Package 'flexdashboard'

October 13, 2022

Type Package

Title R Markdown Format for Flexible Dashboards

Version 0.6.0

Description Format for converting an R Markdown document to a grid oriented dashboard. The dashboard flexibly adapts the size of it's components to the containing web page.

URL https://pkgs.rstudio.com/flexdashboard/, https://github.com/rstudio/flexdashboard/

BugReports https://github.com/rstudio/flexdashboard/issues

Encoding UTF-8

Depends R (>= 3.0.2)

Imports grDevices, tools, utils, jsonlite, htmltools (>= 0.5.1), knitr (>= 1.13), htmlwidgets (>= 0.6), rmarkdown (>= 2.8), shiny (>= 0.13), scales, sass, bslib (>= 0.2.5)

Suggests testthat

License MIT + file LICENSE

RoxygenNote 7.2.1

Config/testthat/edition 3

Config/Needs/website rstudio/quillt

NeedsCompilation no

Author Carson Sievert [aut, cre] (<https://orcid.org/0000-0002-4958-2844>),
Richard Iannone [aut] (<https://orcid.org/0000-0003-3925-190X>),
JJ Allaire [aut],
Barbara Borges [aut],
RStudio [cph],
Keen IO [ctb, cph] (Dashboard CSS),
Abdullah Almsaeed [ctb, cph] (Dashboard CSS),
Jonas Mosbech [ctb, cph] (StickyTableHeaders),
Noel Bossart [ctb, cph] (Featherlight),
Lea Verou [ctb, cph] (Prism),

Dmitry Baranovskiy [ctb, cph] (Raphael.js),
Sencha Labs [ctb, cph] (Raphael.js),
Bojan Djuricic [ctb, cph] (JustGage),
Tomas Sardyha [ctb, cph] (Sly),
Bryan Lewis [ctb, cph] (Examples),
Joshua Kunst [ctb, cph] (Examples),
Ryan Hafen [ctb, cph] (Examples),
Bob Rudis [ctb, cph] (Examples),
Joe Cheng [ctb] (Examples)

Maintainer Carson Sievert <carson@rstudio.com>

Repository CRAN

Date/Publication 2022-08-05 22:50:10 UTC

R topics documented:

flexdashboard-package 2
flex_dashboard 4
gauge 7
gauge-shiny 8
valueBox 9
valueBox-shiny 10

Index **11**

`flexdashboard-package` *flexdashboard: Interactive dashboards for R*

Description

Create interactive dashboards using **rmarkdown**.

Details

- Use R Markdown to publish a group of related data visualizations as a dashboard.
- Ideal for publishing interactive JavaScript visualizations based on htmlwidgets (also works with standard base, lattice, and grid graphics).
- Flexible and easy to specify layouts. Charts are intelligently re-sized to fill the browser and adapted for display on mobile devices.
- Optionally use Shiny to drive visualizations dynamically.

See the flexdashboard website for additional documentation: https://pkgs.rstudio.com/flexdashboard/

Author(s)

Maintainer: Carson Sievert <carson@rstudio.com> (ORCID)

Authors:

- Richard Iannone <rich@rstudio.com> (ORCID)
- JJ Allaire <jj@rstudio.com>
- Barbara Borges <barb.b.ribeiro@gmail.com>

Other contributors:

- RStudio [copyright holder]
- Keen IO (Dashboard CSS) [contributor, copyright holder]
- Abdullah Almsaeed (Dashboard CSS) [contributor, copyright holder]
- Jonas Mosbech (StickyTableHeaders) [contributor, copyright holder]
- Noel Bossart (Featherlight) [contributor, copyright holder]
- Lea Verou (Prism) [contributor, copyright holder]
- Dmitry Baranovskiy (Raphael.js) [contributor, copyright holder]
- Sencha Labs (Raphael.js) [contributor, copyright holder]
- Bojan Djuricic (JustGage) [contributor, copyright holder]
- Tomas Sardyha (Sly) [contributor, copyright holder]
- Bryan Lewis (Examples) [contributor, copyright holder]
- Joshua Kunst (Examples) [contributor, copyright holder]
- Ryan Hafen (Examples) [contributor, copyright holder]
- Bob Rudis (Examples) [contributor, copyright holder]
- Joe Cheng (Examples) [contributor]

See Also

Useful links:

- https://pkgs.rstudio.com/flexdashboard/
- https://github.com/rstudio/flexdashboard/
- Report bugs at https://github.com/rstudio/flexdashboard/issues

flex_dashboard *R Markdown Format for Flexible Dashboards*

Description

Format for converting an R Markdown document to a grid oriented dashboard layout. The dashboard flexibly adapts the size of it's plots and htmlwidgets to its containing web page.

