Package: funneljoin (via r-universe)

September 11, 2024

Type Package

Title Time-Based Joins to Analyze Sequences of Events

Version 0.1.9000

Depends R (>= 2.10)

Maintainer Emily Robinson <robinson.es@gmail.com>

Description Time-based joins to analyze sequence of events, both in memory and out of memory. after_join() joins two tables of events, while funnel_start() and funnel_step() join events in the same table. With the type argument, you can switch between different funnel types, like first-first and last-firstafter.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, knitr, rmarkdown, tibble

RoxygenNote 7.2.1

Imports dplyr, glue, magrittr, broom, purrr, rlang, tidyr, methods, forcats

VignetteBuilder knitr

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

RemoteUrl https://github.com/robinsones/funneljoin

RemoteRef HEAD

RemoteSha 9e5cc4c037db4dca6e72c5ca0c1de87c46befaf0

Contents

`after_join` *Join tables based on one event happening after another*

Description

Join two tables based on observations in one table happening after observations in the other. Each table must have a user_id column, which must always match for two observations to be joined, and a time column, which must be greater in y than in x for the two to be joined. Supports all types of dplyr joins (inner, left, anti, etc.) and requires a type argument to specify which observations in a funnel get kept (see details for supported types).

Usage

```
after_join(
  x,
  y,
  by_time,
  by_user,
  mode = "inner",
  type = "first-first",
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)

after_inner_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)
```

```
after_left_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)

after_right_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)

after_full_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)

after_anti_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)
```

```
after_semi_join(
  x,
  y,
  by_time,
  by_user,
  type,
  max_gap = NULL,
  min_gap = NULL,
  gap_col = FALSE,
  suffix = c(".x", ".y")
)
```

Arguments

x	A tbl representing the first event to occur in the funnel.
y	A tbl representing an event to occur in the funnel.
by_time	A character vector to specify the time columns in x and y. This would typically be a datetime or a date column. These columns are used to filter for time y being after time x.
by_user	A character vector to specify the user or identity columns in x and y.
mode	The method used to join: "inner", "full", "anti", "semi", "right", "left". Each also has its own function, such as `after_inner_join`.
type	The type of funnel used to distinguish between event pairs, such as "first-first", "last-first", or "any-firstafter". See details for more.
max_gap	Optional: the maximum gap allowed between events. Can be a integer representing the number of seconds or a difftime object, such as `as.difftime(2, units = "hours")`.
min_gap	Optional: the maximum gap allowed between events. Can be a integer representing the number of seconds or a difftime object, such as `as.difftime(2, units = "hours")`.
gap_col	Whether to include a numeric column, `.gap`, with the time difference in seconds between the events.
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.

Details

`type` can be any combination of `first`, `last`, `any`, `lastbefore`, `firstwithin` with `first`, `last`, `any`, `firstafter`. Some common ones you may use include:

first-first Take the earliest x and y for each user **before** joining. For example, you want the first time someone entered an experiment, followed by the first time someone **ever** registered. If they registered, entered the experiment, and registered again, you do not want to include that person.

first-firstafter Take the first x, then the first y after that. For example, you want when someone first entered an experiment and the first course they started afterwards. You don't care if they started courses before entering the experiment.

lastbefore-firstafter First x that's followed by a y before the next x. For example, in last click paid ad attribution, you want the last time someone clicked an ad before the first subscription they did afterward.

any-firstafter Take all Xs followed by the first Y after it. For example, you want all the times someone visited a homepage and their first product page they visited afterwards.

any-any Take all Xs followed by all Ys. For example, you want all the times someone visited a homepage and **all** the product pages they saw afterward.

Examples

```
library(dplyr)
landed <- tribble(
  ~user_id, ~timestamp,
  1, "2018-07-01",
  2, "2018-07-01",
  2, "2018-07-01",
  3, "2018-07-02",
  4, "2018-07-01",
  4, "2018-07-04",
  5, "2018-07-10",
  5, "2018-07-12",
  6, "2018-07-07",
  6, "2018-07-08"
) %>%
  mutate(timestamp = as.Date(timestamp))

registered <- tribble(
  ~user_id, ~timestamp,
  1, "2018-07-02",
  3, "2018-07-02",
  4, "2018-06-10",
  4, "2018-07-02",
  5, "2018-07-11",
  6, "2018-07-10",
  6, "2018-07-11",
  7, "2018-07-07"
) %>%
 mutate(timestamp = as.Date(timestamp))

after_inner_join(landed, registered, by_user = "user_id",
           by_time = "timestamp", type = "first-first")

# You can use different methods of joining:
after_left_join(landed, registered, by_user = "user_id",
           by_time = "timestamp", type = "first-first")

after_anti_join(landed, registered, by_user = "user_id",
           by_time = "timestamp", type = "any-any")
```

`after_join_all` *View result for each type of afterjoin*

Description

View result for each type of afterjoin

Usage

```
after_join_all(x, y, by_user, by_time, mode = "inner", ...)
```

Arguments

`x`	A tbl representing the first event to occur in the funnel.
`y`	A tbl representing an event to occur in the funnel.
`by_user`	A character vector to specify the user or identity columns in x and y.
`by_time`	A character vector to specify the time columns in x and y. This would typically be a datetime or a date column. These columns are used to filter for time y being after time x.
`mode`	The method used to join: "inner", "full", "anti", "semi", "right", "left".
`...`	any additional arguments

`as_seconds` *Title*

Description

Title

Usage

```
as_seconds(x, sql = FALSE)
```

Arguments

`x`	a difftime object
`sql`	set to TRUE if you’re working with remote tables and using dbplyr

Value

a difftime object in seconds

`distinct_events` *Distinct events*

Description

Distinct events

Usage

```
distinct_events(.data, time_col, user_col, type)
```

Arguments

`.data`	a dataset, either local or remote
`time_col`	the name of the time column
`user_col`	the name of the user identifying column
`type`	the type of after_join ("first-first", "first-firstafter", etc.)

