Package: chirps (via r-universe)

October 29, 2024

Type Package

Title API Client for CHIRPS and CHIRTS

Version 0.1.5

Description API Client for the Climate Hazards Center 'CHIRPS' and 'CHIRTS'. The 'CHIRPS' data is a quasi-global (50°S – 50°N) high-resolution (0.05 arc-degrees) rainfall data set, which incorporates satellite imagery and in-situ station data to create gridded rainfall time series for trend analysis and seasonal drought monitoring. 'CHIRTS' is a quasi-global (60°S – 70°N), high-resolution data set of daily maximum and minimum temperatures. For more details on 'CHIRPS' and 'CHIRTS' data please visit its official home page <https://www.chc.ucsb.edu/data>.

License MIT + file LICENSE

URL https://docs.ropensci.org/chirps/

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

Depends methods, R (>= 3.5.0)

Imports httr, jsonlite, sf, stats, terra (>= 1.2-10)

Suggests climatrends, knitr, markdown, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.2

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

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

RemoteRef master

RemoteSha 951fd2f8d5fa77d0f5c6946075eb908987e35820

Contents

`as.geojson` *Methods to coerce geographical coordinates into a geojson polygon*

Description

Take single points from geographical coordinates and coerce into a geojson of geometry 'Polygon'

Usage

```
as.geojson(lonlat, dist = 0.00001, nQuadSegs = 2L, ...)

## Default S3 method:
as.geojson(lonlat, dist = 0.00001, nQuadSegs = 2L, ...)

## S3 method for class 'sf'
as.geojson(lonlat, dist = 0.00001, nQuadSegs = 2L, ...)
```

Arguments

`lonlat`	a `data.frame` or matrix with geographical coordinates 'lonlat', in that order, or an object of class `sf` with geometry type 'POINT' or 'POLYGON'
`dist`	numeric, buffer distance for all `lonlat`
`nQuadSegs`	integer, number of segments per quadrant
`...`	further arguments passed to `sf` methods

Value

An object of class 'geosjon' for each row in `lonlat`

Examples

```
# Default S3 Method
# random geographic points within bbox(10, 12, 45, 47)
library("sf")

set.seed(123)
lonlat <- data.frame(lon = runif(1, 10, 12),
                     lat = runif(1, 45, 47))

gjson <- as.geojson(lonlat)

#################

# S3 Method for objects of class 'sf'
# random geographic points within bbox(10, 12, 45, 47)
library("sf")

set.seed(123)
lonlat <- data.frame(lon = runif(5, 10, 12),
                     lat = runif(5, 45, 47))

lonlat <- st_as_sf(lonlat, coords = c("lon","lat"))

gjson <- as.geojson(lonlat)
```

`chirps` *API Client for CHIRPS and CHIRTS*

Description

API Client for the Climate Hazards Center 'CHIRPS' and 'CHIRTS'. The 'CHIRPS' data is a quasi-global (50°S – 50°N) high-resolution (0.05 arc-degrees) rainfall data set, which incorporates satellite imagery and in-situ station data to create gridded rainfall time series for trend analysis and seasonal drought monitoring. 'CHIRTS' is a quasi-global (60°S – 70°N), high-resolution data set of daily maximum and minimum temperatures. For more details on 'CHIRPS' and 'CHIRTS' data please visit its official home page `https://www.chc.ucsb.edu/data`.

Note

While **chirps** does not redistribute the data or provide it in any way, we encourage users to cite Funk et al. (2015) when using CHIRPS and Funk et al. (2019) when using CHIRTS.

Funk et al. (2015). Scientific Data, 2, 150066. doi:10.1038/sdata.2015.66

Funk et al. (2019). Journal of Climate, 32(17), 5639–5658. doi:10.1175/JCLID180698.1

Author(s)

Kauê de Sousa and Adam H. Sparks and Aniruddha Ghosh

See Also

Useful links:

"JOSS paper:" doi:10.21105/joss.02419

"Development repository:" https://github.com/ropensci/chirps

"Static documentation:" https://docs.ropensci.org/chirps/

"Report bugs:" https://github.com/ropensci/chirps/issues

"CHC website:" https://www.chc.ucsb.edu

`get_chirps` *Get CHIRPS precipitation data*

Description

Get daily precipitation data from the "Climate Hazards Group". Two server sources are available. The first, "CHC" (default) is recommended for multiple data-points, while "ClimateSERV" is recommended when few data-points are required (~ 50).

