

Package ‘vtree’

August 10, 2020

Type Package

Title Display Information About Nested Subsets of a Data Frame

Version 5.0.0

Depends R (>= 2.10)

Author Nick Barrowman [aut, cre]

Sebastian Gatscha <kona1@gmx.at> [aut] (Shiny interface)

Andrea Leofreddi <a.leofreddi@vleo.net> [cph] (svg-pan-zoom library)

Maintainer Nick Barrowman <nbarrowman@cheo.on.ca>

Description A tool for calculating and drawing “variable trees”. Variable trees display information about nested subsets of a data frame.

License GPL-3

URL <https://github.com/nbarrowman/vtree>,
<https://nbarrowman.github.io/vtree>

BugReports <https://github.com/nbarrowman/vtree/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

Suggests knitr, rmarkdown, ggplot2, testthat (>= 2.1.0)

Imports DiagrammeR, DiagrammeRsvg, rsvg, htmlwidgets, shiny

NeedsCompilation no

Repository CRAN

Date/Publication 2020-08-10 18:20:02 UTC

R topics documented:

vtree-package	2
build.data.frame	3
CrosstabToCases	4

FakeData	4
FakeRCT	5
grVizToPNG	6
renderVtree	7
svtree	8
use_svgzoom	8
VennTable	10
vtree	11
vtreeOutput	21

Index	23
--------------	-----------

vtree-package	<i>vtree: a tool for calculating and drawing variable trees.</i>
---------------	--

Description

vtree is a flexible tool for generating variable trees — diagrams that display information about nested subsets of a data frame. Given simple specifications, the vtree function produces these diagrams and automatically labels them with counts, percentages, and other summaries.

With vtree, you can:

- explore a data set interactively, and
- produce customized figures for reports and publications.

For a comprehensive introduction, type: `vignette("vtree")`

Author(s)

Nick Barrowman <nbarrowman@cheo.on.ca>

See Also

- <https://nbarrowman.github.io/vtree>
- <https://github.com/nbarrowman/vtree>
- Report bugs at <https://github.com/nbarrowman/vtree/issues>

build.data.frame *Build a data frame to display with vtree*

Description

Build a data frame by specifying variable names and patterns of values together with frequencies.

Usage

```
build.data.frame(varnames, ...)
```

Arguments

varnames A vector of variable names.

... Lists of patterns and the frequency of each pattern. When a pattern is shorter than the list of variable names (for example, 3 variable names but only 2 values in the pattern), NA's are substituted for the missing variable names.

Details

Suppose varnames=c("animal", "size", "hair"), then one pattern would be list("dog", "small", "short", 4), which specifies 4 dogs that are small and short-haired. Another pattern could be list("cat", "large", "long", 101), specifying 101 large cats.

Value

A data frame.

Author(s)

Nick Barrowman <nbarrowman@cheo.on.ca>

Examples

```
# Number of countries in Africa, whether population is over 30 million or not,  
# and whether landlocked or not.  
# https://www.worldometers.info/geography/how-many-countries-in-africa/  
#  
df <- build.data.frame(  
  c("continent", "population", "landlocked"),  
  list("Africa", "Over 30 million", "landlocked", 2),  
  list("Africa", "Over 30 million", "not landlocked", 12),  
  list("Africa", "Under 30 million", "landlocked", 14),  
  list("Africa", "Under 30 million", "not landlocked", 26))
```

`crosstabToCases` *Convert a crosstabulation into a data frame of cases.*

Description

Convert a table of crosstabulated counts into a data frame of cases.

Usage

```
crosstabToCases(x)
```

Arguments

`x` a matrix or table of frequencies representing a crosstabulation.

Value

Returns a data frame of cases.

Author(s)

Nick Barrowman, based on the `countsToCases` function at http://www.cookbook-r.com/Manipulating_data/Converting_between_data_frames_and_contingency_tables/#countstocases-function

Examples

```
# The Titanic data set is in the datasets package.  
# Convert it from a 4 x 2 x 2 x 2 crosstabulation  
# to a 4-column data frame of 2201 individuals  
titanic <- crosstabToCases(Titanic)
```

`FakeData` *Fake clinical dataset*

Description

A dataset consisting of made-up clinical data. Note that some observations are missing (i.e. NAs).

