Package: water (via r-universe)

November 25, 2026

Title Actual Evapotranspiration with Energy Balance Models

Version 1.0

Date 2026-10-19

Description Tools and functions to estimate actual evapotranspiration with surface energy balance models. Implements the METRIC model (Allen et al., 2007, <doi:10.1061/(ASCE)0733-9437(2007)133:4(380)>) for Landsat 7, 8 and 9 images, including Collection 2 Level-2 surface reflectance and surface temperature products, together with weather station data processing and reference evapotranspiration (ASCE Penman-Monteith).

Depends R (>= 3.5.0), terra (>= 1.7-0)

Imports sf (>= 1.0-0), methods, grDevices, graphics, stats, utils

License GPL (>= 2)

URL https://midraed.github.io/water/

BugReports https://github.com/midraed/water/issues/

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-10-21 17:00:26 UTC

RemoteUrl https://github.com/midraed/water

RemoteRef HEAD

RemoteSha 3c9463037e9f447004b46302caacd3452ae1353a

Contents

albedo . . . 3
anchors.control . . . 4
calcAnchors . . . 6
calcH . . . 9
calcH.control . . . 12
calcRadiance . . . 14
calcSR . . . 15
calcTOAr . . . 17
cfmask . . . 18
checkSRTMgrids . . . 19
Crad . . . 20
createAoi . . . 21
dailyET . . . 22
dailyETseries . . . 24
DEM_Talca . . . 25
ET24h . . . 26
ETperiod . . . 28
getDataWS . . . 30
getDEM . . . 31
hourlyET . . . 32
incLWradiation . . . 34
incSWradiation . . . 35
L7_Talca . . . 37
LAI . . . 37
loadImage . . . 39
loadImageSR . . . 40
METRIC.EB . . . 41
METRIC.G . . . 46
METRIC.Rn . . . 49
METRICtopo . . . 52
momentumRoughnessLength . . . 53
netRadiation . . . 55
outLWradiation . . . 56
plot.waterLSEB . . . 58
plot.waterWeatherStation . . . 59
plotAnchors . . . 60
prepareSRTMdata . . . 61
print.waterLSEB . . . 62
print.waterWeatherStation . . . 63
read.WSdata . . . 64
read.WSdata2 . . . 66
soilHeatFlux . . . 68
solarAngles . . . 70
summary.waterLSEB . . . 71
surfaceTemperature . . . 73
SWtransmissivity . . . 75

water-extdata 76
waterOptions 77
writeRaster.waterLSEB 78

Index **80**

`albedo` *Calculates Broadband Albedo from Landsat data*

Description

Broadband surface albedo is estimated considering the integration of all narrowband at-surface reflectances following a weighting function with empirical coefficients (Tasumi et al., 2008).

Usage

```
albedo(image.SR, coeff = "Tasumi", sat = "auto", aoi)
```

Arguments

`image.SR`	surface reflectance SpatRaster with layers named B, G, R, NIR, SWIR1 and SWIR2 (MODIS: R, NIR, B, G, SWIR1, SWIR2 and SWIR3). Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
`coeff`	coefficient to transform narrow to broad band albedo: "Tasumi" (default), "Olmedo", "Liang" or "auto" ("Olmedo" for Landsat 8 and 9, "Tasumi" for Landsat 7 and MODIS; with sat = "auto", Landsat images are taken as Landsat 8/9). See Details.
`sat`	"L7", "L8", "L9", "MODIS" or "auto": MODIS when there is a layer named SWIR3 (as from loadImage()), Landsat otherwise (L7, L8 and L9 use the same coefficients)
`aoi`	Optional area of interest (e.g. from createAoi()) to crop the result. If missing, the result is not cropped

Details

There are different models to convert narrowband data to broadband albedo. You can choose coeff = "Tasumi" to use Tasumi et al. (2008) coefficients, calculated for Landsat 7; coeff = "Liang" to use Liang (2001) Landsat 7 coefficients or coeff = "Olmedo" to use Olmedo coefficients for Landsat 8 and 9.

Value

SpatRaster with broadband surface albedo (unitless). Here the default coeff is "Tasumi" (METRIC.EB() and the other METRIC functions use "auto").

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Tasumi, M., Allen, R. G., & Trezza, R. (2008). At-surface reflectance and albedo from satellite for operational calculation of land surface energy balance. Journal of Hydrologic Engineering, 13(2), 51-63. https://doi.org/10.1061/(ASCE)1084-0699(2008)13:2(51)

Liang, S. (2001). Narrowband to broadband conversions of land surface albedo I: Algorithms. Remote Sensing of Environment, 76(2), 213-238. https://doi.org/10.1016/S0034-4257(00)00205-4

See Also

Other net radiation related functions: `LAI()`, `METRICtopo()`, `SWtransmissivity()`, `incLWradiation()`, `incSWradiation()`, `netRadiation()`, `outLWradiation()`, `soilHeatFlux()`, `solarAngles()`, `surfaceTemperature()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
plot(albedo)
```

`anchors.control` *Parameters for the selection of anchor pixels*

Description

Parameters of calcAnchors() for METRIC.EB(anchors.control =).

Usage

```
anchors.control(
  deltaTemp = 5,
  minDist = 500,
  WSbuffer = 30000,
  min.candidates = 20,
  cold.NDVI = 0.95,
  cold.LAI = 4,
  cold.fraction = 0.2,
  hot.NDVI = 0.1,
  hot.fraction = 0.2,
  homogeneity = 1.5
)
```

Arguments

deltaTemp	temperature margin (K, default 5): cold candidates must be colder than the 5% quantile of Ts over well-vegetated pixels plus deltaTemp, and hot candidates warmer than the maximum Ts over bare-soil pixels minus deltaTemp
minDist	minimum distance (m) between two anchor pixels of the same type
WSbuffer	maximum distance (m) to the weather station. If no pixel is within WSbuffer, the whole image is used, with a warning
min.candidates	minimum number of hot and of cold candidates below which a warning is given (the area may be too small or too homogeneous)
cold.NDVI	anchors.method = "percentile": cold candidates have NDVI at or above this quantile of the NDVI of the area (default 0.95)
cold.LAI	"percentile": and LAI of at least cold.LAI (default 4)
cold.fraction	"percentile": the coldest fraction of the cold candidates are the cold anchors (default 0.2)
hot.NDVI	"percentile": hot candidates have NDVI at or below this quantile (default 0.1), above 0 (no water) and albedo 0.10 to 0.35
hot.fraction	"percentile": the hottest fraction of the hot candidates are the hot anchors (default 0.2)
homogeneity	"percentile": maximum standard deviation of Ts (K) in the 5 x 5 pixels around a candidate (default 1.5)

Value

A list of class waterControl with the parameters.

See Also

calcAnchors

Other METRIC model functions: METRIC.EB(), METRIC.G(), METRIC.Rn(), calcH.control()

Examples

```
anchors.control(minDist = 1000)
```

`calcAnchors` *Select anchor pixels for the H function*

Description

Automatically searches for end members (anchor pixels) within the satellite scene (extreme wet and dry conditions): cold pixels with well-watered full-cover vegetation and hot pixels with dry bare soil.

Usage

```
calcAnchors(
  image,
  Ts,
  LAI,
  albedo,
  Z.om,
  n = 1,
  anchors.method = "percentile",
  WeatherStation,
  plots = interactive(),
  deltaTemp = 5,
  minDist = 500,
  WSbuffer = 30000,
  verbose = FALSE,
  min.candidates = 20,
  control = NULL,
  aoi
)
```

Arguments

`image`	reflectance SpatRaster (top-of-atmosphere reflectance, or surface reflectance when no DN image is available) with layers named R and NIR, used to compute NDVI. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
`Ts`	land surface temperature in K. See `surfaceTemperature`. Must be in Kelvin: values that look like Celsius, or more than 1% of the cells below 200 K (unmasked fill or clouds), stop; fewer cells below 200 K give a warning.
`LAI`	SpatRaster with leaf area index, from LAI()
`albedo`	broadband surface albedo, from albedo()
`Z.om`	momentum roughness length, from momentumRoughnessLength()

n	number of pairs of anchor pixels to select (not used by "percentile")
anchors.method	method for the selection of anchor pixels: "percentile" (default), "flexible", "best" or "random". See Details.
WeatherStation	Optional. waterWeatherStation object (see read.WSdata()). If supplied, the anchors are searched within WSbuffer of the station
plots	Logical. If TRUE (default: only in interactive sessions) the anchors are plotted over LAI: hot in red, cold in blue and the weather station (if supplied) in black
deltaTemp	(not used by "percentile") temperature margin (K, default 5): cold candidates must be colder than the 5% quantile of Ts over well-vegetated pixels plus deltaTemp, and hot candidates warmer than the maximum Ts over bare-soil pixels minus deltaTemp
minDist	(not used by "percentile") minimum distance (m) between two anchor pixels of the same type
WSbuffer	maximum distance (m) to the weather station. If no pixel is within WSbuffer, the whole image is used, with a warning
verbose	Logical. If TRUE, additional data are shown (as messages)
min.candidates	minimum number of hot and of cold candidates (default 20) below which a warning is given: the area may be too small or too homogeneous for the calibration
control	a list from anchors.control; only its percentile parameters are read (cold.NDVI, cold.LAI, cold.fraction, hot.NDVI, hot.fraction and homogeneity): deltaTemp, minDist, WSbuffer and min.candidates are arguments of calcAnchors(). METRIC.EB() fills both from its anchors.control
aoi	Not used (a warning is given when it is passed); kept for backward compatibility

Details

anchors.method = "percentile" (default) uses criteria relative to the scene, in the spirit of the automated calibration of Allen et al. (2013) and Morton et al. (2013): cold candidates are full-cover vegetation, with NDVI at or above its 95th percentile in the area and LAI of at least 4; hot candidates are bare soil, with NDVI at or below its 10th percentile, NDVI above 0 (no water) and albedo 0.10 to 0.35 (no urban surfaces or bright rock). Candidates must have a homogeneous surface temperature (sd of Ts in the 5 x 5 pixels around them up to 1.5 K). The coldest 20% of the cold candidates and the hottest 20% of the hot candidates are the anchors, and calcH() uses their means. The candidates are searched within WSbuffer of the weather station, or in the whole image if there are none. The NDVI quantiles are taken over land (NDVI > 0), so water in the area does not move them. If no candidate is homogeneous, the homogeneity test is dropped; if there are no cold candidates, cold.LAI is lowered to 3 and then cold.NDVI to the 0.85 quantile; if there are no hot candidates, hot.NDVI is raised to the 0.3 quantile. Attribute relaxation records what was relaxed. The parameters are in anchors.control.

For the other methods, cold candidates have LAI 3 to 6, albedo 0.18 to 0.25, Z.om 0.03 to 0.08, NDVI within 0.15 of the maximum NDVI of the image and Ts below the threshold given by deltaTemp. Hot candidates have albedo 0.13 to 0.15, NDVI 0.1 to 0.28, Z.om up to 0.005 and Ts above the threshold given by deltaTemp.

anchors.method = "best" selects the coldest and the hottest candidates; "random" selects them at random among the candidates (CITRA-MCB method); "flexible" works like "best" but, when there

are not enough candidates, relaxes the limits of the criteria (LAI, albedo and Z.om for cold pixels; albedo, NDVI, Z.om and Ts for hot pixels) in 1% steps up to 100%, first near the weather station (within WSbuffer) and then in the whole image, with a warning. The old names "CITRA-MCB" and "CITRA-MCBbc" (now "best") and "CITRA-MCBr" (now "random") are deprecated.

Value

A data.frame with one row per anchor pixel and columns pixel (cell number), X and Y (cell coordinates in the CRS of Ts), Ts (K), LAI, NDVI and type ("hot" or "cold"). For anchors.method = "flexible" a column flex is added: the relaxation applied to the criteria of each type (0 = none, 1 = 100%). Attribute candidates gives the number of cold and hot candidates and, for "flexible" and "percentile", attribute relaxation the criteria used. If plots = TRUE the anchors are also plotted over LAI.

Author(s)

Guillermo Federico Olmedo

Daniel de la Fuente-Sáiz

References

CITRA and MCB (personal communication)

Allen, R. G., Burnett, B., Kramber, W., Huntington, J., Kjaersgaard, J., Kilic, A., Kelly, C., & Trezza, R. (2013). Automated calibration of the METRIC-Landsat evapotranspiration process. Journal of the American Water Resources Association, 49(3), 563-576. https://doi.org/10.1111/jawr.12056

Morton, C. G., Huntington, J. L., Pohll, G. M., Allen, R. G., McGwire, K. C., & Bassett, S. D. (2013). Assessing calibration uncertainty and automation for estimating evapotranspiration from agricultural areas using METRIC. Journal of the American Water Resources Association, 49(3), 549-562. https://doi.org/10.1111/jawr.12054

See Also

Other sensible heat flux functions: `calcH()`, `momentumRoughnessLength()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
```

```
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
G <- soilHeatFlux(image = image.SR, Ts = Ts, albedo = albedo, Rn = Rn, LAI = LAI)
Z.om <- momentumRoughnessLength(LAI = LAI, mountainous = TRUE, method = "short.crops",
                                surface.model = surface.model)
anchors <- calcAnchors(image = image.TOAr, Ts = Ts, LAI = LAI, albedo = albedo, Z.om = Z.om,
                       WeatherStation = WeatherStation, plots = FALSE, verbose = FALSE)
head(anchors)
```

---

calcH *Iterative function to estimate H and R.ah*

Description

Generates an iterative solution to estimate r.ah and H because both are unknown at each pixel.

