Package 'sparklyr.flint'

January 11, 2022

Type Package

Title Sparklyr Extension for 'Flint'

Version 0.2.2

Maintainer Edgar Ruiz <edgar@rstudio.com>

Description This sparklyr extension makes 'Flint' time series library functionalities (<https://github.com/twosigma/flint>) easily accessible through R.

License Apache License 2.0

URL <https://github.com/r-spark/sparklyr.flint>

BugReports https://github.com/r-spark/sparklyr.flint/issues

Depends R (>= 3.2)

Imports dbplyr, dplyr, rlang, sparklyr (>= 1.3)

Suggests knitr, rmarkdown, tibble

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.1

SystemRequirements Spark: 2.x or above

Collate 'imports.R' 'globals.R' 'sdf_utils.R' 'asof_join.R' 'init.R' 'window_exprs.R' 'summarizers.R' 'ols_regression.R' 'reexports.R' 'utils.R'

NeedsCompilation no

Author Yitao Li [aut] (<https://orcid.org/0000-0002-1261-905X>), Edgar Ruiz [aut, cre]

Repository CRAN

Date/Publication 2022-01-11 08:50:13 UTC

R topics documented:

asof_future_left_join *Temporal future left join*

Description

Perform left-outer join on 2 'TimeSeriesRDD's based on inexact timestamp matches, where each record from 'left' with timestamp 't' matches the record from 'right' having the most recent timestamp at or after 't' if 'strict_lookahead' is FALSE (default) or having the most recent timestamp strictly after 't' if 'strict_lookahead' is TRUE. Notice this is equivalent to 'asof_join()' with 'direction' = ">=" if 'strict_lookahead' is FALSE (default) or direction '>' if 'strict_lookahead' is TRUE. See asof_join.

Usage

```
asof_future_left_join(
  left,
  right,
  tol = "0ms",
  key_columns = list(),
  left_prefix = NULL,
  right_prefix = NULL,
  strict_lookahead = FALSE
)
```

Arguments

left	The left 'TimeSeriesRDD'
right	The right 'TimeSeriesRDD'
tol	A character vector specifying a time duration (e.g., "0ns", "5ms", "5s", "1d", etc) as the tolerance for absolute difference in timestamp values between each record from 'left' and its matching record from 'right'. By default, 'tol' is "0ns", which means a record from 'left' will only be matched with a record from 'right' if both contain the exact same timestamps.
key_columns	Columns to be used as the matching key among records from 'left' and 'right': if non-empty, then in addition to matching criteria imposed by timestamps, a record from 'left' will only match one from the 'right' only if they also have equal values in all key columns.
left_prefix	A string to prepend to all columns from 'left' after the join (usually for disambiguation purposes if 'left' and 'right' contain overlapping column names).
right_prefix	A string to prepend to all columns from 'right' after the join (usually for disambiguation purposes if 'left' and 'right' contain overlapping column names).
strict_lookahead	Whether each record from 'left' with timestamp 't' should match record from 'right' with the smallest timestamp strictly greater than 't' (default: FALSE)

See Also

Other Temporal join functions: asof_join(), asof_left_join()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")
if (!is.null(sc)) {
  ts_1 <- copy_to(sc, tibble::tibble(t = seq(10), u = seq(10))) %>%
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_2 <- copy_to(sc, tibble::tibble(t = seq(10) + 1, v = seq(10) + 1L)) %>%
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  future_left_join_ts <- asof_future_left_join(ts_1, ts_2, tol = "1s")
} else {
  message("Unable to establish a Spark connection!")
}
```

---

asof_join *Temporal join*

Description

Perform left-outer join on 2 'TimeSeriesRDD's based on inexact timestamp matches

Usage

```
asof_join(
  left,
  right,
  tol = "0ms",
  direction = c(">=", "<=", "<"),
  key_columns = list(),
  left_prefix = NULL,
  right_prefix = NULL
)
```

Arguments

left	The left 'TimeSeriesRDD'
right	The right 'TimeSeriesRDD'
tol	A character vector specifying a time duration (e.g., "0ns", "5ms", "5s", "1d", etc) as the tolerance for absolute difference in timestamp values between each record from 'left' and its matching record from 'right'. By default, 'tol' is "0ns", which means a record from 'left' will only be matched with a record from 'right' if both contain the exact same timestamps.

direction — Specifies the temporal direction of the join, must be one of ">=", "<=", or "<". If direction is ">=", then each record from ‘left‘ with timestamp ‘tl‘ gets joined with a record from ‘right‘ having the largest/most recent timestamp ‘tr‘ such that ‘tl‘ >= ‘tr‘ and ‘tl‘ - ‘tr‘ <= ‘tol‘ (or equivalently, 0 <= ‘tl‘ - ‘tr‘ <= ‘tol‘). If direction is "<=", then each record from ‘left‘ with timestamp ‘tl‘ gets joined with a record from ‘right‘ having the smallest/least recent timestamp ‘tr‘ such that ‘tl‘ <= ‘tr‘ and ‘tr‘ - ‘tl‘ <= ‘tol‘ (or equivalently, ‘0 <= ‘tr‘ - ‘tl‘ <= ‘tol‘). If direction is "<", then each record from ‘left‘ with timestamp ‘tl‘ gets joined with a record from ‘right‘ having the smallest/least recent timestamp ‘tr‘ such that ‘tr‘ > ‘tl‘ and ‘tr‘ - ‘tl‘ <= ‘tol‘ (or equivalently, 0 < ‘tr‘ - ‘tl‘ <= ‘tol‘).

key_columns — Columns to be used as the matching key among records from ‘left‘ and ‘right‘: if non-empty, then in addition to matching criteria imposed by timestamps, a record from ‘left‘ will only match one from the ‘right‘ only if they also have equal values in all key columns.

left_prefix — A string to prepend to all columns from ‘left‘ after the join (usually for disambiguation purposes if ‘left‘ and ‘right‘ contain overlapping column names).

right_prefix — A string to prepend to all columns from ‘right‘ after the join (usually for disambiguation purposes if ‘left‘ and ‘right‘ contain overlapping column names).

See Also

Other Temporal join functions: `asof_future_left_join()`, `asof_left_join()`

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")
if (!is.null(sc)) {
  ts_1 <- copy_to(sc, tibble::tibble(t = seq(10), u = seq(10))) %>%
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_2 <- copy_to(sc, tibble::tibble(t = seq(10) + 1, v = seq(10) + 1L)) %>%
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  future_left_join_ts <- asof_join(ts_1, ts_2, tol = "1s", direction = "<=")
} else {
  message("Unable to establish a Spark connection!")
}
```

