Package: plank (via r-universe)

November 26, 2026

Title Fast, Serverless Data Frame Viewer in a Single HTML File

Version 0.14.0

Description Browse a data frame or parquet file in the browser without a server. The data is embedded as compressed parquet inside one HTML file and read client-side, with a column picker, per-column filters (including regular expressions and inverted ``everything but'' filters), a SQL WHERE query bar, sorting, number formats, pinned columns, a row detail panel, global search, summary statistics, grouped counts, saved views and export to CSV, Excel, R data and parquet.

License MIT + file LICENSE

URL https://jossammil.github.io/plank/, https://github.com/jossammil/plank

BugReports https://github.com/jossammil/plank/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.2)

Imports htmltools, htmlwidgets, jsonlite, nanoparquet, tools, utils

Suggests askpass, openssl, rstudioapi, shiny, testthat (>= 3.0.0), zip

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev

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

Date/Publication 2026-11-20 22:18:33 UTC

RemoteUrl https://github.com/jossammil/plank

RemoteRef HEAD

RemoteSha a60018519bb239f6ba3501fd19b5c809e1743289

Contents

plank 2
plank-shiny 7
plank_view 8
save_plank 9

Index **11**

`plank` *View a data frame or parquet file in the browser*

Description

Creates an interactive table that runs entirely in the browser. The data is embedded as compressed parquet and decoded client-side, so the result needs no server: pick columns from a searchable list, filter each column, sort, search and export the filtered rows.