Usage

```
calcH(
  anchors,
  Ts,
  Z.om,
  WeatherStation,
  ETp.coef = 1.05,
  Z.om.ws = 0.015,
  mountainous = FALSE,
  DEM,
  Rn,
  G,
  verbose = FALSE,
  maxit = 20,
  u200.min = 2,
  damping = 0,
  ETrF.hot = 0,
  lapse = 6.49,
  plots = interactive(),
  method = "mean"
)
```

Arguments

anchors	anchor pixels: the data.frame returned by calcAnchors(), another data.frame with X, Y (coordinates in the CRS of Ts) and type columns, or an sf POINT object with a type column (transformed to the CRS of Ts). type must be "hot" or "cold" (case is ignored). The anchors are checked before iterating: they must be inside the image, Ts, Rn, G, Z.om and DEM must not be NA there, and the hot anchors must be at least 5 K warmer than the cold ones (Ts corrected to the datum elevation; for method = "lm", in every pair). A smaller difference means that the area lacks the wet and dry extremes that METRIC needs
Ts	Land surface temperature in K. See surfaceTemperature. Must be in Kelvin: values that look like Celsius, or more than 1% of the cells below 200 K (unmasked fill or clouds), stop; fewer cells below 200 K give a warning.
Z.om	momentum roughness length, from momentumRoughnessLength() (which keeps it at least 0.005 m; calcH() uses it as given)
WeatherStation	WeatherStation data at the satellite overpass. Can be a waterWeatherStation object calculated using read.WSdata and MTL file (with read.WSdata(..., overpass.radiation = "clear-sky"), the clear-sky radiation is used)
ETp.coef	ratio of the ET of the cold pixels to the instantaneous alfalfa reference ET (ETr) at the satellite overpass (see Details). Allen et al. (2007) use 1.05, the default
Z.om.ws	momentum roughness length (m) of the weather station surface, used to extrapolate the wind speed to 200 m (Allen et al., 2007, Eq. 32). It is about 0.123 times the height of the vegetation around the station (Allen et al., 1998): 0.015 m (the default) for a standard station over clipped grass 0.12 m tall, 0.03 m for vegetation about 0.25 m tall
mountainous	Logical. If TRUE heat transfer equation will be adjusted for mountainous terrain
DEM	Digital Elevation Model in meters.
Rn	Net radiation. See netRadiation
G	Soil Heat Flux. See soilHeatFlux
verbose	Logical. If TRUE, information about every iteration is shown (as messages)
maxit	Maximum number of iterations. Default 20.
u200.min	minimum wind speed at 200 m (m/s, default 2). See calcH.control
damping	weight of the previous aerodynamic resistance in each iteration (default 0, no damping). See calcH.control
ETrF.hot	reference ET fraction of the hot anchors (default 0). See calcH.control
lapse	lapse rate (K/km) used to refer Ts to the elevation of the weather station (default 6.49). See calcH.control
plots	Logical. If TRUE (default: only in interactive sessions) the aerodynamic resistance of the anchors is plotted at each iteration
method	Method when using more than 1 pair of anchor pixels: "mean" (default) uses the mean values of the cold pixels and of the hot pixels; "lm" iterates each hot/cold pair in turn (maxit applies to each pair) and returns a and b of the last pair

Details

Sensible heat flux is the rate of heat loss to the air by convection and conduction, due to a temperature difference. This parameter is computed using a one-dimensional, aerodynamic, temperature-gradient-based equation for heat transport. This method is difficult to solve because there are two unknowns, rah and dT. To facilitate this computation, METRIC utilizes the two "anchor" pixels and solves for the dT that satisfies the equation given the aerodynamic roughness and wind speed at a given height. Aerodynamic resistance and heat transfer are impacted by buoyancy of heated, light air at the surface, especially when H is large. Therefore, correction to rah is needed to account for buoyancy effects. However, H is needed to make this correction. An iterative solution for both H and rah is used.

The ET of the cold pixels is computed as ETp.coef times the instantaneous alfalfa reference ET (ETr, ASCE standardized equation; see `hourlyET`) at the satellite overpass, as in Allen et al. (2007). ET24h() and dailyET() use the same reference.

Value

A list with a and b (coefficients of dT = a * Ts_datum + b), dT (SpatRaster, near-surface temperature difference, K), H (SpatRaster, sensible heat flux, W/m2), converged (logical), iterations (number of iterations; one per pair for method = "lm") and u200 (wind speed at 200 m used, m/s). The aerodynamic resistance of the anchor pixels at each iteration is plotted with plots = TRUE. The iteration stops with an error when it diverges.

Author(s)

Guillermo Federico Olmedo

Daniel de la Fuente-Sáiz

Fernando Fuentes-Peñailillo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop evapotranspiration: guidelines for computing crop water requirements. FAO Irrigation and Drainage Paper 56. FAO, Rome.

Allen, R., Irmak, A., Trezza, R., Hendrickx, J.M.H., Bastiaanssen, W., Kjaersgaard, J., 2011. Satellite-based ET estimation in agriculture using SEBAL and METRIC. Hydrol. Process. 25, 4011-4027. doi:10.1002/hyp.8408

See Also

Other sensible heat flux functions: `calcAnchors()`, `momentumRoughnessLength()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
            incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
G <- soilHeatFlux(image = image.SR, Ts = Ts, albedo = albedo, Rn = Rn, LAI = LAI)
Z.om <- momentumRoughnessLength(LAI = LAI, mountainous = TRUE, method = "short.crops",
                                surface.model = surface.model)
anchors <- calcAnchors(image = image.TOAr, Ts = Ts, LAI = LAI, albedo = albedo, Z.om = Z.om,
                       WeatherStation = WeatherStation, plots = FALSE, verbose = FALSE)
H <- calcH(anchors = anchors, Ts = Ts, Z.om = Z.om, WeatherStation = WeatherStation,
           DEM = DEM, Rn = Rn, G = G, plots = FALSE, verbose = FALSE)
plot(H$H)
```

`calcH.control` *Parameters for the iteration of H and the aerodynamic resistance*

Description

Parameters of calcH() for METRIC.EB(calcH.control =).

Usage

```
calcH.control(
  maxit = 20,
  u200.min = 2,
  damping = 0,
  ETrF.hot = 0,
  cold.NDVI.rule = FALSE,
```

```
  lapse = 6.49,
  slope.min1 = FALSE
)
```

Arguments

maxit	maximum number of iterations
u200.min	minimum wind speed at 200 m (m/s). With a lower wind speed extrapolated from the station, u200.min is used, with a warning. 2 m/s corresponds to about 1 m/s at 2 m over grass. A choice of this package, not of Allen et al. (2007)
damping	weight of the previous aerodynamic resistance in each iteration, 0 (default, no damping) to below 1: r_ah = (1 - damping) * new + damping * previous. Damping keeps the same solution and helps runs that oscillate instead of converging. A choice of this package
ETrF.hot	reference ET fraction of the hot anchor pixels (default 0, dry bare soil). After rain the soil may still evaporate; Allen et al. (2007, Eq. 47) then compute dT of the hot pixels from Rn - G - LE.hot. METRIC.EB() warns when the station recorded more than 5 mm of rain in the 3 days before the image and ETrF.hot is 0
cold.NDVI.rule	logical (default FALSE). If TRUE, METRIC.EB() lowers the ET fraction of automatically selected cold anchors with NDVI below 0.8: min(ETp.coef, Kc * ETo / ETr), with Kc = 2.11 NDVI - 0.4989 (Reyes-González et al., 2015), for every anchors.method. If FALSE, only "flexible" does it, when the cold criteria had to be relaxed. The NDVI is from top-of-atmosphere reflectance with image.DN, from surface reflectance otherwise
lapse	lapse rate (K/km) of the surface temperature with elevation, for plain = FALSE (default 6.49): Ts is referred to the elevation of the weather station for the dT calibration and the anchors
slope.min1	logical (default FALSE). For plain = FALSE, the roughness adjustment for terrain of Allen et al. (2007, Eq. 35), 1 + (slope - 5) / 20, lowers the roughness on slopes below 5 degrees; if TRUE it is at least 1, as in the METRIC manual

Value

A list of class waterControl with the parameters.

See Also

calcH

Other METRIC model functions: METRIC.EB(), METRIC.G(), METRIC.Rn(), anchors.control()

Examples

```
calcH.control(maxit = 30)
```

`calcRadiance` *Calculates at-sensor radiance*

Description

Converts digital numbers to at-sensor spectral radiance using the RADIANCE_MULT_BAND_x and RADIANCE_ADD_BAND_x values of the MTL file.

Usage

```
calcRadiance(image.DN, sat = "auto", MTL)
```

Arguments

`image.DN`	raw image in digital numbers, e.g. from loadImage(). Bands B, G, R, NIR, SWIR1, SWIR2, Thermal1 and Thermal2 (L8/L9) are used. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
`sat`	"L7", "L8", "L9" or "auto" to detect it from the MTL file (MTL is then required)
`MTL`	path to the Landsat metadata (*_MTL.txt) file. Required

Value

SpatRaster with at-sensor spectral radiance (W m-2 sr-1 um-1) in layers B, G, R, NIR, SWIR1, SWIR2, Thermal1 and, for Landsat 8/9, Thermal2.

Author(s)

Guillermo Federico Olmedo

María Victoria Munafó

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

LPSO. (2004). Landsat 7 science data users handbook, Landsat Project Science Office, NASA Goddard Space Flight Center, Greenbelt, Md., (http://landsathandbook.gsfc.nasa.gov/) (Feb. 5, 2007)

U.S. Geological Survey (2019). Landsat 8 (L8) Data Users Handbook, LSDS-1574. EROS, Sioux Falls, South Dakota. https://www.usgs.gov/landsat-missions/landsat-8-data-users-handbook

See Also

Other remote sensing support functions: `calcSR()`, `calcTOAr()`, `cfmask()`, `loadImage()`, `loadImageSR()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
calcRadiance(image.DN, sat = "L7", MTL = MTL)
```

`calcSR` *Calculates surface reflectance for Landsat 7*

Description

Calculates surface reflectance from top-of-atmosphere reflectance using the model developed by Tasumi et al. (2008) and Allen et al. (2007), applied band by band. Only for Landsat 7: for Landsat 8 and 9 use the Collection 2 Level-2 surface reflectance (see `loadImageSR`).

Usage

```
calcSR(
  image.TOAr,
  sat = "auto",
  incidence.hor,
  WeatherStation,
  surface.model,
  MTL,
  Kt = 1,
  aoi
)
```

Arguments

`image.TOAr`	SpatRaster with top-of-atmosphere reflectance in layers B, G, R, NIR, SWIR1 and SWIR2, e.g. from calcTOAr(). Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
`sat`	Only "L7" is supported; "L8" and "L9" stop with an error. "auto" detects the satellite from the MTL file (MTL is then required)
`incidence.hor`	solar incidence angle (radians) on a horizontal surface, from solarAngles()
`WeatherStation`	waterWeatherStation object (see `read.WSdata`) or the data.frame returned by getDataWS()
`surface.model`	SpatRaster with layers DEM, Slope and Aspect, from METRICtopo()
`MTL`	path to the Landsat metadata (*_MTL.txt) file. Needed for sat = "auto"

Kt	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See SWtransmissivity
aoi	Not used (a warning is given when it is passed); kept for backward compatibility

Value

SpatRaster with surface reflectance (unitless) in layers B, G, R, NIR, SWIR1 and SWIR2.

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

Tasumi, M., Allen, R. G., & Trezza, R. (2008). At-surface reflectance and albedo from satellite for operational calculation of land surface energy balance. Journal of Hydrologic Engineering, 13(2), 51-63. https://doi.org/10.1061/(ASCE)1084-0699(2008)13:2(51)

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other remote sensing support functions: calcRadiance(), calcTOAr(), cfmask(), loadImage(), loadImageSR()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
plot(image.SR)
```

`calcTOAr` *Calculates top-of-atmosphere reflectance*

Description

Calculates the TOA (top-of-atmosphere) reflectance, divided by the cosine of the solar incidence angle of each pixel. For Landsat 8 and 9 the reflectance rescaling coefficients of the Level-1 products (2e-05 and -0.1) are used; for Landsat 7, the gains, biases and ESUN values of the Landsat 7 handbook and the acquisition date in the MTL file.

Usage

```
calcTOAr(image.DN, sat = "auto", incidence.rel, MTL, aoi)
```

Arguments

`image.DN`	raw image in digital numbers, e.g. from loadImage(). Bands B, G, R, NIR, SWIR1 and SWIR2 are used. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
`sat`	"L7", "L8", "L9" or "auto" to detect it from the MTL file (MTL is then required)
`incidence.rel`	solar incidence angle (radians) considering the relief, from solarAngles()
`MTL`	path to the Landsat metadata (*_MTL.txt) file. Needed for Landsat 7 and for sat = "auto"
`aoi`	Not used (a warning is given when it is passed). Crop the inputs beforehand (e.g. loadImage(..., aoi = aoi))

Value

SpatRaster with top-of-atmosphere reflectance (unitless) in layers B, G, R, NIR, SWIR1 and SWIR2.

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

LPSO. (2004). Landsat 7 science data users handbook, Landsat Project Science Office, NASA Goddard Space Flight Center, Greenbelt, Md., (http://landsathandbook.gsfc.nasa.gov/) (Feb. 5, 2007)

U.S. Geological Survey (2019). Landsat 8 (L8) Data Users Handbook, LSDS-1574. EROS, Sioux Falls, South Dakota. https://www.usgs.gov/landsat-missions/landsat-8-data-users-handbook

See Also

Other remote sensing support functions: `calcRadiance()`, `calcSR()`, `cfmask()`, `loadImage()`, `loadImageSR()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
plot(image.TOAr)
```

---

cfmask *Mask clouds using a quality band*

Description

Masks clouds, cloud shadows and fill pixels of an image using its quality band. Use it after loading the image (e.g. with loadImageSR()) and before METRIC.EB(), so that cloudy pixels are not used as anchor pixels. All layers of the image are masked, including the surface temperature (Band_10).

Usage

```
cfmask(path = getwd(), image, cfmask, keep, buffer = 60)
```

Arguments

`path`	folder where the quality band is searched when cfmask is missing: a Collection 2 `*_QA_PIXEL.TIF` file or, for older products, a `*_cfmask.tif` or `*_BQA.TIF` file
`image`	SpatRaster to mask, e.g. from loadImageSR() or loadImage()
`cfmask`	SpatRaster with the quality band. If missing, it is read from path

keep	For older quality bands only: values to keep (all other pixels are masked). Use 0 for cfmask files, 2720 for Landsat 8 Collection 1 BQA files and 672 for Landsat 7 Collection 1 BQA files. If missing, a Collection 2 QA_PIXEL band is decoded by bits: pixels flagged as fill, dilated cloud, cirrus, cloud or cloud shadow (bits 0 to 4) are masked.
buffer	distance (in map units, usually meters) around clouds and cloud shadows that is also masked. Use 0 for no buffer

Value

The image (SpatRaster) with the masked pixels set to NA.

