Package 'countyweather'

October 26, 2016

Type Package

Title Compiles Meterological Data for U.S. Counties

Version 0.1.0

Date 2016-10-25

Description Interacts with NOAA data sources (including the NCDC API at <http://www.ncdc.noaa.gov/cdo-web/webservices/v2> and ISD data) using functions from the 'rnoaa' package to obtain and compile weather time series for U.S. counties. This work was supported in part by grants from the National Institute of Environmental Health Sciences (R00ES022631) and the Colorado State University Water Center.

License GPL-2

LazyData TRUE

Imports dplyr (>= 0.4.3), ggmap (>= 2.6.1), ggplot2 (>= 2.1.0), geosphere (>= 1.5.1), lubridate (>= 1.5.6), purrr (>= 0.2.1), raster (>= 2.5.8), rnoaa (>= 0.6.5), sp (>= 1.2.3), stringi (>= 1.1.1), tibble(>= 1.2), tidyr (>= 0.3.1), tigris (>= 0.3.3)

RoxygenNote 5.0.1

Depends R (>= 3.3.0)

Suggests countytimezones (>= 0.1.0), knitr (>= 1.14.0), pander(>= 0.6.0), rmarkdown (>= 1.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Rachel Severson [aut, cre], Brooke Anderson [aut]

Maintainer Rachel Severson <rachel.severson@colostate.edu>

Repository CRAN

Date/Publication 2016-10-26 00:17:31

R topics documented:

`ave_daily` *Average daily weather data across multiple stations.*

Description

Returns a dataframe with daily weather averaged across stations, as well as columns showing the number of stations contributing to the average for each variable and each day.

Usage

```
ave_daily(weather_data)
```

Arguments

`weather_data`	A dataframe with daily weather observations. This dataframe is returned from the `rnoaa` function `meteo_pull_monitors`.

ave_hourly — *Average hourly weather data across multiple stations.*

Description

Returns a dataframe with hourly weather averaged across stations, as well as columns showing the number of stations contributing to average for each variable and each hour.

Usage

```
ave_hourly(hourly_data)
```

Arguments

hourly_data	A dataframe with hourly weather observations. This dataframe is returned from the df element of the function isd_monitors_data.

county_centers — *County latitude and longitude designations.*

Description

A dataframe containing county-specific data: state, FIPS code, name, latitude and longitude of geographic center, and region code. It includes each U.S. county as of the 2010 census. This dataset was put together using a dataframe from the U.S. Census Bureau, which was pulled from the website listed in "Source." (Note: The names—in county, state format—for each county were pulled from the 2010 U.S. Census file found here: http://www2.census.gov/geo/docs/reference/cenpop2010/county/CenPop2010_Mean_CO.txt.)

Usage

```
county_centers
```

Format

A dataframe with 3,143 rows and 5 variables:

state A character vector giving the two-letter abbreviation for the state of each county

fips A numeric vector giving the county's five-digit Federal Information Processing Standard (FIPS) code

name A character vector giving the name and state for each county

latitude A numeric vector giving the latitude of the geographic center of each county

longitude A numeric vector giving the longitude of the geographic center of each county

region A numeric vector giving the four-digit or five-digit Federal Information Processing Standard (FIPS) code (values in this column are identical to those in the "fips" column, but do not include leading zeros)

Source

https://www.census.gov/geo/maps-data/data/gazetteer2010.html

`county_radius` *County land area data.*

Description

A dataframe containing the FIPS code and estimated radius (in km) for each U.S. county. This dataset was put together using a dataset from the U.S. Census American FactFinder data dissemination tool. This dataset was downloaded from the geographic identifiers 'G001' option for the 2010 Summary File1 (SF1). The file was found by following the instructions recommended by the U.S. census here: https://ask.census.gov/faq.php?id=5000&faqId=7825. The website listed in "Source" gives more information about this dataset.

Usage

```
county_radius
```

Format

A dataframe with 3,143 rows and 2 variables:

fips A character vector giving the county's five-digit Federal Information Processing Standard (FIPS) code

county_radius A numeric vector giving an estimate for each county's radius from its center, in km. This value was calculated by dividing the U.S. Census Land Area estimates (which are in square meters) by 1,000,000, and then taking the square root of each area and dividing by pi. Each county was estimated to be roughly circular for the purposes of estimating county radii.