Usage

```
plank(
  data,
  columns = NULL,
  title = NULL,
  notes = NULL,
  digits = 7,
  where = NULL,
  sort = NULL,
  pin = NULL,
  formats = NULL,
  counts = NULL,
  views = NULL,
  theme = c("auto", "classic", "modern", "dark", "vintage", "starship", "terminal",
    "light"),
  font = c("mono", "sans", "serif"),
  controls = c("top", "bottom"),
  minimal = FALSE,
  password = NULL,
  top_offset = 0,
  rownames = NULL,
  compression = c("gzip", "snappy"),
  max_bytes = 5e+07,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

data	A data frame (or something as.data.frame() understands), or the path to a .parquet file. NULL makes a blank viewer that opens CSV, TSV, Excel and parquet files chosen or dropped by the reader (see **Opening files** below); only title, theme, font, controls, minimal and top_offset apply to it.
columns	Names of the columns shown at first. NULL shows the first ten; character(0) starts with none selected.
title	Optional title shown in the band above the toolbar. It is also used as the file name for CSV exports.
notes	Optional small text shown after the title, for notes such as the data source or date. Notes too long for the line are cut short, with a **more** button that shows them in full. The elements of a character vector are joined with " · " on the line, and shown one per line when the notes are shown in full.
digits	Significant digits used to display non-integer numbers. Exports always keep full precision.
where	Optional initial WHERE condition for the query bar, such as "age > 30 AND region IN ('West', 'South')". It is checked in the browser, which shows any syntax error.
sort	Optional columns to sort by at first, in order of priority. Prefix a name with - for descending order, as in c("region", "-income"). The columns must be among those shown.
pin	Optional columns to pin at first: they move to the left of the table and stay in view when scrolling sideways. They must be among those shown. Readers pin and unpin with the pin button in each header.
formats	Optional display formats for number columns: a named list with, for each column, decimals (a fixed number of decimal places) and big_mark (TRUE for a comma between thousands), as in list(income = list(decimals = 2, big_mark = TRUE)). Formats change only how numbers look; filters, search and exports use the full values. Readers can change them in each column's options.
counts	Optional one or two column names: the **Counts** panel opens with the filtered rows counted by these columns.
views	Optional named list of plank_view() objects, offered in the viewer's **Views** menu so readers can switch between prepared states of the table, as in list("Adults" = plank_view(where = "age >= 18", sort = "-age")).
theme	Starting colour theme: "auto" follows the reader's system light or dark setting; "classic", "modern", "dark", "vintage" (sepia on cream) and "starship" (a starship console: black, with orange and lilac panels) force one. "light" and "terminal" are old names for "classic" and "starship". Readers can switch in the Appearance window, and their choice is remembered in their browser.
font	Starting font for the table: "mono" (the default), "sans" or "serif". Readers can switch in the Appearance window, and their choice is remembered in their browser.
controls	Where the toolbar, WHERE bar and title band sit: "top" (the default) or "bottom", below the table. Readers can switch in the Appearance window, and their choice is remembered in their browser.

`minimal`	Start in the minimal view, with only the table showing (its column filters included)? Readers switch it with the corner button above the row numbers or the `M` key, and their choice is remembered.
`password`	Encrypt the data with this password (a string), or `TRUE` to be asked for it, so it does not sit in your script. See **Passwords** below. `NULL` (the default) leaves the data unencrypted.
`top_offset`	Extra space in pixels above the top band when the viewer fills a page of its own (a saved file or a browser tab), for example `50` where browser chrome such as Safari's full-screen toolbar would cover the toolbar. It is added to the small default gap; elsewhere it is not used.
`rownames`	Include row names as a first column? `NULL` includes them only when the rows are named (as in `mtcars`).
`compression`	How the embedded data is compressed. `"gzip"` (the default) makes files about 30% smaller than `"snappy"`, which is a little faster to write and to open. A parquet file given as `data` is embedded as it is when it uses gzip, snappy or no compression.
`max_bytes`	Warn when the embedded parquet data is larger than this many bytes. Everything is held in browser memory, so very large files will be slow to open.
`width, height`	Widget size. The default fills the page when viewed on its own and is 600px tall inside a document.
`elementId`	Optional element id for the widget.

Details

Printing the result shows it in the RStudio viewer or a browser. It can also be placed in R Markdown and Quarto documents, used in Shiny with `plankOutput()`, or written to one portable file with `save_plank()`.

Value

An htmlwidget.

In the viewer

- The band at the top shows the title and notes on one line (**more** shows long notes in full), the number of rows shown, the size of the embedded data and the estimated size of a CSV export of the current rows and columns.
- **Columns** opens a searchable list of every column with its type, distinct count, share missing, range and a small distribution chart (hover a bar for its counts). The chart darkens the part of each column that the current filters keep. The checkbox adds or removes a column; clicking the rest of an item shows its details (summary statistics and counts per value or range); dragging an item places it in the table (Alt+Up/Down also reorders). Drag the panel's right edge to resize it.
- A funnel appears next to a filtered column's name; click it to switch that filter off and on without losing it.

- Each header has a filter (range, checklist, text or TRUE/FALSE) and an options button for the match mode (contains, regular expression, exact), case sensitivity, "everything but" (invert, missing values included), missing values (show, hide, only) and how the filter combines with the others: AND filters must all match; a row must match at least one OR filter.
- **Counts** opens a panel with the number and share of filtered rows for each value of one or two columns (dates by day, month or year; numbers by value or in ranges), optionally with the mean, median, sum, minimum or maximum of a number column. It updates as the filters change. Click a row to show only that group, or **Exclude** to hide it; both add a condition to the WHERE query, so undo or editing the query reverses them. The counts can be copied or saved as CSV.
- **Views** saves the current columns, filters, search, WHERE query, sort and Counts settings under a name, in the reader's browser, and switches between saved views. Views passed from R with `views` are listed too, and **Copy as R code** writes `plank_view()` code for the current state.
- Click a row number to open the row panel at the right: every column of that row, shown or not, at full precision, with a box to find a column. While it is open, clicking any row shows that row; the arrow buttons (or the Up and Down keys) step through the filtered rows, and **Copy** copies the row as name and value lines.
- A number column's options (the sliders button) also set how its numbers are shown: a fixed number of decimals and a thousands separator.
- The pin button in a header (shown on hover) pins the column: pinned columns move to the left and stay in view, with the row numbers, when scrolling sideways.
- **Mark** adds a marker column and sets a value on every filtered row; clicking a marker cell sets one row. Marker columns can be filtered, sorted and queried, are saved in the reader's browser, and are included in exports together with a `row` column (the original row number) for joining back to the data in R.
- **WHERE** takes a SQL condition over any column, for example `comment REGEXP 'review|call' AND income IS NOT N` Click ? for the syntax. The toolbar's **WHERE** switch shows or hides this bar, and **See Query** shows the column filters, search and WHERE condition combined as one condition, to copy or edit.
- The undo and redo arrows step back and forward through changes to the columns, filters, search, query, sort and marker columns, for example to bring back a column (and its filter) that was unticked by mistake.
- **Export** saves the filtered rows and shown columns as CSV, Excel (.xlsx), R data (.rds, keeping factors, dates and time zones) or parquet, or copies them to the clipboard as tab-separated text.
- The search box has regular-expression (`.*`) and `NOT` toggles.
- The faint palette button at the right of the top band opens the Appearance window: a colour theme (Auto, Classic, Modern, Dark, Vintage or Starship), an accent colour, a text colour, the table font (mono, sans or serif), text size, row spacing and whether the controls sit above or below the table. Under **More options**: decor (coloured bands and filled buttons, Starship's own, which can be turned off or added to any theme), shapes (square, rounded or pill), the controls' font and contrast. The choices are remembered in the reader's browser.
- The faint ? button beside it says what Plank is and lists the keyboard shortcuts.
- The faint copy buttons in the column list copy column names as R code, `c("a", "b")`: those in the table, or those a search found.

- The WHERE bar starts closed; once a reader opens it or types a condition, it stays open for them next time.
- The corner button above the row numbers (or `M`) switches the minimal view, which folds away everything but the table.

Keyboard

Click a cell, or focus the table and press an arrow key, to get a cell cursor. Arrows, Home/End, Page Up/Down and Ctrl/Cmd+arrows move it; Alt/Option+Left/Right jumps to the left or right edge of the screen (again for the next screen); Esc in a text box (WHERE, search, a filter) returns to the table with the cursor where it was; Shift+arrows, Shift+click or dragging select a block of cells, whose count, sum and mean appear beside the row count; Ctrl/Cmd+C copies the block (Ctrl/Cmd+Shift+C with the column names) and Enter opens the row panel. Single keys jump to the tools when the cursor is not in a text box: `/` search, `W` WHERE, `C` columns, `K` Counts, `E` Export, `V` Views, `A` Appearance, `M` minimal view, `O` open a file, `G` go to a row and `?` lists them all. Ctrl/Cmd+K opens a command palette to run any action, go to or sort by any column, or switch views and themes by typing.

Passwords

With `password`, the data is encrypted in R (AES-256, with a key derived from the password by PBKDF2-SHA256 with 300,000 iterations, which takes a few seconds) and the viewer asks for the password before showing anything. Without it the file is unreadable, even to someone who opens its HTML source: the column names, factor levels, notes, query and views are encrypted with the data. Only the title stays readable, as the file's label. Keep the password safe: **a lost password cannot be recovered**. Once unlocked, readers can use the viewer as usual, exports included; marks and saved views of a protected file are kept only until the page is closed, so nothing about the data is left in the browser. Unlocking needs a secure page: a saved file, the RStudio viewer or an https address.

Opening files

O, the open button beside the palette button, or dropping files on the viewer opens CSV, TSV, Excel (.xlsx, first sheet) and parquet files as new tabs beside the data from R, listed along the foot of the viewer. The files are read in the reader's browser and never uploaded. Column types are guessed from CSV text (numbers, TRUE/FALSE, ISO dates and date-times; `NA`, `NULL` and empty cells are missing) and taken from the cell formats in Excel. Parquet files compressed with snappy, gzip or zstd can be opened. `plank(NULL)` (or `save_plank(NULL, "viewer.plank.html")`) makes a blank viewer for this.

Examples

