Package: clifro (via r-universe)

November 27, 2024

Type Package

Title Easily Download and Visualise Climate Data from CliFlo

Version 3.2-5.9003

Date 2023-03-09

VignetteBuilder knitr

Imports methods, stats, graphics, lubridate, xml2, magrittr, utils, ggplot2 (>= 2.0.0), scales, RColorBrewer, reshape2, rvest, httr, stringr

Suggests spelling, knitr, rmarkdown, pander, testthat

Description CliFlo is a web portal to the New Zealand National Climate Database and provides public access (via subscription) to around 6,500 various climate stations (see <https://cliflo.niwa.co.nz/> for more information). Collating and manipulating data from CliFlo (hence clifro) and importing into R for further analysis, exploration and visualisation is now straightforward and coherent. The user is required to have an internet connection, and a current CliFlo subscription (free) if data from stations, other than the public Reefton electronic weather station, is sought.

URL https://docs.ropensci.org/clifro/, https://github.com/ropensci/clifro

BugReports https://github.com/ropensci/clifro/issues

License GPL-2

Encoding UTF-8

Collate 'dataFrame.R' 'cfStation.R' 'cfData.R' 'cfDataList.R' 'cfData-plotMethods.R' 'cfDatatype.R' 'cfQuery.R' 'cfUser.R' 'findStations.R'

RoxygenNote 7.2.3

X-schema.org-keywords r, opensci, zealand, weather, climate, cliflo, data, api, windrose, rain, wind, temperature

X-schema.org-applicationCategory Data Access

X-schema.org-isPartOf ``https://ropensci.org"

Language en-GB

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/clifro

RemoteRef master

RemoteSha 46c72d399ee3d6069cfffa87a4452f85794bdef9

Contents

`+,cfStation,cfStation-method`

Arithmetic Operators for Clifro Objects

Description

This operator allows you to add more datatypes or stations to `cfDatatype` and `cfStation` objects respectively.

Usage

```
## S4 method for signature 'cfStation,cfStation'
e1 + e2

## S4 method for signature 'cfDatatype,cfDatatype'
e1 + e2
```

Arguments

e1	a cfDatatype or cfStation object
e2	an object matching the class of e1

---

cfDatatype-class *The Clifro Datatype Object*

Description

Create a cfDatatype object by selecting one or more CliFlo datatypes to build the **clifro** query.

Usage

```
cf_datatype(
  select_1 = NA,
  select_2 = NA,
  check_box = NA,
  combo_box = NA,
  graphics = FALSE
)
```

Arguments

select_1	a numeric vector of first node selections
select_2	a numeric vector of second node selections
check_box	a list containing the check box selections
combo_box	a numeric vector containing the combo box selection (if applicable)
graphics	a logical indicating whether a graphics menu should be used, if available

Details

An object inheriting from the cfDatatype class is created by the constructor function cf_datatype. The function allows the user to choose datatype(s) interactively (if no arguments are given), or to create datatypes programmatically if the tree menu nodes are known a priori (see examples). This function uses the same nodes, check box and combo box options as CliFlo and can be viewed at the datatype selection page.

Value

cfDatatype object

Note

For the 'public' user (see examples) only the Reefton Ews station data is available.

Currently clifro does not support datatypes from the special datasets (Ten minute, Tier2, Virtual Climate, Lysimeter) or upper air measurements from radiosondes and wind radar.

See Also

cf_user to create a **clifro** user, cf_station to choose the CliFlo stations and vignette("choose-datatype") for help choosing cfDatatypes.

Examples

```
## Not run:
# Select the surface wind datatype manually (unknown tree nodes)
hourly.wind.dt = cf_datatype()
#  2  --> Datatype:        Wind
#  1  --> Datatype 2:      Surface Wind
#  2  --> Options:         Hourly Wind
# (2) --> Another option:  No
#  3  --> Units:           Knots
hourly.wind.dt

# Or select the datatype programatically (using the selections seen above)
hourly.wind.dt = cf_datatype(2, 1, 2, 3)
hourly.wind.dt

## End(Not run)
```

---

cfStation-class *The Clifro Station Object*

Description

Create a cfStation object containing station information for one or more CliFlo stations.

Usage

```
cf_station(...)
```

Arguments

...	comma separated agent numbers

Details

A `cfStation` object is created by the constructor function `cf_station`. The unique agent numbers of the stations are all that is required to create a `cfStation` object using the `cf_station` function. The rest of the station information including the name, network and agent ID, start and end dates, coordinates, as well as other data is scraped from CliFlo.

This function is used for when the agent numbers are already known. For help creating `cfStation` objects when the agent numbers are unknown see the `cf_find_station` function.

Value

`cfStation` object

See Also

`cf_find_station` for creating `cfStation` objects when the agent numbers are not known and `vignette("cfStation")` for working with clifro stations including spatial plotting in R. For saving `cfStation` objects as KML files refer to the vignette or `cf_save_kml`.

Examples

```
## Not run:
# Create a cfStation object for the Leigh 1 and 2 Ews stations
leigh.st = cf_station(1339, 1340)
leigh.st

# Note, this can also be achieved using the '+' operator
leigh.st = cf_station(1339) + cf_station(1340)
leigh.st

# Add another column showing how long the stations have been open for
leigh.df = as(leigh.st, "data.frame")
leigh.df$ndays = with(leigh.df, round(end - start))
leigh.df

# Save the stations to the current working directory as a KML to visualise
# the station locations
cf_save_kml(leigh.st)

## End(Not run)
```

---

cfUser-class *The Clifro User Object*

Description

Create a `cfUser` object to allow the user to log into CliFlo from R and build their query.

