Package 'cellscape'

May 8, 2026

Title Explores single cell copy number profiles in the context of a single cell tree

Version 1.37.0

Description CellScape facilitates interactive browsing of single cell clonal evolution datasets. The tool requires two main inputs: (i) the genomic content of each single cell in the form of either copy number segments or targeted mutation values, and (ii) a single cell phylogeny. Phylogenetic formats can vary from dendrogram-like phylogenies with leaf nodes to evolutionary model-derived phylogenies with observed or latent internal nodes. The CellScape phylogeny is flexibly input as a table of source-target edges to support arbitrary representations, where each node may or may not have associated genomic data. The output of CellScape is an interactive interface displaying a single cell phylogeny and a cell-by-locus genomic heatmap representing the mutation status in each cell for each locus.

License GPL-3

Depends R (>= 3.3)

Imports dplyr (>= 0.4.3), gtools (>= 3.5.0), htmlwidgets (>= 0.5), jsonlite (>= 0.9.19), reshape2 (>= 1.4.1), stringr (>= 1.0.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr

biocViews Visualization

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

git_url https://git.bioconductor.org/packages/cellscape

git_branch devel

git_last_commit 99b5d1a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Author Shixiang Wang [aut, cre] (ORCID: <https://orcid.org/0000-0001-9855-7357>), Maia Smith [aut]

Maintainer Shixiang Wang <w_shixiang@163.com>

Contents

`cellscape` *CellScape*

Description

`cellscape` explores single cell copy number profiles in the context of a single cell phylogeny.

Usage

```
cellscape(
  cnv_data = NULL,
  mut_data = NULL,
  mut_data_matrix = NULL,
  mut_order = NULL,
  tree_edges,
  gtype_tree_edges = NULL,
  sc_annot = NULL,
  clone_colours = "NA",
  timepoint_title = "Timepoint",
  clone_title = "Clone",
  xaxis_title = "Time Point",
  yaxis_title = "Clonal Prevalence",
  phylogeny_title = "Clonal Phylogeny",
  value_type = NULL,
  node_type = "Cell",
  display_node_ids = FALSE,
  prop_of_clone_threshold = 0.2,
  vaf_threshold = 0.05,
  show_warnings = TRUE,
  width = 900,
  height = 800
)
```

Arguments

`cnv_data` `data.frame` (Required if not providing mut_data nor mut_data_matrix) Single cell copy number segments data. Note that every single cell id must be present in the tree_edges data frame. Required columns are:

single_cell_id: `character()` single cell id.

chr: `character()` chromosome number.

start: `numeric()` start position.

end: `numeric()` end position.

copy_number: `numeric()` copy number state.

`mut_data` `data.frame` (Required if not providing cnv_data nor mut_data_matrix) Single cell targeted mutation data frame. Note that every single cell id must be present in the tree_edges data frame. Required columns are:

single_cell_id: `character()` single cell id.

chr: `character()` chromosome number.

coord: `numeric()` genomic coordinate.

VAF: `numeric()` variant allele frequency [0, 1].

`mut_data_matrix`
`matrix` (Required if not providing cnv_data nor mut_data) Single cell targeted mutation matrix. Rows are single cell IDs, columns are mutations. Rows and columns must be named, column names in the format "<chromosome>:<coordinate>". Note that the order of these rows and columns will not be preserved, unless mutation order is the same as that specified in the mut_order parameter. Also note that every single cell id must be present in the tree_edges data frame.

`mut_order` `vector` (Optional) Mutation order for targeted mutation heatmap (each mutation should consist of a string in the form "chrom:coord"). Default will use a clustering function to determine mutation order.

`tree_edges` `data.frame` Edges for the single cell phylogenetic tree. Required columns are:

source: `character()` edge source (single cell id).

target: `character()` edge target (single cell id).

Optional columns are:

dist: `numeric()` edge distance.

`gtype_tree_edges`
`data.frame` (Required for TimeScape) Genotype tree edges of a rooted tree. Required columns are:

source: `character()` source node id.

target: `character()` target node id.

`sc_annot` `data.frame` (Required for TimeScape) Annotations (genotype and sample id) for each single cell. Required columns are:

single_cell_id: `character()` single cell id.

genotype: `character()` genotype assignment.

Optional columns are:

timepoint: `character()` id of the sampled time point. Note: time points will be ordered alphabetically.

