Package: BatchPlanet (via r-universe)

November 2, 2026

Title Batch access and processing of PlanetScope imagery for spatiotemporal analysis in R

Version 1.0.0

Date 2025-10-08

Description The `BatchPlanet` R package provides a reproducible and scalable workflow for accessing and processing PlanetScope satellite imagery, enabling environmental researchers to efficiently perform spatiotemporal analysis of high-resolution remote sensing data. The package streamlines the steps required to work with the Planet API, including the ordering and downloading of imagery, retrieving and cleaning pixel-level time series, and computing derived metrics such as the Enhanced Vegetation Index (EVI) and green-up/down time. This package has supported peer-reviewed research in predicting reproductive phenology in wind-pollinated trees (Song et al., 2025). Generalizable beyond phenological research, this tool is particularly suited for environmental research that involves large volumes of PlanetScope imagery across multiple sites and long time periods.

URL https://yiluansong.r-universe.dev/BatchPlanet, https://zhulabgroup.github.io/BatchPlanet/

BugReports https://github.com/zhulabgroup/BatchPlanet/issues

License file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.1),

Imports ggplot2, tidyr, dplyr, stringr, readr, tibble, rlang, terra, sf, foreach, doSNOW, parallel, dotenv, ptw, imputeTS, segmented, jsonlite, lubridate, httr, plotly, leaflet, shiny, shinyWidgets

Suggests knitr, rmarkdown, testthat, withr, covr

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-04-29 21:19:09 UTC

RemoteUrl https://github.com/zhulabgroup/BatchPlanet

RemoteRef release

RemoteSha e5e2a0336fe6576248189c3ed31360b0e13667d2

Contents

calculate_season_metrics . . . 3
calculate_season_metrics_batch . . . 4
clean_planetscope_time_series . . . 6
clean_planetscope_time_series_batch . . . 7
determine_seasonality . . . 8
download_planetscope_imagery . . . 9
download_planetscope_imagery_batch . . . 10
download_sample_data . . . 11
order_planetscope_imagery . . . 12
order_planetscope_imagery_batch . . . 13
read_data_product . . . 14
retrieve_planetscope_time_series . . . 15
retrieve_planetscope_time_series_batch . . . 16
search_planetscope_imagery . . . 17
set_api_key . . . 18
set_bbox . . . 19
set_planetscope_parameters . . . 20
set_thresholds . . . 21
visualize_coordinates . . . 22
visualize_time_series . . . 23
visualize_true_color_imagery . . . 24
visualize_true_color_imagery_batch . . . 25
whittaker_smoothing_filling . . . 26

Index **27**

`calculate_season_metrics` *Calculate start/end of season metrics for a single time series*

Description

Processes a remote sensing index time series for a single location in one year by gap-filling and smoothing (using Whittaker smoothing), testing for seasonality, and calculating day-of-year (DOY) when the index first goes above or goes below threshold(s) specified in `df_thres`. The input time series should ideally be extended to include the end of the previous year and the beginning of the next year to capture early- and late-year events.

Usage

```
calculate_season_metrics(
  df_index,
  df_thres,
  min_days,
  check_seasonality = T,
  var_index = "evi"
)
```

Arguments

`df_index`	Data frame of remote sensing index time series at one location in one year. Must contain columns doy and the index of interest.
`df_thres`	Data frame containing candidate threshold values, with columns `direction` ("up"/"down") and `threshold` (numeric 0–1).
`min_days`	Numeric. Minimum required number of non-NA data points in one year (default: 80).
`check_seasonality`	Logical. If `TRUE`, tests for significant seasonal changes before calculating start/end of season metrics (default: `TRUE`).
`var_index`	Character. Name of the index column in `df_index` to analyze (default: "evi").

Value

A data frame of the timing (DOY) of threshold-crossing events, or `NULL` if there are fewer than `min_days` valid data points or if the index does not show a seasonal pattern.

Examples

```
## Not run:
# Example: Calculate start of season metrics for a single cleaned time series
df_thres <- set_thresholds(thres_up = c(0.3, 0.5), thres_down = NULL)
df_metrics <- calculate_season_metrics(
  df_index = df_clean,
```

```
  df_thres = df_thres,
  min_days = 80,
  check_seasonality = TRUE,
  var_index = "evi"
)

## End(Not run)
```

`calculate_season_metrics_batch` *Calculate start/end of season metrics for time series from multiple sites and groups*

Description

Reads remote sensing index time series files from the `clean/` subdirectory within the specified `dir`. For each site, group, and year combination, the function extends the time series to include the end of the previous year and the beginning of the next year (to capture early- and late-year events), then calls `calculate_season_metrics()` to calculate start/end of season metrics for each time series, which are measured by the day-of-year (DOY) when the index first goes above or goes below specified threshold(s). Results are saved as `.rds` files under the `doy/` subdirectory within `dir`, prefixed `doy_`.

