Package 'SeverusPTproducts'

June 24, 2023

Type Package

Title Generate SeverusPT products for mapping burn severity

Version 0.1.0

Author Joao Goncalves

Maintainer Joao Goncalves <joao.goncalves@cibio.up.pt>

Description Generate SeverusPT products for mapping burn severity in GEE

License GPL (>=2)

URL https://severus.pt

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

R topics documented:

`ee_drive_to_local_mod` *Modified function to download files from Google Drive to local storage.*

Description

This function downloads files from Google Drive to the local storage.

Usage

```
ee_drive_to_local_mod(
  task,
  dsn,
  overwrite = TRUE,
  consider = TRUE,
  public = FALSE,
  metadata = FALSE,
  quiet = FALSE
)
```

Arguments

task	The GEE task object.
dsn	The destination file path where the files will be downloaded. If not specified, a temporary directory will be used.
overwrite	Logical indicating whether to overwrite existing files with the same name. Default is TRUE.
consider	Logical or character indicating how to handle multiple files with the same name in Google Drive. If TRUE, it prompts the user to select one file. If "last", it selects the most recent file. If "all", it downloads all files. Default is TRUE.
public	Logical indicating whether to make the downloaded files publicly accessible. Default is FALSE.
metadata	Logical indicating whether to include metadata information in the output. Default is FALSE.
quiet	Logical indicating whether to suppress progress messages. Default is FALSE.

Details

This function is a modified version of the ee_drive_to_local function from the **rgee** package. It allows downloading files from Google Drive based on a GEE task object. The files are downloaded to the specified destination file path (dsn) or to a temporary directory if dsn is not provided. The function provides options for handling multiple files with the same name in Google Drive, including selecting one file, selecting the most recent file, or downloading all files. It also supports making the downloaded files publicly accessible and including metadata information in the output.

Value

A character vector of file paths to the downloaded files, or a list with file paths and metadata information if metadata = TRUE.

---

spt_backup_file *Create a Backup of a File*

Description

This function creates a backup of a specified file by appending the suffix "_bkp" to the file name.

Usage

```
spt_backup_file(targetFile)
```

Arguments

targetFile	A character string representing the file to be backed up.

Examples

```
spt_backup_file("data.csv")
```

spt_ba_dataset_code *Burned Area Dataset Code*

Description

This function retrieves the code for a given burned area dataset.

Usage

```
spt_ba_dataset_code(baDataset)
```

Arguments

baDataset	The burnt area dataset name.

Value

Character. The dataset code corresponding to the given burned area dataset.

Examples

```
code <- spt_ba_dataset_code("ICNF")
```

spt_ba_data_url *Burned Area Dataset URL*

Description

This function retrieves the URL of the specified burned area dataset.

Usage

```
spt_ba_data_url(baDataset)
```

Arguments

baDataset	The burned area dataset code (eg., EFFIS, ICNF).

Value

The URL of the specified burned area dataset.

Examples

```
url <- spt_ba_data_url("EFFIS")
```

`spt_bitwise_extract` *Extracts bits from a numeric value using bitwise operations*

Description

This function performs bitwise operations to extract a range of bits from a numeric value. It can extract a single bit or a range of bits specified by the start and end positions.

Usage

```
spt_bitwise_extract(value, fromBit, toBit = NULL)
```

Arguments

`value`	A numeric value from which bits are to be extracted.
`fromBit`	The starting bit position from which extraction should begin.
`toBit`	(optional) The ending bit position at which extraction should stop. If not provided, extraction will be performed for a single bit at the `fromBit` position.

Details

The function performs bitwise right shift and bitwise AND operations to extract the desired bits from the input value. If only `fromBit` is provided, a single bit is extracted. If both `fromBit` and `toBit` are provided, a range of bits is extracted, inclusive of both positions. The extracted bits are returned as a numeric value.

Value

A numeric value representing the extracted bits.

Examples

```
# Extract a single bit
value <- 10  # Binary representation: 1010
extracted_bit <- spt_bitwise_extract(value, fromBit = 1)
extracted_bit  # Output: 0

# Extract a range of bits
value <- 10  # Binary representation: 1010
extracted_bits <- spt_bitwise_extract(value, fromBit = 2, toBit = 3)
extracted_bits  # Output: 1
```

spt_calc_bai *Calculate the Burned Area Index (BAI) for an image.*

Description

This function calculates the Burned Area Index (BAI) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_bai(img)
```

Arguments

img	The input image for which to calculate the BAI.

Details

The Burned Area Index (BAI) is a spectral index used to detect burned areas in remote sensing imagery. It is calculated using the formula: 1 / ((0.1 - RED)^2 + (0.06 - NIR)^2), where RED represents the red band values of the input image and NIR represents the near-infrared band values. The BAI can be used to identify areas affected by fire and assess the severity of burn scars.

Value

An image representing the calculated BAI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
bai_image <- spt_calc_bai(image)
```

spt_calc_csi *Calculate Char Soil Index (CSI)*

Description

Calculates the Char Soil Index (CSI) from an input image using the NIR and SWIR2 bands.

Usage

```
spt_calc_csi(img)
```

Arguments

img	An input image containing NIR and SWIR2 bands.

Details

This function calculates the Char Soil Index (CSI) using the NIR and SWIR2 bands. The formula for CSI is: NIR / SWIR2.

The input image should contain the NIR and SWIR2 bands as specified in the function. The output image will have a band named 'CSI' and will inherit the properties of the input image.

Smith et al. 2007

Value

An image representing the CSI.

Examples

```
# Load an image with NIR and SWIR2 bands
# img <- ee$Image('path/to/image')

# Calculate CSI
# csi <- spt_calc_csi(img)
```

---

spt_calc_dbsi *Calculate the Dry Bareness Index (DBSI) for an image.*

Description

This function calculates the Dry Bareness Index (DBSI) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_dbsi(img)
```

Arguments

img	The input image for which to calculate the DBSI.

Details

The Dry Bareness Index (DBSI) is a spectral index used to quantify the dryness and bareness of an area in remote sensing imagery. It is calculated using the formula: ((SWIR1 - GREEN) / (SWIR1 + GREEN)) - ((NIR - RED) / (NIR + RED)), where RED, NIR, SWIR1, and GREEN represent the respective band values of the input image. The DBSI can be used to assess the degree of bareness and dryness of land surfaces, which is useful for monitoring desertification, land degradation, and vegetation health.

Value

An image representing the calculated DBSI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
dbsi_image <- spt_calc_dbsi(image)
```

`spt_calc_evi` *Calculate Enhanced Vegetation Index (EVI)*

Description

Calculates the EVI from an input image using the NIR (Near Infrared), Red, and Blue bands.

Usage

```
spt_calc_evi(img)
```

Arguments

`img`	An input image containing NIR, Red, and Blue bands.

Details

This function calculates the Enhanced Vegetation Index (EVI) using the NIR, Red, and Blue bands. The formula for EVI is: 2.5 ((NIR - Red) / (NIR + 6 Red - 7.5 Blue + 1)).

The input image should contain the NIR, Red, and Blue bands as specified in the function. The output image will have a band named 'EVI' and will inherit the properties of the input image.

Value

An image representing the EVI.

Examples

```
# Load an image with NIR, Red, and Blue bands
# img <- ee$Image('path/to/image')

# Calculate EVI
# evi <- spt_calc_evi(img)
```

spt_calc_gemi *Calculate the Global Environment Monitoring Index (GEMI) for an image.*

Description

This function calculates the Global Environment Monitoring Index (GEMI) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_gemi(img)
```

Arguments

img	The input image for which to calculate the GEMI.

Details

The Global Environment Monitoring Index (GEMI) is a spectral index used to estimate the greenness and vegetation conditions in remote sensing.

It is calculated using the formula: ((2.0((NIR 2.0)-(RED 2.0)) + 1.5NIR + 0.5RED) / (NIR + RED + 0.5))(1.0 - 0.25((2.0 ((NIR 2.0) - (RED 2)) + 1.5 NIR + 0.5 RED) / (NIR + RED + 0.5)))-((RED - 0.125)/(1 - RED)),

where NIR and RED represent the respective near-infrared and red band values of the input image.

The GEMI can provide information about vegetation health, chlorophyll content, and overall environmental monitoring.

Value

An image representing the calculated GEMI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
gemi_image <- spt_calc_gemi(image)
```

spt_calc_mirbi *Calculate Mid-Infrared Burn Index (MIRBI)*

Description

Calculates the Mid-Infrared Burn Index (MIRBI) from an input image using the SWIR1 and SWIR2 bands.

