

Package: runner (via r-universe)

October 4, 2026

Title Running Operations for Vectors

Type Package

Version 0.5.0

Depends R (>= 3.0)

Language en-US

Encoding UTF-8

Maintainer Dawid Kałedkowski <dawid.kaledkowski@gmail.com>

Description Lightweight library for rolling windows operations.

Package enables full control over the window length, window lag and a time indices. With a runner one can apply any R function on a rolling windows. The package eases work with equally and unequally spaced time series.

License GPL (>= 2)

BugReports <https://github.com/gogonzo/runner/issues>

LinkingTo Rcpp

Imports methods, parallel, Rcpp

Suggests knitr, rmarkdown, tinytest

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository <https://gogonzo.r-universe.dev>

Date/Publication 2026-06-06 22:19:47 UTC

RemoteUrl <https://github.com/gogonzo/runner>

RemoteRef HEAD

RemoteSha 4697b70f73feacc50e66cd0b384198370e2d9f28

Contents

fill_run	2
lag_run	3
length_run	3
max_run	4
mean_run	5
min_run	6
minmax_run	8
run_by	8
runner	10
streak_run	17
sum_run	18
which_run	19
window_run	20
Index	22

fill_run	<i>Fill NA with previous non-NA element</i>
----------	---

Description

Fill NA with last non-NA element.

Usage

```
fill_run(x, run_for_first = FALSE, only_within = FALSE)
```

Arguments

- x (vector, data.frame, matrix, xts, grouped_df) input data.
- run_for_first If first elements are filled with NA, run_for_first = TRUE allows to fill all initial NA with nearest non-NA value. By default run_for_first = FALSE.
- only_within NA are replaced only if previous and next non-NA values are the same. By default only_within = FALSE.

Value

vector - x containing all x elements with NA replaced with previous non-NA element.

Examples

```
fill_run(c(NA, NA, 1:10, NA, NA), run_for_first = TRUE)
fill_run(c(NA, NA, 1:10, NA, NA), run_for_first = FALSE)
fill_run(c(NA, NA, 1, 2, NA, NA, 2, 2, NA, NA, 1, NA, NA), run_for_first = TRUE, only_within = TRUE)
```

lag_run	<i>Lag dependent on variable</i>
---------	----------------------------------

Description

Vector of input lagged along integer vector

Usage

```
lag_run(x, lag = 1L, idx = integer(0), nearest = FALSE)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
nearest	logical single value. Applied when idx is used, then nearest = FALSE returns observation lagged exactly by the specified number of "periods". When nearest = TRUE function returns latest observation within lag window.

Examples

```
lag_run(1:10, lag = 3)
lag_run(letters[1:10], lag = -2, idx = c(1, 1, 1, 2, 3, 4, 6, 7, 8, 10))
lag_run(letters[1:10], lag = 2, idx = c(1, 1, 1, 2, 3, 4, 6, 7, 8, 10), nearest = TRUE)
```

length_run	<i>Length of running windows</i>
------------	----------------------------------

Description

Number of elements in k-long window calculated on idx vector. If idx is an as.integer(date) vector, then k=number of days in window - then the result is number of observations within k days window.

Usage

```
length_run(k = integer(1), lag = integer(1), idx = integer(0))
```

Arguments

k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.

Examples

```
length_run(k = 3, idx = c(1, 2, 2, 4, 5, 5, 5, 5, 5, 5))
```

max_run	<i>Running maximum</i>
---------	------------------------

Description

max_run calculates running max on given x numeric vector, specified k window size.

Usage

```
max_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_rm = TRUE,
  na_pad = FALSE
)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.

idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

max (numeric) vector of length equals length of x.

Examples

```
set.seed(11)
x1 <- sample( c(1,2,3), 15, replace=TRUE)
x2 <- sample( c(NA,1,2,3), 15, replace=TRUE)
k <- sample( 1:4, 15, replace=TRUE)
max_run(x1) # simple cumulative maximum
max_run(x2, na_rm = TRUE) # cumulative maximum with removing NA.
max_run(x2, na_rm = TRUE, k=4) # maximum in 4-element window
max_run(x2, na_rm = FALSE, k=k) # maximum in varying k window size
```

mean_run	<i>Running mean</i>
----------	---------------------

Description

Running mean in specified window of numeric vector.

