

Package: CancerEvolutionVisualization (via r-universe)

October 16, 2026

Title Publication Quality Phylogenetic Tree Plots

Version 3.0.0

Date 2025-06-13

Description Generates tree plots with precise branch lengths, gene annotations, and cellular prevalence. The package handles complex tree structures (angles, lengths, etc.) and can be further refined as needed by the user.

License GPL-2

URL <https://github.com/uclahs-cds/package-CancerEvolutionVisualization>

BugReports

<https://github.com/uclahs-cds/package-CancerEvolutionVisualization/issues>

Depends R (>= 3.5.0), graphics, grid, gridExtra, gtable,

Imports grDevices, utils, BoutrosLab.plotting.general

Suggests testthat, rmarkdown, knitr

VignetteBuilder knitr

LazyLoad yes

LazyData yes

RoxygenNote 7.3.1

Config/pak/sysreqs libjpeg-dev libpng-dev

Repository <https://uclahs-cds.r-universe.dev>

Date/Publication 2026-08-17 06:30:37 UTC

RemoteUrl <https://github.com/uclahs-cds/public-r-cancerevolutionvisualization>

RemoteRef HEAD

RemoteSha 9ba5dfb50a11dfe55412d5db8884f711dd8d6d32

Contents

colours	2
create.ccf.densityplot	2
create.ccf.heatmap	4
create.ccf.summary.heatmap	5
create.clone.genome.distribution.plot	8
create.cluster.heatmap	9
create.phylogenetic.tree	11
data.frame.to.array	12
get.colours.in.order	13
GRCh37	14
GRCh38	14
snv	15
SRCGrob	15
update.descendant.property	19
Index	21

colours	<i>Colour scheme vector</i>
---------	-----------------------------

Description

Default colours used by CP polygons, etc.

Format

Character vector

create.ccf.densityplot	<i>CCF Density Plot</i>
------------------------	-------------------------

Description

Creates a density plot of cancer cell fraction (CCF) distribution across a sample.

Usage

```
create.ccf.densityplot(
  x,
  filename = NULL,
  clone.colours = NULL,
  xlab.label = 'CCF',
  ylab.label = 'SNV Density',
  xlimits = c(0, 1.5),
  xat = seq(0, 1.5, 0.25),
  legend.size = 3,
  legend.title.cex = 1.2,
  legend.label.cex = 1,
  legend.x = 0.8,
  legend.y = 0.9,
  height = 6,
  width = 10,
  size.units = 'in',
  resolution = 1000,
  ...
);
```

Arguments

<code>x</code>	A data-frame with the following column names: 'SNV.id', 'clone.id', 'CCF'.
<code>filename</code>	Filename for tiff output, or if NULL returns the trellis object itself. Defaults to NULL.
<code>clone.colours</code>	Named list to provide a colour scheme for the clone ID covariate bar. If NULL, colours will be randomly generated. Defaults to NULL.
<code>xlab.label</code>	Defaults to "CCF".
<code>ylab.label</code>	Defaults to "SNV Density".
<code>xlimits</code>	Limits for the x-axis. Defaults to <code>c(0, 1.5)</code> .
<code>xat</code>	Positions for the x-axis labels. Defaults to <code>seq(0, 1.5, 0.25)</code> .
<code>legend.size</code>	Width of the legend boxes in 'character' units. Defaults to 3
<code>legend.title.cex</code>	Size of titles in the legends. Defaults to 1.2
<code>legend.label.cex</code>	Size of text labels in the legends. Defaults to 1
<code>legend.x</code>	x position of the legend. Defaults to 0.8
<code>legend.y</code>	y position of the legend. Defaults to 0.9
<code>height</code>	Height of the plot. Defaults to 6
<code>width</code>	Width of the plot. Defaults to 10
<code>size.units</code>	Units for the height and width. Defaults to "in".
<code>resolution</code>	Resolution of the plot. Defaults to 1000
<code>...</code>	Pass through argument. See <code>BoutrosLab.plotting.general::create.histogram()</code> for further details.