Usage

```
cf_user(username = "public", password = character())
```

Arguments

username	a character string to be used as the cliflo username
password	a character string to be used as the cliflo password

Details

An object inheriting from the cfUser class is created by the constructor function cf_user. The user must have an active subscription to cliflo in order to create a valid object, unless a 'public' user is sought. Visit https://cliflo.niwa.co.nz/ for more information and to subscribe to cliflo.

Value

cfUser object

Note

For the 'public' user (see examples) only the Reefton Ews station data is available.

See Also

valid_cfuser for details on the validation of cfUser and summary,cfUser-method to summarise user information.

Examples

```
## Not run:
public.cfuser = cf_user(username = "public")
public.cfuser

## End(Not run)
```

---

cf_curl_opts *Store curl options for use within* **clifro**

Description

The cf_curl_opts function stores specific curl options that are used for all the **clifro** queries.

Usage

```
cf_curl_opts(..., .opts = list())
```

Arguments

...	a name-value pairs that are passed to RCurl curlOptions
.opts	a named list or CURLOptions object that are passed to RCurl curlOptions

Examples

```
## Not run:
# Specify options for use in all the curl handles created in clifro
cf_curl_opts(.opts = list(proxy = "http://xxxxx.yyyy.govt.nz:8080",
                          proxyusername  = "uid",
                          proxypassword  = "pwd",
                          ssl.verifypeer = FALSE))
# Or alternatively:
cf_curl_opts(proxy = "http://xxxxx.yyyy.govt.nz:8080",
             proxyusername  = "uid",
             proxypassword  = "pwd",
             ssl.verifypeer = FALSE)

## End(Not run)
```

---

cf_find_station *Search for Clifro Stations*

Description

Search for **clifro** stations based on name, region, location or network number, and return a cfStation object.

Usage

```
cf_find_station(
  ...,
  search = c("name", "region", "network", "latlong"),
  datatype,
  combine = c("all", "any"),
  status = c("open", "closed", "all")
)
```

Arguments

...	arguments to pass into the search, these differ depending on search.
search	one of name, network, region or latlong indicating the type of search to be conducted.
datatype	cfDatatype object for when the search is based on datatypes.
combine	character string "all" or "any" indicating if the stations contain all or any of the selected datatypes for when the search is based on datatypes.
status	character string indicating "open", "closed" or "all" stations be returned by the search.

Details

The cf_find_station function is a convenience function for finding CliFlo stations in R. It uses the CliFlo Find Stations page to do the searching, and therefore means that the stations are not stored within **clifro**.

If datatype is missing then the search is conducted without any reference to datatypes. If it is supplied then the search will only return stations that have any or all of the supplied datatypes, depending on combine. The default behaviour is to search for stations based on pattern matching the station name and return only the open stations.

If the latlong search type is used the function expects named arguments with names (partially) matching latitude, longitude and radius. If the arguments are passed in without names they must be in order of latitude, longitude and radius (see examples).

Value

cfStation object

Note

Since the searching is done by CliFlo there are obvious restrictions. Unfortunately the pattern matching for station name does not provide functionality for regular expressions, nor does it allow simultaneous searches although **clifro** does provide some extra functionality, see the 'OR query Search' example below.

See Also

cf_save_kml for saving the resulting stations as a KML file, cf_station for creating cfStation objects when the agent numbers are known, vignette("choose-station") for a tutorial on finding **clifro** stations and vignette("cfStation") for working with cfStation objects.

Examples

```
## Not run:
# Station Name Search ------------------------------------------------------
# Return all open stations with 'island' in the name (pattern match search)
# Note this example uses all the defaults

island_st = cf_find_station("island")
island_st

# Region Search ------------------------------------------------------------
# Return all the closed stations from Queenstown (using partial matching)

queenstown.st = cf_find_station("queen", search = "region", status = "closed")
queenstown.st

# Long/Lat Search ----------------------------------------------------------
# Return all open stations within a 10km radius of the Beehive in Wellington
# From Wikipedia: latitude 41.2784 S, longitude 174.7767 E

beehive.st = cf_find_station(lat = -41.2784, long = 174.7767, rad = 10,
```

```
                              search = "latlong")
beehive.st

# Network ID Search ----------------------------------------------------------
# Return all stations that share A42 in their network ID

A42.st = cf_find_station("A42", search = "network", status = "all")
A42.st

# Using Datatypes in the Search ----------------------------------------------
# Is the Reefton EWS station open and does it collect daily rain and/or wind
# data?

# First, create the daily rain and wind datatypes
daily.dt = cf_datatype(c(2, 3), c(1, 1), list(4, 1), c(1, NA))
daily.dt

# Then combine into the search. This will only return stations where at least
# one datatype is available.
cf_find_station("reefton EWS", datatype = daily.dt)  # Yes

# OR Query Search ------------------------------------------------------------
# Return all stations sharing A42 in their network ID *or* all the open
# stations within 10km of the Beehive in Wellington (note this is not
# currently available as a single query in CliFlo).

cf_find_station("A42", search = "network", status = "all") +
cf_find_station(lat = -41.2784, long = 174.7767, rad = 10,
                search = "latlong")

# Note these are all ordered by open stations, then again by their end dates

## End(Not run)
```

`cf_last_query` *Retrieve Last Query Result from CliFlo*

Description

Retrieve the last query submitted to CliFlo instead of querying the database again and losing subscription rows.

