Package 'timeDF'

July 22, 2025

Type Package

Title Subset and Flag Data Frames with Times by the Use of Periods

Version 0.9.1

Date 2024-04-29

Maintainer Toshihiro Umehara <toshi@niceume.com>

Description Data frames with time information are subset and flagged with period information. Data frames with times are dealt as timeDF objects and periods are represented as periodDF objects.

Suggests RUnit

Imports Rcpp (>= 0.11.0), methods

LinkingTo Rcpp

License GPL (>= 3)

URL https://github.com/niceume/timeDF

BugReports https://github.com/niceume/timeDF

NeedsCompilation yes

Author Toshihiro Umehara [aut, cre]

Repository CRAN

Date/Publication 2024-04-29 02:40:03 UTC

Contents

`timeDF-package` *Subset and Flag Data Frames with Times by the Use of Periods*

Description

Data frames with time information are subset and flagged with period information. Data frames with times are dealt as timeDF objects and periods are represented as periodDF objects.

Details

Package timeDF provides functionality to deal with times with the use of periods. With period information, data frames with time information are subset and flagged.

Author(s)

Toshihiro Umehara [aut, cre] Maintainer: Toshihiro Umehara <toshi@niceume.com>

See Also

`timeDF-class` `periodDF-class` `as.timeDF` `as.periodDF` `vec_to_periodDF` `extract_with_periodDF` `flag_with_periodDF`

Examples

```
library(timeDF)
time_df = data.frame(
    time = c("2023-01-01 12:00:00",
             "2023-01-21 12:00:00",
             "2023-02-10 12:00:00",
             "2023-03-02 12:00:00",
             "2023-03-22 12:00:00",
```

```
                "2023-04-11 12:00:00"
    ),
    value = c(123, 144, 150, 100, 130, 145)
)

timeDF = as.timeDF(time_df)

period_df = data.frame(
    start = c("2023-01-01",
              "2023-03-01"),
    end = c("2023-01-31",
            "2023-03-31")
)
periodDF = as.periodDF(period_df, "date")

extract_with_periodDF(timeDF, periodDF, "both")
```

`adjust_periodDF` *Function to adjust starts and ends for periods in periodDF object*

Description

adjust_periodDF function

Usage

```
adjust_periodDF(periodDF, adjStart, adjEnd, units)
```

Arguments

`periodDF`	S3 periodDF class
`adjStart`	values to be added for starts.
`adjEnd`	values to be added for ends.
`units`	units for values of adjStart and adjEnd

Details

adjust_periodDF function adjust starts and ends for periods in periodDF objects.

Value

periodDF object

See Also

`periodDF-class` `timeDF-package`

Examples

```
period_time = data.frame(
     start = c("2023-12-01 03:00:00",
               "2023-12-01 20:00:00",
               "2023-12-02 05:00:00",
               "2023-12-03 21:00:00"),
     end = c("2023-12-01 04:00:00",
             "2023-12-01 21:00:00",
             "2023-12-02 06:00:00",
             "2023-12-03 22:00:00")
)
periodTime = as.periodDF(period_time, "time")
adjust_periodDF(periodTime, -1, 3, units="hours")

period_date = data.frame(
    start = c("2023-01-01",
              "2023-02-01",
              "2023-03-01",
              "2023-04-01"),
    end = c("2023-01-14",
            "2023-02-14",
            "2023-03-14",
            "2023-04-14"),
    label = c("One", "Two", "Three", "Four")
)
periodDate = as.periodDF(period_date, "date")
adjust_periodDF(periodDate, -1, 1, units="days")

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
periodTimeInDay = as.periodDF(period_time_in_a_day, "time_in_a_day", label_var = "label")
adjust_periodDF( periodTimeInDay, 1, 1, "hours")
```

`as.data.frame.timeDF` *Convert timeDF object to a plain dataframe*

Description

Convert timeDF object to a plain dataframe.