Usage

```
calculate_season_metrics_batch(
  dir,
  v_site = NULL,
  v_group = NULL,
  v_year = NULL,
  df_thres = NULL,
  var_index = "evi",
  min_days = 80,
  check_seasonality = T,
  extend_to_previous_year = 275,
  extend_to_next_year = 90,
  num_cores = 3
)
```

Arguments

`dir`	Character. Base directory containing remote sensing index files (expects `clean/` subdirectory).
`v_site`	Character vector, optional. Site identifiers to process; if `NULL`, all sites are included.

v_group — Character vector, optional. Group identifiers to process; if NULL, all groups are included.

v_year — Integer vector, optional. Years to process; if NULL, all years in the data are included.

df_thres — Data frame of thresholds as from set_thresholds(); if NULL, uses default thresholds.

var_index — Character. Name of the remote sensing index column in the input data frame df_index to analyze (default: "evi").

min_days — Numeric. Minimum required number of non-NA data points in one year (default: 80).

check_seasonality — Logical. If TRUE, tests for significant seasonal changes before calculating start/end of season metrics (default: TRUE).

extend_to_previous_year — Integer. Day of year to extend backward to, to capture early-year events (default: 275).

extend_to_next_year — Integer. Day of year to extend forward to, to capture late-year events (default: 90).

num_cores — Integer. Number of parallel workers for processing (default: 3).

Value

Invisibly returns NULL and saves calculated start/end of season metrics as .rds files in the doy/ subdirectory of dir.

Examples

```
## Not run:
# Example: Calculate start of season metrics for two sites and two groups
df_thres <- set_thresholds(thres_up = c(0.3, 0.5))
calculate_season_metrics_batch(
  dir = "alldata/PSdata/",
  v_site = c("Site1", "Site2"),
  v_group = c("Group1", "Group2"),
  v_year = c(2023, 2024),
  df_thres = df_thres,
  var_index = "evi",
  min_days = 80,
  check_seasonality = TRUE,
  extend_to_previous_year = 275,
  extend_to_next_year = 90,
  num_cores = 3
)

## End(Not run)
```

clean_planetscope_time_series *Clean a single PlanetScope time series*

Description

Cleans a single time series data frame by removing low-quality data and optionally calculating NDVI and/or EVI. Low-quality data are defined as:

- The sun elevation angle is less than 0 degrees (i.e., night time images).
- The reflectance values for any band are less than 0.
- The pixel was cloudy, had snow, ice, shadow, haze, or cloud.
- The usable data mask had algorithmic confidence in classification less than 80% for the pixel.

Usage

```
clean_planetscope_time_series(
  df_ts,
  calculate_index = c("ndvi", "evi"),
  filter_range = list(ndvi = c(-1, 1), evi = c(0, 1))
)
```

Arguments

df_ts	Data frame. Raw time series data for a single site/group.
calculate_index	Character vector. Specifies which vegetation indices to calculate. Supported options are NDVI and EVI. Inputs are case-insensitive. If NULL, no indices are calculated. (default: c("ndvi", "evi")).
filter_range	List. A named list specifying the valid range for indices. To disable filtering for an index, set it to NULL or omit it. (default: list(ndvi = c(-1, 1), evi = c(0, 1))).

Value

Data frame of cleaned time series, with NDVI and/or EVI if requested.

Examples

```
## Not run:
df_clean <- clean_planetscope_time_series(
  df_ts = df_ts_example,
  calculate_index = c("ndvi", "evi"),
  filter_range = list(ndvi = c(-1, 1), evi = c(0, 1))
)

## End(Not run)
```

clean_planetscope_time_series_batch *Clean PlanetScope time series for multiple sites and groups*

Description

Cleans raw time series data for all sites and groups, removing low-quality data and optionally calculating NDVI and/or EVI. Low-quality data are defined as:

- The sun elevation angle is less than 0 degrees (i.e., night time images).
- The reflectance values for any band are less than 0.
- The pixel was cloudy, had snow, ice, shadow, haze, or cloud.
- The usable data mask had algorithmic confidence in classification less than 80% for the pixel.

Results are saved as `.rds` files under the `clean/` subdirectory within `dir`, prefixed `clean_`.