Usage

```
cf_last_query()
```

Details

This function is a back up for when the clifro query has been submitted and the data returned but has not been assigned, or inadvertently deleted. This saves the user resubmitting queries and using more rows from their subscription than needed.

Note

Only the data from the last query is saved in `clifro`.

Examples

```
## Not run:
# Query CliFlo for wind at Reefton Ews
cf_query(cf_user(), cf_datatype(2, 1, 1, 1), cf_station(), "2012-01-01 00")

# Oops! Forgot to assign it to a variable...
reefton.wind = cf_last_query()
reefton.wind

## End(Not run)
```

---

cf_query *Retrieve Data from the National Climate Database*

Description

Query the National Climate Database via CliFlo based on the **clifro** user and selected datatypes, stations and dates.

Usage

```
cf_query(
  user,
  datatype,
  station,
  start_date,
  end_date = now(tz),
  date_format = "ymd_h",
  tz = "Pacific/Auckland",
  output_tz = c("local", "NZST", "UTC"),
  quiet = FALSE
)
```

Arguments

`user`	a `cfUser` object.
`datatype`	a `cfDatatype` object containing the datatypes to be retrieved.
`station`	a `cfStation` object containing the stations where the datatypes will be retrieved from.
`start_date`	a character, Date or POSIXt object indicating the start date. If a character string is supplied the date format should be in the form `yyyy-mm-dd-hh` unless `date_format` is specified.

end_date	same as start_date. Defaults to now.
date_format	a character string matching one of "ymd_h", "mdy_h", "ydm_h" or "dmy_h" representing the lubridate-package date parsing function.
tz	the timezone for which the start and end dates refer to. Conversion to Pacific/Auckland time is done automatically through the with_tz function. Defaults to "Pacific/Auckland".
output_tz	the timezone of the output. This can be one of either "local", "UTC", or "NZST".
quiet	logical. When TRUE the function evaluates without displaying customary messages. Messages from CliFlo are still displayed.

Details

The cf_query function is used to combine the **clifro** user (cfUser), along with the desired datatypes (cfDatatype) and stations (cfStation). The query is 'built up' using these objects, along with the necessary dates. The function then uses all these data to query the National Climate Database via the CliFlo web portal and returns one of the many cfData objects if one dataframe is returned, or a cfDataList object if there is more than one dataframe returned from CliFlo. If a cfDataList is returned, each element in the list is a subclass of the cfData class, see the 'cfData Subclasses' section.

Value

a cfData or cfDataList object.

CfData Subclasses

There are 8 cfData subclasses that are returned from cf_query depending on the datatype requested. Each of these subclasses have default plot methods for usability and efficiency in exploring and plotting **clifro** data.

The following table summarises these subclasses and how they are created, see also the examples on how to automatically create some of these subclasses.

Subclass	CliFlo Datatype
cfWind	Any 'Wind' data
cfRain	Any 'Precipitation' data
cfScreen Obs	'Temperature and Humidity' data measured in a standard screen
cfTemp	Maximum and minimum 'Temperature and Humidity' data
cfEarthTemp	'Temperature and Humidity' data at a given depth
cfSunshine	Any 'Sunshine & Radiation' data
cfPressure	Any 'Pressure' data
cfOther	Any other CliFlo 'Daily and Hourly Observations'

See Also

cf_user, cf_datatype and cf_station for creating the objects needed for a query. See plot,cfDataList,missing-metho for general information on default plotting of cfData and cfDataList objects, and the links within.

Examples

```
## Not run:
# Retrieve daily rain data from Reefton Ews
daily.rain = cf_query(cf_user("public"), cf_datatype(3, 1, 1),
                      cf_station(), "2012-01-01 00")
daily.rain

# returns a cfData object as there is only one datatype
class(daily.rain) # 'cfRain' object - inherits 'cfData'

# Look up the help page for cfRain plot methods
?plot.cfRain

# Retrieve daily rain and wind data from Reefton Ews

daily.dts = cf_query(cf_user("public"),
                     cf_datatype(c(2, 3), c(1, 1), list(4, 1), c(1, NA)),
                     cf_station(), "2012-01-01 00", "2013-01-01 00")
daily.dts

# returns a cfDataList object as there is more than one datatype. Each
# element of the cfDataList is an object inheriting from the cfData class.
class(daily.dts)     # cfDataList
class(daily.dts[1])  # cfRain
class(daily.dts[2])  # cfWind

# Create a cfSunshine object (inherits cfData)
# Retrieve daily global radiation data at Reefton Ews
rad.data = cf_query(cf_user(), cf_datatype(5,2,1), cf_station(),
                    "2012-01-01 00")
rad.data

# The cf_query function automatically creates the appropriate cfData subclass
class(rad.data)

# The advantage of having these subclasses is that it makes plotting very easy
plot(rad.data)
plot(daily.rain)
plot(daily.rain, include_runoff = FALSE)
plot(daily.dts)
plot(daily.dts, 2)

## End(Not run)
```

`cf_save_kml` *Save Clifro Station Information to a KML File*

Description

Save `cfStation` object information to a KML file.