Usage

```
FakeData
```

Format

A small data frame in which the rows represent (imaginary) patients and the columns represent variables of possible clinical relevance.

id Integer: Patient ID number

Group Factor: Treatment Group, A or B

Severity Factor representing severity of condition: Mild, Moderate, or Severe

Sex Factor: M or F

Male Integer: Sex coded as 1=M, 0=F

Age Integer: Age in years, continuous

Score Integer: Score on a test

Category Factor: single, double, or triple

Pre Numeric: initial measurement

Post Numeric: measurement taken after something happened

Post2 Numeric: measurement taken at the very end of the study

Time Numeric: time to event, or time of censoring

Event Integer: Did the event occur? 1=yes, 0=no (i.e. censoring)

Ind1 Integer: Indicator variable for a certain characteristic, 1=present, 0=absent

Ind2 Integer: Indicator variable for a certain characteristic, 1=present, 0=absent

Ind3 Integer: Indicator variable for a certain characteristic, 1=present, 0=absent

Ind4 Integer: Indicator variable for a certain characteristic, 1=present, 0=absent

Viral Logical: Does this patient have a viral illness?

FakeRCT

Fake Randomized Controlled Trial (RCT) data

Description

A dataset consisting of made-up RCT data.

Usage

FakeRCT

Format

A small data frame in which the rows represent (imaginary) patients and the columns represent variables of possible clinical relevance.

id String: Patient ID number

eligible Factor: Eligible or Ineligible

randomized Factor: Randomized or Not randomized

group Factor: A or B

followup Factor: Followed up or Not followed up

analyzed Factor: Analyzed or Not analyzed

grVizToPNG

Export an htmlwidget object into a PNG file

Description

Export an htmlwidget object (produced by `DiagrammeR::grViz`) into a PNG file

Usage

```
grVizToPNG(g, width = NULL, height = NULL, folder = ".", filename)
```

Arguments

<code>g</code>	an object produced by the <code>grViz</code> function from the <code>DiagrammeR</code> package
<code>width</code>	the width in pixels of the bitmap
<code>height</code>	the height in pixels of the bitmap
<code>folder</code>	path to folder where the PNG file should stored
<code>filename</code>	an optional filename. If not provided, the filename will be derived from the name of the argument of <code>g</code> .

Details

First the `grViz` object is exported to an SVG file (using `DiagrammeRsvg::export_svg`). Then the SVG file is converted to a bitmap (using `rsvg::rsvg`). Then the bitmap is exported as a PNG file (using `png::writePNG`). Note that the SVG file and the PNG file will be named using the name of the `g` parameter

Value

Returns the full path of the PNG file.

Note

In addition to the `DiagrammeR` package, the following packages are used: `DiagrammeRsvg`, `rsvg`

Author(s)

Nick Barrowman

`renderVtree`*vtree widget*

Description

Shiny bindings for vtree

Usage`renderVtree(expr, env = parent.frame(), quoted = FALSE)`**Arguments**

<code>expr</code>	an expression that generates a variable tree
<code>env</code>	the environment in which to evaluate <code>expr</code> .
<code>quoted</code>	is <code>expr</code> a quoted expression (with <code>quote()</code>)? This is useful if you want to save an expression in a variable.

See Also[vtreeOutput](#), [vtree](#)Other Shiny Functions: [init_js\(\)](#), [inlineCssSetup\(\)](#), [use_svgzoom\(\)](#), [vtreeOutput\(\)](#)**Examples**

```
## Not run:
library(shiny)
library(vtree)

ui <- fluidPage(
  vtreeOutput("vtree", width = "100%", height = "800px")
)

server <- function(input, output, session) {
  output$vtree <- renderVtree({
    vtree(FakeData,"Severity Sex",
          labelnode=list(Sex=(c("Male"="M","Female"="F"))),
          pngknit=FALSE)
  })
}

shinyApp(ui, server)

## End(Not run)
```

`svtree`*Create a Shiny vtree, with svg-pan-zoom functionality.*

Description

‘svtree’ uses Shiny and the `svg-pan-zoom` JavaScript library to create a variable tree with panning and zooming functionality. The mousewheel allows you to zoom in or out. The variable tree can also be dragged to a different position.

Usage

```
svtree(...)
```

Arguments

... parameters to be passed to ‘vtree’

Details

The `svg-pan-zoom` library webpage is <https://github.com/ariutta/svg-pan-zoom>

`use_svgzoom`*Setup for interactive Vtree*

Description

This function must be called in the UI, in order to make the `vtree` interactive.

