Package 'LandsatTS'

January 9, 2023

Title An R package to facilitate retrieval, cleaning, cross-calibration, and phenological modeling of Landsat time-series data

Version 1.1.0

Description This software package facilitates sample-based time series analysis of surface reflectance and spectral indices derived from sensors on the Landsat satellites. The package includes functions that enable extraction of the full Landsat record for point sample locations or small study regions using Google Earth Engine directly accessed from R. Moreover, the package includes functions for (1) data cleaning, (2) cross-sensor calibration, (3) phenological modeling, and (4) time series analysis.

License Modified MIT

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports magrittr, dplyr, tidyr, rgee, sf, crayon, mapview, purrr, data.table, ggplot2, R.utils, stats, stringr, ggpubr, ranger, zoo, zyp

Depends R (>= 3.50)

R topics documented:

`lsat.example.dt` *Landsat surface reflectance for six sample sites in the Arctic*

Description

A dataset containing Landsat surface reflectance measurements and ancillary data for six sample sites in the Arctic. These data are used for the examples included with lsatTS.

Usage

```
lsat.example.dt
```

Format

A data.table with 5296 rows and 23 variables

Source

Generated using example code provided in lsatTS::lsat_export_ts()

`lsat_calc_spectral_index` *Calculate spectral indices*

Description

This function computes some widely used spectral vegetation indices. Only one index can be computed at a time. Current indices include the: Normalized Difference Vegetation Index (NDVI; Rouse et al. 1974), kernel NDVI (kNDVI; Camp-Valls et al. 2020), Green NDVI (gNDVI; Gitelson and Merzlyak 1998), Soil Adjusted Vegetation Index (SAVI; Huete 1998), Wide Dynamic Range Vegetation Index (WDRVI; Gitelson 2004), Enhanced Vegetation Index (EVI; Huete et al. 2002), 2-band EVI (EVI2; Jiang et al. 2008), Near Infrared Vegetation Index (NIRv; Badgley et al. 2017), Moisture Stress Index (MSI; Rock et al. 1986), Normalized Difference Water Index (NDWI; McFeeters 1996), Normalized Difference Moisture Index (NDMI; Gao 1996), Normalized Burn Ratio (NBR, Key and Benson 1999), Normalized Difference Infrared Index (NDII; Hardisky et al. 1983), Plant Senescence Reflectance Index (PSRI; Merzlyak et al. 1999), and the Soil-Adjusted Total Vegetation Index (SATVI; Marsett et al. 2006).

Usage

```
lsat_calc_spectral_index(dt, si)
```

Arguments

`dt`	Data.table containing surface reflectance data.
`si`	Character string specifying abbreviation of the desired spectral index.

Value

The input data.table with an appended column containing the spectral index.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
lsat.dt
```

---

lsat_calc_trend *Calculate non-parametric vegetation greenness trends*

Description

This function evaluates and summarizes interannual trends in vegetation greenness for sample sites over a user-specified time period. Potential interannual trends in vegetation greenness are assessed using Mann-Kendall trend tests and Theil-Sen slope indicators after prewhitening each time series. This trend assessment relies on the zyp.yuepilon() function from the zyp package, which provides further details.

Usage

```
lsat_calc_trend(dt, si, yrs, yr.tolerance = 1, nyr.min.frac = 0.66, sig = 0.1)
```

Arguments

dt	Data.table with columns including site, year, and the vegetation index of interest.
si	Spectral index for which to assess trend (e.g., NDVI).
yrs	A sequence of years over which to assess trends (e.g., 2000:2020).
yr.tolerance	The number of years that a site's first/last years of observations can differ from the start/end of the user-specified time period ('yrs') for a trend to be computed.
nyr.min.frac	Fraction of years within the time period for which observations must be available if a trend is to be computed.
sig	A p-value significance cutoff used to categories trends (e.g., 0.10)

Value

A list that includes: (1) a summary message about the mean relative change across sample sites; (2) a data.table summarizing the number and percentage of sites that fall into each trend category; (3) a data.table with trend statistics for each sample site.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
# lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi', write.output = F)
lsat.pheno.dt <- lsat_fit_phenological_curves(lsat.dt, si = 'ndvi')
lsat.gs.dt <- lsat_summarize_growing_seasons(lsat.pheno.dt, si = 'ndvi')
lsat.trend.dt <- lsat_calc_trend(lsat.gs.dt, si = 'ndvi.max', yrs = 2000:2020)
lsat.trend.dt
```