Source

http://www.census.gov/prod/cen2010/doc/sf1.pdf

`daily_df` *Return average daily weather data for a particular county.*

Description

Returns a list with data on weather and stations for a selected county. This function serves as a wrapper to several functions from the `rnoaa` package, which pull weather data from all relevant stations in a county. This function filters and averages data returned by `rnoaa` functions across all weather stations in a county based on user-specified coverage specifications.

Usage

```
daily_df(stations, coverage = NULL, var = "all", date_min = NULL,
  date_max = NULL, average_data = TRUE)
```

Arguments

stations	A dataframe containing station metadata, returned from the function daily_stations.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., coverage = 0).)
var	A character vector specifying desired weather variables. For example, var = c("tmin", "tmax", "prcp for maximum temperature, minimum temperature, and precipitation. The default is "all", which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors. If you choose FALSE, the function will return a dataframe with separate entries for each monitor, while TRUE (the default) outputs a single estimate for each day in the dataset, giving the average value of the weather metric across all available monitors in the county that day.

Value

A list with two elements. daily_data is a dataframe of daily weather data averaged across multiple monitors and includes columns ("var"_reporting) for each weather variable showing the number of stations contributing to the average for that variable on that day. The element station_df is a dataframe of station metadata for each station contributing weather data. A weather station will have one row per weather variable to which it contributes data. In addition to information such as station id, name, latitude, and longitude, the station_df dataframe includes statistical information about weather values contributed by each station for each weather variable. These statistics include calc_coverage (the percent of non-missing values for each station-weather variable combination for the specified date range), standard_dev (standard deviation), max, and min, (giving the minimum and maximum values), and range, giving the range of values in each station-weather variable

combination. The element `radius` is the calculated radius within which stations were pulled from the county's center. Elements `lat_center` and `lon_center` are the latitude and longitude of the county's center.

Note

Because this function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (http://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA). See the package vignette for more details.

Examples

```
## Not run:
stations <- daily_stations(fips = "12086", date_min = "2010-01-01",
                           date_max = "2010-02-01")
fips_list <- daily_df(stations = stations, coverage = 0.90,
                 var = c("tmax", "tmin", "prcp"),
                 date_min = "2010-01-01", date_max = "2010-02-01")
averaged_data <- fips_list$daily_data
head(averaged_data)
station_info <- fips_list$station_df
head(station_info)

## End(Not run)
```

---

daily_fips *Pull average daily weather data by U.S. county.*

Description

Given a particular county FIPS code, this function returns data and meta-data for weather data, either for all available dates or for dates within a requested date range.

Usage

```
daily_fips(fips, coverage = NULL, date_min = NULL, date_max = NULL,
  var = "all", average_data = TRUE, station_label = FALSE,
  verbose = TRUE)
```

Arguments

`fips`	A string with the five-digit U.S. FIPS code of a county in numeric, character, or factor format.

coverage — A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., `coverage = 0`).)

date_min — A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.

date_max — A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.

var — A character vector specifying desired weather variables. For example, `var = c("tmin", "tmax", "prcp` for maximum temperature, minimum temperature, and precipitation. The default is `"all"`, which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.

average_data — TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors. If you choose FALSE, the function will return a dataframe with separate entries for each monitor, while TRUE (the default) outputs a single estimate for each day in the dataset, giving the average value of the weather metric across all available monitors in the county that day.

station_label — TRUE / FALSE to indicate if you want your plot of weather station locations to include labels with station ids.

verbose — TRUE / FALSE to indicate if you want the function to print out the name of the county it's processing.

Value

A list with three elements. The first element (`daily_data`) is a dataframe of daily weather data averaged across multiple stations, as well as columns (`"var"_reporting`) for each weather variable showing the number of stations contributing to the average for that variable on that day. The second element (`station_metadata`) is a dataframe of station metadata for stations included in the `daily_data` dataframe, as well as statistical information about these values. Columns include `id`, `name`, `var`, `latitude`, `longitude`, `calc_coverage`, `standard_dev`, `min`, `max`, and `range`. The third element (`station_map`) is a plot showing locations of all weather stations for a particular county satisfying the conditions present in `daily_fips`'s arguments (`coverage`, `date_min`, `date_max`, and/or `var`).

Note

Because this function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request

page (http://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA). See the package vignette for more details.

