Package 'dockViewR'

July 22, 2025

Title Layout Manager Widget for R and 'shiny' Apps

Version 0.2.0

Description Provides R bindings to the 'dockview' 'JavaScript' library <https://dockview.dev/>. Create fully customizable grid layouts (docks) in seconds to include in interactive R reports with R Markdown or 'Quarto' or in 'shiny' apps <https://shiny.posit.co/>. In 'shiny' mode, modify docks by dynamically adding, removing or moving panels or groups of panels from the server function. Choose among 8 stunning themes (dark and light), serialise the state of a dock to restore it later.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports htmlwidgets, htmltools, shiny

Suggests knitr, rmarkdown, roxy.shinylive, shinytest2, testthat (>= 3.0.0), visNetwork, quarto, thematic

URL https://github.com/cynkra/dockViewR, https://cynkra.github.io/dockViewR/

BugReports https://github.com/cynkra/dockViewR/issues

Config/testthat/edition 3

Depends R (>= 2.10)

VignetteBuilder quarto

NeedsCompilation no

Author David Granjon [aut, cre], Nelson Stevens [aut], Nicolas Bennett [aut], mathuo [cph], cynkra GmbH [fnd]

Maintainer David Granjon <dgranjon@ymail.com>

Repository CRAN

Date/Publication 2025-07-10 15:10:07 UTC

Contents

`add_panel` *Add Panel dynamically*

Description

Add Panel dynamically

Usage

```
add_panel(dock_id, panel, ..., session = getDefaultReactiveDomain())
```

Arguments

`dock_id`	Dock unique id. When using modules the namespace is automatically added.
`panel`	A panel object. See panel for the different parameters.
`...`	Other options passed to the API. Not used yet.
`session`	shiny session object. See `https://dockview.dev/docs/api/dockview/panelApi/`.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

See Also

`panel()`

dockViewOutput *Shiny bindings for dock_view*

Description

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

Usage

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

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

renderDockView(expr, env = parent.frame(), quoted = FALSE)

render_dock_view(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 dock_view
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.

Value

dockViewOutput and dock_view_output return a Shiny output function that can be used in the UI definition. renderDockView and render_dock_view return a Shiny render function that can be used in the server definition to render a dock_view element.

dock_view *Create a dock view widget*

Description

Creates an interactive dock view widget that enables flexible layout management with draggable, resizable, and dockable panels. This is a wrapper around the dockview.dev JavaScript library, providing a powerful interface for creating IDE-like layouts in Shiny applications or R Markdown documents.

Usage

```
dock_view(
  panels,
  ...,
  theme = c("light-spaced", "light", "abyss", "abyss-spaced", "dark", "vs", "dracula",
    "replit"),
  add_tab = list(enable = FALSE, callback = NULL),
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

panels	Widget configuration. Slot for panel.
...	Other options. See https://dockview.dev/docs/api/dockview/options/.
theme	Theme. One of c("abyss", "dark", "light", "vs", "dracula", "replit").
add_tab	Globally controls the add tab behavior. List with enable and callback. Enable is a boolean, default to FALSE and callback is a JavaScript function passed with JS.
width	Widget width.
height	Widget height.
elementId	When used outside Shiny.

Value

An HTML widget object.

Examples in Shinylive

example-1 Open in Shinylive

---

get_dock *get dock*

Description

get dock

get dock panels

get dock panels ids

get dock active group

get dock grid

get dock groups

get dock groups ids

get dock groups panels

save a dock

restore a dock

Usage

```
get_dock(dock_id, session = getDefaultReactiveDomain())

get_panels(dock_id, session = getDefaultReactiveDomain())

get_panels_ids(dock_id, session = getDefaultReactiveDomain())

get_active_group(dock_id, session = getDefaultReactiveDomain())

get_grid(dock_id, session = getDefaultReactiveDomain())

get_groups(dock_id, session = getDefaultReactiveDomain())

get_groups_ids(dock_id, session = getDefaultReactiveDomain())

get_groups_panels(dock_id, session = getDefaultReactiveDomain())

save_dock(dock_id, session = getDefaultReactiveDomain())

restore_dock(dock_id, data, session = getDefaultReactiveDomain())
```

Arguments

dock_id	Dock unique id. When using modules the namespace is automatically added.
session	shiny session object.
data	Data representing a serialised dock object.

Value

get_dock returns a list of 3 elements:

- grid: a list representing the dock layout.
- panels: a list having the same structure as panel() composing the dock.
- activeGroup: the current active group (a string).

Each other function allows to deep dive into the returned value of get_dock(). get_panels() returns the panels element of get_dock(). get_panels_ids() returns a character vector containing all panel ids from get_panels(). get_active_group() extracts the activeGroup component of get_dock() as a string. get_grid() returns the grid element of get_dock() which is a list. get_groups() returns a list of panel groups from get_grid(). get_groups_ids() returns a character vector of groups ids from get_groups(). get_groups_panels() returns a list of character vector containing the ids of each panel within each group. save_dock() and restore_dock() are used for their side effect to allow to respectively serialise and restore a dock object.

Note

Only works with server side functions like add_panel. Don't call it from the UI.

`move_group` *Move a group dynamically*

Description

move_group moves a group to a different position from within a shiny server function. The parameter from refers to the group-id you want to be moved. Likewise to refers to the group-id of a group you want to select as destination. The difference between `move_group2()` and `move_group()` is that `move_group2()` selects both from and to by panel-id, whereas `move_group()` selects by group-id.

Usage

