Package ‘rwunderground’

October 14, 2022

Type Package
Title R Interface to Weather Underground API
Version 0.1.8
Date 2018-05-01
Author Alex Shum <alex@ALShum.com>
Maintainer Eric Hare <eric@omnianalytics.io>
Description Tools for getting historical weather information and forecasts from wunderground.com. Historical weather and forecast data includes, but is not limited to, temperature, humidity, windchill, wind speed, dew point, heat index. Additionally, the weather underground weather API also includes information on sunrise/sunset, tidal conditions, satellite/webcam imagery, weather alerts, hurricane alerts and historical high/low temperatures.

URL https://github.com/ALShum/rwunderground,
    http://www.wunderground.com/weather/api

BugReports https://github.com/alshum/rwunderground/issues
License GPL (>= 2)
Imports httr, dplyr, countrycode, lubridate, tibble
LazyData TRUE
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2018-05-01 16:28:16 UTC

R topics documented:

   alerts ......................................................... 2
   almanac ....................................................... 3
   as.numeric.nonempty ....................................... 4
   astronomy .................................................... 4
   base_url ..................................................... 5
Alerts

Weather Alerts for United States and Europe

Index

alerts  Weather Alerts for United States and Europe

Description

Weather Alerts for United States and Europe
Usage

\[
\text{alerts}(\text{location}, \text{key} = \text{get_api_key()}, \text{raw} = \text{FALSE}, \text{raw\_JSON} = \text{FALSE},
\text{message} = \text{TRUE})
\]

Arguments

- location: location set by set_location
- key: weather underground API key
- raw: if TRUE return raw httr object
- raw\_JSON: if TRUE return entire alert as JSON
- message: if TRUE print out requested URL

Value

A string containing alert type, message, start time and expiration.

Examples

```r
## Not run:
alerts(set_location(territory = "Hawaii", city = "Honolulu"))
alerts(set_location(airport_code = "SEA"))
alerts(set_location(zip_code = "90210"))
alerts(set_location(territory = "IR", city = "Tehran"))
## End(Not run)
```

---

**almanac**

Average and record high and low temperatures for current date going back as far as weather underground has data or from the national weather service going back 30 years.

Description

Average and record high and low temperatures for current date going back as far as weather underground has data or from the national weather service going back 30 years.

Usage

\[
\text{almanac}(\text{location}, \text{use\_metric} = \text{FALSE}, \text{key} = \text{get_api\_key()}, \text{raw} = \text{FALSE},
\text{message} = \text{TRUE})
\]

Arguments

- location: location set by set_location
- use\_metric: Metric or imperial units
- key: weather underground API key
- raw: if TRUE return raw httr object
- message: if TRUE print out requested URL
Value

tbl_df with columns: location, airport, avg_high, record high, avg_low, record low.

Examples

```r
## Not run:
almanac(set_location(territory = "Hawaii", city = "Honolulu"))
almanac(set_location(airport_code = "SEA"))
almanac(set_location(zip_code = "90210"))
almanac(set_location(territory = "IR", city = "Tehran"))

## End(Not run)
```

as.numeric.nonempty  as.numeric with special handling for length 0 (NULL) objects

Description

as.numeric with special handling for length 0 (NULL) objects

Usage

```r
## S3 method for class 'nonempty'
as.numeric(x)
```

Arguments

x  the object to cast as numeric

Value

value of type double

astronomy  Moon phase, sunrise and sunset times for today.

Description

Moon phase, sunrise and sunset times for today.

Usage

```r
astronomy(location, key = get_api_key(), raw = FALSE, message = TRUE)
```
**base_url**

**Arguments**

- **location**: location set by `set_location`
- **key**: weather underground API key
- **raw**: if TRUE return raw `httr` object
- **message**: if TRUE print out requested URL

**Value**

tbl_df with: location, moon phase, percent visible, moon rise and set times, sun rise and set times.

**Examples**

```r
## Not run:
astronomy(set_location(territory = "Hawaii", city = "Honolulu"))
astronomy(set_location(airport_code = "SEA"))
astronomy(set_location(zip_code = "90210"))
astronomy(set_location(territory = "IR", city = "Tehran"))
## End(Not run)
```

---

**Description**

Base URL for wunderground API

**Usage**

`base_url()`

**Value**

base wunderground URL
### build_url

*Build wunderground request URL*

**Description**

Build wunderground request URL

**Usage**

```r
build_url(key = get_api_key(), request_type, date, location)
```

**Arguments**

- **key**: wunderground API key
- **request_type**: request type TODO::list all request_types
- **date**: Date, only applicable for history requests
- **location**: location set by set_location

### conditions

*Current conditions including current temperature, weather condition, humidity, wind, feels-like, temperature, barometric pressure, and visibility.*

**Description**

Current conditions including current temperature, weather condition, humidity, wind, feels-like, temperature, barometric pressure, and visibility.