Usage

```
get_chirps(object, dates, server, ...)

## Default S3 method:
get_chirps(object, dates, server, as.matrix = FALSE, ...)

## S3 method for class 'SpatVector'
get_chirps(object, dates, server = "CHC", as.raster = TRUE, ...)

## S3 method for class 'SpatRaster'
get_chirps(
  object,
  dates,
  server = "CHC",
  as.matrix = TRUE,
  as.raster = FALSE,
  ...
)

## S3 method for class 'SpatExtent'
get_chirps(object, dates, server = "CHC", as.raster = TRUE, ...)

## S3 method for class 'sf'
get_chirps(object, dates, server, as.sf = FALSE, ...)

## S3 method for class 'geojson'
```

```
get_chirps(object, dates, server, as.geojson = FALSE, ...)

## S3 method for class 'SpatExtent'
get_chirps(object, dates, server = "CHC", as.raster = TRUE, ...)
```

Arguments

object	input, an object of class data.frame (or any other object that can be coerced to data.frame), SpatVector, SpatRaster, sf or geojson
dates	a character of start and end dates in that order in the format "YYYY-MM-DD"
server	a character that represents the server source "CHC" or "ClimateSERV"
...	additional arguments passed to terra or sf methods See details
as.matrix	logical, returns an object of class matrix
as.raster	logical, returns an object of class SpatRaster
as.sf	logical, returns an object of class sf
as.geojson	logical, returns an object of class geojson

Details

Data description at https://data.chc.ucsb.edu/products/CHIRPS-2.0/README-CHIRPS.txt

Additional arguments when using server = "CHC"

resolution: numeric, resolution of CHIRPS tiles either 0.05 (default) or 0.25 degrees

Additional arguments when using server = "ClimateSERV"

dist: numeric, buffer distance for each object coordinate

nQuadSegs: integer, number of segments per buffer quadrant

operation: supported operations for ClimateSERV are:

operation		value
max	=	0
min	=	1
median	=	2
sum	=	4
average	=	5 (*default value*)

Value

A matrix, raster or a data frame of CHIRPS data:

id the index for the rows in object

dates the dates from which CHIRPS was requested

lon the longitude as provided in object

lat the latitude as provided in object

chirps the CHIRPS value in mm

Note

`get_chirps()` may return some warning messages given by `sf`, please look **sf** documentation for possible issues.

References

Funk C. et al. (2015). Scientific Data, 2, 150066.
doi:10.1038/sdata.2015.66

Examples

```
library("chirps")
library("terra")

# Case 1: return as a data.frame
dates <- c("2017-12-15","2017-12-31")
lonlat <- data.frame(lon = c(-55.0281,-54.9857), lat = c(-2.8094, -2.8756))

r1 <- get_chirps(lonlat, dates, server = "CHC")

# Case 2: return a matrix
r2 <- get_chirps(lonlat, dates, server = "CHC", as.matrix = TRUE)

# Case 3: input SpatVector and return raster
f <- system.file("ex/lux.shp", package = "terra")
v <- vect(f)
r3 <- get_chirps(v, dates, server = "CHC", as.raster = TRUE)

# Case 4: input SpatExtent and return a raster within the extent
area <- ext(c(-66, -64, -6, -4))

dates <- c("2017-12-15", "2017-12-31")

r4 <- get_chirps(area, dates, server = "CHC")

# Case 5: using the server "ClimateSERV"
r5 <- get_chirps(lonlat, dates, server = "ClimateSERV")

# Case 6: from "ClimateSERV" and return as a matrix
r6 <- get_chirps(lonlat, dates, server = "ClimateSERV", as.matrix = TRUE)
```

`get_chirts` *Get CHIRTS temperature data*

Description

Get daily maximum and minimum temperature data from the "Climate Hazards Group". CHIRTS-daily is a global 2-m temperature product that combines the monthly CHIRTSmax data set with the ERA5 reanalysis to produce routinely updated data to support the monitoring of temperature extreme. Data is currently available from 1983 to 2016. Soon available to near-present.