---

lsat_calibrate_poly *Cross-calibrate Landsat sensors using polynomial regression*

Description

There are systematic differences in spectral indices (e.g., NDVI) among Landsat 5, 7, and 8 (Landsat Collection 2). It is important to address these differences before assessing temporal trends in spectral data. Failure to address these differences can, for instance, introduce artificial positive trends into NDVI time-series that are based on measurements from multiple Landsat sensors (Ju and Masek 2016, Roy et al. 2016, Berner et al. 2020). This function cross-calibrates individual bands or spectral indices from Landsat 5/8 to match Landsat 7. Landsat 7 is used as a benchmark because it temporally overlaps with the other two sensors. Cross-calibration can only be performed on one band or spectral index at a time. The approach involves determining the typical reflectance at a sample during a portion of the growing season site using Landsat 7 and Landsat 5/8 data that were collected the same years. Polynomial regression models from first to third order are trained to predict Landsat 7 reflectance from Landsat 5/8 reflectance, the most parsimonious model is selected using BIC, and then that model is used to cross-calbirate the data. This approach is most suitable when working with data from 100s to preferably 1000s of sample samples.

The specific steps to cross-calibrating sensors include: (1) Identify the years when both Landsat 7 and Landsat 5/8 measured surface reflectance at a sample sample. (2) Pool the reflectance measurements across those years and compute 15-day moving median reflectance over the course of the growing season for each sensor and sampling sample. (3) Exclude 15-day periods with fewer than a specified number of measurements from both sets of sensors and then randomly select one remaining 15-day period from each sample sample. (4) Split the data into sets for model training and evaluation. (5) Train polynomial regression models that predict Landsat 7 reflectance based on Landsat 5/8 reflectance. Model order (1st to 3rd) is selected using BIC. (6) Apply the polynomial regression models to cross-calibrate measurements.

See Berner et al. (2020) for a full description of the approach.

Usage

```
lsat_calibrate_poly(
  dt,
  band.or.si,
  doy.rng = 152:243,
  min.obs = 5,
  train.with.highlat.data = F,
  frac.train = 0.75,
  trim = T,
  overwrite.col = F,
```

```
  write.output = F,
  outfile.id = band.or.si,
  outdir = NA
)
```

Arguments

dt	Data.table containing the band or spectral index to cross-calibrate.
band.or.si	Character string matching the column name of the band or spectral index to cross-calibrate.
doy.rng	Sequence of numbers specifying the Days of Year to use for model development.
min.obs	Minimum number of paired, seasonally-matched observations from Landsat 7 and Landsat 5/8 required to include a sampling sample.
train.with.highlat.data	(True/False) Should the RF models be trained using an internal high-latitude dataset that sampled the Arctic and Boreal biomes?
frac.train	Fraction of samples to use for training the random forest models. The remaining samples are used for model cross-validation.
trim	(True/False) If true, then for each sample site the percent difference in spectral indices between satellites is determined. The lowest 2.5 and highest 97.5 percentiles are then trimmed. This is meant to reduce potential differences that are not directly attributable to the sensors, but rather to exogenous factors.
overwrite.col	(True/False) Overwrite existing column or (by default) append cross-calibrated data as a new column?
write.output	(True/False) Should RF models and evaluation content be written to disk? Either way, evaluation table and figure are printed in the console.
outfile.id	Identifier used when naming output files. Defaults to the input band, but can be specified if needed such as when performing Monte Carlo simulations.
outdir	Output directory (created if necessary) to which multiple files will be written. The files include: (1) fitted random forest models as R objects, (2) evaluation data in a csv file, (3) summary of model cross-validation in a csv file, and (4) multi-panel scatter plot comparing sensors pre- and post-calibration in jpeg format. If cross-calibrating both Landsat 5 and 8, then the function returns files for both sensors.

