Package 'shinyloadtest'

February 5, 2026

Type Package

Title Load Test Shiny Applications

Version 1.2.1

Description Assesses the number of concurrent users 'shiny' applications
are capable of supporting, and for directing application changes in
order to support a higher number of users. Provides facilities for
recording 'shiny' application sessions, playing recorded sessions
against a target server at load, and analyzing the resulting metrics.

License GPL-3

URL https://rstudio.github.io/shinyloadtest/,

https://github.com/rstudio/shinyloadtest

BugReports https://github.com/rstudio/shinyloadtest/issues

Depends R (>= 2.10)

Imports cli, curl, dplyr (>= 1.0.0), ggplot2 (>= 3.4.0), httpuv (>=
1.5.2), jsonlite, magrittr, R6, rlang (>= 0.1.2), scales,
stringr, svglite, vroom, websocket (>= 1.0.0), xml2

Suggests getPass, glue, gtable, htmltools, lubridate, progress,
rmarkdown, spelling, testthat (>= 3.2.0)

Config/Needs/routine devtools, readr, httr, jsonlite, rprojroot,
usethis, stringr

Config/Needs/website devtools, httr, readr, gh, tidyverse/tidytemplate

Config/testthat/edition 3

Config/usethis/last-upkeep 2026-02-05

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

SystemRequirements pandoc (>= 2.2) - http://pandoc.org

Contents

---

load_runs   *Create Tidy Load Test Results*

Description

The shinycannon tool creates a directory of log files for each load test. This function translates one
or more test result directories into a tidy data frame.

Usage

```
load_runs(..., verbose = vroom::vroom_progress())
```

Arguments

...	Key-value pairs where the key is the desired name for the test and the value is a path to the test result directory.
verbose	Whether or not to print progress for reading loadtest directories

Value

A tidy data frame with the test result data. Each row is an event. Columns include identifiers and timing information for the event. The variables are as follows

run Name of the run

session_id simulated session identifier, 0-based

user_id simulated user identifier, 0-based

iteration user session identifier, 0-based

input_line_number recording line number associated with event

event type of the event

start time the event started, in seconds, relative to the time at which all simulated users were running.

end time the event ended, in seconds, relative to the time at which all simulated users were running

time event duration, in seconds

concurrency number of events that happened at the same time as this one

maintenance whether this event occurred before or after all simulated users were running

label event-specific text label

json raw message JSON and parsed JSON of the event

Output variables

- run: The name of the recording session.
- session_id: An incrementing integer value for every session within a run. Starts at 0.
- user_id: Which simulated user is performing the work within a run. Starts at 0.
- iteration: an incrementing integer value of the session iteration for the #' matching user_id. Starts at 0.
- input_line_number: The line number corresponding to the event in the recording.log file.
- event: the web event being performed. One of the following values:
    - REQ_HOME: initial request for to load the homepage
    - REQ_GET: Request a supporting file (JavaScript / CSS)
    - REQ_TOK: Request a Shiny token
    - REQ_SINF: Request SockJS information
    - REQ_POST: Perform a POST query, such as uploading part of a file
    - WS_RECV_BEGIN_UPLOAD: A file upload is being requested
    - WS_OPEN: Open a new SockJS connection
    - WS_RECV_INIT: Initialize a new SockJS
    - WS_SEND: Send information from the Shiny server to the browser
    - WS_RECV: Send information from the browser to the Shiny server
    - WS_CLOSE: Close the SockJS connection
- start: Start time of the event relative to the beginning of the run's maintenance period
- end: End time of the event relative to the beginning of the run's maintenance period

- time: Total elapsed time of the event
- concurrency: A number of events that are being processed concurrently
- maintenance: A boolean determining whether or not all simulated users are executing a session
- label: A human readable event name
- json: The parsed JSON provided in the recording.log file. If the field message exists, a message_parsed field is added containing a parsed form of the SockJS's JSON message content.