Author(s)

Guillermo Federico Olmedo

See Also

Other remote sensing support functions: calcRadiance(), calcSR(), calcTOAr(), loadImage(), loadImageSR()

Examples

```
scene <- system.file("extdata", "L8_C2", package = "water")
image.SR <- loadImageSR(scene)
image.SR <- cfmask(path = scene, image = image.SR)
plot(image.SR[[1]])
```

---

checkSRTMgrids *Check needed SRTM grids from image extent*

Description

Lists the 1 x 1 degree SRTM tiles needed to cover the extent of an image, with the file names that prepareSRTMdata() reads. The tiles can be downloaded from EarthExplorer (dataset "SRTM 1 Arc-Second Global"). To download a DEM automatically, use getDEM().

Usage

```
checkSRTMgrids(raw.image)
```

Arguments

raw.image	SpatRaster whose extent (and CRS) is used to find the SRTM tiles needed

Value

Character vector with the file names of the SRTM tiles needed (e.g. "s34_w069_1arc_v3.tif"). The number of tiles is also reported.

Author(s)

Guillermo Federico Olmedo

See Also

Other support functions: `createAoi()`, `getDEM()`, `prepareSRTMdata()`, `waterOptions()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
# names of the SRTM tiles needed to cover the image
checkSRTMgrids(image.DN)
```

Crad *Correction of daily ET for sloping terrain*

Description

Calculates C_rad, the correction of daily ET for the slope and aspect of each pixel (Allen et al., 2007, Eqs. 55-57): the ratio of the clear-sky radiation of the pixel to that of a horizontal surface over 24 h, relative to the same ratio at the satellite overpass. ET24h() uses it for METRIC.EB() results computed with a DEM (plain = FALSE).

Usage

```
Crad(surface.model, solar.angles, WeatherStation, Kt = 1)
```

Arguments

`surface.model`	SpatRaster with layers DEM, Slope and Aspect, from METRICtopo()
`solar.angles`	SpatRaster with layers latitude, declination, hour.angle, incidence.hor and incidence.rel, from solarAngles()
`WeatherStation`	waterWeatherStation object (see `read.WSdata`) or the data.frame returned by getDataWS(); its vapour pressure at the overpass is used for the transmissivity
`Kt`	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air. See `SWtransmissivity`

Details

C_rad = (Rso_inst horizontal / Rso_inst pixel) * (Rso_24 pixel / Rso_24 horizontal), with the clear-sky radiation of Eq. 3 integrated over the day in 30-minute steps (Eq. 57). Only self-shading (the sun behind the slope) is considered, not the shadows of other slopes. C_rad is 1 on horizontal pixels. On slopes barely lit at the overpass the cosine of the incidence angle is taken as at least 0.1, and C_rad is limited to 0.5-2 (choices of this package; C_rad reached 3.5 at grazing sun on DEM_Talca).

Value

SpatRaster with C_rad (unitless) in a layer named C.rad.

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other ET functions: `ET24h()`, `ETperiod()`, `dailyET()`, `dailyETseries()`, `hourlyET()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Crad(surface.model, solar.angles, WeatherStation)
```

---

createAoi *Create aoi polygon from topleft and bottomright coordinates*

Description

An AOI (Area of Interest) is created based on two points (topleft and bottomright) using a coordinate reference system.

Usage

```
createAoi(topleft, bottomright, EPSG)
```

Arguments

topleft	a vector with topleft x,y coordinates
bottomright	a vector with bottomright x,y coordinates
EPSG	EPSG code of the coordinate reference system of the coordinates (e.g. 32619 for WGS 84 / UTM zone 19N). Optional: if missing, the CRS is not set

Value

An sf data frame with a single POLYGON geometry (the rectangle defined by topleft and bottomright). Its CRS is set only when EPSG is supplied.

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

See Also

Other support functions: checkSRTMgrids(), getDEM(), prepareSRTMdata(), waterOptions()

Examples

```
tl <- c(493300, -3592700)
br <- c(557200, -3700000)
aoi <- createAoi(topleft = tl, bottomright=br, EPSG=32619)
plot(aoi)
```

---

dailyET — *Calculates daily reference ET as the sum of hourly Penman-Monteith values*

Description

Sums the hourly reference ET (ASCE standardized Penman-Monteith equation, see hourlyET) of the hourly weather station values of one day, in the time zone of the station records: the day of the satellite overpass or another day.

Usage

```
dailyET(
  WeatherStation,
  DOY,
  height,
  lat,
  long,
  elev,
  ET = "ETr",
```

```
  long.z = NULL,
  date = "auto",
  MTL
)
```

Arguments

WeatherStation	waterWeatherStation object, see read.WSdata. Required (other inputs stop)
DOY	Not used
height	weather station sensors height in meters. If missing, taken from WeatherStation
lat	latitude in decimal degrees of the weather station. If missing, taken from WeatherStation
long	longitude in decimal degrees of the weather station. If missing, taken from WeatherStation
elev	elevation in meters of the weather station. If missing, taken from WeatherStation
ET	"ETo" for short crops, similar to clipped, cool-season grass; or "ETr" (default) for tall crops, similar to 0.5 m tall full-cover alfalfa.
long.z	longitude of the centre of the time zone of the weather station clock (decimal degrees, negative west). By default it is computed from the time zone of the records; see hourlyET
date	"auto" (default) uses the acquisition date in the MTL file or, without MTL, the date of the conditions at the overpass of WeatherStation (at.sat); otherwise a date (e.g. "2016-02-09") present in the weather station data, taken in the time zone of the station records. Without MTL, date or at.sat, dailyET() stops
MTL	path to the Landsat metadata (*_MTL.txt) file. Optional: the day of the overpass in the time zone of the station selects the day (see date)

Details

As in ASCE-EWRI (2005), the day is the 24 hourly periods ending at 01:00 to 24:00 (00:00 of the next day); 23 or 25 periods on days when daylight saving time starts or ends. The cloudiness function of the net long-wave radiation is computed while the sun is higher than 0.3 rad, and its last value is carried over dusk, night and dawn (Eq. 46); before the first sunny period the previous evening is used, or the last sunny period of the day if the records do not include the previous day.

Value

Numeric. Daily reference ET (mm/day): the sum of the hourly values of the day. If the records do not cover the whole day, NA with a warning (only the last period, ending at 24:00, may be missing: it is then left out, with a message).

Author(s)

Guillermo Federico Olmedo

References

ASCE-EWRI (2005). The ASCE Standardized Reference Evapotranspiration Equation. Report of the ASCE-EWRI Task Committee on Standardization of Reference Evapotranspiration.

See Also

Other ET functions: `Crad()`, `ET24h()`, `ETperiod()`, `dailyETseries()`, `hourlyET()`

Examples

```
csvfile <- system.file("extdata", "apples.csv", package="water")

WeatherStation <- read.WSdata(WSdata = csvfile, date.format = "%d/%m/%Y",
lat=-35.42222, long= -71.38639, elev=201, height= 2.2, cf=c(1,0.2777778,1),
tz = "America/Santiago")

dailyET(WeatherStation = WeatherStation, ET="ETo", date = "2013-02-15")
```

---

dailyETseries *Daily reference ET for each day of a weather station series*

Description

Applies `dailyET` to every day (or to the days requested) of the records of a weather station, to be used as the reference ET of `ETperiod`.

Usage

```
dailyETseries(WeatherStation, dates = NULL, ET = "ETr", ...)
```

Arguments

`WeatherStation`	`waterWeatherStation` object, see `read.WSdata`
`dates`	Dates to compute, as Date or "YYYY-MM-DD" text. By default all the days with records in the station data.
`ET`	"ETr" (alfalfa reference, default) or "ETo" (grass).
`...`	further arguments of `dailyET`

Details

Days that do not have all their hourly records give NA (and no warning): `ETperiod` stops on them unless `fill.gaps = TRUE`.

Value

data.frame with the columns date and ET (mm/day)

See Also

Other ET functions: `Crad()`, `ET24h()`, `ETperiod()`, `dailyET()`, `hourlyET()`

Examples

```
csvfile <- system.file("extdata", "apples.csv", package="water")
WeatherStation <- read.WSdata(WSdata = csvfile, date.format = "%d/%m/%Y",
lat=-35.42222, long= -71.38639, elev=201, height= 2.2, cf=c(1,0.2777778,1),
tz = "America/Santiago")
dailyETseries(WeatherStation, dates = c("2013-02-14", "2013-02-15"))
```

DEM_Talca *SRTM DEM from east of Talca, Chile*

Description

A SRTM digital elevation model (m) on the same grid as `L7_Talca`, stored as a terra `PackedSpatRaster`. Unpack it with `rast(DEM_Talca)`.

Usage

```
DEM_Talca
```

Format

A `PackedSpatRaster`; `rast(DEM_Talca)` is a SpatRaster with 417 rows, 508 columns and one layer (DEM, 131 to 643 m), WGS 84 / UTM zone 19S.

Details

Original data is float. Example data is integer.

Data available from the U.S. Geological Survey.

Source

https://www.usgs.gov/

Examples

```
DEM <- rast(DEM_Talca)
plot(DEM)
```

ET24h *Calculates daily ET from a surface energy balance and Weather Station*

Description

Calculates the daily (24 h) actual evapotranspiration: the instantaneous ET (from LE = Rn - G - H) divided by the instantaneous alfalfa reference ET (ETr, see hourlyET()) at the satellite overpass gives the reference ET fraction (ETrF), which is multiplied by the daily reference ET. The simplest call is ET24h(EB), with EB from METRIC.EB().

Usage

```
ET24h(
  Rn,
  G,
  H,
  Ts,
  WeatherStation,
  ETr.daily,
  C.rad = 1,
  plots = interactive()
)
```

Arguments

Rn	Net radiation (W/m2), see netRadiation; or the result of METRIC.EB(), which supplies Rn, G, H, Ts and the weather station. See Value
G	Soil Heat Flux (W/m2). See soilHeatFlux
H	Sensible Heat Flux (W/m2). See calcH
Ts	Land surface temperature (K). See surfaceTemperature
WeatherStation	WeatherStation data at the satellite overpass. Can be a waterWeatherStation object calculated using read.WSdata and MTL file (with read.WSdata(..., overpass.radiation = "clear-sky"), the clear-sky radiation is used, as in calcH()). Optional with a METRIC.EB() result
ETr.daily	daily reference ET (mm/day) for the day of the image, e.g. from dailyET() (whose default is the alfalfa reference, ETr). If missing, dailyET(WeatherStation) is used (the day of the overpass)
C.rad	correction term used in sloping terrain to correct for variation in 24 h versus instantaneous energy availability (Allen et al., 2007, Eq. 56): a number or a SpatRaster. With a METRIC.EB() result computed with a DEM (plain = FALSE) it is computed with Crad; 1 otherwise
plots	Logical. If TRUE (default: only in interactive sessions) ET24 is plotted

Details

ETrF is a crop coefficient on the alfalfa reference basis. To compare it with a crop coefficient on the grass reference basis (Kc, FAO-56), multiply it by the ratio of the daily alfalfa and grass reference ET, dailyET(WeatherStation) / dailyET(WeatherStation, ET = "ETo").

LE = Rn - G - H is not limited, so ETinst and ETrF are negative where H > Rn - G (usually hot, dry pixels, or where the calibration does not fit); layer HgtRnG flags them. METRIC.EB() sets LE to 0 there.

Value

With Rn, G, H and Ts: a SpatRaster with daily (24 h) actual evapotranspiration (mm/day) in a layer named ET24; negative values are set to 0. With a METRIC.EB() result: a SpatRaster with layers ETinst (instantaneous ET at the overpass, mm/h), ETrF (reference ET fraction, unitless), ET24 (mm/day, negative values set to 0) and HgtRnG (1 where H > Rn - G, else 0), and C.rad for METRIC.EB() results computed with a DEM.

Author(s)

Guillermo Federico Olmedo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other ET functions: `Crad()`, `ETperiod()`, `dailyET()`, `dailyETseries()`, `hourlyET()`

Examples

```
# the usual way: ET24h() takes the result of METRIC.EB()
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
ET <- ET24h(Energy.Balance)
plot(ET$ET24)

# step by step, with the layers computed one by one:
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
```

```
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
            incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
G <- soilHeatFlux(image = image.SR, Ts = Ts, albedo = albedo, Rn = Rn, LAI = LAI)
Z.om <- momentumRoughnessLength(LAI = LAI, mountainous = TRUE, method = "short.crops",
                                surface.model = surface.model)
anchors <- calcAnchors(image = image.TOAr, Ts = Ts, LAI = LAI, albedo = albedo, Z.om = Z.om,
                       WeatherStation = WeatherStation, plots = FALSE, verbose = FALSE)
H <- calcH(anchors = anchors, Ts = Ts, Z.om = Z.om, WeatherStation = WeatherStation,
           DEM = DEM, Rn = Rn, G = G, plots = FALSE, verbose = FALSE)
# the layers computed above
ET24h(Rn, G, H$H, Ts, WeatherStation = WeatherStation,
      ETr.daily = dailyET(WeatherStation, MTL = MTL))
```

`ETperiod` *Evapotranspiration of a period from the ETrF of several images*

Description

Seasonal or monthly ET from the reference ET fraction (ETrF) of several satellite dates, as in Allen et al. (2007), Eqs. 58 and 59: ETrF is interpolated for each day of the period, multiplied by the daily reference ET of the day and added up.

Usage

```
ETperiod(
  ETrF,
  dates,
  ETr.daily,
  period = NULL,
  C.rad = 1,
  method = c("linear", "spline"),
  fill.gaps = FALSE
)
```

Arguments

ETrF	SpatRaster with one layer per image date (the ETrF layer of ET24h of each image, on the same grid)
dates	Dates of the images (Date or "YYYY-MM-DD" text), one for each layer of ETrF
ETr.daily	Daily reference ET (mm/day) of every day of the period: a data.frame with the columns date and ET, as returned by dailyETseries
period	Start and end of the period (two dates, both included). Default: from the first to the last image date. Days before the first and after the last valid image of a pixel use its closest ETrF
C.rad	Correction for slope and aspect (Eq. 55): a number or a SpatRaster, e.g. the C.rad layer of ET24h; 1 on flat terrain. Only one value can be used for the period.
method	"linear" (default) or "spline" (natural spline, for pixels with 3 or more valid dates) interpolation of ETrF between dates
fill.gaps	If TRUE, days without ETr.daily (NA or missing) are filled by linear interpolation. If FALSE (default) they are an error.

Details

The ETrF of a pixel that is masked (NA, e.g. clouds) in one date is interpolated between the closest valid dates, and the number of valid dates is returned in layer n.valid. A pixel without any valid date is NA. ETrF is used as it comes from ET24h(): values below 0 or above 1.1 are not clipped; mask or clip them before if needed. A pixel with one valid date uses that ETrF for the whole period (n.valid = 1). The layers of ETrF must be on the same grid; their order does not matter (they are sorted by date). ET.period and ETrF.period both include C.rad.