`asof_left_join` *Temporal left join*

Description

Perform left-outer join on 2 'TimeSeriesRDD's based on inexact timestamp matches, where each record from 'left' with timestamp 't' matches the record from 'right' having the most recent timestamp at or before 't'. Notice this is equivalent to 'asof_join()' with 'direction' = "<=". See asof_join.

Usage

```
asof_left_join(
  left,
  right,
  tol = "0ms",
  key_columns = list(),
  left_prefix = NULL,
  right_prefix = NULL
)
```

Arguments

left	The left 'TimeSeriesRDD'
right	The right 'TimeSeriesRDD'
tol	A character vector specifying a time duration (e.g., "0ns", "5ms", "5s", "1d", etc) as the tolerance for absolute difference in timestamp values between each record from 'left' and its matching record from 'right'. By default, 'tol' is "0ns", which means a record from 'left' will only be matched with a record from 'right' if both contain the exact same timestamps.
key_columns	Columns to be used as the matching key among records from 'left' and 'right': if non-empty, then in addition to matching criteria imposed by timestamps, a record from 'left' will only match one from the 'right' only if they also have equal values in all key columns.
left_prefix	A string to prepend to all columns from 'left' after the join (usually for disambiguation purposes if 'left' and 'right' contain overlapping column names).
right_prefix	A string to prepend to all columns from 'right' after the join (usually for disambiguation purposes if 'left' and 'right' contain overlapping column names).

See Also

Other Temporal join functions: asof_future_left_join(), asof_join()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")
if (!is.null(sc)) {
  ts_1 <- copy_to(sc, tibble::tibble(t = seq(10), u = seq(10))) %>%
```

```
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_2 <- copy_to(sc, tibble::tibble(t = seq(10) + 1, v = seq(10) + 1L)) %>%
    from_sdf(is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  left_join_ts <- asof_left_join(ts_1, ts_2, tol = "1s")
} else {
  message("Unable to establish a Spark connection!")
}
```

`collect.ts_rdd` *Collect data from a TimeSeriesRDD*

Description

Collect data from a TimeSeriesRDD into a R data frame

Usage

```
## S3 method for class 'ts_rdd'
collect(x, ...)
```

Arguments

`x`	A com.twosigma.flint.timeseries.TimeSeriesRDD object
`...`	Additional arguments to ‘sdf_collect()‘

Value

A R data frame containing the same time series data the input TimeSeriesRDD contains

See Also

Other Spark dataframe utility functions: `from_rdd()`, `from_sdf()`, `spark_connection.ts_rdd()`, `spark_dataframe.ts_rdd()`, `spark_jobj.ts_rdd()`, `to_sdf()`, `ts_rdd_builder()`

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- from_sdf(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  df <- ts %>% collect()
} else {
  message("Unable to establish a Spark connection!")
}
```

from_rdd *Construct a TimeSeriesRDD from a Spark RDD of rows*

Description

Construct a TimeSeriesRDD containing time series data from a Spark RDD of rows

Usage

```
from_rdd(
  rdd,
  schema,
  is_sorted = FALSE,
  time_unit = .sparklyr.flint.globals$kValidTimeUnits,
  time_column = .sparklyr.flint.globals$kDefaultTimeColumn
)

fromRDD(
  rdd,
  schema,
  is_sorted = FALSE,
  time_unit = .sparklyr.flint.globals$kValidTimeUnits,
  time_column = .sparklyr.flint.globals$kDefaultTimeColumn
)
```

Arguments

rdd	A Spark RDD[Row] object containing time series data
schema	A Spark StructType object containing schema of the time series data
is_sorted	Whether the rows being imported are already sorted by time
time_unit	Time unit of the time column (must be one of the following values: "NANOSECONDS", "MICROSECONDS", "MILLISECONDS", "SECONDS", "MINUTES", "HOURS", "DAYS"
time_column	Name of the time column