Usage

```
mean_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_rm = TRUE,
  na_pad = FALSE
)
```

Arguments

x	numeric vector which running function is calculated on
k	(integer`` vector or single value)\cr Denoting size of the running window. If k is a single value then window size is constant for all elements, otherwise if length(k) == length(x) different window size for each element.
lag	(integer vector or single value) Denoting window lag. If lag is a single value then window lag is constant for all elements, otherwise if length(lag) == length(x) different window size for each element. Negative value shifts window forward.
idx	(integer, Date, POSIXt) Optional integer vector containing sorted (ascending) index of observation. By default idx is index incremented by one. User can provide index with varying increment and with duplicated values. If specified then k and lag are depending on idx. Length of idx have to be equal of length x.
at	(integer, Date, POSIXt, character vector) Vector of any size and any value defining output data points. Values of the vector defines the indexes which data is computed at.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

mean (numeric) vector of length equals length of x.

Examples

```
set.seed(11)
x1 <- rnorm(15)
x2 <- sample(c(rep(NA,5), rnorm(15)), 15, replace = TRUE)
k <- sample(1:15, 15, replace = TRUE)
mean_run(x1)
mean_run(x2, na_rm = TRUE)
mean_run(x2, na_rm = FALSE )
mean_run(x2, na_rm = TRUE, k=4)
```

min_run	<i>Running minimum</i>
---------	------------------------

Description

min_run calculates running min on given x numeric vector, specified k window size.

Usage

```
min_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_rm = TRUE,
  na_pad = FALSE
)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

min (numeric) vector of length equals length of x.

Examples

```
set.seed(11)
x1 <- sample(c(1, 2, 3), 15, replace = TRUE)
x2 <- sample(c(NA, 1, 2, 3), 15, replace = TRUE)
k <- sample(1:4, 15, replace = TRUE)
min_run(x1)
min_run(x2, na_rm = TRUE)
min_run(x2, na_rm = TRUE, k = 4)
min_run(x2, na_rm = FALSE, k = k)
```

minmax_run

Running min/max

Description

min_run calculates running minimum-maximum on given x numeric vector, specified k window size.

Usage

```
minmax_run(x, metric = "min", na_rm = TRUE)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
metric	character what to return, minimum or maximum
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.

Value

list.

run_by

Set window parameters

Description

Set window parameters for [runner\(\)](#). This function sets the attributes to x (only data.frame) object and saves user effort to specify window parameters in further multiple [runner\(\)](#) calls.

Usage

```
run_by(x, idx, k, lag, na_pad, at)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.

k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.

Value

x object which `runner()` can be executed on.

Examples

```
## Not run:
library(dplyr)

data <- data.frame(
  index = c(2, 3, 3, 4, 5, 8, 10, 10, 13, 15),
  a = rep(c("a", "b"), each = 5),
  b = 1:10
)

data %>%
  group_by(a) %>%
  run_by(idx = "index", k = 5) %>%
  mutate(
    c = runner(
      x = .,
      f = function(x) {
        paste(x$b, collapse = ">")
      }
    ),
    d = runner(
      x = .,
      f = function(x) {
        sum(x$b)
      }
    )
  )

## End(Not run)
```

runner

*Apply function on running windows***Description**

Applies any R function `f` on running (Column/sliding) windows defined by `k`, `lag`, `idx` and `at`.

Usage

```
runner(
  x,
  f = function(x) x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_pad = FALSE,
  simplify = TRUE,
  cl = NULL,
  ...
)

## Default S3 method:
runner(
  x,
  f = function(x) x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_pad = FALSE,
  simplify = TRUE,
  cl = NULL,
  ...
)

## S3 method for class 'data.frame'
runner(
  x,
  f = function(x) x,
  k = attr(x, "k"),
  lag = if (!is.null(attr(x, "lag"))) attr(x, "lag") else integer(1),
  idx = attr(x, "idx"),
  at = attr(x, "at"),
  na_pad = if (!is.null(attr(x, "na_pad"))) attr(x, "na_pad") else FALSE,
  simplify = TRUE,
  cl = NULL,
```

```

    ...
)

## S3 method for class 'grouped_df'
runner(
  x,
  f = function(x) x,
  k = attr(x, "k"),
  lag = if (!is.null(attr(x, "lag"))) attr(x, "lag") else integer(1),
  idx = attr(x, "idx"),
  at = attr(x, "at"),
  na_pad = if (!is.null(attr(x, "na_pad"))) attr(x, "na_pad") else FALSE,
  simplify = TRUE,
  cl = NULL,
  ...
)

## S3 method for class 'matrix'
runner(
  x,
  f = function(x) x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_pad = FALSE,
  simplify = TRUE,
  cl = NULL,
  ...
)