Usage

```
cf_save_kml(station, file_name = "my_stations_", file_path = ".")
```

Arguments

station	cfStation object containing one or more stations
file_name	file name for the resulting KML file
file_path	file path for the resulting KML file

Details

The cf_save_kml function is for cfStation objects to allow for the spatial visualisation of the selected stations. The resulting KML file is saved and can then be opened by programs like Google Earth (TM). The resultant KML file has the station names and locations shown with green markers for open and red markers for closed stations. The agent numbers, network ID's and date ranges are contained within the descriptions for each station.

If no file name is specified, unique names are produced in the current R working directory.

Note

The .kml suffix is appended automatically if it isn't already present in the file_name argument.

See Also

cf_station and vignette("cfStation") for working with stations when the agent numbers are known, otherwise cf_find_station and codevignette("choose-station") for creating cfStation objects when the agent numbers are unknown.

Examples

```
## Not run:
# A selection of four Auckland region stations down the East Coast to the
# upper Waitemata Harbour; Leigh 2 Ews, Warkworth Ews, Tiri Tiri Lighthouse
# and Henderson
my.stations = cf_station(17838, 1340, 1401, 12327)
my.stations

# Save these stations to a KML file
cf_save_kml(my.stations)

# Double click on the file to open with a default program (if available). All
# the markers are green, indicating all these stations are open.

# Where is the subscription-free Reefton Ews station?
cf_save_kml(cf_station(), file_name = "reeftonEWS")

# It's located in the sou'west quadrant of Reefton town, in the upper, western
# part of the South Island, NZ.

# Find all the open and closed Christchurch stations (using partial matching)
```

```
all.chch.st = cf_find_station("christ", status = "all", search = "region")

# How many stations in total?
nrow(all.chch.st)

# Save all the Christchurch stations
cf_save_kml(all.chch.st, file_name = "all_Chch_stations")

## End(Not run)
```

---

clifro *From CliFlo to* **clifro***: Enhancing The National Climate Database With* R

Description

Import data from New Zealand's National Climate Database via CliFlo into R for exploring, analysis, plotting, exporting to KML, CSV, or other software.

Details

The **clifro** package is intended to simplify the process of data extraction, formatting and visualisation from the CliFlo web portal. It requires the user to build a query consisting of 3 main components; the user, the datatype(s) and the station(s). These are then combined using the `cf_query` function that sends the query to the CliFlo database and returns the results that can easily be plotted using generic plotting functions.

This package requires the user to already have a current subscription to the National Climate Database unless a public user is sought, where data is limited to Reefton Ews. Subscription is free and can obtained from https://cliflo.niwa.co.nz/pls/niwp/wsubform.intro.

See Also

`cf_user`, `cf_datatype`, and `cf_station` for choosing the clifro user, datatypes and stations, respectively.

Examples

```
## Not run:
# Create a public user ------------------------------------------------

public.user = cf_user() # Defaults to "public"
public.user

# Select datatypes ----------------------------------------------------

# 9am Surface wind (m/s)
wind.dt = cf_datatype(2, 1, 4, 1)

# Daily Rain
```

```
rain.dt = cf_datatype(3, 1, 1)

# Daily temperature extremes
temp.dt = cf_datatype(4, 2, 2)

# Combine them together
all.dts = wind.dt + rain.dt + temp.dt
all.dts

# Select the Reefton Ews station ----------------------------------------

reefton.st = cf_station()
reefton.st

# Submit the query ------------------------------------------------------

# Retrieve all data from ~ six months ago at 9am
reefton.data = cf_query(public.user, all.dts, reefton.st,
                        paste(as.Date(Sys.time()) - 182, "9"))
reefton.data


# Plot the data ---------------------------------------------------------

# Plot the 9am surface wind data (first dataframe in the list) ---
reefton.data[1]

# all identical - although passed to different methods
plot(reefton.data)    #plot,cfDataList,missing-method
plot(reefton.data, 1) #plot,cfDataList,numeric-method
plot(reefton.data[1]) #plot,cfData,missing-method --> plot,cfWind,missing-method

speed_plot(reefton.data)
direction_plot(reefton.data)

# Plot the daily rain data (second dataframe in the list) ---
reefton.data[2]

# With runoff and soil deficit
plot(reefton.data, 2)

# Just plot amount of rain (mm)
plot(reefton.data, 2, include_runoff = FALSE)

# Plot the hourly temperature data (third dataframe in the list) ---
plot(reefton.data, 3)

# Pass an argument to ggplot2::theme
library(ggplot2) # for element_text()
plot(reefton.data, 3, text = element_text(size = 18))

## End(Not run)
```

`dimnames,dataFrame-method`

Dimension Attributes of a Clifro Object

Description

Retrieve the dimensions or dimension names of a `dataFrame` object.

Usage

```
## S4 method for signature 'dataFrame'
dimnames(x)

## S4 method for signature 'dataFrame'
dim(x)
```

Arguments

`x`	a `dataFrame` object Specifically, a `dataFrame` object is any `cfStation` or `cfData` object. These functions are provided for the user to have (some) familiar `data.frame`-type functions available for use on **clifro** objects.

See Also

`cf_query` for creating `cfData` objects, and `cf_station` for creating `cfStation` objects.

`plot,cfEarthTemp,missing-method`

Plot Earth Temperatures

Description

Plot the earth temperature for a given depth (degrees celsius) through time, for each chosen CliFlo station.