**Usage**

```r
conditions(location, use_metric = FALSE, key = get_api_key(), raw = FALSE, message = TRUE)
```

**Arguments**

- **location**: location set by set_location
- **use_metric**: Metric or imperial units
- **key**: weather underground API key
- **raw**: if TRUE return raw httr object
- **message**: if TRUE print out requested URL

**Value**

tbl_df with conditions
Examples

```r
## Not run:
conditions(set_location(territory = "Hawaii", city = "Honolulu"))
conditions(set_location(airport_code = "SEA"))
conditions(set_location(zip_code = "90210"))
conditions(set_location(territory = "IR", city = "Tehran"))

## End(Not run)
```

---

**current_hurricane**

*Current hurricane - within the US only. Note: all times in eastern*

Description

Current hurricane - within the US only. Note: all times in eastern

Usage

```r
current_hurricane(key = get_api_key(), use_metric = FALSE, raw = FALSE, message = TRUE)
```

Arguments

- `key`: weather underground API key
- `use_metric`: Metric or imperial units
- `raw`: if TRUE return raw httr object
- `message`: if TRUE print out requested URL

Value

Hurricane info

Examples

```r
## Not run:
current_hurricane()

## End(Not run)
```
**dst_POSIXct**  
*Return POSIXct time from 7 variables.*

**Description**

In locations with a Daylight Saving/Standard time change that occurs twice annually, the year has one 23 hour day and one 25 hour day, if by day we mean "an ordered set of all instants in time which are assigned the same date". In the US/Los_Angeles timezone, there is one day in the spring where there are no valid times between the moment before 02:00:00 and 03:00:00. Similarly, there is one day in the fall where there are two instants described by all times between 01:00:00 and 01:59:59, first as a set of PDT times, then as a set of PST times. `as.POSIXct()` doesn’t handle this case well. Times inside this region are assigned to DST until the sequence of clock times has a time which is the same or earlier than its predecessor, and all subsequent ambiguous times are assigned to Standard Time.

**Usage**

```r
dst_POSIXct(y, m, d, hr, mn, sec, tz)
```

**Arguments**

- `y` vector of years
- `m` vector of months
- `d` vector of days
- `hr` vector of hours
- `mn` vector of minutes
- `sec` vector of seconds
- `tz` vector of timezones

**Value**

POSIXct time assuming vectors sorted by true chronological order, at least for the hour that "occurs twice", once with Daylight Time, then again with Standard Time. If there are no nonmonotonicities in the times, all times in this hour will be assumed to be Daylight Time.

**dst_repeat_starttime**  
*Find the text to POSIXct ambiguous interval.*

**Description**

Assumes that DST transitions happen on hour boundaries, which is true almost everywhere, and that the wall clock shifts back and repeats exactly 1 hour, again true almost everywhere. This code relies on R and the OS to properly manage DST in all timezones.
encode_NA

Usage

dst_repeat_starttime(y, m, d, tz)

Arguments

y the year
m the month
d the day
tz the timezone

Value

list of two integers between 0000 and 2359, hhmm format. the first integer is the beginning of the
interval of clock times which correspond to 2 separate instants of time, the second is the end of that
interval. The left endpoint is ambiguous, the right endpoint is not since it maps only to Standard
Time.

encode_NA Processes data.frames and replaces wunderground’s -9999/-999 to NAs

Description

Processes data.frames and replaces wunderground’s -9999/-999 to NAs

Usage

encode_NA(df)

Arguments

df the data.frame to process

Value

data.frame with correctly encoded NAs
### forecast10day

**Forecast for the next 10 days.**

**Description**

Forecast for the next 10 days.