Usage

```
spt_calc_mirbi(img)
```

Arguments

img	An input image containing SWIR1 and SWIR2 bands.

Details

This function calculates the Mid-Infrared Burn Index (MIRBI) using the SWIR1 and SWIR2 bands. The formula for MIRBI is: -1 * (10 * SWIR2 - 9.8 * SWIR1 + 2).

The input image should contain the SWIR1 and SWIR2 bands as specified in the function. The output image will have a band named 'MIRBI' and will inherit the properties of the input image.

Trigg, S., & Flasse, S. (2001). An evaluation of different bi-spectral spaces for discriminating burned shrub-savannah. International Journal of RemoteSensing, 22(13), 2641-2647. https://doi.org/10.1080/014311601

NOTE: THIS INDEX WAS ADJUSTED TO NEGATIVE FORM TO GIVE POSITIVE DELTAS HIGH SEVERITY

Value

An image representing the MIRBI.

Examples

```
# Load an image with SWIR1 and SWIR2 bands
# img <- ee$Image('path/to/image')

# Calculate MIRBI
# mirbi <- spt_calc_mirbi(img)
```

---

spt_calc_msavi *Calculate the Modified Soil-Adjusted Vegetation Index (MSAVI) by Rogan and Yool (2001) for an image.*

Description

This function calculates the Modified Soil-Adjusted Vegetation Index (MSAVI) by Rogan and Yool (2001) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_msavi(img)
```

Arguments

img	The input image for which to calculate the MSAVI.

Details

The Modified Soil-Adjusted Vegetation Index (MSAVI) by Rogan and Yool (2001) is a modification of the Soil-Adjusted Vegetation Index (SAVI) that improves the index's performance in areas with low vegetation cover.

It is calculated using the formula:

(2NIR + 1 - sqrt(pow((2NIR + 1), 2) - 8 * (NIR - RED))) / 2,

where NIR and RED represent the respective near-infrared and red band values of the input image. The MSAVI is particularly useful for estimating vegetation greenness in sparse vegetation areas.

Value

An image representing the calculated MSAVI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
msavi_image <- spt_calc_msavi(image)
```

---

spt_calc_nbr *Calculate Normalized Burn Ratio (NBR)*

Description

Calculates the NBR from an input image using the NIR (Near Infrared) and SWIR2 (Shortwave Infrared 2) bands.

Usage

```
spt_calc_nbr(img)
```

Arguments

img	An input image containing NIR and SWIR2 bands.

Details

This function calculates the NBR (Normalized Burn Ratio) by taking the difference between the NIR and SWIR2 bands and normalizing it. The NBR is commonly used to assess burned area severity and vegetation recovery after wildfires.

The input image should contain the NIR and SWIR2 bands as specified in the function. The output image will have a band named 'NBR' and will inherit the properties of the input image.

Value

An image representing the NBR.

Examples

```
# Load an image with NIR and SWIR2 bands
# img <- ee$Image('path/to/image')

# Calculate NBR
# nbr <- spt_calc_nbr(img)
```

`spt_calc_nbr2` *Calculate the Normalized Burn Ratio 2 (NBR2) for an image*

Description

This function calculates the Normalized Burn Ratio 2 (NBR2) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_nbr2(img)
```

Arguments

`img`	The input image for which to calculate the NBR2.

Details

The Normalized Burn Ratio 2 (NBR2) is a spectral index commonly used in remote sensing to assess burned areas and monitor post-fire effects. It is calculated using the formula: (SWIR1 - SWIR2) / (SWIR1 + SWIR2), where SWIR1 and SWIR2 represent the respective short-wave infrared band values of the input image. The NBR2 can help identify and quantify the severity and extent of burn scars, vegetation recovery, and changes in land cover after a fire event.

Value

An image representing the calculated NBR2 with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
nbr2_image <- spt_calc_nbr2(image)
```

`spt_calc_nbrp` *Adapted NBR+ (Normalized Burn Ratio Plus) that are not Sentinel-2*

Description

Calculates the NBR+ (Normalized Burn Ratio Plus) from an input image using the SWIR2, NIR, Green, and Blue bands. This function modifies the calculation to use the NIR band instead of the Red edge-4 band used in the original index formulation.

Usage

```
spt_calc_nbrp(img)
```

Arguments

`img`	An input image containing SWIR2, NIR, Green, and Blue bands.

Details

This function calculates the NBR+ (Normalized Burn Ratio Plus) using the SWIR2, NIR, Green, and Blue bands. The formula for NBR+ is: -1 * ((SWIR2 - NIR - Green - Blue) / (SWIR2 + NIR + Green + Blue)).

The input image should contain the SWIR2, NIR, Green, and Blue bands as specified in the function. The output image will have a band named 'NBRP' and will inherit the properties of the input image.

Please note that the NBR+ index was adjusted to a negative form to give positive deltas high severity. The adjustment was made according to the paper by Alcaras (2022).

Value

An image representing the NBR+.

References

Alcaras, M. 2022. "Satellite-Based Burn Severity Mapping: Implementation of the Normalized Burn Ratio Plus (NBR+) for Sentinel-2 and Landsat-8." Remote Sensing, 14(7), 1727. DOI: 10.3390/rs14071727.

Examples

```
# Load an image with SWIR2, NIR, Green, and Blue bands
# img <- ee$Image('path/to/image')

# Calculate NBR+
# nbrp <- spt_calc_nbrp(img)
```

spt_calc_nbrp_s2 *Calculate NBR+ (Normalized Burn Ratio Plus) for Sentinel-2*

Description

Calculates the NBR+ (Normalized Burn Ratio Plus) for Sentinel-2 images using the SWIR2, RE4, Green, and Blue bands.

Usage

```
spt_calc_nbrp_s2(img)
```

Arguments

img	An input image containing SWIR2, RE4, Green, and Blue bands.

Details

This function calculates the NBR+ (Normalized Burn Ratio Plus) for Sentinel-2 images using the SWIR2, RE4, Green, and Blue bands. The formula for NBR+ is: -1 * ((SWIR2 - RE4 - Green - Blue) / (SWIR2 + RE4 + Green + Blue)).

The input image should contain the SWIR2, RE4, Green, and Blue bands as specified in the function. The output image will have a band named 'NBRP' and will inherit the properties of the input image.

Note: Please note that the NBR+ index was adjusted to a negative form to give positive deltas high severity. The adjustment was made according to the specified note.

Value

An image representing the NBR+ for Sentinel-2.

References

Alcaras, M. 2022. "Satellite-Based Burn Severity Mapping: Implementation of the Normalized Burn Ratio Plus (NBR+) for Sentinel-2 and Landsat-8." Remote Sensing, 14(7), 1727. DOI: 10.3390/rs14071727.

Examples

```
# Load an image with SWIR2, RE4, Green, and Blue bands
# img <- ee$Image('path/to/image')

# Calculate NBR+ for Sentinel-2
# nbrp <- spt_calc_nbrp_s2(img)
```

`spt_calc_nbrswir` *Calculate a novel Normalized Burn Ratio (NBR_SWIR)*

Description

Calculates the Normalized Burn Ratio (NBR_SWIR) from an input image using the SWIR1 and SWIR2 bands.

Usage

```
spt_calc_nbrswir(img)
```

Arguments

`img`	An input image containing SWIR1 and SWIR2 bands.

Details

This function calculates the Normalized Burn Ratio (NBR_SWIR) using the SWIR1 and SWIR2 bands. The formula for NBR_SWIR is: - ((SWIR2 - SWIR1 - 0.02) / (SWIR2 + SWIR1 + 0.1)).

The input image should contain the SWIR1 and SWIR2 bands as specified in the function. The output image will have a band named 'NBRSWIR' and will inherit the properties of the input image.

Based on: Sicong Liu , Yongjie Zheng , Michele Dalponte & Xiaohua Tong (2020) A novel fire index-based burned area change detection approach using Landsat-8 OLI data, European Journal of Remote Sensing, 53:1, 104-112, DOI: 10.1080/22797254.2020.1738900

NOTE: THIS INDEX WAS ADJUSTED TO NEGATIVE FORM TO GIVE POSITIVE DELTAS HIGH SEVERITY

Value

An image representing the NBR_SWIR index.