Value

A ‘grob’ object of the heatmap.

Author(s)

Helena Winata

See Also

[create.histogram](#), [create.scatterplot](#)

create.ccf.heatmap	<i>CCF Distribution Heatmap</i>
--------------------	---------------------------------

Description

Creates a heatmap of cancer cell fraction (CCF) distribution across tumour samples. The function is a wrapper around `BoutrosLab.plotting.general::create.heatmap()` with some changes in the default parameters. All parameter description are the same as in `BoutrosLab.plotting.general::create.heatmap()` except for `ccf.thres`.

Usage

```
create.ccf.heatmap(
  x,
  cluster.dimensions = 'both',
  clustering.method = 'complete',
  distance.method = 'euclidean',
  xaxis.lab = '',
  xlab.label = 'SNVs',
  yaxis.lab = NULL,
  print.colour.key = FALSE,
  colour.scheme = c('white', 'blue'),
  ...
)
```

Arguments

<code>x</code>	Either a data-frame or a matrix from which the heatmap is to created
<code>cluster.dimensions</code>	Defaults to “both”.
<code>clustering.method</code>	Defaults to “complete”.
<code>distance.method</code>	Defaults to “euclidean”.
<code>xaxis.lab</code>	Defaults to an empty string.

xlab.label	Defaults to “SNVs”.
yaxis.lab	Defaults to the column names of array x.
print.colour.key	Defaults to FALSE.
colour.scheme	Defaults to c('white', 'blue').
...	Pass through argument. See BoutrosLab.plotting.general::create.heatmap() for further details.

Value

A ‘grob’ object of the heatmap.

Author(s)

Helena Winata

See Also

[create.heatmap](#)

Examples

```
set.seed(123);
arr <- matrix(rnorm(500), nrow = 100, ncol = 5);
rownames(arr) <- paste0('SNV.', 1:100);
colnames(arr) <- paste0('Sample.', 1:5);
create.ccf.heatmap(arr);
```

```
create.ccf.summary.heatmap
```

CCF Summary Heatmap

Description

Creates a heatmap of cancer cell fraction (CCF) distribution across tumour samples with clone IDs as a covariate beneath the heatmap. Subplot parameters controls the appearance of the heatmap and barplots. See BoutrosLab.plotting.general::create.barplot() or BoutrosLab.plotting.general::create.heatmap() for parameter description. Legend parameters are passed to BoutrosLab.plotting.general::legend.grob().

Usage

```
create.ccf.summary.heatmap(
  DF,
  ccf.limits = NULL,
  median.col = 'median.ccf.per.sample',
  clone.order = NULL,
  sample.order = NULL,
```

```

colour.scheme = c('white', 'blue'),
clone.colours = NULL,
subplot.xlab.cex = 1.2,
subplot.xaxis.cex = 1,
subplot.xaxis.fontface = 'bold',
subplot.xaxis.rot = 90,
subplot.ylab.cex = 1.2,
subplot.yaxis.cex = 1,
subplot.yaxis.fontface = 'bold',
hm.xaxis.rot = 90,
legend.size = 3,
legend.title.cex = 1.2,
legend.label.cex = 1,
legend.x = 0.9,
legend.y = 0.8,
plot.objects.heights = c(0.3, 1),
add.median.text = FALSE,
...
);

