Package 'weatherData'

August 29, 2016

Type Package

Title Get Weather Data from the Web

Description Functions that help in fetching weather data from websites. Given a location and a date range, these functions help fetch weather data (temperature, pressure etc.) for any weather related analysis.

URL http://ram-n.github.io/weatherData/

Version 0.4.1

Date 2014-06-23

Author Ram Narasimhan

Suggests testthat

Imports plyr

Maintainer Ram Narasimhan <ramnarasimhan@gmail.com>

LazyData TRUE

License GPL

Depends R (>= 2.10)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-06-24 17:17:14

R topics documented:

`weatherData-package` *Get Weather & Temperature data from the Web*

Description

The package has functions that can fetch weather data.

Details

Package:	weatherData
Type:	Package
Version:	0.4
Date:	2014-04-29
License:	GPL

These functions don't use APIs. They rely on reading URL's instead. Given a valid city and a date (or date range), the functions in weatherData can fetch them as a clean R data frame.

These functions are useful for anyone interested in doing analysis using weather data.

Author(s)

Ram Narasimhan

Maintainer: Ram Narasimhan <ramnarasimhan@gmail.com>

`checkDataAvailability` *Check if WeatherUnderground has Data for given station and date*

Description

Use this function to check if data is available for station and date If the station code or the date is invalid, function will return 0

Usage

```
checkDataAvailability(station_id, check_date, station_type = "airportCode")
```

Arguments

station_id	is a valid airport code or a valid Weather Station ID
check_date	is a a valid string representing a date in the past (string "YYYY-MM-DD")
station_type	is either airportCode or id

Value

1 if the station does have weather records for input date, 0 if no records were found

References

For a list of valid Weather Stations, try this format http://www.wunderground.com/weatherstation/ListStations.asp?selectedCountry=United+States and replace with your country of interest

Examples

```
## Not run:
data_okay <- checkDataAvailability("HECA", "2014-01-01")


## End(Not run)
```

---

checkDataAvailabilityForDateRange

Quick Check to see if WeatherUnderground has Weather Data for given station for a range of dates

---

Description

Before we attempt to fetch the data for a big time interval of dates, this function is useful to see if the data even exists.

Usage

```
checkDataAvailabilityForDateRange(station_id, start_date, end_date,
  station_type = "airportCode")
```

Arguments

station_id	is a valid 3-letter airport code or a valid Weather Station ID
station_type	is either airportCode or id
start_date	is a valid string representing a date in the past (YYYY-MM-DD, all numeric)
end_date	is a a valid string representing a date in the past (YYYY-MM-DD, all numeric) and is greater than start_date

Details

This functions checks for just the first and the last date in the interval, not the days in between

Value

1 if the Station did have weather records, 0 if nothing was found

Examples

```
## Not run:
data_okay <- checkDataAvailabilityForDateRange("BOS",
                                               "2011-01-01",
                                               "2011-03-31")

## End(Not run)
```

`checkSummarizedDataAvailability`

Quick Check to see if WeatherUnderground has Summarized Weather Data for given station for a custom range of dates

---

Description

Before we attempt to fetch the data for a big time interval of dates, this function is useful to see if the data even exists.

Usage

```
checkSummarizedDataAvailability(station_id, start_date, end_date = NULL,
  station_type = "airportCode")
```

Arguments

`station_id`	is a valid 3-letter airport code or a valid Weather Station ID
`station_type`	is either `airportCode` or `id`
`start_date`	is a valid string representing a date in the past (YYYY-MM-DD, all numeric)
`end_date`	is a a valid string representing a date in the past (YYYY-MM-DD, all numeric) and is greater than start_date. Default is NULL, in which case the end_date is taken to the same as the start_date

Details

This functions build a custom URL and checks for the data. If available, it will find one row for each date in the date range.