Usage

```
flex_dashboard(
  fig_width = 6,
  fig_height = 4.8,
  fig_retina = 2,
  fig_mobile = TRUE,
  dev = "png",
  self_contained = TRUE,
  favicon = NULL,
  logo = NULL,
  social = NULL,
  source_code = NULL,
  navbar = NULL,
  orientation = c("columns", "rows"),
  vertical_layout = c("fill", "scroll"),
  storyboard = FALSE,
  theme = "default",
  highlight = "default",
  mathjax = "default",
  extra_dependencies = NULL,
  css = NULL,
  includes = NULL,
  lib_dir = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  devel = FALSE,
  resize_reload = TRUE,
  ...
)
```

Arguments

fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2). Note that for flexdashboard enabling retina scaling provides for both crisper graphics on retina screens but also much higher quality auto-scaling of R graphics within flexdashboard containers.

`fig_mobile` Create an additional rendering of each R graphics figure optimized for rendering on mobile devices oriented in portrait mode. If `TRUE`, creates a figure which is 3.75 x 4.80 inches wide; if `FALSE`, create no additional figure for mobile devices; if a numeric vector of length 2, creates a mobile figure with the specified width and height.

`dev` Graphics device to use for figure output (defaults to png)

`self_contained` Produce a standalone HTML file with no external dependencies, using data: URIs to incorporate the contents of linked scripts, stylesheets, images, and videos. Note that even for self contained documents MathJax is still loaded externally (this is necessary because of its size).

`favicon` Path to graphic to be used as a favicon for the dashboard. Pass `NULL` to use no favicon.

`logo` Path to graphic to be used as a logo for the dashboard. Pass `NULL` to not include a logo. Note that no scaling is performed on the logo image, so it should fit exactly within the dimensions of the navigation bar (48 pixels high for the default "cosmo" theme, other themes may have slightly different navigation bar heights).

`social` Specify a character vector of social sharing services to automatically add sharing links for them on the `navbar`. Valid values are "twitter", "facebook", "linkedin", and "pinterest" (more than one service can be specified).

`source_code` URL for source code of dashboard (used primarily for publishing flexdashboard examples). Automatically creates a `navbar` item which links to the source code.

`navbar` Optional list of elements to be placed on the flexdashboard navigation bar. Each element should be a list containing a `title` and/or `icon` field, an `href` field. Optional fields `target` (e.g. "_blank") and `align` ("left" or "right") are also supported.

`orientation` Determines whether level 2 headings are treated as dashboard rows or dashboard columns.

`vertical_layout`

Vertical layout behavior: "fill" to vertically resize charts so they completely fill the page; "scroll" to layout charts at their natural height, scrolling the page if necessary.

`storyboard` `TRUE` to use a storyboard layout scheme that places each dashboard component in a navigable storyboard frame. When a storyboard layout is used the `orientation` and `vertical_layout` arguments are ignored. When creating a dashbaord with multiple pages you should apply the ‘.storyboard‘ attribute to individual pages rather than using the global `storyboard` option.

`theme` One of the following: * A [bslib::bs_theme()] object (or a list of [bslib::bs_theme()] argument values) * Use this option to choose any [Bootstrap version](https://rstudio.github.io/bslib/articles versions), [Bootswatch theme](https://rstudio.github.io/bslib/articles/bslib.html#bootswatch-themes), or implement a [custom theme](https://rstudio.github.io/bslib/articles/bslib.html#custom-themes). * In this case, any ‘.scss‘/‘.sass‘ files provided to the ‘css‘ parameter may utilize the ‘theme‘’s underlying Sass utilities (e.g., variables, mixins, etc). * A character string specifying a [Bootswatch 3](https://bootswatch.com/3/) theme name (for backwards-compatibility). The "cosmo" theme is used when "default" is specified.

highlight	Syntax highlighting style. Supported styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", and "haddock". Pass NULL to prevent syntax highlighting.
mathjax	Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass NULL to exclude MathJax entirely.
extra_dependencies	Extra dependencies as a list of the html_dependency class objects typically generated by htmltools::htmlDependency().
css	CSS and/or Sass files to include. Files with an extension of .sass or .scss are compiled to CSS via sass::sass(). Also, if theme is a bslib::bs_theme() object, Sass code may reference the relevant Bootstrap Sass variables, functions, mixins, etc.
includes	Named list of additional content to include within the document (typically created using the includes function).
lib_dir	Directory to copy dependent HTML libraries (e.g. jquery, bootstrap, etc.) into. By default this will be the name of the document with _files appended to it.
md_extensions	Markdown extensions to be added or removed from the default definition of R Markdown. See the rmarkdown_format for additional details.
pandoc_args	Additional command line options to pass to pandoc
devel	Enable development mode (used for development of the format itself, not useful for users of the format).
resize_reload	Disable the auto-reloading behavior when the window is resized. Useful when debugging large flexdashboard applications and this functionality is not needed.
...	Other arguments to [rmarkdown::html_document_base()].