Usage

```
use_svgzoom(  
  minheight = "200px",  
  cursor_all = "all-scroll",  
  overflow = "inherit !important",  
  position = "sticky",  
  fill = "transparent",  
  cursor_text = "pointer",  
  init_event = c("mouseenter", "click", "dblclick"),  
  onwindow_resize = TRUE,  
  shortcuts = TRUE  
)
```

Arguments

minheight	minimum height in "px". Default is "200px".
cursor_all	The cursor symbol for the whole SVG. Default is "all-scroll".
overflow	Overflow value for the whole SVG. Default is "inherit".
position	CSS position of the SVG. Default is "sticky".
fill	Fill color for the SVG background. Default is "transparent".
cursor_text	The cursor symbol for text nodes. Default is "pointer".
init_event	The mouse event to activate zooming and panning. Default is mouseenter.
onwindow_resize	Should the SVG be resized when the window size changes? Default is TRUE.
shortcuts	Should Keyboard shortcuts be used to control the SVG? Default is TRUE.

See Also

[vtreeOutput](#), [vtree](#)

Other Shiny Functions: [init_js\(\)](#), [inlineCssSetup\(\)](#), [renderVtree\(\)](#), [vtreeOutput\(\)](#)

Examples

```
## Not run:
library(shiny)
library(vtree)

ui <- fluidPage(
  use_svgzoom(),
  helpText(div(style="font-weight: 800; font-size: large; color: black;",
    HTML("Zooming and Panning is possible with mouse-drag ",
      "and mouse-wheel <br>, or with shortcuts;",
      " +,- and arrow-keys and CTRL+Backspace to",
      " resize+fit+center the svg.."))),
  vtreeOutput("vtree", width = "100%", height = "500px")
)

server <- function(input, output, session) {
  output$vtree <- renderVtree({
    vtree(FakeData,"Severity Sex",
      labelnode=list(Sex=(c("Male"="M","Female"="F"))),
      pngknit=FALSE)
  })
}

shinyApp(ui, server)

## End(Not run)
```

VennTable

Format an indicator-based pattern table

Description

Given a pattern table produced by `vtree` for indicator (i.e 0/1) variables, `VennTable` returns an augmented table. The augmented table includes an extra row with the total for each indicator variable and an extra row with the corresponding percentage (which will not in general add to 100%). Also, optionally, does some additional formatting for pandoc markdown.

Usage

```
VennTable(
  x,
  markdown = FALSE,
  NAcode = "-",
  unchecked = c("0", "FALSE", "No", "no", "not N/A"),
  checked = c("1", "TRUE", "Yes", "yes", "N/A"),
  sort = TRUE
)
```

Arguments

<code>x</code>	Required: Pattern table produced by <code>vtree</code> for indicator (i.e 0/1) variables
<code>markdown</code>	Format nicely for markdown (see Details).
<code>NAcode</code>	Code to use to represent NAs in markdown formatting
<code>unchecked</code>	Vector of character strings that represent unchecked values; by default: <code>c("0", "FALSE", "No", "no", "not N/A")</code>
<code>checked</code>	Vector of character strings that represent checked values; by default: <code>c("1", "TRUE", "Yes", "yes", "N/A")</code>
<code>sort</code>	Sort variables by frequency?

Details

The column totals ignore missing values.

When `markdown=TRUE`, the row and column headings for percentages are labeled "%", indicator values equal to 1 are replaced by checkmark codes, indicator values equal to 0 are replaced by spaces, and missing indicator values are replaced by dashes. Empty headings are replaced by spaces. Finally the table is transposed.

Value

Returns a character matrix with extra rows containing indicator sums.

Author(s)

Nick Barrowman

Examples

```
# Generate a pattern table for the indicator variables Ind1 and Ind2
ptab <- vtree(FakeData,"Ind1 Ind2",ptable=TRUE)
# Augment the table
ptab2 <- VennTable(ptab)
# Print the result without quotation marks (which are distracting)
print(ptab2,quote=FALSE)
# Generate a table with pandoc markdown formatting
ptab3 <- VennTable(ptab,markdown=TRUE)
```

vtree

Draw a variable tree

Description

vtree is a tool for drawing variable trees. Variable trees display information about nested subsets of a data frame, in which the subsetting is defined by the values of categorical variables.