Usage

```
## S3 method for class 'timeDF'
as.data.frame(x,row.names=NULL,optional=FALSE,format =
"%Y-%m-%d %H:%M:%S",...)
```

Arguments

x	timeDF object
row.names	same as as.data.frame in base
optional	same as as.data.frame in base
format	character element that describes how times in timeDF are converted to chracters. If "as_is" is specified, time objects are not preserved without converting to characters.
...	Further arguments passed to or from other methods

Details

Convert timeDF object to a plain dataframe.

Value

dataframe

See Also

timeDF-class timeDF-package

Examples

```
as.data.frame(timeDF)
```

---

as.periodDF *Function to construct periodDF object from dataframe*

Description

as.periodDF function interpret dataframe and convert it into periodDF class object.

Usage

```
as.periodDF(df, period_type, format = "auto", start_var =
"start", end_var = "end", label_var = NULL)
```

Arguments

df	dataframe that holds columns for starts and ends of periods which are defined by start_var and end_var arguments. Column for labels is optional.
period_type	character element that defines what kind of periods are specified. "time", "date" or "time_in_a_day" is available.
format	character element that defines the formats of starts and end columns. If "auto" is specified, format that corresponds to period_type is automatically selected. If "as_is" is specified, columns for start and end are used as they are without conversion. In this case, column objects need to be compatible with objects that period_type requires. Time requires POSIXlt or POSIXct with UTC timezone, date requires Date, and time_in_a_day requires numeric values from 0 to 24 * 60 * 60.
start_var	character element that specifies the column name for starts.
end_var	character element that specifies the column name for ends.
label_var	character element that specifies the column name for labels.

Details

as.periodDF function constructs periodDF object from dataframe. Types of periodDF are described in periodDF-class.

Value

periodDF object

See Also

periodDF-class timeDF-package

Examples

```
period_time = data.frame(
     start = c("2023-12-01 01:00:00",
               "2023-12-01 02:00:00",
               "2023-12-01 03:00:00",
               "2023-12-02 04:00:00"),
     end = c("2023-12-01 02:00:00",
             "2023-12-01 03:00:00",
             "2023-12-01 04:00:00",
             "2023-12-02 05:00:00")
)
as.periodDF(period_time, "time")

period_date = data.frame(
    start = c("2023-01-01",
              "2023-02-01",
              "2023-03-01"),
    end = c("2023-01-14",
            "2023-02-14",
```

```
            "2023-03-14"),
    label = c("Jan", "Feb", "Mar")
)
as.periodDF(period_date, "date")

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
as.periodDF( period_time_in_a_day,
             "time_in_a_day",
              label_var = "label")
```

`as.timeDF` *Function to construct timeDF object from dataframe*

Description

as.timeDF function converts dataframe into timeDF class object

Usage

```
as.timeDF(df, time_var = "time", format = "%Y-%m-%d %H:%M:%S")
```

Arguments

`df`	dataframe that holds a column for time as character.
`time_var`	character element that specifies column name for time in the dataframe.
`format`	character element that specifies a format for times. If "as_is" is specified, column for start is used as it is without conversion. In this case, column needs to be POSIXlt with UTC timezone.

Details

as.timeDF function constructs timeDF object from dataframe.

Value

timeDF object

See Also

timeDF-class timeDF-package

Examples

```
time_df = data.frame(
    time = c("2023-12-01 01:00:00",
             "2023-12-01 02:00:00",
             "2023-12-01 03:00:00",
             "2023-12-01 04:00:00"),
    value = c(123,
              144,
              150,
              100)
)
as.timeDF(time_df)
```

---

condense_periodDF *Function to condense periods in periodDF object*

Description

condense_periodDF function

Usage

```
condense_periodDF(periodDF, open = TRUE, useData = "start")
```

Arguments

periodDF	S3 periodDF class
open	If this is set TRUE, periods are dealt as open intervals. If FALSE is set, periods are dealt as closed intervals.
useData	"start" or "end" is specified. This decides how columns other than start and end are combined. If "start" is specified, data of a period that contains start timing is used for the combined period. If "end" is specified, data of end period is used.

Details

condense_periodDF function condenses periods in a periodDF object. If periods are overlapped, they are condensed into one period. When periods share the same timing with their start and end, whether they are combined into one period or are dealt separately depends on an argument of open.