Value

1 if the Station did have weather records, 0 if nothing was found

Examples

```
## Not run:
data_okay <- checkSummarizedDataAvailability("GIG",
                                             "2000-01-01",
                                             "2005-12-31")

## End(Not run)
```

---

getCurrentTemperature *Get the latest recorded temperature for a location*

Description

Function will return the latest avialable temperature at a specified location.

Usage

```
getCurrentTemperature(station_id)
```

Arguments

station_id	is a valid Weather Station ID (example: "BUF", "ORD", "VABB" for Mumbai). Valid Weather Station "id" values: "KFLMIAMI75" or "IMOSCOWO2" You can look these up at wunderground.com. You can get station_id's for a given location by calling getStationCode()

Details

A wrapper for getDetailedWeather(), it returns the last record in the web page. Uses Sys.Date() to get current time. This function returns temperature in Farenheit or Celcius depending on the caller's location.

Value

A one row data frame containing:

- Date and Time stamp (for when the latest temperature reading was recorded)
- Temperature for the station in Farenheit (or Celcius)

References

For a list of valid Weather Stations, try this format http://www.wunderground.com/weatherstation/ListStations.asp?selectedCountry=United+States and replace with your country of interest

Examples

```
## Not run:
getCurrentTemperature(station ="HNL")

## End(Not run)
```

`getDetailedWeather` *Gets weather data for a single date (All records)*

Description

Given a valid station and a single date this function will return a dataframe of time-stamped weather data. It does not summarize the data.

Usage

```
getDetailedWeather(station_id, date, station_type = "airportCode",
  opt_temperature_columns = TRUE, opt_all_columns = FALSE,
  opt_custom_columns = FALSE, custom_columns = NULL,
  opt_compress_output = FALSE, opt_verbose = FALSE, opt_warnings = TRUE)
```

Arguments

`station_id`	is a valid 3-letter airport code or a valid Weather Station ID
`date`	is a valid string representing a date in the past (YYYY-MM-DD)
`station_type`	can be `airportCode` which is the default, or it can be `id` which is a weather-station ID
`opt_temperature_columns`	Boolen flag to indicate only Temperature data is to be returned (default TRUE)
`opt_all_columns`	Boolen flag to indicate whether all available data is to be returned (default FALSE)
`opt_custom_columns`	Boolen flag to indicate if only a user-specified set of columns are to be returned. (default FALSE) If TRUE, then the desired columns must be specified via `custom_columns`
`custom_columns`	Vector of integers specified by the user to indicate which columns to fetch. The Date column is always returned as the first column. The column numbers specfied in `custom_columns` are appended as columns of the data frame being returned (default NULL). The exact column numbers can be found by visiting the weatherUnderground URL, and counting from 1. Note that if `opt_custom_columns` is TRUE, then `custom_columns` must be specified.
`opt_compress_output`	Boolean flag to indicate if a compressed output is preferred. If this option is set to be TRUE, only every other record is returned

opt_verbose	Boolean flag to indicate if verbose output is desired
opt_warnings	Boolean flag to turn off warnings. Default value is TRUE, to keep the warnings on.

Value

A data frame with each row containing:

- Date and Time stamp for the date specified
- Temperature and/or other weather columns

See Also

getWeatherForDate, getSummarizedWeather

Examples

```
## Not run:
getDetailedWeather("NRT", "2014-04-29") #just the Temperature Columns

# Returns all columns available
getDetailedWeather("NRT", "2014-04-29", opt_all_columns=T)

wCDG <- getDetailedWeather("CDG", "2013-12-12",opt_custom_columns=T,
                           custom_columns=c(10,11,12))

## End(Not run)
```

---

getStationCode *Gets the Weather Station code for a location (in the US)*

Description

This function goes through the USAirportWeatherStations dataset and looks for matches. Usually, the 4 letter airportCode is what you are after.

