

# Package ‘connectwidgets’

July 23, 2021

**Title** Organize and Curate Your Content Within 'RStudio Connect'

**Version** 0.1.1

**Description** A collection of helper functions and 'htmlwidgets' to help publishers curate content collections on 'RStudio Connect'. The components, Card, Grid, Table, Search, and Filter can be used to produce a showcase page or gallery contained within a static or interactive R Markdown page.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Suggests** knitr, lintr, rmarkdown, testthat, webmockr, withr

**Imports** R6, httr, jsonlite, tibble, htmlwidgets, reactable, dplyr, magrittr, htmltools, glue, rlang, digest, crosstalk, reactR, purrr, bslib, sass

**URL** <https://rstudio.github.io/connectwidgets/>,  
<https://github.com/rstudio/connectwidgets>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Brian Smith [aut, cre],  
Marcos Navarro [aut],  
David Aja [ctb],  
Kelly O'Briant [ctb],  
RStudio [cph]

**Maintainer** Brian Smith <brian@rstudio.com>

**Repository** CRAN

**Date/Publication** 2021-07-23 09:40:02 UTC

## R topics documented:

by\_owners . . . . . 2

by_tags . . . . .	3
Client . . . . .	3
connect . . . . .	4
connectwidgets . . . . .	5
content . . . . .	6
default_theme . . . . .	7
evaluate_widget_input . . . . .	7
gen_theme_dependency . . . . .	8
get_bootswatch_theme_name . . . . .	8
get_current_bootswatch_theme . . . . .	8
get_user_provided_theme . . . . .	9
resolve_theme_dependency . . . . .	9
rsc_card . . . . .	9
rsc_card-shiny . . . . .	10
rsc_cols . . . . .	10
rsc_filter . . . . .	11
rsc_filter-shiny . . . . .	12
rsc_grid . . . . .	13
rsc_grid-shiny . . . . .	13
rsc_search . . . . .	14
rsc_search-shiny . . . . .	14
rsc_table . . . . .	15
rsc_table_sync_theme . . . . .	15
warning_large_content . . . . .	15
warning_widget_input . . . . .	16

## Index 17

---

by_owners	<i>Filter content by owner(s)</i>
-----------	-----------------------------------

---

### Description

Returns content items owned by the specified user(s) (by username)

### Usage

```
by_owners(content, usernames)
```

```
by_owner(content, usernames)
```

### Arguments

content	Content data frame from content(...)
usernames	The username of the owner, or a list of usernames if multiple

### Value

The filtered content data frame

---

by_tags	<i>Filter content by tag(s)</i>
---------	---------------------------------

---

**Description**

Returns content items that have the specified tag(s) (by tag name)

**Usage**

```
by_tags(content, tagnames)
```

```
by_tag(content, tagnames)
```

**Arguments**

content	Content data frame from content(...)
tagnames	The name of the tag, or a list of names if multiple

**Value**

The filtered content data frame

---

Client	<i>Class representing a Connect API client</i>
--------	--

---

**Description**

Class representing a Connect API client

Class representing a Connect API client

**Details**

This class allows a user to interact with a Connect server via the Connect API. Authentication is done by providing an API key.

**Methods****Public methods:**

- [Client\\$new\(\)](#)
- [Client\\$print\(\)](#)
- [Client\\$GET\(\)](#)
- [Client\\$content\(\)](#)
- [Client\\$server\\_settings\(\)](#)
- [Client\\$clone\(\)](#)

**Method new():***Usage:*

Client\$new(server, api\_key)

**Method print():***Usage:*

Client\$print(...)

**Method GET():***Usage:*

Client\$GET(path, ..., writer = httr::write\_memory(), parser = "text")

**Method content():***Usage:*

Client\$content()

**Method server\_settings():***Usage:*

Client\$server\_settings()

**Method clone():** The objects of this class are cloneable with this method.*Usage:*

Client\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

---

`connect`*Create a connection to RStudio Connect*

---

**Description**

Creates a connection to RStudio Connect using the server URL and an api key.

