

# Package ‘shinyloadtest’

February 11, 2021

**Type** Package

**Title** Load Test Shiny Applications

**Version** 1.1.0

**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

**Depends** R (>= 2.10)

**Encoding** UTF-8

**LazyData** true

**URL** <https://rstudio.github.io/shinyloadtest/>,  
<https://github.com/rstudio/shinyloadtest>

**BugReports** <https://github.com/rstudio/shinyloadtest/issues>

**Collate** 'enum.R' 'data.R' 'analysis.R' 'detect.R' 'auth.R'  
'make\_report.R' 'plotting.R' 'shiny-recorder.R' 'url.R'  
'util.R' 'util-pipe.R'

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

**Suggests** getPass, glue, gtable, htmltools, lubridate, progress,  
rmarkdown, testthat, spelling

**RoxygenNote** 7.1.1

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

**Language** en-US

**NeedsCompilation** no

**Author** Barret Schloerke [aut, cre] (<<https://orcid.org/0000-0001-9986-114X>>),  
Alan Dipert [aut],  
Barbara Borges [aut],  
RStudio [cph, fnd]

**Maintainer** Barret Schloerke <barret@rstudio.com>

**Repository** CRAN

**Date/Publication** 2021-02-11 14:50:02 UTC

## R topics documented:

load_runs . . . . .	2
record_session . . . . .	4
shinyloadtest_report . . . . .	5
slt_demo_data_1 . . . . .	6
slt_demo_data_16 . . . . .	7
slt_demo_data_4 . . . . .	8
slt_plot . . . . .	8

**Index** **11**

---

load_runs	<i>Create Tidy Load Test Results</i>
-----------	--------------------------------------

---

### 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	<i>Record a Session for Load Test</i>
----------------	---------------------------------------

---

## 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

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

## 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**

<code>df</code>	data.frame returned from <code>load_runs()</code>
<code>output</code>	File where HTML output should be saved
<code>duration_cutoff</code>	Cutoff value for session duration plot. Defaults to the recording duration used to simulate <code>df</code> or 60 seconds.
<code>http_latency_cutoff</code>	Cutoff value for total http latency plot
<code>max_websocket_cutoff</code>	Cutoff value for max websocket latency plot
<code>open_browser</code>	Whether to open the created output in the browser
<code>self_contained</code>	Boolean that determines if the final output should be a self contained html file
<code>verbose</code>	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 shynycannon 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 <a href="#">load_runs</a>
labels	A vector of labels to include. If none are supplied, all labels will be used.
limits	passed into <a href="#">scale_colour_gradientn</a>
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

## \* datasets

- slt\_demo\_data\_1, 6
- slt\_demo\_data\_16, 7
- slt\_demo\_data\_4, 8

ggplot, 9

load\_runs, 2, 6–9

load\_runs(), 6

record\_session, 4

scale\_colour\_gradientn, 9

shinyloadtest\_report, 5, 6–8

slt\_demo\_data\_1, 6

slt\_demo\_data\_16, 7

slt\_demo\_data\_4, 8

slt\_hist\_loadtimes (slt\_plot), 8

slt\_http\_latency (slt\_plot), 8

slt\_plot, 8

slt\_session (slt\_plot), 8

slt\_session\_duration (slt\_plot), 8

slt\_session\_latency (slt\_plot), 8

slt\_time\_boxplot (slt\_plot), 8

slt\_time\_concurrency (slt\_plot), 8

slt\_user (slt\_plot), 8

slt\_waterfall (slt\_plot), 8

slt\_websocket\_latency (slt\_plot), 8