Package ‘hydrotoolbox’

October 24, 2022

Type Package
Title Hydrological Tools for Handling Hydro-Meteorological Data Records
Version 1.1.0
Date 2022-10-24
Author Ezequiel Toum <etoum@mendoza-conicet.gob.ar>
Maintainer Ezequiel Toum <etoum@mendoza-conicet.gob.ar>
Description Read, plot, manipulate and process hydro-meteorological data records (with special features for Argentina and Chile data-sets).
Depends R (>= 2.10)
License GPL (>= 3)
Imports ggplot2, plotly, lubridate, utils, methods, readxl, reshape2, magrittr, tibble, zoo, Rcpp
URL https://gitlab.com/ezetoum27/hydrotoolbox
BugReports https://gitlab.com/ezetoum27/hydrotoolbox/-/issues
Encoding UTF-8
RoxygenNote 7.2.1
LinkingTo Rcpp
Suggests testthat, knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-10-24 17:05:08 UTC

R topics documented:

agg_table .............................................. 2
cum_sum .................................................. 4
fill_table ............................................... 5
agg_table

Aggregates a data frame to a larger time period

Description

Aggregates a data frame to a larger time period

Usage

agg_table(
  x,
  col_name,
  fun,
  period,
)
out_name = NULL,
allow_na = 0,
start_month = 1,
end_month = 12
)

Arguments

x  data frame or tibble with class Date or POSIX* in the first column.
col_name  string with column(s) name(s) to aggregate.
fun  string with supported aggregation function name (one per 'col_name'): 'mean', 'min', 'max', 'sum', 'last' or 'first'.
period  string with the aggregation time-step: 'hourly', 'daily', 'monthly', 'annually' or 'climatic'. **NOTE:** the 'climatic' option returns the all series annual statistics ('fun').
out_name  optional. String with the output column(s) name(s). Default values coerce the original name plus the 'fun' argument (e.g.: tair_max).
allow_na  optional. Numeric value with the maximum allowed number of NA_real_values. By default the function will not tolerate any NA_real_ (and will return NA_real_ instead).
start_month  optional. Numeric value defining the first month of the annual period (it just make sense if 'period' is either 'annually' or 'climatic'). Default sets to 1 (January). **NOTE:** keep in mind that if you choose 'climatic' as period you have to round off a complete year (e.g.: ..., start_month = 6, end_month = 5, ...)
end_month  optional. Numeric value defining the last month of the annual period (it just make sense if 'period' is either 'annually' or 'climatic'). Default sets to 12 (December). **NOTE:** keep in mind that if you choose 'climatic' as period you have to round off a complete year (e.g.: ..., start_month = 6, end_month = 5, ...)

Value

A data frame with the Date and the aggregated variable(s).

Examples

# set path to file
path <- system.file('extdata', 'snih_qd_guido.xlsx',
  package = 'hydrotoolbox')

# read and load daily streamflow with default column name
guido_qd <- read_snih(path = path, by = 'day', out_name = 'q(m3/s)')

# aggregate daily to monthly discharge
guido_q_month <- agg_table(x = guido_qd, col_name = 'q(m3/s)',
  fun = 'mean', period = 'monthly',

# suppose that we are interested on getting the annual maximum # daily mean discharge for every hydrological year (since this # station is located at the Mendoza River Basin ~32.9º S, we will # consider that annual period starts on July)
guido_q_annual <- agg_table(x = guido_qd, col_name = 'q(m3/s)',
    fun = 'max', period = 'annually',
    out_name = 'qmax(m3/s)',
    start_month = 7, end_month = 6)

# now we want the mean, maximum and minimum monthly discharges
guido_q_stats <- agg_table(x = guido_qd, col_name = rep('q(m3/s)', 3),
    fun = c('mean', 'max', 'min'),
    period = 'monthly')

---

**cum_sum**

**Cumulative sum**

**Description**

The function supports `NA_real_` values. It could be very useful when dealing with incomplete precipitation series.

**Usage**

```r
cum_sum(x, col_name, out_name = NULL)
```

**Arguments**

- `x` data frame or tibble with class `Date` or `POSIXt` in the first column.
- `col_name` string with column(s) name(s) where to apply the function.
- `out_name` optional. String with new column(s) name(s). If you set it as `NULL`, the function will overwrite the original table.

**Value**

The same table but with the new series.

**Examples**

```r
## Not run:
# set path to file
path <- system.file('extdata', 'ianigla_cuevas.csv',
    package = 'hydrotoolbox')

# read the file and add the new column with cumulative precipitation
cuevas <-
  read_ianigla(path = path) %>%
  cum_sum(col_name = 'Precip_Total', out_name = 'p_cum')

# plot it
plot(x = cuevas[, 'date', drop = TRUE],
     y = cuevas[, 'p_cum', drop = TRUE],
     col = 'red', type = 'l',
     xlab = 'Date', ylab = 'Pcum(mm)')

## End(Not run)

---

**fill_table**

*Find non-reported dates and fill them with NA_*

**Description**

Automatically finds non recorded date periods and fills them with `NA_real_` values.

**Usage**

```r
fill_table(x, col_name = "all", by = NULL)
```

**Arguments**

- **x**: data frame (or tibble) with class `Date` or `POSIXt` in the first column.
- **col_name**: string with column(s) name(s) to fill.
- **by**: string with a valid time step (e.g.: "month", "day", "6 hour", "3 hour", "1 hour", "15 min").

**Value**

A data frame (or tibble) with the date and the filled numeric variable(s).

**Examples**

```r
# let's use a synthetic example to illustrate the use of the function
dates <- seq.Date(from = as.Date('1980-01-01'),
                  to = as.Date('2020-01-01'), by = 'day')
var <- runif(n = length(dates), min = 0, max = 100)
met_var <- data.frame(date = dates, random = var)[-c(50:100, 251, 38), ]
met_var_fill <- fill_table(x = met_var, by = 'day')
```
hm_agg

Aggregates the table inside a slot to a larger time period

Description

This method allows you to get your data temporally aggregated.

Usage

hm_agg(
  obj,
  slot_name,
  col_name,
  fun,
  period,
  out_name = NULL,
  allow_na = 0,
  start_month = 1,
  end_month = 12,
  relocate = NULL
)

## S4 method for signature 'hydromet_station'
hm_agg(
  obj,
  slot_name,
  col_name,
  fun,
  period,
  out_name = NULL,
  allow_na = 0,
  start_month = 1,
  end_month = 12,
  relocate = NULL
)

## S4 method for signature 'hydromet_compact'
hm_agg(
  obj,
  slot_name,
  col_name,
  fun,
  period,
  out_name = NULL,
  allow_na = 0,
  start_month = 1,
  end_month = 12
)
Arguments

- **obj**: a valid hydromet\_XXX class object.
- **slot\_name**: string with the name of the slot to aggregate.
- **col\_name**: string with column(s) name(s) to aggregate.
- **fun**: string with supported aggregation function name (one per `col\_name`): 'mean', 'min', 'max', 'sum', 'last' or 'first'.
- **period**: string with the aggregation time-step: 'hourly', 'daily', 'monthly', 'annually' or 'climatic'. **NOTE 1**: the 'climatic' option returns the all series annual statistics ('fun'). **NOTE 2**: when using 'annually' as `period`, the method will return the starting dates in the first slot column.
- **out\_name**: string with the output column(s) name(s). Default values coerce the original name plus the 'fun' argument (e.g.: `tair_max`).
- **allow\_na**: optional. Numeric value with the maximum allowed number of NA\_real\_ values. By default the function will not tolerate any NA\_real\_ (and will return NA\_real\_ instead).
- **start\_month**: optional. Numeric value defining the first month of the annual period (it just make sense if `period` is either 'annually' or 'climatic'). Default sets to 1 (January). **NOTE**: keep in mind that if you choose 'climatic' as period you have to round off a complete year (e.g.: ..., start\_month = 6, end\_month = 5, ...)
- **end\_month**: optional. Numeric value defining the last month of the annual period (it just make sense if `period` is either 'annually' or 'climatic'). Default sets to 12 (December). **NOTE**: keep in mind that if you choose 'climatic' as period you have to round off a complete year (e.g.: ..., start\_month = 6, end\_month = 5, ...)
- **relocate**: optional. String with the name of the slot where to allocate the aggregated table. It only make sense for hydromet\_station class. When using it you must keep in mind that all aggregated series are allocated in a single slot.

Value

A data frame with the Date and the aggregated variable(s) inside the specified slot.

Functions

- `hm\_agg(hydromet\_station)`: temporal aggregation method for station class
- `hm\_agg(hydromet\_compact)`: temporal aggregation method for compact class

Examples