Value

SpatRaster with layers ET.period (mm in the period), ETrF.period (Eq. 59, ET.period divided by the reference ET of the period, so it includes C.rad) and n.valid. On large scenes it can take minutes: the interpolation is done pixel by pixel

References

Allen, R.G., Tasumi, M., Trezza, R., 2007. Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model. Journal of Irrigation and Drainage Engineering 133, 380-394.

See Also

Other ET functions: Crad(), ET24h(), dailyET(), dailyETseries(), hourlyET()

Examples

```
# the ETrF of two images (a toy raster; in practice c(ET24h(EB1)$ETrF, ET24h(EB2)$ETrF))
f <- terra::rast(nrows = 2, ncols = 2, nlyrs = 2, vals = c(0.2, 0.4, 0.6, NA, 0.8, 0.6, 0.3, 0.5))
etr <- data.frame(date = as.Date("2013-02-01") + 0:9, ET = 6)
```

```
ETperiod(f, dates = c("2013-02-02", "2013-02-08"), ETr.daily = etr)
# with a weather station, dailyETseries() gives the daily reference ET:
# etr <- dailyETseries(WeatherStation)
```

`getDataWS` *Export data.frame from waterWeatherStation Object*

Description

Export weather conditions at the satellite overpass from a waterWeatherStation object

Usage

```
getDataWS(WeatherStation)
```

Arguments

`WeatherStation`	waterWeatherStation object. See `read.WSdata`

Value

A one-row data.frame with the conditions at the satellite overpass: wind (m/s), RH (%), temp (Celsius), radiation (W/m2), height (m), lat, long, elev (m), DOY (day of year), hours (decimal hour of the overpass in the weather station time zone) and long.z (longitude of the centre of that time zone, 15 degrees per hour of UTC offset, used by hourlyET()). An error is given if the object has no conditions at the overpass (at.sat).

Author(s)

Guillermo Federico Olmedo

See Also

Other Weather station related functions: `plot.waterWeatherStation()`, `print.waterWeatherStation()`, `read.WSdata()`, `read.WSdata2()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
getDataWS(WeatherStation)
```

getDEM *Download a digital elevation model for an image*

Description

Downloads the Copernicus DEM GLO-30 (30 m) tiles that cover the image and returns the DEM on the image grid, ready for METRIC.EB(plain = FALSE, DEM =). The tiles are Cloud Optimized GeoTIFFs in the AWS Open Data registry, read over HTTPS: no account is needed, and only the area of the image is downloaded. It needs an internet connection. To use SRTM tiles already downloaded, see checkSRTMgrids and prepareSRTMdata().

Usage

```
getDEM(image, source = "copernicus")
```

Arguments

image	SpatRaster (e.g. the Landsat image) that defines the extent, CRS and grid of the DEM
source	DEM source. Currently only "copernicus" (Copernicus DEM GLO-30) Tiles that do not exist (ocean) are skipped and reported, any other error (e.g. no internet connection) stops, and the share of NA cells is reported.

Value

SpatRaster with one layer named DEM: elevation (m, relative to the EGM2008 geoid) on the grid of image.

Author(s)

Guillermo Federico Olmedo

References

Copernicus DEM GLO-30: (c) DLR e.V. 2010-2014 and (c) Airbus Defence and Space GmbH 2014-2018, provided under COPERNICUS by the European Union and ESA; all rights reserved. Distributed through the Registry of Open Data on AWS, https://registry.opendata.aws/copernicus-dem/

See Also

Other support functions: checkSRTMgrids(), createAoi(), prepareSRTMdata(), waterOptions()

Examples

```
## Not run:
image.DN <- rast(L7_Talca)
DEM <- getDEM(image.DN)
plot(DEM)

## End(Not run)
```

`hourlyET` *Calculates hourly reference ET with the ASCE Penman-Monteith equation*

Description

Calculates the hourly (or instantaneous) reference evapotranspiration with the ASCE standardized Penman-Monteith equation, for a short (ETo, grass) or a tall (ETr, alfalfa) reference crop.

Usage

```
hourlyET(
  WeatherStation,
  hours,
  DOY,
  long.z = NULL,
  ET.instantaneous = FALSE,
  ET = "ETr",
  height = 2,
  lat,
  long,
  elev
)
```

Arguments

`WeatherStation`	data.frame with the needed fields (see example), or a waterWeatherStation object with the conditions at the satellite overpass; in that case they are used and ET.instantaneous is set to TRUE
`hours`	time of the day in decimal hours (24 h clock), e.g. 10.5 for 10:30
`DOY`	day of year
`long.z`	longitude of the centre of the time zone of the weather station clock, in decimal degrees (negative west), for the solar time correction. By default it is computed from the UTC offset of the station records (15 degrees per hour, e.g. -45 for UTC-3, including daylight saving time): from WeatherStation$long.z (see `getDataWS`) or from the time zone of WeatherStation$datetime. If the time zone is unknown, the station longitude is used (no correction)

`ET.instantaneous`
: Logical. TRUE to calculate instantaneous ET instead of hourly ET. See Details.

`ET`
: "ETo" for short crops, similar to clipped, cool-season grass; or "ETr" (default) for tall crops, similar to 0.5 m tall full-cover alfalfa.

`height`
: weather station sensors height in meters

`lat`
: latitude of weather station in decimal degrees. Negative values for south latitude

`long`
: longitude of weather station in decimal degrees. Negative values for west longitude

`elev`
: elevation of weather station in meters

Details

The only difference on instantaneous ET is how the hour is interpreted. With FALSE, and for example at 11:00, ET is calculated between 10:00 and 11:00; with TRUE, ET is calculated at 11:00.

The equations follow ASCE-EWRI (2005): Rs/Rso is limited to 0.3-1 (Eq. 45), the extraterrestrial radiation is integrated between sunrise and sunset (Eqs. 48 and 56-58), the daytime and nighttime coefficients Cd and G are chosen by Rn > 0 (Table 1), and T + 273 is used in the aerodynamic term and T + 273.16 in the net long-wave radiation. For a single period at night (sun angle below 0.3 rad) the cloudiness function cannot be computed and Rs/Rso = 0.3 is used; dailyET() carries the value of the last sunny period over the night instead (Eq. 46).

Value

Numeric. Hourly (or, with ET.instantaneous = TRUE, instantaneous) reference ET (mm/h).

Author(s)

Guillermo Federico Olmedo

References

ASCE-EWRI (2005). The ASCE Standardized Reference Evapotranspiration Equation. Report of the ASCE-EWRI Task Committee on Standardization of Reference Evapotranspiration.

See Also

Other ET functions: `Crad()`, `ET24h()`, `ETperiod()`, `dailyET()`, `dailyETseries()`

Examples

```
WeatherStation  <- data.frame(wind=4.72,
                              RH=59,
                              temp=24.3,
                              radiation=675,
                              height=2.2,
                              lat=-35.37,
                              long=-71.5946,
                              elev=124)
  hourlyET(WeatherStation, hours=10.5, DOY=363, long.z=-71.635)
```

`incLWradiation` *Calculates incoming long-wave radiation*

Description

This function estimates the incoming long-wave radiation using the Stefan-Boltzmann equation. In addition, the empirical equation of Bastiaanssen (1995) with coefficients developed by Allen (2000) is used to estimate the effective atmospheric emissivity.

Usage

```
incLWradiation(WeatherStation, DEM, solar.angles, Ts, Kt = 1)
```

Arguments

`WeatherStation`	waterWeatherStation object (see `read.WSdata`) or the data.frame returned by getDataWS()
`DEM`	SpatRaster with a digital elevation model (m)
`solar.angles`	SpatRaster with layers latitude, declination, hour.angle, incidence.hor and incidence.rel (radians), from solarAngles()
`Ts`	land surface temperature (K), from surfaceTemperature()
`Kt`	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See `SWtransmissivity`

Value

SpatRaster with incoming long-wave radiation (W/m2).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Bastiaanssen W. Regionalization of surface flux densities and moisture indicators in composite terrain: A remote sensing approach under clear skies in Mediterranean climates. Ph.D. dissertation, CIP Data Koninklijke Bibliotheek, Den Haag, The Netherlands, 1995, p273

Allen R. RAPID long-wave radiation calculations and model comparisons Internal report, University of Idaho, Kimberly, Idaho, 2000

See Also

Other net radiation related functions: LAI(), METRICtopo(), SWtransmissivity(), albedo(), incSWradiation(), netRadiation(), outLWradiation(), soilHeatFlux(), solarAngles(), surfaceTemperature()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
plot(Rl.inc)
```

---

incSWradiation *Calculates Incoming Solar Radiation*

Description

This function calculates incoming solar radiation from the surface model, the solar angles and the weather station data.

Usage

```
incSWradiation(surface.model, solar.angles, WeatherStation, Kt = 1)
```

Arguments

surface.model	SpatRaster with layers DEM, Slope and Aspect, from METRICtopo()
solar.angles	SpatRaster with layers latitude, declination, hour.angle, incidence.hor and incidence.rel (radians), from solarAngles()

WeatherStation	waterWeatherStation object (see read.WSdata) or the data.frame returned by getDataWS()
Kt	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See SWtransmissivity

Value

SpatRaster with incoming short-wave radiation (W/m2) at the time of the satellite overpass. Slopes facing away from the sun (cosine of the incidence angle below 0) get no direct radiation.

Author(s)

Guillermo Federico Olmedo

Daniel de la Fuente-Sáiz

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other net radiation related functions: LAI(), METRICtopo(), SWtransmissivity(), albedo(), incLWradiation(), netRadiation(), outLWradiation(), soilHeatFlux(), solarAngles(), surfaceTemperature()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
plot(Rs.inc)
```

L7_Talca *Landsat 7 scene from east of Talca, Chile*

Description

A Landsat 7 ETM+ subset (Level-1, digital numbers) stored as a terra `PackedSpatRaster`. Unpack it with `rast(L7_Talca)`. Layers are, in this order, B, G, R, NIR, SWIR1, Thermal1 (band 6, low gain) and SWIR2 (calcTOAr() and calcRadiance() select the bands by name).

Usage

```
L7_Talca
```

Format

A `PackedSpatRaster`; `rast(L7_Talca)` is a SpatRaster with 417 rows, 508 columns and 7 layers, 30 m resolution, WGS 84 / UTM zone 19S.

Details

Metadata for this image is also provided as system file. You can load it with: system.file("extdata", "L7.MTL.txt", package="water")

Data available from the U.S. Geological Survey.

Source

https://www.usgs.gov/

Examples

```
image.DN <- rast(L7_Talca)
names(image.DN)
```

LAI *Estimate LAI from Landsat Data*

Description

This function implements empirical models to estimate LAI (Leaf Area Index) from satellite images. Models were extracted from METRIC publications and other works developed on different crops.

Usage

```
LAI(image, method = "metric2010", L = 0.1, aoi, sat)
```

Arguments

image	SpatRaster with layers named R and NIR (a stack without standard names: the third and fourth layers, with a warning; a named stack without R or NIR is an error): top-of-atmosphere reflectance for "metric", "metric2010" and "MCB"; surface reflectance for "turner"; radiance for "vineyard" (see calcRadiance)
method	Method used to estimate LAI from spectral data: "metric2010" (default), "metric", "MCB", "vineyard" or "turner". See Details.
L	L factor used in method = "metric" or "metric2010" to estimate SAVI, defaults to 0.1
aoi	Not used (a warning is given when it is passed); kept for backward compatibility
sat	Not used; kept for backward compatibility

Details

LAI is computed from top-of-atmosphere reflectance ("metric", "metric2010" and "MCB"), surface reflectance ("turner") or radiance ("vineyard"). LAI and other indices such as NDVI and SAVI are used to describe the vegetation, depending on the preferences of the user. Available methods are: "metric", "metric2010", "MCB", "vineyard" and "turner".

"turner" inverts the relationship between surface reflectance NDVI and LAI of Turner et al. (1999, Table 3), NDVI = 0.5724 + 0.0989 LAI - 0.0114 LAI^2 + 0.0004 LAI^3, fitted for forest, grassland and shrub plots: LAI is 0 for NDVI below 0.572 and 6 for NDVI above 0.842.

When METRIC.EB(), METRIC.Rn() or METRIC.G() run without image.DN, the "metric", "metric2010" and "MCB" methods receive surface reflectance, although their equations were described for top-of-atmosphere reflectance.

Value

SpatRaster with leaf area index (m2/m2) in a layer named LAI; negative values are set to 0.

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

Fernando Fuentes-Peñailillo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Carrasco-Benavides, M., Ortega-Farias, S., Lagos, L., Kleissl, J., Morales-Salinas, L., & Kilic, A. (2014). Parameterization of the Satellite-Based Model (METRIC) for the Estimation of Instantaneous Surface Energy Balance Components over a Drip-Irrigated Vineyard. Remote Sensing, 6(11), 11342-11371. http://doi.org/10.3390/rs61111342

Johnson, L. F. (2003). Temporal stability of an NDVI-LAI relationship in a Napa Valley vineyard. Australian Journal of Grape and Wine Research, 9(2), 96-101. https://doi.org/10.1111/j.1755-0238.2003.tb00258.x

Turner, D. P., Cohen, W. B., Kennedy, R. E., Fassnacht, K. S., & Briggs, J. M. (1999). Relationships between leaf area index and Landsat TM spectral vegetation indices across three temperate zone sites. Remote Sensing of Environment, 70(1), 52–68. http://doi.org/10.1016/S0034-4257(99)00057-7

See Also

Other net radiation related functions: `METRICtopo()`, `SWtransmissivity()`, `albedo()`, `incLWradiation()`, `incSWradiation()`, `netRadiation()`, `outLWradiation()`, `soilHeatFlux()`, `solarAngles()`, `surfaceTemperature()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
plot(LAI)
```

`loadImage` *Load Landsat data from a folder*

Description

Loads Landsat 7, 8 or 9 Level-1 bands (digital numbers), or MODIS bands, from a folder. Landsat file names can follow the Collection 1/2 naming (e.g. `LC08_L1TP_232083_20160209_20200907_02_T1_B2.TIF`, any tier) or the pre-Collection naming (e.g. `LC82320832016040LGN00_band2.tif`); Level-2 surface reflectance files are skipped (use loadImageSR()). If the folder holds more than one image, the first one is used, with a warning. For Landsat, the fill value (DN 0) is set to NA.