Examples

```
# Load an image with SWIR1 and SWIR2 bands
# img <- ee$Image('path/to/image')

# Calculate NBR_SWIR
# nbr_swir <- spt_calc_nbrswir(img)
```

spt_calc_ndvi *Calculate Normalized Difference Vegetation Index (NDVI)*

Description

Calculates the NDVI from an input image using the NIR (Near Infrared) and Red bands.

Usage

```
spt_calc_ndvi(img)
```

Arguments

img	An input image containing NIR and Red bands.

Details

This function calculates the NDVI (Normalized Difference Vegetation Index) by taking the difference between the NIR and Red bands and normalizing it. The NDVI is a common index used to assess vegetation health and density.

The input image should contain the NIR and Red bands as specified in the function. The output image will have a band named 'NDVI' and will inherit the properties of the input image.

Value

An image representing the NDVI.

Examples

```
# Load an image with NIR and Red bands
# img <- ee$Image('path/to/image')

# Calculate NDVI
# ndvi <- spt_calc_ndvi(img)
```

spt_calc_ndwi *Calculate the Normalized Difference Water Index (NDWI) for an image.*

Description

This function calculates the Normalized Difference Water Index (NDWI) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_ndwi(img)
```

Arguments

img	The input image for which to calculate the NDWI.

Details

The Normalized Difference Water Index (NDWI) is a derived index used to estimate the leaf water content at the canopy level. It is particularly useful for mapping water bodies and assessing vegetation water stress. The NDWI is calculated using the formula:

(NIR - SWIR1) / (NIR + SWIR1)

where NIR and SWIR1 represent the respective near-infrared and shortwave infrared band values of the input image. Higher NDWI values indicate the presence of more water content, while lower values indicate less water content.

Reference: [NDWI Factsheet](https://edo.jrc.ec.europa.eu/documents/factsheets/factsheet_ndwi.pdf)

Value

An image representing the calculated NDWI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
ndwi_image <- spt_calc_ndwi(image)
```

---

spt_calc_savi *Calculate the Soil-Adjusted Vegetation Index (SAVI) for an image.*

Description

This function calculates the Soil-Adjusted Vegetation Index (SAVI) for an image in Google Earth Engine (GEE).

Usage

```
spt_calc_savi(img)
```

Arguments

img	The input image for which to calculate the SAVI.

Details

The Soil-Adjusted Vegetation Index (SAVI) is a vegetation index that adjusts the standard NDVI (Normalized Difference Vegetation Index) by taking into account the soil background reflectance. It is calculated using the formula: (1 + L) * (NIR - RED) / (NIR + RED + L),

where NIR and RED represent the respective near-infrared and red band values of the input image, and L is a soil adjustment factor (typically set to 0.5).

The SAVI is useful for vegetation analysis in areas with varying soil backgrounds and can help minimize the influence of soil reflectance.

Value

An image representing the calculated SAVI with appropriate metadata.

Examples

```
# Assuming 'image' is a valid GEE image object
savi_image <- spt_calc_savi(image)
```

---

spt_calc_tctb_l5 *Calculate TCTB (Tasseled Cap Transformation Brightness) for Landsat-5*

Description

Calculates the TCTB (Tasseled Cap Transformation Brightness) for Landsat-5 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctb_l5(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTB is calculated using the formula described in the references.

Value

An image containing the TCTB values.

---

spt_calc_tctb_l7 *Calculate TCTB (Tasseled Cap Transformation Brightness) for Landsat-7*

Description

Calculates the TCTB (Tasseled Cap Transformation Brightness) for Landsat-7 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctb_l7(img)
```

Arguments

`img`	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTB is calculated using a specific formula.

Value

An image containing the TCTB values.

`spt_calc_tctb_l8` *Calculate TCTB (Tasseled Cap Transformation Brightness) for Landsat-8 and 9*

Description

Calculates the TCTB (Tasseled Cap Transformation Brightness) for Landsat-8/9 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctb_l8(img)
```

Arguments

`img`	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTB is calculated using a specific formula.

Value

An image containing the TCTB values.

spt_calc_tctb_mod09a1 *Calculate TCTB (Tasseled Cap Transformation Brightness) for MOD09A1*

Description

Calculates the TCTB (Tasseled Cap Transformation Brightness) for MOD09A1 images using the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctb_mod09a1(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Details

The TCTB is calculated using a specific formula.

Value

An image containing the TCTB values.

spt_calc_tctb_s2 *Calculate TCTB (Tasseled Cap Transformation Brightness) for Sentinel-2*

Description

Calculates the TCTB (Tasseled Cap Transformation Brightness) for Sentinel-2 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctb_s2(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

This function calculates the TCTB (Tasseled Cap Transformation Brightness) for Sentinel-2 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands. The formula for TCTB is: (BLUE x 0.351) + (GREEN x 0.3813) + (RED x 0.3437) + (NIR x 0.7196) + (SWIR1 x 0.2396) + (SWIR2 x 0.1949).

The input image should contain the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands as specified in the function. The output image will have a band named 'TCTB' and will inherit the properties of the input image.

Value

An image representing the TCTB for Sentinel-2.

Examples

```
# Load an image with Blue, Green, Red, NIR, SWIR1, and SWIR2 bands
# img <- ee$Image('path/to/image')

# Calculate TCTB for Sentinel-2
# tctb <- spt_calc_tctb_s2(img)
```

---

spt_calc_tctg_l5 *Calculate TCTG (Tasseled Cap Transformation Greenness) for Landsat-5*

Description

Calculates the TCTG (Tasseled Cap Transformation Greenness) for Landsat-5 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctg_l5(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTG is calculated using a specific formula.

Value

An image containing the TCTG values.

---

spt_calc_tctg_l7 *Calculate TCTG (Tasseled Cap Transformation Greenness) for Landsat-7*

Description

Calculates the TCTG (Tasseled Cap Transformation Greenness) for Landsat-7 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctg_l7(img)
```

Arguments

`img`	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTG is calculated using a specific formula.

Value

An image containing the TCTG values.

`spt_calc_tctg_l8` *Calculate TCTG (Tasseled Cap Transformation Greenness) for Landsat-8 and 9*

Description

Calculates the TCTG (Tasseled Cap Transformation Greenness) for Landsat-8/9 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctg_l8(img)
```

Arguments

`img`	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTG is calculated using a specific formula.

Value

An image containing the TCTG values.

spt_calc_tctg_mod09a1 *Calculate TCTG (Tasseled Cap Transformation Greenness) for MOD09A1*

Description

Calculates the TCTG (Tasseled Cap Transformation Greenness) for MOD09A1 images using the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctg_mod09a1(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Details

The TCTG is calculated using a specific formula.

Value

An image containing the TCTG values.

spt_calc_tctg_s2 *Calculate TCTG (Tasseled Cap Transformation Greenness) for Sentinel-2*

Description

Calculates the TCTG (Tasseled Cap Transformation Greenness) for Sentinel-2 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctg_s2(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTG is calculated using the following formula:

$$TCTG = (BLUE \times -0.3599) + (GREEN \times -0.3533) + (RED \times -0.4734) + (NIR \times 0.6633) + (SWIR1 \times 0.0087) + ($$

Value

An image containing the TCTG values.

Examples

```
# Load a Sentinel-2 image
image <- ee$Image('COPERNICUS/S2_SR/20210901T184201_20210901T184205_T18TVJ')

# Calculate TCTG
tctg <- spt_calc_tctg_s2(image)
```

---

spt_calc_tctw_l5 *Calculate TCTW (Tasseled Cap Transformation Wetness) for Landsat-5*

Description

Calculates the TCTW (Tasseled Cap Transformation Wetness) for Landsat-5 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctw_l5(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTW is calculated using a specific formula.

Value

An image containing the TCTW values.

---

spt_calc_tctw_l7 *Calculate TCTW (Tasseled Cap Transformation Wetness) for Landsat-7*

Description

Calculates the TCTW (Tasseled Cap Transformation Wetness) for Landsat-7 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctw_l7(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTW is calculated using a specific formula.

Value

An image containing the TCTW values.

`spt_calc_tctw_l8` *Calculate TCTW (Tasseled Cap Transformation Wetness) for Landsat-8/9*

Description

Calculates the TCTW (Tasseled Cap Transformation Wetness) for Landsat-8 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctw_l8(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTW is calculated using a specific formula.