**Usage**

```
connect(  
  server = Sys.getenv("CONNECT_SERVER", NA_character_),  
  api_key = Sys.getenv("CONNECT_API_KEY", NA_character_)  
)
```

**Arguments**

server	The server URL for accessing RStudio Connect. Defaults to environment variable <code>CONNECT_SERVER</code>
api_key	The API key to authenticate with RStudio Connect. Defaults to environment variable <code>CONNECT_API_KEY</code>

**Value**

An Client object

---

connectwidgets

*connectwidgets: Curate your content on RStudio Connect*

---

**Description**

connectwidgets provides UI components to help publishers provide curated organization of content on RStudio Connect.

**Connection**

Set `CONNECT_SERVER` and `CONNECT_API_KEY` environment variables and `connect()`. Fetch a tibble listing server content with `content()`. Use the `by_owner()` and `by_tags()` filter helpers to narrow your search.

**Components**

Present your content with different components:

- `rsc_card()` for highlighting individual content items
- `rsc_grid()` for groups of cards
- `rsc_table()` for a tabular view
- `rsc_search()` a text input for searching grids and tables
- `rsc_filter()` an input to filter grids or tables by owner, type, or tag

**Author(s)**

**Maintainer:** Brian Smith <brian@rstudio.com>

Authors:

- Marcos Navarro <marcos@rstudio.com>

Other contributors:

- David Aja <david@rstudio.com> [contributor]
- Kelly O'Briant <kelly.obriant@rstudio.com> [contributor]
- RStudio [copyright holder]

**See Also**

Useful links:

- <https://rstudio.github.io/connectwidgets/>
- <https://github.com/rstudio/connectwidgets>

---

`content`*Get Content Items*

---

**Description**

Returns content items as a data frame from the Connect server. It will only return content that is visible to the API key's user account.

**Usage**

```
content(client, unpublished = FALSE)
```

**Arguments**

<code>client</code>	A Client object (see <code>connect</code> )
<code>unpublished</code>	A boolean value specifying whether to return content that has not successfully published

**Value**

A data frame (tibble) of content items

- `id` - Auto-incrementing identifier for each content item (legacy)
- `guid` - Unique identifier for each content item (preferred)
- `app_mode` - The type of the content item (examples: `shiny`, `rmd-static`, `static`, `python-dash`, etc.)
- `content_category` - For static app modes, the specific category of content (examples: `site`, `plot`, `pin`, etc.)
- `name` - The name of the content item as set at initial publishing
- `title` - The user-provided title of the content item
- `description` - The user-provided description of the content item
- `url` - The URL to the content item
- `owner_guid` - Unique identifier of the owner of the content item
- `owner_username` - Username of the owner of the content item
- `owner_first_name` - First name of the owner of the content item
- `owner_last_name` - Last name of the owner of the content item
- `tags` - A data frame of the tags associated with the content item, with format: (`id`, `name`, `parent_id`, `created_time`, `updated_time`)
- `created_time` - The timestamp at which the content item was created
- `updated_time` - The timestamp at which the content item was last updated

---

default_theme	<i>Principal (most used) theming variables</i>
---------------	--

---

### Description

bg fg primary secondary success info warning danger base\_font code\_font heading\_font font\_scale

### Usage

```
default_theme()
```

### Details

Visit <https://rstudio.github.io/bslib/articles/bs4-variables.html> to see the full list of theming variables available

---

evaluate_widget_input	<i>Evaluates required columns for widget's input</i>
-----------------------	--

---

### Description

Evaluates required columns for widget's input

### Usage

```
evaluate_widget_input(widget, colnames, required)
```

### Arguments

widget	Widget's name to identify errors thrown
colnames	Column names provided from the widget content input
required	List of required columns to look for in the widget's input

---

gen\_theme\_dependency    *Generate the theme's bslib::bs\_dependency to be used by a widget.*

---

### Description

Generate the theme's bslib::bs\_dependency to be used by a widget.

### Usage

```
gen_theme_dependency(widget_name, theme, default_base = FALSE)
```

### Arguments

widget_name	The name of the widget (e.g: rsc_grid)
theme	The bslib theme to generate the CSS dependency
default_base	Using the default theme or not

---

get\_bootstrap\_theme\_name  
*Get docmeta theme name (output: html\_document: theme)*

---

### Description

Get docmeta theme name (output: html\_document: theme)

### Usage

```
get_bootstrap_theme_name()
```

---

get\_current\_bootstrap\_theme  
*Get the current bootstrap theme if any*

---

### Description

Get the current bootstrap theme if any

### Usage

```
get_current_bootstrap_theme()
```



---

 get\_user\_provided\_theme

*Get the current user provided bslib theme if any*

---

**Description**

Get the current user provided bslib theme if any

**Usage**

```
get_user_provided_theme()
```

---

resolve\_theme\_dependency

*Resolve and get theme to be used by a widget. It could be the default connectwidgets styling theme or one provided by the user.*

---

**Description**

Resolve and get theme to be used by a widget. It could be the default connectwidgets styling theme or one provided by the user.