Usage

```
loadImage(path = getwd(), sat = "auto", aoi)
```

Arguments

`path`	folder where band files are stored
`sat`	"L7" for Landsat 7, "L8" for Landsat 8, "L9" for Landsat 9, "MODIS" for MODIS or "auto" to detect the Landsat satellite from the SPACECRAFT_ID of an MTL file in path or, failing that, from the Landsat file names

aoi — Optional area of interest (e.g. from createAoi()) to crop the image. If missing, the image is not cropped

Value

SpatRaster in digital numbers. Landsat 8/9: layers B, G, R, NIR, SWIR1, SWIR2, Thermal1 and Thermal2 (bands 2 to 7, 10 and 11). Landsat 7: B, G, R, NIR, SWIR1, SWIR2 and Thermal1 (bands 1 to 5, 7 and 6 low gain). MODIS: R, NIR, B, G, SWIR1, SWIR2, SWIR3 (reflectance), LST (K) and Time (hours).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other remote sensing support functions: calcRadiance(), calcSR(), calcTOAr(), cfmask(), loadImageSR()

Examples

```
# Landsat 8 Level-1 example bands (digital numbers)
image.DN <- loadImage(path = system.file("extdata", package = "water"), sat = "L8")
image.DN
```

---

loadImageSR — *Load Landsat 8/9 surface reflectance (and surface temperature) from a folder*

Description

Loads Landsat 8 or 9 Collection 2 Level-2 surface reflectance bands 2 to 7 (*_SR_B2.TIF to *_SR_B7.TIF) from a folder, scaled as DN * 0.0000275 - 0.2. Pixels where any band is fill (0) or outside the valid range of the product (7273 to 43636 DN, reflectance 0 to 1) are set to NA in all layers, with a message giving the number of out-of-range pixels: slightly negative blue reflectance over dense crops is masked too. All six bands are required. If *_ST_B10.TIF is present it is added as the last layer, "Band_10", in Kelvin (DN * 0.00341802 + 149, fill value 0 set to NA). It can be passed as surf.temp to METRIC.EB(), METRIC.Rn() or METRIC.G(). Landsat 7 Level-2 products are not supported (different band numbers).

Usage

```
loadImageSR(path = getwd(), aoi)
```

Arguments

path	folder where band files are stored
aoi	Optional area of interest (e.g. from createAoi()) to crop the image. If missing, the image is not cropped

Value

SpatRaster with surface reflectance (unitless) in layers B, G, R, NIR, SWIR1 and SWIR2 (Landsat 8/9 bands 2 to 7). If `*_ST_B10.TIF` is present, a seventh layer Band_10 with surface temperature (K) is added.

Author(s)

Guillermo Federico Olmedo

See Also

Other remote sensing support functions: `calcRadiance()`, `calcSR()`, `calcTOAr()`, `cfmask()`, `loadImage()`

Examples

```
scene_folder <- system.file("extdata", "L8_C2", package = "water")
image.SR <- loadImageSR(path = scene_folder)
names(image.SR)
plot(image.SR[["NIR"]])
```

METRIC.EB *Estimates Energy Balance using METRIC2010 Model*

Description

Estimates the instantaneous land surface energy balance (Rn, G, H and LE) with the METRIC model (Allen et al., 2007). For Landsat 8 and 9 use Collection 2 Level-2 surface reflectance (image.SR) and, optionally, surface temperature (surf.temp), both from loadImageSR().

Usage

```
METRIC.EB(
  image.DN,
  image.SR,
  WeatherStation,
  MTL,
```

```
  sat = "auto",
  thermalband,
  plain = TRUE,
  DEM,
  aoi,
  G.method = "Tasumi",
  alb.coeff = "auto",
  LST.method = "auto",
  LAI.method = "metric2010",
  L = 0.1,
  Zom.method = "short.crops",
  anchors.method = "percentile",
  anchors,
  n = 1,
  ETp.coef = 1.05,
  Z.om.ws = 0.015,
  verbose = FALSE,
  extraParameters = vector(),
  surf.temp,
  Kt = 1,
  anchors.control = NULL,
  calcH.control = NULL,
  plots = interactive(),
  crop.height = NULL
)
```

Arguments

image.DN	Raw image in digital counts. Optional if image.SR is supplied. If supplied (L7/L8/L9) it is used for TOA reflectance (LAI and anchors) and for surface temperature; if missing, image.SR is used instead. Needed for LAI.method = "vineyard" and, without surf.temp, for MODIS (LST layer).
image.SR	Surface reflectance image with layers B, G, R, NIR, SWIR1 and SWIR2, e.g. from loadImageSR(). If supplied it is used directly; if missing it is computed from image.DN (L7 via calcSR(), MODIS uses image.DN). Required for L8 and L9. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
WeatherStation	waterWeatherStation object with the conditions at the satellite overpass (from read.WSdata() with the MTL file)
MTL	path to the Landsat metadata (*_MTL.txt) file. Required
sat	Satellite: "L7", "L8", "L9", "MODIS" or "auto" to detect it from the MTL file
thermalband	Thermal band in digital numbers (L7: band 6 low gain; L8/L9: band 10). If missing, image.DN$Thermal1 is used. Not needed if surf.temp is supplied
plain	Logical. If TRUE (default) the surface is assumed flat: DEM is replaced by a constant raster at the weather station elevation

DEM	SpatRaster with a digital elevation model (m) of the study area, on the same grid as the image. Not needed if plain = TRUE
aoi	Optional sf polygon or SpatVector. If given, the inputs are cropped to it
G.method	method used for the G estimation: "Tasumi" (Tasumi, 2003; default) or "Bastiaanssen" (Bastiaanssen, 2000). See soilHeatFlux
alb.coeff	coefficient to transform narrow to broad band albedo: "auto" (default; "Olmedo" for L8 and L9, "Tasumi" for L7 and MODIS), "Tasumi", "Olmedo" or "Liang". See Details.
LST.method	Land surface temperature when surf.temp is not supplied: "auto" (default) uses the Level-2 surface temperature (image.SR$Band_10 from loadImageSR()) for L8 and L9 when present (also when image.DN is given), and the METRIC equations (Allen et al., 2007, Eqs. 20-21) with image.DN otherwise; "L2" requires Band_10; "METRIC" uses the thermal band of image.DN; "SW" (split window, L8 and L9) is deprecated. For a per-pixel atmospheric correction, compute Ts with surfaceTemperature(atmosphere =) and pass it as surf.temp. See surfaceTemperature
LAI.method	Method used to estimate LAI from spectral data. See Details.
L	L value for SAVI calculation. See LAI
Zom.method	method selected to calculate momentum roughness length. Use "short.crops" for short crops methods from Allen et al (2007); "custom" for custom method also in Allen et al (2007); Or "Perrier" to use Perrier equation as in Santos et al (2012) and Pocas et al (2014).
anchors.method	method for the automatic selection of the anchor pixels: "percentile" (default), "flexible", "best" or "random". See calcAnchors
anchors	data.frame with X, Y (coordinates in the CRS of the image) and type ("hot" or "cold") columns, e.g. from calcAnchors(), or an sf POINT object with a type column. If missing, the anchors are selected automatically with calcAnchors(). They are checked by calcH(): type "hot" or "cold" (case ignored), inside the image, inputs not NA there, and the hot anchors at least 5 K warmer than the cold ones
n	number of pairs of anchor pixels to select (not used by anchors.method = "percentile")
ETp.coef	ratio of the ET of the cold pixels to the instantaneous alfalfa reference ET (ETr) at the satellite overpass. Allen et al. (2007) use 1.05, the default. It can be updated with anchors.method = "flexible", or for every method with calcH.control(cold.NDVI.rule = TRUE) (see Details)
Z.om.ws	momentum roughness length (m) of the weather station surface, used to extrapolate the wind speed to 200 m (Allen et al., 2007, Eq. 32). It is about 0.123 times the height of the vegetation around the station (Allen et al., 1998): 0.015 m (the default) for a standard station over clipped grass 0.12 m tall, 0.03 m for vegetation about 0.25 m tall
verbose	Logical. If TRUE, the anchors and other details are shown (as messages)
extraParameters	named numeric vector with extra parameters for the non default methods, e.g. c(fLAI = 0.5, h = 1.8) for Zom.method = "Perrier". See momentumRoughnessLength and the Extra Parameters section.

`surf.temp` Optional. Surface temperature raster in Kelvin, on the same grid as image.SR (e.g. image.SR$Band_10 from loadImageSR()). If supplied it is used as Ts and surfaceTemperature() is not called. Must be in Kelvin: values that look like Celsius, or more than 1% of the cells below 200 K (unmasked fill or clouds), stop; fewer cells below 200 K give a warning.

`Kt` clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See `SWtransmissivity`

`anchors.control` parameters of the automatic selection of anchor pixels: a list from `anchors.control` (or a plain list with some of its elements)

`calcH.control` parameters of the iteration of H: a list from `calcH.control` (or a plain list with some of its elements)

`plots` Logical. If TRUE the anchors and the iteration of H are plotted (default: only in interactive sessions)

`crop.height` Only for Zom.method = "Perrier": crop height in m, a number or a SpatRaster on the grid of the image (e.g. rasterized from a field map), instead of h in extraParameters. Cells with 0 or NA (bare soil, roads) get the minimum Z.om of 0.005 m. Default NULL.

Details

There are different models to convert narrowband data to broadband albedo. With alb.coeff = "auto" (default) the coefficients are Olmedo for Landsat 8 and 9, Tasumi for Landsat 7 and MODIS. You can choose alb.coeff = "Tasumi" to use Tasumi et al. (2008) coefficients, calculated for Landsat 7; alb.coeff = "Liang" to use Liang (2001) Landsat 7 coefficients or alb.coeff = "Olmedo" to use Olmedo coefficients for Landsat 8 and 9. See `albedo`.

LAI.method can be "metric2010" (default), "metric", "MCB", "turner" or "vineyard". See `LAI`. Without image.DN, the leaf area index is computed and the anchor pixels are selected from the surface reflectance (image.SR), although the "metric", "metric2010" and "MCB" LAI equations were described for top-of-atmosphere reflectance.

The ET of the cold pixels is ETp.coef times the instantaneous alfalfa reference ET (ETr, ASCE standardized equation) at the satellite overpass (see `calcH`), the same reference used by ET24h() and dailyET(), as in Allen et al. (2007). With anchors.method = "flexible", if the cold-pixel criteria had to be relaxed and the mean NDVI of the cold pixels is below 0.8, ETp.coef is replaced by Kc * ETo / ETr, where Kc = 2.11 * NDVI - 0.4989 is a crop coefficient on a grass reference basis (Reyes-González et al., 2015) and ETo / ETr is the ratio of the grass and alfalfa reference ET at the overpass. With calcH.control(cold.NDVI.rule = TRUE), automatic cold anchors with mean NDVI below 0.8 get min(ETp.coef, Kc * ETo / ETr) for every anchors.method, instead of the "flexible" rule.

METRIC.EB() gives a message when the scene has clouds (CLOUD_COVER in the MTL file) and the image was not masked with cfmask(), and a warning when the weather station recorded more than 5 mm of rain in the 3 days before the image (the hot anchors may not be dry; see calcH.control(ETrF.hot =)).

Method arguments (sat, alb.coeff, LAI.method, LST.method and, where they apply, G.method, Zom.method and anchors.method) must be one of the listed names, exactly (case included): an unknown name stops before any computation, with a hint for typos.

Value

An object of class `waterLSEB`, a list with the elements below. A message summarises the calibration, and a warning is given when ETrF is above 1.1 on more than 10% of the pixels (cold anchors too warm):

EB SpatRaster with layers NetRadiation, SoilHeat, SensibleHeat and LatentHeat (W/m2), and surfaceTemperature (K).

WeatherStation the weather station data used.

anchors sf POINT object with the hot and cold anchor pixels.

calibration list with a and b (dT = a * Ts_datum + b), converged, iterations and u200 (wind speed at 200 m used, m/s), from calcH(); for automatic anchors, the number of candidates and the relaxation of the criteria, from calcAnchors(); and quality checks: Ts.hot and Ts.cold (mean Ts of the anchors, K, referred to the station elevation with plain = FALSE), dT.anchors (their difference), ETrF (5%, 50% and 95% quantiles and maximum of the reference ET fraction at the overpass, before LE is set to 0), ETrF.above.1.1, ETrF.below.0 and H.above.Rn.G (shares of the pixels). See `summary.waterLSEB`.

terrain with plain = FALSE, SpatRaster with the surface model and the solar angles, for the slope correction of daily ET (see `Crad` and `ET24h`).

methods named character vector with the methods and parameters used.

Extra Parameters

extraParameters must be a named numeric vector with these names (unknown names stop). a and b are required for Zom.method = "custom", and fLAI and h for "Perrier":

- for momentumRoughnessLength() when Zom.method = "Perrier": fLAI, h (numbers or SpatRasters);
- for momentumRoughnessLength() when Zom.method = "custom": a, b;
- deprecated: minDist, WSbuffer and deltaTemp for calcAnchors(); use anchors.control = anchors.control(...) instead.

Author(s)

Guillermo Federico Olmedo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop evapotranspiration: guidelines for computing crop water requirements. FAO Irrigation and Drainage Paper 56. FAO, Rome.

Reyes-González, A., Hay, C., Kjaersgaard, J., & Neale, C. M. U. (2015). Use of remote sensing to generate crop coefficient and estimate actual crop evapotranspiration. 2015 ASABE Annual International Meeting, New Orleans, Louisiana. https://doi.org/10.13031/aim.20152190105

See Also

Other METRIC model functions: METRIC.G(), METRIC.Rn(), anchors.control(), calcH.control()

Examples

```
### Data preparation: Landsat 8 Collection 2 Level-2 example crop
library(water)
scene_folder <- system.file("extdata", "L8_C2", package = "water")
image.SR <- loadImageSR(path = scene_folder)
image.SR <- cfmask(path = scene_folder, image = image.SR)
MTLfile <- list.files(scene_folder, pattern = "MTL.txt$", full.names = TRUE)
csvfile <- system.file("extdata", "INTA.csv", package = "water")
WeatherStation <- read.WSdata(WSdata = csvfile, datetime.format = "%Y/%m/%d %H:%M",
                              columns = c("date" = 1, "time" = 1, "radiation" = 5,
                                          "wind" = 6, "RH" = 3, "temp" = 2, "rain" = 4),
                              lat = -33.00513, long = -68.86469, elev = 927, height = 2,
                              tz = "America/Argentina/Mendoza", MTL = MTLfile)

### LSEB with surface reflectance and surface temperature
Energy.Balance <- METRIC.EB(image.SR = image.SR, surf.temp = image.SR$Band_10,
                            WeatherStation = WeatherStation, MTL = MTLfile,
                            alb.coeff = "Olmedo", n = 5, Z.om.ws = 0.03)

### LSEB with "Perrier" method for Zom and extra parameters
Energy.Balance <- METRIC.EB(image.SR = image.SR, surf.temp = image.SR$Band_10,
                            WeatherStation = WeatherStation, MTL = MTLfile,
                            alb.coeff = "Olmedo", n = 5, Z.om.ws = 0.03,
                            Zom.method = "Perrier",
                            extraParameters = c(fLAI = 0.5, h = 1.8))
```

METRIC.G *Estimates Soil Heat Flux as in METRIC Model*

Description

Estimates the soil heat flux (G) as in the METRIC model. Inputs are the same as in METRIC.Rn(). For Landsat 8 and 9, use Collection 2 Level-2 surface reflectance as image.SR and surface temperature as surf.temp.