```

Arguments

DF	A data-frame with the following column names: 'ID', 'SNV.id', 'clone.id', 'CCF'.
ccf.limits	CCF limits to be applied to the heatmap. Must be a vector of length 2 for min and max thresholds. Defaults to NULL
median.col	Defaults to "median.ccf.per.sample"
clone.order	Define clone ID order. Defaults to NULL
sample.order	Define sample ID order. Defaults to NULL
colour.scheme	Heatmap colour scheme. Defaults to c('white', 'blue')
clone.colours	A named vector specifying the color to use for each clone to generate a covariate heatmap. If NULL, no covariates will be added.
subplot.xlab.cex	Subplot parameter. Defaults to 1.2
subplot.xaxis.cex	Subplot parameter. Defaults to 1
subplot.xaxis.fontface	Subplot parameter. Defaults to "bold"
subplot.xaxis.rot	Subplot parameter. Defaults to 90
subplot.ylab.cex	Subplot parameter. Defaults to 1.2
subplot.yaxis.cex	Subplot parameter. Defaults to 1
subplot.yaxis.fontface	Subplot parameter. Defaults to "bold"

hm.xaxis.rot	Subplot parameter. Defaults to 90
legend.size	Legend parameter. Defaults to 3
legend.title.cex	Legend parameter. Defaults to 1.2
legend.label.cex	Legend parameter. Defaults to 1
legend.x	Legend parameter. Defaults to 0.9
legend.y	Legend parameter. Defaults to 0.8
plot.objects.heights	If clone.colours is specified, this must be an array of length 3. Defaults to c(0.3, 1). See BoutrosLab.plotting.general::create.multipanelplot()
add.median.text	If TRUE, median CCF values will be added to the heatmap. Defaults to FALSE
...	Pass through argument. See BoutrosLab.plotting.general::create.multipanelplot() for further details.

Value

A ‘grob’ object of the summary plot.

Author(s)

Helena Winata

See Also

[legend.grob](#), [create.multipanelplot](#), [create.heatmap](#)

Examples

```
set.seed(123);
snv <- do.call(rbind, lapply(split(snv, list(snv$ID, snv$clone.id)), function(df) {
  df$median.ccf.per.sample <- median(df$CCF)
  df$total.nsnv <- length(unique(df$SNV.id))
  df
}));

create.ccf.summary.heatmap(snv);
```

```
create.clone.genome.distribution.plot
```

Create Clone Genome Distribution Plot

Description

This function creates a plot showing the distribution of clones across the genome. It generates a scatter plot of the SNVs colored by clone ID and a density plot showing the density of each clone across the genome. The function can handle both single and multi-sample inputs.

Usage

```
create.clone.genome.distribution.plot(
  snv.df,
  genome.build = 'GRCh37',
  clone.order = NULL,
  clone.colours = NULL,
  filename = NULL,
  multi.sample = FALSE,
  alpha = 0.25,
  legend.x = 0.1,
  legend.y = 0.55,
  ...
)
```

Arguments

<code>snv.df</code>	A data frame containing the SNV data. It must have columns 'chr', 'pos', and 'clone.id'. If <code>multi.sample = TRUE</code> , it must also have a column 'ID' specifying the sample ID for each SNV.
<code>genome.build</code>	The genome build to use. Defaults to "GRCh37".
<code>clone.order</code>	The order in which to plot the clones. If <code>NULL</code> , clones will be sorted alphabetically.
<code>clone.colours</code>	A named vector specifying the color to use for each clone. If <code>NULL</code> , colors will be automatically assigned.
<code>filename</code>	Directory or filepath to save the plot in. If <code>multi.sample = TRUE</code> , this must be a directory. if <code>multi.sample = FALSE</code> , this must be a filepath. If <code>NULL</code> , the plot will not be saved.
<code>multi.sample</code>	Logical indicating whether the input data contains multiple samples. Defaults to <code>FALSE</code> .
<code>alpha</code>	Transparency level of the points in the scatter plot. Defaults to 0.25.
<code>legend.x</code>	Legend parameter. Defaults to 0.1
<code>legend.y</code>	Legend parameter. Defaults to 0.55
<code>...</code>	Additional arguments to be passed to <code>BoutrosLab.plotting.general::create.multipanelplot()</code> .

Details

This function preprocesses the input data frame, extracts chromosome information, and iterates over each sample to create a clone genome distribution plot. For each sample, it calculates the density of each clone across the genome and creates a scatter plot of the SNVs colored by clone ID and a density plot showing the density of each clone.