Usage

```
vtree(
  z,
  vars,
  auto = FALSE,
  splitspaces = TRUE,
  prune = list(),
  prunebelow = list(),
  keep = list(),
  follow = list(),
  prunelone = NULL,
  pruneNA = FALSE,
  prunesmaller = NULL,
  labelnode = list(),
  tlabelnode = NULL,
  labelvar = NULL,
  varminwidth = NULL,
  varminheight = NULL,
  varlabelloc = NULL,
  fillcolor = NULL,
  fillnodes = TRUE,
  NAfillcolor = "white",
  rootfillcolor = "#EFF3FF",
  palette = NULL,
  gradient = TRUE,
  revgradient = FALSE,
  sortfill = FALSE,
```

```
singlecolor = 2,
colorvarlabels = TRUE,
title = "",
sameline = FALSE,
Venn = FALSE,
check.is.na = FALSE,
seq = FALSE,
pattern = FALSE,
ptable = FALSE,
showroot = TRUE,
text = list(),
ttext = list(),
plain = FALSE,
squeeze = 1,
showvarinnode = FALSE,
shownodelabels = TRUE,
showvarnames = TRUE,
showlevels = TRUE,
showpct = TRUE,
showlpct = TRUE,
showcount = TRUE,
showlegend = FALSE,
varnamepointsize = 24,
varnamebold = FALSE,
legendpointsize = 14,
HTMLtext = FALSE,
digits = 0,
cdigits = 1,
splitwidth = 20,
vsplitwidth = 8,
lsplitwidth = 15,
getscript = FALSE,
nodesep = 0.5,
ranksep = 0.5,
margin = 0.2,
vp = TRUE,
horiz = TRUE,
summary = "",
runsummary = NULL,
retain = NULL,
width = NULL,
height = NULL,
graphattr = "",
nodeattr = "",
edgeattr = "",
color = c("blue", "forestgreen", "red", "orange", "pink"),
colornodes = FALSE,
mincount = 1,
```

```

maxcount,
showempty = FALSE,
rounded = TRUE,
nodefunc = NULL,
nodeargs = NULL,
choicechecklist = TRUE,
arrowhead = "normal",
pxwidth,
pxheight,
imagewidth = "",
imageheight = "",
folder,
pngknit = TRUE,
as.if.knit = FALSE,
maxNodes = 1000,
unchecked = c("0", "FALSE", "No", "no"),
checked = c("1", "TRUE", "Yes", "yes"),
just = "c",
verbose = FALSE,
parent = 1,
last = 1,
root = TRUE,
subset = 1:nrow(z)
)

```

Arguments

<code>z</code>	Required: Data frame, or a single vector.
<code>vars</code>	Required (unless <code>z</code> is a vector): Either a character string of whitespace-separated variable names or a vector of variable names.
<code>auto</code>	Automatically choose variables? (<code>vars</code> should not be specified)
<code>splitspaces</code>	When <code>vars</code> is a character string, split it by spaces to get variable names? It is only rarely necessary to use this parameter. This should only be <code>FALSE</code> when a single variable name that contains spaces is specified.
<code>prune</code>	List of vectors that specifies nodes to prune. The name of each element of the list must be one of the variable names in <code>vars</code> . Each element is a vector of character strings that identifies the values of the variable (i.e. the nodes) to prune.
<code>prunebelow</code>	Like <code>prune</code> but instead of pruning the specified nodes, their descendants are pruned.
<code>keep</code>	Like <code>prune</code> but specifies which nodes to <i>keep</i> . The other nodes will be pruned.
<code>follow</code>	Like <code>keep</code> but specifies which nodes to "follow", i.e. which nodes' <i>descendants</i> to keep.
<code>prunelone</code>	(Deprecated) A vector of values specifying "lone nodes" (of <i>any</i> variable) to prune. A lone node is a node that has no siblings (an "only child").
<code>pruneNA</code>	(Deprecated) Prune all missing values? This is problematic because "valid" percentages are hard to interpret when NAs are pruned.