Usage

```
METRIC.G(
  image.DN,
  image.SR,
  WeatherStation,
  MTL,
  sat = "auto",
  thermalband,
```

```
  plain = TRUE,
  DEM,
  aoi,
  G.method = "Tasumi",
  alb.coeff = "auto",
  LST.method = "auto",
  LAI.method = "metric2010",
  L = 0.1,
  surf.temp,
  Kt = 1,
  Rn,
  plots = interactive()
)
```

Arguments

image.DN	Raw image in digital counts. Optional if image.SR and surf.temp are supplied. If supplied (L7/L8/L9) it is used for TOA reflectance (LAI) and, without surf.temp, for surface temperature.
image.SR	Surface reflectance image with layers B, G, R, NIR, SWIR1 and SWIR2, e.g. from loadImageSR(). Required for L8 and L9; for L7 it is computed from image.DN if missing. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
WeatherStation	waterWeatherStation object with the conditions at the satellite overpass (from read.WSdata() with the MTL file)
MTL	path to the Landsat metadata (*_MTL.txt) file. Required
sat	Satellite: "L7", "L8", "L9", "MODIS" or "auto" to detect it from the MTL file
thermalband	Thermal band in digital numbers (L7: band 6 low gain; L8/L9: band 10). If missing, image.DN$Thermal1 is used. Not needed if surf.temp is supplied
plain	Logical. If TRUE (default) the surface is assumed flat: DEM is replaced by a constant raster at the weather station elevation
DEM	SpatRaster with a digital elevation model (m) of the study area, on the same grid as the image. Not needed if plain = TRUE
aoi	Optional sf polygon or SpatVector. If given, the inputs are cropped to it
G.method	method used for the G estimation: "Tasumi" (Tasumi, 2003; default) or "Bastiaanssen" (Bastiaanssen, 2000). See soilHeatFlux
alb.coeff	coefficient to transform narrow to broad band albedo: "auto" (default; "Olmedo" for L8 and L9, "Tasumi" for L7 and MODIS), "Tasumi", "Olmedo" or "Liang". See Details.
LST.method	Land surface temperature when surf.temp is not supplied: "auto" (default) uses the Level-2 surface temperature (image.SR$Band_10 from loadImageSR()) for L8 and L9 when present (also when image.DN is given), and the METRIC equations (Allen et al., 2007, Eqs. 20-21) with image.DN otherwise; "L2" requires

	Band_10; "METRIC" uses the thermal band of image.DN; "SW" (split window, L8 and L9) is deprecated. For a per-pixel atmospheric correction, compute Ts with surfaceTemperature(atmosphere =) and pass it as surf.temp. See surfaceTemperature
LAI.method	Method used to estimate LAI from spectral data. See Details.
L	L value for SAVI calculation. See LAI
surf.temp	Optional. Surface temperature raster in Kelvin, on the same grid as image.SR (e.g. image.SR$Band_10 from loadImageSR()). If supplied it is used as Ts and surfaceTemperature() is not called. Must be in Kelvin: values that look like Celsius, or more than 1% of the cells below 200 K (unmasked fill or clouds), stop; fewer cells below 200 K give a warning.
Kt	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See SWtransmissivity
Rn	Net radiation in W/m2, e.g. from METRIC.Rn(). If missing it is computed as in METRIC.Rn()
plots	Logical. If TRUE the results are plotted (default: only in interactive sessions)

Details

There are different models to convert narrowband data to broadband albedo. With alb.coeff = "auto" (default) the coefficients are Olmedo for Landsat 8 and 9, Tasumi for Landsat 7 and MODIS. You can choose alb.coeff = "Tasumi" to use Tasumi et al. (2008) coefficients, calculated for Landsat 7; alb.coeff = "Liang" to use Liang (2001) Landsat 7 coefficients or alb.coeff = "Olmedo" to use Olmedo coefficients for Landsat 8 and 9. See albedo.

LAI.method can be "metric2010" (default), "metric", "MCB", "turner" or "vineyard". See LAI. Without image.DN, the leaf area index is computed from the surface reflectance (image.SR), although the "metric", "metric2010" and "MCB" equations were described for top-of-atmosphere reflectance.

Method arguments (sat, alb.coeff, LAI.method, LST.method and, where they apply, G.method, Zom.method and anchors.method) must be one of the listed names, exactly (case included): an unknown name stops before any computation, with a hint for typos.

Value

SpatRaster with soil heat flux G (W/m2). Plotted with plots = TRUE.

Author(s)

Guillermo Federico Olmedo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other METRIC model functions: METRIC.EB(), METRIC.Rn(), anchors.control(), calcH.control()

Examples

```
scene_folder <- system.file("extdata", "L8_C2", package = "water")
image.SR <- loadImageSR(path = scene_folder)
MTLfile <- list.files(scene_folder, pattern = "MTL.txt$", full.names = TRUE)
csvfile <- system.file("extdata", "INTA.csv", package = "water")
WeatherStation <- read.WSdata(WSdata = csvfile, datetime.format = "%Y/%m/%d %H:%M",
                              columns = c("date" = 1, "time" = 1, "radiation" = 5,
                                          "wind" = 6, "RH" = 3, "temp" = 2, "rain" = 4),
                              lat = -33.00513, long = -68.86469, elev = 927, height = 2,
                              tz = "America/Argentina/Mendoza", MTL = MTLfile)
G <- METRIC.G(image.SR = image.SR, surf.temp = image.SR$Band_10,
              WeatherStation = WeatherStation, MTL = MTLfile,
              alb.coeff = "Olmedo")
```

METRIC.Rn *Estimates Net Radiation as in METRIC Model*

Description

Estimates the net radiation (Rn) as in the METRIC model, with the same inputs and steps as METRIC.EB(). For Landsat 8 and 9, use Collection 2 Level-2 products: surface reflectance as image.SR and surface temperature as surf.temp (both from loadImageSR()). For Landsat 7, image.SR and Ts can also be computed from image.DN.

Usage

```
METRIC.Rn(
  image.DN,
  image.SR,
  WeatherStation,
  MTL,
  sat = "auto",
  thermalband,
  plain = TRUE,
  DEM,
  aoi,
  alb.coeff = "auto",
  LST.method = "auto",
  LAI.method = "metric2010",
  L = 0.1,
  surf.temp,
  Kt = 1,
  plots = interactive()
)
```

Arguments

image.DN	Raw image in digital counts. Optional if image.SR and surf.temp are supplied. If supplied (L7/L8/L9) it is used for TOA reflectance (LAI) and, without surf.temp, for surface temperature.
image.SR	Surface reflectance image with layers B, G, R, NIR, SWIR1 and SWIR2, e.g. from loadImageSR(). Required for L8 and L9; for L7 it is computed from image.DN if missing. Layers are selected by name; a stack without the standard names is used by position, with a warning, and a stack with these names that lacks a needed layer is an error.
WeatherStation	waterWeatherStation object with the conditions at the satellite overpass (from read.WSdata() with the MTL file)
MTL	path to the Landsat metadata (*_MTL.txt) file. Required
sat	Satellite: "L7", "L8", "L9", "MODIS" or "auto" to detect it from the MTL file
thermalband	Thermal band in digital numbers (L7: band 6 low gain; L8/L9: band 10). If missing, image.DN$Thermal1 is used. Not needed if surf.temp is supplied
plain	Logical. If TRUE (default) the surface is assumed flat: DEM is replaced by a constant raster at the weather station elevation
DEM	SpatRaster with a digital elevation model (m) of the study area, on the same grid as the image. Not needed if plain = TRUE
aoi	Optional sf polygon or SpatVector. If given, the inputs are cropped to it
alb.coeff	coefficient to transform narrow to broad band albedo: "auto" (default; "Olmedo" for L8 and L9, "Tasumi" for L7 and MODIS), "Tasumi", "Olmedo" or "Liang". See Details.
LST.method	Land surface temperature when surf.temp is not supplied: "auto" (default) uses the Level-2 surface temperature (image.SR$Band_10 from loadImageSR()) for L8 and L9 when present (also when image.DN is given), and the METRIC equations (Allen et al., 2007, Eqs. 20-21) with image.DN otherwise; "L2" requires Band_10; "METRIC" uses the thermal band of image.DN; "SW" (split window, L8 and L9) is deprecated. For a per-pixel atmospheric correction, compute Ts with surfaceTemperature(atmosphere =) and pass it as surf.temp. See surfaceTemperature
LAI.method	Method used to estimate LAI from spectral data. See Details.
L	L value for SAVI calculation. See LAI
surf.temp	Optional. Surface temperature raster in Kelvin, on the same grid as image.SR (e.g. image.SR$Band_10 from loadImageSR()). If supplied it is used as Ts and surfaceTemperature() is not called. Must be in Kelvin: values that look like Celsius, or more than 1% of the cells below 200 K (unmasked fill or clouds), stop; fewer cells below 200 K give a warning.
Kt	clearness (turbidity) coefficient of the atmosphere, 0 < Kt <= 1: 1 (default) for clean air, 0.5 for extremely turbid, dusty or polluted air (Allen et al., 2007). See SWtransmissivity
plots	Logical. If TRUE the results are plotted (default: only in interactive sessions)

Details

There are different models to convert narrowband data to broadband albedo. With alb.coeff = "auto" (default) the coefficients are Olmedo for Landsat 8 and 9, Tasumi for Landsat 7 and MODIS. You can choose alb.coeff = "Tasumi" to use Tasumi et al. (2008) coefficients, calculated for Landsat 7; alb.coeff = "Liang" to use Liang (2001) Landsat 7 coefficients or alb.coeff = "Olmedo" to use Olmedo coefficients for Landsat 8 and 9. See albedo.

LAI.method can be "metric2010" (default), "metric", "MCB", "turner" or "vineyard". See LAI. Without image.DN, the leaf area index is computed from the surface reflectance (image.SR), although the "metric", "metric2010" and "MCB" equations were described for top-of-atmosphere reflectance.

Method arguments (sat, alb.coeff, LAI.method, LST.method and, where they apply, G.method, Zom.method and anchors.method) must be one of the listed names, exactly (case included): an unknown name stops before any computation, with a hint for typos.

Value

SpatRaster with net radiation (W/m2). Plotted with plots = TRUE.

Author(s)

Guillermo Federico Olmedo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other METRIC model functions: METRIC.EB(), METRIC.G(), anchors.control(), calcH.control()

Examples

```
scene_folder <- system.file("extdata", "L8_C2", package = "water")
image.SR <- loadImageSR(path = scene_folder)
MTLfile <- list.files(scene_folder, pattern = "MTL.txt$", full.names = TRUE)
csvfile <- system.file("extdata", "INTA.csv", package = "water")
WeatherStation <- read.WSdata(WSdata = csvfile, datetime.format = "%Y/%m/%d %H:%M",
                              columns = c("date" = 1, "time" = 1, "radiation" = 5,
                                          "wind" = 6, "RH" = 3, "temp" = 2, "rain" = 4),
                              lat = -33.00513, long = -68.86469, elev = 927, height = 2,
                              tz = "America/Argentina/Mendoza", MTL = MTLfile)
Rn <- METRIC.Rn(image.SR = image.SR, surf.temp = image.SR$Band_10,
                WeatherStation = WeatherStation, MTL = MTLfile,
                alb.coeff = "Olmedo")
```

METRICtopo *Calculates surface model used in METRIC*

Description

The DEM is used to generate the surface representation of the image through aspect and slope maps. This procedure helps to avoid differences in the surface temperature (and finally evapotranspiration) caused by different incidence angles and/or elevations in mountainous areas. A constant DEM (plain = TRUE in METRIC.EB()) gives a flat surface everywhere, without an NA border.