Usage

```
## S4 method for signature 'cfEarthTemp,missing'
plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

x	a cfEarthTemp object.
y	missing.
ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.

See Also

plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfEarthTemp objects.

Refer to theme for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public earth temperature data for the last 30 days at Reefton Ews
# station, at a depth of 10cm

# Subtract 30 days from today's date to get the start date
last_month = paste(as.character(Sys.Date() - 30), 0)

reefton_earth = cf_query(cf_user(), cf_datatype(4, 3, 2), cf_station(),
                         start_date = last_month)

class(reefton_earth) # cfTemp object

# Plot the temperature data using the defaults
plot(reefton_earth)

# Enlarge the text and add the observations as points
library(ggplot2) # for element_text() and geom_point()
plot(reefton_earth, ggtheme = "bw", text = element_text(size = 16)) +
  geom_point(size = 3, shape = 1)

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_earthTemp_plot.png")

## End(Not run)
```

plot,cfPressure,missing-method

Plot Mean Sea Level Atmospheric Pressure

Description

Plot the MSL atmospheric pressure through time.

Usage

```
## S4 method for signature 'cfPressure,missing'
plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

x	a cfPressure object.
y	missing.
ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.

See Also

plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfPressure objects.

Refer to theme for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public hourly atmospheric pressure data for the last 30 days at
# Reefton Ews station

# Subtract 30 days from today's date to get the start date
last_month = paste(as.character(Sys.Date() - 30), 0)
```

```
reefton_pressure = cf_query(cf_user(), cf_datatype(7, 1, 1), cf_station(),
                            start_date = last_month)

class(reefton_pressure) # cfPressure object

# Plot the atmospheric pressure data using the defaults
plot(reefton_pressure)

# Enlarge the text and add the observations as points
library(ggplot2) # for element_text() and geom_point()
plot(reefton_pressure, ggtheme = "bw", text = element_text(size = 16)) +
  geom_point(size = 3, shape = 1)

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_pressure_plot.png")

## End(Not run)
```

`plot,cfRain,missing-method`

Plot Rain Time series

Description

Plot the amount of rainfall (mm) through time, with optional available soil water capacity and runoff amounts (if applicable).

Usage

```
## S4 method for signature 'cfRain,missing'
plot(
  x,
  y,
  include_runoff = TRUE,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

`x`	a `cfRain` object.
`y`	missing.
`include_runoff`	a logical indicating whether to plot the soil moisture deficit and runoff as well as the rainfall, if the data is available (default `TRUE`).

ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.

Details

When there is a rain event, the amount of runoff, if any, is dependent on how much capacity the soil has available for more water. If there is no available water capacity left in the soil then more rain will lead to a runoff event. If include_runoff = TRUE, the available water capacity is plotted as negative values and the runoff as positive values to signify this negative relationship.

See Also

plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfRain objects.

Refer to theme for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public rain data for a month from CliFlo (at Reefton Ews station)
reefton_rain = cf_query(cf_user(), cf_datatype(3, 1, 1), cf_station(),
                        start_date = "2012-08-01-00",
                        end_date = "2012-09-01-00")

class(reefton_rain) # cfRain object

# Plot the rain data using the defaults
plot(reefton_rain)

# Change the ggtheme and enlarge the text
library(ggplot2) # for element_text()
plot(reefton_rain, ggtheme = "bw", text = element_text(size = 16))

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_rain_plot.png")

## End(Not run)
```

plot,cfScreenObs,missing-method
Plot Screen Observations

Description

Plot temperature data from screen observations (degrees celsius) through time.

Usage

```
## S4 method for signature 'cfScreenObs,missing'
plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

x	a cfScreenObs object.
y	missing.
ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.

Details

Temperature data from screen observations include the air, and wet bulb, temperature at the time the measurement was taken (dry bulb and wet bulb respectively), and the dew point. The dew point is the air temperature at which dew starts to form. That is the temperature to which a given air parcel must be cooled at constant pressure and constant water vapour content in order for saturation to occur.

The resulting figure plots the dry bulb, wet bulb and dew point temperatures on the same scale, for each station.

References

Screen Observation details.

See Also

plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfScreenObs objects.

Refer to theme for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public temperature data from screen observations for the last week
# at Reefton Ews station

# Subtract 7 days from today's date to get the start date
last_week = paste(as.character(Sys.Date() - 7), 0)

reefton_screenobs = cf_query(cf_user(), cf_datatype(4, 1, 1), cf_station(),
                             start_date = last_week)

class(reefton_screenobs) # cfScreenObs object

# Plot the temperature data using the defaults
plot(reefton_screenobs)

# Enlarge the text and add the observations as points
library(ggplot2) # for element_text() and geom_point()
plot(reefton_screenobs, ggtheme = "bw", text = element_text(size = 16)) +
  geom_point(size = 3, shape = 1)

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_screenobs_plot.png")

## End(Not run)
```

---

plot,cfSunshine,missing-method

Plot Sunshine Hours

Description

Plot the duration of accumulated bright sunshine hours through time.