Value

periodDF object

See Also

periodDF-class timeDF-package

Examples

```
period_time = data.frame(
     start = c("2023-12-01 01:00:00",
               "2023-12-01 02:00:00",
               "2023-12-01 03:00:00",
               "2023-12-02 04:00:00"),
     end = c("2023-12-01 02:00:00",
             "2023-12-01 03:00:00",
             "2023-12-01 04:00:00",
             "2023-12-02 05:00:00")
)
periodTime = as.periodDF(period_time, "time")
condense_periodDF(periodTime, open = TRUE)
condense_periodDF(periodTime, open = FALSE)

period_date = data.frame(
    start = c("2023-01-01",
              "2023-01-14",
              "2023-02-14",
              "2023-03-14"),
    end = c("2023-01-31",
            "2023-02-14",
            "2023-03-14",
            "2023-04-14"),
    label = c("One", "Two", "Three", "Four")
)
periodDate = as.periodDF(period_date, "date")
condense_periodDF(periodDate, open=TRUE)
condense_periodDF(periodDate, open=FALSE)

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
periodTimeInDay = as.periodDF(period_time_in_a_day, "time_in_a_day", label_var = "label")
condense_periodDF( periodTimeInDay, open = TRUE)
condense_periodDF( periodTimeInDay, open = FALSE)
```

convert_periodDF *Function to convert type of periodDF object*

Description

convert_periodDF function

Usage

```
convert_periodDF(periodDF, period_type, base_date = NULL)
```

Arguments

periodDF	S3 periodDF class
period_type	period_type to be conveted to
base_date	only used when converting time_in_a_day into time type

Details

convert_periodDF function converts period types of periodDF object. Conversions from "date" to "time", "time" to "date", "time" to "time_in_a_day" and "time_in_a_day" to "time" are supported.

Value

periodDF object

See Also

periodDF-class timeDF-package

Examples

```
period_date = data.frame(
    start = c("2023-01-01",
              "2023-02-01",
              "2023-03-01",
              "2023-04-01"),
    end = c("2023-01-14",
            "2023-02-14",
            "2023-03-14",
            "2023-04-14"),
    label = c("One", "Two", "Three", "Four")
)
periodDate = as.periodDF(period_date, "date")
convert_periodDF(periodDate, "time")

period_time = data.frame(
     start = c("2023-12-01 03:00:00",
               "2023-12-01 20:00:00",
```

```
                "2023-12-02 05:00:00",
                "2023-12-03 21:00:00"),
     end = c("2023-12-01 04:00:00",
             "2023-12-01 21:00:00",
             "2023-12-02 06:00:00",
             "2023-12-03 22:00:00")
)
periodTime = as.periodDF(period_time, "time")
convert_periodDF(periodTime, "date")

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
periodTimeInDay = as.periodDF(period_time_in_a_day, "time_in_a_day", label_var = "label")
convert_periodDF( periodTimeInDay, "time", base_date = "2023-12-01")
```

`extract_with_periodDF` *Extract time records from timeDF object within periods of periodDF object*

Description

Extract time records from timeDF object that are included within periods of periodDF object. periodDF object has one of some timescales, and how this function extracts time records depends on the timescale. Also, when the time is on either end of a time period, whether the time record is extracted or not depends on the include argument. "both" means including both sides, "right" means including only the right side, "left" means including only the left side, and "none" does not include any sides.

Usage

```
extract_with_periodDF(timeDF, periodDF, include, modStart = 0, modEnd = 0,
units = NULL, outputAsBool = FALSE)
```

Arguments

`timeDF`	timeDF object
`periodDF`	periodDF object
`include`	character element that specifies whether each end of periods is included or not
`modStart`	values to be added for starts of periods.

modEnd	values to be added for ends of periods.
units	units for values of modStart and modEnd
outputAsBool	boolean value; if this is TRUE, the return value is a boolean vector or boolean vectors that represent(s) records to be extracted

Value

If periodDF does not have labels, timeDF object or a boolean object is returned. If periodDF has labels, a list of timeDF objects with keys of label names or a list of boolean vectors with label name keys is retruned.