```
move_group(
  dock_id,
  from,
  to,
  position = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`dock_id`	Dock unique id. When using modules the namespace is automatically added.
`from`	Group-id of a panel within the group that should be moved.
`to`	Group-id of a panel within the group you want as a destination.
`position`	Group position options: one of `"left"`, `"right"`, `"top"`, `"bottom"`, `"center"`.
`session`	shiny session object. See `https://dockview.dev/docs/api/dockview/panelApi/`.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

move_group2 *Move a group dynamically*

Description

move_group2 moves a group to a different position from within a shiny server function. The parameter from refers to a panel-id of a panel within the group you want to move. Likewise to refers to a panel-id of a panel within the group you want to select as to. The difference between `move_group2()` and `move_group()` is that `move_group2()` selects both from and to by panel-id, whereas `move_group()` selects by group-id.

Usage

```
move_group2(
  dock_id,
  from,
  to,
  position = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`dock_id`	Dock unique id. When using modules the namespace is automatically added.
`from`	Panel-id of a panel within the group that should be moved.
`to`	Panel-id of a panel within the group you want as a to.
`position`	Group position options: one of `"left"`, `"right"`, `"top"`, `"bottom"`, `"center"`.
`session`	shiny session object. See `https://dockview.dev/docs/api/dockview/panelApi/`.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

move_panel *Move Panel dynamically*

Description

Move Panel dynamically

Usage

```
move_panel(
  dock_id,
  id,
  position = NULL,
  group = NULL,
  index = NULL,
  session = getDefaultReactiveDomain()
)
```

Arguments

dock_id	Dock unique id. When using modules the namespace is automatically added.
id	Panel id.
position	Panel position options: one of "left", "right", "top", "bottom", "center".
group	ID of the panel you want to move the target to. They must belong to different groups.
index	Panel index. If panels belong to the same group, you can use index to move the target panel at the desired position. When group is left NULL, index must be passed and cannot exceed the total number of panels or be negative.
session	shiny session object. See https://dockview.dev/docs/api/dockview/panelApi/.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

---

panel *Dock panel*

Description

Create a panel for use within a dock_view() widget. Panels are the main container components that can be docked, dragged, resized, and arranged within the dockview interface.

Usage

```
panel(
  id,
  title,
  content,
  active = TRUE,
  remove = list(enable = FALSE, mode = "auto"),
  ...
)
```

Arguments

id	Panel unique id.
title	Panel title.
content	Panel content. Can be a list of Shiny tags.
active	Is active?
remove	List with two fields: enable and mode. Enable is a boolean and mode is one of manual, auto (default to auto). In auto mode, dockview JS removes the panel when it is closed and all its content. If you need more control over the panel removal, set it to manual so you can explicitly call remove_panel() and perform other tasks. On the server side, a shiny input is available input[["<dock_ID>_panel-to-remove"]] so you can create observers with custom logic.
...	Other options passed to the API. See https://dockview.dev/docs/api/dockview/panelApi/. If you pass position, it must be a list with 2 fields: • referencePanel: reference panel id. • direction: one of above, below, left, right or within (above, below, left, right put the panel in a new group, while within puts the panel after its reference panel in the same group). Position is relative to the reference panel target.

Value

A list representing a panel object to be consumed by dock_view:

- id: unique panel id (string).
- title: panel title (string).
- content: panel content (shiny.tag.list or single shiny.tag).
- active: whether the panel is active or not (boolean).
- ...: extra parameters to pass to the API.

---

remove_panel *Remove Panel dynamically*

Description

Remove Panel dynamically

Usage

```
remove_panel(dock_id, id, session = getDefaultReactiveDomain())
```

Arguments

dock_id	Dock unique id. When using modules the namespace is automatically added.
id	Id of the panel that ought to be removed.
session	shiny session object. See https://dockview.dev/docs/api/dockview/panelApi/.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

`select_panel` *Select a panel dynamically*

Description

Select a panel dynamically

Usage

```
select_panel(dock_id, id, session = getDefaultReactiveDomain())
```

Arguments

`dock_id`	Dock unique id. When using modules the namespace is automatically added.
`id`	Panel id.
`session`	shiny session object. See https://dockview.dev/docs/api/dockview/panelApi/.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

`update_dock_view` *Update options for dockview instance*

Description

This does not rerender the widget, just update options like global theme.

Usage

```
update_dock_view(dock_id, options, session = getDefaultReactiveDomain())
```

Arguments

`dock_id`	The id of the dock view widget to update.
`options`	List of options for the dock_view instance.
`session`	Shiny session object.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

Index