```r
## Not run:
# cuevas station
path <- system.file('extdata', package = 'hydrotoolbox')
```
# use the build method
hm_cuevas <-
  hm_create() %>%
  hm_build(bureau = 'ianigla', path = path,
    file_name = 'ianigla_cuevas.csv',
    slot_name = c('tair', 'rh', 'patm',
      'precip', 'wspd', 'wdir',
      'kin', 'hsnow', 'tsoil'),
    by = 'hour',
    out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
      'p(mm)', 'wspd(km/hr)', 'wdir(°)',
      'kin(kW/m²)', 'hsnow(cm)', 'tsoil(°C)'))

# aggregate air temperature data to mean value
hm_agg(obj = hm_cuevas, slot_name = 'tair', col_name = 'tair(°C)',
  fun = 'mean', period = 'daily', out_name = 't_mean') %>%
hm_show(slot_name = 'tair')

# the previous command overwrites the original slot, so now we are going
# to relocate the agg values
hm_agg(obj = hm_cuevas, slot_name = 'tair',
  col_name = 'tair(°C)',
  fun = 'mean',
  period = 'daily',
  relocate = 'tmean',
  out_name = 'tmean(°C)',
  ) %>%
hm_show(slot_name = 'tmean')

## End(Not run)

---

**hm_build**

*Load native data files automatically*

**Description**

The method allows you to automatically load your native data inside the hydromet_station slots.

**Usage**

```r
hm_build(
  obj,
  bureau,
  path,
  file_name,
  slot_name,
  by,
```
`hm_build`

```r
out_name = NULL,
sheet = NULL
```

## S4 method for signature 'hydromet_station'

```r
hm_build(
  obj,
  bureau,
  path,
  file_name,
  slot_name,
  by,
  out_name = NULL,
  sheet = NULL
)
```

### Arguments

- **obj**: a valid `hydromet_station` class object.
- **bureau**: string value containing one of the available options: 'aic', 'cr2', 'dgi', 'ianigla', 'mnemos' or 'snih'.
- **path**: string vector with the path(s) to the `file_name` argument. If you set a single string it will be recycled for all the files.
- **file_name**: string vector with the native file(s) name(s).
- **slot_name**: string vector with the slot(s) where to set the file(s) or sheet(s).
- **by**: string vector with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). If you set it as 'none', the function will ignore automatic gap filling. If you set a single string it will be recycled for all the files.
- **out_name**: optional. String vector with user defined variable(s) column(s) name(s).
- **sheet**: optional. Sheet to read. Either a string vector (the name of a sheet), or an integer vector (the position of the sheet). If neither argument specifies the sheet, defaults to the first sheet. This argument just make sense for:
  - 'aic': you must provide a single name or integer indicating the met-station to read.
  - 'dgi': just keep it as `NULL`.
  - 'mnemos': just keep it as `NULL`.

### Value

A `hydromet_station` object with the required data loaded inside.

### Functions

- `hm_build(hydromet_station)`: build method for hydromet station object
## Examples

```r
## Not run:
# path to all example files
path <- system.file('extdata', package = 'hydrotoolbox')

# ianigla file
hm_create() %>%
  hm_build(bureau = 'ianigla', path = path,
           file_name = 'ianigla_cuevas.csv',
           slot_name = c('tair', 'rh', 'patm',
                         'precip', 'wspd', 'wdir',
                         'kin', 'hsnow', 'tsoil'),
           by = 'hour',
           out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
                        'p(mm)', 'wspd(km/hr)', 'wdir(*)',
                        'kin(KW/m2)', 'hsnow(cm)', 'tsoil(°C)'))
  ) %>%
  hm_show()

# cr2 file
hm_create() %>%
  hm_build(bureau = 'cr2', path = path,
           file_name = 'cr2_tmax_yeso_embalse.csv',
           slot_name = c('tmax'),
           by = 'day',
           out_name = c('tair(°C)'))
  ) %>%
  hm_show()

# dgi file
hm_create() %>%
  hm_build(bureau = 'dgi', path = path,
           file_name = 'dgi_toscas.xlsx',
           slot_name = c('swe', 'tmax',
                         'tmin', 'tmean', 'rh', 'patm'),
           by = 'day')
  ) %>%
  hm_show()

# snih file
hm_create() %>%
  hm_build(bureau = 'snih', path = path,
           file_name = c('snih_hq_guido.xlsx',
                         'snih_qd_guido.xlsx'),
           slot_name = c('hq', 'qd'),
           by = c('none', 'day'))
  ) %>%
  hm_show()

# aic => you have to request for this files to AIC.

# mnemos => the data are the same of snih but generated
#           with MNEMOSIII software.
```
hm_build_generic

## End(Not run)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hm_build_generic</td>
<td>Load native data files automatically</td>
</tr>
</tbody>
</table>

### Description

The method allows you to automatically load your native data inside the hydromet_station slots.

### Usage

```r
hm_build_generic(
  obj,  path,  file_name,  slot_name,
  by = "none",  out_name = NULL,  sheet = NULL,
  FUN,  ...
)
```

### Arguments

- **obj**: a valid hydromet_station class object.
- **path**: string vector with the path(s) to the file_name argument. If you set a single string it will be recycled for all the files.
- **file_name**: string vector with the native file(s) name(s).
- **slot_name**: string vector with the slot(s) where to set the file(s) or sheet(s).
**by**

string vector with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). If you set it as "none", the function will ignore automatic gap filling. If you set a single string, it will be recycled for all the files.

**out_name**

optional. A list containing string vectors with user defined variable(s) column(s) name(s). The list length should be equal to the slot_name length.

**sheet**

Sheet to read (only excel files). Either a string vector (the name of a sheet) or an integer vector (the position of the sheet). This argument just make sense for excel files.

**FUN**

function name for reading the data (e.g.: `read_csv()`). The method will always use the path + file as first argument(s) to FUN.

... FUN arguments to pass.

**Value**

A hydromet_station object with the required data loaded inside.

**Functions**

* `hm_build_generic(hydromet_station)`: build method for hydromet station object

**Examples**

```r
## Not run:
# you can download the data from:
# https://gitlab.com/ezetoum27/hydrotoolbox/-/tree/master/my_data

# set the data path
my_path <- "/home/my_folder/my_data"

# Rectangular data
# txt, csv, csv2 and others.
# See readr package.

library(readr)

#* Case 1: single file - many numeric variables

hm_create() %>%
hm_build_generic(path = my_path,
  file_name = "ianigla_cuevas.csv",
  slot_name = c("tair", "rh", "patm",
    "precip", "wspd", "wdir",
    "kin", "hsnow", "tsoil"),
  by = c("hour"),
  FUN = read_csv,
  col_select = !Est & !YJday & !hh.mm.ss & !bat.Volts
) %>%

hm_show()
```
hm_build_generic

** Case 2: multiple files (one per observation)**

```r
hm_create() %>%
  hm_build_generic(path = my_path,
                   file_name = c("h_relativa_cuevas.csv",
                                  "p_atm_cuevas.csv",
                                  "precip_total_cuevas.csv",
                                  "temp_aire_cuevas.csv",
                                  "vel_viento_cuevas.csv"),
                   slot_name = c("rh", "patm", "precip",
                                  "tair", "wspd"),
                   by = c("hour", "45 min", "30 min", "1 hour", "15 min"),
                   FUN = read_csv ) %>%
hm_show()
```

# Excel files

###Recommended package => readxl

```r
library(readxl)
```

** Case 1: single file - one sheet - many numeric variables**

```r
hm_create() %>%
  hm_build_generic(path = my_path,
                   file_name = "mnemos_guido.xlsx",
                   slot_name = c("qd"),
                   by = c("day"),
                   FUN = read_excel,
                   sheet = 1L,
                   skip = 3,
                   out_name = list("q_m3/s")
) %>% hm_show()
```

** Case 2: single file - multiple sheets (one per variable)**

