

# Package ‘SkyWatchr’

June 30, 2017

**Type** Package

**Title** Search and Download Satellite Imagery using the SkyWatch API

**Version** 0.8-2

**Date** 2017-06-30

**Author** Ali Santacruz, SkyWatch API Developers

**Maintainer** Ali Santacruz <amsantac@unal.edu.co>

## Description

Query and download satellite imagery and climate/atmospheric datasets using the SkyWatch API. Search datasets by wavelength (band), cloud cover, resolution, location, date, etc. Get the query results as data frame and as HTML. To learn more about the Sky-Watch API, see <<https://github.com/skywatchspaceapps/api>>.

**Depends** R (>= 2.14.0), httr, htmlTable, sp, methods

**License** GPL (>= 2)

**URL** <https://github.com/amsantac/SkyWatchr>

**Encoding** latin1

**LazyData** true

**NeedsCompilation** no

## R topics documented:

SkyWatchr-package . . . . .	2
downloadSW . . . . .	2
getPolygon . . . . .	3
querySW . . . . .	4
<b>Index</b>	<b>8</b>

---

SkyWatchr-package

*Search and Download Satellite Imagery using the SkyWatch API*


---

### Description

Query and download satellite imagery and climate/atmospheric datasets using the SkyWatch API. Search datasets by wavelength (band), cloud cover, resolution, location, date, etc. Get the query results as data frame and as HTML.

To learn more about the SkyWatch API, see <https://github.com/skywatchspaceapps/api>. An API key for free access to the SkyWatch API can be requested at <http://www.skywatch.co/request-access>.

### Details

```
Package:    SkyWatchr
Type:       Package
Version:    0.8-2
Date:       2017-06-30
License:    GPL (>= 2)
LazyLoad:   yes
```

### Author(s)

Ali Santacruz, SkyWatch API Developers

Maintainer: Ali Santacruz <amsantac@unal.edu.co>

---

downloadSW

*Download satellite imagery and climate/atmospheric datasets using the SkyWatch API*


---

### Description

Download satellite imagery and climate/atmospheric datasets using the SkyWatch API based on a query output object obtained from the querySW function.

### Usage

```
downloadSW(x, subset, api_key = NULL)
```

### Arguments

```
x                data.frame returned by the querySW function
subset           logical expression indicating elements or rows to keep. See details
api_key          string; personal alphanumeric API key. See Details section
```

**Details**

`x` must be a data.frame returned by the querySW function.

If `indexes` is NULL, all files in the `download_path` column in `x` are downloaded. A numeric vector indicating rows of `x` can be provided to `indexes` to download only selected files in the `download_path` column of `x`.

The subset argument works on the rows of the `x` object. Columns can be referred to (by name) as variables in the expression (see the examples).

**Value**

Returns nothing

**Examples**

```
## Not run:
api_key <- "your_personal_alphanumeric_api_key"

# Set the SkyWatchr.apikey option
options(SkyWatchr.apikey = api_key)

res <- querySW(time_period = "2015-06", longitude_latitude = "48.676074,31.321119",
               data_level = 3)
View(res)

# Download all files
downloadSW(res)

# Download a subset
downloadSW(res[c(1,3), ])

# Use an expression to subset files to be downloaded
downloadSW(res, source == "MOPITT" & size_kb < 2400)

## End(Not run)
```

---

getPolygon

*Get the boundary box for a given dataset retrieved in a query*


---

**Description**

getPolygon creates an SpatialPolygonsDataFrame object that displays the boundary box for a given dataset based on info from the `*area*` column in the data.frame generated by querySW.

**Usage**

```
getPolygon(x, index)
```

**Arguments**

<code>x</code>	data.frame returned by the querySW function
<code>index</code>	numeric; index (row) of a record in <code>x</code> . See Details section

## Details

getPolygon creates an SpatialPolygonsDataFrame object that displays the boundary box for a given dataset (corresponding to row index) based on info from the *\*area\** column in the data.frame generated by querySW.