Value

An image containing the TCTW values.

spt_calc_tctw_mod09a1 *Calculate TCTW (Tasseled Cap Transformation Wetness) for MOD09A1*

Description

Calculates the TCTW (Tasseled Cap Transformation Wetness) for MOD09A1 images using the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctw_mod09a1(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, NIR2, SWIR1, and SWIR2 bands.

Details

The TCTW is calculated using a specific formula.

Value

An image containing the TCTW values.

spt_calc_tctw_s2 *Calculate TCTW (Tasseled Cap Transformation Wetness) for Sentinel-2*

Description

Calculates the TCTW (Tasseled Cap Transformation Wetness) for Sentinel-2 images using the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Usage

```
spt_calc_tctw_s2(img)
```

Arguments

img	An input image containing the Blue, Green, Red, NIR, SWIR1, and SWIR2 bands.

Details

The TCTW is calculated using the following formula:

$TCTW = (BLUE \times 0.2578) + (GREEN \times 0.2305) + (RED \times 0.0883) + (NIR \times 0.1071) + (SWIR1 \times -0.7611) + (SW$

Value

An image containing the TCTW values.

spt_check_gee_task *Check the status of a Google Earth Engine (GEE) task and wait until completion*

Description

This function checks the status of a GEE task at regular intervals and waits until the task is completed, failed, or cancelled.

Usage

```
spt_check_gee_task(geeTask, sleep = 60, verbose = FALSE)
```

Arguments

geeTask	The GEE task object.
sleep	The number of seconds to sleep between status checks (default is 60 seconds).
verbose	If TRUE, verbose output will be printed during the status check (default is FALSE).

Details

The function initializes the ‘status‘ variable as "RUNNING" and starts a timer. It then enters a ‘repeat‘ loop where it calls the ‘spt_gee_task_status‘ function to retrieve the current status of the GEE task. The loop continues until the task status is one of "COMPLETED", "FAILED", "CANCELLED", or "CANCEL_REQUESTED". During each iteration, the function calculates the time difference and prints the GEE process ID, task status, and approximate run time if ‘verbose‘ is set to TRUE.

Once the loop is exited, the function checks the final status and prints appropriate messages if ‘verbose‘ is TRUE. It then returns the final status of the GEE task as a list.

Value

The final status of the GEE task as a list.

See Also

spt_gee_task_status

Examples

```
# Assuming 'geeTask' is a valid GEE task object
spt_check_gee_task(geeTask)
```

spt_check_values *Check Validity of Parameter Values*

Description

This function checks the validity of parameter values based on predefined criteria or value ranges.

Usage

```
spt_check_values(value, what, verbose = TRUE)
```

Arguments

value	The value(s) to be checked.
what	A character string specifying the parameter name or type.
verbose	A logical value indicating whether to display a message when a parameter cannot be checked. Default is TRUE.

Value

A logical value indicating whether the value(s) pass the validity check.

spt_cloud_mask_fun *Get a specific Cloud Mask Function by satellite name*

Description

This function selects and returns the corresponding cloud mask function based on the specified parameters.

Usage

```
spt_cloud_mask_fun(satCode, modisProduct = NULL)
```

Arguments

satCode	The satellite code (character).
modisProduct	The MODIS product (character). Default is NULL.

Value

The cloud mask function.

spt_create_file *Create File with Current Timestamp*

Description

This function creates a new file with the specified filename and writes the current timestamp to it. If a file with the same name already exists, it will be removed before creating the new file.

Usage

```
spt_create_file(filename)
```

Arguments

filename	A character string representing the name of the file to be created.

Examples

```
spt_create_file("output.txt")
```

spt_current_date_time *Get the Current Date and Time*

Description

This function returns the current date and time in the format "YYYY-MM-DD HH:MM:SS".

Usage

```
spt_current_date_time()
```

Value

A character string representing the current date and time in "YYYY-MM-DD HH:MM:SS" format.

Examples

```
spt_current_date_time()
```

spt_current_year *Get the Current Year*

Description

This function returns the current year as a character string.

Usage

```
spt_current_year()
```

Value

A character string representing the current year.

Examples

```
spt_current_year()
```

spt_df_to_md *Convert Data Frame to Markdown Table*

Description

This function converts a data frame into a Markdown table format.

Usage

```
spt_df_to_md(df)
```

Arguments

df	A data frame to be converted to Markdown.

Value

A knitr_kable object representing the data frame in Markdown table format.

Examples

```
spt_df_to_md(iris)
```

spt_download_gdrive *Download a Google Earth Engine (GEE) task result from Google Drive*

Description

This function downloads a GEE task result from Google Drive to a local folder.

Usage

```
spt_download_gdrive(geeTask, outFolder, dataFormat = "tif")
```

Arguments

geeTask	The GEE task object or GEE status list.
outFolder	The output folder to save the downloaded file.
dataFormat	The format of the downloaded file (default is "tif").

Details

The function first checks the status of the GEE task using the `spt_gee_task_status` function. If the task is in the "COMPLETED" state, it proceeds to download the result from Google Drive to the specified `outFolder`. The downloaded file is named based on the task description and the specified `dataFormat`.

If the input `geeTask` is a GEE task object of class `"ee.batch.Task"`, the function calls `ee_drive_to_local` the task has already been completed and calls `ee_drive_to_local` directly.

The function returns the path to the downloaded file if the download is successful. Otherwise, it returns NULL.

Value

The path to the downloaded file if successful, NULL otherwise.

See Also

`spt_gee_task_status`, `ee_drive_to_local`

spt_download_unzip *Download and Unzip File*

Description

This function downloads a file from the specified URL and optionally unzips it if it is a ZIP file.

Usage

```
spt_download_unzip(url, dest_file, extdir)
```

Arguments

url	A character string representing the URL of the file to be downloaded.
dest_file	A character string representing the destination file path where the downloaded file will be saved.
extdir	A character string representing the path to extract the contents of the ZIP file. Default is the current working directory.

---

spt_etm_to_oli *Radiometric corrections between Landsat TM/ETM and OLI sensors*

Description

Converts/merges Landsat Thematic Mapper (TM) and Enhanced Thematic Mapper Plus (ETM+) imagery to Operational Land Imager (OLI) reflectance using the provided coefficients.

Usage

```
spt_etm_to_oli(img)
```

Arguments

img	An Earth Engine image object representing Landsat TM/ETM+ imagery.

Value

An Earth Engine image object representing Landsat OLI reflectance.

Examples

```
# Convert Landsat ETM+ image to OLI reflectance
etm_image <- ee$Image('LANDSAT/LE07/C01/T1_SR/LE07_044034_20181028')
oli_image <- spt_etm_to_oli(etm_image)
```

---

spt_export_meta_to_json *Export Metadata to JSON*

Description

This function exports a data frame to a prettified JSON file.

Usage

```
spt_export_meta_to_json(x, outFilePath)
```

Arguments

x	The data frame to be exported.
outFilePath	The file path for the JSON output file.

Examples

```
spt_export_meta_to_json(myDataFrame, "path/to/output.json")
```

---

spt_export_to_md *Export Data Frame to Markdown File*

Description

This function exports a data frame to a Markdown file.

Usage

```
spt_export_to_md(df, filename)
```

Arguments

df	A data frame to be exported to Markdown.
filename	A character string representing the name of the Markdown file.

Examples

```
spt_export_to_md(iris, "output.md")
```

---

spt_file_dates *Check File Creation and Modification Dates*

Description

This function checks the creation and modification dates of a file and returns TRUE if either of the dates falls within a specified time interval from the current time.

Usage

```
spt_file_dates(filename, tdelta = 60)
```

Arguments

filename	A character string representing the name of the file to check.
tdelta	The time interval in minutes. If the file's creation or modification date is within this interval from the current time, TRUE is returned. Default is 60 minutes.

Value

A logical value indicating whether the file's creation or modification date falls within the specified time interval.

Examples

```
spt_file_dates("data.txt", tdelta = 60)
spt_file_dates("data.txt")
```

spt_fill_meta_template *Fill Metadata Template*

Description

This function fills in a metadata template with values from a metadata list.

Usage

```
spt_fill_meta_template(metaTemplate, metaList)
```

Arguments

metaTemplate	The metadata template data frame.
metaList	The metadata list containing parameter-value pairs.