## S3 method for class 'xts'
runner(
  x,
  f = function(x) x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_pad = FALSE,
  simplify = TRUE,
  cl = NULL,
  ...
)

```

Arguments

x (vector, data.frame, matrix, xts, grouped_df)
input data.

f	(function) Function applied to each window. Defaults to identity (function(x) x).
k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.
simplify	(logical or character) Simplify result like <code>base::sapply()</code> . TRUE returns vector/matrix, "array" may return a higher-dimensional array.
cl	(cluster) <i>experimental</i> Parallel cluster from <code>parallel::makeCluster()</code> .
...	additional arguments passed to f.

Details

Window types:

Sliding window (k):

k sets the number of elements in each window. When k is a single constant, the window slides along the data with a fixed size. The first k-1 windows are shorter since there aren't enough preceding elements.

k = 4

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |##|##|##|##|   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
                <----->
                window at i=10

```

```

# 4-element sliding sum
runner(1:15, k = 4, f = sum)

```

If k is omitted, windows are cumulative — each window grows from the first element to the current one, similar to `cumsum()`.

k not specified (cumulative)

```

i= 1:|#|                window [1, 1]
i= 2:|##|               window [1, 2]
i= 3:|###|              window [1, 3]
i= 4:|####|             window [1, 4]
...
i=15:|#####|           window [1, 15]

```

cumulative sum (k omitted)

```
runner(1:15, f = sum)
```

k can also be a vector of length(x) to use a different window size at each position.

Lag (lag):

lag shifts the window backward (positive values) or forward (negative values) relative to the current element. Default is lag = 0. Like k, lag can be a single value or a vector of length(x).

k = 4, lag = 2

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | | | |##|##|##|##| | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
          <-----> ^
          window    i=10 (shifted by lag=2)

```

```
runner(1:15, k = 4, lag = 2, f = sum)
```

Index-based windows (idx):

By default, runner treats elements as equally spaced (index increments by 1). Real data often has gaps — missing weekends, holidays, irregular timestamps. Setting idx makes k and lag refer to index distance instead of element count, so the number of elements per window varies with the spacing.

For example, a 5-day window (k = 5) on unevenly-spaced dates will contain different numbers of observations at each step:

k = 5, lag = 1

```
idx:  4   6   7  13  17  18  18  21  27  31  37  42  44  47  48
```

```

4: [-2,  3] NA      (no data in range)
6: [ 1,  5] =       {4}
7: [ 2,  6] ==      {4, 6}
13: [ 8, 12] NA     (no data in range)
17: [12, 16] =      {13}
18: [13, 17] ==     {13, 17}
18: [13, 17] ==     {13, 17, 18}
21: [16, 20] ===    {17, 18, 18}
27: [22, 26] NA     (no data in range)
31: [26, 30] =      {27}
37: [32, 36] NA     (no data in range)
42: [37, 41] =      {37}
44: [39, 43] ==     {42, 44}
47: [42, 46] ==     {42, 44}
48: [43, 47] ===    {44, 47}

```

k and lag also accept time-interval strings using the same syntax as seq.POSIXt(by = ...), e.g. "5 days", "2 weeks", "month".

```
idx <- c(4, 6, 7, 13, 17, 18, 18, 21, 27, 31, 37, 42, 44, 47, 48)
runner(idx, k = 5, lag = 1, idx = idx, f = mean)
```

```
# equivalent with Date index
runner(idx, k = "5 days", lag = "day", idx = Sys.Date() + idx, f = mean)
```

Evaluation at specific points (at):

By default, runner returns one result per element of x. Setting at restricts evaluation to specific index positions — the output length equals length(at). This is useful when you only need results at certain dates or milestones, not at every observation.

```
k = 5, lag = 1, at = c(18, 27, 48, 31)
idx:  4   6   7  13  17  18  18  21  27  31  37  42  44  47  48
```

```
at=18: [13, 17] ==      {13, 17}
at=27: [22, 26] NA      (no data in range)
at=48: [43, 47] ===     {44, 47}
at=31: [26, 30] =       {27}
```

```
runner(1:15, k = 5, lag = 1, idx = idx, at = c(18, 27, 48, 31), f = mean)
```

at can also be a single time-interval string, which generates a regular sequence over the idx range. For example, at = "4 months" evaluates at every 4-month interval from min(idx) to max(idx).

Time-interval syntax:

k, lag and at accept time-interval strings (requires idx to be set) using the same syntax as the by argument in `base::seq.POSIXt()`: "sec", "min", "hour", "day", "week", "month", "quarter", "year", or "<n> <unit>s" (e.g. "5 days", "-2 weeks"). Both k and lag can also be vectors, allowing different window sizes and shifts at each position.

Parallel computing:

Pass a `parallel::makeCluster()` object via cl to run windows in parallel. Objects referenced inside f (other than its arguments) must be exported with `parallel::clusterExport()` beforehand. Parallel execution adds overhead and is only beneficial for expensive computations.

Value

vector with aggregated values for each window. Length equals length(x) or length(at) if specified. Type depends on f.

Examples