```r
hm_create() %>%
  hm_build_generic(path = my_path,
                   file_name = "mnemos_guido.xlsx",
                   slot_name = c("qd", "evap", "tair",
                                  "tmax", "tmin"),
                   by = c(q = "day", evap = "day", tair = "6 hour",
                              tmax = "day", tmin = "day"),
                   FUN = read_excel,
                   sheet = c(1L:5L),
                   skip = 3,
                   out_name = list( c("q_m3/s", "flag"),
                                  c("evap_mm", "flag"),
                                  c("tair", "flag"),
                                  )
) %>% hm_show()
```
hm_create

hm_create() =

 hm_build_generic(path = my_path, 
 file_name = c("discharge_daily.xlsx", 
"air_temperature_subdaily.xlsx"), 
 slot_name = c("qd", "tair"), 
 by = c(q = "day", tair = "6 hour"), 
 FUN = read_excel, 
 sheet = c(1L, 1L), 
 skip = 3, 
 out_name = list( c("q_m3/s", "flag"), 
 c("tair", "flag"))

) %>%
hm_show()

#* Case 3: multiple files - one sheet per file
hm_create() %>%
 hm_build_generic(path = my_path, 
 file_name = c("discharge_daily.xlsx", 
"air_temperature_subdaily.xlsx"), 
 slot_name = c("qd", "tair"), 
 by = c(q = "day", tair = "6 hour"), 
 FUN = read_excel, 
 sheet = c(1L, 1L), 
 skip = 3, 
 out_name = list( c("q_m3/s", "flag"), 
 c("tair", "flag"))

) %>%
hm_show()

## End(Not run)

hm_create

Creates an hydromet object.

Description
This function is the constructor of hydromet class and its subclass.

Usage
hm_create(class_name = "station")

Arguments

class_name string with the name of the class. Valid arguments are: hydromet, station or compact.

Value
An S4 object of class hydromet.
hm_get

Examples

# create class 'hydromet'
hym_metadata <- hm_create(class_name = 'hydromet')

# subclass 'station'
hym_station <- hm_create(class_name = 'station')

# subclass 'compact'
hym_compact <- hm_create(class_name = 'compact')

hm_get

Extract the slot

Description

Get the table (or metadata) that you want from an hydromet or hydromet_XXX class.

Usage

hm_get(obj, slot_name = NA_character_)

## S4 method for signature 'hydromet'
hm_get(obj, slot_name = NA_character_)

## S4 method for signature 'hydromet_station'
hm_get(obj, slot_name = NA_character_)

## S4 method for signature 'hydromet_compact'
hm_get(obj, slot_name = NA_character_)

Arguments

obj an hydromet or hydromet_XXX class object.
slot_name string with slot to extract.

Value

The required data frame or metadata.

Functions

- hm_get(hydromet): get method for generic hydromet object
- hm_get(hydromet_station): get method for station class
- hm_get(hydromet_compact): get method for compact class
Examples

```r
## Not run:
# set path to file
path_file <- system.file('extdata', 'ianigla_cuevas.csv',
                         package = 'hydrotoolbox')

# read file
cuevas <-
  read_ianigla(path = path_file,
              out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
                           'p(mm)', 'wspd(km/hr)', 'wdir(°)',
                           'kin(kW/m2)', 'hsnow(cm)', 'tsoil(°C)'))

# create and set one the variables
hm_cuevas <-
  hm_create() %>%
  hm_set(tair = cuevas[, c('date', 'tair(°C)')])

# now extract the slot of air temperature
head(hm_get(obj = hm_cuevas, slot_name = 'tair'))
```

## End(Not run)

hm_melt  

*Melt many objects into an hydromet_compact class object*

Description

This method allows you merge several tables (inside hydromet_station and/or hydromet_compact class objects) into a single one and set them into the compact slot (hydromet_compact class object).

Usage

```r
hm_melt(obj, melt, slot_name, col_name, out_name = NULL)
```

## S4 method for signature 'hydromet_compact'

hm_melt(obj, melt, slot_name, col_name, out_name = NULL)

Arguments

- `obj` : a valid hydromet_compact class object.
- `melt` : string vector containing the hydromet_xxx class objects names (as you have in the Global Environment) that you want for melting.
- `slot_name` : list (one element per melt vector name) with the slot(s) to extract per every hydromet_xxx class object (as string vectors).
hm_melt

col_name  string vector with the name of the variables to keep. You must comply the following name convention 'melt_slot_variable' (e.g.: 'guido_qd_Q(m3/s)' - where guido is your object name, qd is the slot with daily mean river discharge and Q(m3/s) is the required column name inside that slot). Another option is to set this argument just with the string 'all' and the method will preserve all the slot(s) columns. Other minimal option is to choose the string 'last': in this case you will get only the last column of each slot(s).

out_name  optional. String vector with the output names of the final table. If you use the default value (NULL) the method will add the object and slot name (provided in melt and slot_name argument) at the beginning of every column (e.g.: 'guido_qd_q(m3/s)').

Value

An hydromet_compact class object with a data frame inside the compact slot with all variables that you provided in col_name.

Functions

• hm_melt(hydromet_compact): plot method for compact class

Note

Remember that all the chosen variables should have the same temporal resolution. The method itself will not warn you about bad entries.

Examples

## Not run:
# lets say that we want to put together snow water equivalent from Toscas (dgi)
# and daily streamflow discharge from Guido (snih)

# path to all example files
path <- system.file('extdata', package = 'hydrotoolbox')

# on the first place we build the stations
# dgi file
toscas <-
  hm_create() %>%
  hm_build(bureau = 'dgi', path = path,
     file_name = 'dgi_toscas.xlsx',
     slot_name = c('swe', 'tmax',
       'tmin', 'tmean', 'rh', 'patm'),
     by = 'day',
     out_name = c('swe', 'tmax',
       'tmin', 'tmean', 'rh', 'patm') )

# snih file
guido <-
  hm_create() %>%
  hm_build(bureau = 'snih', path = path,
hm_mutate

Create, modify and delete columns inside a slot

Description

This method allows you to modify whatever (except 'date' column) you want inside a slot data frame. Since this package was designed with the aim of providing useful objects to store and track changes in hydro-meteorological series, is not recommend to delete or change the original data, but it is upon to you.

Usage

hm_mutate(obj, slot_name, FUN, ...)

## S4 method for signature 'hydromet_station'
hm_mutate(obj, slot_name, FUN, ...)

## S4 method for signature 'hydromet_compact'
hm_mutate(obj, slot_name, FUN, ...)

Arguments

obj a valid hydromet_XXX class object.
slot_name string with the a valid name.
function name. The function output must be a data frame with the first column being the Date. Note that hydrotoolbox provides common used hydrological functions: see for example `mov_avg`. An interesting function to use is `mutate` from `dplyr` package.

... FUN arguments to pass.

Value

The same object but with the modified slot’s data frame

Functions

- `hm_mutate(hydromet_station)`: method for station class.
- `hm_mutate(hydromet_compact)`: method for compact class.

Examples

```r
## Not run:
# path to all example files
path <- system.file('extdata', package = 'hydrotoolbox')

# build the snih station file
guido <-
hm_create() %>%
  hm_build(bureau = 'snih', path = path,
    file_name = c('snih_hq_guido.xlsx',
      'snih_qd_guido.xlsx'),
    slot_name = c('hq', 'qd'),
    by = c('none', 'day')) %>%
hm_name(slot_name = 'qd',
  col_name = 'q(m3/s)')

# apply a moving average windows to streamflow records
hm_mutate(obj = guido, slot_name = 'qd',
  FUN = mov_avg, k = 10,
  pos = 'c', out_name = 'mov_avg') %>%
hm_plot(slot_name = 'qd',
  col_name = list(c('q(m3/s)', 'mov_avg')) ),
  interactive = TRUE,
  line_color = c('dodgerblue', 'red3'),
  y_lab = 'Q(m3/s)',
  legend_lab = c('original', 'mov_avg') )