Examples

```
## Not run:
denver_ex <- daily_fips("08031", coverage = 0.90, date_min = "2010-01-01",
                 date_max = "2010-02-01", var = "prcp")

head(denver_ex$daily_data)
denver_ex$station_map

mobile_ex <- daily_fips("01097", date_min = "1997-07-13",
                        date_max = "1997-07-25", var = "prcp",
                        average_data = FALSE)
library(ggplot2)
ggplot(mobile_ex$daily_data, aes(x = date, y = prcp, color = id)) +
   geom_line()

## End(Not run)
```

`daily_stationmap` *Plot daily weather stations for a particular county.*

Description

Produces a map with points indicating stations that contribute to the weather data in the `daily_data` data frame output by `daily_fips`.

Usage

```
daily_stationmap(fips, daily_data, point_color = "firebrick",
  point_size = 2, station_label = FALSE)
```

Arguments

`fips`	A five-digit FIPS county code.
`daily_data`	A list returned from the function `daily_df` (see helpfile for `daily_df`).
`point_color`	Character string with color for points mapping the locations of weather stations (passes to `ggplot`).
`point_size`	Character string with size for for points mapping the locations of weather stations (passes to `ggplot`).
`station_label`	TRUE / FALSE Whether to include labels for each weather station.

Value

A ggplot object mapping all weather stations for a particular county satisfying the conditions present in daily_df's arguments (date range, coverage, and/or weather variables). 2011 U.S. Census cartographic boundary shapefiles are used to provide county outlines.

Examples

```
## Not run:
miami_stations <- daily_stations(fips = "12086", date_min = "1992-08-01",
                          date_max = "1992-08-31")
daily_data <- daily_df(stations = miami_stations, coverage = 0.90,
                       var = c("tmax", "tmin", "prcp"),
                       date_min = "1992-08-01", date_max = "1992-08-31")
daily_stationmap(fips = "12086", daily_data = daily_data)

## End(Not run)
```

---

daily_stations *NOAA NCDC station IDs per county.*

Description

Returns a dataframe with NOAA NCDC station IDs for a single U.S. county. This function has options to filter stations based on maximum and minimum dates, as well as percent data coverage.

Usage

```
daily_stations(fips, date_min = NULL, date_max = NULL)
```

Arguments

fips	A string with the five-digit U.S. FIPS code of a county in numeric, character, or factor format.
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.

Value

A dataframe with NOAA NCDC station IDs for a single U.S. county.

Note

Because this function uses the NOAA API to identify the weather monitors within a U.S. county, you will need to get an access token from NOAA to use this function. Visit NOAA's token request page (http://www.ncdc.noaa.gov/cdo-web/token) to request a token by email. You then need to set that API code in your R session (e.g., using `options(noaakey = "your key")`, replacing "your key" with the API key you've requested from NOAA). See the package vignette for more details.

Examples

```
## Not run:
stations_36005 <- daily_stations("36005")
stations_36005

miami_stations <- daily_stations("12086", date_min = "1999-01-01",
                                 date_max = "2012-12-31")
miami_stations

## End(Not run)
```

`filter_coverage` *Filter stations based on "coverage" requirements.*

Description

Filters available weather stations based on a specified required minimum coverage (i.e., percent non-missing daily observations). Weather stations with non-missing data for fewer days than specified by `coverage` will be excluded from the county average.

Usage

```
filter_coverage(coverage_df, coverage = 0)
```

Arguments

`coverage_df`	A dataframe as returned by the `meteo_coverage` function in the `rnoaa` package
`coverage`	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors).

Value

A dataframe with stations that meet the specified coverage requirements for weather variables included in the `coverage_df` dataframe passed to the function.

filter_hourly *Filter NOAA ISD stations based on "coverage" requirements, and calculate coverage and statistical information for each station-variable combination.*

Description

Filters available weather stations based on a specified minimum coverage (i.e., percent non-missing hourly observations). Weather stations with non-missing data for fewer days than specified by `coverage` will be excluded from the county average.

Usage

```
filter_hourly(fips, hourly_data, coverage = NULL)
```

Arguments

`fips`	A character string giving the five-digit U.S. FIPS county code of the county for which the user wants to pull weather data.
`hourly_data`	A dataframe as returned by the `df` element from an `isd_monitors_data` call.
`coverage`	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for each weather variable (i.e., what percent of each weather variable must be non-missing to include the data from a station when calculating hourly values averaged across stations).