Value

A TimeSeriesRDD useable by the Flint time series library

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Other Spark dataframe utility functions: collect.ts_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  rdd <- spark_dataframe(sdf) %>% invoke("rdd")
  schema <- spark_dataframe(sdf) %>% invoke("schema")
  ts <- from_rdd(
    rdd, schema,
    is_sorted = TRUE, time_unit = "SECONDS", time_column = "t"
  )
} else {
  message("Unable to establish a Spark connection!")
}
```

---

from_sdf *Construct a TimeSeriesRDD from a Spark DataFrame*

Description

Construct a TimeSeriesRDD containing time series data from a Spark DataFrame

Usage

```
from_sdf(
  sdf,
  is_sorted = FALSE,
  time_unit = .sparklyr.flint.globals$kValidTimeUnits,
  time_column = .sparklyr.flint.globals$kDefaultTimeColumn
)

fromSDF(
  sdf,
  is_sorted = FALSE,
  time_unit = .sparklyr.flint.globals$kValidTimeUnits,
  time_column = .sparklyr.flint.globals$kDefaultTimeColumn
)
```

Arguments

sdf	A Spark DataFrame object
is_sorted	Whether the rows being imported are already sorted by time

time_unit	Time unit of the time column (must be one of the following values: "NANOSECONDS", "MICROSECONDS", "MILLISECONDS", "SECONDS", "MINUTES", "HOURS", "DAYS"
time_column	Name of the time column

Value

A TimeSeriesRDD useable by the Flint time series library

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- from_sdf(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
} else {
  message("Unable to establish a Spark connection!")
}
```

---

init *Dependencies and initialization procedures*

Description

Functions in this file specify all runtime dependencies of sparklyr.flint and package-wide constants in '.sparklyr.flint.globals'.

`ols_regression` *OLS regression*

Description

Ordinary least squares regression

Usage

```
ols_regression(
  ts_rdd,
  formula,
  weight = NULL,
  has_intercept = TRUE,
  ignore_const_vars = FALSE,
  const_var_threshold = 1e-12
)
```

Arguments

`ts_rdd`	Timeseries RDD containing dependent and independent variables
`formula`	An object of class "formula" (or one that can be coerced to that class) which symbolically describes the model to be fitted, with the left-hand-side being the column name of the dependent variable, and the right-hand-side being column name(s) of independent variable(s) delimited by '+', e.g., 'mpg ~ hp + weight + am' for predicting 'mpg' based on 'hp', 'weight' and 'am'
`weight`	Name of the weight column if performing a weighted OLS regression, or NULL if otherwise. Default: NULL.
`has_intercept`	Whether to include an intercept term (default: TRUE). If FALSE, then the resulting regression plane will always pass through the origin.
`ignore_const_vars`	Whether to ignore independent variables that are constant or nearly constant based on const_threshold (default: FALSE). If TRUE, the scalar fields of regression result are the same as if the constant variables are not included as independent variables. The output beta, tStat, stdErr columns will still have the same dimension number of elements as the number of independent variables. However, entries corresponding to independent variables that are considered constant will have 0.0 for beta and stdErr; and Double.NaN for tStat. If FALSE and at least one independent variable is considered constant, the regression will output Double.NaN for all values. Note that if there are multiple independent variables that can be considered constant and if the resulting model should have an intercept term, then it is recommended to set both ignore_const_vars and has_intercept to TRUE.
`const_var_threshold`	Consider an independent variable 'x' as constant if ((number of observations) * variance(x)) is less than this value. Default: 1e-12.

Value

A TimeSeries RDD with the following schema: * - "samples": [[LongType]], the number of samples * - "beta": [[ArrayType]] of [[DoubleType]], beta without the intercept component * - "intercept": [[DoubleType]], the intercept * - "hasIntercept": [[BooleanType]], whether the model has an intercept term * - "stdErr_intercept": [[DoubleType]], the standard error of the intercept * - "stdErr_beta": [[ArrayType]] of [[DoubleType]], the standard error of beta * - "rSquared": [[DoubleType]], the r-squared statistics * - "r": [[DoubleType]], the squared root of r-squared statistics * - "tStat_intercept": [[DoubleType]], the t-stats of the intercept * - "tStat_beta": [[ArrayType]] of [[DoubleType]], the t-stats of beta * - "logLikelihood": [[DoubleType]], the log-likelihood of the data given the fitted betas * - "akaikeIC": [[DoubleType]], the Akaike information criterion * - "bayesIC": [[DoubleType]], the Bayes information criterion * - "cond": [[DoubleType]], the condition number of the Gram matrix X^TX where X is the matrix formed by row vectors of independent variables (including a constant entry corresponding to the intercept if ‘has_intercept‘ is TRUE) * - "const_columns": [[ArrayType]] of [[StringType]], the list of independent variables that are considered constants

See Also

Other summarizers: summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  mtcars_sdf <- copy_to(sc, mtcars, overwrite = TRUE) %>%
    dplyr::mutate(time = 0L)
  mtcars_ts <- from_sdf(mtcars_sdf, is_sorted = TRUE, time_unit = "SECONDS")
  model <- ols_regression(
    mtcars_ts, mpg ~ cyl + disp + hp + drat + wt + vs + am + gear + carb
  ) %>%
      collect()
} else {
  message("Unable to establish a Spark connection!")
}
```

sdf_utils *Utility functions for importing a Spark data frame into a TimeSeriesRDD*

Description

These functions provide an interface for specifying how a Spark data frame should be imported into a TimeSeriesRDD (e.g., which column represents time, whether rows are already ordered by time, and time unit being used, etc)

Arguments

sc	Spark connection
is_sorted	Whether the rows being imported are already sorted by time
time_unit	Time unit of the time column (must be one of the following values: "NANOSECONDS", "MICROSECONDS", "MILLISECONDS", "SECONDS", "MINUTES", "HOURS", "DAYS"
time_column	Name of the time column

spark_connection *Retrieve Spark connection associated with an R object*

Description

See spark_connection for more details.

spark_connection.ts_rdd *Retrieve Spark connection associated with an R object*

Description

See spark_connection for more details.

Usage

