Package 'muir'

August 29, 2016

Type Package

Title Exploring Data with Tree Data Structures

Version 0.1.0

Description A simple tool allowing users to easily and dynamically explore or document a data set using a tree structure.

URL https://github.com/alforj/muir

BugReports https://github.com/alforj/muir/issues

Depends R (>= 3.1.2)

License GPL (>= 2)

LazyData true

Imports DiagrammeR (>= 0.6), dplyr (>= 0.4.1), stringr (>= 0.6.2)

Suggests htmlwidgets (>= 0.3.2)

Author Justin Alford [aut, cre]

Maintainer Justin Alford <justin.alford@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-05-02 16:46:05

R topics documented:

build_tree *Build dynamic tree using DiagrammeR and muir-generated data frame*

Description

This function allows users to easily and dynamically explore or document a dataset using a tree structure.

Usage

```
build_tree(data, tree.dir = "LR", tree.height = NULL, tree.width = NULL)
```

Arguments

data	A muir-generated data frame to be processed into a tree using DiagrammeR
tree.dir	Direction of tree graph. Use either "LR" for left-to-right, "RL" for right-to left, "TB" for top-to-bottom, or "BT" for bottom-to-top.
tree.height	Control tree height to zoom in/out on nodes. Defaults to NULL
tree.width	Control tree width to zoom in/out on nodes. Defaults to NULL

Value

An object of class `htmlwidget` (via DiagrammeR)that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

muir *Explore Datasets with Trees*

Description

This function allows users to easily and dynamically explore or document a data.frame using a tree data structure. Columns of interest in the data.frame can be provided to the function, as well as critieria for how they should be represented in discrete nodes, to generate a data tree representing those columns and filters.

Usage

```
muir(data, node.levels, node.limit = 3, level.criteria = NULL,
  label.vals = NULL, tree.dir = "LR", show.percent = TRUE,
  num.precision = 2, show.empty.child = FALSE, tree.height = -1,
  tree.width = -1)
```

Arguments

`data`	A data.frame to be explored using trees
`node.levels`	A character vector of columns from `data` that will be used to construct the tree that are provided in the order that they should appear in the tree levels. For each column, the user can add a suffix to the columnn name to indicate whether to generate nodes for all distinct values of the column in the date.frame, a specific number of values (i.e., the "Top (n)" values), and whether or not to aggregate remaining values into a separate "Other" node, or to use user-provided filter criteria for the column as provided in the `level.criteria` parameter. This does mean that the column names cannot have a ":" and must be replaced in the data.frame before being passed in to `muir` as the `data` param. Values can be provided as "colname", "colname:*", "colname:3", "colname:+", or "colname:*+". The separator character ":" and the special characters in the suffix that follow (as outlined below) indicate which approach to take for each column. • Providing just the column name itself (e.g, "hp") will return results based on the operators and values provided in the `level.criteria` parameter for that column name. See `level.criteria` for more details. • Providing the column name with an ":*" suffix (e.g., "hp:*") will return a node for all distinct values for that column up to the limit imposed by the `node.limit` value. If the number of distinct values is greater than the `node.limit`, only the top "n" values (based on number of occurences) will be returned. • Providing the column name with an ":n" suffix (e.g., "hp:3"), where `n` = a positive integer, will return a node for all distinct values for that column up to the limit imposed by the integer provided in `n`. If the number of distinct values is greater than the value provided in `n`, only the top "n" values (based on number of occurences) will be returned. • Providing the column name ending with an ":+" suffix (e.g., "hp:+") will return all the values provided in the `level.criteria` parameter for that column plus an extra node titled "Other" for that column that aggregates all the remaining values not included in the filter criteria provided in `level.criteria` for that column. • Providing a column name ending with both symbols (e.g., "hp:*+", "hp:3+") in the suffix will return a node for all distinct values for that column up to the limit imposed by either the `node.limit` or the `n` value plus an additional "Other" node aggregating any remaining values beyond the `node.limit` or `n`, if applicable. If the number of distinct values is <= the `node.limit` or `n` then the "Other" node will not be created.
`node.limit`	Numeric value. When providing a column in `node.levels` with an ":*" suffix, the `node.limit` will limit how many distinct values to actually process to prevent run-away queries and unreadable trees. The limit defaults to 3 (not including an additional 4th if requesting to provide an "Other" node as well with a ":*+" suffix). If the number of distinct values for the column is greater than the `node.limit`, the tree will include the Top "X" values based on count, where "X" = `node.limit`. If the `node.limit` is greater than the number of distinct values for the column, it will be ignored.

`level.criteria` A data.frame consisting of 4 character columns containing column names (matching – without suffixes – the columns in `node.levels` that will use the criteria in `level.criteria` to determine the filters used for each node), an operator or boolean function (e.g., "==",">", "is.na", "is.null"), a value, and a corresponding node title for the node displaying that criteria.