Usage

```
clean_planetscope_time_series_batch(
  dir,
  v_site = NULL,
  v_group = NULL,
  num_cores = 3,
  calculate_index = c("ndvi", "evi"),
  filter_range = list(ndvi = c(-1, 1), evi = c(0, 1))
)
```

Arguments

`dir`	Character. Base directory containing raw time series files (expects a `ts/` subdirectory).
`v_site`	Character vector, optional. Site identifiers to process; if `NULL`, all sites in `ts/` are included.
`v_group`	Character vector, optional. Group identifiers to process; if `NULL`, all groups in filenames are included.
`num_cores`	Integer. Number of parallel workers to use (default: 3).
`calculate_index`	Character vector. Specifies which vegetation indices to calculate. Supported options are NDVI and EVI. Inputs are case-insensitive. If `NULL`, no indices are calculated. (default: `c("ndvi", "evi")`).
`filter_range`	List. A named list specifying the valid range for indices. To disable filtering for an index, set it to `NULL` or omit it. (default: `list(ndvi = c(-1, 1), evi = c(0, 1))`).

Value

Invisibly returns `NULL` and saves cleaned time series as `.rds` files in the `clean/` subdirectory of `dir`.

Examples

```
## Not run:
clean_planetscope_time_series_batch(
  dir = "alldata/PSdata/",
  v_site = c("HARV", "SJER"),
  v_group = c("Acer", "Quercus"),
  num_cores = 3,
  calculate_index = c("ndvi", "evi"),
  filter_range = list(ndvi = c(-1, 1), evi = c(0, 1))
)

## End(Not run)
```

`determine_seasonality` *Test for seasonality in a time series Using AIC-based model selection*

Description

Fits a simple linear regression model and segmented regression models (with 1 to 3 breakpoints) to a numeric time series. Compares their Akaike Information Criterion (AIC) values, with a penalty parameter `k`, to determine if the time series is best described as flat (no seasonality) or as having significant seasonal changes

Usage

```
determine_seasonality(ts, doy = 1:length(ts), k = 50)
```

Arguments

`ts`	Numeric vector. The time series signal to evaluate (e.g., EVI, NDVI, or other index).
`doy`	Numeric vector. The day-of-year indices corresponding to the time series. Default is `1:length(ts)`. If your `ts` is not daily, you need to provide the day-of-year indices.
`k`	Numeric. Penalty parameter for the AIC calculation. Higher values make the function more conservative in detecting seasonality. Default is 50.

Value

Logical. Returns `TRUE` if the time series is seasonal (segmented model preferred), or `FALSE` if the time series is flat (linear model preferred).

Examples

```
## Not run:
# Simulate a seasonal time series with noise and missing values
t <- 1:365
simulate_ts <- sin(2 * pi * t / 365) + rnorm(365, sd = 0.1)
simulate_ts[sample(1:365, 30)] <- NA # introduce some missing data

# Test for seasonality
determine_seasonality(ts = simulate_ts, k = 50)

# Example with a flat (non-seasonal) time series
flat_ts <- rnorm(365, mean = 0.5, sd = 0.05)
determine_seasonality(ts = flat_ts)

## End(Not run)
```

---

download_planetscope_imagery

Download a PlanetScope order

Description

Downloads all files for a given PlanetScope order ID, saving them to the specified folder. The function will wait until the order is successfully processed by Planet API, partially processed, fail to be processed, or is cancelled.

Usage

```
download_planetscope_imagery(
  order_id,
  exportfolder,
  api_key,
  overwrite = FALSE
)
```

Arguments

`order_id`	Character. The PlanetScope order ID.
`exportfolder`	Character. Directory in which to save downloaded files (created if needed).
`api_key`	Character. Your Planet API key.
`overwrite`	Logical. If `TRUE`, existing files will be overwritten (default: `FALSE`).

Value

Invisibly returns `NULL` and writes all imagery files to the specified `exportfolder`.

Examples

```
## Not run:
download_planetscope_imagery(
  order_id = "abc123-order-id",
  exportfolder = "data/raw/SJER/SJER_202405_121_151",
  api_key = set_api_key(),
  overwrite = TRUE
)

## End(Not run)
```

`download_planetscope_imagery_batch`
Download PlanetScope imagery for multiple sites and years

Description

Downloads all ordered PlanetScope imagery for specified sites and years. The function will wait until the order is successfully processed by Planet API, and then download the files.