`clone_colours` `data.frame` (Optional) Clone ids and their corresponding colours (in hex format). Required columns are:

clone_id: `character()` clone id.

colour: `character()` the corresponding Hex colour for each clone id.

`timepoint_title`
`character()` (Optional) Legend title for timepoint groups. Default is "Timepoint".

`clone_title` `character()` (Optional) Legend title for clones. Default is "Clone".

`xaxis_title` `character()` (Optional) For TimeScape - x-axis title. Default is "Time Point".

`yaxis_title` `character()` (Optional) For TimeScape - y-axis title. Default is "Clonal Prevalence".

`phylogeny_title`
: `character()` (Optional) For TimeScape - legend phylogeny title. Default is "Clonal Phylogeny".

`value_type`
: `character()` (Optional) The type of value plotted in heatmap - will affect legend and heatmap tooltips. Default is "VAF" for mutation data, and "CNV" for copy number data.

`node_type`
: `character()` (Optional) The type of node plotted in single cell phylogeny - will affect phylogeny tooltips. Default is "Cell".

`display_node_ids`
: `logical()` (Optional) Whether or not to display the single cell ID within the tree nodes. Default is FALSE.

`prop_of_clone_threshold`
: `numeric()` (Optional) Used for the ordering of targeted mutations. The minimum proportion of a clone to have a mutation in order to consider the mutation as present within that clone. Default is 0.2.

`vaf_threshold`
: `numeric()` (Optional) Used for the ordering of targeted mutations. The minimum variant allele frequency for a mutation to be considered as present within a single cell. Default is 0.05.

`show_warnings`
: `logical()` (Optional) Whether or not to show any warnings. Default is TRUE.

`width`
: `numeric()` (Optional) Width of the plot.

`height`
: `numeric()` (Optional) Height of the plot.

Details

Interactive components:

1. Mouseover any single cell in the phylogeny to view its corresponding genomic profile in the heatmap, and vice versa.
2. Mouseover any part of the heatmap to view the CNV or VAF value for that copy number segment or mutation site, respectively.
3. Mouseover any branch of the phylogeny to view downstream single cells, both in the phylogeny and heatmap.
4. Mouseover any clone to view its corresponding single cells in the phylogeny and heatmap.
5. Click any node in the phylogeny to flip the order of its descendant branches.
6. Use the selection tool in the tool bar to select single cell genomic profiles and view their corresponding single cells in the phylogeny.
7. Use the tree trimming tool in the tool bar to remove any branch of the phylogeny by clicking it.
8. Use the switch view tool in the tool bar to change the phylogeny view from force-directed to unidirectional, and vice versa.
9. Use the re-root phylogeny tool to root the phylogeny at a clicked node.
10. Use the flip branch tool to vertically rotate any branch by clicking its root node.
11. If present, use the scale tree/graph tool in the tool bar to scale the phylogeny by the provided edge distances.
12. If time-series information is present such that the TimeScape is displayed below the CellScape, clones and time points are interactively linked in both views on mouseover.
13. Click the download buttons to download a PNG or SVG of the view.

Note:

See TimeScape repo (https://bitbucket.org/MO_BCCRC/timescape) for more information about TimeScape.

Examples

```
library("cellscape")


# EXAMPLE 1 - TARGETED MUTATION DATA

# single cell tree edges
tree_edges <- read.csv(system.file("extdata", "targeted_tree_edges.csv",
  package = "cellscape"
))

# targeted mutations
targeted_data <- read.csv(system.file("extdata", "targeted_muts.csv",
  package = "cellscape"
))

# genotype tree edges
gtype_tree_edges <- data.frame("source" = c(
  "Ancestral", "Ancestral", "B",
  "C", "D"
), "target" = c("A", "B", "C", "D", "E"))

# annotations
sc_annot <- read.csv(system.file("extdata", "targeted_annots.csv",
  package = "cellscape"
))

# mutation order
mut_order <- scan(system.file("extdata", "targeted_mut_order.txt",
  package = "cellscape"
), what = character())

# run cellscape
cellscape(
  mut_data = targeted_data, tree_edges = tree_edges, sc_annot =
    sc_annot, gtype_tree_edges = gtype_tree_edges, mut_order = mut_order
)


# EXAMPLE 2 - COPY NUMBER DATA

# single cell tree edges
tree_edges <- read.csv(system.file("extdata", "cnv_tree_edges.csv",
  package = "cellscape"
))

# cnv segments data
cnv_data <- read.csv(system.file("extdata", "cnv_data.csv",
  package =
    "cellscape"
))

# annotations
sc_annot <- read.csv(system.file("extdata", "cnv_annots.tsv",
  package =
```

```
    "cellscape"
), sep = "\t")

# custom clone colours
clone_colours <- data.frame(
  clone_id = c("1", "2", "3"),
  colour = c("7fc97f", "beaed4", "fdc086")
)

# run cellscape
cellscape(
  cnv_data = cnv_data, tree_edges = tree_edges, sc_annot = sc_annot,
  width = 800, height = 475, show_warnings = FALSE,
  clone_colours = clone_colours
)
```