Usage

```
getStationCode(stationName, region = NULL)
```

Arguments

stationName	String that you want to get the weatherStation code for
region	A qualifier about the station's location. It could be a continent or a country. If in the US, region is a two-letter state abbreviation. Ex. "AK" for Alaska

Value

A one row data frame containing:

- A string of Station Name that matched
- the region. (two-letter state abbreviation if in the US)
- The 4-letter weather station ID. (This is the string you use when calling `getDetailedWeather()`)

References

For a world-wide list of possible stations, be sure to look at http://weather.rap.ucar.edu/surface/stations.txt The ICAO (4-letter code is what needs to be input to `getDetailedWeather()`)

Examples

```
getStationCode("Fiji")
getStationCode("Athens", region="GA") # in the US State of Georgia
```

`getSummarizedWeather` *Gets daily summary weather data (One record per day)*

Description

Given a valid station and a single date this function will return a dataframe of time-stamped weather data. All the records are summarized into one record per day. If and `end_date` is specified the function returns 1 record for each day in the date range.

Usage

```
getSummarizedWeather(station_id, start_date, end_date = NULL,
  station_type = "airportCode", opt_temperature_columns = TRUE,
  opt_all_columns = FALSE, opt_custom_columns = FALSE,
  custom_columns = NULL, opt_verbose = FALSE)
```

Arguments

`station_id`	is a valid 3-letter airport code or a valid Weather Station ID
`start_date`	string representing a date in the past ("YYYY-MM-DD")
`end_date`	(optional) string representing a date in the past ("YYYY-MM-DD"), and later than or equal to start_date.
`station_type`	can be `airportCode` which is the default, or it can be `id` which is a weather-station ID
`opt_temperature_columns`	Boolen flag to indicate only Temperature data is to be returned (default TRUE)
`opt_all_columns`	Boolen flag to indicate whether all available data is to be returned (default FALSE)

opt_custom_columns
: Boolen flag to indicate if only a user-specified set of columns are to be returned. (default FALSE) If TRUE, then the desired columns must be specified via custom_columns

custom_columns
: Vector of integers specified by the user to indicate which columns to fetch. The Date column is always returned as the first column. The column numbers specfied in custom_columns are appended as columns of the data frame being returned (default NULL). The exact column numbers can be found by visiting the weatherUnderground URL, and counting from 1. Note that if opt_custom_columns is TRUE, then custom_columns must be specified.

opt_verbose
: Boolean flag to indicate if verbose output is desired

Value

A data frame with each row containing:

- Date stamp for the date specified
- Additional columns of Weather data depending on the options specified

See Also

getWeatherForDate, getDetailededWeather

Examples

```
## Not run:
paris_in_fall<- getSummarizedWeather("CDG", "2013-09-30") #will get Temp columns by default
#
windLHR <- getSummarizedWeather("LHR", "2012-12-12", "2012-12-31",
                                opt_custom_columns=TRUE,
                                custom_columns=c(17,18,19,23))


## End(Not run)
```

---

getWeatherForDate *Getting data for a range of dates*

Description

This function will return a (fairly large) data frame. If you are going to be using this data for future analysis, you can store the results in a CSV file by setting opt_write_to_file to be TRUE

Usage

```
getWeatherForDate(station_id, start_date, end_date = NULL,
  station_type = "airportCode", opt_detailed = FALSE,
  opt_write_to_file = FALSE, opt_temperature_columns = TRUE,
  opt_all_columns = FALSE, opt_custom_columns = FALSE,
  custom_columns = NULL, opt_verbose = FALSE, daily_min = FALSE,
  daily_max = FALSE)
```