Value

The filled metadata template with updated values.

Examples

```
filledTemplate <- spt_fill_meta_template(metaTemplate, metaList)
```

spt_gee_task_status *Retrieve the status of a Google Earth Engine (GEE) task and convert it to an R list object*

Description

This function retrieves the status of a GEE task and converts it to an R list object.

Usage

```
spt_gee_task_status(geeTask)
```

Arguments

geeTask	The GEE task object.

Details

The function checks the status of the GEE task using `ee$batch$Task$status(geeTask)` and converts the result from Python to an R list object using the `ee_utils_py_to_r` function. It then adds extra parameters to the output list, such as the file export options, filename prefix, and file format, which are needed for the download function but cannot be stored directly from the GEE task object.

If the task has finished and the `"destination_uris"` field exists in the output list, the function retrieves the basename of the URI for the Google Drive file and stores it as `basename_uri`.

Value

A list containing the task status and additional parameters for the download function.

See Also

ee_utils_py_to_r, basename

Examples

```
# Assuming 'geeTask' is a valid GEE task object
spt_gee_task_status(geeTask)
```

`spt_generate_tasks` *Generate Tasks*

Description

This function generates a table of tasks based on the provided analysis parameters.

Usage

```
spt_generate_tasks(
  taskTable = NULL,
  satCode,
  procLevel,
  modisProduct,
  baseIndex,
  severityIndicator,
  burntAreaDataset,
  referenceYear,
  preFireRef,
  preFireType,
  postFireRef,
  minFireSize
)
```

Arguments

`taskTable`	An optional existing tasks table to append the generated tasks to.
`satCode`	The satellite code.
`procLevel`	The processing level.
`modisProduct`	The MODIS product.
`baseIndex`	The base index.
`severityIndicator`	The severity indicator.
`burntAreaDataset`	The burnt area dataset.
`referenceYear`	The reference year.

preFireRef	The pre-fire reference period in months.
preFireType	The type of pre-fire reference period.
postFireRef	The post-fire reference period in months.
minFireSize	The minimum fire size.

Value

A data frame representing the tasks table with appended rows for the generated tasks.

---

spt_get_ba_dataset *Get and pre-process the Burned Area Dataset*

Description

This function retrieves burned area (BA) data from the specified dataset and Google Earth Engine (GEE) asset. Performs a set of filters needed to remove NA's and the minimum area size.

Usage

```
spt_get_ba_dataset(
  baDataset,
  baGEEasset,
  referenceYear,
  minFireSizeHa = 5,
  dateField,
  yearField,
  areaField
)
```

Arguments

baDataset	The dataset name ("ICNF" or "EFFIS").
baGEEasset	The GEE asset representing the burned area dataset.
referenceYear	The reference year for retrieving BA data.
minFireSizeHa	The minimum fire size in hectares to filter the BA data (default: 5).
dateField	The field name representing the date in the BA dataset.
yearField	The field name representing the year in the BA dataset.
areaField	The field name representing the area in hectares in the BA dataset.

Value

The burned area data as an ee.FeatureCollection object.

Examples

```
baData <- spt_get_ba_dataset("ICNF", "users/username/ICNF_BA", 2022, 10, "date", "year", "area")
```

spt_get_sat_imgcol *Get an Earth Engine image collection*

Description

Retrieves the Earth Engine image collection based on the satellite code, processing level, and MODIS product (if applicable).

Usage

```
spt_get_sat_imgcol(satCode = "S2MSI", procLevel = "L2A", modisProduct = NULL)
```

Arguments

satCode	Character. Satellite code specifying the satellite mission. Supported values are "S2MSI" (Sentinel-2), "MOD" (MODIS), "MYD" (Aqua/MODIS), "MCD" (Combined Aqua/Terra MODIS), "L5TM" (Landsat-5/TM), "L7ETM" (Landsat-7/ETM), "L8OLI" (Landsat-8/OLI) and "L9OLI" (Landsat-9/OLI).
procLevel	Character. Processing level of the satellite imagery. Supported values depend on the satellite mission. For Sentinel-2, supported values are "L2A" and "L1C". For Landsat missions, supported values are "L2A" and "L1C". Default is "L2A".
modisProduct	Character. MODIS product code specifying the specific product to retrieve. Required for MODIS and Aqua/MODIS missions. Supported values depend on the satellite mission. Default is NULL.

Value

An Earth Engine image collection corresponding to the specified satellite mission, processing level, and MODIS product (if applicable).

Examples

```
# Retrieve Sentinel-2 Level-2A image collection
s2_collection <- spt_get_sat_imgcol(satCode = "S2MSI", procLevel = "L2A")

# Retrieve MODIS MOD09A1 image collection
modis_collection <- spt_get_sat_imgcol(satCode = "MOD", modisProduct = "MOD09A1")
```

spt_get_task *Get a single task by its identifiers*

Description

This function retrieves a specific task from a task table based on the provided parameters. It can search for a task using taskID, taskUID, geeTaskID, geeTaskCode, postProcTaskID, or closingTaskID.

Usage

```
spt_get_task(
  taskTable = NULL,
  taskID = NULL,
  taskUID = NULL,
  geeTaskID = NULL,
  geeTaskCode = NULL,
  postProcTaskID = NULL,
  closingTaskID = NULL,
  taskTablePath
)
```

Arguments

taskTable	A data frame representing the task table. If not provided, it will be read from the specified taskTablePath.
taskID	The ID of the task to retrieve.
taskUID	The UID of the task to retrieve.
geeTaskID	The GEE task ID of the task to retrieve.
geeTaskCode	The GEE task code of the task to retrieve.
postProcTaskID	The post-processing task ID of the task to retrieve.
closingTaskID	The closing task ID of the task to retrieve.
taskTablePath	The path to the task table file.

Value

A data frame representing the retrieved task, with an additional "task" class.

Examples

```
spt_get_task(taskTable = myTaskTable, taskID = "12345")
```

---

spt_lt7_interpolate *Interpolates missing values in Landsat 7 imagery*

Description

This function applies a focal mean filter to the Landsat 7 imagery in order to interpolate missing values due to SLC-off issues. It calculates the average of neighboring pixels within a square kernel of size 8x8 and replaces the missing alues with the interpolated values.

Usage

```
spt_lt7_interpolate(img)
```

Arguments

img	An Earth Engine image object representing Landsat 7 imagery.

Details

The function uses the `focal_mean` function to calculate the average of neighboring pixels within a square kernel of size 8x8. This kernel is applied to the input image, and the resulting values are used to replace the missing values in the image. The interpolation helps fill in the gaps caused by missing or corrupted data in the Landsat 7 imagery caused by SLC-off.

Value

An Earth Engine image object with missing values interpolated using a focal mean filter.

`spt_make_tasks_table` *Create Tasks Table*

Description

This function creates an empty tasks table with predefined columns to store task-related information.

Usage

```
spt_make_tasks_table(n)
```

Arguments

`n`	The number of rows to create in the tasks table.

Value

A data frame representing the tasks table with empty columns.

`spt_mask_clouds_lt5` *Masks clouds in Landsat 5 imagery*

Description

This function applies a cloud mask to Landsat 5 imagery based on the quality assessment (QA) layer. Clouds and cloud shadows are identified using specific bit masks and are then masked out in the image.

Usage

```
spt_mask_clouds_lt5(img)
```

Arguments

`img`	An Earth Engine image object representing Landsat 5 imagery.

Details

The function performs the following steps:

1. Selects the quality assessment (QA) layer from the input image using the `select` function.
2. Defines the cloud and cloud shadow bit masks using the `ee$Number` function and bitwise left shift operator (`%<<%`).
3. Applies the cloud and cloud shadow masks by performing bitwise operations (`bitwiseAnd`) on the QA layer and checking if the corresponding bits are set to zero.
4. Updates the image mask using the `updateMask` function, where pixels identified as clouds or cloud shadows are masked out.

Value

An Earth Engine image object with clouds and cloud shadows masked out.

`spt_mask_clouds_lt7` *Masks clouds in Landsat-7 imagery*

Description

This function applies a cloud mask to Landsat 7 imagery based on the quality assessment (QA) layer. Clouds and cloud shadows are identified using specific bit masks and are then masked out in the image.

Usage

```
spt_mask_clouds_lt7(img)
```

Arguments

`img`	An Earth Engine image object representing Landsat 7 imagery.