Value

A list with two elements: `df` and `stations`. `df` is a dataframe of hourly weather data filtered based on the specfified coverage, as well as columns (`"var"_reporting`) for each weather variable showing the number of stations contributing to the average for that variable for each hour. The second element, `stations`, is a dataframe giving statistical information for stations that meet the specified coverage requirements. The column `station` gives the station id (USAF and WBAN identification numbers pasted together, separated by "-"). Note: One of these identification ids is sometimes missing. For example, a value in `station` might be `722029-NA`. The column `var` gives the weather variable associated with the row of statistical values for each station and variable combination. `calc_coverage` gives the percentage coverage for each station-weather variable combination. These values will all be greater than or equal to the specified `coverage` value. `standard_dev` gives the standard deviation of values for each station-weather variable combination. `max` and `min` give the minimum and maximum values, and `range` gives the range of values in each station-weather variable combination. These last four statistical calculations (`standard_dev`, `max`, `min`, and `range`) are only included for the seven core hourly weather variables, which include `"wind_direction"`, `"wind_speed"`, `"ceiling_height"`, `"visibility_distance"`, `"temperature"`, `"temperature_dewpoint"`, and `"air_pressure"`. (The values of these columns are set to `NA` for other variables, such as quality flag data.)

hourly_df *Return average hourly weather data for a particular county.*

Description

Returns a dataframe of average daily weather values for a particular county, year, weather variables, and/or specified coverage.

Usage

```
hourly_df(fips, year, var = "all", average_data = TRUE, coverage = NULL)
```

Arguments

fips	A character string of the five-digit U.S. FIPS code of a U.S. county.
year	A four-digit number or vector of numbers indicating the year or years for which you want to pull hourly data. Values for year can be in the range from 1901 to the current year.
var	A character vector specifying desired weather variables. For example, var = c("wind_speed", "temper pulls data on hourly wind speed and temperature. The core weather variables available include "wind_direction", "wind_speed", "ceiling_height", "visibility_distance", "temperature", "temperature_dewpoint", "air_pressure". Alternatively, you can specify var = "all" to include additional flag and quality codes.
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors).

Details

This function serves as a wrapper to several functions from the rnoaa package, which provides weather data from all relevant stations in a county. This function filters and averages across NOAA ISD/ISH stations based on user-specified coverage specifications.

Value

A list with five elements. The first element, hourly_data, is a dataframe of hourly weather data averaged across multiple stations, as well as columns ("var"_reporting) for each weather variable showing the number of stations contributing to the average for that variable for each hour. station_df is a dataframe of station metadata for each station contributing weather data. A weather station will have one row per weather variable to which it contributes data. In addition to information such as USAF and WBAN ids and station names, this dataframe includes statistical information about weather values contributed by each station for each weather variable. These statistics include calculated coverage (calc_coverage), which is the percent of non-missing values for each station

and variable for the specified date range, standard_dev (standard deviation), max, min, and range values for each station-weather variable combination. The element radius is the calculated radius within which stations were pulled from the county's center. Elements lat_center and lon_center are the latitude and longitude of the county's geographic center.

Note

Observation times are based on Coordinated Universal Time Code (UTC).

References

For more information on this dataset and available weather and flag/quality variables, see ftp://ftp.ncdc.noaa.gov/pub/data/noaa/ish-format-document.pdf.

Examples

```
## Not run:
df <- hourly_df(fips = "12086", year = 1992,
                var = c("wind_speed", "temperature"))
head(df$hourly_data)
head(df$station_df)
df$radius

## End(Not run)
```

---

hourly_fips *Return average hourly weather data and a plot showing the location of weather stations for a particular county.*

Description

Given a particular county FIPS code, this function returns a list with two elements: data, a dataframe of hourly average weather values, and plot, a plot showing the location of weather stations contributing to the average weather in data.

Usage

```
hourly_fips(fips, year, var = "all", coverage = NULL, average_data = TRUE,
  station_label = FALSE, verbose = TRUE)
```

Arguments

fips	A character string of the five-digit U.S. FIPS code of a U.S. county.
year	A four-digit number or vector of numbers indicating the year or years for which you want to pull hourly data. Values for year can be in the range from 1901 to the current year.

var	A character vector specifying desired weather variables. For example, var = c("wind_speed", "temper pulls data on hourly wind speed and temperature. The core weather variables available include "wind_direction", "wind_speed", "ceiling_height", "visibility_distance", "temperature", "temperature_dewpoint", "air_pressure". Alternatively, you can specify var = "all" to include additional flag and quality codes.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors).
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors.
station_label	TRUE / FALSE to indicate if you want your plot of weather station locations to include labels indicating station usaf id numbers.
verbose	TRUE / FALSE to indicate if you want the function to print out the name of the county it's processing.