See Also

timeDF-class periodDF-class timeDF-package

Examples

```
time_df = data.frame(
    time = c("2023-01-01 12:00:00",
             "2023-01-21 12:00:00",
             "2023-02-10 12:00:00",
             "2023-03-02 12:00:00",
             "2023-03-22 12:00:00",
             "2023-04-11 12:00:00"
    ),
    value = c(123, 144, 150, 100, 130, 145)
)
timeDF = as.timeDF(time_df)

period_df = data.frame(
    start = c("2023-01-01",
              "2023-02-01",
              "2023-03-01"),
    end = c("2023-01-31",
            "2023-02-28",
            "2023-03-31"),
    label = c("Jan", "Feb", "Mar")
)
periodDF = as.periodDF(period_df, "date", label_var = "label")

extract_with_periodDF(timeDF, periodDF, "both")
```

---

flag_with_periodDF *Flag time records from timeDF object within periods of periodDF object*

Description

Flag time records of timeDF object that are included within periods of periodDF object. Which time records are flagged follows the same rule as extract_with_periodDF function.

Usage

```
flag_with_periodDF(timeDF, periodDF, flag_var, include, modStart = 0,
modEnd = 0, units = NULL)
```

Arguments

timeDF	timeDF object
periodDF	periodDF object
flag_var	character element that specifies the column name to which flaggs are added
include	character element that specifies whether each end of periods is included or not
modStart	values to be added for starts of periods.
modEnd	values to be added for ends of periods.
units	units for values of modStart and modEnd

Value

timeDF object flagged with labels

See Also

timeDF-class periodDF-class timeDF-package

Examples

```
time_df = data.frame(
    time = c("2023-01-01 12:00:00",
             "2023-01-21 12:00:00",
             "2023-02-10 12:00:00",
             "2023-03-02 12:00:00",
             "2023-03-22 12:00:00",
             "2023-04-11 12:00:00"
    ),
    value = c(123, 144, 150, 100, 130, 145)
)
timeDF = as.timeDF(time_df)

period_df = data.frame(
    start = c("2023-01-01",
              "2023-02-01",
              "2023-03-01"),
    end = c("2023-01-31",
            "2023-02-28",
            "2023-03-31"),
    label = c("Jan", "Feb", "Mar")
)
periodDF = as.periodDF(period_df, "date", label_var = "label")

flag_with_periodDF(timeDF, periodDF, "month_label", "both")
```

listTimeDF_to_timeDF *converts a list of timeDF objects into a timeDF object*

Description

listTimeDF_to_timeDF function combines timeDF objects in the original list. Each timeDF name in the original list is assigned to a column specified by name_var argument.

Usage

```
listTimeDF_to_timeDF(listTimeDF, name_var = "name")
```

Arguments

listTimeDF	a list of timeDF objects
name_var	column name holding names from the original list

Value

timeDF object

See Also

timeDF-class

Examples

```
listTimeDF = extract_with_periodDF(
                time_df,
                period_df,
                include="both")
listTimeDF_to_timeDF(listTimeDF)
```

periodDF-class *periodDF S3 class*

Description

periodDF object stores definitions of periods.

Details

periodDF object stores definitions of periods and the periods can be defined in one of some timescales, "time", "date" or "time_in_a_day". If "time" is specified, each period means period between two timepoints. If "date" is specified, each period represents period between two dates. If "time_in_a_day" is used, each period means period between two timepoints in a day. periodDF object is used with functions in timeDF-package, and those functions behave diffrently based on the timescale.

See Also

as.periodDF vec_to_periodDF timeDF-package

---

period_type *Function to obtain period type of periodDF object*

Description

period_type function returns the period type of periodDF object.

Usage

```
period_type(periodDF)
```

Arguments

periodDF	periodDF object

Details

period_type function returns the period type of periodDF object.