Examples

```
## Not run:
load_runs(
  `1 core` = "results/run-1/",
  `2 cores` = "results/run-2/"
)

## End(Not run)
```

---

record_session   *Record a Session for Load Test*

Description

This function creates a reverse proxy at http://host:port (http://127.0.0.1:8600 by default) that intercepts and records activity between your web browser and the Shiny application at target_app_url.

Usage

```
record_session(
  target_app_url,
  host = "127.0.0.1",
  port = 8600,
  output_file = "recording.log",
  open_browser = TRUE,
  connect_api_key = NULL
)
```

Arguments

target_app_url	The URL of the deployed application.
host	The host where the proxy will run. Usually localhost is used.
port	The port for the reverse proxy. Default is 8600. Change this default if port 8600 is used by another service.
output_file	The name of the generated recording file.
open_browser	Whether to open a browser on the proxy (default=TRUE) or not (FALSE).
connect_api_key	
	An Posit Connect api key. It may be useful to use Sys.getenv("CONNECT_API_KEY").

Details

By default, after creating the reverse proxy, a web browser is opened automatically. As you interact with the application in the web browser, activity is written to the output_file (recording.log by default).

To shut down the reverse proxy and complete the recording, close the web browser tab or window.

Recordings are used as input to the shinycannon command-line load-generation tool which can be obtained from the shinyloadtest documentation site.

Value

Creates a recording file that can be used as input to the shinycannon command-line load generation tool.

fileInput/DT/HTTP POST **support**

Shiny's shiny::fileInput() input for uploading files, the DT package, and potentially other packages make HTTP POST requests to the target application. Because POST requests can be large, they are not stored directly in the recording file. Instead, new files adjacent to the recording are created for each HTTP POST request intercepted.

The adjacent files are named after the recording with the pattern <output_file>.post.<N>, where <output_file> is the chosen recording file name and <N> is the number of the request.

If present, these adjacent files must be kept alongside the recording file when the recording is played back with the shinycannon tool.

See Also

shinyloadtest articles

Examples

```
## Not run:
record_session("https://example.com/your-shiny-app/")

## End(Not run)
```

---

shinyloadtest_report   *Make shinyloadtest Report*

Description

Make shinyloadtest Report

Usage

```
shinyloadtest_report(
  df,
  output = "shinyloadtest_report.html",
  duration_cutoff = c(attr(df, "recording_duration"), 60)[1],
  http_latency_cutoff = 5,
  max_websocket_cutoff = 20,
  open_browser = TRUE,
  self_contained = TRUE,
  verbose = TRUE
)
```

Arguments

df	data.frame returned from [load_runs()](#)
output	File where HTML output should be saved
duration_cutoff	
	Cutoff value for session duration plot. Defaults to the recording duration used to simulate df or 60 seconds.
http_latency_cutoff	
	Cutoff value for total http latency plot
max_websocket_cutoff	
	Cutoff value for max websocket latency plot
open_browser	Whether to open the created output in the browser
self_contained	Boolean that determines if the final output should be a self contained html file
verbose	Boolean that determines if progress output is displayed

Value

The path to the report, invisibly

Examples

```
## Not run:
shinyloadtest_report(slt_demo_data_1)

## End(Not run)
```

---

slt_demo_data_1   *Example metrics for a 1-user load test*

Description

An example dataset like that returned by [load_runs](#), but without the json variable for portability. Contains latency data for 1813 shinycannon events suitable for passing to [shinyloadtest_report](#).

Usage

```
slt_demo_data_1
```

Format

A data frame with 1813 rows and 12 variables:

run Name of the run

session_id simulated session identifier, 0-based

user_id simulated user identifier, 0-based

iteration user session identifier, 0-based

input_line_number recording line number associated with event

event type of the event

start time the event started, in seconds, relative to the time at which all simulated users were running.

end time the event ended, in seconds, relative to the time at which all simulated users were running

time event duration, in seconds

concurrency number of events that happened at the same time as this one

maintenance whether this event occurred before or after all simulated users were running

label event-specific text label

---

slt_demo_data_16 *Example metrics for a 16-user load test*

Description

An example dataset like that returned by [load_runs](#), but without the json variable for portability. Contains latency data for 5402 shinycannon events suitable for passing to [shinyloadtest_report](#).