prunesmaller	Prune any nodes with count less than specified number.
labelnode	List of vectors used to change how values of variables are displayed. The name of each element of the list is one of the variable names in vars. Each element of the list is a vector of character strings, representing the values of the variable. The names of the vector represent the labels to be used in place of the values.
tlabelnode	A list of vectors, each of which specifies a particular node, as well as a label for that node (a "targeted" label). The names of each vector specify variable names, except for an element named label, which specifies the label to use.
labelvar	A named vector of labels for variables.
varminwidth	A named vector of minimum initial widths for nodes of each variable. (Sets the Graphviz width attribute.)
varminheight	A named vector of minimum initial heights for nodes of each variable. (Sets the Graphviz height attribute.)
varlabelloc	A named vector of vertical label locations ("t", "c", or "b" for top, center, or bottom, respectively) for nodes of each variable. (Sets the Graphviz labelloc attribute.)
fillcolor	A named vector of colors for filling the nodes of each variable. If an unnamed, scalar color is specified, all nodes will have this color.
fillnodes	Fill the nodes with color?
NAfillcolor	Fill-color for missing-value nodes. If NULL, fill colors of missing value nodes will be consistent with the fill colors in the rest of the tree.
rootfillcolor	Fill-color for the root node.
palette	A vector of palette numbers (which can range between 1 and 14). The names of the vector indicate the corresponding variable. See Palettes below for more information.
gradient	Use gradients of fill color across the values of each variable? A single value (with no names) specifies the setting for all variables. A logical vector of TRUE values for named variables is interpreted as TRUE for those variables and FALSE for all others. A logical vector of FALSE values for named variables is interpreted as FALSE for those variables and TRUE for all others.
revgradient	Should the gradient be reversed (i.e. dark to light instead of light to dark)? A single value (with no names) specifies the setting for all variables. A logical vector of TRUE values for named variables is interpreted as A logical vector of FALSE values for named variables is interpreted as FALSE for those variables and TRUE for all others.
sortfill	Sort colors in order of node count? When a gradient fill is used, this results in the nodes with the smallest counts having the lightest shades and the nodes with the largest counts having the darkest shades.
singlecolor	When a variable has a single value, this parameter is used to specify whether nodes should have a (1) light shade, (2) a medium shade, or (3) a dark shade. specify singlecolor=1 to assign a light shade.
colorvarlabels	Color the variable labels?
title	Optional title for the root node of the tree.

sameline	Display node labels on the same line as the count and percentage?
Venn	Display multi-way set membership information? This provides an alternative to a Venn diagram. This sets <code>showpct=FALSE</code> and <code>shownodelabels=FALSE</code> . Assumption: all of the specified variables are logicals or 0/1 numeric variables.
check.is.na	Replace each variable named in <code>vars</code> with a logical vector indicating whether or not each of its values is missing?
seq	Display the variable tree using "sequences"? Each unique sequence (i.e. pattern) of values will be shown separately. The sequences are sorted from least frequent to most frequent.
pattern	Same as <code>seq</code> , but lines without arrows are drawn, and instead of a sequence variable, a <code>pattern</code> variable is shown.
ptable	Generate a pattern table instead of a variable tree? Only applies when <code>pattern=TRUE</code> .
showroot	Show the root node? When <code>seq=TRUE</code> , it may be useful to set <code>showroot=FALSE</code> .
text	A list of vectors containing extra text to add to nodes corresponding to specified values of a specified variable. The name of each element of the list must be one of the variable names in <code>vars</code> . Each element is a vector of character strings. The names of the vector identify the nodes to which the text should be added. (See Formatting codes below for information on how to format text.)
ttext	A list of vectors, each of which specifies a particular node, as well as text to add to that node ("targeted" text). The names of each vector specify variable names, except for an element named <code>text</code> , which specifies the text to add.
plain	Use "plain" settings? These settings are as follows: for each variable all nodes are the same color, namely a shade of blue (with each successive variable using a darker shade); all variable labels are black; and the <code>squeeze</code> parameter is set to 0.6.
squeeze	The degree (between 0 and 1) to which the tree will be "squeezed". This controls two Graphviz parameters: <code>margin</code> and <code>nodesep</code> .
showvarinnode	Show the variable name in each node?
shownodelabels	Show node labels? A single value (with no names) specifies the setting for all variables. Otherwise, a named logical vector indicates which variables should have their node labels shown. If the vector consists of only <code>TRUE</code> values, it is interpreted as <code>TRUE</code> for those variables and <code>FALSE</code> for all others. Similarly, if the vector consists of only <code>FALSE</code> values, it is interpreted as <code>FALSE</code> for those variables and <code>TRUE</code> for all others.
showvarnames	Show the name of the variable next to each level of the tree?
showlevels	(Deprecated) Same as <code>showvarnames</code> .
showpct	Show percentage in each node? A single value (with no names) specifies the setting for all variables. A logical vector of <code>TRUE</code> for named variables is interpreted as A logical vector of <code>FALSE</code> for named variables is interpreted as <code>FALSE</code> for those variables and <code>TRUE</code> for all others.
showlpct	Show percentages (for the marginal frequencies) in the legend?
showcount	Show count in each node? A single value (with no names) specifies the setting for all variables. A logical vector of <code>TRUE</code> for named variables is interpreted as A logical vector of <code>FALSE</code> for named variables is interpreted as <code>FALSE</code> for those variables and <code>TRUE</code> for all others.