Usage

```
## S4 method for signature 'cfSunshine,missing'
plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
```

```
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

x	a cfSunshine object.
y	missing.
ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.

See Also

plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfSunshine objects.

Refer to theme for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public hourly sunshine data for the last 7 days at Reefton Ews
# station

# Subtract 7 days from today's date to get the start date
last_week = paste(as.character(Sys.Date() - 7), 0)

reefton_sun = cf_query(cf_user(), cf_datatype(5, 1, 2), cf_station(),
                       start_date = last_week)

class(reefton_sun) # cfSunshine object

# Plot the temperature data using the defaults
plot(reefton_sun)

# Enlarge the text and add the observations as points
library(ggplot2) # for element_text() and geom_point()
plot(reefton_sun, ggtheme = "bw", text = element_text(size = 16)) +
  geom_point(size = 3, shape = 1)

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_sunshine_plot.png")

## End(Not run)
```

`plot,cfTemp,missing-method`
Plot Temperature Range

Description

Plot minimum and maximum temperature data for a given period (degrees celsius) through time, for each chosen CliFlo station.

Usage

```
## S4 method for signature 'cfTemp,missing'
plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)
```

Arguments

`x`	a cfTemp object.
`y`	missing.
`ggtheme`	character string (partially) matching the `ggtheme` to be used for plotting, see 'Theme Selection' below.
`scales`	character string partially matching the `scales` argument in the `link[ggplot2]{facet_wrap}` function.
`n_col`	the number of columns of plots (default 1).
`...`	further arguments passed to `theme`.

Details

This plotting method shows the temperature extremes as a grey region on the plot, with a black line indicating the average temperature (if available).

See Also

`plot,cfDataList,missing-method` for general information on default plotting of `cfData` and `cfDataList` objects, and the links within. See `cf_query` for creating `cfTemp` objects.

Refer to `theme` for more possible arguments to pass to these methods.

Examples

```
## Not run:
# Retrieve public hourly minimum and maximum temperature data for the last
week at Reefton Ews station

# Subtract 7 days from today's date to get the start date
last_week = paste(as.character(Sys.Date() - 7), 0)

reefton_temp = cf_query(cf_user(), cf_datatype(4, 2, 2), cf_station(),
                        start_date = last_week)

class(reefton_temp) # cfTemp object

# Plot the temperature data using the defaults
plot(reefton_temp)

# Enlarge the text and add the observations as points
library(ggplot2) # for element_text() and geom_point()
plot(reefton_temp, ggtheme = "bw", text = element_text(size = 16)) +
  geom_point(size = 3, shape = 1)

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_temperature_plot.png")

## End(Not run)
```

---

plot,cfWind,missing-method

Plot Clifro Wind Objects

Description

Various plot methods for exploring wind speed and direction patterns for given CliFlo stations.

Usage

```
## S4 method for signature 'cfWind,missing'
plot(
  x,
  y,
  n_directions = 12,
  n_speeds = 5,
  speed_cuts = NULL,
  col_pal = "GnBu",
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  n_col = 1,
  ...
```

```
)

## S4 method for signature 'cfWind,missing'
direction_plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  contours = 10,
  n_col = 1,
  ...
)

## S4 method for signature 'cfDataList,missing'
direction_plot(x, y, ...)

## S4 method for signature 'cfDataList,numeric'
direction_plot(x, y, ...)

## S4 method for signature 'cfWind,missing'
speed_plot(
  x,
  y,
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  scales = c("fixed", "free_x", "free_y", "free"),
  n_col = 1,
  ...
)

## S4 method for signature 'cfDataList,missing'
speed_plot(x, y, ...)

## S4 method for signature 'cfDataList,numeric'
speed_plot(x, y, ...)
```

Arguments

x	a cfWind or cfDataList object.
y	missing if x is a .cfWind object, otherwise a number indicating the dataframe to plot in the cfDataList (defaults to 1).
n_directions	the number of direction bins to plot (petals on the rose). The number of directions defaults to 12.
n_speeds	the number of equally spaced wind speed bins to plot. This is used if spd_cuts is NA (default 5).
speed_cuts	numeric vector containing the cut points for the wind speed intervals, or NA (default).
col_pal	character string indicating the name of the RColorBrewer colour palette to be used for plotting, see 'Theme Selection' below.

ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
n_col	the number of columns of plots (default 1).
...	further arguments passed to theme.
contours	the number of contour lines to draw (default 10).
scales	character string partially matching the scales argument in the link[ggplot2]{facet_wrap} function.

Details

If x is a cfDataList, by default the first datatype will be plotted unless y is supplied.

Theme Selection

For black and white windroses that may be preferred if plots are to be used in journal articles for example, recommended ggthemes are 'bw', 'linedraw', 'minimal' or 'classic' and the col_pal should be 'Greys'. Otherwise, any of the sequential RColorBrewer colour palettes are recommended for colour plots.

Note

If x is a cfDataList object and y refers to a **clifro** dataframe that is not a cfWind object then it will be passed to another method, if available.

The default plot method plots a different windrose for each CliFlo station. The direction_plot method plots wind direction contours through time to visualise temporal patterns in wind directions. The speed_plot method plots the time series of wind speeds with a +/- standard deviation region (if applicable).

Given a value on the x-axis, the ends of the density function along the y-axis are not constrained to be equal for any of the derivatives for the direction_plot method. That is, the contours at direction = 0, do not match the contours at direction = 360.

@seealso plot,cfDataList,missing-method for general information on default plotting of cfData and cfDataList objects, and the links within. See cf_query for creating cfWind objects or windrose for plotting any wind data. Refer to theme for more possible arguments to pass to these methods. summary,cfWind-method for summarising wind information at each CliFlo station.

