DT: An R interface to the DataTables library

The R package **DT** provides an R interface to the JavaScript library **DataTables** (http://datatables.net). R data objects (matrices or data frames) can be displayed as tables on HTML pages, and **DataTables** provides filtering, pagination, sorting, and many other features in the tables.

You may install the stable version from CRAN, or the development version using `devtools::install_github('rstudio/DT')` if necessary (this website reflects the development version of **DT**):

```
if (!require("DT")) install.packages('DT')
sessionInfo()
```

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 15.04
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] DT_0.1.39
##
## loaded via a namespace (and not attached):
##  [1] htmlwidgets_0.5 magrittr_1.5    formatR_1.2.2   tools_3.2.2
##  [5] htmltools_0.2.9 yaml_2.1.13     stringi_0.5-5   rmarkdown_0.8.1
##  [9] knitr_1.11.6    methods_3.2.2   stringr_1.0.0   digest_0.6.8
## [13] evaluate_0.8
```

Please use Github issues (https://github.com/rstudio/DT/issues) if you want to file bug reports or feature requests, and you may use StackOverflow (http://stackoverflow.com/questions/tagged/r+datatables) or the shiny-discuss (https://groups.google.com/forum/#!forum/shiny-discuss) mailing list to ask questions.

1 Usage

The main function in this package is `datatable()`. It creates an HTML widget to display R data objects with **DataTables**.

```
datatable(data, options = list(), class = "display", callback = JS("return table;"),
    rownames, colnames, container, caption = NULL, filter = c("none", "bottom",
        "top"), escape = TRUE, style = "default", width = "100%", selection = c("multiple",
        "single", "none"), extensions = list(), plugins = NULL)
```

Here is a "hello world" example with zero configuration:

```
library(DT)
datatable(iris)
```

Show 10 ⬍ entries   Search: [   ]

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

2 Arguments

If you are familiar with **DataTables** already, you may use the `options` argument to customize the table. See the page Options (options.html) for details. Here we explain the rest of the arguments of the `datatable()` function.

2.1 Table CSS Classes

The `class` argument specifies the CSS classes of the table. The possible values can be found on the page of default styling options (http://datatables.net/manual/styling/classes). The default value `display` basically enables row striping, row highlighting on mouse over, row borders, and highlighting ordered columns. You can choose a different combination of CSS classes, such as `cell-border` and `stripe`:

```
datatable(head(iris), class = 'cell-border stripe')
```

Show 10 ⬍ entries   Search: _____

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

2.2 Styling

Currently, **DT** only supports the Bootstrap style besides the default style. You can use the argument `style = 'bootstrap'` to enable the Bootstrap style, and adjust the table classes accordingly using Bootstrap table class names (http://getbootstrap.com/css/#tables), such as `table-stripe` and `table-hover`. Actually, **DT** will automatically adjust the class names even if you provided the DataTables class names such as `stripe` and `hover`.

```
DT:::DT2BSClass('display')
## [1] "table table-striped table-hover"
DT:::DT2BSClass(c('compact', 'cell-border'))
## [1] "table table-condensed table-bordered"
```

Note you can only use one style for all tables on one page. Please see this separate page (005-bootstrap.html) for examples using the Bootstrap style.

2.3 Display Row Names

If the data object has row names, they will be displayed as the first column of the table by default. You can suppress row names via the argument `rownames = FALSE`, and you can also change row names by providing a different character vector to `rownames`.

```
datatable(head(mtcars))
```

Show 10 ⬍ entries   Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1	4	4

Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

Showing 1 to 6 of 6 entries   Previous   1   Next

```
datatable(head(mtcars), rownames = FALSE)   # no row names
```

Show 10 ⬍ entries   Search:

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21	6	160	110	3.9	2.62	16.46	0	1	4	4
21	6	160	110	3.9	2.875	17.02	0	1	4	4
22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