Usage

```
download_planetscope_imagery_batch(
  dir,
  v_site = NULL,
  v_year = 2017:(as.integer(format(Sys.Date(), "%Y"))),
  v_month = 1:12,
  setting,
  num_cores = 12,
  overwrite = F
)
```

Arguments

`dir`	Character. Base directory where the raw satellite data will be stored, expects a `raw/` subdirectory.
`v_site`	Character vector, optional. Site names to filter; if `NULL`, all sites are used.
`v_year`	Numeric vector. Years for which to download data. Default is from 2017 to the current year.
`v_month`	Integer vector. Months (1-12) to download (default: 1:12, i.e., all months).
`setting`	List. Contains API settings including the API key (i.e., `setting$api_key`).
`num_cores`	Integer. Number of parallel workers to use (default: 12).
`overwrite`	Logical. If `TRUE`, existing files will be overwritten (default: `FALSE`).

Value

Invisibly returns NULL and saves downloaded imagery to the raw/ subdirectory of the specified dir.

Examples

```
## Not run:
download_planetscope_imagery_batch(
  dir = "alldata/PSdata/",
  v_site = c("HARV", "SJER"),
  v_year = 2025,
  v_month = 4:6,
  setting = setting,
  num_cores = 3,
  overwrite = FALSE
)

## End(Not run)
```

---

download_sample_data *Download sample data from GitHub*

Description

This function clones the sample-data branch of the BatchPlanet GitHub repository into the current working directory.

Usage

```
download_sample_data()
```

Details

Requires that git is installed and available on your system PATH. If the sample-data folder already exists, the function will not overwrite it.

Value

Path where the data was downloaded to.

`order_planetscope_imagery` *Place a PlanetScope order*

Description

Submits an order to Planet API, with specified image IDs and tools.

Usage

```
order_planetscope_imagery(
  api_key,
  bbox,
  items,
  item_name,
  product_bundle,
  harmonized,
  order_name,
  mostrecent = 0
)
```

Arguments

`api_key`	Character. Your Planet API key.
`bbox`	An `sf`-style bounding box (a named numeric list with `xmin`, `ymin`, `xmax`, `ymax`).
`items`	Character vector. Planet image IDs to include in this order.
`item_name`	Character. Item type (e.g. `"PSScene"`).
`product_bundle`	Character. Product bundle (e.g. `"analytic_sr_udm2"`).
`harmonized`	Logical. If `TRUE`, applies harmonization tool in the order.
`order_name`	Character. A unique name for this order.
`mostrecent`	Integer. If >0, only the most recent N images are ordered (default: 0 = all).

Value

Character. The ID of the order.

Examples

```
## Not run:
order_id <- order_planetscope_imagery(
  api_key = set_api_key(),
  bbox = set_bbox(df_coordinates, "SJER"),
  items = ids,
  item_name = "PSScene",
  product_bundle = "analytic_sr_udm2",
  harmonized = TRUE,
```

```
  order_name = "SJER_20240501_20240531",
  mostrecent = 0
)

## End(Not run)
```

`order_planetscope_imagery_batch` *Order PlanetScope imagery for multiple sites and years*

Description

Orders PlanetScope imagery for all sites and years specified in a coordinate data frame. Saves order IDs and metadata for each site and year.

Usage

```
order_planetscope_imagery_batch(
  dir,
  df_coordinates,
  v_site = NULL,
  v_year = 2017:(as.integer(format(Sys.Date(), "%Y"))),
  v_month = 1:12,
  setting
)
```

Arguments

`dir`	Character. Base directory for saving orders, expects a `raw/` subdirectory.
`df_coordinates`	Data frame of coordinates of interest with columns `site`, `lon`, `lat`, and `id`.
`v_site`	Character vector, optional. Site names to process; if `NULL`, all unique sites in `df_coordinates` are used.
`v_year`	Numeric vector. Years to process (default: 2017 to current year).
`v_month`	Integer vector. Months (1-12) to process (default: 1:12, i.e., all months).
`setting`	List. API settings including API key, item name, asset type, cloud limit, product bundle, and harmonized flag.

Value

Invisibly returns `NULL` and writes order metadata as `.rds` files under `orders/` subdirectory of specified `dir`.