Value

A ‘grob’ object.

Author(s)

Helena Winata, Selina Wu

See Also

[legend.grob](#), [create.multipanelplot](#), [create.scatterplot](#), [create.densityplot](#)

Examples

```
set.seed(123);  
create.clone.genome.distribution.plot(snv);
```

create.cluster.heatmap

Cluster Heatmap

Description

Creates a heatmap of cancer cell fraction (CCF) distribution across tumour samples with clone IDs as a covariate beneath the heatmap.

Usage

```
create.cluster.heatmap(  
  DF,  
  ccf.limits = NULL,  
  sample.order = NULL,  
  clone.order = NULL,  
  clone.colours = NULL,  
  height = 6,  
  width = 11,  
  xaxis.col = NULL,  
  legend.size = 3,  
  legend.title.cex = 1.2,  
  legend.label.cex = 1,  
  filename = NULL,
```

```

xlab.label = 'SNVs',
xlab.cex = 1.2,
xaxis.cex = 1,
xaxis.fontface = 'bold',
y.spacing = 1,
colour.scheme = c('white', 'blue'),
plot.objects.heights = c(1, 0.2),
...
);

```

Arguments

DF	A data-frame with the following column names: 'ID', 'SNV.id', 'clone.id', 'CCF'.
ccf.limits	CCF limits to be applied to the heatmap. Must be a vector of length 2 for min and max thresholds. Defaults to NULL
clone.order	Define clone ID order. Defaults to NULL
sample.order	Define sample ID order. Defaults to NULL
clone.colours	Named list to provide a colour scheme for the clone ID covariate bar. If NULL, colours will be randomly generated. Defaults to NULL.
height	Defaults to 6
width	Defaults to 11
xaxis.col	Column in DF to extract x-axis labels from. Defaults to NULL.
legend.size	Width of the legend boxes in 'character' units. Defaults to 3
legend.title.cex	Size of titles in the legends. Defaults to 1.2
legend.label.cex	Size of text labels in the legends. Defaults to 1
filename	Filename for tiff output, or if NULL returns the trellis object itself. Defaults to NULL.
xlab.label	Defaults to "SNVs".
xlab.cex	Defaults to 1.2
xaxis.cex	Defaults to 1
xaxis.fontface	Defaults to "bold".
y.spacing	Spacing between heatmap and clone covariate bar. Defaults to 1
colour.scheme	Colour scheme for the heatmap. Defaults to c('white', 'blue').
plot.objects.heights	Object heights. Defaults to c(1, 0.2).
...	Pass through argument. See BoutrosLab.plotting.general::create.heatmap() for further details.

Value

A 'grob' object of the heatmap.

Author(s)

Helena Winata

See Also[create.ccf.heatmap](#), [create.heatmap](#)**Examples**

```
set.seed(123);

create.cluster.heatmap(
  DF = snv,
  ccf.limits = c(0, 1),
  ylab.label = 'Sample'
);
```

`create.phylogenetic.tree`*Phylogenetic Tree Plot*

Description

Create a phylogenetic tree plot using ‘SRCGrob’. The ‘width’, ‘height’, ‘units’, ‘res’, and ‘bg’ arguments are passed to the ‘grDevices’ function to save the plot to file.

Usage

```
create.phylogenetic.tree(
  tree,
  filename = NULL,
  width = 5,
  height = 5,
  units = 'in',
  res = 500,
  bg = 'white',
  ...
)
```

Arguments

<code>tree</code>	Tree structure data.frame
<code>filename</code>	Filename for plot output, or if NULL displays the tree using <code>grid.draw</code> . Defaults to NULL.
<code>width</code>	Width of the plot. Defaults to 5.
<code>height</code>	Height of the plot. Defaults to 5.

units	Units of the plot. Defaults to “in”.
res	Resolution of the plot. Defaults to 500.
bg	Background colour of the plot. Defaults to “white”.
...	Pass through argument. See SRCGrob() for further details.