Value

A list with six elements. The first element (hourly_data) is a dataframe of daily weather data averaged across multiple stations, as well as columns ("var"_reporting) for each weather variable showing the number of stations contributing to the average for that variable for that hour. The second element (station_metadata) is a dataframe of station metadata for stations included in the daily_data dataframe, as well as statistical information about the values contriuted to each weather variable by each station. The third element (station_map) is a plot showing points for all weather stations for a particular county satisfying the conditions present in hourly_fips's arguments (year, coverage, and/or weather variables). radius is the calculated radius within which stations were pulled from the county's center. Elements lat_center and lon_center are the latitude and longitude of the county's center.

Note

Observation times are based on Coordinated Universal Time Code (UTC).

Examples

```
## Not run:

ex <- hourly_fips("12086", coverage = 0.90, year = c(1994, 1995),
                  var = "temperature")

data <- ex$hourly_data
station_data <- ex$station_metadata
station_map <- ex$station_map

## End(Not run)
```

hourly_stationmap *Plot hourly weather stations for a particular county.*

Description

Produces a ggplot object mapping stations that contribute to the weather data returned by hourly_data.

Usage

```
hourly_stationmap(fips, hourly_data, point_color = "firebrick",
  point_size = 2, station_label = FALSE)
```

Arguments

fips	A five-digit FIPS county code.
hourly_data	A list returned from the function hourly_df.
point_color	Character string with color for points mapping the locations of weather stations (passes to ggplot).
point_size	Character string with size for for points mapping the locations of weather stations (passes to ggplot).
station_label	TRUE / FALSE Whether to include labels for each weather station.

Value

A ggplot object mapping all weather stations for a particular county that satisfy the conditions present in hourly_df's arguments (year(s), coverage, and/or weather variables). Because hourly data is pulled by radius from each county's geograph center, this plot includes the calculated radius from which stations are pulled for that county. This radius is calculated for each county using 2010 U.S. Census Land Area data. 2011 U.S. Census cartographic boundary shapefiles are used to proved county outlines, included on this plot as well. Note: Because stations are pulled within a radius from the county's geographic center, depending on the shape of the county, weather stations from outside the county's boundaries may sometimes be providing data for that county and some weather stations within the county may not be included.

Examples

```
## Not run:
hourly_data <- hourly_df(fips = "12086", year = 1992,
                         var = c("wind_speed", "temperature"))
hourly_stationmap("12086", hourly_data)

## End(Not run)
```

`int_surface_data` *Get hourly data for a single monitor.*

Description

Wraps the `isd` function from the `rnoaa` package and provides some additional data cleaning.

Usage

```
int_surface_data(usaf_code, wban_code, year, var = "all")
```

Arguments

`usaf_code`	A character string with a six-digit USAF code for the weather station.
`wban_code`	A character string with a five-digit WBAN code for the weather station.
`year`	A four-digit numeric giving the year for which to pull data.
`var`	A character vector listing the weather variables to pull. In addition quality flag data, choices for main weather variables to pull include `wind_direction`, `wind_speed`, `ceiling_height`, `visibility_distance`, `temperature`, `temperature_dewpoint` and `air_pressure`.

Value

This function returns the same type of dataframe as that returned by the `isd` function from the `rnoaa` package, but with the dataframe limited to the selected weather variables and cleaned a bit more.

References

For more information on this dataset, see ftp://ftp.ncdc.noaa.gov/pub/data/noaa/ish-format-document.pdf.

Examples

```
## Not run:
ids <- isd_fips_stations(fips = "12086")$stations
kendall_station <- int_surface_data(usaf_code = ids$usaf[11],
                                    wban_code = ids$wban[11],
                                    year = 1992,
                                    var = c("wind_speed", "temperature"))
head(kendall_station)

## End(Not run)
```

isd_fips_stations *Get station list for a particular U.S. county.*

Description

A wrapper to the `isd_stations_search` function in the `rnoaa` package, allowing you to search by FIPS code rather than having to know the latitude and longitude of the center of each county. The `isd_stations_search` function requires a radius within which to search for stations. This radius is estimated from 2010 U.S. Census Land Area data.