Usage

```
get_chirts(object, dates, var, ...)

## Default S3 method:
get_chirts(object, dates, var, as.matrix = FALSE, ...)

## S3 method for class 'SpatVector'
get_chirts(object, dates, var, as.raster = TRUE, ...)

## S3 method for class 'SpatRaster'
get_chirts(object, dates, var, as.raster = TRUE, ...)

## S3 method for class 'SpatExtent'
get_chirts(object, dates, var, as.raster = TRUE, ...)
```

Arguments

object	an object of class data.frame (or any other object that can be coerced to a data.frame), SpatVector, or SpatRaster
dates	a character of start and end dates in that order in the format "YYYY-MM-DD"
var	character, A valid variable from the options: “Tmax”, “Tmin”, “RHum” and “HeatIndex”
...	further arguments passed to terra
as.matrix	logical, returns an object of class matrix
as.raster	logical, returns an object of class SpatRaster

Details

Variable description from https://data.chc.ucsb.edu/products/CHIRTSdaily/aaa.Readme.txt

Tmax Daily average maximum air temperature at 2 m above ground

Tmin Daily average minimum air temperature at 2 m above ground

RHum Daily average relative humidity

HeatIndex Daily average heat index

Value

A SpatRaster object if as.raster=TRUE, else matrix, list, or data.frame

Additional arguments

interval: supported intervals are "daily", "pentad", "dekad", "monthly", "2-monthly", "3-monthly", and "annual". Currently hard coded to "daily".

Examples

```
library("chirps")
library("terra")

# Case 1: input a data frame return a data frame in the long format
dates <- c("2010-12-15","2010-12-31")
lonlat <- data.frame(lon = c(-55.0281,-54.9857),
                     lat = c(-2.8094, -2.8756))

temp1 <- get_chirts(lonlat, dates, var = "Tmax")

# Case 2: input a data frame return a matrix
temp2 <- get_chirts(lonlat, dates, "Tmax", as.matrix = TRUE)

# Case 3: input a raster and return raster
f <- system.file("ex/lux.shp", package="terra")
v <- vect(f)
temp3 <- get_chirts(v, dates, var = "Tmax", as.raster = TRUE)

# Case 4: input a raster and return raster
temp4 <- get_chirts(v, dates, var = "Tmax", as.matrix = TRUE)
```

---

get_esi *Get evaporative stress index (ESI) data*

Description

Get evaporative stress index (ESI) from SERVIR Global via ClimateSERV API Client. ESI is available every four (or twelve) weeks from 2001 to present. The dataset may contain cloudy data which is returned as NAs. ClimateSERV works with 'geojson' of type 'Polygon'. The input `object` is then transformed into polygons with a small buffer area around the point.

Usage

```
get_esi(object, dates, operation = 5, period = 1, ...)

## Default S3 method:
get_esi(object, dates, operation = 5, period = 1, ...)

## S3 method for class 'sf'
get_esi(object, dates, operation = 5, period = 1, as.sf = FALSE, ...)

## S3 method for class 'geojson'
get_esi(object, dates, operation = 5, period = 1, as.geojson = FALSE, ...)
```

Arguments

object	input, an object of class data.frame (or any other object that can be coerced to data.frame), SpatVector, SpatRaster, sf or geojson
dates	a character of start and end dates in that order in the format "YYYY-MM-DD"
operation	optional, an integer that represents which type of statistical operation to perform on the dataset
period	an integer value for the period of ESI data, four weeks period = 1, twelve weeks = 2
...	additional arguments passed to terra or sf methods See details
as.sf	logical, returns an object of class sf
as.geojson	logical, returns an object of class geojson

Details

operation: supported operations are:

operation		value
max	=	0
min	=	1
median	=	2
sum	=	4
average	=	5 (*default value*)

dist: numeric, buffer distance for each object coordinate

nQuadSegs: integer, number of segments per buffer quadrant

Value

A data frame of ESI data:

id	the index for the rows in object
dates	the dates from which ESI was requested
lon	the longitude as provided in object
lat	the latitude as provided in object
esi	the ESI value

Note

get_esi() may return some warning messages given by sf, please check the **sf** documentation for possible issues.

Examples

```
lonlat <- data.frame(lon = c(-55.0281,-54.9857),
                     lat = c(-2.8094, -2.8756))

dates <- c("2017-12-15","2018-06-20")

# by default the function sets a very small buffer around the points which
# can return NAs due to cloudiness in ESI data

dt <- get_esi(lonlat, dates = dates)

# the argument dist passed through sf increase the buffer area

dt <- get_esi(lonlat, dates = dates, dist = 0.1)
```

`get_imerg` *Get Integrated Multisatellite Retrievals for GPM (IMERG) data*

Description

The IMERG dataset provides near-real time global observations of rainfall at 10km resolution, which can be used to estimate total rainfall accumulation from storm systems and quantify the intensity of rainfall and flood impacts from tropical cyclones and other storm systems. IMERG is a daily precipitation dataset available from 2015 to present within the latitudes 70 and -70 degrees.

