Package ‘sift’

July 5, 2021

Type Package
Title Intelligently Peruse Data
Version 0.1.0
Maintainer Scott McKenzie <sccmckenzie@gmail.com>
Description Facilitate extraction of key information from common datasets.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports pastecs (>= 1.3.21), stats, dplyr (>= 1.0.0), rlang (>= 0.4.3), tidyr (>= 1.0.0), tibble, purrr, glue, tidyselect
RoxygenNote 7.1.1
Suggests knitr, ggplot2, testthat (>= 3.0.0), rmarkdown, mopac, hms, stringr, readr
Config/testthat/edition 3
Depends R (>= 3.3.0)
VignetteBuilder knitr
LinkingTo cpp11
SystemRequirements C++11
NeedsCompilation yes
Author Scott McKenzie [aut, cre],
   RStudio [cph] (internal functions from dplyr.R)
Repository CRAN
Date/Publication 2021-07-05 09:10:02 UTC

R topics documented:

break_join ........................................ 2
comms ............................................ 3
conjecture ....................................... 3
break_join

Join tables based on overlapping intervals.

Description

User-friendly interface that synthesizes power of dplyr::left_join and findInterval.

Usage

break_join(x, y, brk = character(), by = NULL, ...)

Arguments

x A data frame.
y Data frame containing desired reference information.
brk Name of column in x and y to join by via interval overlapping. Must be coercible to numeric.
by Joining variables, if needed. See mutate-joins.
... additional arguments automatically directed to findInterval and dplyr::left_join. No partial matching.

Value

An object of the same type as x.

- All x rows will be returned.
- All columns between x and y are returned.
- Rows in y are matched with x based on overlapping values of brk (e.g. findInterval(x$brk, y$brk, ...)).

Examples

# joining USA + UK leaders with population time-series
break_join(us_uk_pop, us_uk_leaders, brk = c(\text{\texttt{date}} = \text{\texttt{\texttt{start}}}))

# simple dataset
set.seed(1)
a <- data.frame(p = c(rep\("\text{\texttt{A}}", 10), rep\("\text{\texttt{B}}", 10))
q = runif\((20, 0, 10))
b <- data.frame(p = c\("\text{\texttt{A}}", \text{\texttt{A}}, \text{\texttt{B}}, \text{\texttt{B}}),
q = c\((3, 5, 6, 9), r = c\("\text{\texttt{a1}}, \text{\texttt{a2}}, \text{\texttt{b1}}, \text{\texttt{b2}}))
break_join(a, b, brk = \"q\") # p identified as common variable automatically
break_join(a, b, brk = \"q\", by = \"p\") # same result
```
break_join(a, b, brk = "q", all.inside = TRUE) # note missing values have been filled

# joining toll prices with vehicle time-series
library(mopac)
library(dplyr, warn.conflicts = FALSE)
library(hms)

express %>%
  mutate(time_hms = as_hms(time)) %>%
  break_join(rates, brk = c("time_hms" = "time"))
```
kluster

Automatically cluster 1-dimensional continuous data.

Usage

kluster(x, bw = "SJ", fixed = FALSE)

Arguments

x Vector to be clustered. Must contain at least 1 non-missing value.
bw kernel bandwidth. Default "SJ" should suffice more application, however you can supply a custom numeric value. See ?stats::density for more information.
fixed logical; if TRUE, performs simple 1-dimensional clustering without kernel density estimation. default FALSE.

Value

An integer vector identifying the cluster corresponding to each element in x.
Examples

# Below vector clearly has 2 groups.
# kluster will identify these groups using kernel density estimation.
kuster(c(0.1, 0.2, 1))

# kluster shines in cases where manually assigning groups via "eyeballing" is impractical.
# Suppose we obtained vector 'x' without knowing how it was generated.
set.seed(1)
nodes <- runif(10, min = 0, max = 100)
x <- lapply(nodes, function(x) rnorm(10, mean = x, sd = 0.1))
x <- unlist(x)

kuster(x) # kluster reveals the natural grouping

kuster(x, bw = 10) # adjust bandwidth depending on application

# Example with faithful dataset
faithful$k <- kuster(faithful$eruptions)
library(ggplot2)
ggplot(faithful, aes(eruptions)) +
  geom_density() +
  geom_rug(aes(color = factor(k))) +
  theme_minimal() +
  scale_color_discrete(name = "k")

Description

Includes selected headlines and additional metadata for NYT articles throughout 2020. This dataset is not a comprehensive account of all major events from 2020.

Usage

nyt2020

Format

A data frame with 1,830 rows and 6 variables:

headline  Article Headline
abstract  Brief summary of article
byline  Contributing Writers
pub_date  Date of Publication
section_name  NYT section in which article was published
web_url  Article URL ...
sift

Augmented data frame filtering.

Description

Imagine `dplyr::filter` that includes neighboring observations. Choose how many observations to include by adjusting inputs `sift.col` and `scope`.

Usage

```r
sift(.data, sift.col, scope, ...)
```

Arguments

- `.data` A data frame.
- `sift.col` Column name, as symbol, to serve as "sifting/augmenting" dimension. Must be non-missing and coercible to numeric.
- `scope` Specifies augmentation bandwidth relative to "key" observations. Parameter should share the same scale as `sift.col`.
  - If length 1, bandwidth used is +/- `scope`.
- `...` Expressions passed to `dplyr::filter`, of which the results serve as the "key" observations. The same data-masking rules used in `dplyr::filter` apply here.

Details

`sift()` can be understood as a 2-step process:

1. `.data` is passed to `dplyr::filter`, using subsetting expression(s) provided in `...`. We'll refer to these intermediate results as "key" observations.
2. For each key observation, `sift` expands the row selection bidirectionally along dimension specified by `sift.col`. Any row from the original dataset within `scope` units of a key observation is captured in the final result.

Essentially, this allows us to "peek" at neighboring rows surrounding the key observations.

Value

A sifted data frame, with 2 additional columns:

- `.cluster <int>`: Identifies resulting group formed by each key observation and its neighboring rows. When the key observations are close enough together, the clusters will overlap.
- `.key <lgl>`: TRUE indicates key observation.
### Examples

- See current events from same timeframe as 2020 Utah Monolith discovery.
  ```r
  sift(nyt2020, pub_date, scope = 2, grepl("Monolith", headline))
  ```

- or Biden’s presidential victory.
  ```r
  sift(nyt2020, pub_date, scope = 2, grepl("Biden is elected", headline))
  ```

- We can specify lower & upper scope to see what happened AFTER Trump tested positive.
  ```r
  sift(nyt2020, pub_date, scope = c(0, 2), grepl("Trump Tests Positive", headline))
  ```

- sift recognizes dplyr group specification.
  ```r
  library(dplyr)
  library(mopac)
  express %>%
    group_by(direction) %>%
    sift(time, 30, plate == "EAS-1671") # row augmentation performed within groups.
  ```

### us_uk_pop

**Fragments of US & UK population & leaders**

### Description

These datasets are intended to demonstrate usage of `sift::break_join`.

### Usage

- `us_uk_pop`

- `us_uk_leaders`

### Source

See `tidyr::who` and `ggplot2::presidential`. 
Index

* datasets
  comms, 3
  nyt2020, 5
  us_uk_pop, 7

break_join, 2

comms, 3
conjecture, 3

filter, 6

kluster, 4

mutate-joins, 2

nyt2020, 5

presidential, 7

sift, 6

us_uk_leaders(us_uk_pop), 7
us_uk_pop, 7

who, 7