**Usage**

```
resolve_theme_dependency(widget_name)
```

**Arguments**

widget_name	The name of the widget (e.g: rsc_grid)
-------------	--

---

rsc\_card

*Card view for content*

---

**Description**

Renders a card view for the provided content

**Usage**

```
rsc_card(content)
```

**Arguments**

content	A data frame from Connect's content. Requires the following columns "guid," "url", "title". And, although optional, expects an "owner_username", "description" and "updated_time" columns.
---------	--

---

rsc\_card-shiny      *Shiny bindings for rsc\_card*

---

### Description

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

### Usage

```
rsc_card_output(outputId, width = "100%", height = "400px")
renderRscCard(expr, env = parent.frame(), quoted = FALSE)
```

### Arguments

<code>outputId</code>	output variable to read from
<code>width, height</code>	Must be a valid CSS unit (like <code>'100%'</code> , <code>'400px'</code> , <code>'auto'</code> ) or a number, which will be coerced to a string and have <code>'px'</code> appended.
<code>expr</code>	An expression that generates a <code>rsc_card</code>
<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.

---

rsc\_cols      *Arrange HTML elements or widgets in Bootstrap columns*

---

### Description

This helper function makes it easy to put HTML elements side by side. It can be called directly from the console but is especially designed to work in an R Markdown document. Note that this is ripped directly from <https://github.com/rstudio/crosstalk> without the additional bootstrap dependency (since it is already expected in `connectwidgets`)

### Usage

```
rsc_cols(..., widths = NA, device = c("xs", "sm", "md", "lg"))
```

**Arguments**

...	htmltools tag objects, lists, text, HTML widgets, or NULL. These arguments should be unnamed.
widths	The number of columns that should be assigned to each of the ... elements (the total number of columns available is always 12). The width vector will be recycled if there are more ... arguments. NA columns will evenly split the remaining columns that are left after the widths are recycled and non-NA values are subtracted.
device	The class of device which is targeted by these widths; with smaller screen sizes the layout will collapse to a one-column, top-to-bottom display instead. xs: never collapse, sm: collapse below 768px, md: 992px, lg: 1200px.

**Value**

A [browsable](#) HTML element.

**Examples**

```
library(htmltools)

# If width is unspecified, equal widths will be used
rsc_cols(
  div(style = css(width="100%", height="400px", background_color="red")),
  div(style = css(width="100%", height="400px", background_color="blue"))
)

# Use NA to absorb remaining width
rsc_cols(widths = c(2, NA, NA),
  div(style = css(width="100%", height="400px", background_color="red")),
  div(style = css(width="100%", height="400px", background_color="blue")),
  div(style = css(width="100%", height="400px", background_color="green"))
)

# Recycling widths
rsc_cols(widths = c(2, 4),
  div(style = css(width="100%", height="400px", background_color="red")),
  div(style = css(width="100%", height="400px", background_color="blue")),
  div(style = css(width="100%", height="400px", background_color="red")),
  div(style = css(width="100%", height="400px", background_color="blue"))
)
```

**Description**

Filter content rows with owner, content type and tags, expects the exact same frame passed to the view widget being filtered.

**Usage**

```
rsc_filter(content)
```

**Arguments**

content	A data frame from Connect's content. Although optional, expects an "owner_username", "app_mode" and "tags" columns.
---------	---

---

rsc_filter-shiny	<i>Shiny bindings for rsc_filter</i>
------------------	--------------------------------------

---

**Description**

Output and render functions for using rsc\_filter within Shiny applications and interactive Rmd documents.

