# Package 'shinydataviewer'

May 9, 2026

**Title** Reusable Data Viewer Module for 'shiny'

**Version** 0.1.0

**Description** Provides a reusable 'shiny' module for viewing tabular data with a searchable 'reactable' table and a variable summary sidebar built with 'bslib'.

**License** MIT + file LICENSE

**BugReports** https://github.com/Ryan-W-Harrison/shinydataviewer/issues

**Imports** bslib, htmltools, reactable, shiny

**Suggests** knitr, pkgdown, rmarkdown, testthat (>= 3.0.0)

**Config/Needs/website** pkgdown, knitr, rmarkdown

**Config/Needs/README** knitr, rmarkdown

**Config/testthat/edition** 3

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3

**URL** https://ryan-w-harrison.github.io/shinydataviewer/, https://github.com/Ryan-W-Harrison/shinydataviewer

**NeedsCompilation** no

**Author** Ryan Harrison [aut, cre, cph]

**Maintainer** Ryan Harrison <harrison.ryan.w@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-04-09 09:00:03 UTC

## Contents

`data_viewer_card_ui` *Data Viewer Module Wrapped in a Card*

## Description

Data Viewer Module Wrapped in a Card

## Usage

```
data_viewer_card_ui(
  id,
  title = NULL,
  full_screen = TRUE,
  sidebar_title = NULL,
  table_controls_position = c("top", "bottom")
)
```

## Arguments

| | |
|---|---|
| `id` | Module id. |
| `title` | Optional card title. |
| `full_screen` | Whether the wrapper card can enter full screen mode. |
| `sidebar_title` | Optional title for the variable summary sidebar. |
| `table_controls_position` | Where table pagination controls should appear. One of `"top"` or `"bottom"`. |

## Value

A card containing the module UI.

## Examples

```
ui <- bslib::page_fillable(
  theme = bslib::bs_theme(version = 5),
  bslib::layout_columns(
    col_widths = c(4, 8),
    bslib::card(
      bslib::card_header("Context"),
      bslib::card_body("Supporting content")
    ),
    data_viewer_card_ui(
      "viewer",
      title = "Dataset",
      full_screen = FALSE
    )
  )
)
```

```
server <- function(input, output, session) {
  data_viewer_server(
    "viewer",
    data = shiny::reactive(mtcars)
  )
}

if (interactive()) {
  shiny::shinyApp(ui, server)
}
```

---

`data_viewer_server` *Data Viewer Module Server*

---

## Description

Data Viewer Module Server

## Usage

```
data_viewer_server(
  id,
  data,
  top_n = 6,
  default_page_size = NULL,
  page_size_options = c(15, 25, 50, 100),
  searchable = TRUE,
  filterable = TRUE,
  sortable = TRUE,
  summary_card_fn = variable_summary_card,
  reactable_theme = NULL,
  default_col_def = NULL,
  reactable_args = list()
)
```

## Arguments

| | |
|---|---|
| `id` | Module id. |
| `data` | Reactive returning a data frame. Supported column classes are numeric, integer, character, factor, logical, `Date`, and `POSIXct`/`POSIXt`. |
| `top_n` | Maximum number of categorical levels to keep in compact summary views before collapsing the remainder into `"Other"`. |
| `default_page_size` | Optional default number of rows to show in the table. If `NULL`, the module shows all rows up to the largest page-size option. |

page_size_options
: Page-size options shown in the table controls.

searchable
: Whether the table search box is enabled.

filterable
: Whether column filters are enabled.

sortable
: Whether column sorting is enabled.

summary_card_fn
: Function used to render each variable summary card. It must accept at least `(summary_row, index)` and return a Shiny UI tag.

reactable_theme
: Optional `reactable::reactableTheme()` object. If `NULL`, the module uses its built-in theme.

default_col_def
: Optional default `reactable::colDef()` to apply to all columns. If `NULL`, the module uses its built-in default column definition.

reactable_args
: Optional named list of additional arguments passed to `reactable::reactable()`. These values override the module defaults.

## Value

Invisibly returns a list of reactives named `data` and `summary`.

## Examples

```
custom_summary_card <- function(summary_row, index) {
  htmltools::tags$div(
    class = "custom-summary-card",
    sprintf("%s: %s", summary_row$var_name[[1]], summary_row$type[[1]])
  )
}

ui <- bslib::page_fillable(
  theme = bslib::bs_theme(version = 5),
  data_viewer_card_ui("viewer", title = "Iris")
)

server <- function(input, output, session) {
  data_viewer_server(
    "viewer",
    data = shiny::reactive(iris),
    searchable = TRUE,
    filterable = TRUE,
    sortable = TRUE,
    summary_card_fn = custom_summary_card
  )
}

if (interactive()) {
  shiny::shinyApp(ui, server)
}
```

`data_viewer_ui` *Data Viewer Module UI*

## Description

Data Viewer Module UI

## Usage

```
data_viewer_ui(
  id,
  standalone = TRUE,
  table_title = NULL,
  sidebar_title = NULL,
  table_controls_position = c("top", "bottom")
)
```

## Arguments

| | |
|---|---|
| `id` | Module id. |
| `standalone` | Whether to render with the built-in table card. Set to `FALSE` when embedding inside another `bslib::card()`. |
| `table_title` | Optional title for the table region. |
| `sidebar_title` | Optional title for the variable summary sidebar. |
| `table_controls_position` | Where table pagination controls should appear. One of `"top"` or `"bottom"`. |

## Value

The module UI.

## Examples

```
ui <- bslib::page_fillable(
  theme = bslib::bs_theme(version = 5),
  data_viewer_ui("viewer")
)

server <- function(input, output, session) {
  data_viewer_server(
    "viewer",
    data = shiny::reactive(iris)
  )
}

if (interactive()) {
  shiny::shinyApp(ui, server)
}
```

summarize_columns *Summarize Columns in a Data Frame*

## Description

Summarize Columns in a Data Frame

## Usage

```
summarize_columns(df, top_n = 6)
```

## Arguments

| | |
|---|---|
| df | A data frame to summarize. Supported column classes are numeric, integer, character, factor, logical, Date, and POSIXct/POSIXt. |
| top_n | Maximum number of categorical levels to keep before collapsing the remainder into "Other". |

## Value

A data frame with one row per column and the following columns: var_name, type, n_missing, pct_missing, n_unique, summary_stats, and distribution_data. summary_stats is a list-column containing per-type summary values used by the details accordion. distribution_data is a list-column containing precomputed histogram or categorical count payloads used by the compact mini charts.

# Index