```
## S3 method for class 'ts_rdd'
spark_connection(x, ...)
```

Arguments

x	An R object from which a 'spark_connection' can be obtained.
...	Optional arguments; currently unused.

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  print(spark_connection(ts))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

spark_dataframe *Retrieve a Spark DataFrame*

Description

See spark_dataframe for more details.

---

spark_dataframe.ts_rdd *Retrieve a Spark DataFrame*

Description

Retrieve a Spark DataFrame from a TimeSeriesRDD object

Usage

```
## S3 method for class 'ts_rdd'
spark_dataframe(x, ...)
```

Arguments

x	An R object wrapping, or containing, a Spark DataFrame.
...	Optional arguments; currently unused.

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_jobj.ts_rdd(), to_sdf(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- from_sdf(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  print(ts %>% spark_dataframe())
  print(sdf %>% spark_dataframe()) # the former should contain the same set of
                                   # rows as the latter does, modulo possible
                                   # difference in types of timestamp columns
} else {
  message("Unable to establish a Spark connection!")
}
```

---

spark_jobj *Retrieve a Spark JVM Object Reference*

Description

See spark_jobj for more details.

---

spark_jobj.ts_rdd *Retrieve a Spark JVM Object Reference*

Description

See spark_jobj for more details.

Usage

```
## S3 method for class 'ts_rdd'
spark_jobj(x, ...)
```

Arguments

x	An R object containing, or wrapping, a 'spark_jobj'.
...	Optional arguments; currently unused.

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), to_sdf(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  print(spark_jobj(ts))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarizers *Wrapper functions for commonly used summarizer functions*

Description

R wrapper functions for commonly used Flint summarizer functionalities such as sum and count.

Arguments

ts_rdd	Timeseries RDD being summarized
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
column	Column to be summarized
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE

and ‘key_columns‘ is empty, then for each record in ‘ts_rdd‘, the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and ‘key_columns‘ is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

`summarize_avg` *Average summarizer*

Description

Compute moving average of ‘column‘ and store results in a new column named ‘<column>_mean‘

Usage

```
summarize_avg(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

`ts_rdd`	Timeseries RDD being summarized
`column`	Column to be summarized
`window`	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps
`key_columns`	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: `ols_regression()`, `summarize_corr2()`, `summarize_corr()`, `summarize_count()`, `summarize_covar()`, `summarize_dot_product()`, `summarize_ema_half_life()`, `summarize_ewma()`, `summarize_geometric_mean()`, `summarize_kurtosis()`, `summarize_max()`, `summarize_min()`, `summarize_nth_central_moment()`, `summarize_nth_moment()`, `summarize_product()`, `summarize_quantile()`, `summarize_skewness()`, `summarize_stddev()`, `summarize_sum()`, `summarize_var()`, `summarize_weighted_avg()`, `summarize_weighted_corr()`, `summarize_weighted_covar()`, `summarize_z_score()`

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_avg <- summarize_avg(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

`summarize_corr` *Correlation summarizer*

Description

Compute pairwise correations among the list of columns specified and store results in new columns named with the following pattern: '<column1>_<column2>_correlation' and '<column1>_<column2>_correlationTStat', where column1 and column2 are names of any 2 distinct columns

Usage

```
summarize_corr(ts_rdd, columns, key_columns = list(), incremental = FALSE)
```

Arguments

`ts_rdd`	Timeseries RDD being summarized
`columns`	A list of column names
`key_columns`	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
`incremental`	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1

or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), u = rnorm(10), v = rnorm(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_corr <- summarize_corr(ts, columns = c("u", "v"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_corr2 *Pairwise correlation summarizer*

Description

Compute pairwise correations for all possible pairs of columns such that the first column of each pair is one of ‘xcolumns‘ and the second column of each pair is one of ‘ycolumns‘, storing results in new columns named with the following pattern: ‘<column1>_<column2>_correlation‘ and ‘<column1>_<column2>_correlationTStat‘ for each pair of columns (column1, column2)

Usage

```
summarize_corr2(
  ts_rdd,
  xcolumns,
  ycolumns,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
xcolumns	A list of column names
ycolumns	A list of column names disjoint from xcolumns
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(
    sc,
   tibble::tibble(t = seq(10), x1 = rnorm(10), x2 = rnorm(10), y1 = rnorm(10), y2 = rnorm(10))
  )
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_corr2 <- summarize_corr2(ts, xcolumns = c("x1", "x2"), ycolumns = c("y1", "y2"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_count *Count summarizer*

Description

Count the total number of records if no column is specified, or the number of non-null values within the specified column within each time window or within each group of records with identical timestamps