```
datatable(head(mtcars), rownames = head(LETTERS))  # new row names
```

Show 10 ▲▼ entries   Search: _____

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
A	21	6	160	110	3.9	2.62	16.46	0	1	4	4
B	21	6	160	110	3.9	2.875	17.02	0	1	4	4
C	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1
D	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
E	18.7	8	360	175	3.15	3.44	17.02	0	0	3	2
F	18.1	6	225	105	2.76	3.46	20.22	1	0	3	1

Showing 1 to 6 of 6 entries   Previous 1 Next

Influence of Row Names on Column Indices in JavaScript

Row names are essentialy a new column added to the original data (via `cbind(rownames(data), data)`). This has an important consequence in terms of the column indices. JavaScript indexes from 0 instead of 1, so the index of the n-th element is actually $n - 1$.[1] When thinking of the column indices (which you will often have to do if you customize options (options.html)), use

- $n - 1$ as the index of the n-th column in the *original data* if you do not display row names;
- n as the index of the n-th column in the *original data* if you want to display row names, because the original index is $n - 1$ in JavaScript but we added the row names as the first column, and $(n - 1) + 1 = n$;

It is very important to remember this when using DataTables options (options.html).

2.4 Custom Column Names

By default, `datatable()` shows the column names of the data in the table, and you can use a custom character vector for the table header. There are a few possibilities. The first one is, you provide a new character vector to completely replace the column names of the data, e.g.

```
# colnames(iris) is a character vector of length 5, and we replace it
datatable(head(iris), colnames = c('Here', 'Are', 'Some', 'New', 'Names'))
```

Show 10 ⬍ entries   Search: [   ]

	Here	Are	Some	New	Names
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Showing 1 to 6 of 6 entries   Previous   1   Next

This can be cumbersome if you only want to replace one or two names, and you do not want to provide a whole vector of names. Then here is the second possibility: you can provide a shorter numeric or character vector as the index vector to replace a subset of the column names. For example, if you only want the 2nd name to be `'A Nicer Name'`, you can use `datatable(..., colnames = c('A Nicer Name' =`

2)); or if you want to replace the name `'X5'` with `'A Better Name'`, you can use `colnames = c('A Better Name' = 'X5')`.

```
datatable(head(iris), colnames = c('A Better Name' = 'Sepal.Width'))
```

Show [10 ⬍] entries   Search: [   ]

	Sepal.Length	A Better Name	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Showing 1 to 6 of 6 entries   Previous [1] Next

```
datatable(head(iris), colnames = c('Another Better Name' = 2, 'Yet Another Name' = 4))
```

Show [10 ⬍] entries   Search: [   ]

	Another Better Name	Sepal.Width	Yet Another Name	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa

3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Showing 1 to 6 of 6 entries   Previous 1 Next

When you display row names of the data, its column name will be a white space by default. That is why you cannot see its column name. You can certainly choose to use a column name for rownames as well, e.g.

```
# change the first column name to 'ID'
datatable(head(iris), colnames = c('ID' = 1))
```

Show 10 entries   Search:

ID	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

2.5 Custom Table Container

The `container` argument allows you to provide a different table container to hold the table cells. By default, the container is generated from the column names. Below is an example of a custom table header:

```r
# a custom table container
sketch = htmltools::withTags(table(
  class = 'display',
  thead(
    tr(
      th(rowspan = 2, 'Species'),
      th(colspan = 2, 'Sepal'),
      th(colspan = 2, 'Petal')
    ),
    tr(
      lapply(rep(c('Length', 'Width'), 2), th)
    )
  )
))
print(sketch)
```

```
<table class="display">
  <thead>
    <tr>
      <th rowspan="2">Species</th>
      <th colspan="2">Sepal</th>
      <th colspan="2">Petal</th>
    </tr>
    <tr>
      <th>Length</th>
      <th>Width</th>
      <th>Length</th>
      <th>Width</th>
    </tr>
  </thead>
</table>
```

```
# use rownames = FALSE here because we did not generate a cell for row names in
# the header, and the header only contains five columns
datatable(iris[1:20, c(5, 1:4)], container = sketch, rownames = FALSE)
```

Show 10 ▲▼ entries   Search: [   ]

| Species | Sepal | | Petal | |
	Length	Width	Length	Width
setosa	5.1	3.5	1.4	0.2
setosa	4.9	3	1.4	0.2
setosa	4.7	3.2	1.3	0.2

setosa	4.6	3.1	1.5	0.2
setosa	5	3.6	1.4	0.2
setosa	5.4	3.9	1.7	0.4
setosa	4.6	3.4	1.4	0.3
setosa	5	3.4	1.5	0.2
setosa	4.4	2.9	1.4	0.2
setosa	4.9	3.1	1.5	0.1

Showing 1 to 10 of 20 entries   Previous 1 2 Next

You can also add a footer to the table container, and here is an example:

```
# a custom table with both header and footer
sketch = htmltools::withTags(table(
  tableHeader(iris),
  tableFooter(iris)
))
print(sketch)
```

```
<table>
  <thead>
    <tr>
      <th>Sepal.Length</th>
      <th>Sepal.Width</th>
      <th>Petal.Length</th>
      <th>Petal.Width</th>
      <th>Species</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th>Sepal.Length</th>
      <th>Sepal.Width</th>
      <th>Petal.Length</th>
      <th>Petal.Width</th>
      <th>Species</th>
    </tr>
  </tfoot>
</table>
```

```
datatable(
  head(iris, 10),
  container = sketch, options = list(pageLength = 5, dom = 'tip'), rownames = FALSE
)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa

Showing 1 to 5 of 10 entries

Previous 1 2 Next

2.6 Table Caption

You can add a table caption via the `caption` argument. It can be either a character vector, or a tag object created from `htmltools::tags$caption()`. See this blog post (http://datatables.net/blog/2014-11-07) for more information on table captions.

```
datatable(
  head(iris),
  caption = 'Table 1: This is a simple caption for the table.'
)
```

Show 10 entries   Search: _____

Table 1: This is a simple caption for the table.

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Showing 1 to 6 of 6 entries   Previous   1   Next

```r
# display the caption at the bottom, and <em> the caption
datatable(
  head(iris),
  caption = htmltools::tags$caption(
    style = 'caption-side: bottom; text-align: center;',
    'Table 2: ', htmltools::em('This is a simple caption for the table.')
  )
)
```

Show 10 entries   Search:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa

6	5.4	3.9	1.7	0.4 setosa

Table 2: *This is a simple caption for the table.*

Showing 1 to 6 of 6 entries   Previous | 1 | Next

2.7 Column Filters

DataTables does not provide column filters by default. There is only a global filter (the search box on the top-right). We added a `filter` argument in `datatable()` to automatically generate column filters. By default, the filters are not shown since `filter = 'none'`. You can enable these filters by `filter = 'top'` or `'bottom'`, depending on whether you want to put the filters on the top or bottom of the table.

```
iris2 = iris[c(1:10, 51:60, 101:110), ]
datatable(iris2, filter = 'top', options = list(
  pageLength = 5, autoWidth = TRUE
))
```

Show 5 entries   Search:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
All	All	All	All	All
1	5.1	3.5	1.4	0.2 setosa
2	4.9	3	1.4	0.2 setosa
3	4.7	3.2	1.3	0.2 setosa
4	4.6	3.1	1.5	0.2 setosa

5	5	3.6	1.4	0.2 setosa

Showing 1 to 5 of 30 entries   Previous   **1**   2   3   4   5   6   Next

Depending on the type of a column, the filter control can be different. Initially, you see search boxes for all columns. When you click the search boxes, you may see different controls:

- For numeric/date/time columns, range sliders (http://refreshless.com/nouislider/) are used to filter rows within ranges;
- For factor columns, selectize inputs (http://brianreavis.github.io/selectize.js/) are used to display all possible categories, and you can select multiple categories there (note you can also type in the box to search in all categories);
- For character columns, ordinary search boxes are used to match the values you typed in the boxes;

When you leave the initial search boxes, the controls will be hidden and the filtering values (if there are any) are stored in the boxes:

- For numeric/date/time columns, the values displayed in the boxes are of the form `low ... high`;
- For factor columns, the values are serialized as a JSON array of the form `["value1", "value2", "value3"]`;

When a column is filtered, there will be a clear button ⊗ in its search box, and you can click the button to clear the filter. If you do not want to use the controls, you can actually type in the search boxes directly, e.g. you may type `2 ... 5` to filter a numeric column, and the range of its slider will automatically adjusted to `[2, 5]`. In case you find a search box too narrow and it is difficult to read the values in it, you may mouse over the box and its values will be displayed as a tooltip. See this example (008-filter.html) for how to hide the clear buttons, and use plain text input styles instead of Bootstrap.