**Usage**

```
rscfilterOutput(outputId, width = "100%", height = "400px")
```

```
renderRscfilter(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 rsc_filter
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.

---

rsc_grid	<i>Grid view for content</i>
----------	------------------------------

---

**Description**

Renders a grid view of the provided content items

**Usage**

```
rsc_grid(content)
```

**Arguments**

content	A data frame from Connect's content. Requires the following columns "guid", "url", "title", "app_mode", "owner_username". And, although optional, expects an "updated_time" column.
---------	---

---

rsc_grid-shiny	<i>Shiny bindings for rsc_grid</i>
----------------	------------------------------------

---

**Description**

Output and render functions for using rsc\_grid within Shiny applications and interactive Rmd documents.

**Usage**

```
rscgridOutput(outputId, width = "100%", height = "400px")
renderRscgrid(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 rsc_grid
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.

---

rsc_search	<i>Search widget</i>
------------	----------------------

---

### Description

Search for matching content name/title within the shared data object passed. Expects the exact same frame passed to the view widget being filtered.

### Usage

```
rsc_search(content)
```

### Arguments

content	A shared object from Connect's content
---------	--

---

rsc_search-shiny	<i>Shiny bindings for rsc_search</i>
------------------	--------------------------------------

---

### Description

Output and render functions for using rsc\_search within Shiny applications and interactive Rmd documents.

### Usage

```
rscsearchOutput(outputId, width = "100%", height = "400px")
```

```
renderRscsearch(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 rsc_search
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.

---

rsc_table	<i>Reactable table of the content</i>
-----------	---------------------------------------

---

**Description**

Renders a reactable table of the provided content items

**Usage**

```
rsc_table(content)
```

**Arguments**

content	The tibble of content provided by <code>connectwidgets::content()</code> Requires the columns "guid", "url", "title", "app_mode", "owner_username" and "updated_time".
---------	--

---

rsc_table_sync_theme	<i>Resolve reactable theme for rsc_table</i>
----------------------	--

---

**Description**

Resolve reactable theme for rsc\_table

**Usage**

```
rsc_table_sync_theme()
```

---

warning_large_content	<i>Show warnings for large content</i>
-----------------------	--

---

**Description**

Show warnings for large content

**Usage**

```
warning_large_content(content, max_rows = 500)
```

**Arguments**

content	Content data frame or Crosstalk SharedData object
max_rows	Maximum number of rows before warning (default 500)

---

warning\_widget\_input *Show warnings of expected columns for widget's input*

---

**Description**

Show warnings of expected columns for widget's input

**Usage**

```
warning_widget_input(widget, colnames, expected)
```

**Arguments**

widget	Widget's name to identify warnings thrown
colnames	Column names provided from the widget content input
expected	List of expected columns to look for in the widget's input



# Index

## \* R6 classes

- Client, 3
- browsable, 11
- by\_owner (by\_owners), 2
- by\_owner(), 5
- by\_owners, 2
- by\_tag (by\_tags), 3
- by\_tags, 3
- by\_tags(), 5
- Client, 3
- connect, 4
- connect(), 5
- connectwidgets, 5
- connectwidgets-package
  - (connectwidgets), 5
- content, 6
- content(), 5
- default\_theme, 7
- evaluate\_widget\_input, 7
- gen\_theme\_dependency, 8
- get\_bootswatch\_theme\_name, 8
- get\_current\_bootswatch\_theme, 8
- get\_user\_provided\_theme, 9
- renderRsccard (rsc\_card-shiny), 10
- renderRscfilter (rsc\_filter-shiny), 12
- renderRscgrid (rsc\_grid-shiny), 13
- renderRscsearch (rsc\_search-shiny), 14
- resolve\_theme\_dependency, 9
- rsc\_card, 9
- rsc\_card(), 5
- rsc\_card-shiny, 10
- rsc\_cols, 10
- rsc\_filter, 11
- rsc\_filter(), 5
- rsc\_filter-shiny, 12
- rsc\_grid, 13
- rsc\_grid(), 5
- rsc\_grid-shiny, 13
- rsc\_search, 14
- rsc\_search(), 5
- rsc\_search-shiny, 14
- rsc\_table, 15
- rsc\_table(), 5
- rsc\_table\_sync\_theme, 15
- rsccardOutput (rsc\_card-shiny), 10
- rscfilterOutput (rsc\_filter-shiny), 12
- rscgridOutput (rsc\_grid-shiny), 13
- rscsearchOutput (rsc\_search-shiny), 14
- warning\_large\_content, 15
- warning\_widget\_input, 16