showlegend	Show legend (including marginal frequencies) for each variable?
varnamepointsize	Font size (in points) to use when displaying variable names.
varnamebold	Show the variable name in bold?
legendpointsize	Font size (in points) to use when displaying legend.
HTMLtext	Is the text formatted in HTML?
digits	Number of decimal digits to show in percentages.
cdigits	Number of decimal digits to show in continuous values displayed via the summary parameter.
splitwidth	The minimum number of characters before an automatic linebreak is inserted.
vsplitwidth	In variable names, the minimum number of characters before an automatic linebreak is inserted.
lsplitwidth	(Deprecated) In legends, the minimum number of characters before an automatic linebreak is inserted.
getscript	Instead of displaying the variable tree, return the DOT script as a character string?
nodesep	Graphviz attribute: Node separation amount.
ranksep	Graphviz attribute: Rank separation amount.
margin	Graphviz attribute: node margin.
vp	Use "valid percentages"? Valid percentages are computed by first excluding any missing values, i.e. restricting attention to the set of "valid" observations. The denominator is thus the number of non-missing observations. When vp=TRUE, nodes for missing values show the number of missing values but do not show a percentage; all the other nodes show valid percentages. When vp=FALSE, all nodes (including nodes for missing values) show percentages of the total number of observations.
horiz	Should the tree be drawn horizontally? (i.e. parent node on the left, with the tree growing to the right)
summary	A character string used to specify summary statistics to display in the nodes. See Displaying summary information below for details.
runsummary	A list of functions, with the same length as summary. Each function must take a data frame as its sole argument, and return a logical value. Each string in summary will only be interpreted if the corresponding logical value is TRUE. the corresponding string in summary will be evaluated.
retain	Vector of names of additional variables in the data frame that need to be available to execute the functions in runsummary.
width	Width (in pixels) to be passed to DiagrammeR::grViz.
height	Height (in pixels) to be passed to DiagrammeR::grViz.
graphattr	Character string: Additional attributes for the Graphviz graph.
nodeattr	Character string: Additional attributes for Graphviz nodes.
edgeattr	Character string: Additional attributes for Graphviz edges.

color	A vector of color names for the <i>outline</i> of the nodes at each level.
colornodes	Color the node outlines?
mincount	Minimum count to include in a pattern tree or pattern table.
maxcount	Maximum count to include in a pattern tree or pattern table. (Overrides mincount.)
showempty	Show nodes that do not contain any observations?
rounded	Use rounded boxes for nodes?
nodefunc	A node function (see Node functions below).
nodeargs	A list containing named arguments for the node function specified by nodefunc.
choicechecklist	When REDCap checklists are specified using the stem: syntax, automatically extract the names of choices and use them as variable names?
arrowhead	DOT arrowhead style. Defaults to "normal". Other choices include "none", "vee".
pxwidth	Width in pixels of the PNG bitmap to be rendered when vtree is called from R Markdown. If neither pxwidth nor pxheight is specified, pxwidth is automatically set to 2000 pixels.
pxheight	Height in pixels of the PNG bitmap to be rendered when vtree is called from R Markdown.
imagewidth	A character string specifying the width of the PNG image to be rendered when vtree is called from R Markdown, e.g. "4in"
imageheight	A character string specifying the height of the PNG image to be rendered when vtree is called from R Markdown, e.g. "5in". If neither imageheight nor imagewidth is specified, imageheight is set to 3 inches.
folder	Optional path to a folder where the PNG file should stored when called during knit
pngknit	Generate a PNG file when called during knit?
as.if.knit	Behave as if called while knitting?
maxNodes	An error occurs if the number of nodes exceeds maxNodes, which defaults to 1000.
unchecked	Vector of string values interpreted as "unchecked".
checked	Vector of string values interpreted as "checked".
just	Text justification ("l"=left, "c"=center, "r"=right).
verbose	Report additional details?
parent	Parent node number (Internal use only.)
last	Last node number (Internal use only.)
root	Is this the root node of the tree? (Internal use only.)
subset	A vector representing the subset of observations. (Internal use only.)