## End(Not run)
```
hm_name

Set new column names

Description

Change slot’s column names.

Usage

hm_name(obj, slot_name, col_name)

## S4 method for signature 'hydromet_station'
hm_name(obj, slot_name, col_name)

## S4 method for signature 'hydromet_compact'
hm_name(obj, slot_name, col_name)

Arguments

obj a valid hydromet_* class object.
slot_name string with the a valid name.
col_name string vector with new column names.

Value

The same object but with new column names.

Functions

- hm_name(hydromet_station): set new column name for station class
- hm_name(hydromet_compact): set new column name for compact class

Examples

## Not run:
# path to all example files
path <- system.file('extdata', package = 'hydrotoolbox')

# we first build the snih station file
guido <-
hm_create() %>%
hm_build(bureau = 'snih', path = path,
file_name = c('snih_hq_guido.xlsx', 'snih_qd_guido.xlsx'),
slot_name = c('hq', 'qd'),
by = c('none', 'day') )

guido %>% hm_show(slot_name = 'qd')
# now we can change default names
hm_name(obj = guido, slot_name = 'qd',
        col_name = 'q(m3/s)') %>%
hm_show(slot_name = 'qd')

## End(Not run)

hm_plot

Methods to easily use ggplot2 or plotly (interactive)

Description

This method allows you to make plots (using simple and expressive arguments) of the variables contained inside an hydromet_XXX class object. The plot outputs can be static (ggplot2) or dynamic (plotly).

Usage

hm_plot(
  obj,
  slot_name,
  col_name,
  interactive = FALSE,
  line_type = NULL,
  line_color = NULL,
  line_size = NULL,
  line_alpha = NULL,
  x_lab = "date",
  y_lab = "y",
  title_lab = NULL,
  legend_lab = NULL,
  dual_yaxis = NULL,
  from = NULL,
  to = NULL,
  scatter = NULL
)

## S4 method for signature 'hydromet_station'
hm_plot(
  obj,
  slot_name,
  col_name,
  interactive = FALSE,
  line_type = NULL,
  line_color = NULL,
  line_size = NULL,
line_alpha = NULL,
x_lab = "date",
y_lab = "y",
title_lab = NULL,
legend_lab = NULL,
dual_yaxis = NULL,
from = NULL,
to = NULL,
scatter = NULL
)

## S4 method for signature 'hydromet_compact'
hm_plot(
  obj,
  slot_name,
  col_name,
  interactive = FALSE,
  line_type = NULL,
  line_color = NULL,
  line_size = NULL,
  line_alpha = NULL,
  x_lab = "date",
  y_lab = "y",
  title_lab = NULL,
  legend_lab = NULL,
  dual_yaxis = NULL,
  from = NULL,
  to = NULL,
  scatter = NULL
)

Arguments

**obj**  
a valid hydromet.XXX class object.

**slot_name**  
string vector with the name of the slot(s) to use in plotting.

**col_name**  
list containing the column name of the variables to plot. Every element inside the list belongs to the previous defined slot(s).

**interactive**  
logical. Default value, FALSE, will return a ggplot2 class object. Otherwise you will get a plotly one.

**line_type**  
string with the name of the line dash type (ggplot2) or mode in the plotly case. ggplot2: 'solid' (default value), 'twodash', 'longdash', 'dotted', 'dotdash', 'dashed' or 'blank'. plotly: 'lines' (default value), 'lines+markers' or 'markers'. **NOTE:** when using scatter plot this arguments goes through the shape argument (in geom_point()) as numeric.

**line_color**  
string with a valid color name. See 'colors()' or Rcolor document.

**line_size**  
numeric vector containing the size of every line to plot. If you use the NULL value it will return the plots with default(s) for either ggplot2 or plotly.
hm_plot

<table>
<thead>
<tr>
<th>feature</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line_alpha</td>
<td>numeric vector with line(s) transparency. From 0 (invisible) to 1.</td>
</tr>
<tr>
<td>x_lab</td>
<td>string with x axis label. Default is 'Date'.</td>
</tr>
<tr>
<td>y_lab</td>
<td>string with y axis label. In case you use dual_yaxis argument you must supply both c('ylab', 'y2lab').</td>
</tr>
<tr>
<td>title_lab</td>
<td>string with the title of the plot. Default is a plot without title.</td>
</tr>
<tr>
<td>legend_lab</td>
<td>string vector with plot label(s) name(s).</td>
</tr>
<tr>
<td>dual_yaxis</td>
<td>string vector suggesting which variables are assign either to the 'left' or 'right' y axis.</td>
</tr>
<tr>
<td>from</td>
<td>string value for 'Date' class or POSIXct(lt) class for date-time data with the starting Date. You can use 'from' without 'to'. In this case you will subset your data 'from' till the end.</td>
</tr>
<tr>
<td>to</td>
<td>string value for 'Date' class or POSIXct(lt) class for date-time data with the ending Date. You can use 'to' without 'from'. In this case you will subset your data from the beginning till 'to'.</td>
</tr>
<tr>
<td>scatter</td>
<td>string vector (of length two) suggesting which variables goes in the 'x' and 'y' axis respectively. Valid character entries are 'x' and 'y'.</td>
</tr>
</tbody>
</table>

Value

A ggplot2 or plotly object.

Functions

- `hm_plot(hydromet_station)`: plot method for station class
- `hm_plot(hydromet_compact)`: plot method for compact class

Examples

```r
## Not run:
# lets work with the cuevas station
path <- system.file('extdata', package = 'hydrotoolbox')

# use the build method
hm_cuevas <-
  hm_create() %>%
  hm_build(bureau = 'ianigla', path = path,
    file_name = 'ianigla_cuevas.csv',
    slot_name = c('tair', 'rh', 'patm',
      'precip', 'wspd', 'wdir',
      'kin', 'hsnow', 'tsoil'),
    by = 'hour',
    out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
      'p(mm)', 'wspd(km/hr)', 'wdir(*)',
      'kin(kW/m2)', 'hsnow(cm)', 'tsoil(°C)')
  )

# let's start by making a single variable static plot
hm_plot(obj = hm_cuevas, slot_name = 'tair',
```
col_name = list('tair(°C)')

# we add labels, change color, line type and we focus
# on specific date range
hm_plot(obj = hm_cuevas, slot_name = 'tair',
col_name = list('tair(°C)'),
line_type = 'longdash',
line_color = 'dodgerblue',
x_lab = 'Date time', y_lab = 'T(°C)',
title_lab = 'Hourly temperature at Cuevas',
legend_lab = 'Tair',
from = ISOdate(2020, 7, 1),
to = ISOdate(2020, 7, 5))

# compare air with soil temperature
hm_plot(obj = hm_cuevas, slot_name = c('tair', 'tsoil'),
col_name = list('tair(°C)', 'tsoil(°C)'),
line_type = c('longdash', 'solid'),
line_color = c('dodgerblue', 'tan4'),
x_lab = 'Date time', y_lab = c('T(°C)'),
title_lab = 'Hourly temperature at Cuevas',
legend_lab = c('Tair', 'Tsoil'),
from = ISOdate(2020, 7, 1),
to = ISOdate(2020, 7, 5))

# let's add relative humidity on the right y-axis
hm_plot(obj = hm_cuevas, slot_name = c('tair', 'tsoil', 'rh'),
col_name = list('tair(°C)', 'tsoil(°C)', 'rh(%)'),
line_type = c('longdash', 'solid', 'solid'),
line_color = c('dodgerblue', 'tan4', 'red'),
x_lab = 'Date time', y_lab = c('T(°C)', 'RH(%))',
title_lab = 'Hourly meteo data at Cuevas',
legend_lab = c('Tair', 'Tsoil', 'RH'),
dual_yaxis = c('left', 'left', 'right'),
from = ISOdate(2020, 7, 1),
to = ISOdate(2020, 7, 5))

# we decide to analize the previous variables in detail
# with a dynamic plot
hm_plot(obj = hm_cuevas, slot_name = c('tair', 'tsoil', 'rh'),
col_name = list('tair(°C)', 'tsoil(°C)', 'rh(%)'),
line_color = c('dodgerblue', 'tan4', 'red'),
x_lab = 'Date time', y_lab = c('T(°C)', 'RH(%))',
title_lab = 'Hourly meteo data at Cuevas',
legend_lab = c('Tair', 'Tsoil', 'RH'),
dual_yaxis = c('left', 'left', 'right'),
interactive = TRUE)

# click on the Zoom icon and play a little...

# suppose now that we want to make a scatter plot to show
# the negative correlation between air temperature and
# relative humidity
hm_report

hm_plot(obj = hm_cuevas, slot_name = c('tair', 'rh'),
        col_name = list('tair(°C)', 'rh(%)'),
        line_color = 'DodgerBlue',
        x_lab = 'Tair', y_lab = 'RH',
        scatter = c('x', 'y'))

## End(Not run)

hm_report

*Get a summary report of your data*

Description

Returns a list with two elements: the first one contains basic statistics (mean, sd, max and min) values and the second one is a table with summary of miss data (see also `report_miss`).

Usage

```
hm_report(obj, slot_name, col_name = "all")
```

## S4 method for signature 'hydromet_station'

```
hm_report(obj, slot_name, col_name = "all")
```

## S4 method for signature 'hydromet_compact'

```
hm_report(obj, slot_name = "compact", col_name = "all")
```

Arguments

- `obj`  a valid `hydromet_XXX` class object.
- `slot_name`  string with the name of the slot to report.
- `col_name`  string vector with the column(s) name(s) to report. By default the function will do it in all columns inside the slot.

Value

A list summarizing basic statistics and missing data. The missing data table presents a data frame (one per `col_name`) with three columns: start-date, end-date and number of missing time steps. In the last row of this table you will find the total number of missing measurements (under "time_step" column). The "first" and "last" columns will have a `NA_character` for this last row.

Functions

- `hm_report(hydromet_station)`: report method for station class
- `hm_report(hydromet_compact)`: report method for compact class
Examples

```r
# Not run:
# cuevas station
path <- system.file('extdata', package = 'hydrotbxbox')