Examples

```
## Not run:
# Retrieve maximum wind gust data at the Reefton Ews station from CliFlo
# (public data)
reefton_wind = cf_query(cf_user(), cf_datatype(2, 2, 1, 1), cf_station(),
                        start_date = "2012-01-01-00")

class(reefton_wind)

# Examples of the default plots ------------------------------------------

# Plot a windrose
```

```
plot(reefton_wind)

# Plot the wind direction contours
direction_plot(reefton_wind)

# Plot the wind speed time-series
speed_plot(reefton_wind)

# Examples of changing the defaults ----------------------------------------

# Plot black and white windroses
plot(reefton_wind, ggtheme = "bw", col_pal = "Greys")
plot(reefton_wind, ggtheme = "linedraw", col_pal = "Greys")
plot(reefton_wind, ggtheme = "classic", col_pal = "Greys")
plot(reefton_wind, ggtheme = "minimal", col_pal = "Greys")

# Plot the wind directions using 20 contours and the ggtheme 'classic'
direction_plot(reefton_wind, ggtheme = "classic", contours = 20)

# Enlarge all the text to 18pt
library(ggplot2) # for element_text() and geom_point()
direction_plot(reefton_wind, ggtheme = "classic", contours = 20,
               text = element_text(size = 18))

# Include the actual observations in the plots
direction_plot(reefton_wind) + geom_point(alpha = .2, size = 3)

speed_plot(reefton_wind, ggtheme = "classic", text = element_text(size = 16)) +
  geom_point(shape = 1, size = 3)
# or equivalently using base graphics:
plot(reefton_wind$Date, reefton_wind$Speed, type = 'o',
     xlab = NA, ylab = "Daily max gust (m/s)", las = 1, main = "Reefton Ews")

# Example of plotting a cfDataList -----------------------------------------
# Collect both surface wind run and hourly surface wind observations from
# Reefton Ews
reefton_list = cf_query(cf_user(), cf_datatype(2, 1, 1:2, 1),
                        cf_station(), "2012-01-01 00", "2012-02-01 00")

reefton_list

class(reefton_list) #cfDataList

# Plot the first (default) dataframe
plot(reefton_list) # Error - no wind directions for wind run datatypes
# Try speed_plot instead
speed_plot(reefton_list)

# Plot the second dataframe in the cfDataList
plot(reefton_list, 2)           # identical to plot(reefton_list[2])
speed_plot(reefton_list, 2)     # identical to speed_plot(reefton_list[2])
direction_plot(reefton_list, 2) # identical to direction_plot(reefton_list[2])
```

```
# Save the ggplot externally ----------------------------------------------

# Save the plot as a png to the current working directory
library(ggplot2) # for ggsave()
ggsave("my_wind_plot.png")

## End(Not run)
```

`plot.cfDataList` *Default Clifro Plotting*

Description

Plot **clifro** data based on the datatype.

Usage

```
## S4 method for signature 'cfDataList,numeric'
plot(x, y, ...)

## S4 method for signature 'cfDataList,missing'
plot(x, y, ...)

## S4 method for signature 'cfOther,missing'
plot(x, y)
```

Arguments

`x`	a `cfData` or `cfDataList` object.
`y`	missing for `cfData` objects, or a number representing the dataframe to plot if `x` is a `cfDataList` object.
`...`	arguments passed onto the different plotting methods.

These methods are intended to simplify the data visualisation and exploration of CliFlo data. The type of plot is determined by the type of the data output from a **clifro** query. All of these methods plot individual plots for each CliFlo station (if there is more than one in the query). If `x` is a `cfDataList`, by default the first datatype will be plotted unless `y` is supplied.

The following table links the datatypes to the corresponding plot methods:

Datatype	Method
Wind	`plot.cfWind` for windrose, wind speed and direction contour plots
Rain	`plot.cfRain` for plotting rainfall (mm) through time
Screen Obs	`plot.cfScreenObs` for time series plots of air, wet bulb, and dew-point temperature plots
Max/Min Temp	`plot.cfTemp` for maximum, minimum and average temperature time series plots
Earth Temp	`plot.cfEarthTemp` for earth temperature time series plots
Sunshine	`plot.cfSunshine` for accumulated, hourly or daily sunshine, time series plots

Pressure	`plot.cfPressure` for mean sea level atmospheric pressure time series plots
Other data	No default plot methods

See Also

`cf_query` to retrieve the CliFlo data and create `cfData` objects.

Refer to `theme` for more possible arguments to pass to these methods.

`summary,cfUser-method` *Summarise User Information*

Description

Show the subscription status for the **clifro** user

Usage

```
## S4 method for signature 'cfUser'
summary(object)
```

Arguments

`object`	an object of class `cfUser`.

`summary,cfWind-method` *Summarise Clifro Wind Data*

Description

This is a summary method for `cfWind` objects.

Usage

```
## S4 method for signature 'cfWind'
summary(object, calm_wind = 0)
```

Arguments

`object`	a `cfWind` object.
`calm_wind`	a single number containing the wind speed that is considered calm.

Details

A dataframe is returned containing the percentage of calm days (wind speed >= `calm_days`), percentage of variable days (wind speed = 990), and quantiles from the empirical cumulative distribution functions for each CliFlo station at which there is wind data.

See Also

plot.cfWind for default plotting of clifro wind data, and cf_query for creating cfWind objects.