Details

The function performs the following steps:

1. Selects the quality assessment (QA) layer from the input image using the `select` function.
2. Defines the cloud and cloud shadow bit masks using the `ee$Number` function and bitwise left shift operator (`%<<%`).
3. Applies the cloud and cloud shadow masks by performing bitwise operations (`bitwiseAnd`) on the QA layer and checking if the corresponding bits are set to zero.
4. Updates the image mask using the `updateMask` function, where pixels identified as clouds or cloud shadows are masked out.

Value

An Earth Engine image object with clouds and cloud shadows masked out.

spt_mask_clouds_lt8 *Masks clouds in Landsat 8 imagery*

Description

This function applies a cloud mask to Landsat 8 imagery based on the quality assessment (QA) layer. Clouds, cloud shadows, and cirrus clouds are identified using specific bit masks and are then masked out in the image.

Usage

```
spt_mask_clouds_lt8(img)
```

Arguments

img	An Earth Engine image object representing Landsat 8 imagery.

Details

The function performs the following steps:

1. Selects the quality assessment (QA) layer from the input image using the `select` function.
2. Defines the bit masks for cirrus clouds, clouds, and cloud shadows using the `ee$Number` function and bitwise left shift operator (`%<<%`).
3. Applies the masks for cirrus clouds, clouds, and cloud shadows by performing bitwise operations (`bitwiseAnd`) on the QA layer and checking if the corresponding bits are set to zero.
4. Updates the image mask using the `updateMask` function, where pixels identified as cirrus clouds, clouds, or cloud shadows are masked out.

Value

An Earth Engine image object with clouds, cloud shadows, and cirrus clouds masked out.

spt_mask_clouds_lt9 *Masks clouds in Landsat 9 imagery*

Description

This function applies a cloud mask to Landsat 9 imagery based on the quality assessment (QA) layer. Clouds, cloud shadows, and cirrus clouds are identified using specific bit masks and are then masked out in the image.

Usage

```
spt_mask_clouds_lt9(img)
```

Arguments

img	An Earth Engine image object representing Landsat 9 imagery.

Details

The function performs the following steps:

1. Selects the quality assessment (QA) layer from the input image using the `select` function.
2. Defines the bit masks for cirrus clouds, clouds, and cloud shadows using the `ee$Number` function and bitwise left shift operator (`%<<%`).
3. Applies the masks for cirrus clouds, clouds, and cloud shadows by performing bitwise operations (`bitwiseAnd`) on the QA layer and checking if the corresponding bits are set to zero.
4. Updates the image mask using the `updateMask` function, where pixels identified as cirrus clouds, clouds, or cloud shadows are masked out.

Value

An Earth Engine image object with clouds, cloud shadows, and cirrus clouds masked out.

`spt_mask_clouds_mod09a1`

Masks clouds, cloud shadows, and cirrus in MOD09A1 imagery

Description

This function applies cloud, cloud shadow, and cirrus masks to MOD09A1 imagery based on the StateQA band values. It extracts specific bits from the StateQA band using the `spt_bitwise_extract` function and applies masks based on the extracted values.

Usage

```
spt_mask_clouds_mod09a1(image)
```

Arguments

`image`	An ee$Image object representing the MOD09A1 imagery.

Details

The function applies masks to the input image using the StateQA band values. It extracts specific bits from the StateQA band using the `spt_bitwise_extract` function and creates masks based on the extracted values. Clouds, cloud shadows, and cirrus are masked out by setting the corresponding pixels to `NA` (Not Available) in the output image.

Value

An ee$Image object with clouds, cloud shadows, and cirrus masked out.

spt_mask_clouds_mod13q1 *Masks clouds in MOD13Q1 vegetation index imagery*

Description

This function applies a cloud mask to MOD13Q1 vegetation index imagery based on the SummaryQA band values.

Usage

```
spt_mask_clouds_mod13q1(image)
```

Arguments

image	An ee$Image object representing the MOD13Q1 vegetation index imagery.

Details

The function applies a mask to the input image using the SummaryQA band values. It extracts specific bits from the SummaryQA band using the `spt_bitwise_extract` function and creates a mask based on the extracted values. Cloudy pixels are masked out by setting them to `NA` (Not Available) in the output image.

Value

An ee$Image object with clouds masked out.

spt_mask_clouds_s2 *Masks clouds and cirrus from Sentinel-2 imagery*

Description

This function applies a cloud and cirrus mask to Sentinel-2 imagery based on the quality assessment (QA) layer. It identifies cloud and cirrus pixels using specific bit masks and creates a mask to remove those pixels from the image.

Usage

```
spt_mask_clouds_s2(img)
```

Arguments

img	An Earth Engine image object representing Sentinel-2 imagery.

Details

The function performs the following steps:

1. Selects the quality assessment (QA) layer from the input image.
2. Defines bit masks for cloud and cirrus (bits 10 and 11).
3. Creates a mask by checking that both cloud and cirrus flags are set to zero, indicating clear conditions.
4. Applies the mask to the input image using the `updateMask` function.

Value

An Earth Engine image object with clouds and cirrus masked.

`spt_pad_number` *Pad a Number with Leading Zeros*

Description

This function pads a number with leading zeros to a specified width.

Usage

```
spt_pad_number(x, width = 3)
```

Arguments

`x`	A numeric or character value to be padded.
`width`	An integer specifying the desired width of the padded number. Default is 3.

Value

A character string representing the padded number.

Examples

```
spt_pad_number(7, width = 5)
spt_pad_number(123, width = 8)
spt_pad_number("42", width = 4)
```

spt_periods_to_today *Calculate the Number of Periods from a Date to Today*

Description

This function calculates the number of periods from a given date to today, based on a specified number of days per period.

Usage

```
spt_periods_to_today(date, ndays)
```

Arguments

date	A character or Date object representing the starting date.
ndays	An integer specifying the number of days per period.

Value

An integer indicating the number of periods from the starting date to today.

Examples

```
spt_periods_to_today("2022-01-01", 7)
spt_periods_to_today(as.Date("2021-12-15"), 30)
```

spt_post_window_days.task *Post-Fire Window Days for Task*

Description

This function calculates the number of days in the post-fire window for a given task, based on the pre-fire and post-fire references.

Usage

```
## S3 method for class 'task'
spt_post_window_days(x)
```

Arguments

x	A task object containing information about the pre-fire and post-fire references.

Value

A numeric vector with two elements: the number of days in the post-fire window from pre-fire reference to post-fire reference, and the number of days in the post-fire reference.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_post_window_days.task(task)
```

`spt_process_gee_task` *Process GEE Task*

Description

This function processes a GEE task by performing various calculations and operations on satellite imagery and burned area data.

Usage

```
spt_process_gee_task(
  task,
  boundBox,
  coordRefSys,
  baGEEasset,
  dateField,
  yearField,
  areaField,
  outFolder
)
```

Arguments

`task`	An object representing the GEE task.
`boundBox`	A bounding box specifying the spatial extent of the analysis.
`coordRefSys`	The coordinate reference system (CRS) code for the analysis.
`baGEEasset`	The GEE asset containing the burned area data.
`dateField`	The name of the field in the burned area data representing the date.
`yearField`	The name of the field in the burned area data representing the year.
`areaField`	The name of the field in the burned area data representing the area.
`outFolder`	The output folder where the processed data will be saved.

Value

None.

spt_proc_levels *Processing Levels description*

Description

This function provides a description of the processing levels for satellite data.

Usage

```
spt_proc_levels(procLevel)
```

Arguments

procLevel	The processing level of the satellite data.

Value

A description of the processing level.

Examples

```
levelDesc <- spt_proc_levels("L1C")
```

spt_product_name *Generate the Product Name*

Description

This function generates a product name based on the provided parameters.

Usage

```
spt_product_name(
  ProjectAccronym,
  SeverityIndicator,
  BaseIndex,
  SatCode,
  BurntAreaDataset,
  ReferenceYear,
  RefPeriods,
  CRScode,
  addCalcDate = FALSE,
  VersionNumber
)
```

Arguments

ProjectAccronym	A character string representing the project acronym.
SeverityIndicator	A character string representing the severity indicator.
BaseIndex	A character string representing the base index.
SatCode	A character string representing the satellite code.
BurntAreaDataset	A character string representing the burnt area dataset.
ReferenceYear	A character string representing the reference year.
RefPeriods	A character string representing the reference periods.
addCalcDate	A logical value indicating whether to include the calculation date in the product name. Default is FALSE.
VersionNumber	A character string representing the version number.
PrimaryCRScode	A character string representing the primary CRS code.

Value

A character string representing the generated product name.

Examples