# use the build method
hm_cuevas <-
  hm_create() %>%
  hm_build(bureau = 'ianigla', path = path,
           file_name = 'ianigla_cuevas.csv',
           slot_name = c('tair', 'rh', 'patm',
                         'precip', 'wspd', 'wdir',
                         'kin', 'hsnow', 'tsoil'),
           by = 'hour',
           out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
                        'p(mm)', 'wspd(km/hr)', 'wdir(°)',
                        'kin(kW/m2)', 'hsnow(cm)', 'tsoil(°C)'))

# report incoming solar radiation
hm_report(obj = hm_cuevas, slot_name = 'kin')
```n
## End(Not run)

---

**hm_set**

*Set the data of an hydromet object or its subclass*

**Description**

With this method you can set (or change) an specific slot value (change the table).

**Usage**

```r
hm_set(
  obj = NULL,
  id = NULL,
  agency = NULL,
  station = NULL,
  lat = NULL,
  long = NULL,
  alt = NULL,
  country = NULL,
  province = NULL,
  river = NULL,
  active = NULL,
  basin_area = NULL,
  basin_eff = NULL,
)```
other_1 = NULL,
other_2 = NULL,
...
)

## S4 method for signature 'hydromet'
hm_set(
    obj = NULL,
id = NULL,
agency = NULL,
station = NULL,
lat = NULL,
long = NULL,
alt = NULL,
country = NULL,
province = NULL,
river = NULL,
active = NULL,
basin_area = NULL,
basin_eff = NULL,
other_1 = NULL,
other_2 = NULL,
...
)

## S4 method for signature 'hydromet_station'
hm_set(
    obj = NULL,
id = NULL,
agency = NULL,
station = NULL,
lat = NULL,
long = NULL,
alt = NULL,
country = NULL,
province = NULL,
river = NULL,
active = NULL,
basin_area = NULL,
basin_eff = NULL,
other_1 = NULL,
other_2 = NULL,
hq = NULL,
hw = NULL,
qh = NULL,
qd = NULL,
qa = NULL,
qm = NULL,
wspd = NULL,
wdir = NULL,
evap = NULL,
anem = NULL,
patm = NULL,
rh = NULL,
tair = NULL,
tmax = NULL,
tmin = NULL,
tmean = NULL,
tsoil = NULL,
precip = NULL,
rainfall = NULL,
swe = NULL,
hsnow = NULL,
kin = NULL,
kout = NULL,
lin = NULL,
lout = NULL,
unvar = NULL
)

## S4 method for signature 'hydromet_compact'

hm_set(
  obj = NULL,
  id = NULL,
  agency = NULL,
  station = NULL,
  lat = NULL,
  long = NULL,
  alt = NULL,
  country = NULL,
  province = NULL,
  river = NULL,
  active = NULL,
  basin_area = NULL,
  basin_eff = NULL,
  other_1 = NULL,
  other_2 = NULL,
  compact = NULL
)

Arguments

- **obj**: an hydromet or hydromet_XXX class object.
- **id**: ANY. This is the ID assigned by the agency.
- **agency**: character. The name of the agency (or institution) that provides the data of the station.
station character. The name of the (hydro)-meteorological station.
latt numeric. Latitude of the station.
longt numeric. Longitude of the station.
altt numeric. Altitude of the station.
country character. Country where the station is located. Argentina is set as default value.
province character. Name of the province where the station is located. Mendoza is set as default value.
river character. Basin river’s name.
active logical. It indicates whether or not the station is currently operated. Default value is TRUE.
basin_area numeric. The basin area (km2) of the catchment upstream of the gauge.
basin_eff numeric. The effective area (km2) of the basin upstream of the gauge. In Canada, many basins have variable contributing fractions. In these basins, the effective area of the basin contributes flow to the outlet at least one year in two.
other_1 ANY. It is the first free-to-fill slot in order to give you the chance to write extra information about your hydro-met station.
other_2 ANY. It is the second free-to-fill slot in order to give you the chance to write extra information about your hydro-met station.
... arguments to be passed to methods. They rely on the slots of the obj subclass.
hq water-height vs stream-discharge measurements.
hlw water level records.
qh hourly mean river discharge.
qd daily mean river discharge.
qa annual river discharge.
qm monthly mean river discharge.
wspd wind speed.
wdir wind direction.
evap pan-evaporation.
anem anemometer wind speed records (usually installed above the pan-evap tank).
patm atmospheric pressure.
rh relative humidity.
tair air temperature (typically recorded at hourly time-step).
tmax daily maximum recorded air temperature.
tmin daily minimum recorded air temperature.
tmean daily mean air temperature.
tsoil soil temperature.
precip total (snow and rain) precipitation records.
rainfall liquid only precipitation measurements.
swe snow water equivalent (typically recorded on snow pillows).
hsnow  snow height from ultrasonic devices.
k in   incoming short-wave radiation.
kout   outgoing short-wave radiation.
lin    incoming long-wave radiation.
lout   outgoing long-wave radiation.
unvar  reserved for non-considered variables.
compact data frame with Date as first column. All other columns are hydro-meteorological variables.

Value

The hydromet object with the slots set.

Functions

- `hm_set(hydromet)`: set method for generic object
- `hm_set(hydromet_station)`: set method for station object
- `hm_set(hydromet_compact)`: set method for compact object

Examples

```r
## Not run:
# create an hydro-met station
hm_guido <- hm_create(class_name = 'station')

# assign altitude
hm_guido <- hm_set(obj = hm_guido, alt = 2480)

# now we read streamflow - water height measurements
path_file <- system.file('extdata', 'snih_hq_guido.xlsx', package = 'hydrotoolbox')
guido_hq <- read_snih(path = path_file, by = 'none',
                      out_name = c('h(m)', 'q(m^3/s)',
                                  'q_coarse_solid(kg/s)',
                                  'q_fine_solid(kg/s)'))

# set the new data frame
# note: you can do it manually but using the hm_build() method
# is strongly recommended
hm_guido <- hm_set(obj = hm_guido, hq = guido_hq)
hm_show(obj = hm_guido)
```

```
## End(Not run)
```
hm_show

Easy access to see your data

Description

This method shows the 'head' or 'tail' of a specific slot.

Usage

hm_show(obj, slot_name = "fill", show = "head")

## S4 method for signature 'hydromet'
hm_show(obj, slot_name = "fill", show = "head")

## S4 method for signature 'hydromet_station'
hm_show(obj, slot_name = "fill", show = "head")

## S4 method for signature 'hydromet_compact'
hm_show(obj, slot_name = "compact", show = "head")

Arguments

obj a valid hydromet_XXX class object.
slot_name string vector with the name of the slot(s) to show. Alternatively you can use 'fill' or 'empty' to get the data frames with or without data respectively.
show string with either 'head' or 'tail'.

Value

It prints the data inside the required slot.

Functions

• hm_show(hydromet): print method for hydromet class
• hm_show(hydromet_station): print method for station class
• hm_show(hydromet_compact): print method for compact class

Examples

## Not run:
# lets work with the cuevas station
path <- system.file('extdata', package = 'hydrotoolbox')

# use the build method
hm_cuevas <-
  hm_create() %>%
  hm_build(bureau = 'ianigla', path = path,
hm_subset

Subset your data by dates

Description

The method will subset the required slot.

Usage

hm_subset(obj, slot_name = "all", from = NULL, to = NULL)

## S4 method for signature 'hydromet_station'
hm_subset(obj, slot_name = "all", from = NULL, to = NULL)

## S4 method for signature 'hydromet_compact'
hm_subset(obj, slot_name = "all", from = NULL, to = NULL)

Arguments

obj a valid hydromet_XXX class object.

slot_name string vector with the name(s) of the slot(s) to subset. If you use 'all' as argument
the method will subset all the variables with data.
from string Date or POSIX* value with the starting date. You can use from without to. In this case you will subset your data from till the end.

to string Date or POSIX* value with the starting date. You can use to without from. In this case you will subset your data from the beginning till to.

Value

The same hydromet XXX class object provided in obj but subsetted.