Arguments

station_id	is a valid 3- or 4-letter Airport code or a valid Weather Station ID (example: "BUF", "ORD", "VABB" for Mumbai). Valid Weather Station "id" values: "KFLMIAMI75" or "IMOSCOWO2" You can look these up at wunderground.com
start_date	string representing a date in the past ("YYYY-MM-DD", all numeric)
end_date	If an interval is to be specified, end_date is a string representing a date in the past ("YYYY-MM-DD", all numeric) and greater than the start_date (Optional)
station_type	= "airportCode" (3- or 4-letter airport code) or "ID" (Wx call Sign)
opt_detailed	Boolen flag to indicate if detailed records for the station are desired. (default FALSE). By default only one records per date is returned.
opt_verbose	Boolean flag to indicate if verbose output is desired
daily_min	A boolean indicating if only the Minimum Temperatures are desired
daily_max	A boolean indicating if only the Maximum Temperatures are desired
opt_write_to_file	If TRUE, the resulting dataframe will be stored in a CSV file. Default is FALSE
opt_temperature_columns	Boolen flag to indicate only Temperature data is to be returned (default TRUE)
opt_all_columns	Boolen flag to indicate whether all available data is to be returned (default FALSE)
opt_custom_columns	Boolen flag to indicate if only a user-specified set of columns are to be returned. (default FALSE) If TRUE, then the desired columns must be specified via custom_columns
custom_columns	Vector of integers specified by the user to indicate which columns to fetch. The Date column is always returned as the first column. The column numbers specfied in custom_columns are appended as columns of the data frame being returned (default NULL). The exact column numbers can be found by visiting the weatherUnderground URL, and counting from 1. Note that if opt_custom_columns is TRUE, then custom_columns must be specified.

Details

For each day in the date range, this function fetches Weather Data. Internally, it makes multiple calls to getDetailedWeather.

Value

A data frame with each row containing:

- Date and Time stamp (for each date specified)
- Temperature and/or other weather columns sought

References

For a list of valid Weather Stations, try this format http://www.wunderground.com/weatherstation/ListStations.asp?selectedCountry=United+States and replace with your country of interest

Examples

```
## Not run:
dat <- getWeatherForDate("PHNL", "2013-08-10", 2013-08-31")
d3<- getWeatherForDate("CWWF", start_date="2014-03-01",
                        end_date = "2014-03-03",
                        opt_detailed = TRUE,
                        opt_all_columns = TRUE)

## End(Not run)
```

---

getWeatherForYear *Get weather data for one full year*

Description

Function will return a data frame with all the records for a given station_id and year. If the current year is supplied, it will returns records until the current Sys.Date() ("today")

Usage

```
getWeatherForYear(station_id, year, station_type = "airportCode",
  opt_detailed = FALSE, opt_write_to_file = FALSE)
```

Arguments

station_id	is a valid Weather Station ID (example: "BUF", "ORD", "VABB" for Mumbai). Valid Weather Station "id" values: "KFLMIAMI75" or "IMOSCOWO2" You can look these up at wunderground.com. You can get station_id's for a given location by calling `getStationCode()`
year	is a valid year in the past (numeric, YYYY format)
station_type	= "airportCode" (3 or 4 letter airport code) or "ID" (Wx call Sign)
opt_detailed	Boolen flag to indicate if detailed records for the station are desired. (default FALSE). By default only one records per date is returned.
opt_write_to_file	If TRUE, the resulting dataframe will be stored in a CSV file. Default is FALSE

Details

Note that this function is a light wrapper for getWeatherForDate with the two end dates being Jan-01 and Dec-31 of the given year.

Value

A data frame with each row containing:

- Date and Time stamp (for each date specified)
- Temperature and/or other weather columns sought

References

For a list of valid Weather Stations, try this format http://www.wunderground.com/weatherstation/ListStations.asp?selectedCountry=United+States and replace with your country of interest

Examples

```
## Not run:
dat <- getWeatherForYear("KLGA", 2013)

# If opt_detailed is turned on, you will get a large data frame
wx_Singapore <- getWeatherForYear("SIN", 2014, opt_detailed=TRUE)

## End(Not run)
```

IntlWxStations *Data - International Weather Stations*

Description

This is a data frame of the 1602 stations in Weather Underground's database. The 4-letter "ICAO" is used by the functions in this package to check and get the weather data. Note that not all the stations have weather data.