E.g.,"wt, ">=", "4000", "Heavy Cars"

`label.vals` Character vector of additional values to include in the node provided as a character vector. The values must take the form of dplyr `summarise` functions (as characters) and include the columns the functions should be run against (e.g., "min(hp)", "mean(hp)", etc.). If no custom suffix is added, the summary function itself will be used as the label. Similar to `node.levels` a custom suffix can be added using ":" to print a more meaningful label (e.g., "mean(hp):Avg HP"). In this example, the label printed in the node will be "Avg HP:", otherwise it would be mean_hp (note that the parens "(" and ")" are removed to be rendered in HTML without error). As with `node.levels`, the column name itself cannot have a ":" and must be replaced in the data.frame before being passed in to `muir` as the `data` param.

`tree.dir` Character. The direction the tree graph should be rendered. Defaults to "LR"

1. Use "LR" for left-to-right
2. Use "RL" for right-to left
3. Use "TB" for top-to-bottom
4. User "BT" for bottom-to-top

`show.percent` Logical. Should nodes show the percent of records represented by that node compared to the total number of records in `data`. Defaults to TRUE

`num.precision` Number of digits to print numeric label values out to

`show.empty.child`

Logical. Show a balanced tree with children nodes that are all empty or stop expanding the tree once there is a parent node that is empty. Defaults to FALSE – don't show empty children nodes

`tree.height` Numeric. Control tree height to zoom in/out on nodes. Passed to DiagrammeR as `height` param. Defaults to -1, which appears to optimize the tree size for viewing (still researching why exactly that works! :-))

`tree.width` Numberic. Control tree width to zoom in/out on nodes. Passed to DiagrammeR as `width` param. Defaults to -1, which appears to best optimize the tree size for viewing (still researching why exactly that works! :-))

Value

An object of class `htmlwidget` (via DiagrammeR) that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

Examples

```
## Not run:
# Load in the 'mtcars' dataset
```

```
data(mtcars)

# Basic exploration - show all values
mtTree <- muir(data = mtcars, node.levels = c("cyl:*", "carb:*"))
mtTree

# Basic exploration - show all values overriding default node.limit
mtTree <- muir(data = mtcars, node.levels = c("cyl:*", "carb:*"), node.limit = 5)
mtTree

# Show all values overriding default node.limit differently for each column
mtTree <- muir(data = mtcars, node.levels = c("cyl:2", "carb:5"))
mtTree

# Show all values overriding default node.limit for each column
# and aggregating all distinct values above the node.limit into a
# separate "Other" column to collect remaining values

# Top 2 occurring 'carb' values will be returned in their own nodes,
# remaining values/counts will be aggregated into a separate "Other" node
mtTree <- muir(data = mtcars, node.levels = c("cyl:2", "carb:2+"))
mtTree

# Add additional calculations to each node output (dplyr::summarise functions)
mtTree <- muir(data = mtcars, node.levels = c("cyl:2", "carb:2+"),
label.vals = c("min(wt)", "max(wt)"))
mtTree

# Make new label values more reader-friendly
mtTree <- muir(data = mtcars, node.levels = c("cyl:2", "carb:2+"),
label.vals = c("min(wt):Min Weight", "max(wt):Max Weight"))
mtTree

# Instead of just returning top counts for columns provided in \code{node.levels},
# provide custom filter criteria and custom node titles in \code{label.vals}
# (criteria could also be read in from a csv file as a data.frame)
criteria <- data.frame(col = c("cyl", "cyl", "carb"),
oper = c("<", ">=", "=="),
val = c(4, 4, 2),
title = c("Less Than 4 Cylinders", "4 or More Cylinders", "2 Carburetors"))

mtTree <- muir(data = mtcars, node.levels = c("cyl", "carb"),
level.criteria = criteria,
label.vals = c("min(wt):Min Weight", "max(wt):Max Weight"))
mtTree

# Use same criteria but show all other values for the column where NOT
# EQUAL to the combination of the filters provided for that column (e.g., for cyl
# where !(cyl < 4 | cyl >= 4) in an "Other" node
mtTree <- muir(data = mtcars, node.levels = c("cyl:+", "carb:+"),
level.criteria = criteria,
label.vals = c("min(wt):Min Weight", "max(wt):Max Weight"))
mtTree
```

```
# Show empty child nodes (balanced tree)
mtTree <- muir(data = mtcars, node.levels = c("cyl:+", "carb:+"),
level.criteria = criteria,
label.vals = c("min(wt):Min Weight", "max(wt):Max Weight"),
show.empty.child = TRUE)
mtTree

# Save tree to HTML file with \code{htmlwidgets} package to working directory
mtTree <- muir(data = mtcars, node.levels = c("cyl:2", "carb:2+"))
htmlwidgets::saveWidget(mtTree, "mtTree.html")

## End(Not run)
```

Index