Package: quartabs (via r-universe)

November 22, 2026

Title Dynamically Generate Tabset Panels in 'Quarto' HTML Documents

Version 0.2.0

Description Dynamically generate tabset panels
<https://quarto.org/docs/output-formats/html-basics.html#tabsets>
in 'Quarto' HTML documents using a data frame as input.

License MIT + file LICENSE

URL https://sayuks.github.io/quartabs/,
https://github.com/sayuks/quartabs

BugReports https://github.com/sayuks/quartabs/issues

Suggests altdoc, dplyr (>= 1.0.0), DT, flextable, gt (>= 0.9.0),
htmltools, knitr, plotly, purrr, quarto, reactable,
sessioninfo, spelling, testthat (>= 3.0.0), tibble, tidyr,
tinytable, utils

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Yusuke Sasaki [aut, cre]

Maintainer Yusuke Sasaki <sayuks.dev@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-22 16:27:18 UTC

RemoteUrl https://github.com/cran/quartabs

RemoteRef HEAD

RemoteSha bb5d22c4587d28c552eeb9f24d3ec6aff073329d

Contents

render_tabset 2

Index **6**

render_tabset *Dynamically Generate Tabset Panels in Quarto HTML Documents*

Description

`render_tabset()` takes a data frame as input and outputs the markdown that generates the tabset to stdout (console). **Only works with Quarto HTML documents.** See Get started for details.

Usage

```
render_tabset(
  data,
  tabset_vars,
  output_vars,
  layout = NULL,
  heading_levels = NULL,
  active = NULL,
  pills = FALSE,
  tabset_width = "default"
)
```

Arguments

`data`	A data frame.
`tabset_vars`	Columns to use as tabset labels. Internally passed to the `select` argument of `subset()`. Accepts raw column names, strings, numbers and logical values.
`output_vars`	Columns to display in each tabset panel. Internally passed to the `select` argument of `subset()`. Accepts raw column names, strings, numbers and logical values.
`layout`	`NULL` or a character vector of length 1 for specifying layout in tabset panel. If not `NULL`, `layout` must begin with at least three or more repetitions of ":" (e.g. ":::"). Closing div (e.g. ":::") is inserted automatically. See for details: `https://quarto.org/docs/authoring/figures.html#complex-layouts`.
`heading_levels`	`NULL` or a vector consisting of natural numbers and missing values. The length is equal to the number of columns specified in `tabset_vars`. This controls whether it is partially (or entirely) displayed as normal header instead of tabset. • If `heading_levels` is a `NULL`, all output is tabset. • If `heading_levels` is a vector of positive natural number, the elements of the vector correspond to the columns specified in `tabset_vars`. – If the element is integer, the tabset column is displayed as headers with their level, not tabset. (e.g. 2 means h2 header). Levels 1 to 6 are recommended. The reason is that quarto supports headers up to 6. 7 and above will also work, but they are displayed as normal text. In addition, considering the chapter format, it is preferable to gradually increase the level, as in 1, 2 and 3.

- If the element is NA, tabset is displayed.

`active` `NULL` or a named list that chooses the active tab, the tab shown by default when the page loads (Quarto's `.active` class, which requires Quarto 1.7 or later; older versions ignore it). Each name is a column specified in `tabset_vars`, and each element is either:

- A single non-missing value: the tab with this value is active in every tabset of that column that contains it. It is an error if no tab of that column has this value.
- A function: it is called once per tabset of that column (that is, for the sibling tabs under each parent tab) with the unique tab values in display order, and must return one of them, or `NA` for no active tab. For example, `max` makes the latest tab active under each parent tab, and `function(x) x[length(x)]` the last one.

Functions are applied before the conversion of factor, Date and POSIXt columns to character, so numbers are compared as numbers and factors keep their levels. A character value is compared with the tab labels (e.g. `"2024-01-01"` for a Date tab); other values are compared with the original column (e.g. `as.Date("2024-01-01")`). Without an active tab, a tabset shows its first tab. Columns displayed as headings by `heading_levels` cannot be specified.

`pills` Logical, use pills or not. See https://getbootstrap.com/docs/5.2/components/navs-tabs/#pills for details. If `heading_levels` is specified, this will be ignored.