**Usage**

```r
forecast10day(location, use_metric = FALSE, key = get_api_key(), raw = FALSE, message = TRUE)
```

**Arguments**

- `location`: location set by `set_location`
- `use_metric`: Metric or imperial units
- `key`: weather underground API key
- `raw`: if TRUE return raw htr object
- `message`: if TRUE print out requested URL

**Value**

tbl_df with date (in posix format), high and low temp, conditions, precipitation, rain, snow, max and avg wind speed, max/min and avg humidity

**Examples**

```r
## Not run:
forecast10day(set_location(territory = "Hawaii", city = "Honolulu"))
forecast10day(set_location(airport_code = "SEA"))
forecast10day(set_location(zip_code = "90210"))
forecast10day(set_location(territory = "IR", city = "Tehran"))
## End(Not run)
```

### forecast3day

**Forecast for the next 3 days.**

**Description**

Forecast for the next 3 days.

**Usage**

```r
forecast3day(location, use_metric = FALSE, key = get_api_key(),
             raw = FALSE, message = TRUE)
```
Arguments

location: location set by set_location
use_metric: Metric or imperial units
key: weather underground API key
raw: if TRUE return raw httr object
message: if TRUE print out requested URL

Value

tbl_df with date (in posix format), high and low temp, conditions, precipitation, rain, snow, max and avg wind speed, max/min and avg humidity

Examples

## Not run:
forecast3day(set_location(territory = "Hawaii", city = "Honolulu"))
forecast3day(set_location(airport_code = "SEA"))
forecast3day(set_location(zip_code = "90210"))
forecast3day(set_location(territory = "IR", city = "Tehran"))

## End(Not run)
Examples

```r
## Not run:
geolookup(set_location(territory = "Hawaii", city = "Honolulu"))
geolookup(set_location(airport_code = "SEA"))
geolookup(set_location(zip_code = "90210"))
geolookup(set_location(territory = "IR", city = "Tehran"))

## End(Not run)
```

---

### get_api_key

*Description*

Returns the wunderground API key

*Usage*

```r
get_api_key()
```

*Value*

API key

*Examples*

```r
## Not run:
get_api_key()

## End(Not run)
```

---

### has_api_key

*Description*

Detects if wunderground API key is set

*Usage*

```r
has_api_key()
```

*Value*

TRUE if API key set, otherwise FALSE
history

Hourly weather data for specified date.

Description

Hourly weather data for specified date.

Usage

history(location, date = "20150101", use_metric = FALSE, key = get_api_key(), raw = FALSE, message = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>location set by set_location</td>
</tr>
<tr>
<td>date</td>
<td>Date as YYYYMMDD format</td>
</tr>
<tr>
<td>use_metric</td>
<td>Metric or imperial units</td>
</tr>
<tr>
<td>key</td>
<td>weather underground API key</td>
</tr>
<tr>
<td>raw</td>
<td>if TRUE return raw httr object</td>
</tr>
<tr>
<td>message</td>
<td>if TRUE print out requested URL</td>
</tr>
</tbody>
</table>

Value

tbl_df with date, temperature, dew point, humidity, wind speed, gust and direction, visibility, pressure, wind chill, heat index, precipitation, condition, fog, rain, snow, hail, thunder, tornado

Examples

```r
## Not run:
history(set_location(territory = "Hawaii", city = "Honolulu"), "20130101")
history(set_location(airport_code = "SEA"), "20130101")
history(set_location(zip_code = "90210"), "20130131")
history(set_location(territory = "IR", city = "Tehran"), "20140131")

## End(Not run)
```
history_daily  
*Summarized weather data for specified date.*

**Description**

Summarized weather data for specified date.

**Usage**

```r
history_daily(location, date = "20150101", use_metric = FALSE,
key = get_api_key(), raw = FALSE, message = TRUE)
```

**Arguments**

- `location`: location set by `set_location`
- `date`: Date as YYYMMDD format
- `use_metric`: Metric or imperial units
- `key`: weather underground API key
- `raw`: if TRUE return raw htr object
- `message`: if TRUE print out requested URL

**Value**

`tbl_df` of summarized weather

**Examples**

```r
## Not run:
history_daily(set_location(territory = "Hawaii", city = "Honolulu"), "20130101")
history_daily(set_location(airport_code = "SEA"), "20130101")
history_daily(set_location(zip_code = "90210"), "20130131")
history_daily(set_location(territory = "IR", city = "Tehran"), "20140131")

## End(Not run)
```

---

history_range  
*Hourly weather data for specified date range.*

**Description**

Hourly weather data for specified date range.
hourly

Usage

```r
history_range(location, date_start = "20150101", date_end = "20150105",
limit = 10, no_api = FALSE, use_metric = FALSE, key = get_api_key(),
raw = FALSE, message = TRUE)
```