Value

The value returned by `vtree` varies depending on both the parameter values specified and the context in which `vtree` is called.

First, there are two special cases where `vtree` does not show a variable tree:

- If `ptable=TRUE`, the return value is a data frame representing a pattern table.
- Otherwise, if `getscript=TRUE`, the return value is a character string, consisting of a DOT script that describes the variable tree.

If neither of the above cases applies, the return value is as follows. If knitting is *not* taking place (such as when `vtree` is used **interactively**):

- the return value is an object of class `htmlwidget` (see [DiagrammeR](#)). It will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

If knitting *is* taking place:

- If `pngknit=TRUE` (the default), the return value is a character string of pandoc markdown code to embed a PNG file with fully-specified path. The character string will have class `knit_asis` so that `knitr` will treat it as `asis` (the effect is the same as the chunk option `results = 'asis'`) when it is written to the output. (See `?knitr::asis_output`)
- If `pngknit=FALSE`, the return value is the same as when knitting is not taking place, i.e. an object of class `htmlwidget`.

Nested list returned as the ‘info’ attribute

- The top level of the list represents the root node. Within this list is a list named after the first variable in the tree.
- In turn, within this list are lists named after the observed values of that variable.
- In turn, each of these lists is an element named after the next variable in the tree.
- And so on ...

The root element as well as each list element named after a value of a variable also contains elements `.n` (representing the number of observations), `.pct` (representing the percentage), and `.txt` (representing additional text such as summaries).

R Markdown

As noted in the **Value** section above, `vtree` has special support for R Markdown.

By default, when knitting an R Markdown file, `vtree` generates PNG files and embeds them automatically in the output document. This feature is needed when knitting to a `.docx` file. When knitting to HTML, it is not necessary to generate PNG files because HTML browsers can directly display `htmlwidgets`.

To generate `htmlwidgets` instead of PNG files, specify `pngknit=FALSE`. (Note, however, that there are some advantages to embedding PNG files in an HTML file. For example, some browsers perform poorly when numerous `htmlwidgets` are included in an HTML file.)

When PNG files are generated, they are stored by default in a temporary folder. The folder can also be specified using the `folder` parameter. (Using the base R function options, a custom option `vtree_folder` is used to automatically keep track of this.) Successive PNG files generated by an R Markdown file are named `vtree1.png`, `vtree2.png`, etc. (A custom option `vtree_count` is used to automatically keep track of the number of PNG files.)

Displaying summary information

The `summary` parameter allows you to specify information to display in each node. The parameter can be specified as a vector of character strings, where each element represents a different variable to summarize. When an element of `summary` is specified as a single variable name, the following default set of summary statistics is shown: the variable name, number of missing values, mean and standard deviation, median and interquartile range and range. A customized summary is shown when an element of `summary` is specified as a character string with the following structure:

- First, the name of the variable for which a summary is desired.
- Next a space.
- The remainder of the string specifies what to display, with text as well as special codes (see **Summary codes** below) to indicate the type of summary desired and to control which nodes display the summary, etc.

Summary codes

- `%mean%` mean. Variant `%meanx%` does not report missing values.
- `%SD%` standard deviation. Variant `%SDx%` does not report missing values.
- `%sum%` sum. Variant `%sumx%` does not report missing values.
- `%min%` minimum. Variant `%minx%` does not report missing values.
- `%max%` maximum. Variant `%maxx%` does not report missing values.
- `%range%` range. Variant `%rangex%` does not report missing values.
- `%pX%` Xth percentile, e.g. `p50` means the 50th percentile
- `%median%` median. Variant `%medianx%` does not report missing values.
- `%IQR%` interquartile range. Variant `%IQRx%` does not report missing values.
- `%npct%` number and percentage of TRUE values
- `%pct%` percentage of TRUE values
- `%freqpct%` frequency and percentage of values of a variable. Variant `%freqpct_%` shows each value on a separate line
- `%freq%` frequency of values of a variable. Variant `%freq_%` shows each value on a separate line
- `%list%` list of the individual values. Variant `%list_%` shows each value on a separate line
- `%mv%` the number of missing values
- `%nonmv%` the number of non-missing values
- `%v%` the name of the variable
- `%noroot%` flag: Do not show summary in the root node.