Usage

```
isd_fips_stations(fips, verbose = FALSE)
```

Arguments

`fips`	A five-digit FIPS county code.
`verbose`	TRUE / FALSE to indicate if you want the function to print the county or vector of counties it's saving files for as the function runs.

Value

A list with four elements. The first element, `stations`, is a dataframe of monitors within a calculated radius of the geographic center of the county specified by the FIPS code. This will have the same dataframe format as the output from the `isd_stations_search` function in the `rnoaa` package. The second element, `radius`, gives the radius (in km) within which stations were pulled from the county's geographic center. Elements `lat_center` and `lon_center` are the latitude and longitude of the county's population-weighted center.

Examples

```
## Not run:
fips_list <- isd_fips_stations(fips = "12086")
ids <- fips_list$stations
head(ids)

## End(Not run)
```

isd_monitors_data *Pull hourly data for multiple monitors.*

Description

Pull all available data for all weather monitors within a calculated radius of the geographic center of a U.S. county, based on the county's FIPS code. The radius for each county is calculated using 2010 U.S. Census Land Area data.

Usage

```
isd_monitors_data(fips, year, var = "all")
```

Arguments

fips	A five-digit FIPS county code.
year	A four-digit numeric giving the year for which to pull data.
var	A character vector listing the weather variables to pull. The main available weather variables are wind_direction, wind_speed, ceiling_height, visibility_distance, temperature, temperature_dewpoint and air_pressure.

Value

A list with five elements. ids is a dataframe of station metadata for all avaiable stations in the given fips code. df is a data frame with hourly weather data for the given variable(s) and date range. radius is the calculated radius within which stations were pulled from the county's geographic center. Elements lat_center and lon_center are the latitude and longitude of the county's geographic center.

Examples

```
## Not run:
fips_list <- isd_monitors_data(fips = "12086", year = 1992,
                               var = c("wind_speed", "temperature"))
stationdata <- fips_list$df
ggplot(stationdata, aes(x = date_time, y = wind_speed)) +
   geom_point(alpha = 0.5, size = 0.2) +
   facet_wrap(~ usaf_station, ncol = 1)

## End(Not run)
```

---

plot_daily_timeseries *Write plot files for daily weather timeseries dataframes.*

Description

Writes a directory with plots for every weather data time series file in the specified directory (as produced by the daily_timeseries function and saved in the "data" subdirectory of the directory given in that function's arguments) for a particular weather variable.

Usage

```
plot_daily_timeseries(var, date_min, date_max, data_directory, plot_directory,
  data_type = "rds")
```

Arguments

var	A character string specifying which weather variable for which you would like to produce plots (the variable must be present in the timeseries dataframe).
date_min	A character string giving the earliest date present in the timeseries dataframe in "yyyy-mm-dd" format.
date_max	A character string giving the latest date present in the timeseries dataframe in "yyyy-mm-dd" format.
data_directory	The absolute or relative pathname for the directory where your daily timeseries dataframes (produced by daily_timeseries) are saved.
plot_directory	The absolute or relative pathname for the directory where you would like the plots to be saved.
data_type	A character string indicating the type of timeseries files you would like to produce plots for (either "rds" or "csv"). This option defaults to .rds files.

Value

Writes out a directory with plots of timeseries data for a given weather variable for each file present in the directory specified.

Examples

```
## Not run:
write_daily_timeseries(fips = c("37055", "15005"), coverage = 0.90,
                       date_min = "1995-01-01", date_max = "1995-01-31",
                       var = c("tmax", "tmin", "prcp"),
                       out_directory = "~/timeseries")
plot_daily_timeseries(var = "prcp", date_min = "1995-01-01",
                      date_max = "1995-01-31",
                      data_directory = "~/timeseries/data",
                      plot_directory = "~/timeseries/plots_prcp")

## End(Not run)
```

---

plot_hourly_timeseries

Write plot files for hourly weather time series dataframes.

Description

Writes a directory with plots for every weather data timeseries file present in the specified directory (as produced by the write_hourly_timeseries function) for a particular weather variable. These plots are meant to aid in initial exploratory analysis.

