Package 'reconstructr'

July 26, 2018

Type Package

Title Session Reconstruction and Analysis

Version 2.0.2

Date 2018-07-26

Author Oliver Keyes

Maintainer Oliver Keyes <ironholds@gmail.com>

Description Functions to reconstruct sessions from web log or other user trace data and calculate various metrics around them, producing tabular, output that is compatible with 'dplyr' or 'data.table' centered processes.

License MIT + file LICENSE

LinkingTo Rcpp

Imports Rcpp, openssl

Suggests testthat, knitr

VignetteBuilder knitr

URL https://github.com/Ironholds/reconstructr

BugReports https://github.com/Ironholds/reconstructr/issues

RoxygenNote 6.0.1

Depends R (>= 3.3.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-07-26 19:00:03 UTC

R topics documented:

`bounce_rate` *calculate the bounce rate within a session dataset*

Description

Calculates the "bounce rate" within a set of sessions - the proportion of sessions consisting only of a single event.

Usage

```
bounce_rate(sessions, user_id = NULL, precision = 2)
```

Arguments

`sessions`	a sessions dataset, presumably generated with `sessionise`.
`user_id`	a column that contains unique user IDs. NULL by default; if set, the assumption will be that you want *per-user* bounce rates.
`precision`	the number of decimal places to round the output to - set to 2 by default.

Value

either a single numeric value, representing the percentage of sessions *overall* that are bounces, or a data.frame of user IDs and bounce rates if `user_id` is set to a column rather than NULL.

See Also

`sessionise` for session reconstruction, and `session_length`, `session_count` and `time_on_page` for other session-related metrics.

Examples

```
#Load and sessionise the dataset
data("session_dataset")
sessions <- sessionise(session_dataset, timestamp, uuid)

# Calculate overall bounce rate
rate <- bounce_rate(sessions)

# Calculate bounce rate on a per-user basis
per_user <- bounce_rate(sessions, user_id = uuid)
```

reconstructr *functions for session reconstruction and analysis*

Description

sessionreconstruct provides functions to aid in reconstructing and analysing user sessions. Although primarily designed for web sessions (see the introductory vignette), its session approach is plausibly applicable to other domains.

Author(s)

Oliver Keyes <okeyes@wikimedia.org>

sessionise *Reconstruct sessions (experimental)*

Description

`sessionise` takes a data.frame of events (including timestamps and user IDs) and sessionises them, returning the same data.frame but with two additional columns - one containing a unique session ID, and one the time difference between successive events in the same session.

Usage

```
sessionise(x, timestamp, user_id, threshold = 3600)
```

Arguments

`x`	a data.frame of events.
`timestamp`	the name of the column of `x` containing timestamps, which should be (either) a representation of the number of seconds, or a POSIXct or POSIXlt date/time object. If it is neither, `strptime` can be used to convert most representations of date-times into POSIX formats.
`user_id`	the name of the column of `x` containing unique user IDs.
`threshold`	the number of seconds to use as the intertime threshold - the time that can elapse between two events before the second is considered part of a new session. Set to 3600 (one hour) by default.

Value

`x`, ordered by userID and timestamp, with two new columns - `session_id` (containing a unique ID for the session a row is in) and `delta` (containing the time elapsed between that row's event, and the previous event, if they were both in the same session).

See Also

bounce_rate, time_on_page, session_length and session_count - common metrics that can be calculated with a sessionised dataset.

Examples

```
# Take a dataset with URLs and similar metadata and sessionise it -
# retaining that metadata

data("session_dataset")
sessionised_data <- sessionise(x = session_dataset,
                               timestamp = timestamp,
                               user_id = uuid,
                               threshold = 1800)
```

---

session_count *Count the number of sessions in a sessionised dataset*

Description

link{session_count} counts the number of sessions in a sessionised dataset, producing either a count for the overall dataset or on a per-user basis (see below).

Usage

```
session_count(sessions, user_id = NULL)
```

Arguments

sessions	a dataset of sessions, presumably generated by sessionise
user_id	the column of sessions containing user IDs. If NULL (the default), a single count of sessions for the entire dataset will be generated. Otherwise, a data.frame of user IDs and the session count for each user ID will be returned.

Value

either a single integer value or a data.frame (see above).

Examples

```
#Load and sessionise the dataset
data("session_dataset")
sessions <- sessionise(session_dataset, timestamp, uuid)

# Calculate overall bounce rate
count <- session_count(sessions)
```

```
# Calculate session count on a per-user basis
per_user <- session_count(sessions, user_id = uuid)
```

`session_dataset` *Example event dataset*

Description

an example dataset of events, for experimenting with session reconstruction and analysis

Usage

```
session_dataset
```

Format

a data.frame of 63,524 rows consisting of:

uuid Hashed and salted unique identifiers representing 10,000 unique clients.

timestamp timestamps, as POSIXct objects

url URLs, to demonstrate the carrying-along of metadata through the sessionisation process

Source

The `uuid` and `timestamp` columns come from an anonymised dataset of Wikipedia readers; the URLs are from NASA's internal web server, because space is awesome.

`session_length` *Calculate session length*

Description

Calculate the overall length of each session.

Usage

```
session_length(sessions)
```

Arguments

`sessions`	a dataset of sessions, presumably generated with `sessionise`.

Value

a data.frame of two columns - `session_id`, containing unique session IDs, and `session_length`, containing the length (in seconds) of that particular session.

Please note that these lengths should be considered a *minimum*; because of how sessions behave, calculating the time-on-page of the last event in a session is impossible.

See Also

`sessionise` for session reconstruction, and `time_on_page`, `session_count` and `bounce_rate` for other session-related metrics.

Examples

```
#Load and sessionise the dataset
data("session_dataset")
sessions <- sessionise(session_dataset, timestamp, uuid)

# Calculate session length
len <- session_length(sessions)
```

---

time_on_page *Calculate time-on-page metrics*

Description

`time_on_page` generates metrics around the mean (or median) time-on-page - on an overall, per-user, or per-session basis.

Usage

```
time_on_page(sessions, by_session = FALSE, median = FALSE, precision = 2)
```

Arguments

`sessions`	a sessions dataset, presumably generated with `sessionise`.
`by_session`	Whether to generate time-on-page for the dataset overall (FALSE), or on a per-session basis (TRUE). FALSE by default.
`median`	whether to generate the median (TRUE) or mean (FALSE) time-on-page. FALSE by default.
`precision`	the number of decimal places to round the output to - set to 2 by default.

Value

either a single numeric value, representing the mean/median time on page for the overall dataset, or a data.frame of session IDs and numeric values if `by_session` is TRUE.

See Also

sessionise for session reconstruction, and session_length, session_count and bounce_rate for other session-related metrics.

Examples

```
#Load and sessionise the dataset
data("session_dataset")
sessions <- sessionise(session_dataset, timestamp, uuid)

# Calculate overall time on page
top <- time_on_page(sessions)

# Calculate time-on-page on a per_session basis
per_session <- time_on_page(sessions, by_session = TRUE)

# Use median instead of mean
top_med <- time_on_page(sessions, median = TRUE)
```

Index