Value

string

See Also

periodDF-class

Examples

```
period_time = data.frame(
     start = c("2023-12-01 01:00:00",
               "2023-12-01 02:00:00",
               "2023-12-01 03:00:00",
               "2023-12-02 04:00:00"),
     end = c("2023-12-01 02:00:00",
             "2023-12-01 03:00:00",
             "2023-12-01 04:00:00",
             "2023-12-02 05:00:00")
)
periodTime = as.periodDF(period_time, "time")
period_type(periodTime)

period_date = data.frame(
    start = c("2023-01-01",
              "2023-01-14",
              "2023-02-14",
```

```
                "2023-03-14"),
    end = c("2023-01-31",
            "2023-02-14",
            "2023-03-14",
            "2023-04-14"),
    label = c("One", "Two", "Three", "Four")
)
periodDate = as.periodDF(period_date, "date",
label_var = "label")
period_type(periodDate)

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
periodTimeInDay = as.periodDF(period_time_in_a_day,
"time_in_a_day", label_var = "label")
period_type(periodTimeInDay)
```

`select_timeDF` *Function to select columns in timeDF object*

Description

select_timeDF function

Usage

```
select_timeDF(timeDF, colnames)
```

Arguments

`timeDF`	timeDF object
`colnames`	column names to be selected, chracter vector

Details

select_timeDF function returns a new timeDF object with columns specified and the column holding time information.

Value

timeDF object

See Also

timeDF-class

Examples

```
time_df = data.frame(
    time = c("2023-12-01 01:00:00",
             "2023-12-01 05:00:00",
             "2023-12-01 09:00:00",
             "2023-12-01 13:00:00",
             "2023-12-01 17:00:00",
             "2023-12-01 21:00:00"),
    value = c(123,
              144,
              150,
              100,
              200,
              180),
    phase = c("A",
              "A",
              "B",
              "B",
              "C",
              "C")
)
timeDF = as.timeDF(time_df)
select_timeDF(timeDF, c("phase"))
```

---

sort_periodDF *Function to sort periods in periodDF object*

Description

sort_periodDF function

Usage

```
sort_periodDF(periodDF, by="start")
```

Arguments

periodDF	S3 periodDF class
by	"start" or "end" is set. Periods are sorted by start_var or end_var in periodDF.

Details

sort_periodDF function sort periods in a periodDF object.

Value

periodDF object

See Also

periodDF-class timeDF-package

Examples

```
period_time = data.frame(
     start = c("2023-12-01 01:00:00",
               "2023-12-01 02:00:00",
               "2023-12-01 03:00:00",
               "2023-12-02 04:00:00"),
     end = c("2023-12-01 02:00:00",
             "2023-12-01 03:00:00",
             "2023-12-01 04:00:00",
             "2023-12-02 05:00:00")
)
periodTime = as.periodDF(period_time, "time")
sort_periodDF(periodTime)

period_date = data.frame(
    start = c("2023-01-01",
              "2023-01-14",
              "2023-02-14",
              "2023-03-14"),
    end = c("2023-01-31",
            "2023-02-14",
            "2023-03-14",
            "2023-04-14"),
    label = c("One", "Two", "Three", "Four")
)
periodDate = as.periodDF(period_date, "date")
sort_periodDF(periodDate)

period_time_in_a_day = data.frame(
    start = c("04:00",
              "11:00",
              "17:00"),
    end = c("11:00",
            "17:00",
            "24:00"),
    label = c("morning",
              "afternoon",
              "evening")
)
periodTimeInDay = as.periodDF(period_time_in_a_day, "time_in_a_day", label_var = "label")
sort_periodDF(periodTimeInDay)
```

`sort_timeDF` *Function to sort records in timeDF object*

Description

sort_timeDF function

Usage

```
sort_timeDF(timeDF, decreasing=FALSE)
```

Arguments

`timeDF`	timeDF object
`decreasing`	boolean value to specify whether the sorting is conducted in decreasing order or not.

Details

sort_timeDF function sorts records in timeDF object.

Value

timeDF object

See Also

`timeDF-class`

Examples

```
sort_timeDF(timeDF)
```

`split_timeDF_by_intervals` *Function to split timeDF into a list by regular intervals*

Description

split_timeDF_by_intervals splits timeDF into a list of timeDF objects by regular intervals. Intervals can be spcified by numeric value and its unit. For example, if the interval is specified as two days, timeDF object is split by two day interval. The start time for this interval is decided by the minimum time and the interval unit in the original timeDF. If there are no records present for some intervals, their corresponding results are timeDF objects with zero rows.