- `%leafonly%` flag: Only show summary in leaf nodes.
- `%var=V%` flag: Only show summary in nodes of variable V.
- `%node=N%` flag: Only show summary in nodes with value N.
- `%trunc=n%` flag: Truncate the summary to the first n characters.

Formatting codes

Formatting codes for the text argument. Also used by `labelnode` and `labelvar`.

- `\n` line break
- `*...*` italics
- `**...**` bold
- `^...^` superscript (using 10 point font)
- `~...~` subscript (using 10 point font)
- `%red ...%` display text in red (or whichever color is specified)

Palettes

The following palettes (obtained from RColorBrewer) are used in the order indicated:

1	Reds	5	Purples	9	YlOrBr	13	RdYlGn
2	Blues	6	YlGn	10	PuBuGn	14	Set1
3	Greens	7	PuBu	11	BuPu		
4	Oranges	8	PuRd	12	YlOrRd		

Author(s)

Nick Barrowman <nbarrowman@cheo.on.ca>

Examples

```
# Call to vtree
vtree(FakeData,"Sex Severity")

# R Markdown inline call to vtree
# `r vtree(FakeData,"Sex Severity")`

# A single-level hierarchy
vtree(FakeData,"Severity")

# A two-level hierarchy
vtree(FakeData,"Severity Sex")

# Rename some nodes
vtree(FakeData,"Severity Sex",labelnode=list(Sex=(c("Male"="M","Female"="F"))))
```

```

# Rename a variable
vtree(FakeData,"Severity Sex",labelvar=c(Severity="How bad?"))

# Show legend. Put labels on the same line as counts and percentages
vtree(FakeData,"Severity Sex Viral",sameline=TRUE,showlegend=TRUE)

# Using the summary parameter to list ID numbers (truncated to 40 characters) in specified nodes
vtree(FakeData,"Severity Sex",summary="id \nid = %list% %var=Severity% %trunc=40%")

# Adding text to specified nodes of a tree ("targeted text")
vtree(FakeData,"Severity Sex",ttext=list(
  c(Severity="Severe",Sex="M",text="\nMales with Severe disease"),
  c(Severity="NA",text="\nUnknown severity")))

```

vtreeOutput

vtree widget

Description

Shiny bindings for vtree. It is actually a wrapper around [grViz](#).

Usage

```
vtreeOutput(outputId, width = "100%", height = "100%")
```

Arguments

outputId	output variable to read from
width, height	must be a valid CSS unit in pixels or a number, which will be coerced to a string and have "px" appended.

See Also

[renderVtree](#)

Other Shiny Functions: [init_js\(\)](#), [inlineCssSetup\(\)](#), [renderVtree\(\)](#), [use_svgzoom\(\)](#)

Examples

```

## Not run:
library(shiny)
library(vtree)

ui <- fluidPage(
  vtreeOutput("vtree", width = "100%", height = "800px")
)

server <- function(input, output, session) {
  output$vtree <- renderVtree({

```

```
    vtree(FakeData,"Severity Sex",
          labelnode=list(Sex=c("Male"="M","Female"="F")),
          pngknit=FALSE)
  })
}

shinyApp(ui, server)

## End(Not run)
```

Index

* Shiny Functions

- [renderVtree](#), 7
- [use_svgzoom](#), 8
- [vtreeOutput](#), 21

* datasets

- [FakeData](#), 4
- [FakeRCT](#), 5

[build.data.frame](#), 3

[crosstabToCases](#), 4

[DiagrammeR](#), 18

[FakeData](#), 4

[FakeRCT](#), 5

[grViz](#), 21

[grVizToPNG](#), 6

[init_js](#), 7, 9, 21

[inlineCssSetup](#), 7, 9, 21

[renderVtree](#), 7, 9, 21

[svtree](#), 8

[use_svgzoom](#), 7, 8, 21

[VennTable](#), 10

[vtree](#), 7–9, 11

[vtree-package](#), 2

[vtreeOutput](#), 7, 9, 21