Examples

```
## Not run:
order_planetscope_imagery_batch(
  dir = "alldata/PSdata/",
  df_coordinates = df_coordinates,
  v_site = c("HARV", "SJER"),
  v_year = 2025,
  v_month = 4:6,
  setting = setting
)

## End(Not run)
```

`read_data_product` *Read and combine processed data products*

Description

Reads and combines data files processed with this package(e.g., time series, cleaned remote sensing index, or phenological metrics). Supports reading a subset of data from specified sites and groups. Three data products are supported:

- "ts": raw time series data
- "clean": cleaned time series data, with optionally calculated Enhanced Vegetation Index (EVI)
- "doy": extracted phenological metrics

Usage

```
read_data_product(dir, v_site = NULL, v_group = NULL, product_type = "clean")
```

Arguments

`dir`	Character. Base directory containing processed data product files (e.g., the parent directory of "ts/", "clean/", or "doy/").
`v_site`	Character vector, optional. Site identifiers to include; if `NULL`, all sites are included.
`v_group`	Character vector, optional. Group identifiers to include; if `NULL`, all groups are included.
`product_type`	Character. Type of product to read (e.g., "ts", "clean", or "doy"). Default is "clean".

Value

A data frame combining all matching data product files, with columns for site and group.

Examples

```
## Not run:
# Example: Read all cleaned time series for two sites and one group
df_clean <- read_data_product(
  dir = "alldata/PSdata/",
  v_site = c("HARV", "SJER"),
  v_group = "Quercus",
  product_type = "clean"
)

# Example: Read all phenological metrics for a site
df_doy <- read_data_product(
  dir = "alldata/PSdata/",
  v_site = "SJER",
  product_type = "doy"
)

## End(Not run)
```

`retrieve_planetscope_time_series` *Retrieve a single set of PlanetScope time series*

Description

Extracts and combines reflectance, QA, and metadata for a set of spatial points.

Usage

```
retrieve_planetscope_time_series(dir_site, sf_coordinates, num_cores = 12)
```

Arguments

`dir_site`	Character. Path to the site-specific raw data directory.
`sf_coordinates`	An `sf` object with point coordinates (must have an `id` column).
`num_cores`	Integer. Number of parallel workers for processing (default: 12).

Value

Data frame with columns for point ID, coordinates, reflectance bands, QA, and metadata.

Examples

```
## Not run:
df_coordinates_example <- df_coordinates |> dplyr::filter(site == "SJER", group == "Quercus")
df_ts_example <- retrieve_planetscope_time_series(
  dir_site = file.path("alldata/PSdata/raw", "SJER"),
```

```
  sf_coordinates = sf::st_as_sf(df_coordinates_example, coords = c("lon", "lat"), crs = 4326),
  num_cores = 12
)

## End(Not run)
```

`retrieve_planetscope_time_series_batch` *Retrieve PlanetScope time series for multiple sites and groups*

Description

Extracts time series data from PlanetScope imagery for all points in a coordinate data frame, grouped by site and group. For each site and group, reflectance, QA, and metadata are extracted and combined Results are saved as `.rds` files under the `ts/` subdirectory of the specified `dir`, prefixed `ts_`.

Usage

```
retrieve_planetscope_time_series_batch(
  dir,
  df_coordinates,
  v_site = NULL,
  v_group = NULL,
  max_sample = 2000,
  num_cores = 12
)
```

Arguments

`dir`	Character. Base directory containing raw PlanetScope data (expects a `raw/` subdirectory).
`df_coordinates`	Data frame with columns `site`, `lon`, `lat`, `id`, and optionally `group`.
`v_site`	Character vector, optional. Site identifiers to process; if `NULL`, all sites in `df_coordinates` are included.
`v_group`	Character vector, optional. Group identifiers to process; if `NULL`, all groups in `df_coordinates` are included.
`max_sample`	Integer. Maximum number of samples per group (default: 2000). This is to avoid processing too many points at once, which can be slow.
`num_cores`	Integer. Number of parallel workers for processing (default: 12).

Value

Invisibly returns `NULL` and saves one `.rds` file per site-group to the `ts/` subdirectory in the specified `dir`.

Examples

```
## Not run:
retrieve_planetscope_time_series_batch(
  dir = "alldata/PSdata/",
  df_coordinates = df_coordinates,
  v_site = c("HARV", "SJER"),
  v_group = c("Acer", "Quercus"),
  max_sample = 2000,
  num_cores = 12
)

## End(Not run)
```

`search_planetscope_imagery` *Search for available PlanetScope imagery*

Description

Queries the Planet API for imagery overlapping the specified bounding box and date range, then filters results by cloud cover, ground control, and quality. Returns only IDs for which you have download permission.