## Value

a SpatialPolygonsDataFrame object

## Examples

```
## Not run:
api_key <- "your_personal_alphanumeric_api_key"

# Set the SkyWatchr.apikey option
options(SkyWatchr.apikey = api_key)

res <- querySW(time_period = "2015-8", longitude_latitude = "-71.1043443,-42.3150676")
sppolygon <- getPolygon(res, 55)

library(mapview)
mapView(sppolygon)

## End(Not run)
```

---

querySW	<i>Query the SkyWatch API for satellite imagery and climate/atmospheric datasets</i>
---------	--

---

## Description

Search satellite imagery and climate/atmospheric datasets by wavelength (band), cloud cover, resolution, location, date, etc. using the SkyWatch API. Available datasets include ACOS, AIRS, CAI, FTS-SWIR, Landsat-8, MOPITT, OCO2, Sentinel-2 and TES. To learn more about the SkyWatch API, see <https://github.com/skywatchspaceapps/api>.

## Usage

```
querySW(api_key = NULL, time_period, longitude_latitude, instrument_satellite = NULL,
        data_level = NULL, max_resolution = NULL, max_cloudcover = NULL,
        wavelength_band = NULL, output = "data.frame")
```

## Arguments

api_key	string; personal alphanumeric API key. See Details section
time_period	string; one or two UTC timestamps in ISO format (yyyy-mm-ddThh:mm:ss.sssst+/-zzzz). See Details section
longitude_latitude	string or an Spatial object. See Details section
instrument_satellite	string; source of the data, either the instrument on-board the satellite or the satellite itself. See Details section

data_level	numeric; data processing levels for Earth observation data. See Details section
max_resolution	numeric; maximum spatial resolution (in meters). See Details section
max_cloudcover	numeric; maximum cloud cover (in percentage). See Details section
wavelength_band	string; wavelength bands for imagery (i.e. Landsat-8) and by file type for non-imagery data (e.g. Hierarchical-Data-Format). See Details section
output	string; either "data.frame" (default) or "html". "html" returns a data.frame and prints it as html.

## Details

api\_key is a personal alphanumeric API key created for and provided to a user once registered at <http://www.skywatch.co/request-access>. For a cleaner code, users can set the global option SkyWatchr.apikey (recommended) once per session via: options(SkyWatchr.apikey = 'your\_api\_key').

time\_period corresponds to one or two UTC timestamps in ISO format (yyyy-mm-ddThh:mm:ss.sssst+/-zzzz). Partial dates can also be specified: 2009, 2009-12, 2009-12-25, 2009-12-25T13:25:00.0000+0000. If only one timestamp is passed in, a one day range is assumed. For example, if 2009-12-25 is specified, the search takes place as if 2009-12-25,2009-12-26 was specified.

longitude\_latitude can be an object of class 'Spatial' (in geographic coordinates) or a string. When supplied as string, longitude\_latitude corresponds to a list of longitude, latitude coordinate pairs as a flat, comma-separated list. A list of two numbers represents a point, four numbers is a square area where the coordinates are the corners, or if there are more than four numbers the coordinates represent a closed polygon, where the first point equals the last point in the list. Because this list represents a number of points, there always has to be an even number of numbers in the list. Examples:

Point: -71.1043443253,-42.3150676016

Square: -71.1043443253471,-42.3150676015829,71.1043443253471,42.3150676015829

Polygon: -71.1043443253471,-42.3150676015829,71.1043443253471,-42.3150676015829,71.1043443253471,42.3150676015829,-71.1043443253471,42.3150676015829,-71.1043443253471,-42.3150676015829

instrument\_satellite corresponds to the source of the data, either the instrument on-board the satellite or the satellite itself. Single or multiple sources can be specified. Choice of sources are: ACOS, AIRS, CAI, FTS-SWIR, Landsat-8, MOPITT, OCO2, Sentinel-2 and TES. This field is not case-sensitive, and multiple sources can be specified (separated by commas).

data\_level corresponds to the data processing levels for Earth observation data. Level 1, 2, and 3 (L1, L2, L3) datasets are available. If no data level is specified, datasets of all levels will be returned. Only a single level can be specified. Choices are: 1, 2, and 3.