Usage

```
METRICtopo(DEM)
```

Arguments

DEM	SpatRaster with a digital elevation model (m)

Value

SpatRaster with layers DEM (m), Slope and Aspect (radians; Aspect is shifted by -pi, as METRIC expects).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other net radiation related functions: LAI(), SWtransmissivity(), albedo(), incLWradiation(), incSWradiation(), netRadiation(), outLWradiation(), soilHeatFlux(), solarAngles(), surfaceTemperature()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
```

```
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
plot(surface.model)
```

---

momentumRoughnessLength

Calculates Momentum Roughness Length

Description

Estimates the momentum roughness length (Zom, m) for each pixel from LAI ("short.crops"), from NDVI and albedo ("custom") or from LAI and crop height ("Perrier").

Usage

```
momentumRoughnessLength(
  method = "short.crops",
  LAI,
  NDVI,
  albedo,
  a,
  b,
  fLAI,
  h,
  mountainous = FALSE,
  surface.model,
  slope.min1 = FALSE
)
```

Arguments

method	method selected to calculate momentum roughness length. Use "short.crops" for short crops methods from Allen et al (2007); "custom" for custom method also in Allen et al (2007); Or "Perrier" to use Perrier equation as in Santos et al (2012) and Pocas et al (2014).
LAI	SpatRaster with leaf area index, from LAI(). Needed for method = "short.crops" and "Perrier"
NDVI	SpatRaster with the Normalized Difference Vegetation Index. Only needed for method = "custom"
albedo	broadband surface albedo, from albedo(). Only needed for method = "custom"
a	"a" coefficient for Allen (2007) custom function to estimate Momentum roughness length. Only needed for method = "custom"
b	"b" coefficient for Allen (2007) custom function to estimate Momentum roughness length. Only needed for method = "custom"

fLAI	proportion of LAI lying above h/2: a number or a SpatRaster on the grid of LAI. Only needed for method = "Perrier"
h	crop height in meters: a number or a SpatRaster on the grid of LAI (e.g. rasterized from a field map; cells with 0 or NA, such as bare soil, get the minimum Z.om of 0.005 m). Only needed for method = "Perrier"
mountainous	Logical. If TRUE applies the empirical adjustment for the effects of general terrain roughness on momentum and heat transfer (Allen et al., 2007); needs surface.model
surface.model	SpatRaster from METRICtopo() with a layer named Slope. Only needed if mountainous = TRUE
slope.min1	logical. If TRUE, the terrain factor 1 + (slope - 5) / 20 is at least 1 (METRIC manual), so slopes below 5 degrees do not lower Z.om. See calcH.control

Details

According to Allen et al. (2010), Zom is a measure of the form drag and skin friction for the layer of air that interacts with the surface. Values below 0.005 m (bare agricultural soil, Allen et al., 2007) are set to 0.005 m, for every method.

Value

SpatRaster with the momentum roughness length (m).

Author(s)

Guillermo Federico Olmedo

Daniel de la Fuente-Sáiz

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Pocas, I., Paco, T.A., Cunha, M., Andrade, J.A., Silvestre, J., Sousa, A., Santos, F.L., Pereira, L.S., Allen, R.G., 2014. Satellite-based evapotranspiration of a super-intensive olive orchard: Application of METRIC algorithms. Biosystems Engineering 128, 69-81. doi:10.1016/j.biosystemseng.2014.06.019

Santos, C., Lorite, I.J., Allen, R.G., Tasumi, M., 2012. Aerodynamic Parameterization of the Satellite-Based Energy Balance (METRIC) Model for ET Estimation in Rainfed Olive Orchards of Andalusia, Spain. Water Resour Manage 26, 3267-3283. doi:10.1007/s11269-012-0071-8

See Also

Other sensible heat flux functions: calcAnchors(), calcH()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Z.om <- momentumRoughnessLength(LAI = LAI, mountainous = TRUE, method = "short.crops",
                                surface.model = surface.model)
plot(Z.om)
```

---

netRadiation *Estimates net radiation*

Description

This function estimates net radiation considering a surface radiation balance. Equations use the information from the image, in addition to measurements of actual vapor pressure and altitude.

Usage

```
netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
```

Arguments

LAI	SpatRaster with leaf area index, from LAI()
albedo	broadband surface albedo, from albedo()
Rs.inc	incoming short-wave radiation (W/m2), from incSWradiation()
Rl.inc	incoming long-wave radiation (W/m2), from incLWradiation()
Rl.out	outgoing long-wave radiation (W/m2), from outLWradiation()

Value

SpatRaster with net radiation (W/m2).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other net radiation related functions: `LAI()`, `METRICtopo()`, `SWtransmissivity()`, `albedo()`, `incLWradiation()`, `incSWradiation()`, `outLWradiation()`, `soilHeatFlux()`, `solarAngles()`, `surfaceTemperature()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
plot(Rn)
```

`outLWradiation` *Calculates outgoing long-wave radiation*

Description

This function estimates the outgoing long-wave radiation using the Stefan-Boltzmann equation.

Usage

```
outLWradiation(LAI, Ts)
```

Arguments

LAI	SpatRaster with leaf area index, from LAI()
Ts	land surface temperature (K), from surfaceTemperature()

Value

SpatRaster with outgoing long-wave radiation (W/m2).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other net radiation related functions: LAI(), METRICtopo(), SWtransmissivity(), albedo(), incLWradiation(), incSWradiation(), netRadiation(), soilHeatFlux(), solarAngles(), surfaceTemperature()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
plot(Rl.out)
```

`plot.waterLSEB` *Plot method for waterLSEB S3 class*

Description

Plot method for waterLSEB S3 class

Usage

```
## S3 method for class 'waterLSEB'
plot(x, ...)
```

Arguments

`x`	waterLSEB object.
`...`	further arguments passed to or from other methods.

Value

No return value, called for side effects (plots the NetRadiation, SoilHeat, SensibleHeat and LatentHeat layers of x$EB).

Author(s)

Guillermo Federico Olmedo

See Also

Other LSEB objects related functions: `plotAnchors()`, `print.waterLSEB()`, `summary.waterLSEB()`, `writeRaster.waterLSEB`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
plot(Energy.Balance)
```

plot.waterWeatherStation *Plot method for waterWeatherStation S3 class*

Description

Plot method for waterWeatherStation S3 class

Usage

```
## S3 method for class 'waterWeatherStation'
plot(x, hourly = FALSE, sat = TRUE, date, ...)
```

Arguments

x	waterWeatherStation object. See read.WSdata
hourly	If TRUE will plot only hourly data, instead of all data from the waterWeatherStation object
sat	If TRUE, and if the waterWeatherStation object was created using a Landsat Metadata File, will plot only data from day of the satellite overpass
date	When sat = FALSE, the date to plot in a multi-day object
...	additional parameters to pass to plot()

Value

No return value, called for side effects (plots solar radiation, wind speed, temperature, vapor pressure and, if available, rain).

Author(s)

Guillermo Federico Olmedo

See Also

Other Weather station related functions: getDataWS(), print.waterWeatherStation(), read.WSdata(), read.WSdata2()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
plot(WeatherStation, hourly = TRUE)
```

plotAnchors *Plot the anchor pixels of a METRIC energy balance*

Description

Plots the hot (red) and cold (blue) anchor pixels of a METRIC.EB() result, and the weather station (black), over a false-colour composite of the image (NIR, R, G) or over the surface temperature.

Usage

```
plotAnchors(x, image.SR, ...)
```

Arguments

x	waterLSEB object, from METRIC.EB().
image.SR	Optional. Reflectance image with layers NIR, R and G, e.g. from loadImageSR(). If missing, the surface temperature is plotted.
...	further arguments passed to terra::plotRGB() or plot().

Value

No return value, called for its side effect (a plot).

Author(s)

Guillermo Federico Olmedo

See Also

Other LSEB objects related functions: plot.waterLSEB(), print.waterLSEB(), summary.waterLSEB(), writeRaster.waterLSEB

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
plotAnchors(Energy.Balance)
```

prepareSRTMdata *Create a DEM mosaic from SRTM grids for the image extent*

Description

Mosaics the SRTM 1 arc-second tiles found in path (see checkSRTMgrids) and projects the result to the grid of the image.

Usage

```
prepareSRTMdata(path = getwd(), format = "tif", extent)
```

Arguments

path	folder where SRTM files are stored
format	file extension of the SRTM grid files (default "tif"); files must be named like s34_w069_1arc_v3.tif
extent	SpatRaster (e.g. the Landsat image) whose grid is used for the output DEM, or an sf / SpatVector polygon with a CRS, in which case a 30 m grid in that CRS is used

Value

SpatRaster with one layer named DEM (m), on the grid defined by extent.

Author(s)

Guillermo Federico Olmedo

See Also

Other support functions: checkSRTMgrids(), createAoi(), getDEM(), waterOptions()

Examples

```
image.DN <- terra::rast(L7_Talca)
# tiles needed to cover the image (to download from EarthExplorer)
checkSRTMgrids(image.DN)
## Not run:
# with the tiles in a folder: mosaic and crop them to the grid of the image
DEM <- prepareSRTMdata(path = "folder/with/the/SRTM/tiles", extent = image.DN)

## End(Not run)
```

`print.waterLSEB` *Print method for waterLSEB S3 class*

Description

Print method for waterLSEB S3 class

Usage

```
## S3 method for class 'waterLSEB'
print(x, ...)
```

Arguments

`x`	waterLSEB object.
`...`	further arguments passed to or from other methods.

Value

x, invisibly (called for its side effect of printing the energy balance layers, the weather station, the anchor pixels, the methods and the calibration summary, see `summary.waterLSEB`).

Author(s)

Guillermo Federico Olmedo

See Also

Other LSEB objects related functions: `plot.waterLSEB()`, `plotAnchors()`, `summary.waterLSEB()`, `writeRaster.waterLSEB`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
print(Energy.Balance)
```

print.waterWeatherStation *Print method for waterWeatherStation S3 class*

Description

Print method for waterWeatherStation S3 class

Usage

```
## S3 method for class 'waterWeatherStation'
print(x, ...)
```

Arguments

x	waterWeatherStation object. See read.WSdata
...	additional parameters to pass to print()

Value

Called for its side effect: prints the station location, a summary of the records and, if available, the conditions at the satellite overpass. Returns x invisibly.

Author(s)

Guillermo Federico Olmedo

See Also

Other Weather station related functions: getDataWS(), plot.waterWeatherStation(), read.WSdata(), read.WSdata2()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
print(WeatherStation)
```

`read.WSdata` *Prepares weather station data*

Description

Reads weather station records from a csv file or a data.frame, interpolates them every hour and, with a Landsat metadata (MTL) file, at the time of the satellite overpass.

Usage

```
read.WSdata(
  WSdata,
  ...,
  height = 2.2,
  lat,
  long,
  elev,
  columns = c(date = 1, time = 2, radiation = 3, wind = 4, RH = 6, temp = 7, rain = 8),
  date.format = "%Y-%m-%d",
  time.format = "%H:%M:%S",
  datetime.format = "%Y-%m-%d %H:%M:%S",
  tz = "",
  cf = c(1, 1, 1),
  MTL,
  timestamps = "instant",
  overpass.radiation = "measured"
)
```

Arguments

`WSdata`	csv file with weather station data or data.frame
`...`	additional parameters passed to read.csv()
`height`	weather station sensors height in meters
`lat`	latitude of weather station in decimal degrees. Negative values for south latitude
`long`	longitude of weather station in decimal degrees. Negative values for west longitude
`elev`	elevation of weather station in meters
`columns`	numeric vector with the column numbers in WSdata of the date, time, radiation, wind, RH, temperature and rain. If date and time are in the same column, give the same number twice (or use datetime). A named vector is matched by name, in any order: date and time (or datetime), radiation, wind, RH, temp and, optionally, rain. An unnamed vector is taken in that order. Unknown or repeated names stop; pp is accepted for rain. A warning is given when the names give the vector another meaning than its order (before version 1.0 only the order was used).

`date.format` date format. See strptime format argument.

`time.format` time format. See strptime format argument.

`datetime.format`
datetime format. See strptime format argument.

`tz` time zone of the weather station clock (e.g. "America/Santiago"; see `OlsonNames`). If "" (default) the time zone of the computer is assumed, with a warning. Always set it: the overpass time in the MTL file is UTC and is converted to this time zone. See strptime for details.

`cf` conversion factor to convert radiation, wind, and temperature to W/m2; m/s and Celsius. See Details.

`MTL` path to the Landsat metadata (MTL) file. When given, the weather conditions at the satellite overpass are interpolated (element at.sat); if the overpass is outside the records, a warning is given and at.sat is not computed. If missing, at.sat is not computed (with a message).

`timestamps` "instant" (default) if each record is the value at its time; "end" or "start" if it is the mean over the period ending or starting at its time (e.g. hourly means). For period means, the conditions at the overpass are interpolated between the middles of the periods. The hourly and daily values are not shifted: hourlyET() and dailyET() already take hourly values as means over the hour ending at their time.

`overpass.radiation`
"measured" (default) uses the station radiation at the overpass; "clear-sky" uses the ASCE-EWRI (2005) clear-sky radiation instead (the measured value is kept in at.sat$radiation.measured), e.g. when clouds shade the station but not the image. calcH() (ET of the cold pixels) and ET24h() then use the same clear-sky reference ET. See Details.

Details

For cf, if your data is in W/m2, km/h and Celsius (radiation, wind, temperature), cf should be: cf = c(1,0.2777778,1)

The records are sorted by time. Exact duplicates are removed, and records with the same time and different values (e.g. the repeated hour when daylight saving time ends, within one clock hour) are averaged, with a warning. Many conflicting times outside one clock hour stop (a logger fault).

The values (after cf) are checked. Impossible values stop: temperature outside -60 to 60 C (Kelvin and Fahrenheit are suggested), RH outside 0 to 110% or never above 1 (a fraction), wind speed outside 0 to 75 m/s, radiation outside -50 to 2000 W/m2 and negative rain. Unusual values give a warning that points to a likely unit problem: temperature above 45 C, wind speed above 20 m/s (km/h), radiation above 1400 W/m2, or below 50 W/m2 over at least 12 hours (MJ m-2 h-1). Wind speed in km/h is not caught when it is below 20 km/h. A file with semicolon separators stops, pointing to read.WSdata2().

METRIC needs a clear image, so the station radiation at the overpass (Rs) should be close to the clear-sky radiation (Rso, ASCE-EWRI 2005, over the hour centred on the overpass). A warning is given when Rs/Rso is below 0.8 (clouds near the station, or a faulty or miscalibrated pyranometer) or above 1.1 (units or calibration). It is not checked when Rso is below 200 W/m2 (low sun).

Value

An object of class waterWeatherStation: a list with location (lat, long, elev, height), alldata (all records), daily (daily summaries), hourly (values interpolated every hour) and, when an MTL file is available, at.sat (conditions interpolated at the satellite overpass).

Author(s)

Guillermo Federico Olmedo

References

Landsat 7 Metadata example file available from the U.S. Geological Survey. Weather Station example file courtesy of CITRA, Universidad de Talca, Chile

ASCE-EWRI (2005). The ASCE Standardized Reference Evapotranspiration Equation. Report of the ASCE-EWRI Task Committee on Standardization of Reference Evapotranspiration.

See Also

`read.WSdata2` for files with semicolon separators and decimal commas (read.csv2())

Other Weather station related functions: `getDataWS()`, `plot.waterWeatherStation()`, `print.waterWeatherStation()`, `read.WSdata2()`

Examples

```
csvfile <- system.file("extdata", "apples.csv", package="water")
MTLfile <- system.file("extdata", "L7.MTL.txt", package="water")
# wind speed in km/h: cf converts it to m/s
WS <- read.WSdata(WSdata = csvfile, date.format = "%d/%m/%Y",
                  lat=-35.42222, long= -71.38639, elev=201, height= 2.2,
                  columns=c("date" = 1, "time" = 2, "radiation" = 3,
                  "wind" = 4, "RH" = 6, "temp" = 7, "rain" = 8),
                  cf = c(1, 0.2777778, 1), MTL = MTLfile, tz = "America/Santiago")
print(WS)
plot(WS, hourly=FALSE)
plot(WS, hourly=TRUE)
```

---

read.WSdata2 *Prepares weather station data from csv2 files*

Description

Like read.WSdata(), but reads the file with read.csv2() (semicolon separators and decimal commas). A data.frame is passed to read.WSdata() unchanged.

Usage