Usage

```
summarize_count(ts_rdd, column = NULL, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	If not NULL, then report the number of values in the column specified that are not NULL or NaN within each time window or group of records with identical timestamps, and store the counts in a new column named '<column>_count'. Otherwise the number of records within each time window or group of records with identical timestamps is reported, and stored in a column named 'count'.
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2

records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_count <- summarize_count(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_covar *Covariance summarizer*

Description

Compute covariance between values from ‘xcolumn‘ and ‘ycolumn‘ within each time window or within each group of records with identical timestamps, and store results in a new column named ‘<xcolumn>_<ycolumn>_covariance‘

Usage

```
summarize_covar(ts_rdd, xcolumn, ycolumn, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
xcolumn	Column representing the first random variable
ycolumn	Column representing the second random variable
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), u = rnorm(10), v = rnorm(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_covar <- summarize_covar(ts, xcolumn = "u", ycolumn = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

summarize_dot_product *Dot product summarizer*

Description

Compute dot product of values from ‘xcolumn‘ and ‘ycolumn‘ within a moving time window or within each group of records with identical timestamps and store results in a new column named ‘<xcolumn>_<ycolumn>_dotProduct‘

Usage

```
summarize_dot_product(
  ts_rdd,
  xcolumn,
  ycolumn,
  window = NULL,
  key_columns = list()
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
xcolumn	Name of the first column
ycolumn	Name of the second column
window	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(),

summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), u = seq(10, 1, -1), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
 ts_dot_product <- summarize_dot_product(ts, xcolumn = "u", ycolumn = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_ema_half_life

EMA half-life summarizer

Description

Calculate the exponential moving average of a time series using the half- life specified and store the result in a new column named '<column>_ema' See https://github.com/twosigma/flint/blob/master/doc/ema.md for details on different EMA implementations.

Usage

```
summarize_ema_half_life(
  ts_rdd,
  column,
  half_life_duration,
  window = NULL,
  time_column = "time",
  interpolation = c("previous", "linear", "current"),
  convention = c("legacy", "convolution", "core"),
  key_columns = list()
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized

half_life_duration
: A time duration specified in string form (e.g., "1d", "1h", "15m", etc) representing the half-life duration

window
: Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize the EMA of ‘column‘ within the time interval of [t - 1h, t] for each timestamp ‘t‘, ‘in_future("5s")‘ to summarize EMA of ‘column‘ within the time interval of [t, t + 5s] for each timestamp ‘t‘), or ‘NULL‘ to summarize EMA of ‘column‘ within the time interval of (-inf, t] for each timestamp ‘t‘

time_column
: Name of the column containing timestamps (default: "time")

interpolation
: Method used for interpolating values between two consecutive data points, must be one of "previous", "linear", and "current" (default: "previous"). See https://github.com/twosigma/flint/b for details on different interpolation methods.

convention
: Convolution convention, must be one of "convolution", "core", and "legacy" (default: "legacy"). See https://github.com/twosigma/flint/blob/master/doc/ema.md for details.

key_columns
: Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ewma(), summarize_geometric_mea summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  price_sdf <- copy_to(
    sc,
    data.frame(time = seq(1000), price = rnorm(1000))
  )
  ts <- fromSDF(price_sdf, is_sorted = TRUE, time_unit = "SECONDS")
  ts_ema <- summarize_ema_half_life(
    ts,
    column = "price",
```

```
    half_life_duration = "100s"
  )
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_ewma *Exponential weighted moving average summarizer*

Description

Compute exponential weighted moving average (EWMA) of ‘column‘ and store results in a new column named ‘<column>_ewma‘ At time t[n], the i-th value x[i] with timestamp t[i] will have a weighted value of [weight(i, n) * x[i]], where weight(i, n) is determined by both ‘alpha‘ and ‘smoothing_duration‘.

Usage

```
summarize_ewma(
  ts_rdd,
  column,
  alpha = 0.05,
  smoothing_duration = "1d",
  time_column = "time",
  convention = c("core", "legacy"),
  key_columns = list()
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
alpha	A smoothing factor between 0 and 1 (default: 0.05) – a higher alpha discounts older observations faster
smoothing_duration	A time duration specified in string form (e.g., "1d", "1h", "15m", etc) or "constant". The weight applied to a past observation from time t[p] at time t[n] is jointly determined by ‘alpha‘ and ‘smoothing_duration‘. If ‘smoothing_duration‘ is a fixed time duration such as "1d", then weight(p, n) = (1 - alpha) ^ [(t[n] - t[p]) / smoothing_duration] If ‘smoothing_duration‘ is "constant", then weight(p, n) = (1 - alpha) ^ (n - p) (i.e., this option assumes the difference between consecutive timestamps is equal to some constant ‘diff‘, and ‘smoothing_duration‘ is effectively also equal to ‘diff‘, so that t[n] - t[p] = (n - p) * diff and weight(p, n) = (1 - alpha) ^ [(t[n] - t[p]) / smoothing_duration] = (1 - alpha) ^ [(n - p) * diff / diff] = (1 - alpha) ^ (n - p))

time_column	Name of the column containing timestamps (default: "time")
convention	One of "core" or "legacy" (default: "core") If ‘convention‘ is "core", then the output will be weighted sum of all observations divided by the sum of all weight coefficients (see https://github.com/twosigma/flint/blob/master/doc/ If ‘convention‘ is "legacy", then the output will simply be the weighted sum of all observations, without being normalized by the sum of all weight coefficients (see https://github.com/twosigma/flint/blob/master/doc/ema.md#legacy).
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  price_sdf <- copy_to(
    sc,
    data.frame(
      time = ceiling(seq(12) / 2),
      price = seq(12) / 2,
      id = rep(c(3L, 7L), 6)
    )
  )
  ts <- fromSDF(price_sdf, is_sorted = TRUE, time_unit = "DAYS")
  ts_ewma <- summarize_ewma(
    ts,
    column = "price",
    smoothing_duration = "1d",
    key_columns = "id"
  )
} else {
  message("Unable to establish a Spark connection!")
```

```
}
```