```
# runner returns windows as is by default
runner(1:10)

# mean on k = 3 elements windows
runner(1:10, f = mean, k = 3)

# mean on k = 3 elements windows with different specification
runner(1:10, k = 3, f = function(x) mean(x, na.rm = TRUE))

# concatenate two columns
```

```

runner(
  data.frame(
    a = letters[1:10],
    b = 1:10
  ),
  f = function(x) paste(paste0(x$a, x$b), collapse = "+")
)

# concatenate two columns with additional argument
runner(
  data.frame(
    a = letters[1:10],
    b = 1:10
  ),
  f = function(x, xxx) {
    paste(paste0(x$a, xxx, x$b), collapse = " + ")
  },
  xxx = "...")
)

# number of unique values in each window (varying window size)
runner(letters[1:10],
  k = c(1, 2, 2, 4, 5, 5, 5, 5, 5, 5),
  f = function(x) length(unique(x))
)

# concatenate only on selected windows index
runner(letters[1:10],
  f = function(x) paste(x, collapse = "-"),
  at = c(1, 5, 8)
)

# 5 days mean
idx <- c(4, 6, 7, 13, 17, 18, 18, 21, 27, 31, 37, 42, 44, 47, 48)
runner::runner(
  x = idx,
  k = "5 days",
  lag = 1,
  idx = Sys.Date() + idx,
  f = function(x) mean(x)
)

# 5 days mean at 4-indices
runner::runner(
  x = 1:15,
  k = 5,
  lag = 1,
  idx = idx,
  at = c(18, 27, 48, 31),
  f = mean
)

# runner with data.frame

```

```

df <- data.frame(
  a = 1:13,
  b = 1:13 + rnorm(13, sd = 5),
  idx = seq(as.Date("2022-02-22"), as.Date("2023-02-22"), by = "1 month")
)
runner(
  x = df,
  idx = "idx",
  at = "6 months",
  f = function(x) {
    cor(x$a, x$b)
  }
)

# parallel computing
library(parallel)
data <- data.frame(
  a = runif(100),
  b = runif(100),
  idx = cumsum(sample(rpois(100, 5)))
)
const <- 0
cl <- makeCluster(1)
clusterExport(cl, "const", envir = environment())

runner(
  x = data,
  k = 10,
  f = function(x) {
    cor(x$a, x$b) + const
  },
  idx = "idx",
  cl = cl
)
stopCluster(cl)

# Panel data: rolling mean within groups
# (equivalent to Stata's: bysort firm (year): rolling mean(x), window(3)
library(dplyr)
df <- data.frame(
  firm = rep(c("A", "B"), each = 5),
  year = rep(2010:2014, 2),
  sales = c(100, 110, 125, 130, 145, 200, 215, 230, 250, 270)
)
df %>%
  arrange(firm, year) %>%
  group_by(firm) %>%
  mutate(roll_mean = runner(sales, f = mean, k = 3)) %>%
  ungroup()

# runner with matrix
data <- matrix(data = runif(100, 0, 1), nrow = 20, ncol = 5)
runner(

```

```

x = data,
f = function(x) {
  tryCatch(
    cor(x),
    error = function(e) NA
  )
}
)

```

streak_run

Running streak length

Description

Calculates running series of consecutive elements

Usage

```

streak_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_rm = TRUE,
  na_pad = FALSE
)

```

Arguments

x	any type vector which running function is calculated on
k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

streak [numeric](#) vector of length equals length of x containing number of consecutive occurrences.

Examples

```
set.seed(11)
x1 <- sample(c("a","b"), 15, replace = TRUE)
x2 <- sample(c(NA_character_, "a", "b"), 15, replace = TRUE)
k <- sample(1:4, 15, replace = TRUE)
streak_run(x1) # simple streak run
streak_run(x1, k = 2) # streak run within 2-element window
streak_run(x2, na_pad = TRUE, k = 3) # streak run within k=3 with padding NA
streak_run(x1, k = k) # streak run within varying window size specified by vector k
```

sum_run	<i>Running sum</i>
---------	--------------------

Description

Running sum in specified window of numeric vector.

Usage

