "colour" - the corresponding Hex colour for each clone id.
xaxis_title	– String (Optional) of x-axis title. Default is "Time Point".
yaxis_title	– String (Optional) of y-axis title. Default is "Clonal Prevalence".
phylogeny_title	– String (Optional) of Legend phylogeny title. Default is "Clonal Phylogeny".
alpha	– Number (Optional) of Alpha value for sweeps, range [0, 100].
genotype_position	– String (Optional) of How to position the genotypes from ["centre", "stack", "space"] "centre" – genotypes are centred with respect to their ancestors "stack" – genotypes are stacked such that no genotype is split at any time point "space" – genotypes are stacked but with a bit of spacing at the bottom
perturbations	– data frame (Optional) of any perturbations that occurred between two time points. Format: columns are (1) character() "pert_name" - the perturbation name (2) character() "prev_tp" - the time point (as labelled in clonal prevalence data) BEFORE perturbation.
sort	– Boolean (Optional) of whether (TRUE) or not (FALSE) to vertically sort the genotypes by their emergence values (descending). Default is FALSE. Note that genotype sorting will always retain the phylogenetic hierarchy, and this parameter will only affect the ordering of siblings.
show_warnings	– Boolean (Optional) of Whether or not to show any warnings. Default is TRUE.
mutation_info	– processed mutation_info
mutation_prevalences	– mutation_prevalences data from user
width	– Number (Optional) of width of the plot. Minimum width is 450.
height	– Number (Optional) of height of the plot. Minimum height with and without mutations is 500 and 260, respectively.
mutations	– mutations provided by user
height	– height provided by user

width	– width provided by user
clonal_prev	– clonal_prev provided by user
tree_edges	– tree_edges provided by user
alpha	– alpha provided by user
clonal_prev	– clonal prevalence provided by user
tree_edges	– tree edges provided by user
genotype_position	– genotype_position provided by user
clone_colours	– clone_colours provided by user
perturbations	– perturbations provided by user
mutations	– mutations data from user
tree_edges	– tree edges data from user
clonal_prev	– clonal prevalence data from user
clonal_prev	– clonal_prev data from user
tree_edges	– tree edges data from user
clone_colours	– clone_colours data from user
mutations	– mutations data from user

Value

None

None

Returns the ready list of user input data for htmlwidget

None

None

None

Clonal prevalence data after checkint it for column names and content types

Tree edges data after checkint it for column names and content types

None

None

Perturbations after checking them for content types and column names

List of mutation information and mutation prevalences

List of data frames with spaces replaced

```
timescapeOutput(1, '100%', '300px')
timescapeOutput(1, '80%', '300px')
checkMinDims(data.frame(chr = c("11"), coord = c(104043), VAF = c(0.1)), "700px", "700px")
checkRequiredInputs(data.frame(timepoint = c(rep("Diagnosis", 6), rep("Relapse", 1)), clone_id = c("1", "2", "3", "4", "5", "6", "7")), "700px", "700px")
data.frame(source = c("1", "1", "2", "2", "5", "6"), target=c("2", "5", "3", "4", "6", "7"))
checkRequiredInputs(data.frame(timepoint = c(rep("Diagnosis", 6), rep("Relapse", 1)), clone_id = c("1", "2", "3", "4", "5", "6", "7")), "700px", "700px")
data.frame(source = c("1", "1", "2", "2", "5", "6"), target=c("2", "5", "3", "4", "6", "7"))
checkAlpha(4)
checkAlpha(100)
checkClonalPrev(data.frame(timepoint=c(1), clone_id=c(2), clonal_prev=c(0.1)))
checkTreeEdges(data.frame(source = c("1", "1", "2", "2", "5", "6"), target=c("2", "5", "3", "4", "6", "7")))
checkGtypePositioning("centre")
checkCloneColours(data.frame(clone_id = c("1", "2", "3", "4"), colour = c("#beaed4", "#fdc086", "#beaed4", "#beaed4")))
checkPerts(data.frame(pert_name = c("New Drug"), prev_tp = c("Diagnosis")))
getMutationsData(data.frame(chrom = c("11"), coord = c(104043), VAF = c(0.1), clone_id=c(1), timepoint=c("Relapse"))
data.frame(source = c("1", "1", "2", "2", "5", "6"), target=c("2", "5", "3", "4", "6", "7")),
data.frame(timepoint = c(rep("Diagnosis", 6), rep("Relapse", 1)), clone_id = c("1", "2", "3", "4", "5", "6", "7"), clon
replaceSpaces(mutations = data.frame(chrom = c("11"), coord = c(104043), VAF = c(0.1), clone_id=c(1), timepoint=c("Relapse"))
tree_edges = data.frame(source = c("1", "1", "2", "2", "5", "6"), target=c("2", "5", "3", "4", "6", "7")),
clonal_prev = data.frame(timepoint = c(rep("Diagnosis", 6), rep("Relapse", 1)), clone_id = c("1", "2", "3", "4", "5", "6", "7")),
mutation_prevalences = list("X:6154028" = data.frame(timepoint = c("Diagnosis"), VAF = c(0.5557))), mutation_info = data.frame(
clone_colours = data.frame(clone_id = c("1", "2", "3", "4"), colour = c("#beaed4", "#fdc086", "#beaed4", "#beaed4"))
```


Index

`checkAlpha (timescapeOutput)`, [4](#)
`checkClonalPrev (timescapeOutput)`, [4](#)
`checkCloneColours (timescapeOutput)`, [4](#)
`checkGtypePositioning`
 `(timescapeOutput)`, [4](#)
`checkMinDims (timescapeOutput)`, [4](#)
`checkPerts (timescapeOutput)`, [4](#)
`checkRequiredInputs (timescapeOutput)`, [4](#)
`checkTreeEdges (timescapeOutput)`, [4](#)

`getMutationsData (timescapeOutput)`, [4](#)

`processUserData (timescapeOutput)`, [4](#)

`renderTimescape (timescapeOutput)`, [4](#)
`replaceSpaces (timescapeOutput)`, [4](#)

`timescape`, [2](#)
`timescapeOutput`, [4](#)