rpivotTable

Enzo Martoglio

2018-01-30

rpivotTable: A pivot table for R

The rpivotTable package is an R htmlwidget built around the pivottable library.

PivotTable.js is a Javascript Pivot Table visualization library with drag'n'drop functionality built on top of jQuery / jQueryUI and written in CoffeeScript (then compiled to JavaScript) by Nicolas Kruchten at Datacratic. It is available under a MIT license

Many thanks to everyone that contributes bugs and prs, and of course thanks to Nicolas Kruchten for PivotTable.js.

Installation

The rpivotTable package depends on htmlwidgets package, so you need to install both packages. You can do this using the **devtools** package as follows:

```
# devtools::install_github(c("ramnathv/htmlwidgets", "smartinsightsfromdata/rpivotTable"))
```

Or directly from CRAN:

```
# install.packages('htmlwidgets', 'rpivotTable')
```

Usage

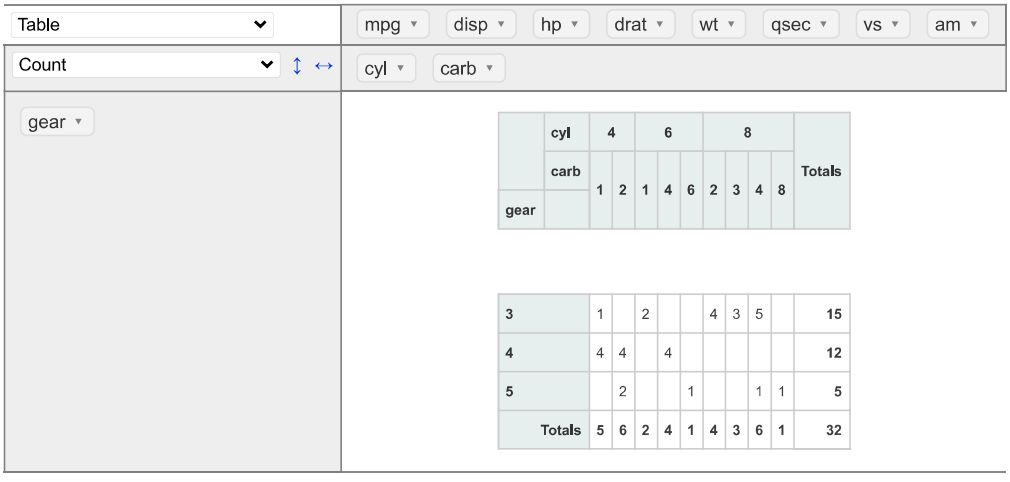
Call the package with

```
library(rpivotTable)  # No need to explicitly load htmlwidgets: this is done automatically
```

Just plug in your `data.frame`, `table` or `data.table` to `rpivotTable()`.

It is as simple as this:

```
data(mtcars)
rpivotTable(mtcars,rows="gear", cols=c("cyl","carb"),width="100%", height="400px")
```



The pivot table should appear in your RStudio Viewer or your browser of choice.

For additional technical information please refer to the examples and explanations here.

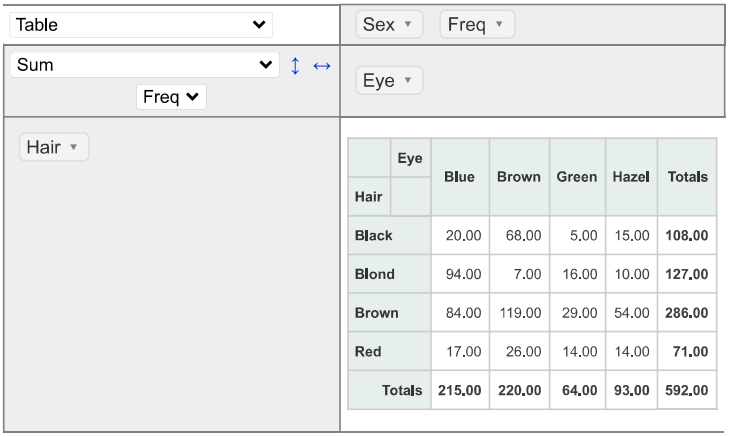
`rpivotTable` parameters decide how the pivot table will look like the firs time it is opened:

- `data` can be a `data.frame` `table` or `data.table`. Nothing else is needed. If only the data is selected the pivot table opens with nothing on rows and columns (but you can at any time drag and drop any variable in rows or columns at your leasure)
- `rows` and `cols` allow the user to create a report, i.e. to indicate which element will be on rows and columns.
- `aggregatorName` indicates the type of aggregation. Options here are numerous: Count, Count Unique Values, List Unique Values, Sum, Integer Sum, Average, Sum over Sum, 80% Upper Bound, 80% Lower

Bound, Sum as Fraction of Total, Sum as Fraction of Rows, Sum as Fraction of Columns, Count as Fraction of Total, Count as Fraction of Rows, Count as Fraction of Columns
- `vals` specifies the variable to use with `aggregatorName` (if any).
- `renderers` dictates the type of graphic rendering used for display, like Table, Treemap etc.
- `sorters` allow to implement a javascript function to specify the ad hoc sorting of certain values. See vignette for an example. It is especially useful with time divisions like days of the week or months of the year (where the alphabetical order does not work)