Usage

```
split_timeDF_by_intervals(timeDF,byN,byUnits,
                          modStart=0,modEnd=0,modUnits = "auto")
```

Arguments

timeDF	timeDF object
byN	interval value, numeric
byUnits	interval unit, "days", "hours" or "mins"
modStart	values to be added for starts of intervals
modEnd	values to be added for ends of intervals
modUnits	units for values of modStart and modEnd

Value

list of timeDF objects

See Also

timeDF-class

Examples

```
time_df = data.frame(
    time = c("2023-01-01 08:00:00",
             "2023-01-01 12:00:00",
             "2023-01-01 16:00:00",
             "2023-01-02 08:00:00",
             "2023-02-02 10:00:00",
             "2023-03-03 11:00:00",
             "2023-03-03 16:00:00",
             "2023-03-05 12:00:00"
    ),
    value = c(123, 144, 150, 100,
              130, 145, 180, 100)
)
timeDF = as.timeDF(time_df)

split_timeDF_by_intervals(timeDF, 1, "days")
```

summary.periodDF *Summarize periodDF S3 object*

Description

summary function for periodDF S3 object.

Usage

```
## S3 method for class 'periodDF'
summary(object,...)
```

Arguments

object	S3 periodDF class
...	Further arguments passed to or from other methods.

Details

summary function for periodDF S3 object. This enables users to obtain the summary of periods.

Value

List that have properties of periodDF object.

See Also

periodDF-class

Examples

```
summary(periodDF)
```

summary.timeDF *Summarize timeDF S3 object*

Description

summary function for timeDF S3 object.

Usage

```
## S3 method for class 'timeDF'
summary(object,...)
```

Arguments

object	S3 timeDF class
...	Further arguments passed to or from other methods.

Details

summary function for timeDF S3 object. This enables users to obtain the summary of periods.

Value

List that have properties of timeDF object.

See Also

timeDF-class

Examples

```
summary(timeDF)
```

---

timeDF-class *timeDF S3 class*

Description

timeDF object stores definitions of periods.

Details

timeDF object stores records with time information. The column to hold time information can be specified as "time_var" attribute.

See Also

as.timeDF timeDF-package

time_vec *time vector and column name for times of timeDF*

Description

Functions to obtain time vector and column name for times of timeDF

Usage

```
time_vec(timeDF)

time_var(timeDF)
```

Arguments

timeDF	timeDF object

Details

time_vec function returns times in timeDF object.

time_var function returns the column name for times in timeDF object, i.e. returns the value of time_var attribute.

Value

For time_vec, a vector of time objects

For time_var, a string of the column name for tiems

See Also

timeDF-class

Examples

```
time_vec(timeDF)
time_var(timeDF)
```

`validate_listTimeDF` *Checks whether the object is a list of timeDF objects*

Description

Checks whether the object is list of timeDF objects

Usage

```
validate_listTimeDF(listTimeDF, noerror=FALSE)
```

Arguments

`listTimeDF`	is expected to be a list of timeDF objects
`noerror`	boolean value determins whether the function raises an error or returns FALSE when the object is not a valid timeDF object.

Details

validate_listTimeDF function checks whether the object is a list of timeDF objects. If noerror is FALSE and the object is not a list of timeDF objects, this function raises an error. If noerror is TRUE, this function returns FALSE when the object is not a list of timeDF objects.