Usage

```
plot_hourly_timeseries(var, year, data_directory, plot_directory,
  data_type = "rds")
```

Arguments

var	A character string specifying which weather variable for which you would like to produce plots (the variable must be present in the timeseries dataframe).
year	A year or vector of years giving the year(s) present in the time series dataframe.
data_directory	The absolute or relative pathname for the directory where your daily timeseries dataframes (produced by daily_timeseries) are saved.
plot_directory	The absolute or relative pathname for the directory where you would like the plots to be saved.
data_type	A character string indicating the type of timeseries files you would like to produce plots for (either "rds" or "csv"). This option defaults to .rds files.

Value

Writes out a directory with plots of time series data for a given weather variable for each file present in the directory specified.

Examples

```
## Not run:
write_hourly_timeseries(fips = c("08031", "12086"), year = c(1994, 1995),
                        coverage = 0.90, var = c("wind_speed", "temperature"),
                        out_directory = "~/timeseries_hourly")
plot_hourly_timeseries(var = "wind_speed", year = c(1994, 1995),
                       data_directory = "~/timeseries_hourly/data",
                       plot_directory = "~/timeseries_hourly/plots_wind_speed")
plot_hourly_timeseries(var = "temperature", year = c(1994, 1995),
                       data_directory = "~/timeseries_hourly/data",
                       plot_directory = "~/timeseries_hourly/plots_temperature")

## End(Not run)
```

---

write_daily_timeseries

Write daily weather timeseries files for U.S. counties.

Description

Given a vector of U.S. county FIPS codes, this function saves each element of the lists created from the function daily_fips to a separate folder within a given directory. This function therefore allows you to pull and save weather data time series for multiple counties at once. The dataframe daily_data is saved to a a subdirectory of the given directory called "data." This timeseries dataframe gives the values for specified weather variables and the number of weather stations contributing to the average value for each day within the specified date range. The element station_metadata, which gives information about stations contributing to the time series, as well as statistical information about the values contributed by these stations, is saved in a subdirectory called "metadata." The element station_map, which is a map of contributing station locations, is saved in a subdirectory called "maps."

Usage

```
write_daily_timeseries(fips, coverage = NULL, date_min = NULL,
  date_max = NULL, var = "all", out_directory, data_type = "rds",
  metadata_type = "rds", average_data = TRUE, station_label = FALSE,
  keep_map = TRUE, verbose = TRUE)
```

Arguments

fips	A string with the five-digit U.S. FIPS code of a county in numeric, character, or factor format.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors. The default is to include all monitors with any available data (i.e., `coverage = 0`).)
date_min	A string with the desired starting date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates including and after the specified date.
date_max	A string with the desired ending date in character, ISO format ("yyyy-mm-dd"). The dataframe returned will include only stations that have data for dates up to and including the specified date.
var	A character vector specifying desired weather variables. For example, `var = c("tmin", "tmax", "prcp` for maximum temperature, minimum temperature, and precipitation. The default is `"all"`, which includes all available weather variables at any weather station in the county. For a full list of all possible variable names, see NOAA's README file for the Daily Global Historical Climatology Network (GHCN-Daily) at http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt. Many of the weather variables are available for some, but not all, monitors, so your output from this function may not include all the variables specified using this argument. If you specify a variable here but it is not included in the output dataset, it means that it was not available in the time range for any monitor in the county.
out_directory	The absolute or relative pathname for the directory where you would like the three subdirectories ("data", "metadata", and "plots") to be created.
data_type	A character string indicating that you would like either .rds files (data_type = "rds") or .csv files (data_type = "csv") for the timeseries output. This option defaults to .rds files.
metadata_type	A character string indicating that you would like either .rds files (metadata_type = "rds") or .csv files (metadata_type = "csv") for the station metadata output. This option defaults to .rds files.
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors. If you choose FALSE, the function will return a dataframe with separate entries for each monitor, while TRUE (the default) outputs a single estimate for each day in the dataset, giving the average value of the weather metric across all available monitors in the county that day.
station_label	TRUE / FALSE to indicate whether to include station labels in the station map.

keep_map — TRUE / FALSE indicating if a map of the stations should be included. The map can substantially increase the size of the files, so if file size is a concern, you should consider setting this option to FALSE. If FALSE, the "maps" subdirectory will not be created.

verbose — TRUE / FALSE to indicate if you want the function to print the county or vector of counties it's saving files for as the function runs.