Usage

```
data(IntlWxStations)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

This data frame has been created by http://weather.rap.ucar.edu/surface/stations.txt maintained by Greg Thompson of NCAR.

London2013 *Data - Ambient Temperature for the City of London for all of 2013*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has two entries (columns). The Timestamp (YYYY-MM-DD HH:MM:SS) and the Temperature (in degrees F)

Usage

```
data(London2013)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/EGLL/2013/1/1/DailyHistory.html?format=1

Mumbai2013 *Data - Ambient Temperature for the City of Mumbai, India for all of 2013*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has two entries (columns). The Timestamp (YYYY-MM-DD HH:MM:SS) and the Temperature (in degrees F)

Usage

```
data(Mumbai2013)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/VABB/2014/1/1/DailyHistory.html?format=1

NewYork2013 *Data - Ambient Temperature for New York City for all of 2013*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has two entries (columns). The Timestamp (YYYY-MM-DD HH:MM:SS) and the Temperature (in degrees F)

Usage

```
data(NewYork2013)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/KLGA/2013/1/1/DailyHistory.html?format=1

SFO2012 *Data - Ambient Temperature for the City of San Francisco for all of 2012*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has two entries (columns). The Timestamp (YYYY-MM-DD HH:MM:SS) and the Temperature (in degrees F)

Usage

```
data(SFO2012)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/KSFO/2012/1/1/DailyHistory.html?format=1

SFO2013 *Data - Ambient Temperature for the City of San Francisco for all of 2013*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has two entries (columns). The Timestamp (YYYY-MM-DD HH:MM:SS) and the Temperature (in degrees F)

Usage

```
data(SFO2013)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/KSFO/2013/1/1/DailyHistory.html?format=1

SFO2013Summarized *Data - Summarized Daily Temperature for the City of San Francisco for all of 2013*

Description

This is a data frame of Ambient temperature data, extracted from Weather Undergound. Each row has four columns. The Timestamp (YYYY-MM-DD HH:MM:SS) and three Temperature Columns: Daily Max, Mean and Min (in degrees F) In comparison with the SFO2013 dataset which has 9507 rows, this dataset has exactly 365 rows, one for each day in 2013.

Usage

```
data(SFO2013Summarized)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/history/airport/SFO/2013/1/1/CustomHistory.html?dayend=31&monthend=12&yearend=2013&req_city=NA&req_state=NA&req_statename=NA&format=1

showAvailableColumns *Shows all the available Weather Data Columns*

Description

Displays all the columns that are available in the website, for the given station, and date range. Useful when only a subset of the columns are desired. Those can be specfied using the custom_columns vector. Note: There are different columns available for summarized vs. detailed data. Be sure to turn the opt_detailed flag to be TRUE if multiple records per day is desired.

Usage

```
showAvailableColumns(station_id, start_date, end_date = NULL,
  station_type = "airportCode", opt_detailed = FALSE, opt_verbose = FALSE)
```

Arguments

station_id	is a valid 3-letter airport code or a valid Weather Station ID
start_date	string representing a date in the past ("YYYY-MM-DD")
end_date	string representing a date in the past ("YYYY-MM-DD"), and later than or equal to start_date.
station_type	can be airportCode which is the default, or it can be id which is a weather-station ID
opt_detailed	Boolen flag to indicate if detailed records for the station are desired. (default FALSE). By default only one record per date is returned.
opt_verbose	Boolean flag to indicate if verbose output is desired (default FALSE)

Examples

```
## Not run:
showAvailableColumns("NRT", "2014-04-04")

#if you want to see the columns for the *detailed* weather, turn on opt_detailed
showAvailableColumns("CDG", "2013-12-12", opt_detailed=T)

## End(Not run)
```

USAirportWeatherStations

Data - US Weather Stations ID's

Description

This is a data frame of the 1602 stations in Weather Underground's database. The 4-letter "airportCode" is used by functions to check and get the weather data.

Usage

```
data(USAirportWeatherStations)
```

Author(s)

Ram Narasimhan <ramnarasimhan@gmail.com>

References

http://www.wunderground.com/about/faq/US_cities.asp

Index