Value

A ‘grob’ of class "SRCGrob".

Author(s)

Helena Winata

See Also

[SRCGrob](#), [pdf](#), [png](#), [tiff](#), [svg](#)

Examples

```
# Simple Tree Plot
simple.tree <- data.frame(
  parent = c(NA, 1, 1)
);
create.phylogenetic.tree(simple.tree);

# Non-numeric node.id
phy.dt <- data.frame(
  node.id = c('MRCA', 'C01', 'C02', 'C03', 5:7),
  parent = c(NA, 'MRCA', 'MRCA', 'MRCA', 'C02', 'C03', 'C03')
);
create.phylogenetic.tree(phy.dt);

# Numeric disordered node.id with labels
phy.dt <- data.frame(
  node.id = c(1, 2, 5, 6, 4, 9, 10),
  parent = c(NA, 1, 1, 1, 5, 6, 6),
  label = c('MRCA', 'C01', 'C02', 'C03', 5:7)
);
create.phylogenetic.tree(phy.dt);
```

data.frame.to.array *Convert Data Frame to Array*

Description

Converts a long-format data frame into a 2D matrix/array where: - Rows correspond to unique values from the x.axis column - Columns correspond to unique values from the y.axis column - Cell values are taken from the value column - NA values are replaced with 0

Usage

```
data.frame.to.array(
  DF,
  value = 'CCF',
  x.axis = 'SNV.id',
  y.axis = 'ID'
);
```

Arguments

DF	A data frame in long format containing the data to be reshaped.
value	The name of the column containing the values to be reshaped. Defaults to 'CCF'.
x.axis	The name of the column to be used as row names in the output matrix. Defaults to 'SNV.id'.
y.axis	The name of the column to be used as column names in the output matrix. Defaults to 'ID'.

Value

A 2D 'matrix'.

Author(s)

Helena Winata

Examples

```
df <- data.frame(
  SNV.id = c(rep(paste0('snv', 1:3), 3)),
  ID = c(rep('S1', 3), rep('S2', 3), rep('S3', 3)),
  CCF = runif(9)
);

arr <- data.frame.to.array(df);
```

get.colours.in.order	<i>Generate a named vector of colors for every value specified, ordered by the value in 'value.order'.</i>
----------------------	--

Description

Assigns colors to values and ensures they follow a specified order. Any colors specified in 'pre-determined.colours' are maintained in the order specified and are used as the first colors for the 'value.list'.

Usage

```
get.colours.in.order(
  value.list,
  value.order = NULL,
  predetermined.colours = NULL
)
```

Arguments

value.list A vector of values.

value.order An optional vector specifying the order of values. If 'NULL', value order is not guaranteed.

predetermined.colours A vector of colors assigned to values. If 'NULL', colors will be generated automatically.

Value

A list containing:

colours A named vector of colors assigned to each value.

value.order The ordered values.

GRCh37

GRCh37 Chromosom Information

Description

Chromosome information for the GRCh37 genome build. Used for plotting.

Format

data.frame

GRCh38

GRCh38 Chromosom Information

Description

Chromosome information for the GRCh38 genome build. Used for plotting.

Format

data.frame

snv	<i>SNV dataframe</i>
-----	----------------------

Description

Dataframe containing the following columns: 'ID', 'SNV.id', 'CCF', 'clone.id', 'chr', 'pos'. Used to generate plot examples.

Format

data.frame

SRCGrob	<i>Subclone Tree Plot</i>
---------	---------------------------

Description

Creates a phylogenetic tree using subclonal reconstruction data. An optional node text data.frame can be used to annotate notable genes alongside branches.