Usage

```
slt_demo_data_16
```

Format

A data frame with 5402 rows and 12 variables:

run Name of the run

session_id simulated session identifier, 0-based

user_id simulated user identifier, 0-based

iteration user session identifier, 0-based

input_line_number recording line number associated with event

event type of the event

start time the event started, in seconds, relative to the time at which all simulated users were running.

end time the event ended, in seconds, relative to the time at which all simulated users were running

time event duration, in seconds

concurrency number of events that happened at the same time as this one

maintenance whether this event occurred before or after all simulated users were running

label event-specific text label

---

slt_demo_data_4   *Example metrics for a 4-user load test*

Description

An example dataset like that returned by [load_runs](#), but without the json variable for portability. Contains latency data for 4514 shinycannon events suitable for passing to [shinyloadtest_report](#).

Usage

```
slt_demo_data_4
```

Format

A data frame with 4514 rows and 12 variables:

run Name of the run

session_id simulated session identifier, 0-based

user_id simulated user identifier, 0-based

iteration user session identifier, 0-based

input_line_number recording line number associated with event

event type of the event

start time the event started, in seconds, relative to the time at which all simulated users were running.

end time the event ended, in seconds, relative to the time at which all simulated users were running

time event duration, in seconds

concurrency number of events that happened at the same time as this one

maintenance whether this event occurred before or after all simulated users were running

label event-specific text label

---

slt_plot   *Plotting outputs for shinyloadtest*

Description

Many different plotting routines to display different loadtest information.

Usage

```
slt_time_boxplot(df, labels = NULL)

slt_time_concurrency(df, labels = NULL)

slt_waterfall(df, limits = NULL)

slt_hist_loadtimes(df, max_load_time = 5)

slt_user(df)

slt_session(df)

slt_session_duration(df, cutoff = NULL)

slt_session_latency(df)

slt_http_latency(df, cutoff = 5)

slt_websocket_latency(df, cutoff = 5)
```

Arguments

df	data frame returned from [load_runs](#)
labels	A vector of labels to include. If none are supplied, all labels will be used.
limits	passed into [scale_colour_gradientn](#)
max_load_time	The amount of time users will wait for the page to load when first requesting the app.
cutoff	Where to draw a horizontal or vertical line to display a reasonable cutoff line for requests.

Value

A [ggplot](#) plot object

Functions

- `slt_time_boxplot()`: Box plot of load times for each event in each run
- `slt_time_concurrency()`: Time on concurrency for each event for each run
- `slt_waterfall()`: Event waterfall for each session within each run
- `slt_hist_loadtimes()`: Histogram of page load times
- `slt_user()`: Gantt chart of event duration for each user within each run
- `slt_session()`: Event gantt chart of each user session within each run
- `slt_session_duration()`: Event gantt chart of fastest to slowest session times within each run
- `slt_session_latency()`: Stacked bar chart of event duration for each session within each run
- `slt_http_latency()`: Bar chart of total HTTP latency for each session within each run
- `slt_websocket_latency()`: Bar chart of maximum calculation (websocket) latency for each session within each run

Examples

```
slt_user(slt_demo_data_4)
slt_session(slt_demo_data_4)
slt_session_duration(slt_demo_data_4)

slt_waterfall(slt_demo_data_4)
slt_time_boxplot(slt_demo_data_4)
slt_time_concurrency(slt_demo_data_4)

slt_session_latency(slt_demo_data_4)
slt_http_latency(slt_demo_data_4)
slt_websocket_latency(slt_demo_data_4)
slt_hist_loadtimes(slt_demo_data_4)
```

Index