Value

The input data.table with an appended column titled band.xcal, where "band" is your specified band or spectral index.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
lsat.dt <- lsat_calibrate_poly(lsat.dt, band.or.si = 'ndvi',
                               train.with.highlat.data = TRUE,
                               write.output = FALSE)
lsat.dt
```

`lsat_calibrate_rf` *Cross-calibrate Landsat sensors using Random Forests models*

Description

There are systematic differences in spectral indices (e.g., NDVI) among Landsat 5, 7, and 8 (Landsat Collection 2). It is important to address these differences before assessing temporal trends in spectral data. Failure to address these differences can, for instance, introduce artificial positive trends into NDVI time-series that are based on measurements from multiple Landsat sensors (Ju and Masek 2016, Roy et al. 2016, Berner et al. 2020). This function cross-calibrates individual bands or spectral indices from Landsat 5/8 to match Landsat 7. Landsat 7 is used as a benchmark because it temporally overlaps with the other two sensors. Cross-calibration can only be performed on one band or spectral index at a time. The approach involves determining the typical reflectance at a sample during a portion of the growing season site using Landsat 7 and Landsat 5/8 data that were collected the same years. A Random Forest model is then trained to predict Landsat 7 reflectance from Landsat 5/8 reflectance. To account for potential seasonal and regional differences between sensors, the Random Forest models also include as covariates the midpoint of each 15-day period (day of year), the spatial coordinates of each sample sample, and potentially other use-specified variables. This approach is most suitable when working with data from 100s to preferably 1000s of sample samples.

The specific steps to cross-calibrating sensors include: (1) Identify the years when both Landsat 7 and Landsat 5/8 measured surface reflectance at a sample sample. (2) Pool the reflectance measurements across those years and compute 15-day moving median reflectance over the course of the growing season for each sensor and sampling sample. (3) Exclude 15-day periods with fewer than a specified number of measurements from both sets of sensors and then randomly select one remaining 15-day period from each sample sample. (4) Split the data into sets for model training and evaluation. (5) Train Random Forest models that predict Landsat 7 reflectance based on Landsat 5/8 reflectance. The models also account for potential seasonal and regional differences between sensors by including as covariates the midpoint of each 15-day period (day of year) and the spatial coordinates of each sampling sample. The models are trained using the ranger function from the ranger package (Wright and Ziegler, 2017). (6) Apply the fitted Random Forest models to cross-calibrate measurements.

See Berner et al. (2020) for a full description of the approach.

Usage

```
lsat_calibrate_rf(
  dt,
  band.or.si,
  doy.rng = 152:243,
  min.obs = 5,
  train.with.highlat.data = F,
  add.predictors = NULL,
  frac.train = 0.75,
  trim = T,
  overwrite.col = F,
  write.output = F,
  outfile.id = band.or.si,
  outdir = NA
)
```

Arguments

dt	Data.table containing the band or spectral index to cross-calibrate.
band.or.si	Character string matching the column name of the band or spectral index to cross-calibrate.
doy.rng	Sequence of numbers specifying the Days of Year to use for model development.
min.obs	Minimum number of paired, seasonally-matched observations from Landsat 7 and Landsat 5/8 required to include a sampling sample.
train.with.highlat.data	(True/False) Should the RF models be trained using an internal high-latitude dataset that sampled the Arctic and Boreal biomes?
add.predictors	Vector of additional predictors to use in the Random Forest models. These should be time-invariant and match column names.
frac.train	Fraction of samples to use for training the random forest models. The remaining samples are used for model cross-validation.
trim	(True/False) If true, then for each sample site the percent difference in spectral indices between satellites is determined. The lowest 2.5 and highest 97.5 percentiles are then trimmed. This is meant to reduce potential differences that are not directly attributable to the sensors, but rather to exogenous factors.
overwrite.col	(True/False) Overwrite existing column or (by default) append cross-calibrated data as a new column?
write.output	(True/False) Should RF models and evaluation content be written to disk? Either way, evaluation table and figure are printed in the console.
outfile.id	Identifier used when naming output files. Defaults to the input band, but can be specified if needed such as when performing Monte Carlo simulations.
outdir	Output directory (created if necessary) to which multiple files will be written. The files include: (1) fitted random forest models as R objects, (2) evaluation data in a csv file, (3) summary of model cross-validation in a csv file, and (4) multi-panel scatter plot comparing sensors pre- and post-calibration in jpeg format. If cross-calibrating both Landsat 5 and 8, then the function returns files for both sensors.

Value

The input data.table with an appended column titled band.xcal, where "band" is your specified band or spectral index.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi',
                             train.with.highlat.data = TRUE,
                             write.output = FALSE)
lsat.dt
```