`funnel_start` *Start a funnel*

Description

Start a funnel

Usage

```
funnel_start(tbl, moment_type, moment, tstamp, user)
```

Arguments

`tbl`	A table of different moments and timestamps
`moment_type`	The first moment in the funnel
`moment`	The name of the column with the moment_type
`tstamp`	The name of the column with the timestamps of the moments
`user`	The name of the column indicating the user who did the moment

Examples

```
library(dplyr)

activity <- tibble::tribble(
  ~ "user_id", ~ "event", ~ "timestamp",
  1, "landing", "2019-07-01",
  1, "registration", "2019-07-02",
  1, "purchase", "2019-07-07",
  1, "purchase", "2019-07-10",
  2, "landing", "2019-08-01",
  2, "registration", "2019-08-15",
  3, "landing", "2019-05-01",
  3, "registration", "2019-06-01",
  3, "purchase", "2019-06-04",
  4, "landing", "2019-06-13")

activity %>%
  funnel_start(moment_type = "landing",
               moment = "event",
               tstamp = "timestamp",
               user = "user_id")
```

`funnel_step` *Continue to funnel*

Description

Continue to funnel

Usage

```
funnel_step(tbl, moment_type, type, name = moment_type, optional = FALSE, ...)

funnel_steps(tbl, moment_types, type, ...)
```

Arguments

`tbl`	A table of different moments and timestamps
`moment_type`	The next moment in the funnel
`type`	The type of after_join (e.g. "first-first", "any-any")
`name`	If you want a custom name instead of the moment_type; needed if the moment type is already in the sequence
`optional`	Whether this step in the funnel should be optional. If so, the following step will also try joining in a way that skips this step. Note that multiple optional steps in a row aren't supported.

`...`	Extra arguments passed on to after_left_join. For `funnel_steps`, these are passed on to `funnel_step`.
`moment_types`	For `funnel_steps`, a character vector of moment types, which are applied in order

Examples

```
library(dplyr)

activity <- tibble::tribble(
  ~ "user_id", ~ "event", ~ "timestamp",
  1, "landing", "2019-07-01",
  1, "registration", "2019-07-02",
  1, "purchase", "2019-07-07",
  1, "purchase", "2019-07-10",
  2, "landing", "2019-08-01",
  2, "registration", "2019-08-15",
  3, "landing", "2019-05-01",
  3, "registration", "2019-06-01",
  3, "purchase", "2019-06-04",
  4, "landing", "2019-06-13")

activity %>%
  funnel_start(moment_type = "landing",
               moment = "event",
               tstamp = "timestamp",
               user = "user_id")  %>%
funnel_step(moment_type = "registration",
           type = "first-firstafter")
```

`landed` *Example dataset of landing events*

Description

An example dataset for trying out after_join. The variables are as follows:

Usage

```
landed
```

Format

A data frame with 9 rows and 2 variables:

user_id A numeric column for identifying people

timestamp A date column for the date the landing happened

`reclass` *Copy class and attributes from the original version of an object to a modified version.*

Description

Copied over from https://github.com/tidyverse/dplyr/issues/719

Usage

```
reclass(x, result)
```

Arguments

`x`	The original object, which has a class/attributes to copy
`result`	The modified object, which is / might be missing the class/attributes.

Value

`result`, now with class/attributes restored.

`registered` *Example dataset of registration events*

Description

An example dataset for trying out after_join. The variables are as follows:

Usage

```
registered
```

Format

A data frame with 8 rows and 2 variables:

user_id A numeric column for identifying people

timestamp A date column for the date the registration happened

summarize_conversions *Summarize Left-joined table into conversion count*

Description

Summarize Left-joined table into conversion count

Usage

```
summarize_conversions(x, converted)
```

Arguments

`x`	A tbl with one row per user
`converted`	The name of the column representing whether the user converted (treated as FALSE if NA or FALSE, otherwise TRUE)

Value

A table with columns for your groups, along with 'nb_users', 'nb_conversions', and 'pct_converted'

summarize_funnel *Summarize after funnel start and funnel step(s)*

Description

Summarize after funnel start and funnel step(s)

Usage

```
summarize_funnel(tbl_funnel)
```

Arguments

`tbl_funnel`	a table from funnel start and funnel step(s)

Value

A tibble with one row for each moment_type and grouping variable, with columns:

nb_step The number of users who reached this moment

pct_cumulative The percentage of original users who reached this moment

pct_step The percentage of users who reached the last step reaching this moment summarize_prop_tests *Summarise after join funnel with proportion test*

Description

Summarise after join funnel with proportion test

Usage

```
summarize_prop_tests(
  x,
  alternative_name = alternative.name,
  base_level = "control",
  ...,
  ungroup = TRUE
)
```

Arguments

x	a data.frame with columns nb_conversions and nb_users
alternative_name	the name of the column indicating the experiment group
base_level	the name of the control experiment group
...	any additional arguments
ungroup	whether the table needs to be ungrouped

Value

a data.frame with proportion test results

Index