Usage

```
search_planetscope_imagery(
  api_key,
  bbox,
  date_end = NULL,
  date_start = NULL,
  item_name = "PSScene",
  asset = "ortho_analytic_4b_sr",
  cloud_lim = 0.1,
  ground_control = TRUE,
  quality = "standard"
)
```

Arguments

`api_key`	Character. Your Planet API key.
`bbox`	Named numeric list with `xmin`, `ymin`, `xmax`, `ymax` defining the search area.
`date_end`	Character. End date ("YYYY-MM-DD") for the search (inclusive).
`date_start`	Character. Start date ("YYYY-MM-DD") for the search (inclusive).
`item_name`	Character. Planet item type to search (default: `"PSScene"`).
`asset`	Character. Asset type to filter permissions (default: `"ortho_analytic_4b_sr"`).

cloud_lim	Numeric. Maximum allowed cloud cover fraction (0–1, default: 0.1).
ground_control	Logical. If TRUE, require ground control metadata (default: TRUE).
quality	Character. Quality category filter (default: "standard").

Value

Character vector of image IDs with download permission. Returns NULL if none are found.

Examples

```
## Not run:
ids <- search_planetscope_imagery(
  api_key = set_api_key(),
  bbox = my_bbox,
  date_start = "2023-06-01",
  date_end = "2023-06-30",
  cloud_lim = 1,
  ground_control = TRUE,
  quality = "standard"
)

## End(Not run)
```

---

set_api_key *Set or change the Planet API key*

Description

Prompts the user to enter a Planet API key and saves it in a hidden .env file in the working directory.

Usage

```
set_api_key(change_key = F)
```

Arguments

change_key	Logical. If TRUE, prompts for a new API key even if one already exists (default: FALSE).

Value

Invisibly returns the API key.

Note

You will need an active Planet account and an API key to access the PlanetScope API. You can sign up for an account on the Planet website. Once you have an account, you can copy your API key from your account settings.

Examples

```
## Not run:
set_api_key() # Set the API key for the first time
set_api_key(change_key = TRUE) # Change the API key

## End(Not run)
```

`set_bbox` *Generate a Bounding Box from Coordinate Data*

Description

Creates a bounding box for a specified location based on a data frame of coordinates. The function filters the data for the given location, removes any rows with missing longitude or latitude values, and computes the minimum and maximum coordinate values. A buffer is applied to expand the bounding box by a specified distance (in degrees).

Usage

```
set_bbox(df_coordinates, siteoi, buffer = 5e-04)
```

Arguments

`df_coordinates`	A data frame containing coordinate data. Must include the columns `site`, `lon`, and `lat`.
`siteoi`	Character. The site identifier (i.e., the value in the `site` column) for which to generate the bounding box.
`buffer`	Numeric. A buffer distance (in degrees) to expand the bounding box in all four directions. Default is 0.0005.

Value

An object of class `bbox` (from the **sf** package) representing the bounding box for the specified location, or `NULL` if no valid coordinates are found.

Examples

```
## Not run:
df_coords <- data.frame(
  site = c("SiteA", "SiteA", "SiteB"),
  lon = c(-77.05, -77.00, -76.95),
  lat = c(38.80, 38.85, 38.90)
)
bbox <- set_bbox(df_coords, "SiteA", buffer = 0.001)

## End(Not run)
```

`set_planetscope_parameters` *Set PlanetScope API Parameters*

Description

Constructs a named list of parameters required for interacting with the PlanetScope API.

Usage

```
set_planetscope_parameters(
  api_key,
  item_name = "PSScene",
  asset = "ortho_analytic_4b_sr",
  product_bundle = "analytic_sr_udm2",
  cloud_lim = 1,
  harmonized = TRUE
)
```

Arguments

`api_key`	Character. API key for authentication. We recommend using `set_api_key()` to save your API key in a hidden .env file.
`item_name`	Character. Name of the satellite data product (default: `"PSScene"`).
`asset`	Character. Type of asset to retrieve (default: `"ortho_analytic_4b_sr"`).
`product_bundle`	Character. Product bundle selection (default: `"analytic_sr_udm2"`).
`cloud_lim`	Numeric. Cloud coverage limit (between 0 and 1). Images with a cloud coverage above this fraction will be ignored. (default: 1).
`harmonized`	Logical. Indicates whether to use the Planset API tool to harmonize data with Sentinel-2 (default: `TRUE`).

Details

You will need an active Planet account and an API key to access the PlanetScope API. You can sign up for an account on the Planet website. Once you have an account, you can copy your API key from your account settings.

An **item_type** represents an imagery product. The default item type is "PSScene", which is PlanetScope 3, 4, and 8 band scenes captured by the Dove satellite constellation. Other item types include "TanagerScene", "TanagerMethane", "REOrthoTile", "REScene", "SkySatScene", "SkySatCollect", and "SkySatVideo". Please refer to the Planet catalog for the most updated available item types.