Functions

- hm_subset(hydromet_station): subset method for station class
- hm_subset(hydromet_compact): subset method for compact class

Examples

```r
## Not run:
# cuevas station
path <- system.file(’extdata’, package = ’hydrotoolbox’)

# use the build method
hm_cuevas <-
  hm_create() %>%
  hm_build(bureau = ’ianigla’, path = path,
           file_name = ’ianigla_cuevas.csv’,
           slot_name = c(’tair’, ’rh’, ’patm’,
                         ’precip’, ’wspd’, ’wdir’,
                         ’kin’, ’hsnow’, ’tsoil’),
           by = ’hour’,
           out_name = c(’tair(°C)’, ’rh(%)’, ’patm(mbar)’,
                        ’p(mm)’, ’wspd(km/hr)’, ’wdir(°)’,
                        ’kin(kW/m2)’, ’hsnow(cm)’, ’tsoil(°C)’)
  )

# subset relative humidity and plot it
hm_subset(obj = hm_cuevas, slot_name = ’rh’,
          from = ISOdate(2020, 2, 1),
          to = ISOdate(2020, 4, 1) %>%
hm_plot(slot_name = ’rh’,
        col_name = list(’rh(%)’),
        interactive = TRUE,
        y_lab = ’RH(%)’)

## End(Not run)
**hydromet-class**  
**hydromet superclass object**

---

**Description**

A suitable object for store basic information about an hydro-meteorological station.

**Value**

A basic hydromet class object. This class is provided in order to set the meta-data of the station.

**Slots**

- **id** ANY. This is the ID assigned by the agency.
- **agency** string. The name of the agency (or institution) that provides the data of the station.
- **station** string. The name of the (hydro)-meteorological station.
- **lat** numeric. Latitude of the station.
- **long** numeric. Longitude of the station.
- **alt** numeric. Altitude of the station.
- **country** string. Country where the station is located. Argentina is set as default value.
- **province** string. Name of the province where the station is located. Mendoza is set as default value.
- **river** string. Basin river's name.
- **active** logical. It indicates whether or not the station is currently operated. Default value is TRUE.
- **basin_area** numeric. The basin area (km2) of the catchment upstream of the gauge.
- **basin_eff** numeric. The effective area (km2) of the basin upstream of the gauge. In Canada, many basins have variable contributing fractions. In these basins, the effective area of the basin contributes flow to the outlet at least one year in two.
- **other_1** ANY. It is the first free-to-fill slot in order to give you the chance to write extra information about your hydro-met station.
- **other_2** ANY. It is the second free-to-fill slot in order to give you the chance to write extra information about your hydro-met station.

**Examples**

```r
## Not run:
# create class hydromet
hm_create(class_name = "hydromet")

## End(Not run)
```
hydromet_compact-class

**Description**

This subclass is useful for storing in a single data frame ready to use hydro-meteorological series or many variables of the same kind (e.g. precipitation series).

**Value**

A hydromet_compact class object.

**Slots**

compact data.frame with Date as first column (class 'Date' or 'POSIXct'). All other columns are the numeric hydro-meteorological variables (double). This subclass was though to join in a single table ready to use data (e.g. in modeling). You can also use it to put together variables of the same kind (e.g. precipitation records) to make some regional analysis.

**Examples**

```r
## Not run:
# create an compact station
hm_create(class_name = "compact")

## End(Not run)
```

hydromet_station-class

**Description**

A suitable object for store your hydro-meteorological data.

**Value**

An hydromet_station class object.
Slots

hq  water-height vs stream-discharge measurements.
hw  water level records.
qh  hourly mean river discharge.
qd  daily mean river discharge.
qm  monthly mean river discharge.
qa  annual river discharge.
wspd  wind speed.
wdir  wind direction.
evp  pan-evaporation.
anem  anemometer wind speed records (usually installed above the pan-evap tank).
patm  atmospheric pressure.
rh  relative humidity.
tair  air temperature (typically recorded at hourly time-step).
tmax  daily maximum recorded air temperature.
tmin  daily minimum recorded air temperature.
tmean  daily mean air temperature.
tsoll  soil temperature.
precip  total (snow and rain) precipitation records.
rainfall  liquid only precipitation measurements.
swe  snow water equivalent (typically recorded on snow pillows).
hsnow  snow height from ultrasonic devices.
kin  incoming short-wave radiation.
kout  outgoing short-wave radiation.
lin  incoming long-wave radiation.
lout  outgoing long-wave radiation.
unvar  reserved for non-considered variables.

Examples

```r
## Not run:
# create an hydromet station
hm_create(class_name = "station")

## End(Not run)
```
interpolate  

Interpolation

Description

This function applies interpolation to fill in missing (or non-recorded) values.

Usage

interpolate(
  x,
  col_name,
  out_name = NULL,
  miss_table,
  threshold,
  method = "linear"
)

Arguments

- **x**: data frame with class Date or POSIX* in the first column and numeric on the others.
- **col_name**: string with column name of the series to interpolate.
- **out_name**: optional. String with new column name. If you set it as NULL, the function will overwrite the original data frame.
- **miss_table**: data frame with three columns: first and last date of interpolation (first and second column respectively). The last and third column, is of class numeric with the number of steps to interpolate. See `report_miss`.
- **threshold**: numeric variable with the maximum number of dates in which to apply the interpolation.
- **method**: string with the interpolation method. In this version only 'linear' method is allowed.

Value

The same data frame but with interpolated values.

Examples

```r
# read cuevas station file
path <- system.file('extdata', 'ianigla_cuevas.csv',
  package = 'hydrotoolbox')

cuevas <- read_ianigla(path = path)
```
# get the miss_table
miss_data <- report_miss(x = cuevas, col_name = 'Irradiancia')[[1]]

# apply interpolation function when gap is less than 3 hours
cuevas_interpo <- interpolate(x = cuevas, 
       col_name = 'Irradiancia', 
       out_name = 'kin_interpo', 
       miss_table = miss_data, 
       threshold = 3)

report_miss(x = cuevas_interpo, 
       col_name = c('Irradiancia', 'kin_interpo'))

---

**mov_avg**

*Moving average windows*

**Description**

Smooth numeric series with a moving average windows.

**Usage**

```r
cf{mov_avg(x, col_name = "last", k, pos = "c", out_name = NULL, from = NULL, to = NULL)}
```

**Arguments**

- `x` data frame (or tibble) with class Date or POSIX* in the first column.
- `col_name` string vector with the column(s) name(s) of the series to smooth. The default value uses the 'last' column. Another single string choice is to use 'all'. Is important to keep in mind that this argument **commands**, so if you provide two columns names, `k` and `pos` arguments must be of length two; if not the single value will be recycled.
- `k` numeric vector with the windows size. E.g.: `k = 5`.
- `pos` string vector with the position of the windows:
  - 'c': center (default). The output value is in the middle of the window.
  - 'l': left aligned. The output value is on the left, so the function weights the `(k - 1)` values at the right side.
qm_vol

- `r`: right aligned. The output value is on the right, so the function weights the \((k - 1)\) values at the left side.

**out_name**
optional. String vector with new column names. If you set it as NULL the function will overwrite the original series.

**from**
optional. String value for 'Date' class or POSIX** class for date-time data containing the starting Date.

**to**
optional. String value for 'Date' class or POSIX** class for date-time data containing the ending Date.

**Value**
The same data frame but with the smooth series.

**Examples**

```r
# read guido daily streamflow records
path <- system.file('extdata', 'snih_qd_guido.xlsx',
    package = 'hydrotoolbox')

# read and apply the function
qd_guido <-
    read_snih(path = path, by = 'day', out_name = 'q(m3/s)') %>%
    mov_avg(k = 5, out_name = 'q_smooth')
```

### qm_vol

**Monthly river discharge [m3/s] to volume [hm3]**

**Description**
Converts mean monthly river discharge [m3/s] to total volume discharge [hm3].

**Usage**

```r
qm_vol(x, col_name, out_name = NULL)
```

**Arguments**

- **x**
data frame with class Date in the first column and numeric on the others.

- **col_name**
string with column(s) name(s) where to apply the function.

- **out_name**
optional. String with new column(s) name(s). If you set it as NULL, the function will overwrite the original data frame.
Value

The same data frame but with the total volume discharge.

Examples

```r
# read guido daily streamflow records
path <- system.file('extdata', 'snih_qd_guido.xlsx',
                     package = 'hydrotoolbox')

# read, aggregate the function to monthly resolution and get the volume
qm_guido <-
  read_snih(path = path, by = 'day', out_name = 'q(m3/s)') %>%
  agg_table(col_name = 'q(m3/s)', fun = 'mean', period = 'monthly',
            out_name = 'qm(m3/s)') %>%
  qm_vol(col_name = 'qm(m3/s)', out_name = 'vm(hm3)')
```

---

**read_aic**

Reads data from AIC

### Description

Reads excel files provided by the AIC.

### Usage

```r
read_aic(
  path,
  by = "day",
  out_name = NULL,
  sheet = NULL,
  skip = 12,
  get_sheet = FALSE
)
```

### Arguments

- **path**
  - path to the xlsx file.
- **by**
  - string with the time step of the series (e.g. 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). By default this argument is set to 'day'. If you set it as 'none', the function will ignore automatic gap filling.
- **out_name**
  - optional. String vector with user defined variable(s) column(s) name(s).
- **sheet**
  - optional. Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). If neither argument specifies the sheet, defaults to the first sheet.
**read_cr2**

```r
skip
```

optional. Minimum number of rows to skip before reading anything, be it column names or data. Leading empty rows are automatically skipped, so this is a lower bound.