```
spt_product_name("PROJ", "SEV", "BASE", "SAT", "BA", "2022", "PERIODS", "CRS", addCalcDate = TRUE, "V1.0")
spt_product_name("PROJ", "SEV", "BASE", "SAT", "BA", "2022", "PERIODS", "CRS", "V1.0")
```

---

spt_project_to_pttm06 *Project Raster to ETRS89/PT TM06 Coordinate Reference System*

Description

This function projects a raster object or a raster file to the ETRS89/PT TM06 (EPSG:3763) coordinate reference system. It can optionally write the projected raster to a new file with the updated CRS code in the filename.

Usage

```
spt_project_to_pttm06(x, writeData = TRUE, outPath = NULL, ...)
```

Arguments

x	A raster object or a character string representing the path to the raster file.
writeData	A logical value indicating whether to write the projected raster to a new file. Default is TRUE.
outPath	A character string representing the output file path for the projected raster. Ignored if writeData is FALSE.
...	Additional arguments to be passed to the project function.

Value

If writeData is TRUE, the function returns a raster object representing the projected raster. Otherwise, it returns NULL.

spt_raster_zeros_to_na

Convert Zeros to NA in Raster

Description

This function reads a raster file, converts all zero values to NA (missing values), and optionally writes the modified raster to the same file.

Usage

```
spt_raster_zeros_to_na(rstPath, writeRast = TRUE, ...)
```

Arguments

rstPath	A character string representing the path to the raster file.
writeRast	A logical value indicating whether to write the modified raster to the same file. Default is TRUE.
...	Additional arguments to be passed to the writeRaster function if writeRast is TRUE.

Value

If writeRast is TRUE, the function returns a raster object representing the modified raster. Otherwise, it returns NULL.

spt_read_gee_status_list

Read a Google Earth Engine (GEE) status list from disk.

Description

This function reads a GEE status list from disk in RDS format.

Usage

```
spt_read_gee_status_list(task, geeTaskPath)
```

Arguments

task	The GEE task object.
geeTaskPath	The path to the GEE task folder where the status list is stored.

Details

The function reads the GEE status list from disk using the readRDS function. It assumes that the status list was previously saved in the specified geeTaskPath using the spt_save_gee_status_list function.

Value

The GEE status list read from disk.

spt_read_tasks_table *Read Tasks Table*

Description

This function reads a task table from a specified file path and returns the table as a data frame. Uses a file lock.

Usage

```
spt_read_tasks_table(taskTablePath)
```

Arguments

taskTablePath	The file path of the task table.

Value

A data frame representing the read task table.

spt_replace_crs_code *Replace the Coordinate Reference System (CRS Code) in a String*

Description

This function replaces a specific CRS (Coordinate Reference System) code in a given string with a new CRS code. The default CRS codes used are "32629" for the source CRS and "3763" for the target CRS.

Usage

```
spt_replace_crs_code(x, from = "32629", to = "3763")
```

Arguments

x	A character string in which the CRS code will be replaced.
from	A character string representing the CRS code to be replaced. Default is "32629".
to	A character string representing the new CRS code to replace the original CRS code. Default is "3763".

Value

A character string with the replaced CRS code.

Examples

```
spt_replace_crs_code("data_32629.csv")
spt_replace_crs_code("data_32629.csv", from = "32629", to = "4326")
```

spt_rm_uncompleted_tasks *Remove Uncompleted Tasks*

Description

This function removes uncompleted tasks from the task table.

Usage

```
spt_rm_uncompleted_tasks(taskTable = NULL, taskTablePath)
```

Arguments

taskTable	The task table to remove uncompleted tasks from (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the removal is successful, FALSE otherwise.

Examples

```
spt_rm_uncompleted_tasks(taskTable = myTaskTable, taskTablePath = "path/to/taskTable.csv")
```

spt_sat_code *Define the generic methods to access task parameters*

Description

This function is a dispatch method that performs various satellite analysis operations based on the input object. The specific method to be executed is determined by the class of the input object.

Usage

```
spt_sat_code(x)
```

Arguments

x	An object for which the satellite analysis operation is to be performed.
...	Additional arguments to be passed to the dispatched method.

spt_sat_code.task *Define the get functions to access task parameters*

Description

Define the get functions to access task parameters

Usage

```
## S3 method for class 'task'
spt_sat_code(x)
```

spt_sat_mission_fullname *Satellite Mission Full Name*

Description

This function retrieves the full name of a satellite mission based on its code.

Usage

```
spt_sat_mission_fullname(satCode)
```

Arguments

satCode	The code of the satellite

Value

The full name of the satellite mission corresponding to the given code.

Examples

```
fullname <- spt_sat_mission_fullname("S2MSI")
```

spt_save_gee_status_list

Save a Google Earth Engine (GEE) status list to disk.

Description

This function saves a GEE status list to disk as RDS and CSV files.

Usage

```
spt_save_gee_status_list(task, geeStatusList, geeTaskPath)
```

Arguments

task	The GEE task object.
geeStatusList	The GEE status list to be saved.
geeTaskPath	The path to the GEE task folder where the files will be saved.

Details

The function saves the GEE status list to disk in two formats: RDS and CSV. It uses the saveRDS function to save the status list as an RDS file, and write.csv to save it as a CSV file. The files are named based on the task ID andtask UID to ensure uniqueness.

spt_scale_fun

Get a specific Scale Function based on the satellite code and processing level

Description

This function selects and returns the corresponding scaling function based on the specified parameters.

Usage

```
spt_scale_fun(satCode, procLevel = NULL, modisProduct = NULL)
```

Arguments

satCode	The satellite code (character).
procLevel	The processing level (character). Default is NULL.
modisProduct	The MODIS product (character). Default is NULL.

Value

The scaling function.

spt_scale_lt_oli_sr *Scales Landsat OLI surface reflectance imagery*

Description

This function scales the surface reflectance bands of Landsat OLI imagery to their corresponding surface reflectance values using pre-defined scaling factors and offsets. It applies the scaling to the optical bands (Coastal, Blue, Green, Red, NIR, SWIR1, SWIR2) and adds the scaled bands as new bands to the input image.

Usage

```
spt_scale_lt_oli_sr(img)
```

Arguments

img	An Earth Engine image object representing Landsat OLI surface reflectance imagery.

Details

The function performs the following steps:

1. Selects the optical bands (Coastal, Blue, Green, Red, NIR, SWIR1, SWIR2) from the input image using the `select` function.
2. Scales the selected bands by multiplying them with a scaling factor (0.0000275) and subtracting an offset (-0.2) using the `multiply` and `add` functions.
3. Adds the scaled bands as new bands to the input image using the `addBands` function.

Value

An Earth Engine image object with the optical bands scaled to surface reflectance values.

spt_scale_lt_tm_sr *Scales Landsat TM surface reflectance imagery in GEE*

Description

This function scales the surface reflectance bands of Landsat TM imagery to their corresponding surface reflectance values using pre-defined scaling factors and offsets. It applies the scaling to the optical bands (Blue, Green, Red, NIR, SWIR1, SWIR2) and adds the scaled bands as new bands to the input image.

Usage

```
spt_scale_lt_tm_sr(img)
```

Arguments

img	An Earth Engine image object representing Landsat TM surface reflectance imagery.

Details

The function performs the following steps:

1. Selects the optical bands (Blue, Green, Red, NIR, SWIR1, SWIR2) from the input image using the `select` function.
2. Scales the selected bands by multiplying them with a scaling factor (0.0000275) and subtracting an offset (-0.2) using the `multiply` and `add` functions.
3. Adds the scaled bands as new bands to the input image using the `addBands` function.

Value

An Earth Engine image object with the optical bands scaled to surface reflectance values.

`spt_scale_lt_toa` *Scales Landsat top-of-atmosphere (TOA) imagery (placeholder function)*

Description

This function maintains the top-of-atmosphere (TOA) reflectance (placeholder function because values area already scaled in this case)

Usage