`lsat_clean_data` *Clean Landsat surface reflectance data*

Description

This function enables users to filter out surface reflectance measurements that exhibit: (1) clouds, cloud shadows, snow, or water flagged by the CFMask algorithm; (2) surface water over the Landsat record; (2) impossibly high reflectance (>1.0) and abnormally low reflectance (<0.005); (3) scene cloud cover above a user-defined threshold; (4) geometric uncertainty above a user-defined threshold; (5) solar zenith angle above a user-defined threshold.

Usage

```
lsat_clean_data(
  dt,
  cloud.max = 80,
  geom.max = 30,
  sza.max = 60,
  filter.cfmask.snow = T,
  filter.cfmask.water = T,
  filter.jrc.water = T
)
```

Arguments

`dt`	Data.table generated by calling lsat_format_data().
`cloud.max`	Maximum allowable cloud cover in Landsat scene (percentage).
`geom.max`	Maximum allowable geometric uncertainty (meters).
`sza.max`	Maximum allowable solar zenith angle (degrees).
`filter.cfmask.snow`	(TRUE/FALSE) Remove measurements with CFmask flag = snow.
`filter.cfmask.water`	(TRUE/FALSE) Remove measurements with CFmask flag = water.
`filter.jrc.water`	(TRUE/FALSE) Remove sample sites that were ever innundated based on the maximum surface water extent variable from the JRC Global Surface Water Dataset.

Value

A data.table that includes Landsat measurements that met the quality control criteria.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt
```

`lsat_evaluate_phenological_max` *Evaluate estimates of annual phenological maximum*

Description

Assess how the number of annual Landsat measurements impacts estimates of annual maximum vegetation greenness derived from raw measurements and phenological modeling. The algorithm computes annual maximum vegetation greenness using site x years with a user-specific number of measurements and then compares these with estimates derived when using progressively smaller subsets of measurements. This lets the user determine the degree to which annual estimates of maximum vegetation greenness are impacted by the number of available measurements.

Usage

```
lsat_evaluate_phenological_max(
  dt,
  si,
  min.frac.of.max = 0.75,
  zscore.thresh = 3,
  min.obs = 6,
  reps = 10,
  outdir = NA
)
```

Arguments

`dt`	Data.table output from lsat_fit_phenological_curves().
`si`	Character string specifying the spectral index (SI) to evaluate (e.g., NDVI).
`min.frac.of.max`	Numeric threshold (0-1) that defines the "growing season" as the seasonal window when the phenological curves indicate the SI is within a specified fraction of the maximum SI. In other words, an observation is considered to be from the "growing season" when the SI is within a user-specified fraction of the curve-fit growing season maximum SI.
`zscore.thresh`	Numeric threshold specifying the Z-score value beyond which individual measurements are filtered before computing the maximum SI.
`min.obs`	Minimum number of measurements needed for a site x year to be included in the evaluation (Default = 10).
`reps`	Number of times to bootstrap the assessment (Default = 10).
`outdir`	If desired, specify the output directory where evaluation data and figure should be written. If left as NA, then no output is only displayed in the console and not written to disk.

Value

A data.table and a figure summarizing how estimates of annnual maximum SI vary with the number of Landsat measurements made during each growing season.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
# lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi', write.output = FALSE)
lsat.pheno.dt <- lsat_fit_phenological_curves(lsat.dt, si = 'ndvi')
lsat_evaluate_phenological_max(lsat.pheno.dt, si = 'ndvi')
```

---

lsat_export_ts *Export reflectance time-series from the Landsat record using rgee*

Description

This function exports surface reflectance time series for a set of point-coordinates from the whole Landsat Collection 2 record using the Google Earth Engine. The resulting time-series can then be processed using the remainder of the lsatTS workflow.