```r
get_sheet
```

logical indicating whether you want to print available sheet names (TRUE) in the file or not.

**Value**

A data frame with the data inside the xlsx file. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically.

**Examples**

```r
# This files are provided by AIC under legal agreement only.
```

---

**read_cr2 (Reads data from Explorador Climático (CR2 - Chile))**

**Description**

Reads csv files downloaded from the CR2 web page as a data frame.

**Usage**

```r
read_cr2(path, by = "day", out_name = NULL)
```

**Arguments**

- **path**: path to the csv file.
- **by**: string with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). The default and unique possible value is 'day'.
- **out_name**: optional. String vector with user defined variable(s) column(s) name(s).

**Value**

A data frame with the data inside the csv file. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically.
Examples

```r
# list cr2 files
list.files( system.file('extdata', package = 'hydrotoolbox'), pattern = 'cr2' )

# set path to file
path_tmax <- system.file('extdata', 'cr2_tmax_yeso_embalse.csv',
                          package = 'hydrotoolbox')

# read file with default colname
head( read_cr2(path = path_tmax) )

# assign a column name
head( read_cr2(path = path_tmax, out_name = 'tmax(°C)') )
```

---

read_dgi

*Reads data from Departamento General de Irrigación - Hydrological Division (DGI - Mendoza - Argentina)*

Description

Reads excel files provided by the DGI (Hydrological Division).

Usage

```r
read_dgi(path, by = "day", out_name = NULL, sheet = NULL, get_sheet = FALSE)
```

Arguments

- **path**  
  path to the xlsx file.

- **by**  
  string with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). By default this argument is set to 'day'. If you set it as 'none', the function will ignore automatic gap filling.

- **out_name**  
  optional. String vector with user defined variable(s) column(s) name(s).

- **sheet**  
  optional. Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). If neither argument specifies the sheet, defaults to the first sheet.

- **get_sheet**  
  logical indicating whether you want to print available sheet names (TRUE) in the file or not.

Value

A data frame with the data inside the xlsx file. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically.
Examples

```r
# set path to file
path_file <- system.file('extdata', 'dgi_toscas.xlsx',
                          package = 'hydrotoolbox')

# because dgi files has multiple sheets we take a look
# on them
read_dgi(path = path_file, get_sheet = TRUE)

# read swe with default column names
head( read_dgi(path = path_file, sheet = 'swe') )

# assign name
head( read_dgi(path = path_file, sheet = 'swe', out_name = 'swe(mm') )

# now read relative humidity
head( read_dgi(path = path_file, sheet = 'hr', out_name = 'rh(%)') )
```

---

**read_ianigla**

*Reads data from Sistema de Monitoreo Meteorológico de Alta Montaña (IANIGLA - Argentina)*

**Description**

Reads csv files downloaded from the Sistema de Monitoreo Meteorológico de Alta Montaña web page as a data frame.

**Usage**

`read_ianigla(path, by = "1 hour", out_name = NULL)`

**Arguments**

- `path` - path to the csv file.
- `by` - string with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). The default value is '1 hour'. If you set it as 'none', the function will ignore automatic gap filling.
- `out_name` - optional. String vector with user defined variable(s) column(s) name(s).

**Value**

A data frame with the data inside the csv file. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically.
Examples

```r
# set path to file
path_file <- system.file('extdata', 'ianigla_cuevas.csv',
                          package = 'hydrotoolbox')

# read with default names
head( read_ianigla(path = path_file) )

# set column names
head( read_ianigla(path = path_file,
                  out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
                               'p(mm)', 'wspd(km/hr)', 'wdir(*)',
                               'kin(kW/m2)', 'hsnow(cm)', 'tsoil(°C)') )
)
```

read_mnemos

Reads data provided by MNEMOS software (SNIH - Argentina)

Description

Reads xlsx files generated with MNEMOS software.

Usage

```r
read_mnemos( 
  path, 
  by = "none", 
  out_name = NULL, 
  sheet = NULL, 
  skip = 3, 
  get_sheet = FALSE 
)
```

Arguments

- **path**: path to the xlsx file.
- **by**: string with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). If you set it as 'none', the function will ignore automatic gap filling.
- **out_name**: optional. String vector with user defined variable(s) column(s) name(s).
- **sheet**: optional. Sheet to read. Either a string (the name of a sheet), or an integer (the position of the sheet). If neither argument specifies the sheet, defaults to the first sheet.
**read_snih**

*skip* optional. Minimum number of rows to skip before reading anything, be it column names or data. Leading empty rows are automatically skipped, so this is a lower bound.

*get_sheet* logical indicating whether you want to print available variables (TRUE) in every file sheet or not.

**Value**

A data frame with the data inside the specified sheet. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically. In case you set `get_sheet = TRUE` the function will return a list with the variables inside each sheet.

**Examples**

```r
# list mnemos files
list.files( system.file('extdata', package = 'hydrotoolbox'), pattern = 'mnemos' )

# set path
path <- system.file('extdata', 'mnemos_guido.xlsx', package = 'hydrotoolbox')

# we can see which variables are inside the sheet's file
read_mnemos(path = path, get_sheet = TRUE)

# now we want to read the maximum temperature
tmax_guido <- read_mnemos(path = path, by = 'day',
                        out_name = 'tmax(ºC)', sheet = '11413-016')
```

**Description**

reads data from Servicio Nacional de Información Hídrica [R](https://back.argentina.gob.ar/obras-publicas/hidricas/base-de-datos-hidrologica-integrada(SNIH - Argentina))

**Usage**

```r
read_snih(path, by, out_name = NULL)
```
**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>path to the xlsx file.</td>
</tr>
<tr>
<td>by</td>
<td>string with the time step of the series (e.g.: 'month', 'day', '6 hour', '3 hour', '1 hour', '15 min'). If you set it as 'none', the function will ignore automatic gap filling.</td>
</tr>
<tr>
<td>out_name</td>
<td>optional. String vector with user defined variable(s) column(s) name(s).</td>
</tr>
</tbody>
</table>

**Value**

A data frame with the data inside the xlsx file. Gaps between dates are filled with `NA_real_` and duplicated rows are eliminated automatically.

**Examples**

```r
# set path to file
doors_file <- system.file('extdata', 'snih_qd_guido.xlsx', package = 'hydrotoolbox')

# read daily streamflow with default column name
head( read_snih(path = doors_file, by = 'day') )

# now we use the function with column name
head( read_snih(path = doors_file, by = 'day', out_name = 'qd(m3/s)') )
```

---

**Description**

Creates a data frame with reported dates and number of times-step of missing or not recorded data.

**Usage**

```r
report_miss(x, col_name = "all")
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>data frame with hydro-meteo data. First column is date and the second the numeric vector to be reported.</td>
</tr>
<tr>
<td>col_name</td>
<td>string vector with the column(s) name(s) to report. By default the function will report all numeric columns.</td>
</tr>
</tbody>
</table>
Value

A list containing a data frame (one per `col_name`) with three columns: start-date, end-date and number of missing time steps. In the last row of the table you will find the total number of missing measurements (under "time_step" column). The "first" and "last" columns will have a `NA_character` for this last row.

Examples

```r
# read guido daily streamflow records
path <- system.file('extdata', 'snih_qd_guido.xlsx',
                     package = 'hydrotoolbox')

# load raw data
qd_guido <-
  read_snih(path = path, by = 'day', out_name = 'q(m3/s)') %>%
  mov_avg(k = 5, out_name = 'q_smooth')

# get the data report
qd_guido %>%
  report_miss()
```

---

<table>
<thead>
<tr>
<th><code>rm_spike</code></th>
<th>Remove spikes</th>
</tr>
</thead>
</table>

Description

Remove spikes and set their value as `NA_real_`.

Usage

`rm_spike(x, col_name, out_name = NULL, tolerance)`

Arguments

- `x` : data frame or tibble with class Date or POSIX* in the first column.
- `col_name` : string with column(s) name(s) where to apply the function.
- `out_name` : optional. String with new column(s) name(s). If you set it as NULL, the function will overwrite the original table.
- `tolerance` : numeric vector with the maximum tolerance between a number and its successor. If you provide a single value it will be recycled.