max\_resolution is only applicable to imagery that's available through the API (i.e. Landsat-8). Resolution is in meters (m). Resolutions less-than or equal-to this value will be returned. The resolution for Landsat-8 is 15 m. All climate/atmospheric datasets have a resolution of 0 m, because it is not applicable. The maximum resolution is 30 m. If resolution is omitted all imagery or data matching other search criteria will be returned.

max\_cloudcover is only applicable to imagery that's available through the API (i.e. Landsat-8). Cloud cover is given as a percentage (

wavelength\_band can be specified by the wavelength bands for imagery (i.e. Landsat-8) and by file type for non-imagery data (e.g. Hierarchical-Data-Format). Choices of bands are: Blue, Cirrus, Coastal-Aerosol, Green, Hierarchical-Data-Format, Near-Infrared, Panchromatic, Red, Short-Wave-Infrared-1, Short-Wave Infrared-2, Thermal-Infrared-1, and Thermal-Infrared-2. This field is not case-sensitive, and multiple bands can be specified (separated by commas).

**Value**

data.frame containing query output

**Author(s)**

Ali Santacruz; thanks to Joshua Kunst for kindly contributing the SkyWatchr.apikey option code

**Examples**

```
## Not run:
api_key <- "your_personal_alphanumeric_api_key"

# Set the SkyWatchr.apikey option
options(SkyWatchr.apikey = api_key)

# Query data for one of the world largest landfills "Olusosun Dump" in Nigeria
querySW(time_period = "2015-8", longitude_latitude = "3.367358,6.566358,3.387358,6.586358")

# If the SkyWatchr.apikey option is not set, then provide the API key
querySW(api_key, time_period = "2015-8", longitude_latitude = "3.367358,6.566358,3.387358,6.586358")

# An Imperial Oil refinery in Canada
res <- querySW(time_period = 2015, longitude_latitude = "-82.4169987,42.954811", data_level = 3)
View(res)

# The smoggiest city on Earth Ahvaz, Iran
querySW(time_period = "2015-06", longitude_latitude = "48.676074,31.321119", data_level = 3)

# Ahvaz, Iran in September, 2015
querySW(time_period = "2015-9", longitude_latitude = "48.676074,31.321119", data_level = 1,
        wavelength_band = "red,green,blue")

# Other examples
querySW(time_period = "2009-12-25", longitude_latitude = "-71.1043443253471,-42.3150676015829",
        data_level = 2)
querySW(time_period = "2009-12-25", longitude_latitude = "-71.1043443253471,-42.3150676015829")

querySW(time_period = "2016-07-11,2016-07-12",
        longitude_latitude = "-71.1,-42.3,71.1,-42.3,71.1,42.3,-71.1,42.3,-71.1,-42.3",
        instrument_satellite = "Landsat-8", data_level = 1, max_resolution = 30,
        max_cloudcover = 100, wavelength_band = "Blue")

# Print output data.frame as html
querySW(time_period = "2015-8", longitude_latitude = "3.367358,6.566358,3.387358,6.586358",
        output = "html")
querySW(time_period = "2016-07-11,2016-07-12",
        longitude_latitude = "-71.1,-42.3,71.1,-42.3,71.1,42.3,-71.1,42.3,-71.1,-42.3",
        instrument_satellite = "Landsat-8", data_level = 1, max_resolution = 30,
        max_cloudcover = 100, wavelength_band = "Blue", output = "html")

# Queries can also be performed using objects of class Spatial (as defined by the sp package)
# projected in geographic coordinates
ex1 <- data.frame(x = -71.1043443253, y = -42.3150676016, data = "point")
coordinates(ex1) <- ~ x + y
class(ex1)
querySW(time_period = "2015-8", longitude_latitude = ex1)
```

*querySW*

7

## End(Not run)

# Index

## \*Topic **package**

SkyWatchr-package, [2](#)

## \*Topic **spatial**

downloadSW, [2](#)

getPolygon, [3](#)

querySW, [4](#)

SkyWatchr-package, [2](#)

downloadSW, [2](#)

getPolygon, [3](#)

querySW, [4](#)

SkyWatchr (SkyWatchr-package), [2](#)

SkyWatchr-package, [2](#)