For polygon geometries consider using lsat_get_pixel_centers() to generate pixel center coordinates for all pixels within a given polygon first.

Please note: Unlike other functions in this package, this function does NOT return the time-series as an object, instead it returns a list of the EE tasks issued for the export. The actual time-series are exported as CSV objects via the EE to the user's Google Drive. This way of exporting allows for a more efficient scheduling, larger exports, and does not require the R session to continue to run in the background while the requests are processed on the EE.

The progress of the exports can be monitored using the list of tasks returned in combination with ee_monitoring() from the rgee package, or simply by using the task overview in the web code-editor of the EE (https://code.earthengine.google.com).

Usage

```
lsat_export_ts(
  pixel_coords_sf,
  sample_id_from = "sample_id",
  chunks_from = NULL,
  this_chunk_only = NULL,
  max_chunk_size = 250,
  drive_export_dir = "lsatTS_export",
  file_prefix = "lsatTS_export",
  start_doy = 152,
  end_doy = 243,
  start_date = "1984-01-01",
  end_date = "today",
  buffer_dist = 0,
  scale = 30,
  mask_value = 0
)
```

Arguments

pixel_coords_sf	Simple feature object of point coordinates for the sample.
sample_id_from	The column name that specifies the unique sample identifier in pixel_coords_sf (defaults to "sample_id" as generated by lsat_get_pixel_centers).
chunks_from	Column name in pixel_coords_sf to divide the exports into chunks. Over-rides chunk division by size (see max_chunk_size).
this_chunk_only	Name of a specific chunk to be exported. Useful for re-exporting a single chunk should the export fail for some reason.
max_chunk_size	Maximum number of sample coordinates to be exported in each chunk. Defaults to 250.
drive_export_dir	Folder on the user's Google Drive to export the records to. Defaults to "lsatTS_export".
file_prefix	Optional file_prefix for the exported files.
start_doy	Optional first day of year to extract for. Defaults to 152.
end_doy	Optional last day of year to extract for. Defaults to 243.
start_date	Optional extraction start date (as string, format YYYY-MM-DD). Defaults to "1984-01-01".
end_date	Optional extraction end date (as string, format YYYY-MM-DD). Defaults to today's date.
buffer_dist	Buffer distance around sample coordinates. Wrapper for lsat_get_pixel_centers() to find all Landsat pixel centers around each point in pixel_coords_sf within the specified buffer distance (square)). Can be slow if the number of points is large. Defaults to 0 m.
scale	scale for extraction. Defaults to 30 m nominal Landsat pixel size.
mask_value	Optional masking value for global surface water mask. Defaults to 0.

Value

List of initiated rgee tasks.

Author(s)

Jakob J. Assmann and Richard Massey

Examples

```
# Using sf, dplyr and rgee
library(sf)
library(dplyr)
library(rgee)

# Initialize EE
ee_Initialize()

# Generate test points
test_points_sf <- st_sfc(st_point(c(-149.6026, 68.62574)),
                         st_point(c(-149.6003, 68.62524)),
                         st_point(c(-75.78057, 78.87038)),
```

```
                        st_point(c(-75.77098, 78.87256)),
                        st_point(c(-20.56182, 74.47670)),
                        st_point(c(-20.55376, 74.47749)), crs = 4326) %>%
  st_sf() %>%
  mutate(sample_id = c("toolik_1",
                       "toolik_2",
                       "ellesmere_1",
                       "ellesmere_1",
                       "zackenberg_1",
                       "zackenberg_2"),
         region = c("toolik", "toolik",
                    "ellesmere", "ellesmere",
                    "zackenberg", "zackenberg"))

# Export time-series using lsat_export_ts()
task_list <- lsat_export_ts(test_points_sf)

# Export time-series using with a chunk size of 2
task_list <- lsat_export_ts(test_points_sf, max_chunk_size = 2)

# Export time-series in chunks by column
task_list <- lsat_export_ts(test_points_sf, chunks_from = "region")
```