`dfs_tree` *Get depth first search of a tree*

Description

Get depth first search of a tree

Widget output function for use in Shiny

Widget render function for use in Shiny

Function to get data frame of pixels

function to get min and max values for each chromosome

function to get chromosome box pixel info

function to get the genome length

function to get the number of base pairs per pixel

function to get information (chr, start, end, mode_cnv) for each pixel

function to get mutation order for targeted data

function to get targeted heatmap information

function to find the mode of a vector

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

```
dfs_tree(edges, cur_root, dfs_arr)

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

renderCnvTree(expr, env = parent.frame(), quoted = FALSE)

getEmptyGrid(hm_sc_ids_ordered, ncols)

getChromBounds(chroms, cnv_data)

getChromBoxInfo(chrom_bounds, n_bp_per_pixel)

getGenomeLength(chrom_bounds)

getNBPPerPixel(ncols, chrom_bounds, genome_length)

getCNVHeatmapForEachSC(cnv_data, chrom_bounds, n_bp_per_pixel)

getMutOrder(mut_data)

getTargetedHeatmapForEachSC(mut_data, mut_order, heatmapWidth)

findMode(x)

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, show_warnings = TRUE)

replaceSpaces(
  clonal_prev,
  tree_edges,
  clone_colours,
  mutation_info,
  mutations,
  mutation_prevalences
)
```

Arguments

edges	– edges of tree
cur_root	– current root of the tree
dfs_arr	– array of depth first search results to be filled
outputId	– id of output
width	– width provided by user
height	– height provided by user
expr	– expression for Shiny
env	– environment for Shiny
quoted	– default is FALSE
hm_sc_ids_ordered	– array of single cell ids in order
ncols	– integer of number of columns (pixels) to fill
chroms	– vector of chromosome names
cnv_data	– data frame of copy number variant segments data
chrom_bounds	– data frame of chromosome boundaries
n_bp_per_pixel	– integer of number of base pairs per pixel
genome_length	– integer of length of the genome
mut_data	– data frame of mutations data
mut_order	– array of order of mutations for heatmap (chromosome:coordinate)
heatmapWidth	– number for width of the heatmap (in pixels)
x	– vector of numbers
clonal_prev	– clonal_prev data from user
tree_edges	– tree edges data from user
mutations	– mutations data from user
clone_colours	– clone_colours data from user
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 – alpha provided by user

genotype_position – genotype_position provided by user

perturbations – perturbations provided by user

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

Examples

```
dfs_tree(
  data.frame(
    source = c("1", "1", "2", "2", "5", "6"),
    target = c("2", "5", "3", "4", "6", "7")
  ), "1", c()
)
cellscapeOutput(1, "100%", "300px")
cellscapeOutput(1, "80%", "300px")
findMode(c(1, 1, 19, 1))
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"),
    clonal_prev = c("0.1", "0.22", "0.08", "0.53", "0.009", "0.061", "1")
  ),
  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"),
    clonal_prev = c("0.12", "0.12", "0.18", "0.13", "0.009", "0.061", "1")
  ),
  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"),
    clonal_prev = c("0.12", "0.12", "0.18", "0.13", "0.009", "0.061", "1")
  )
)
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"),
    clonal_prev = c("0.12", "0.12", "0.18", "0.13", "0.009", "0.061", "1")
  ),
  mutation_prevalences = list(
    "X:6154028" = data.frame(timepoint = c("Diagnosis"), VAF = c(0.5557))
  ),
  mutation_info = data.frame(clone_id = c(1)),
  clone_colours = data.frame(
    clone_id = c("1", "2", "3", "4"),
    colour = c("#beaed4", "#fdc086", "#beaed4", "#beaed4")
  )
)
```

Index