Arguments

- `location`: location set by `set_location`
- `date_start`: start date
- `date_end`: end date
- `limit`: Maximum number of API requests per minute, NULL to have no limits
- `no_api`: bypass API and use URL requests
- `use_metric`: Metric or imperial units
- `key`: weather underground API key
- `raw`: if TRUE return raw httr object
- `message`: if TRUE print out requested URL

Value

tbl_df with date, temperature, dew point, humidity, wind speed, gust and direction, visibility, pressure, wind chill, heat index, precipitation, condition, fog, rain, snow, hail, thunder, tornado

Examples

```r
## Not run:
history_range(set_location(territory = "Hawaii", city = "Honolulu"), "20130101", "20130105")
history_range(set_location(airport_code = "SEA"), "20130101", "20130105")
history_range(set_location(zip_code = "90210"), "20130131", "20130205")
history_range(set_location(territory = "IR", city = "Tehran"), "20140131", "20140202")
## End(Not run)
```

hourly

*Hourly forecast for the next 24 hours.*

Description

Hourly forecast for the next 24 hours.

Usage

```r
hourly(location, use_metric = FALSE, key = get_api_key(), raw = FALSE, message = TRUE)
```
hourly10day

Hourly forecast for the next 10 days.

Description

Hourly forecast for the next 10 days.

Usage

hourly10day(location, use_metric = FALSE, key = get_api_key(),
raw = FALSE, message = TRUE)

Arguments

location     location set by set_location
use_metric   Metric or imperial units
key          weather underground API key
raw          if TRUE return raw httr object
message      if TRUE print out requested URL

Value

tbl_df with date, temperature, dew point, condition, wind speed and direction, UV index, humidity,
windchill, heat index, real feel, rain, snow, pop, mslp
### is_fall_back_day

Check if a date is a “fall back” transition from DST.

#### Description
Check if a date is a “fall back” transition from DST.

#### Usage

```r
is_fall_back_day(y, m, d, tz)
```

#### Arguments

- `y`: the year
- `m`: the month
- `d`: the day
- `tz`: the timezone

#### Value

logical

### is_valid_airport

Checks if airport code is valid.

#### Description
Checks if airport code is valid.

#### Usage

```r
is_valid_airport(name)
```

#### Arguments

- `name`: Airport code either IATA or ICAO
**is_valid_territory**  
*Checks if country/state is a valid one*

**Description**  
Checks if country/state is a valid one

**Usage**  
is_valid_territory(name)

**Arguments**  
name  
Name of state or country

**Value**  
TRUE if valid state or country otherwise FALSE

---

**list_airports**  
*Returns a data.frame of valid airport codes (ICAO and IATA).*

**Description**  
This dataset is from the openflights.org airport database. It can be found at [http://openflights.org/data.html#airport](http://openflights.org/data.html#airport). This data is provided under the open database license – more information can be found here: [http://opendatacommons.org/licenses/odbl/1.0/](http://opendatacommons.org/licenses/odbl/1.0/).

**Usage**  
list_airports()

**Value**  
data.frame of airport codes with country and city

**Examples**  
```r
## Not run:
list_airports()

## End(Not run)
```
list_countries

Description
Returns a data.frame of valid countries with iso abbreviations and region

Usage
list_countries()

Value
data.frame of valid country names with iso codes

Examples
## Not run:
list_countries()
## End(Not run)

list_states

Description
Returns a data.frame of valid states with abbreviations and regions

Usage
list_states()

Value
data.frame of states with abbreviation and region

Examples
## Not run:
list_states()
## End(Not run)
lookup_airport

Lookup airport code (IATA and ICAO code). weatherunderground API might not recognize the IATA/ICAO code for smaller airports.

Description

Lookup airport code (IATA and ICAO code). weatherunderground API might not recognize the IATA/ICAO code for smaller airports.