`lsat_fit_phenological_curves` *Characterize land surface phenology using spectral vegetation index time series*

Description

This function characterizes seasonal land surface phenology at each sample site using flexible cubic splines that are iteratively fit to time series of spectral vegetation indices (e.g., NDVI). This function facilitates estimating annual maximum NDVI and other spectral vegetation indices with lsat_summarize_growing_seasons(). For each site, cubic splines are iteratively fit to measurements pooled over years within a moving window that has a user-specified width. Each cubic spline is iteratively fit, with each iteration checking if there are outliers and, if so, excluding outliers and refitting. The function returns information about typical phenology at a sample site and about the relative phenological timing of each individual measuremenent. This function was designed for situations where the seasonal phenology is hump-shaped. If you are using a spectral index that is typically negative (e.g., Normalized Difference Water Index) then multiply the index by -1 before running this function, then back-transform your index after running the lsat_summarize_growing_seasons() function.

Usage

```
lsat_fit_phenological_curves(
  dt,
  si,
  window.yrs = 7,
  window.min.obs = 20,
  si.min = 0.15,
  spar = 0.78,
  pcnt.dif.thresh = c(-30, 30),
```

```
  weight = T,
  spl.fit.outfile = F,
  progress = T,
  test.run = F
)
```

Arguments

dt	Data.table with a multi-year time series a vegetation index.
si	Character string specifying the spectral index (e.g., NDVI) to use for determining surface phenology. This must correspond to an existing column in the data.table.
window.yrs	Number specifying the focal window width in years that is used when pooling data to fit cubic splines (use odd numbers).
window.min.obs	Minimum number of focal window observations necessary to fit a cubic spline.
si.min	Minimum value of spectral index necessary for observation to be used when fitting cubic splines. Defaults to 0.15 which for NDVI is about when plants are present. Note that si.min must be >= 0 because the underlying spline fitting function will error out if provided negative values.
spar	Smoothing parameter typically around 0.70 - 0.80 for this application. A higher value means a less flexible spline. Defaults to 0.78.
pcnt.dif.thresh	Vector with two numbers that specifcy the allowable negative and positive percent difference between individual observations and fitted cubic spline. Observations that differ by more than these thresholds are filtered out and the cubic spline is iteratively refit. Defaults to -30% and 30%.
weight	When fitting the cubic splines, should individual observations be weighted by their year of acquisition relative to the focal year? If so, each observation is weighted by exp(-0.25*n.yrs.from.focal) when fitting the cubic splines.
spl.fit.outfile	(Optional) Name of output csv file containing the fitted cubic splines for each sample site. Useful for subsequent visualization.
progress	(TRUE/FALSE) Print a progress report?
test.run	(TRUE/FALSE) If TRUE, then algorithm is run using a small random subset of data and only a figure is output. This is used for model parameterization.

Value

Data.table that provides, for each observation, information on the phenological conditions for that specific day of year during the focal period. These data can then be used to estimate annual maximum spectral index and other growing season metrics using lsat_summarize_growing_season(). A figure is also generated that shows observation points and phenological curves for nine random sample locations.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi', write.output = F, train.with.highlat.data = T)
```

```
lsat.pheno.dt <- lsat_fit_phenological_curves(lsat.dt, si = 'ndvi')
lsat.pheno.dt
```

`lsat_format_data` *Formats Landsat data for analysis*

Description

This function takes Landsat data exported from GEE and formats it for subsequent use. The function parses sample site coordinates and time period of each measurement, scales band values, and formats column names as needed for subsequent analysis using the LandsatTS package.

Usage

```
lsat_format_data(dt)
```

Arguments

`dt`	Data.table with Landsat data exported from Google Earth Engine using lsat_export_ts().

Value

Data.table with formatted and scaled values.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt
```

`lsat_get_pixel_centers`
Get Landsat 8 pixel centers for a polygon or a buffered point

---

Description

A convenience helper function that determines the Landsat 8 grid (pixel) centers within a polygon plus an optional buffer. It can also be applied to a single point to retrieve all pixels within a buffer.

Does not work for large polygons. The default maximum number of pixels set by EE is 10000000 this should not be exceeded. Consider whether extraction for a large polygon is a good idea, if yes split the polygon into manageable chunks.