Usage

```
SRCGrob(  
  tree,  
  node.text = NULL,  
  main = NULL,  
  plotting.direction = 'down',  
  horizontal.padding = 0.1,  
  scale1 = 1,  
  scale2 = 1,  
  yat = NULL,  
  yaxis1.label = NULL,  
  yaxis2.label = NULL,  
  xlab.cex = 1.55,  
  ylab.cex = 1.55,  
  xaxis.cex = 1.45,  
  yaxis.cex = 1.45,  
  xaxis.label = 'CCF',  
  label.cex = NA,  
  node.text.cex = 0.85,  
  main.y = NULL,  
  main.cex = 1.7,  
  node.text.line.dist = 0.1,  
  polygon.colour.scheme = 'grey',  
  polygon.scale = 1,  
)
```

```

add.normal = FALSE,
use.radians = FALSE,
normal.cex = 1,
label.nodes = TRUE,
disable.polygons = FALSE,
length.from.node.edge = TRUE,
size.units = 'npc',
scale.bar = FALSE,
scale.bar.coords = c(0.5, 1),
scale.size.1 = NA,
scale.size.2 = NA,
scale.padding = 1
);

```

Arguments

<code>tree</code>	Tree structure data.frame
<code>node.text</code>	Dataframe for text labels to be displayed next to nodes
<code>main</code>	Main plot title
<code>plotting.direction</code>	Sets the direction of flow for the tree. Can be one of 'down', 'right', 'up', 'left', or a number in degrees. (Default is 0, downward).
<code>horizontal.padding</code>	Increase/reduce the plot's horizontal padding proportionally. A positive value will expand the padding, and a negative value will reduce it.
<code>scale1</code>	Proportionally scale the values of the first branch length column in the tree input.
<code>scale2</code>	Proportionally scale the values of the second branch length column in the tree input.
<code>yat</code>	Specific values to be used for the y-axis ticks. A list is required, with each element corresponding to an axis.
<code>yaxis1.label</code>	Text label for the first, leftmost y-axis
<code>yaxis2.label</code>	Text label for the second, rightmost y-axis
<code>xlab.cex</code>	Font size for the x-axis label
<code>ylab.cex</code>	Font size for the y-axis labels
<code>xaxis.cex</code>	Font size for the x-axis tick labels
<code>yaxis.cex</code>	Font size for the y-axis tick labels
<code>xaxis.label</code>	Text label for the x-axis
<code>label.cex</code>	Font size for the node labels
<code>node.text.cex</code>	Font size for the node text
<code>main.y</code>	Move the main plot title position up or down
<code>main.cex</code>	Font size for the main plot title
<code>node.text.line.dist</code>	Distance between node text and tree branches (as a value between 0 and 1)

<code>add.normal</code>	Adds a normal
<code>use.radians</code>	Unit to be used for 'angle' column (degrees or radians)
<code>normal.cex</code>	Font size within the normal 'box'
<code>label.nodes</code>	Enable/disable node labels
<code>disable.polygons</code>	Disables CCF polygon drawing (even when CCF values are provided)
<code>polygon.colour.scheme</code>	Vector of colour values to be used for CCF polygons. Creates a gradient color scheme.
<code>polygon.scale</code>	Multiplier to scale CCF polygons (must be positive).
<code>length.from.node.edge</code>	Sets the branch length to be calculated from the edge of the node instead of the centre
<code>size.units</code>	Grid units to be used for all specific size/length parameters
<code>scale.bar</code>	Set to TRUE to add a scale bar instead of y-axis
<code>scale.bar.coords</code>	Coordinates for the scale bar placement <code>c(x.pos, y.pos)</code>
<code>scale.size.1</code>	Override size of the first scale bar
<code>scale.size.2</code>	Override size of the second scale bar
<code>scale.padding</code>	Padding between scale bars if more than 1 is present

Value

A 'grob' of class 'SRCGrob'

Author(s)