Value

The same table but with the peaks removed.
### Examples

```r
# set path to file
path_file <- system.file('extdata', 'ianigla_cuevas.csv',
                          package = 'hydrotoolbox')

# read with default names
cuevas <- read_ianigla(path = path_file,
                       out_name = c('tair(°C)', 'rh(%)', 'patm(mbar)',
                                    'p(mm)', 'wspd(km/hr)', 'wdir(°)',
                                    'kin(kW/m2)', 'hsnow(cm)', 'tsoil(°C)'))

# remove spikes in snow height series
cuevas %>%
  rm_spike(col_name = 'hsnow(cm)',
           out_name = 'hsnow',
           tolerance = 50)  # 50 cm of snow is OK for this zone
```

---

### roll_fun

**Rolling functions**

#### Description

It provides a generic function to rolling table columns. Internally it is using rollapplyr from package zoo.

#### Usage

```r
roll_fun(
  x,
  col_name = "last",
  k,
  pos = "c",
  FUN,
  ...,
  out_name = NULL,
  from = NULL,
  to = NULL
)
```

#### Arguments

- **x**: data frame (or tibble) with class Date or POSIX* in the first column.
- **col_name**: string vector with the column(s) name(s) of the series to roll. The default value uses the 'last' column. Another single string choice is to use 'all'. It is important to keep in mind that this argument commands, so if you provide two
columns names, k and pos arguments must be of length two; if not the single value will be recycled.

k	numeric vector with the windows size. E.g.: k = 5.
pos
string vector with the position of the windows:
  • 'c': center (default). The output value is in the middle of the window.
  • 'l': left aligned. The output value is on the left, so the function weights the \((k - 1)\) values at the right side.
  • 'r': right aligned. The output value is on the right, so the function weights the \((k - 1)\) values at the left side.

FUN
the function to be applied.

Value
The same table but with the rolling series.

Examples

```r
# read guido daily streamflow records
path <- system.file('extdata', 'snih_qd_guido.xlsx',
  package = 'hydrotoolbox')

# read and apply the function
qd_guido <-
  read_snih(path = path, by = 'day', out_name = 'q(m3/s)') %>%
  roll_fun(k = 5, FUN = mean, na.rm = TRUE, out_name = 'q_smooth')
```

### set_threshold

**Set a threshold**

**Description**

Set tolerable extreme values (maximum or minimum). Records greater or equal than ('\(\geq\)') or lesser or equal than ('\(\leq\)') 'threshold' argument are set to `NA_real_`. 
Usage

```r
set_threshold(x, col_name, out_name = NULL, threshold, case = ">=")
```

Arguments

- `x`: data frame or tibble with class Date or POSIX* in the first column.
- `col_name`: string with column(s) name(s) where to apply the function.
- `out_name`: optional. String with new column(s) name(s). If you set it as NULL, the function will overwrite the original table.
- `threshold`: numeric vector with the threshold value(s). If you provide a single value it will be recycled among col_name strings.
- `case`: string with either ">=" (greater or equal than) or "<=" (lesser or equal than) symbol. Default string is ">=".

Value

The same data frame but with the threshold set.

Examples

```r
# set path to file
path_file <- system.file('/extdata', 'ianigla_cuevas.csv',
                          package = 'hydrotoolbox')

# read with default names
cuevas <- read_ianigla(path = path_file,
                       out_name = c('tair(\degree C)', 'rh(\%)', 'patm(mbar)',
                                    'p(mm)', 'wspd(km/hr)', 'wdir(\degree)',
                                    'kin(kW/m2)', 'hsnow(cm)', 'tsoil(\degree C)'))

# remove values higher than 1.50 meters
cuevas %>%
  set_threshold(col_name = 'hsnow(cm)',
                out_name = 'hsnow_thres',
                threshold = 150)
```

---

**set_value**

Set user defined values

Description

Specify specific values between dates.
set_value

Usage

set_value(x, col_name, out_name = NULL, value, from, to)

Arguments

x        data frame or tibble with class Date or POSIX* in the first column.
col_name string with column(s) name(s) to set.
out_name optional. String with new column(s) name(s). If you set it as NULL, the function will overwrite the original data frame.
value    numeric vector with the numeric values to set between dates (from and to). If you provide a number it will be recycled. When using a multiple dates (i.e.: "date" vector in from and to) use a list with a numeric vector inside each element.
from     string vector for 'Date' class or POSIX* class for date-time data with the starting date.
to       string vector for 'Date' class or POSIX* class for date-time data with the ending date.

Value

The same table but with the set numeric values between the dates.

Examples

# create a data frame
dates <- seq.Date(from = as.Date('1990-01-01'), to = as.Date('1990-12-01'), by = 'm')
met_var <- runif(n = 12, 0, 10)
met_table <- data.frame(dates, met_var)

# set single value recycling
set_value(x = met_table, col_name = 'met_var', value = 10,
           from = '1990-01-01', to = '1990-06-01')

# set different periods
set_value(x = met_table, col_name = 'met_var', value = list(NA_real_, c(1, 2)),
           from = c('1990-01-01', '1990-11-01'), to = c('1990-06-01', '1990-12-01'))

# now set as new columns
set_value(x = met_table, col_name = 'met_var', out_name = 'met_set',
           value = list(NA_real_, c(1, 2)),
           from = c('1990-01-01', '1990-11-01'),
           to = c('1990-06-01', '1990-12-01'))
swe_derive

_Snow Water Equivalent to melt or snowfall_

**Description**

Derive melt or snowfall series from snow water equivalent measurements (snow pillows measurements).

**Usage**

```
swe_derive(x, col_name, out_name = NULL, case)
```

**Arguments**

- `x` data frame or tibble with class Date or POSIX* in the first column.
- `col_name` string with column(s) name(s) where to apply the function.
- `out_name` optional. String with new column(s) name(s). If you set it as NULL, the function will overwrite the original table.
- `case` string vector with "sf" (meaning snowfall) or "m" (meaning melt).

**Value**

The same data frame but with the derived series.

**Examples**

```r
# set path to file
path_file <- system.file('extdata', 'dgi_toscas.xlsx',
                         package = 'hydrotoolbox')

# swe table
swe_toscas <- read_dgi(path = path_file,
                        sheet = 'swe',
                        out_name = 'swe(mm)')

# add melt and snowfall
swe_toscas <-
  swe_toscas %>%
  swe_derive(col_name = rep('swe(mm)', 2),
             out_name = c('melt(mm)', 'snowfall(mm)'),
             case = c('m', 'sf'))
```
Index

agg_table, 2

cum_sum, 4

fill_table, 5

hm_agg, 6
hm_agg, hydromet_compact-method (hm_agg), 6
hm_agg, hydromet_station-method (hm_agg), 6
hm_build, 8
hm_build, hydromet_station-method (hm_build), 8
hm_build_generic, 11
hm_build_generic, hydromet_station-method (hm_build_generic), 11
hm_create, 14
hm_get, 15
hm_get, hydromet-method (hm_get), 15
hm_get, hydromet_compact-method (hm_get), 15
hm_get, hydromet_station-method (hm_get), 15
hm_melt, 16
hm_melt, hydromet_compact-method (hm_melt), 16
hm_mutate, 18
hm_mutate, hydromet_compact-method (hm_mutate), 18
hm_mutate, hydromet_station-method (hm_mutate), 18
hm_name, 20
hm_name, hydromet_compact-method (hm_name), 20
hm_name, hydromet_station-method (hm_name), 20
hm_plot, 21
hm_plot, hydromet_compact-method (hm_plot), 21
hm_plot, hydromet_station-method (hm_plot), 21
hm_report, 25
hm_report, hydromet_compact-method (hm_report), 25
hm_report, hydromet_station-method (hm_report), 25
hm_set, 26
hm_set, hydromet-method (hm_set), 26
hm_set, hydromet_compact-method (hm_set), 26
hm_set, hydromet_station-method (hm_set), 26
hm_show, 31
hm_show, hydromet-method (hm_show), 31
hm_show, hydromet_compact-method (hm_show), 31
hm_show, hydromet_station-method (hm_show), 31
hm_subset, 32
hm_subset, hydromet_compact-method (hm_subset), 32
hm_subset, hydromet_station-method (hm_subset), 32
hydromet (hydromet-class), 34
hydromet-class, 34
hydromet_compact (hydromet_compact-class), 35
hydromet_compact-class, 35
hydromet_station (hydromet_station-class), 35
hydromet_station-class, 35

interpolate, 37

mov_avg, 19, 38

qm_vol, 39

read_aic, 40
read_cr2, 41
read_dgi, 42
read_ianigla, 43
read_mnemos, 44
read_snih, 45
report_miss, 25, 37, 46
rm_spike, 47
roll_fun, 48

set_threshold, 49
set_value, 50
swe_derive, 52