`tabset_width` Character, one of "default", "fill" and "justified". See https://getbootstrap.com/docs/5.2/components/navs-tabs/#fill-and-justify for details. If `heading_levels` is specified, this will be ignored.

Details

- Write `#| results: asis` at the beginning of the chunk or `results='asis'` in the chunk options.
- If multiple `tabset_vars` are given, create nested tabsets.
- For columns specified in `output_vars`, columns of type list are output with `print()` and normal columns are output with `cat()`.
- The `data` is sorted internally by `tabset_vars`, so tabs are displayed in ascending order. Factor columns are sorted by their levels (before the conversion to character described below), so the tab order can be controlled with the levels. For example, `factor(x, levels = unique(x))` keeps the order of first appearance in the data. Since levels apply to the whole column, nested tabs cannot have a different order under each parent tab.
- If `tabset_vars` or `output_vars` have "factor", "Date" and "POSIXt" columns, they are converted internally to character. This is to prevent it being displayed as numeric when `cat()` is executed. Sorting by `tabset_vars` is performed before conversion to string.

Value

`NULL` invisibly. This function outputs the markdown that generates the tabset to stdout (console).

Limitations

- `layout` is intended for simplified use cases and complex layouts may not work.
- When outputting tables or figures that use JavaScript (such as `{plotly}`, `{leaflet}`, `{DT}`, `{reactable}`, etc.), it seems JavaScript dependencies need to be resolved. A simple solution is to wrap the output in `htmltools::div()` and create a dummy plot in another chunk. See the Get started for details.
- When `tabset_vars` and `output_vars` have the following columns, they may not display well:
  - A column of type list contains a named vector or list (This is for `output_vars`. `tabset_vars` must not contain list columns).
  - Classes with their own printing methods, such as "difftime", "ts", .etc.
- When specifying a list-type column that includes ggplot objects in `output_vars`, setting the chunk option `echo: fenced` may cause the plots to not display correctly.
- If the chunk label starts with `fig-` (e.g. `#| label: fig-example`), Quarto holds all plots until the end of the chunk, so they are displayed below the tabset instead of inside each tab panel. Use a label that does not start with `fig-`, or no label. To make each plot a cross-referenceable figure, see the Get started for details.

References

As this function is focused on quickly and dynamically generating tabsets and chunks, it is difficult to customize it on a chunk-by-chunk basis. The regular way to dynamically create chunks is to use functions such as `knitr::knit()`, `knitr::knit_child()`, `knitr::knit_expand()`, etc. For more information on these, see the following links.

- Heiss, Andrew. 2024. "Guide to Generating and Rendering Computational Markdown Content Programmatically with Quarto." November 4, 2024. doi:10.59350/pa44jcc302.
- https://pkg.yihui.org/rmarkdown-cookbook/child-document.html#child-document
- https://pkg.yihui.org/rmarkdown-cookbook/knit-expand.html

Examples

```
# sample data
df <- data.frame(
  group1 = c(rep("A", 3), rep("B", 3)),
  group2 = rep(c("X", "Y", "Z"), 2),
  value1 = 1:6,
  value2 = letters[1:6]
)

# Here are examples of the output before it is converted to tabset.
# If you want it to actually work, in the .qmd file,
# set `results='asis'` in the chunk options or
# write `#| results: asis` at the beginning of the chunk.

# Basic usage
render_tabset(df, group1, value1)
```

```
# Nested tabset, two outputs side by side with a width of 1:1
render_tabset(
  df,
  c(group1, group2),
  c(value1, value2),
  layout = "::: {layout-ncol=2}"
)

# Use heading instead of tabset
render_tabset(
  df,
  c(group1, group2),
  value1,
  heading_levels = c(2, 3)
)

# Show tab "B" and, under each tab, the last nested tab by default
render_tabset(
  df,
  c(group1, group2),
  value1,
  active = list(group1 = "B", group2 = max)
)
```

Index

cat(), *3*

htmltools::div(), *4*

knitr::knit(), *4*
knitr::knit_child(), *4*
knitr::knit_expand(), *4*

print(), *3*

render_tabset, 2

subset(), *2*