For the unlikely case that a polygon exceeds the boundaries of the Landsat tile closest to the polygon's center, the polygon is clipped at the boundaries of the Landsat tile and a warning is issued. Again, if this is the case, consider processing smaller polygons instead.

Please note: The approximation of the tile overlap with the polygon generates a warning by the sf package that the coordinates are assumed to be planar. This can be ignored.

Usage

```
lsat_get_pixel_centers(
  polygon_sf,
  pixel_prefix = "pixel",
  pixel_prefix_from = NULL,
  buffer = 15,
  plot_map = F,
  lsat_WRS2_scene_bounds = NULL
)
```

Arguments

polygon_sf	Simple feature with a simple feature collection of type "sfc_POLYGON" containing a single polygon geometry. Alternatively, a simple feature containing a simple feature collection of type 'sfc_POINT' with a single point.
pixel_prefix	Prefix for the generated pixel identifiers (output column "sample_id"). Defaults to "pixel".
pixel_prefix_from	Optional, a column name in the simple feature to specify the pixel_prefix. Overrides the "pixel_prefix" argument.
buffer	Buffer surrounding the geometry to be included. Specified in m. Defaults to 15 m - the nominal half-width of a Landsat pixel.
plot_map	Optional, default is FALSE. If TRUE the retrieved pixel centers and the polygon are plotted on a summer Landsat 8 image (grey-scale red band) using mapview. If a character is supplied an additional output to a file is generated (png, pdf, and jpg supported, see mapview::mapshot). Note: Both slow down the execution of this function notably, especially for large polygons! Only use in interactive R sessions.
lsat_WRS2_scene_bounds	File path to the Landsat WRS2 path row scene boundaries. If not specified the boundaries are downloaded to a temporary file when the function is executed for the first time during a session. To avoid future downloads, the file may be downloaded manually and it's file path specified using this argument. The file can be found here: https://prd-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/atoms/files/WRS-2_bound_world_0.kml See also: https://www.usgs.gov/core-science-systems/nli/landsat/ landsat-shapefiles-and-kml-files

Value

sf object of point geometries for Landsat 8 pixel centers within the polygon or the buffer around the point coordinate specified. For use in lsat_export_ts().

Author(s)

Jakob J. Assmann

Examples

```
# Using sf, dplyr, rgee and purr
library(sf)
library(dplyr)
```

```
library(rgee)
library(purrr)

# Initialize EE
ee_Initialize()

# Specify a region to retrieve pixel centers for
test_poly_sf <- list(matrix(c(-138.90125, 69.58413,
              -138.88988, 69.58358,
              -138.89147, 69.58095,
              -138.90298, 69.57986,
              -138.90125, 69.58413),
            ncol = 2, byrow = TRUE)) %>%
           st_polygon() %>%
           st_sfc(crs = 4326) %>%
           st_sf()

# Retrieve pixel centers and plot to mapview
pixels <- lsat_get_pixel_centers(test_poly_sf, plot_map = TRUE)


## Ge pixel centers for multiple regions
# Create multi-polygon sf
ellesmere <- st_polygon(list(matrix(c(-75.78526, 78.86973,
                                      -75.78526, 78.87246,
                                      -75.77116, 78.87246,
                                      -75.77116, 78.86973,
                                      -75.78526, 78.86973),
                                      ncol = 2, byrow = TRUE)))
zackenberg <- st_polygon(list(matrix(c(-20.56254, 74.47469,
                                     -20.56254, 74.47740,
                                     -20.55242, 74.47740,
                                     -20.55242, 74.47469,
                                     -20.56254, 74.47469),
                                   ncol = 2, byrow = TRUE)))
toolik <- st_polygon(list(matrix(c(-149.60686, 68.62364,
                                    -149.60686, 68.62644,
                                    -149.59918, 68.62644,
                                    -149.59918, 68.62364,
                                    -149.60686, 68.62364),
                                    ncol = 2, byrow = TRUE)))
test_regions_sf <- st_sfc(ellesmere, zackenberg, toolik, crs = 4326) %>%
  st_sf() %>%
  mutate(region = c("ellesmere", "zackenberg", "toolik"))

# Split and map lsat_get_pixel_centers using dplyr and purrr
pixel_list <- test_regions_sf %>%
   split(.$region) %>%
   map(lsat_get_pixel_centers,
       pixel_prefix_from = "region") %>%
   bind_rows()
```