Value

boolean

See Also

`timeDF-class`

Examples

```
time_df = as.timeDF(
    data.frame(
        time = c("2024-01-01 01:00:00",
                 "2024-02-02 02:00:00",
                 "2024-03-03 03:00:00",
                 "2024-04-04 04:00:00",
                 "2024-05-05 05:00:00"),
        value = c(123,
                  144,
                  150,
                  100,
                  180)
    ))
period_df = as.periodDF(
    data.frame(
        start = c(
```

```
            "2024-01-01",
            "2024-02-01",
            "2024-03-01",
            "2024-04-01",
            "2024-05-01"
        ),
        end = c(
            "2024-01-31",
            "2024-02-29",
            "2024-03-31",
            "2024-04-30",
            "2024-05-31"
        ),
        label = c(
            "Jan",
            "Feb",
            "Mar",
            "Apr",
            "May"
        )
    ),
    period_type = "date",
    label_var = "label"
)
listTimeDF = extract_with_periodDF(time_df,
                                   period_df,
                                   include="both")
validate_listTimeDF(listTimeDF, noerror=TRUE)
```

`validate_sorted_timeDF` *Checks whether the object is a valid sorted timeDF object*

Description

Checks whether the object is a timeDF object sorted by its time information.

Usage

```
validate_sorted_timeDF(timeDF, noerror=FALSE)
```

Arguments

`timeDF`	timeDF object
`noerror`	boolean value determins whether the function raises an error or returns FALSE when the object is not a valid sorted timeDF object.

Details

validate_sorted_timeDF function checks whether the object is a timeDF object sorted by its time information. If noerror is FALSE and the object is not a valid sorted timeDF object, this function raises an error. If noerror is TRUE, this function returns FALSE when the object is not a valid sorted timeDF object.

Value

boolean

See Also

timeDF-class

Examples

```
validate_sorted_timeDF(timeDF)
```

---

validate_timeDF *Checks whether the object is a valid timeDF object*

Description

Checks whether the object is a valid timeDF object.

Usage

```
validate_timeDF(timeDF, noerror=FALSE)
```

Arguments

timeDF	timeDF object
noerror	boolean value determins whether the function raises an error or returns FALSE when the object is not a valid timeDF object.

Details

validate_timeDF function checks whether the object is a valid timeDF object. If noerror is FALSE and the object is not a valid timeDF object, this function raises an error. If noerror is TRUE, this function returns FALSE when the object is not a valid timeDF object.

Value

boolean

See Also

timeDF-class

Examples

```
validate_timeDF(timeDF)
```

`vec_to_periodDF` *Function to construct periodDF object from vector*

Description

vec_to_periodDF function takes a vector of timepoints or dates to start and each duration, and constructs periodDF class object.

Usage

```
vec_to_periodDF(vec, period_type, duration, units, format = "auto",
labels = NULL, pre_margin = 0)
```

Arguments

`vec`	vector that represents starts of periods. If pre_margin argument is specified, each period extends forward from the starts.
`period_type`	character element that defines what kind of periods are specified. "time", "date" or "time_in_a_day" is available.
`duration`	numeric values represent duration of each period.
`units`	character element represents unit of duration.
`labels`	labels that are used for each period.
`format`	character element that defines the formats of vec. If "auto" is specified, format that corresponds to period_type is automatically selected. If "as_is" is specified, the vector is used for starts of periods as it is without conversion. In this case, the vector needs to be compatible with objects that period_type requires. Time requires POSIXlt or POSIXct with UTC timezone, date requires Date, and time_in_a_day requires numeric values from 0 to 24 * 60 * 60.
`pre_margin`	numeric values; if values are set, each period extends forward from the starts specified in vec argument.

Details

vec_to_periodDF function takes a vector of timepoints or dates to start and each duration, and constructs periodDF class object. Types of periodDF are described in `periodDF-class`.

Value

periodDF object

See Also

`periodDF-class` `timeDF-package`

Examples

```
start_time = c("2023-12-01 01:00:00",
               "2023-12-02 02:00:00",
               "2023-12-03 03:00:00",
               "2023-12-04 04:00:00")
vec_to_periodDF(start_time, "time", 1, "hours")

start_date = c("2023-01-01",
               "2023-02-01",
               "2023-03-01")
vec_to_periodDF(start_date, "date", 14, "days",
                      labels = c("Jan", "Feb", "Mar"))

start_time_in_a_day = c("06:00",
                        "11:00",
                        "18:00")
vec_to_periodDF(start_time_in_a_day, "time_in_a_day",
                      4, "hours",
                      labels = c("morning", "afternoon", "evening"))
```

Index