```
if (interactive()) {
  plank(mtcars)
  plank(iris, columns = c("Species", "Sepal.Length"), title = "Iris",
        notes = "Fisher's iris data, 150 flowers")
  plank(mtcars, where = "mpg > 25 OR cyl = 8", font = "sans")
  plank(mtcars, sort = c("cyl", "-mpg"), counts = "cyl",
        views = list("Heavy" = plank_view(where = "wt > 4", sort = "-wt")))
}
```

plank-shiny *Shiny bindings for Plank*

Description

Output and render functions for using `plank()` in Shiny applications.

Usage

```
plankOutput(outputId, width = "100%", height = "600px")

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

Arguments

`outputId`	Output variable to read from.
`width, height`	Must be a valid CSS unit (like `"100%"` or `"600px"`).
`expr`	An expression that returns a `plank()` widget.
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression?

Details

The viewer reports what the reader is looking at as Shiny inputs, named after the output id:

- `input$<id>_rows`: the row numbers (of the data given to `plank()`) of the filtered rows, in the order shown.
- `input$<id>_state`: a list with `columns` (shown, in order), `sort` (`-` marks descending), `pinned`, `where` (every filter, the search and the WHERE query as one condition, as See Query shows it), `rows_shown` and `rows_total`.
- `input$<id>_row`: the row number open in the row panel, or `NULL`.

They update a quarter of a second after the reader stops changing things.

Value

`plankOutput()` returns a Shiny output element; `renderPlank()` returns a render function.