Examples

```
## Not run:
# Retrieve maximum wind gust data at the Reefton Ews station from CliFlo
# (public data)
reefton_wind = cf_query(cf_user(), cf_datatype(2, 2, 1, 1), cf_station(),
                        start_date = "2012-01-01-00")

class(reefton_wind) # cfWind object

# Summarise the information
summary(reefton_wind)

## End(Not run)
```

---

windrose *Plot a Wind Rose*

Description

Plot a wind rose showing the wind speed and direction for given facets using **ggplot2**.

Usage

```
windrose(
  speed,
  direction,
  facet,
  n_directions = 12,
  n_speeds = 5,
  speed_cuts = NA,
  col_pal = "GnBu",
  ggtheme = c("grey", "gray", "bw", "linedraw", "light", "minimal", "classic"),
  legend_title = "Wind Speed",
  calm_wind = 0,
  variable_wind = 990,
  n_col = 1,
  ...
)
```

Arguments

speed	numeric vector of wind speeds.
direction	numeric vector of wind directions.

facet	character or factor vector of the facets used to plot the various wind roses.
n_directions	the number of direction bins to plot (petals on the rose). The number of directions defaults to 12.
n_speeds	the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	numeric vector containing the cut points for the wind speed intervals, or NA (default).
col_pal	character string indicating the name of the RColorBrewer colour palette to be used for plotting, see 'Theme Selection' below.
ggtheme	character string (partially) matching the ggtheme to be used for plotting, see 'Theme Selection' below.
legend_title	character string to be used for the legend title.
calm_wind	the direction of wind that is considered calm. Following convention of the National Weather Service, winds with a direction of 0 are considered calm by default.
variable_wind	numeric code for variable winds (if applicable).
n_col	The number of columns of plots (default 1).
...	further arguments passed to theme.

Details

This is intended to be used as a stand-alone function for any wind dataset. A different wind rose is plotted for each level of the faceting variable which is coerced to a factor if necessary. The facets will generally be the station where the data were collected, seasons or dates. Currently only one faceting variable is allowed and is passed to facet_wrap with the formula ~facet.

Note that calm winds are excluded from the wind rose.

Value

a ggplot object.

Theme Selection

For black and white wind roses that may be preferred if plots are to be used in journal articles for example, recommended ggthemes are 'bw', 'linedraw', 'minimal' or 'classic' and the col_pal should be 'Greys'. Otherwise, any of the sequential RColorBrewer colour palettes are recommended for colour plots.

See Also

theme for more possible arguments to pass to windrose.

Examples

```
# Create some dummy wind data with predominant south to westerly winds, and
# occasional yet higher wind speeds from the NE (not too dissimilar to
# Auckland).

wind_df = data.frame(wind_speeds = c(rweibull(80, 2, 4), rweibull(20, 3, 9)),
                     wind_dirs = c(rnorm(80, 135, 55), rnorm(20, 315, 35)) %% 360,
                     station = rep(rep(c("Station A", "Station B"), 2),
                                   rep(c(40, 10), each = 2)))

# Plot a simple wind rose using all the defaults, ignoring any facet variable
with(wind_df, windrose(wind_speeds, wind_dirs))

# Create custom speed bins, add a legend title, and change to a B&W theme
with(wind_df, windrose(wind_speeds, wind_dirs,
                       speed_cuts = c(3, 6, 9, 12),
                       legend_title = "Wind Speed\n(m/s)",
                       legend.title.align = .5,
                       ggtheme = "bw",
                       col_pal = "Greys"))

# Note that underscore-separated arguments come from the windrose method, and
# period-separated arguments come from ggplot2::theme().

# Include a facet variable with one level
with(wind_df, windrose(wind_speeds, wind_dirs, "Artificial Auckland Wind"))

# Plot a windrose for each level of the facet variable (each station)
with(wind_df, windrose(wind_speeds, wind_dirs, station, n_col = 2))

## Not run:
# Save the plot as a png to the current working directory
library(ggplot2)
ggsave("my_windrose.png")

## End(Not run)
```

[[,dataFrame-method *Subsetting Methods for Clifro Objects*

Description

Operators acting on `cfDataList`, `cfDatatype`, `cfStation`, and `dataFrame` objects.

Usage

```
## S4 method for signature 'dataFrame'
x[[i]]
```

```
## S4 method for signature 'dataFrame,ANY,ANY,ANY'
x[i, j, drop]

## S4 method for signature 'dataFrame'
x$name

## S4 method for signature 'cfStation,ANY,ANY,ANY'
x[i, j, drop = TRUE]

## S4 method for signature 'cfDataList,ANY,ANY,ANY'
x[i, j]

## S4 method for signature 'cfDataList'
x[[i]]

## S4 method for signature 'cfDatatype,ANY,missing,missing'
x[i, j, drop]
```

Arguments

x	a **clifro** object
i	indices specifying elements to extract. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Character vectors will be matched to the names of the object.
j	indices specifying elements to extract. Indices are `numeric` or `character` vectors or empty (missing) or `NULL`. Character vectors will be matched to the names of the object.
drop	if `TRUE`, the result is coerced to the lowest possible dimension. See `drop` for further details.
name	a literal character string. This is partially matched to the names of the object.

Details

These are methods for the generic operators for classes within **clifro**. They are intended to give the user the familiar functionality of subsetting `data.frame` objects.

Index