

Package: datasetviewer (via r-universe)

July 9, 2026

Title 'SAS Studio'-Style Interactive Dataset Viewer

Version 0.1.1

Description An interactive dataset viewer that renders a fast, scrollable grid with a column-selection panel, per-column property metadata, and a names-versus-labels header toggle, modelled on the 'SAS Studio' table viewer. Runs from one codebase in interactive 'Shiny' apps and in static HTML documents, with a free-text row filter, header sort, and CSV export, and handles large datasets without row sampling by querying them in the browser with 'DuckDB-WASM'.

License MIT + file LICENSE

URL <https://github.com/vthanik/datasetviewer>,
<https://vthanik.github.io/datasetviewer/>

BugReports <https://github.com/vthanik/datasetviewer/issues>

Encoding UTF-8

Imports cli, htmltools, htmlwidgets, jsonlite, nanoparquet, rlang

Suggests artoo, dplyr, hms, knitr, quarto, shiny, testthat (>= 3.0.0),
tibble, withr

VignetteBuilder quarto

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Vignesh Thanikachalam [aut, cre, cph]

Maintainer Vignesh Thanikachalam <about.vignesh@gmail.com>

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

Date/Publication 2026-07-09 09:50:02 UTC

RemoteUrl <https://github.com/cran/datasetviewer>

RemoteRef HEAD

RemoteSha 1654bc74f3941188b757f95ba998577266508e59

Contents

dataset_viewer	2
datasetviewer-shiny	3

Index	5
--------------	----------

dataset_viewer	<i>View a dataset in an interactive SAS Studio-style grid</i>
----------------	---

Description

Renders a fast, scrollable data grid with a column-selection panel and per-column property metadata. The same widget renders in interactive Shiny apps and in static HTML documents.

Usage

```
dataset_viewer(
  x,
  ...,
  view = c("names", "labels"),
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

x	<i>Dataset to view.</i> <data.frame character(1)>. A data frame, or a file path read via <code>artoo::read_dataset()</code> (xpt, Dataset-JSON, NDJSON, 'Parquet', RDS). An artoo-conformed frame supplies labels, formats, and lengths to the property panel; a plain frame uses synthesized metadata.
...	Reserved for future arguments.
view	<i>Initial header mode.</i> <character(1)>. "names" (default, matching SAS Studio) shows column names as headers; "labels" shows labels, falling back to names when a label is absent.
width, height	<i>Widget sizing.</i> <character(1) numeric(1) NULL>. Passed through to <code>htmlwidgets::createWidget()</code> .
elementId	<i>Explicit DOM id.</i> <character(1) NULL>. Usually left NULL so <code>htmlwidgets</code> assigns one.

Details

Query engine. The data is sent to the browser once as 'Parquet' and queried in place with DuckDB-WASM, so filter, sort, and paging run over the whole dataset with no row sampling. The engine (~35 MB) loads from a CDN by default but is fetched into the package at install time when reachable, so a Shiny app can serve it to browsers with no internet at runtime. Set `options(datasetviewer.use_local_engine`

= FALSE) to force the CDN (for small self-contained HTML). See vignette("datasetviewer") for offline and corporate deployment, including the DATASETVIEWER_DUCKDB_* install-time environment variables.

Value

An *htmlwidget*. Print it to render, or use it as a Shiny output.

See Also

Shiny bindings: [datasetviewerOutput\(\)](#), [renderDatasetViewer\(\)](#).

Examples

```
# ---- Example 1: view a plain data frame ----
#
# Wrap any data frame to get the interactive grid. Printing the widget in
# an interactive session or a rendered document shows it; here we inspect
# the payload so the example stays headless and self-contained (printing a
# widget would launch a browser under R CMD check).
viewer <- dataset_viewer(mtcars)
viewer$x$n_rows

# ---- Example 2: CDISC labels as headers ----
#
# With the sibling artoo package, a CDISC-conformed frame supplies column
# labels, formats, and storage lengths to the property pane and the
# names-versus-labels header toggle. Start on labels with view = "labels".
if (requireNamespace("artoo", quietly = TRUE)) {
  labelled <- dataset_viewer(artoo::cdisc_ads1, view = "labels")
  labelled$x$columns[[1]]$label
}
```

datasetviewer-shiny *Shiny bindings for the dataset viewer*

Description

Output and render functions to embed [dataset_viewer\(\)](#) in a Shiny app. The widget pushes its live view state back to Shiny as inputs, so the app can reuse the user's filter, sort, and column selection server-side.

Usage

```
datasetviewerOutput(outputId, width = "100%", height = "500px")

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

Arguments

outputId	<i>Output slot id.</i> <character(1)>. Matched by datasetviewerOutput() and renderDatasetViewer().
width, height	<i>CSS sizing.</i> <character(1)>. Any valid CSS size; the grid fills the element.
expr	<i>Render expression.</i> A call that returns a dataset_viewer() widget.
env, quoted	<i>Evaluation control.</i> Standard htmlwidgets render plumbing; leave at their defaults.

Details

Inputs published. For an output with id "viewer" the widget sets, on every change:

- input\$viewer_columns (<character>): selected column names.
- input\$viewer_filter (<character(1)>): the filter expression.
- input\$viewer_sort (<list>): sort keys (name, dir).
- input\$viewer_view (<character(1)>): "names" or "labels".

Value

datasetviewerOutput() returns a Shiny output UI element; renderDatasetViewer() returns a Shiny render function.

See Also

[dataset_viewer\(\)](#) for the widget these bindings embed.

Examples

```
# ---- Example 1: read the viewer's filter and selection server-side ----
#
# The app shows a dataset and echoes the filter expression and selected
# columns the user builds in the grid. Runs only in an interactive session.
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    datasetviewerOutput("viewer", height = "500px"),
    verbatimTextOutput("state")
  )
  server <- function(input, output, session) {
    output$viewer <- renderDatasetViewer(dataset_viewer(mtcars))
    output$state <- renderText({
      paste0(
        "filter: ", input$viewer_filter, "\n",
        "columns: ", paste(input$viewer_columns, collapse = ", ")
      )
    })
  }
  shinyApp(ui, server)
}
```

Index

`artoo::read_dataset()`, 2

`dataset_viewer`, 2

`dataset_viewer()`, 3, 4

`datasetviewer-shiny`, 3

`datasetviewerOutput`

 (`datasetviewer-shiny`), 3

`datasetviewerOutput()`, 3

`htmlwidgets::createWidget()`, 2

`renderDatasetViewer`

 (`datasetviewer-shiny`), 3

`renderDatasetViewer()`, 3