Value

Writes out three subdirectories of a given directory with daily weather files saved in "data", station metadata saved in "metadata", and a map of weather station locations saved in "maps" for each FIPS code specified provided there is available data for that county. The user can specify either .rds or .csv format for the data and metadata files, using the arguments `data_type` and `metadata_type`, respectively. Maps are saved as .png files.

Note

If the function is unable to pull weather data for a particular county given the specified percent coverage, date range, and/or weather variables, `daily_timeseries` will not produce files for that county.

Examples

```
## Not run:
write_daily_timeseries(fips = c("37055", "15005"), coverage = 0.90,
                       date_min = "1995-01-01", date_max = "1995-01-31",
                       var = c("tmax", "tmin", "prcp"),
                       out_directory = "~/timeseries")

## End(Not run)
```

---

write_hourly_timeseries

Write hourly weather time series files for U.S. counties.

Description

Given a vector of U.S. county FIPS codes, this function saves each element of the lists created from the function `daily_fips` to a separate folder within a given directory. The dataframe `daily_data` is saved to a subdirectory of the given directory called "data." This time series dataframe gives the values for specified weather variables and the number of weather stations contributing to the average for each day within the specified year(s). Metadata about the weather stations and county are saved in a list with four elements in a subdirectory called "metadata." These elements include `station_metadata` (station metadata for stations contributing to the time series dataframe), `radius` (the radius, in km, within which weather stations were pulled from each county's center), `lat_center`, and `lon_center` (the latitude and longitude of the county's geographic center). If the user specifies "csv" output for the `metadata_type` argument, `radius`, `lat_center`, and `lon_center` are added to the `station_metadata` dataframe as three additional columns.

Usage

```
write_hourly_timeseries(fips, year, coverage = NULL, var = "all",
  out_directory, data_type = "rds", metadata_type = "rds",
  average_data = TRUE, station_label = FALSE, keep_map = TRUE,
  verbose = TRUE)
```

Arguments

fips	A character string of the five-digit U.S. FIPS code of a U.S. county.
year	A four-digit number or vector of numbers indicating the year or years for which you want to pull hourly data. Values for year can be in the range from 1901 to the current year.
coverage	A numeric value in the range of 0 to 1 that specifies the desired percentage coverage for the weather variable (i.e., what percent of each weather variable must be non-missing to include data from a monitor when calculating daily values averaged across monitors).
var	A character vector specifying desired weather variables. For example, var = c("wind_speed", "temper pulls data on hourly wind speed and temperature. The core weather variables available include "wind_direction", "wind_speed", "ceiling_height", "visibility_distance", "temperature", "temperature_dewpoint", "air_pressure". Alternatively, you can specify var = "all" to include additional flag and quality codes.
out_directory	The absolute or relative pathname for the directory where you would like the time series files to be saved.
data_type	A character string indicating that you would like either .rds files (data_type = "rds") or .csv files (data_type = "csv") for the time series output. This option defaults to .rds files.
metadata_type	A character string indicating that you would like either .rds files (metadata_type = "rds") or .csv files (metadata_type = "csv") for the station and county metadata output. This option defaults to .rds files, in which case a list of four elements is saved (station_metadata, radius, lat_center, and lon_center). If the user specifies "csv" output, radius, lat_center, and lon_center are added to the station_metadata dataframe as additional columns.
average_data	TRUE / FALSE to indicate if you want the function to average daily weather data across multiple monitors.
station_label	TRUE / FALSE to indicate whether to include station labels in the station map.
keep_map	TRUE / FALSE indicating if a map of the stations should be included. The map can substantially increase the size of the files. If FALSE, the "maps" subdirectory will not be created.
verbose	TRUE / FALSE to indicate if you want the function to print out the county or vector of counties it's saving files for.

Value

Writes out three subdirectories of a given directory, with hourly weather files saved in "data", station and county metadata saved in "metadata", and a map of weather station locations saved in

"maps" for each FIPS code specified. The user can specify either .rds or .csv files for the data and metadatafiles, using the arguments `data_type` and `metadata_type`, respectively. Maps are saved as .png files.

Note

If the function is unable to pull weather data for a particular county given the specified percent coverage, date range, and/or weather variables, `write_hourly_timeseries` will not produce a file for that county.

Examples

```
## Not run:
write_hourly_timeseries(fips = c("08031", "12086"), year = c(1994, 1995),
                        coverage = 0.90, var = c("wind_speed", "temperature"),
                        out_directory = "~/timeseries_hourly")

## End(Not run)
```

Index