`summarize_geometric_mean` *Geometric mean summarizer*

Description

Compute geometric mean of values from ‘column‘ within a moving time window or within each group of records with identical timestamps and store results in a new column named ‘<column>_geometricMean‘

Usage

```
summarize_geometric_mean(
  ts_rdd,
  column,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

`ts_rdd`	Timeseries RDD being summarized
`column`	Column to be summarized
`key_columns`	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.
`incremental`	If FALSE and ‘key_columns‘ is empty, then apply the summarizer to all records of ‘ts_rdd‘. If FALSE and ‘key_columns‘ is non-empty, then apply the summarizer to all records within each group determined by ‘key_columns‘. If TRUE and ‘key_columns‘ is empty, then for each record in ‘ts_rdd‘, the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and ‘key_columns‘ is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), u = seq(10, 1, -1)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_geometric_mean <- summarize_geometric_mean(ts, column = "u")
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_kurtosis *Kurtosis summarizer*

Description

Compute the excess kurtosis (fourth standardized moment minus 3) of ‘column‘ and store the result in a new column named ‘<column>_kurtosis‘

Usage

```
summarize_kurtosis(ts_rdd, column, key_columns = list(), incremental = FALSE)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

incremental — If FALSE and ‘key_columns‘ is empty, then apply the summarizer to all records of ‘ts_rdd‘. If FALSE and ‘key_columns‘ is non-empty, then apply the summarizer to all records within each group determined by ‘key_columns‘. If TRUE and ‘key_columns‘ is empty, then for each record in ‘ts_rdd‘, the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and ‘key_columns‘ is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_max(), summarize_min(), summarize_nth_central_mo summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  price_sdf <- copy_to(
    sc,
    data.frame(
      time = ceiling(seq(12) / 2),
      price = seq(12) / 2,
      id = rep(c(3L, 7L), 6)
    )
  )
  ts <- fromSDF(price_sdf, is_sorted = TRUE, time_unit = "DAYS")
  ts_kurtosis <- summarize_kurtosis(ts, column = "price")
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_max *Maximum value summarizer*

Description

Find maximum value among values from ‘column‘ within each time window or within each group of records with identical timestamps, and store results in a new column named ‘<column>_max‘

Usage

```
summarize_max(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
window	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_max <- summarize_max(ts, column = "v", window = in_past("3s"))
```

```
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_min *Minimum value summarizer*

Description

Find minimum value among values from ‘column‘ within each time window or within each group of records with identical timestamps, and store results in a new column named ‘<column>_min‘

Usage

```
summarize_min(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
window	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_min <- summarize_min(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_nth_central_moment

N-th central moment summarizer

Description

Compute n-th central moment of the column specified and store result in a new column named ‘<column>_<n>thCentralMoment‘

Usage

```
summarize_nth_central_moment(
  ts_rdd,
  column,
  n,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
n	The order of moment to calculate
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

incremental — If FALSE and ‘key_columns‘ is empty, then apply the summarizer to all records of ‘ts_rdd‘. If FALSE and ‘key_columns‘ is non-empty, then apply the summarizer to all records within each group determined by ‘key_columns‘. If TRUE and ‘key_columns‘ is empty, then for each record in ‘ts_rdd‘, the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and ‘key_columns‘ is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = rnorm(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_4th_central_moment <- summarize_nth_central_moment(ts, column = "v", n = 4L)
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_nth_moment *N-th moment summarizer*

Description

Compute n-th moment of the column specified and store result in a new column named ‘<column>_<n>thMoment‘

Usage

```
summarize_nth_moment(
  ts_rdd,
  column,
  n,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
n	The order of moment to calculate
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = rnorm(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_4th_moment <- summarize_nth_moment(ts, column = "v", n = 4L)
} else {
  message("Unable to establish a Spark connection!")
}
```

`summarize_product` *Product summarizer*

Description

Compute product of values from the given column within a moving time window new column named '<column>_product'

Usage

```
summarize_product(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

`ts_rdd`	Timeseries RDD being summarized
`column`	Column to be summarized
`window`	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
`key_columns`	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_product <- summarize_product(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_quantile *Quantile summarizer*

Description

Compute quantiles of ‘column‘ within each time window or within each group of records with identical time-stamps, and store results in new columns named ‘<column>_<quantile value>quantile‘

Usage

```
summarize_quantile(ts_rdd, column, p, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
p	List of quantile probabilities
window	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps

key_columns Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_quantile <- summarize_quantile(
    ts, column = "v", p = c(0.5, 0.75, 0.99), window = in_past("3s")
  )
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_skewness *Skewness summarizer*

Description

Compute skewness (third standardized moment) of 'column' and store the result in a new column named '<column>_skewness'

Usage

```
summarize_skewness(ts_rdd, column, key_columns = list(), incremental = FALSE)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  price_sdf <- copy_to(
    sc,
    data.frame(
      time = ceiling(seq(12) / 2),
      price = seq(12) / 2,
      id = rep(c(3L, 7L), 6)
```