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  shiny::shinyApp(
    shiny::fluidPage(plankOutput("cars"), shiny::verbatimTextOutput("picked")),
    function(input, output) {
      output$cars <- renderPlank(plank(mtcars))
      output$picked <- shiny::renderPrint({
        rows <- unlist(input$cars_rows)
```

```
        c(rows = length(rows), mean_mpg = mean(mtcars$mpg[rows]))
      })
    }
  )
}
```

---

plank_view *Define a view for the viewer's Views menu*

Description

A view is a named state of the table that readers can switch to from the **Views** menu: which columns are shown and in what order, which are pinned, which rows are kept, how they are sorted, and optionally the **Counts** panel. Pass a named list of views to `plank()`'s `views` argument.

Usage

```
plank_view(
  columns = NULL,
  where = NULL,
  sort = NULL,
  counts = NULL,
  pin = NULL
)
```

Arguments

`columns`	Names of the columns shown, in order. `NULL` shows the same columns as the viewer starts with (`plank(columns = )`, or the first ten).
`where`	A WHERE condition in the viewer's query syntax, such as `"age > 30 AND region IN ('West', 'South')"`. It is checked in the browser when the view is used.
`sort`	Columns to sort by, in order of priority. Prefix a name with `-` to sort it in descending order, as in `c("region", "-income")`. The columns must be among those shown.
`counts`	One or two column names to count rows by in the **Counts** panel, which opens when the view is used. `NULL` leaves the panel as it is.
`pin`	Columns to pin: they move to the left of the table and stay in view when scrolling sideways. They must be among those shown.

Details

In the viewer, **Views > Copy as R code** writes this call for the current state, with the column filters and search turned into `where`.

Value

A `plank_view` object.

See Also

plank()

Examples

```
if (interactive()) {
  plank(mtcars,
    views = list(
      "Fuel-efficient" = plank_view(
        columns = c("mpg", "cyl", "hp", "wt"),
        where = "mpg > 25",
        sort = "-mpg"
      ),
      "By cylinders" = plank_view(counts = c("cyl", "gear"))
    )
  )
}
```

---

save_plank *Save a viewer as one self-contained HTML file*

Description

Writes a single `.html` file with the data, scripts and styles all inline. The file opens by double-click, can be emailed or put on a shared drive, and needs no server or internet connection.

Usage

```
save_plank(x, file, ..., page_title = NULL, open = FALSE, zip = FALSE)
```

Arguments

`x`	A `plank()` widget, or anything `plank()` accepts as `data`. `NULL` writes a blank viewer that opens the reader's own CSV, TSV, Excel and parquet files.
`file`	Path of the file to write. See **File names** below.
`...`	Passed to `plank()` when `x` is not already a widget.
`page_title`	Text for the browser tab. Defaults to the widget title, then the file name.
`open`	Open the file in a browser afterwards? With `zip = TRUE`, a copy of the HTML file in a temporary folder is opened.
`zip`	Write a zip archive holding the HTML file instead of the HTML file itself? See **Zipped files** below.

Details

Unlike `htmlwidgets::saveWidget()` with `selfcontained = TRUE`, this does not run pandoc, so it stays fast when the embedded data is large.

Value

The path to the file written (the zip archive with `zip = TRUE`), invisibly.

File names

Files from Plank are named `name.plank.html` by convention: the `.html` ending makes the file open in a browser on any computer, and the `.plank` before it makes plank files easy to find (search for `.plank.html`) and tell apart from other web pages. A `file` without an extension, or ending in `.plank` (or the older `.pqv`), gets `.plank.html` (or `.html`) added. A file name that does not end in `.html` or `.htm` gives a warning, because browsers show such a file as plain text instead of opening the viewer.

Zipped files

With `zip = TRUE` (or a `file` ending in `.zip`), the HTML file is put in a zip archive, `name.plank.zip`, which is smaller to email or upload. The embedded data is already compressed, so the zip file is about 30% smaller than the HTML file for a large table, and up to 70% smaller for a small one, where the viewer's own code is most of the file. Whoever receives it unzips it and double-clicks the `.plank.html` file inside. The archive is written with the **zip** package when it is installed, and otherwise with the `zip` command line program (see `utils::zip()`).

Examples

```
if (interactive()) {
  # Writes mtcars.plank.html in the temporary folder
  save_plank(mtcars, file.path(tempdir(), "mtcars"), title = "mtcars", open = TRUE)

  # Writes mtcars.plank.zip, holding mtcars.plank.html
  save_plank(mtcars, file.path(tempdir(), "mtcars"), zip = TRUE)
}
```

Index

as.data.frame(), *3*

htmlwidgets::saveWidget(), *9*

plank, 2
plank(), *7–9*
plank-shiny, 7
plank_view, 8
plank_view(), *3*, *5*
plankOutput *(*plank-shiny*)*, 7
plankOutput(), *4*

renderPlank *(*plank-shiny*)*, 7

save_plank, 9
save_plank(), *4*

utils::zip(), *10*