```
spt_scale_lt_toa(img)
```

Arguments

`img`	An Earth Engine image object representing Landsat top-of-atmosphere (TOA) imagery.

Details

The function simply multiplies the input image by a scaling factor of 1 using the `multiply` function. This operation retains the original TOA values of the image.

Value

An Earth Engine image object with the TOA reflectance values

spt_scale_mod *Scales the values of MODIS images*

Description

This function scales the pixel values of an image in MOD format by multiplying them with a scaling factor of 0.0001.

Usage

```
spt_scale_mod(img)
```

Arguments

img	An ‘ee$Image‘ object representing the MOD-format image.

Details

The function applies a scaling factor of 0.0001 to the pixel values of the input image. This scaling is commonly used to convert MOD-format pixel values to their physical measurement units.

Value

An ‘ee$Image‘ object with scaled pixel values.

spt_scale_s2 *Scales Sentinel-2 imagery data*

Description

This function scales the pixel values of Sentinel-2 imagery by multiplying them with a scaling factor of 0.0001. It is commonly used to convert the data from integer format to floating-point format.

Usage

```
spt_scale_s2(img)
```

Arguments

img	An Earth Engine image object representing Sentinel-2 imagery.

Details

The function performs the following step:

1. Multiplies the pixel values of the input image by a scaling factor of 0.0001 using the `multiply` function.

Value

An Earth Engine image object with scaled pixel values.

spt_severity_indicator_form

Get the Severity Indicator Formula

Description

This function generates the formula for a severity indicator based on the selected spectral index (spi) and severity indicator (si). Available options are delta, Relative delta and Relativized Burn Ratio.

Usage

```
spt_severity_indicator_form(spi, si)
```

Arguments

spi	The selected spectral index.
si	The selected severity indicator.

Value

The formula for the severity indicator based on the selected spectral index and severity indicator.

Examples

```
formula <- spt_severity_indicator_form("NDVI", "DELTA")
```

spt_spatial_resolution.task

Spatial Resolution for Task

Description

This function determines the spatial resolution for a given task based on the satellite code and MODIS product (if applicable).

Usage

```
## S3 method for class 'task'
spt_spatial_resolution(x)
```

Arguments

x	A task object containing information about the satellite code and the MODIS product (if applicable).

Value

The spatial resolution in meters.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_spatial_resolution.task(task)
```

---

spt_spectral_index_fun

Gets a Spectral Index Function based on the index name and satellite

---

Description

This function selects and returns the corresponding spectral index calculation function based on the specified parameters.

Usage

```
spt_spectral_index_fun(baseIndex, satCode = NULL, modisProduct = NULL)
```

Arguments

baseIndex	The spectral index to be calculated (character).
satCode	The satellite code (character). Default is NULL.
modisProduct	The MODIS product (character). Default is NULL.

Value

The spectral index calculation function.

---

spt_spec_index_formula

Spectral Index Formula

Description

This function retrieves the formula of a spectral index based on its acronym and satellite code.

Usage

```
spt_spec_index_formula(spiAcronym, satCode)
```

Arguments

spiAcronym	The acronym of the spectral index.
satCode	The satellite code.

Value

Character. The formula of the spectral index corresponding to the given acronym and satellite code.

Examples

```
formula <- spt_spec_index_formula("TCTB", "MOD")
```

`spt_spec_index_fullname` *Spectral Index Full Name*

Description

This function retrieves the full name of a spectral index based on its acronym.

Usage

```
spt_spec_index_fullname(spiAcronym)
```

Arguments

`spiAcronym`	The acronym of the spectral index.

Value

Character. The full name of the spectral index corresponding to the given acronym.

Examples

```
fullname <- spt_spec_index_fullname("NBR")
```

`spt_start_gee_task` *Start GEE Task status*

Description

This function updates the status of a task in the task table to indicate that the associated Google Earth Engine (GEE) task has started.

Usage

```
spt_start_gee_task(task, taskTable = NULL, taskTablePath)
```

Arguments

`task`	The task object containing information about the task.
`taskTable`	The task table to update (optional). If not provided, it will be read from the specified `taskTablePath`.
`taskTablePath`	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_start_gee_task(task, taskTable = myTaskTable, taskTablePath = "path/to/taskTable.csv")
```

---

spt_status_list *Get List of Status Values*

Description

This function returns a character vector containing a list of status values for a specific process.

Usage

```
spt_status_list()
```

Value

A character vector containing the list of status values.

---

spt_task_filename *Generate Task Filename*

Description

This function generates a filename for a given task based on its properties and the specified project acronym and version number.

Usage

```
spt_task_filename(task, projectAccronym, versionNumber)
```

Arguments

task	A task object containing information about the task.
versionNumber	A numeric value specifying the version number.
projectAcronym	A character string specifying the project acronym.

Value

A character string representing the generated filename for the task.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_task_filename(task, projectAcronym = "ABC", versionNumber = 1.0)
```

spt_task_to_dataframe *Convert Task to Data Frame*

Description

This function converts a task object into a data frame, where each column represents a task parameter and the corresponding value.

Usage

```
spt_task_to_dataframe(task)
```

Arguments

task	A task object containing information about the task.

Value

A data frame with two columns: "taskParam" representing the task parameter names, and "taskValue" representing the corresponding task values.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_task_to_dataframe(task)
```

spt_update_closing_task *Update Closing Task status*

Description

This function updates the status of a closing task in the task table.

Usage

```
spt_update_closing_task(task, state, taskTable = NULL, taskTablePath)
```

Arguments

task	The task object containing information about the task.
state	The new state of the closing task.
taskTable	The task table to update (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_update_closing_task(task, state = "COMPLETED", taskTable = myTaskTable,
taskTablePath = "path/to/taskTable.csv")
```

---

spt_update_gee_task *Update GEE Task*

Description

This function updates the status and information of a Google Earth Engine (GEE) task in the task table.

Usage

```
spt_update_gee_task(geeTask, task, taskTable = NULL, taskTablePath)
```

Arguments

geeTask	The GEE task object to update.
task	The task object containing information about the task.
taskTable	The task table to update (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
geeTask <- spt_create_gee_task(task)
spt_update_gee_task(geeTask, task, taskTable = myTaskTable, taskTablePath = "path/to/taskTable.csv")
```

---

spt_update_main_task *Update Main Task status*

Description

This function updates the status of a main task in the task table.

Usage

```
spt_update_main_task(task, state, taskTable = NULL, taskTablePath)
```

Arguments

task	The task object containing information about the task.
state	The new state of the main task.
taskTable	The task table to update (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_update_main_task(task, state = "COMPLETED", taskTable = myTaskTable,
taskTablePath = "path/to/taskTable.csv")
```

---

spt_update_meta_task *Update Metadata Task status*

Description

This function updates the status of a metadata task in the task table.

Usage

```
spt_update_meta_task(task, state, taskTable = NULL, taskTablePath)
```

Arguments

task	The task object containing information about the task.
state	The new state of the metadata task.
taskTable	The task table to update (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_update_meta_task(task, state = "COMPLETED", taskTable = myTaskTable,
taskTablePath = "path/to/taskTable.csv")
```

spt_update_post_task *Update Post-Processing Task status*

Description

This function updates the status of a post-processing task in the task table.

Usage

```
spt_update_post_task(task, state, taskTable = NULL, taskTablePath)
```

Arguments

task	The task object containing information about the task.
state	The new state of the post-processing task.
taskTable	The task table to update (optional). If not provided, it will be read from the specified taskTablePath.
taskTablePath	The file path to the task table.

Value

TRUE if the update is successful, FALSE otherwise.

Examples

```
task <- spt_get_task(taskTable = myTaskTable, taskID = "12345")
spt_update_post_task(task, state = "COMPLETED", taskTable = myTaskTable,
taskTablePath = "path/to/taskTable.csv")
```

spt_write_tasks_table *Write Tasks Table*

Description

This function writes a task table to a specified file path. Uses a file lock over the to ensure concurrence management.

Usage

```
spt_write_tasks_table(taskTable, taskTablePath)
```

Arguments

taskTable	The task table to be written (data frame).
taskTablePath	The file path where the task table should be written.

Value

TRUE if the writing is successful, FALSE otherwise.

spt_ymd_date *Get the Current Date in YYYYMMDD Format*

Description

This function returns the current date in the format "YYYYMMDD".

Usage

```
spt_ymd_date()
```

Value

A character string representing the current date in YYYYMMDD format.

Examples

```
spt_ymd_date()
```

Index