```
    )
  )
  ts <- fromSDF(price_sdf, is_sorted = TRUE, time_unit = "DAYS")
  ts_skewness <- summarize_skewness(ts, column = "price")
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_stddev *Standard deviation summarizer*

Description

Compute unbiased (i.e., Bessel's correction is applied) sample standard deviation of values from 'column' within each time window or within each group of records with identical timestamps, and store results in a new column named '<column>_stddev'

Usage

```
summarize_stddev(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_stddev <- summarize_stddev(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_sum *Sum summarizer*

Description

Compute moving sums on the column specified and store results in a new column named '<column>_sum'

Usage

```
summarize_sum(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps

key_columns — Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_var(), summarize_weighted_avg(summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_sum <- summarize_sum(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_var *Variance summarizer*

Description

Compute variance of values from 'column' within each time window or within each group of records with identical timestamps, and store results in a new column named '<column>_variance', with Bessel's correction applied to the results

Usage

```
summarize_var(ts_rdd, column, window = NULL, key_columns = list())
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_weighted_avg(summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_var <- summarize_var(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

summarize_weighted_avg *Weighted average summarizer*

Description

Compute moving weighted average, weighted standard deviation, weighted t- stat, and observation count with the column and weight column specified and store results in new columns named '<column>_<weighted_column>_mean', '<column>_<weighted_column>_weightedStandardDeviation', '<column>_<weighted_column>_weightedTStat', and '<column>_<weighted_column>_observationCount',

Usage

```
summarize_weighted_avg(
  ts_rdd,
  column,
  weight_column,
  window = NULL,
  key_columns = list()
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
weight_column	Column specifying relative weight of each data point
window	Either an R expression specifying time windows to be summarized (e.g., 'in_past("1h")' to summarize data from looking behind 1 hour at each time point, 'in_future("5s")' to summarize data from looking forward 5 seconds at each time point), or 'NULL' to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_corr(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10), w = seq(1, 0.1, -0.1)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_weighted_avg <- summarize_weighted_avg(
    ts,
    column = "v", weight_column = "w", window = in_past("3s")
  )
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_weighted_corr

Pearson weighted correlation summarizer

Description

Compute Pearson weighted correlation between ‘xcolumn‘ and ‘ycolumn‘ weighted by ‘weight_column‘ and store result in a new columns named ‘<xcolumn>_<ycolumn>_<weight_column>_weightedCorrelation‘

Usage

```
summarize_weighted_corr(
  ts_rdd,
  xcolumn,
  ycolumn,
  weight_column,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
xcolumn	Column representing the first random variable
ycolumn	Column representing the second random variable
weight_column	Column specifying relative weight of each data point
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_covar(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
 sdf <- copy_to(sc, tibble::tibble(t = seq(10), x = rnorm(10), y = rnorm(10), w = 1.1^seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
 ts_weighted_corr <- summarize_weighted_corr(ts, xcolumn = "x", ycolumn = "y", weight_column = "w")
```

```
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_weighted_covar *Weighted covariance summarizer*

Description

Compute unbiased weighted covariance between values from ‘xcolumn‘ and ‘ycolumn‘ within each time window or within each group of records with identical timestamps, using values from ‘weight_column‘ as relative weights, and store results in a new column named ‘<xcolumn>_<ycolumn>_<weight_column>_v

Usage

```
summarize_weighted_covar(
  ts_rdd,
  xcolumn,
  ycolumn,
  weight_column,
  window = NULL,
  key_columns = list()
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
xcolumn	Column representing the first random variable
ycolumn	Column representing the second random variable
weight_column	Column specifying relative weight of each data point
window	Either an R expression specifying time windows to be summarized (e.g., ‘in_past("1h")‘ to summarize data from looking behind 1 hour at each time point, ‘in_future("5s")‘ to summarize data from looking forward 5 seconds at each time point), or ‘NULL‘ to compute aggregate statistics on records grouped by timestamps
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, ‘key_colums‘ is empty and all records are considered to be part of a single time series.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_z_score()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
 sdf <- copy_to(sc, tibble::tibble(t = seq(10), u = rnorm(10), v = rnorm(10), w = 1.1^seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_weighted_covar <- summarize_weighted_covar(
    ts,
    xcolumn = "u", ycolumn = "v", weight_column = "w", window = in_past("3s")
  )
} else {
  message("Unable to establish a Spark connection!")
}
```

---

summarize_z_score *Z-score summarizer*

Description

Compute z-score of value(s) in the column specified, with respect to the sample mean and standard deviation observed so far, with the option for out- of-sample calculation, and store result in a new column named '<column>_zScore'.

Usage