```
sum_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_rm = TRUE,
  na_pad = FALSE
)
```

Arguments

x	numeric vector which running function is calculated on
k	(integer`` vector or single value)\cr Denoting size of the running window. If k is a single value then window size is constant for all elements, otherwise if length(k) == length(x) different window size for each element.
lag	(integer vector or single value) Denoting window lag. If lag is a single value then window lag is constant for all elements, otherwise if length(lag) == length(x) different window size for each element. Negative value shifts window forward.
idx	(integer, Date, POSIXt) Optional integer vector containing sorted (ascending) index of observation. By default idx is index incremented by one. User can provide index with varying increment and with duplicated values. If specified then k and lag are depending on idx. Length of idx have to be equal of length x.

at	(integer, Date, POSIXt, character vector) Vector of any size and any value defining output data points. Values of the vector defines the indexes which data is computed at.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

sum (numeric) vector of length equals length of x.

Examples

```
set.seed(11)
x1 <- rnorm(15)
x2 <- sample(c(rep(NA, 5), rnorm(15)), 15, replace = TRUE)
k <- sample(1:15, 15, replace = TRUE)
sum_run(x1)
sum_run(x2, na_rm = TRUE)
sum_run(x2, na_rm = FALSE)
sum_run(x2, na_rm = TRUE, k = 4)
```

which_run	<i>Running which</i>
-----------	----------------------

Description

which_run calculates running which - returns index of element where x == TRUE.

Usage

```
which_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  which = "last",
  na_rm = TRUE,
  na_pad = FALSE
)
```

Arguments

x	(vector, data.frame, matrix, xts, grouped_df) input data.
k	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when idx is set.
lag	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when idx is set.
idx	(integer, Date, POSIXt) Sorted index of observations. When set, k and lag refer to index distance rather than element count. Must be same length as x.
at	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of idx.
which	character value "first" or "last" denoting if the first or last TRUE index is returned from the window.
na_rm	logical single value (default na_rm = TRUE) - if TRUE sum is calculating excluding NA.
na_pad	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

integer vector of indexes of the same length as x.

Examples

```
set.seed(11)
x1 <- sample(c(1, 2, 3), 15, replace = TRUE)
x2 <- sample(c(NA, 1, 2, 3), 15, replace = TRUE)
k <- sample(1:4, 15, replace = TRUE)
which_run(x1)
which_run(x2, na_rm = TRUE)
which_run(x2, na_rm = TRUE, k = 4)
which_run(x2, na_rm = FALSE, k = k)
```

window_run

List of running windows

Description

Creates list of windows with given arguments settings. Length of output list is equal

Usage

```

window_run(
  x,
  k = integer(0),
  lag = integer(1),
  idx = integer(0),
  at = integer(0),
  na_pad = FALSE
)

```

Arguments

<code>x</code>	(vector, data.frame, matrix, xts, grouped_df) input data.
<code>k</code>	(integer or character) Window size. Single value or vector of length(x). Omit for cumulative windows. Accepts time-interval strings (e.g. "5 days") when <code>idx</code> is set.
<code>lag</code>	(integer or character) Window shift. Positive shifts back, negative shifts forward. Single value or vector of length(x). Accepts time-interval strings when <code>idx</code> is set.
<code>idx</code>	(integer, Date, POSIXt) Sorted index of observations. When set, <code>k</code> and <code>lag</code> refer to index distance rather than element count. Must be same length as <code>x</code> .
<code>at</code>	(integer, Date, POSIXt, character) Indices at which to evaluate windows. Output length equals length(at) instead of length(x). A single time-interval string (e.g. "month") generates a regular sequence over the range of <code>idx</code> .
<code>na_pad</code>	(logical) If TRUE, return NA for windows that extend beyond the data range.

Value

list of vectors (windows). Length of list is the same as length(x) or length(at) if specified, and length of each window is defined by `k` (unless window is out of range).

Examples

```

window_run(1:10, k = 3, lag = -1)
window_run(letters[1:10], k = c(1, 2, 2, 4, 5, 5, 5, 5, 5, 5))

```

Index

`base::sapply()`, [12](#)
`base::seq.POSIXt()`, [14](#)

`cumsum()`, [12](#)

`fill_run`, [2](#)

`lag_run`, [3](#)
`length_run`, [3](#)

`max_run`, [4](#)
`mean_run`, [5](#)
`min_run`, [6](#)
`minmax_run`, [8](#)

`numeric`, [18](#)

`parallel::clusterExport()`, [14](#)
`parallel::makeCluster()`, [12](#), [14](#)

`run_by`, [8](#)
`runner`, [10](#)
`runner()`, [8](#), [9](#)

`streak_run`, [17](#)
`sum_run`, [18](#)

`which_run`, [19](#)
`window_run`, [20](#)