An **asset** is a product derived from the item's source data, and can be used for various analytic, visual or other purposes. For example, a full list of available assets for "PSScene" can be found here. Please visit the Planet catalog for a full list of available assets for each item type.

A **product bundle** comprises of a group of assets for an item. This is often useful when you want to download associated metadata and data quality masks. A full list of bundles by item type can be found here.

The `cloud_lim` parameter is set to 1 by default, which means we will download all images regardless of cloud coverage. You can set this parameter to a lower value (e.g., 0.5) to filter out images with more than 50% cloud coverage.

PlanetScope API allow users to apply a tool named "harmonize" that applies scene-level normalization and harmonization, such that all PlanetScope data were consistent and approximately comparable to data from Sentinel-2. This tool is useful when integrating or comparing images from different times and locations. Refer to PlanetScope technical documentation for more details.

Value

A named list containing the PlanetScope API parameters.

Examples

```
## Not run:
setting <- set_planetscope_parameters(
  api_key = set_api_key(),
  item_name = "PSScene",
  asset = "ortho_analytic_4b_sr",
  product_bundle = "analytic_sr_udm2",
  cloud_lim = 1,
  harmonized = TRUE
)

## End(Not run)
```

---

set_thresholds *Specify thresholds for detecting start/end of season*

Description

Creates a data frame with threshold values for both increasing ("up") and decreasing ("down") trends. These values are used to detect threshold-crossing events in the time series that indicate the start/end of season.

Usage

```
set_thresholds(
  thres_up = round(seq(from = 0, to = 1, by = 0.1), 1),
  thres_down = round(seq(from = 1, to = 0, by = -0.1), 1)
)
```

Arguments

thres_up	Numeric vector. Threshold values for increasing trends (default: seq(from = 0, to = 1, by = 0.1) |> round(1)).
thres_down	Numeric vector. Threshold values for decreasing trends (default: seq(from = 1, to = 0, by = -0.1) |> round(1)).

Value

A data frame with two columns:

direction	A character vector indicating "up" for increasing trends or "down" for decreasing trends.
threshold	A numeric vector containing threshold values between 0 and 1.

Examples

```
# Get default thresholds (for increasing and decreasing trends)
default_thres <- set_thresholds()

# Get thresholds with custom rule
custom_thres <- set_thresholds(thres_up = c(0.3, 0.5), thres_down = NULL)
```

---

visualize_coordinates *Visualize coordinates from a data frame*

Description

Creates a scatter plot of coordinate data.

Usage

```
visualize_coordinates(df_coordinates)
```

Arguments

df_coordinates	Data frame containing longitude and latitude columns.

Value

An interactive `leaflet` map object when supported, a static `ggplot` object otherwise.

Examples

```
## Not run:
visualize_coordinates(df_coordinates)

## End(Not run)
```

visualize_time_series *Visualize time series interactively*

Description

Generates an interactive plotly time series plot from a data frame, optionally overlaying phenological events.

Usage

```
visualize_time_series(
  df_ts,
  df_doy = NULL,
  var = "value",
  ylab = "Value",
  smooth = FALSE,
  lambda = 50,
  facet_var = NULL,
  color_palette = "viridis"
)
```

Arguments

df_ts	Data frame. Must contain either a time (POSIX) or date (Date) column, an id column, and the variable to plot.
df_doy	Optional data frame. Time of phenological events with columns year and doy. Default: NULL.
var	Character. Name of the column in df_ts to plot (default: "value").
ylab	Character. Label for the y-axis (default: "Value").
smooth	Logical. If TRUE, applies gap-filling and smoothing to data points with Whittaker smoothing (default: FALSE).
lambda	Numeric. Smoothing parameter for Whittaker smoothing (default: 50).
facet_var	Character or NULL. Column name in df_ts and df_doy to facet by (e.g., "site" or "id").
color_palette	Character. Name of a viridis palette for line colors (default: "viridis").

Value

An interactive plotly object when supported, a static ggplot object otherwise.

Examples

```
## Not run:
# Example: Visualize a time series with phenological events
visualize_time_series(
```

```
  df_ts = df_ts,
  df_doy = df_doy,
  var = "value",
  ylab = "Value",
  smooth = TRUE,
  lambda = 50,
  facet_var = "id",
  color_palette = "viridis"
)

## End(Not run)
```

`visualize_true_color_imagery` *Visualize true-color raster imagery*

Description

Reads a multi-band raster, converts the red, green, and blue bands to normalized RGB values, and creates a ggplot2 tile plot of the true-color composite. Optionally overlays point locations. Automatically normalizes image brightness to the global average (computed at app startup) for legibility, then applies the user brightness slider (0-10, default 5) as a multiplier.