For example, to display a table with frequency of colour combinations of eyes and hair, you can specify:

```
library(rpivotTable)
data(HairEyeColor)
rpivotTable(data = HairEyeColor, rows = "Hair",cols="Eye", vals = "Freq", aggregatorName = "Sum",
  rendererName = "Table", width="100%", height="400px")
```

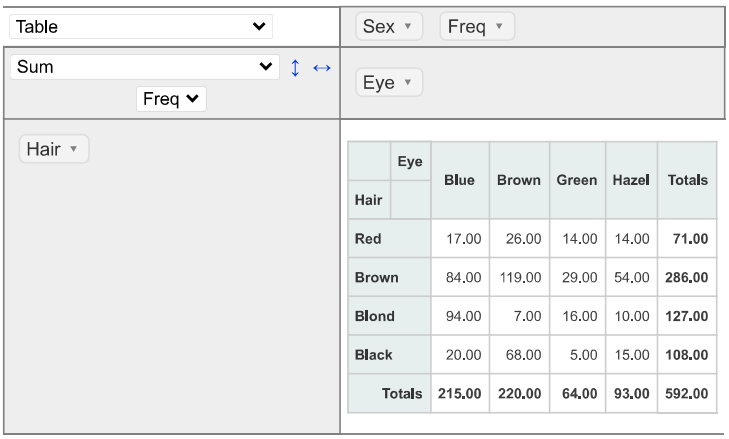


Hair \ Eye	Blue	Brown	Green	Hazel	Totals
Black	20.00	68.00	5.00	15.00	108.00
Blond	94.00	7.00	16.00	10.00	127.00
Brown	84.00	119.00	29.00	54.00	286.00
Red	17.00	26.00	14.00	14.00	71.00
Totals	215.00	220.00	64.00	93.00	592.00

This will display the resulting table. Switching the `aggregatorName` to `Sum as Fraction of Rows` will give the row percentages (and the column totals will give the percentages over the gran total).

To display the `Hair` values in reverse order:

```
library(rpivotTable)
data(HairEyeColor)
rpivotTable(data = HairEyeColor, rows = "Hair",cols="Eye", vals = "Freq", aggregatorName = "Sum",
  rendererName = "Table", sorters = "
function(attr) {
var sortAs = $.pivotUtilities.sortAs;
if (attr == \"Hair\") { return sortAs([\"Red\", \"Brown\", \"Blond\", \"Black\"]); }
}", width="100%", height="400px")
```



Hair \ Eye	Blue	Brown	Green	Hazel	Totals
Red	17.00	26.00	14.00	14.00	71.00
Brown	84.00	119.00	29.00	54.00	286.00
Blond	94.00	7.00	16.00	10.00	127.00
Black	20.00	68.00	5.00	15.00	108.00
Totals	215.00	220.00	64.00	93.00	592.00

This function could be useful for example to sort time divisions like months of the year or days of the week in their proper, non alphabetical order (thanks to palatinuse for its implementation).

You can also use the very visual new subtotals:

```
data(mtcars)
rpivotTable(mtcars,rows="gear", cols=c("cyl","carb"),subtotals=TRUE, width="100%", height="400px")
```

Table With Subtotal | mpg | disp | hp | drat | wt | qsec | vs | am

Count | cyl | carb

gear

	cyl	4			6			8				Totals
gear	carb	1	2		1	4	6	2	3	4	8	
3		1		1	2		2	4	3	5		12	15
4		4	4	8		4	4						12
5			2	2			1	1			1	1	2	5
Totals		5	6	11	2	4	1	7	4	3	6	1	14	32

Or if you want to include it as part of your `dplyr` / `magrittr` pipeline, you can do that also.

```
# suppressMessages(
  library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

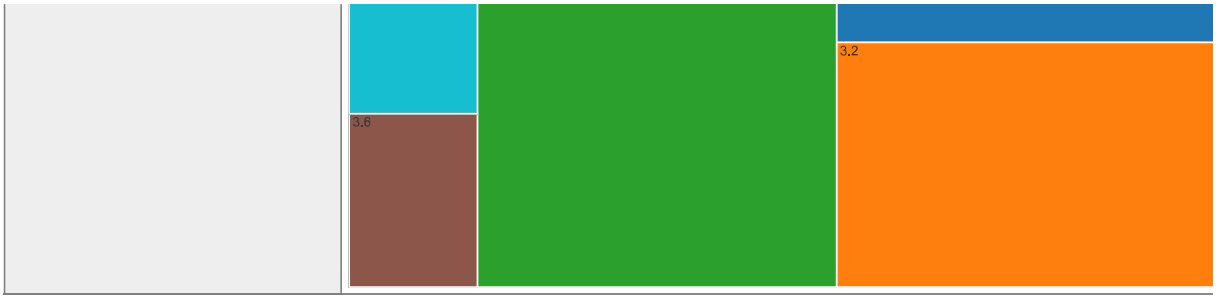
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
  # )
iris %>%
tbl_df %>%
filter( Sepal.Width > 3 & Sepal.Length > 5 ) %>%
rpivotTable(rows="Sepal.Width",  rendererName="Treemap")
```

Treemap | Sepal.Length | Petal.Length | Petal.Width | Species

Count

Sepal.Width


3.6
3.2