Usage

```
get_imerg(object, dates, operation = 5, ...)

## Default S3 method:
get_imerg(object, dates, operation = 5, ...)

## S3 method for class 'sf'
get_imerg(object, dates, operation = 5, as.sf = FALSE, ...)

## S3 method for class 'geojson'
get_imerg(object, dates, operation = 5, as.geojson = FALSE, ...)
```

Arguments

`object`	input, an object of class `data.frame` (or any other object that can be coerced to `data.frame`), `SpatVector`, `SpatRaster`, `sf` or `geojson`
`dates`	a character of start and end dates in that order in the format "YYYY-MM-DD"
`operation`	optional, an integer that represents which type of statistical operation to perform on the dataset

...	additional arguments passed to terra or sf methods See details
as.sf	logical, returns an object of class sf
as.geojson	logical, returns an object of class geojson

Details

operation: supported operations are:

operation		value
max	=	0
min	=	1
median	=	2
sum	=	4
average	=	5 (*default value*)

dist: numeric, buffer distance for each object coordinate

nQuadSegs: integer, number of segments per buffer quadrant

Value

A data frame of IMERG data:

id	the index for the rows in object
dates	the dates from which imerg was requested
lon	the longitude as provided in object
lat	the latitude as provided in object
imerg	the IMERG value

Examples

```
lonlat <- data.frame(lon = c(-55.0281,-54.9857),
                     lat = c(-2.8094, -2.8756))

dates <- c("2017-12-15", "2017-12-31")

dt <- get_imerg(lonlat, dates)

dt
```

precip_indices *Compute precipitation indices over a time series*

Description

Compute precipitation indices over a time series

Usage

```
precip_indices(object, timeseries = FALSE, intervals = NULL)
```

Arguments

object	an object of class chirps as provided by get_chirps
timeseries	logical, FALSE for a single point time series observation or TRUE for a time series based on *intervals*
intervals	integer no lower than 5, for the days intervals when *timeseries* = TRUE

Value

A data frame with precipitation indices:

MLDS	maximum length of consecutive dry day, rain < 1 mm (days)
MLWS	maximum length of consecutive wet days, rain >= 1 mm (days)
R10mm	number of heavy precipitation days 10 >= rain < 20 mm (days)
R20mm	number of very heavy precipitation days rain >= 20 (days)
Rx1day	maximum 1-day precipitation (mm)
Rx5day	maximum 5-day precipitation (mm)
R95p	total precipitation when rain > 95th percentile (mm)
R99p	total precipitation when rain > 99th percentile (mm)
Rtotal	total precipitation (mm) in wet days, rain >= 1 (mm)
SDII	simple daily intensity index, total precipitation divided by the number of wet days (mm/days)

References

Aguilar E., et al. (2005). Journal of Geophysical Research, 110(D23), D23107.

Kehel Z., et al. (2016). In: Applied Mathematics and Omics to Assess Crop Genetic Resources for Climate Change Adaptive Traits (eds Bari A., Damania A. B., Mackay M., Dayanandan S.), pp. 151–174. CRC Press.

Examples

```
lonlat <- data.frame(lon = c(-55.0281,-54.9857),
                     lat = c(-2.8094, -2.8756))

dates <- c("2017-12-15", "2017-12-31")

dt <- get_chirps(lonlat, dates, server = "ClimateSERV")

# take the indices for the entire period
precip_indices(dt, timeseries = FALSE)

# take the indices for periods of 7 days
precip_indices(dt, timeseries = TRUE, intervals = 7)
```

---

tapajos *Tapajos National Forest*

Description

Geometries for the Tapajos National Forest, a protected area in the Brazilian Amazon

Usage

```
tapajos
```

Format

An object of class 'sfc_POLYGON' within the bounding box xmin: -55.41127 ymin: -4.114584 xmax: -54.7973 ymax: -2.751706

Source

The data was provided by the Chico Mendes Institute via https://www.protectedplanet.net/en

Index