Dan Knight

Examples

```
# Simple Tree Plot
simple.tree <- data.frame(
  parent = c(NA, 1, 1)
);

SRCGrob(simple.tree);

# Specify Branch Lengths
branch.lengths.tree <- data.frame(
  simple.tree,
  length1 = c(10, 20, 15)
);

SRCGrob(branch.lengths.tree);
```

```
# Cellular Prevalence
CCF.tree <- data.frame(
  branch.lengths.tree,
  CCF = c(1, 0.3, 0.5)
);

SRCGrob(CCF.tree);

# Scaled Branch Lengths
SRCGrob(
  branch.lengths.tree,
  scale1 = 0.8
);

# Override Branch Angles in Degrees
degrees.tree <- data.frame(
  parent = c(NA, 1, 2),
  angle = c(NA, NA, 90)
);

SRCGrob(degrees.tree);

# Override Branch Angles in Radians
radians.tree <- data.frame(
  parent = c(NA, 1, 2),
  angle = c(NA, NA, pi / 2)
);

SRCGrob(
  radians.tree,
  use.radians = TRUE
);

# Plot Title
SRCGrob(
  simple.tree,
  main = 'Example Plot'
);

# Y-Axis Label
SRCGrob(
  branch.lengths.tree,
  yaxis1.label = 'SNVs'
);

# Y-Axis Ticks
```

```
yaxis1.ticks <- seq(0, 10, 2);

SRCGrob(
  branch.lengths.tree,
  yat = list(yaxis1.ticks)
);

# Scale Bar
SRCGrob(
  branch.lengths.tree,
  yaxis1.label = 'SNVs',
  scale.bar = TRUE
);

# Normal
SRCGrob(
  simple.tree,
  add.normal = TRUE
);

# Nodeless Mode
nodeless.tree <- data.frame(
  parent = c(NA, 1, 2, 2),
  draw.node = c(TRUE, FALSE, TRUE, TRUE)
);

SRCGrob(nodeless.tree);
```

update.descendant.property

Update Tree Descendant Property

Description

Function to iteratively update a specified property for all descendants of a given parent node(s) in a tree structure. The function works recursively, meaning it will update values for all descendants, not just immediate children. If a parent_id has no children, the original data frame is returned unchanged.

Usage

```
update.descendant.property(
  x,
  parent.id,
  property,
  value,
  overwrite = TRUE
);
```

Arguments

<code>x</code>	A data frame representing a tree structure with at least the following columns: <code>parent</code> (parent node identifiers), and the column specified by <code>property</code>
<code>parent.id</code>	Character or numeric vector specifying the identifier(s) of the parent node(s)
<code>property</code>	Character string specifying the name of the column/property to be updated
<code>value</code>	The value to assign to the property column
<code>overwrite</code>	Logical; if TRUE, overwrites existing values; if FALSE, only fills in NA values. Defaults to TRUE

Value

A 'data.frame' with modified columns.

Author(s)

Helena Winata

Examples

```
tree <- data.frame(
  node.id = c('A', 'B', 'C', 'D', 'E', 'F'),
  parent = c(NA, 'A', 'B', 'A', 'C', 'E'),
  node.size = 1
);

tree2 <- update.descendant.property(
  x = tree,
  parent.id = 'B',
  property = 'node.size',
  value = 2
);
```

Index

colours, [2](#)
create.ccf.densityplot, [2](#)
create.ccf.heatmap, [4](#), [11](#)
create.ccf.summary.heatmap, [5](#)
create.clone.genome.distribution.plot,
 [8](#)
create.cluster.heatmap, [9](#)
create.densityplot, [9](#)
create.heatmap, [5](#), [7](#), [11](#)
create.histogram, [4](#)
create.multipanelplot, [7](#), [9](#)
create.phylogenetic.tree, [11](#)
create.scatterplot, [4](#), [9](#)

data.frame.to.array, [12](#)

get.colours.in.order, [13](#)
GRCh37, [14](#)
GRCh38, [14](#)

legend.grob, [7](#), [9](#)

pdf, [12](#)
png, [12](#)

snv, [15](#)
SRCGrob, [12](#), [15](#)
svg, [12](#)

tiff, [12](#)

update.descendant.property, [19](#)