Usage

lookup_airport(location, region = NULL)

Arguments

location  
location string
region  
region string

Value

data.frame of matching airport name and IATA/ICAO codes

Examples

## Not run:
lookup_airport("Honolulu")
lookup_airport("Pyongyang")
lookup_airport("Portland", region = "Los_Angeles")

## End(Not run)

lookup_country_code

Lookup ISO country code weatherunderground API doesn’t recognize iso codes uniformly for every country.name

Description

Lookup ISO country code weatherunderground API doesn’t recognize iso codes uniformly for every country.name

Usage

lookup_country_code(name, region = NULL)
measurement_exists

Arguments

<table>
<thead>
<tr>
<th>name</th>
<th>Name of country</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>Geographic region</td>
</tr>
</tbody>
</table>

Value
data.frame of country codes

Examples

```r
## Not run:
lookup_country_code("Korea")
lookup_country_code("Guinea", region = "Africa")
## End(Not run)
```

---

measurement_exists \textit{Check if a variable exists for a PWS. If not set the value to -9999}

Description

Check if a variable exists for a PWS. If not set the value to -9999

Usage

```r
measurement_exists(x, class = "numeric")
```

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>the value to check</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>a character given the desired class for the variable</td>
</tr>
</tbody>
</table>

---

nonempty \textit{return object, or NA for length 0 (NULL) objects}

Description

return object, or NA for length 0 (NULL) objects

Usage

```r
nonempty(x)
```

Arguments

| x              | the object to cast as numeric |
planner

Weather summary based on historical information between the specified dates

Description

Weather summary based on historical information between the specified dates

Usage

```r
planner(location, use_metric = FALSE, start_date = "0501", end_date = "0531", key = get_api_key(), raw = FALSE, message = TRUE)
```

Arguments

- `location`: location set by `set_location`
- `use_metric`: Metric or imperial units
- `start_date`: Start date as MMDD
- `end_date`: End date as MMDD
- `key`: weather underground API key
- `raw`: if TRUE return raw httr object
- `message`: if TRUE print out requested URL

Value

- `tbl_df`

Examples

```r
## Not run:
planner(set_location(territory = "Hawaii", city = "Honolulu"),
         start_date = "0101", end_date = "0131")
planner(set_location(territory = "Washington", city = "Seattle"),
         start_date = "01201", end_date = "1231")
planner(set_location(territory = "Louisiana", city = "New Orleans"),
         start_date = "0501", end_date = "0531")
```

## End(Not run)
Raw Tidal data with data every 5 minutes for US locations. Tidal information only available for US cities. Units are in feet.

Usage

```r
rawtide(location, key = get_api_key(), raw = FALSE, message = TRUE)
```

Arguments

- `location`: location set by `set_location`
- `key`: weather underground API key
- `raw`: if TRUE return raw `httr` object
- `message`: if TRUE print out requested URL

Value

tbl_df with time (epoch) and height

Examples

```r
## Not run:
rawtide(set_location(territory = "Hawaii", city = "Honolulu"))
rawtide(set_location(territory = "Washington", city = "Seattle"))
rawtide(set_location(territory = "Louisiana", city = "New Orleans"))

## End(Not run)
```

Returns image URL for satellite imagery

Description

Returns image URL for satellite imagery

Usage

```r
satellite(location, key = get_api_key(), raw = FALSE, message = TRUE)
```
Arguments

location  location set by set_location
key      weather underground API key
raw      if TRUE return raw httr object
message  if TRUE print out requested URL

Value

URL to satellite imagery

Examples

## Not run:
satellite(set_location(territory = "Hawaii", city = "Honolulu"))
satellite(set_location(territory = "Washington", city = "Seattle"))
satellite(set_location(territory = "Louisiana", city = "New Orleans"))

## End(Not run)

---

set_api_key

Sets the wunderground API key

Description

Sets the wunderground API key

Usage

set_api_key(key)

Arguments

key  wunderground API key

Value

API key

Examples

## Not run:
set_api_key("1a2b3c4d")

## End(Not run)
set_location

Specifies location of request

Description

This is a wrapper function that will validate and format location strings for requesting data from weather underground.

Usage

set_location(zip_code = NULL, territory = NULL, city = NULL,
airport_code = NULL, PWS_id = NULL, lat_long = NULL, autoip = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip_code</td>
<td>zip code</td>
</tr>
<tr>
<td>territory</td>
<td>state if in US, otherwise country</td>
</tr>
<tr>
<td>city</td>
<td>city name</td>
</tr>
<tr>
<td>airport_code</td>
<td>IATA/ICAO airport code</td>
</tr>
<tr>
<td>PWS_id</td>
<td>personal weather station ID</td>
</tr>
<tr>
<td>lat_long</td>
<td>latitude and longitude, as a comma-separated string</td>
</tr>
<tr>
<td>autoip</td>
<td>location based on IP</td>
</tr>
</tbody>
</table>