---

lsat_neighborhood_mean

Compute Neighborhood Average Landsat Surface Reflectance

Description

For each band, this function computes average surface reflectance across neighboring voxels at a sample site. Use this function when working with Landsat data extracted for buffered points. Also, make sure to have previously cleaning the individual observations using lsat_clean_data().

Usage

```
lsat_neighborhood_mean(dt)
```

Arguments

dt	A data.table containing coincident surface reflectance measurements for multiple Landsat pixels at each sample site.

Value

A data.table with average surface reflectance

---

lsat_plot_trend_hist *Create a histogram summarizing relative temporal changes in a spectral index across all sample sites.*

Description

Create a histogram summarizing relative temporal changes in a spectral index across all sample sites.

Usage

```
lsat_plot_trend_hist(dt, xlim = c(-30, 30))
```

Arguments

dt	A data.table output from lsat_calc_trend()
xlim	Numeric vector specifying the minimum and maximum values for the histogram x-axis.

Value

A histogram generated by ggplot2

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
# lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi', write.output = F)
lsat.pheno.dt <- lsat_fit_phenological_curves(lsat.dt, si = 'ndvi')
lsat.gs.dt <- lsat_summarize_growing_seasons(lsat.pheno.dt, si = 'ndvi')
lsat.trend.dt <- lsat_calc_trend(lsat.gs.dt, si = 'ndvi.max', yrs = 2000:2020)
lsat_plot_trend_hist(lsat.trend.dt)
```

lsat_summarize_data *Summarize availability of Landsat data for each sample site*

Description

This little function summarizes the temporal period and availability of observations at each sample site.

Usage

```
lsat_summarize_data(dt)
```

Arguments

dt	Data.table with columns named "sample.id" and "year".

Value

Data.table summarizing for each sample site the first, last, and number of years with observations, the minimum and maximum number of observations in a year, and the total number of observations across years. Also returns a figure showing the median (2.5 and 97.5 percentiles) number of observations per sample site across years for each Landsat satellite.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat_summarize_data(lsat.dt)
```

lsat_summarize_growing_seasons *Summarize growing season characteristics using spectral vegetation indices*

Description

This function not only computes mean, median, and 90th percentile of a spectral index (SI) using observations for a user-specified "growing season," but also estimates the annual maximum SI and associated day of year using phenology modeling and growing season observations.

Usage

```
lsat_summarize_growing_seasons(
  dt,
  si,
  min.frac.of.max = 0.75,
  zscore.thresh = 3
)
```

Arguments

dt	Data.table generated by the function lsat_fit_phenological_curves().
si	Character string specifying the spectral vegetation index to summarize (e.g., NDVI).
min.frac.of.max	Numeric threshold (0-1) that defines the "growing season" as the seasonal window when the phenological curves indicate the SI is within a specified fraction of the maximum SI. In other words, an observation is considered to be from the "growing season" when the SI is within a user-specified fraction of the curve-fit growing season maximum SI.
zscore.thresh	Numeric threshold specifying the Z-score value beyond which individual observations are filtered before summarizing growing season SI.

Value

Data.table summarizing annual growing season conditions based on a spectral index.

Examples

```
data(lsat.example.dt)
lsat.dt <- lsat_format_data(lsat.example.dt)
lsat.dt <- lsat_clean_data(lsat.dt)
lsat.dt <- lsat_calc_spectral_index(lsat.dt, 'ndvi')
# lsat.dt <- lsat_calibrate_rf(lsat.dt, band.or.si = 'ndvi', write.output = F)
lsat.pheno.dt <- lsat_fit_phenological_curves(lsat.dt, si = 'ndvi')
lsat.gs.dt <- lsat_summarize_growing_seasons(lsat.pheno.dt, si = 'ndvi')
lsat.gs.dt
```

Index