```
summarize_z_score(
  ts_rdd,
  column,
  include_current_observation = FALSE,
  key_columns = list(),
  incremental = FALSE
)
```

Arguments

ts_rdd	Timeseries RDD being summarized
column	Column to be summarized
include_current_observation	If true, then use unbiased sample standard deviation with current observation in z-score calculation, otherwise use unbiased sample standard deviation excluding current observation
key_columns	Optional list of columns that will form an equivalence relation associating each record with the time series it belongs to (i.e., any 2 records having equal values in those columns will be associated with the same time series, and any 2 records having differing values in those columns are considered to be from 2 separate time series and will therefore be summarized separately) By default, 'key_colums' is empty and all records are considered to be part of a single time series.
incremental	If FALSE and 'key_columns' is empty, then apply the summarizer to all records of 'ts_rdd'. If FALSE and 'key_columns' is non-empty, then apply the summarizer to all records within each group determined by 'key_columns'. If TRUE and 'key_columns' is empty, then for each record in 'ts_rdd', the summarizer is applied to that record and all records preceding it, and the summarized result is associated with the timestamp of that record. If TRUE and 'key_columns' is non-empty, then for each record within a group of records determined by 1 or more key columns, the summarizer is applied to that record and all records preceding it within its group, and the summarized result is associated with the timestamp of that record.

Value

A TimeSeriesRDD containing the summarized result

See Also

Other summarizers: ols_regression(), summarize_avg(), summarize_corr2(), summarize_corr(), summarize_count(), summarize_covar(), summarize_dot_product(), summarize_ema_half_life(), summarize_ewma(), summarize_geometric_mean(), summarize_kurtosis(), summarize_max(), summarize_min(), summarize_nth_central_moment(), summarize_nth_moment(), summarize_product(), summarize_quantile(), summarize_skewness(), summarize_stddev(), summarize_sum(), summarize_var(), summarize_weighted_avg(), summarize_weighted_corr(), summarize_weighted_covar()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = rnorm(10)))
```

```
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_z_score <- summarize_z_score(ts, column = "v", include_current_observation = TRUE)
} else {
  message("Unable to establish a Spark connection!")
}
```

---

to_sdf *Export data from TimeSeriesRDD to a Spark dataframe*

Description

Construct a Spark dataframe containing time series data from a TimeSeriesRDD

Usage

```
to_sdf(ts_rdd)

toSDF(ts_rdd)
```

Arguments

ts_rdd	A TimeSeriesRDD object

Value

A Spark dataframe containing time series data exported from ‘ts_rdd‘

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), ts_rdd_builder()

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), ts_rdd_builder()

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- from_sdf(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_avg <- summarize_avg(ts, column = "v", window = in_past("3s"))
  # now export the average values from `ts_avg` back to a Spark dataframe
  # named `sdf_avg`
```

```
  sdf_avg <- ts_avg %>% to_sdf()
} else {
  message("Unable to establish a Spark connection!")
}
```

`try_spark_connect` *Attempt to establish a Spark connection*

Description

Attempt to connect to Apache Spark and return a Spark connection object upon success

Usage

```
try_spark_connect(...)
```

Arguments

`...`	Parameters for sparklyr::spark_connect

Value

a Spark connection object if attempt was successful, or NULL otherwise

Examples

```
try_spark_connect(master = "local")
```

`ts_rdd_builder` *TimeSeriesRDD builder object*

Description

Builder object containing all required info (i.e., isSorted, timeUnit, and timeColumn) for importing a Spark data frame into a TimeSeriesRDD

Usage

```
ts_rdd_builder(
  sc,
  is_sorted = FALSE,
  time_unit = .sparklyr.flint.globals$kValidTimeUnits,
  time_column = .sparklyr.flint.globals$kDefaultTimeColumn
)
```

Arguments

sc	Spark connection
is_sorted	Whether the rows being imported are already sorted by time
time_unit	Time unit of the time column (must be one of the following values: "NANOSECONDS", "MICROSECONDS", "MILLISECONDS", "SECONDS", "MINUTES", "HOURS", "DAYS"
time_column	Name of the time column

Value

A reusable TimeSeriesRDD builder object

See Also

Other Spark dataframe utility functions: collect.ts_rdd(), from_rdd(), from_sdf(), spark_connection.ts_rdd(), spark_dataframe.ts_rdd(), spark_jobj.ts_rdd(), to_sdf()

---

window_exprs *Time window specifications*

Description

Functions for specifying commonly used types of time windows, which should only be used within the context of summarize_* functions (e.g., 'summarize_count(ts_rdd, in_past("3s"))'). When passing a time window specification to some summarize_* function, the Spark connection parameter ('sc') for the time window object will be injected and will be the same Spark connection the underlying timeseries RDD object is associated with, so, 'sc' never needs to be specified explicitly.

Create a sliding time window capuring data within the closed interval of [current time - duration, current time]

Create a sliding time window capuring data within the closed interval of [current time, current time + duration]

Usage

```
in_past(duration, sc)

in_future(duration, sc)
```

Arguments

duration	String representing length of the time window containing a number followed by a time unit (e.g., "10s" or "10sec"), where time unit must be one of the following: "d", "day", "h", "hour", "min", "minute", "s", "sec", "second", "ms", "milli", "millisecond", " μ s", "micro", "microsecond", "ns", "nano", "nanosecond"

sc	Spark connection (does not need to be specified within the context of ‘summarize_*‘ functions)

Value

A time window object useable by the Flint time series library

Examples

```
library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_count <- summarize_count(ts, column = "v", window = in_past("3s"))
} else {
  message("Unable to establish a Spark connection!")
}


library(sparklyr)
library(sparklyr.flint)

sc <- try_spark_connect(master = "local")

if (!is.null(sc)) {
  sdf <- copy_to(sc, tibble::tibble(t = seq(10), v = seq(10)))
  ts <- fromSDF(sdf, is_sorted = TRUE, time_unit = "SECONDS", time_column = "t")
  ts_count <- summarize_count(ts, column = "v", window = in_future("3s"))
} else {
  message("Unable to establish a Spark connection!")
}
```

Index