Usage

```
visualize_true_color_imagery(
  file,
  df_coordinates = NULL,
  brightness = 5,
  global_brightness = 0.05
)
```

Arguments

`file`	Character. Path to a multi-band raster file (e.g., PlanetScope SR clip).
`df_coordinates`	Optional data frame. Point locations to overlay; must contain `lon`, `lat`, and `site`. Default: `NULL`.
`brightness`	Numeric. Brightness multiplier for RGB values (slider value, default: 5).
`global_brightness`	Numeric. The global average brightness (from sampled images at app startup).

Value

A ggplot2 object.

Examples

```
## Not run:
visualize_true_color_imagery(
  file = "alldata/PSdata/raw/SJER/20240501_SR_harmonized_clip.tif",
  df_coordinates = df_coordinates_SJER,
  brightness = 5,
  global_brightness = 0.05
)

## End(Not run)
```

`visualize_true_color_imagery_batch`
Launch a Shiny App to Visualize True-Color Imagery

Description

Starts an interactive Shiny app for browsing and visualizing true-color PlanetScope imagery in a directory.

Usage

```
visualize_true_color_imagery_batch(dir, df_coordinates = NULL, cloud_lim = 1)
```

Arguments

`dir`	Character. Base directory containing raw imagery (expects a `raw/` subdirectory).
`df_coordinates`	Optional data frame. Point locations to overlay; must contain `lon`, `lat`, and `site`. Default: `NULL`.
`cloud_lim`	Numeric. Maximum allowed cloud cover (0-1, default: 1). Only images with cloud cover <= cloud_lim are shown.

Value

None. Launches a Shiny app in the default web browser.

Examples

```
## Not run:
visualize_true_color_imagery_batch(
  dir = "alldata/PSdata/",
  df_coordinates = df_coordinates,
  cloud_lim = 0.1
)

## End(Not run)
```

`whittaker_smoothing_filling` *Smooth and gap-fill a time series using Whittaker smoothing*

Description

Applies weighted Whittaker smoothing to a numeric time series, filling short gaps (NAs) and smoothing the signal at the same time. This function is useful for environmental time series with missing or noisy data. Note that the input time series need to be sampled at a regular interval. If not, you need to resample it first, with NAs inserted at the missing time points.

Usage

```
whittaker_smoothing_filling(x, lambda, maxgap = Inf, minseg = 2)
```

Arguments

`x`	Numeric vector. The time series signal to be smoothed (e.g., EVI, NDVI, or other index).
`lambda`	Numeric. Smoothing parameter; larger values result in a smoother output.
`maxgap`	Numeric. Maximum number of consecutive NAs to interpolate (default: Inf). Gaps longer than this will remain NA.
`minseg`	Numeric. Minimum segment length for smoothing (default: 2). Segments shorter than this will be replaced with NA.

Value

Numeric vector. The smoothed and gap-filled signal, with the same length as `x`.

Examples

```
## Not run:
# Simulate a noisy, gappy time series
set.seed(42)
t <- 1:365
x <- sin(2 * pi * t / 365) + rnorm(365, sd = 0.1)
x[sample(1:365, 30)] <- NA

# Smooth and fill gaps
x_sm <- whittaker_smoothing_filling(x, lambda = 50, maxgap = 14, minseg = 2)
plot(t, x, type = "p", col = "grey", main = "Whittaker Smoothing")
lines(t, x_sm, col = "darkgreen", lwd = 2)

## End(Not run)
```

Index

calculate_season_metrics, 3
calculate_season_metrics_batch, 4
clean_planetscope_time_series, 6
clean_planetscope_time_series_batch, 7

determine_seasonality, 8
download_planetscope_imagery, 9
download_planetscope_imagery_batch, 10
download_sample_data, 11

order_planetscope_imagery, 12
order_planetscope_imagery_batch, 13

read_data_product, 14
retrieve_planetscope_time_series, 15
retrieve_planetscope_time_series_batch, 16

search_planetscope_imagery, 17
set_api_key, 18, *20*
set_bbox, 19
set_planetscope_parameters, 20
set_thresholds, 21
set_thresholds(), *5*

visualize_coordinates, 22
visualize_time_series, 23
visualize_true_color_imagery, 24
visualize_true_color_imagery_batch, 25

whittaker_smoothing_filling, 26