Value

formatted and validated location string

Examples

set_location(zip_code = "90210")
set_location(territory = "Hawaii", city = "Honolulu")
set_location(territory = "Kenya", city = "Mombasa")
set_location(airport_code = "SEA")
set_location(PWS_id = "KMNCHASK10")
set_location(lat_long="40.6892,-74.0445")
set_location(autoip = "172.227.205.140")
set_location()
stop_for_error  
_Detect and stop for any wunderground request errors_

**Description**
Detect and stop for any wunderground request errors

**Usage**
```r
stop_for_error(httr_parsed_req)
```

**Arguments**
- `httr_parsed_req`: httr request object

---

tide  
_Tidal information for a location within the USA. Tidal information only available for US cities. Units are in feet._

**Description**
Tidal information for a location within the USA. Tidal information only available for US cities. Units are in feet.

**Usage**
```r
tide(location, key = get_api_key(), raw = FALSE, message = TRUE)
```

**Arguments**
- `location`: location set by `set_location`
- `key`: weather underground API key
- `raw`: if TRUE return raw httr object
- `message`: if TRUE print out requested URL

**Value**
tbl_df with date, height and type

**Examples**
```r
## Not run:
tide(set_location(territory = "Hawaii", city = "Honolulu"))
tide(set_location(territory = "Washington", city = "Seattle"))
tide(set_location(territory = "Louisiana", city = "New Orleans"))
## End(Not run)
```
webcam

Returns locations of personal weather stations along with URLs for their webcam images

Description

Returns locations of personal weather stations along with URLs for their webcam images

Usage

webcam(location, key = get_api_key(), raw = FALSE, message = TRUE)

Arguments

- **location**: location set by set_location
- **key**: weather underground API key
- **raw**: if TRUE return raw httr object
- **message**: if TRUE print out requested URL

Value

tbl_df of weather stations including: handle, id, city, state, country, tz, lat, lon, last updated, image URL and cam URL.

Examples

```r
## Not run:
webcam(set_location(territory = "Hawaii", city = "Honolulu"))
webcam(set_location(territory = "Iowa", city = "Iowa City"))
webcam(set_location(territory = "Iraq", city = "Baghdad"))
## End(Not run)
```

wunderground_request

wunderground api requests

Description

wunderground api requests

Usage

wunderground_request(request_type, location, date = NULL,
key = get_api_key(), message = TRUE)
Arguments

- `request_type` Request type TODO::list all types
- `location` locations set of set_location
- `date` Date, only applicable for history requests
- `key` wunderground API key
- `message` if TRUE print out requested

Value

httr request object

---

yesterday | Weather data for yesterday

Description

Weather data for yesterday

Usage

```r
yesterday(location, use_metric = FALSE, key = get_api_key(), raw = FALSE, message = TRUE, summary = FALSE)
```

Arguments

- `location` location set by set_location
- `use_metric` Metric or imperial units
- `key` weather underground API key
- `raw` if TRUE return raw httr object
- `message` if TRUE print out requested URL
- `summary` If TRUE return daily summary otherwise hourly data

Value

tbl_df with date, temperature, dew point, humidity, wind speed, gust and direction, visibility, pressure, wind chill, heat index, precipitation, condition, fog, rain, snow, hail, thunder, tornado

Examples

```r
## Not run:
yesterday(set_location(territory = "Hawaii", city = "Honolulu"))
yesterday(set_location(territory = "Iowa", city = "Iowa City"))
yesterday(set_location(territory = "Iraq", city = "Baghdad"))
yesterday(set_location(territory = "IR", city = "Tehran"), summary = TRUE)

## End(Not run)
```
Index

alerts, 2
almanac, 3
as.numeric.nonempty, 4
astronomy, 4

base_url, 5
build_url, 6

conditions, 6
current_hurricane, 7
dst_POSIXct, 8
dst_repeat_starttime, 8

encode_NA, 9
forecast10day, 10
forecast3day, 10

geolookup, 11
get_api_key, 12

has_api_key, 12
history, 13
history_daily, 14
history_range, 14
hourly, 15
hourly10day, 16

is_fall_back_day, 17
is_valid_airport, 17
is_valid_territory, 18

list_airports, 18
list_countries, 19
list_states, 19
lookup_airport, 20
lookup_country_code, 20

measurement_exists, 21
nonempty, 21

planner, 22
rawtide, 23
satellite, 23
set_api_key, 24
set_location, 25
stop_for_error, 26
tide, 26
webcam, 27
wunderground_request, 27
yesterday, 28