Details

See the flexdashboard website for additional documentation: https://pkgs.rstudio.com/flexdashboard/

Examples

```
## Not run:

library(rmarkdown)
library(flexdashboard)

# simple invocation
render("dashboard.Rmd", flex_dashboard())

# specify the theme option
render("pres.Rmd", flex_dashboard(theme = "yeti"))

## End(Not run)
```

gauge *Create a gauge component for a dashboard.*

Description

A gauge displays a numeric value on a meter that runs between specified minimum and maximum values.

Usage

```
gauge(
  value,
  min,
  max,
  sectors = gaugeSectors(),
  symbol = NULL,
  label = NULL,
  abbreviate = TRUE,
  abbreviateDecimals = 1,
  href = NULL
)

gaugeSectors(
  success = NULL,
  warning = NULL,
  danger = NULL,
  colors = c("success", "warning", "danger")
)
```

Arguments

value	Numeric value to display
min	Minimum numeric value
max	Maximum numeric value
sectors	Custom colored sectors (e.g. "success", "warning", "danger"). By default all values are colored using the "success" theme color
symbol	Optional symbol to show next to value (e.g. 'kg')
label	Optional label to display beneath the value
abbreviate	Abbreviate large numbers for min, max, and value (e.g. 1234567 -> 1.23M). Defaults to TRUE.
abbreviateDecimals	Number of decimal places for abbreviated numbers to contain (defaults to 1).
href	An optional URL to link to. Note that this can be an anchor of another dashboard page (e.g. "#details").

success	Two-element numeric vector defining the range of values to color as "success" (specific color provided by theme or custom colors)
warning	Two-element numeric vector defining the range of values to color as "warning" (specific color provided by theme or custom colors)
danger	Two-element numeric vector defining the range of values to color as "danger" (specific color provided by theme or custom colors)
colors	Vector of colors to use for the success, warning, and danger ranges. Colors can be standard theme colors ("success", "warning", "danger", "primary", and "info") or any other valid CSS color specifier. Note that if no custom sector ranges are defined, this parameter can be a single color value rather than a vector of three values

Details

See the flexdashboard website for additional documentation: <https://pkgs.rstudio.com/flexdashboard/articles/using.html#gau

Examples

```
library(flexdashboard)

gauge(42, min = 0, max = 100, symbol = '%', gaugeSectors(
  success = c(80, 100), warning = c(40, 79), danger = c(0, 39)
))
```

---

gauge-shiny *Shiny bindings for gauge*

Description

Output and render functions for using gauge within Shiny applications and interactive Rmd documents.

Usage

```
gaugeOutput(outputId, width = "100%", height = "200px")

renderGauge(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a gauge
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

valueBox *Create a value box component for a dashboard.*

Description

A value box displays a value (usually a number) in large text, with a smaller caption beneath, and a large icon on the right side.

Usage

```
valueBox(value, caption = NULL, icon = NULL, color = NULL, href = NULL)
```

Arguments

value	The value to display in the box. Usually a number or short text.
caption	The caption to display beneath the value.
icon	An icon for the box (e.g. "fa-comments")
color	Background color for the box. This can be one of the built-in background colors ("primary", "info", "success", "warning", "danger") or any valid CSS color value.
href	An optional URL to link to. Note that this can be an anchor of another dashboard page (e.g. "#details").

Details

See the flexdashboard website for additional documentation: <https://pkgs.rstudio.com/flexdashboard/articles/using.html#val boxes-1>

Examples

```
library(flexdashboard)

valueBox(42, caption = "Errors", icon="fa-thumbs-down")
valueBox(107, caption = "Trials", icon="fa-tag")
valueBox(247, caption = "Connections", icon="fa-random")
```

`valueBox-shiny` *Shiny bindings for valueBox*

Description

Output and render functions for using valueBox within Shiny applications and interactive Rmd documents.

Usage

```
valueBoxOutput(outputId, width = "100%", height = "160px")

renderValueBox(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a gauge
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Index

bslib::bs_theme(), *6*

flex_dashboard, 4
flexdashboard *(*flexdashboard-package*)*, 2
flexdashboard-package, 2

gauge, 7
gauge-shiny, 8
gaugeOutput *(*gauge-shiny*)*, 8
gaugeSectors *(*gauge*)*, 7

htmlDependency, *6*

includes, *6*

renderGauge *(*gauge-shiny*)*, 8
renderValueBox *(*valueBox-shiny*)*, 10
rmarkdown_format, *6*

valueBox, 9
valueBox-shiny, 10
valueBoxOutput *(*valueBox-shiny*)*, 10