Below is a simple example to demonstrate filters for character, date, and time columns:

```
d = data.frame(
  names = rownames(mtcars),
  date = as.Date('2015-03-23') + 1:32,
  time = as.POSIXct('2015-03-23 12:00:00', tz = 'UTC') + (1:32) * 5000,
  stringsAsFactors = FALSE
)
str(d)
## 'data.frame':    32 obs. of  3 variables:
##  $ names: chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ date : Date, format: "2015-03-24" "2015-03-25" ...
##  $ time : POSIXct, format: "2015-03-23 13:23:20" "2015-03-23 14:46:40" ...
datatable(d, filter = 'bottom', options = list(pageLength = 5))
```

Show 5 ⬍ entries   Search: [   ]

	names	date	time
1	Mazda RX4	2015-03-24	2015-03-23T13:23:20Z
2	Mazda RX4 Wag	2015-03-25	2015-03-23T14:46:40Z
3	Datsun 710	2015-03-26	2015-03-23T16:10:00Z
4	Hornet 4 Drive	2015-03-27	2015-03-23T17:33:20Z
5	Hornet Sportabout	2015-03-28	2015-03-23T18:56:40Z
	All	All	All

Showing 1 to 5 of 32 entries   Previous 1 2 3 4 5 6 7 Next

Filtering in the above examples was done on the client side (using JavaScript in your web browser). Column filters also work in the server-side processing (server.html) mode, in which case filtering will be processed on the server, and there may be some subtle differences (e.g. JavaScript regular expressions are different with R). See here for an example (https://yihui.shinyapps.io/DT-filter/) of column filters working on the server side.

Known Issues of Column Filters
The position of column filters may be off when scrolling is enabled in the table, e.g. via the options `scrollX` and/or `scrollY`. The appearance may be affected by Shiny sliders, as reported in #49 (https://github.com/rstudio/DT/issues/49).

2.8 The `callback` argument

The argument `callback` takes the body of a JavaScript function that will be applied to the **DataTables** object after initialization. Below is an example to show the next page after the table is initialized[2]:

```
datatable(head(iris, 30), callback = JS('table.page("next").draw(false);'))
```

Show 10 entries   Search:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3	1.4	0.1	setosa
14	4.3	3	1.1	0.1	setosa
15	5.8	4	1.2	0.2	setosa

16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa

Showing 11 to 20 of 30 entries   Previous   1   2   3   Next

In the above example, the actual callback function on the JavaScript side is this (`callback` is only the body of the function):

```
function(table) {
  table.page("next").draw(false);
}
```

After we initialize the table via the `.DataTable()` method in DataTables, the DataTables instance is passed to this callback function. Below are a few more examples:

- Show extra information in child rows (002-rowdetails.html)

Please note this `callback` argument is only an argument of the `datatable()` function, and do not confuse it with the callbacks in the **DataTables** options (options.html). The purpose of this argument is to allow users to manipulate the **DataTables** object after its creation.

2.9 Escaping Table Content

The argument `escape` determines whether the HTML entities in the table are escaped or not. There can be potential security problems when the table is rendered in dynamic web applications such as Shiny if you do not escape them. Here is a quick example:

```
m = matrix(c(
  '<b>Bold</b>', '<em>Emphasize</em>', '<a href="http://rstudio.com">RStudio</a>',
  '<a href="#" onclick="alert(\'Hello World\');">Hello</a>'
), 2)
colnames(m) = c('<span style="color:red">Column 1</span>', '<em>Column 2</em>')
datatable(m)  # escape = TRUE by default
```

Show [10 ⬍] entries   Search: [   ]

\Column 1\	\Column 2\
\Bold\	\RStudio\
\Emphasize\	\Hello\

Showing 1 to 2 of 2 entries   Previous [1] Next

```
datatable(m, escape = FALSE)
```

Show [10 ⬍] entries   Search: [   ]

Column 1	*Column 2*
Bold	RStudio (http://rstudio.com)
Emphasize	Hello

Showing 1 to 2 of 2 entries   Previous [1] Next

Besides `TRUE` and `FALSE`, you can also specify which columns you want to escape, e.g.

```r
datatable(m, escape = 1)   # escape the first column
datatable(m, escape = 2)   # escape the second column
datatable(m, escape = c(TRUE, FALSE))   # escape the first column
colnames(m) = c('V1', 'V2')
datatable(m, escape = 'V1')
```

1. By comparison, R indexes from 1.↵

2. See the documentation for the page() (http://datatables.net/reference/api/page()) API.↵