```
read.WSdata2(
  WSdata,
  ...,
  height = 2.2,
  lat,
  long,
  elev,
 columns = c(date = 1, time = 2, radiation = 3, wind = 4, RH = 6, temp = 7, rain = 8),
  date.format = "%d/%m/%Y",
  time.format = "%H:%M:%S",
  datetime.format = "%Y-%m-%d %H:%M:%S",
  tz = "",
  cf = c(1, 1, 1),
  MTL,
  timestamps = "instant",
  overpass.radiation = "measured"
)
```

Arguments

WSdata	csv2 file with weather station data, or a data.frame
...	additional parameters passed to read.csv2()
height	weather station sensors height in meters
lat	latitude of weather station in decimal degrees. Negative values for south latitude
long	longitude of weather station in decimal degrees. Negative values for west longitude
elev	elevation of weather station in meters
columns	numeric vector with the column numbers in WSdata of the date, time, radiation, wind, RH, temperature and rain. If date and time are in the same column, give the same number twice (or use datetime). A named vector is matched by name, in any order: date and time (or datetime), radiation, wind, RH, temp and, optionally, rain. An unnamed vector is taken in that order. Unknown or repeated names stop; pp is accepted for rain. A warning is given when the names give the vector another meaning than its order (before version 1.0 only the order was used).
date.format	date format. See strptime format argument.
time.format	time format. See strptime format argument.
datetime.format	datetime format. See strptime format argument.
tz	time zone of the weather station clock (e.g. "America/Santiago"; see OlsonNames). If "" (default) the time zone of the computer is assumed, with a warning. Always set it: the overpass time in the MTL file is UTC and is converted to this time zone. See strptime for details.

`cf` conversion factor to convert radiation, wind, and temperature to W/m2; m/s and Celsius. See Details.

`MTL` path to the Landsat metadata (MTL) file. When given, the weather conditions at the satellite overpass are interpolated (element at.sat); if the overpass is outside the records, a warning is given and at.sat is not computed. If missing, at.sat is not computed (with a message).

`timestamps` "instant" (default) if each record is the value at its time; "end" or "start" if it is the mean over the period ending or starting at its time (e.g. hourly means). For period means, the conditions at the overpass are interpolated between the middles of the periods. The hourly and daily values are not shifted: hourlyET() and dailyET() already take hourly values as means over the hour ending at their time.

`overpass.radiation` "measured" (default) uses the station radiation at the overpass; "clear-sky" uses the ASCE-EWRI (2005) clear-sky radiation instead (the measured value is kept in at.sat$radiation.measured), e.g. when clouds shade the station but not the image. calcH() (ET of the cold pixels) and ET24h() then use the same clear-sky reference ET. See Details.

Details

For cf, if your data is in W/m2, km/h and Celsius (radiation, wind, temperature), cf should be: cf = c(1,0.2777778,1). The records are sorted and checked as in read.WSdata().

Value

A waterWeatherStation object, see `read.WSdata`.

Author(s)

Guillermo Federico Olmedo

See Also

Other Weather station related functions: `getDataWS()`, `plot.waterWeatherStation()`, `print.waterWeatherStation()`, `read.WSdata()`

---

soilHeatFlux *Estimates Soil Heat Flux*

Description

This function implements models to estimate soil heat flux for different surfaces and considering different inputs.

Usage

```
soilHeatFlux(image, Ts, albedo, LAI, Rn, method = "Tasumi", aoi)
```

Arguments

image	surface reflectance SpatRaster with layers named R and NIR, used for NDVI (a stack without standard names: the third and fourth layers, with a warning; a named stack without R or NIR is an error)
Ts	land surface temperature (K), from surfaceTemperature()
albedo	broadband surface albedo, from albedo()
LAI	SpatRaster with leaf area index, from LAI()
Rn	net radiation (W/m2), from netRadiation()
method	method used for the G estimation: "Tasumi" (Tasumi, 2003; default) or "Bastiaanssen" (Bastiaanssen, 2000)
aoi	Not used (a warning is given when it is passed); kept for backward compatibility

Value

SpatRaster with soil heat flux (W/m2).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Tasumi, M. (2003). Progress in operational estimation of regional evapotranspiration using satellite imagery. Ph.D. dissertation, University of Idaho, Moscow, Idaho.

Bastiaanssen, W. G. M. (2000). SEBAL-based sensible and latent heat fluxes in the irrigated Gediz Basin, Turkey. Journal of Hydrology, 229(1-2), 87-100. https://doi.org/10.1016/S0022-1694(99)00202-4

See Also

Other net radiation related functions: LAI(), METRICtopo(), SWtransmissivity(), albedo(), incLWradiation(), incSWradiation(), netRadiation(), outLWradiation(), solarAngles(), surfaceTemperature()

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
```

```
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
             incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)
Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                         solar.angles = solar.angles, Ts = Ts)
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
G <- soilHeatFlux(image = image.SR, Ts = Ts, albedo = albedo, Rn = Rn, LAI = LAI)
plot(G)
```

`solarAngles` *Calculates solar angles*

Description

Metadata, aspect and slope maps are combined to estimate solar angles for the entire image.

Usage

```
solarAngles(surface.model, MTL, WeatherStation)
```

Arguments

`surface.model`	SpatRaster with layers DEM, Slope and Aspect, from METRICtopo()
`MTL`	path to the Landsat metadata (*_MTL.txt) file. Required
`WeatherStation`	waterWeatherStation object (see `read.WSdata`) or the data.frame returned by getDataWS()

Details

Latitude is computed for each row of the grid; declination and hour angle are computed from the acquisition date and scene centre time in the MTL file and the weather station longitude. Solar incidence angles are computed for a horizontal surface (incidence.hor) and for the sloping surface given by Slope and Aspect (incidence.rel), following Allen et al. (2007).

Value

SpatRaster with layers (radians) latitude, declination, hour.angle (negative before solar noon), incidence.hor (solar incidence angle on a horizontal surface) and incidence.rel (incidence angle on the sloping surface).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

Fernando Fuentes-Peñailillo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

See Also

Other net radiation related functions: `LAI()`, `METRICtopo()`, `SWtransmissivity()`, `albedo()`, `incLWradiation()`, `incSWradiation()`, `netRadiation()`, `outLWradiation()`, `soilHeatFlux()`, `surfaceTemperature()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
plot(solar.angles)
```

---

summary.waterLSEB *Summary of the calibration of a METRIC energy balance*

Description

Prints the quality checks of the calibration of a METRIC.EB() result: the anchor pixels, the iteration of H and the share of pixels with unlikely values of the reference ET fraction (ETrF).

Usage

```
## S3 method for class 'waterLSEB'
summary(object, ...)
```

Arguments

object	waterLSEB object, from METRIC.EB().
...	not used.

Details

In METRIC, the cold anchors have ETrF = ETp.coef (1.05) and few pixels should be above about 1.1 (Allen et al., 2007): many pixels above 1.1 mean that the cold anchors are too warm. Pixels with H > Rn - G have negative ETrF, and LE is set to 0 there.

Value

The calibration list (object$calibration), invisibly.

Author(s)

Guillermo Federico Olmedo

See Also

Other LSEB objects related functions: plot.waterLSEB(), plotAnchors(), print.waterLSEB(), writeRaster.waterLSEB

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
summary(Energy.Balance)
```

surfaceTemperature *Estimates Land Surface Temperature from Landsat Data*

Description

Estimates the land surface temperature from the thermal band in digital numbers with the METRIC equations (Allen et al., 2007, Eqs. 20-21): the at-sensor radiance is corrected for the atmosphere and for the surface emissivity, and converted to temperature with the K1 and K2 constants of the band. For Landsat 8 and 9, the Level-2 surface temperature (image.SR$Band_10 from loadImageSR(), used by default by METRIC.EB()) is preferred.

Usage

```
surfaceTemperature(
  image.DN,
  sat = "auto",
  LAI,
  method = "METRIC",
  WeatherStation,
  thermalband,
  MTL,
  atmosphere,
  aoi
)
```

Arguments

image.DN	raw image in digital numbers, e.g. from loadImage()
sat	"L7", "L8", "L9" or "auto" to detect it from the MTL file (MTL is then required)
LAI	SpatRaster with leaf area index, from LAI()
method	"METRIC" (default; Allen et al., 2007, Eqs. 20-21) or "SW" (split window, Landsat 8 and 9 only; deprecated). "SC" was removed in version 1.0
WeatherStation	Not used; kept for backward compatibility
thermalband	thermal band in digital numbers. Optional: if missing, image.DN$Thermal1 is used (L7: band 6 low gain; L8/L9: band 10)
MTL	path to the Landsat metadata (*_MTL.txt) file. For L8 and L9 the radiance rescaling and thermal constants are read from it. Required for L9; without it L8 uses Landsat 8 defaults
atmosphere	Optional atmospheric correction: a named list or SpatRaster with tau (narrow-band transmissivity), Rp (path radiance) and Rsky (downward thermal radiance, both W m-2 sr-1 um-1). For Landsat 8 and 9 Level-2 products, the per-pixel bands ST_ATRAN * 0.0001, ST_URAD * 0.001 and ST_DRAD * 0.001. If missing, the general values of Allen et al. (2007) are used: tau = 0.866, Rp = 0.91 and Rsky = 1.32
aoi	Not used (a warning is given when it is passed); kept for backward compatibility

Details

Rc = (Lt - Rp) / tau - (1 - eNB) Rsky (Eq. 21), Ts = K2 / ln(eNB K1 / Rc + 1) (Eq. 20), with the narrow-band emissivity eNB = 0.97 + 0.0033 LAI (0.98 for LAI > 3). The general atmospheric values come from MODTRAN runs for Landsat 5 and 7 in Idaho; for Landsat 8 and 9 band 10 they can bias Ts by several K (-6 K in the example crop against the Level-2 product), so a warning is given when they are used for Landsat 8 or 9: use the Level-2 surface temperature, or the per-pixel atmosphere bands.

Value

SpatRaster with land surface temperature (K).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Jimenez-Munoz, J. C., Sobrino, J. A., Skokovic, D., Mattar, C., & Cristobal, J. (2014). Land surface temperature retrieval methods from Landsat-8 thermal infrared sensor data. IEEE Geoscience and Remote Sensing Letters, 11(10), 1840-1843. http://doi.org/10.1109/LGRS.2014.2312032

See Also

Other net radiation related functions: `LAI()`, `METRICtopo()`, `SWtransmissivity()`, `albedo()`, `incLWradiation()`, `incSWradiation()`, `netRadiation()`, `outLWradiation()`, `soilHeatFlux()`, `solarAngles()`

Examples

```
# a small window of the Landsat 7 example scene, to run quickly
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
DEM <- terra::rast(DEM_Talca)[1:60, 1:80, drop = FALSE]
surface.model <- METRICtopo(DEM)
solar.angles <- solarAngles(surface.model = surface.model,
                            WeatherStation = WeatherStation, MTL = MTL)
Rs.inc <- incSWradiation(surface.model = surface.model, solar.angles = solar.angles,
                         WeatherStation = WeatherStation)
image.TOAr <- calcTOAr(image.DN, sat = "L7", MTL = MTL,
                       incidence.rel = solar.angles$incidence.rel)
image.SR <- suppressWarnings(calcSR(image.TOAr, sat = "L7", surface.model = surface.model,
```

```
            incidence.hor = solar.angles$incidence.hor, WeatherStation = WeatherStation))
albedo <- albedo(image.SR, coeff = "Tasumi", sat = "L7")
LAI <- LAI(method = "metric2010", image = image.TOAr, L = 0.1)
Ts <- suppressWarnings(surfaceTemperature(image.DN = image.DN, LAI = LAI, sat = "L7",
                                          WeatherStation = WeatherStation))
plot(Ts)
```

`SWtransmissivity` *Calculates broadband short-wave transmissivity*

Description

Short-wave transmissivity is estimated for broad-band considering an extended equation developed by Allen (1996), based on Majumdar et al. (1972), using coefficients developed by ASCE-EWRI (2005).

Usage

```
SWtransmissivity(Kt = 1, ea, dem, incidence.hor)

SWtrasmisivity(Kt = 1, ea, dem, incidence.hor)
```

Arguments

`Kt`	unitless turbidity coefficient 0<Kt<=1.0, where Kt=1.0 for clean air and Kt=0.5 for extremely turbid, dusty, or polluted air
`ea`	near-surface vapor pressure (kPa)
`dem`	SpatRaster with a digital elevation model (m)
`incidence.hor`	solar incidence angle (radians) on a horizontal surface, from solarAngles()

Value

SpatRaster with broadband short-wave atmospheric transmissivity (unitless).

Author(s)

Guillermo Federico Olmedo

David Fonseca-Luengo

References

R. G. Allen, M. Tasumi, and R. Trezza, "Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC) - Model" Journal of Irrigation and Drainage Engineering, vol. 133, p. 380, 2007

Majumdar, N.; Mathur, B. & Kaushik, S. Prediction of direct solar radiation for low atmospheric turbidity Solar Energy, Elsevier, 1972, 13, 383-394

ASCE-EWRI The ASCE Standardized Reference Evapotranspiration Equation Report of the ASCE-EWRI Task Committee on Standardization of Reference Evapotranspiration, 2005

See Also

Other net radiation related functions: `LAI()`, `METRICtopo()`, `albedo()`, `incLWradiation()`, `incSWradiation()`, `netRadiation()`, `outLWradiation()`, `soilHeatFlux()`, `solarAngles()`, `surfaceTemperature()`

Examples

```
# Kt = 1 for clean air; ea in kPa, dem in m, incidence in radians
SWtransmissivity(Kt = 1, ea = 1.5, dem = 200, incidence.hor = 0.5)
```

`water-extdata` *Example files of the water package*

Description

Files in `system.file("extdata", package = "water")`, used in the examples and vignettes:

apples.csv Weather station in an apple orchard near Talca, Chile (latitude -35.42222, longitude -71.38639, 201 m, sensors at 2.2 m), with records every 15 minutes on 15 February 2013, in local time (America/Santiago). Columns: Date (dd/mm/yyyy), Time, Rad (solar radiation, W/m2), wind_speed (km/h), wind_dir (degrees), RH (%), temp (Celsius) and pp (rain, mm). Read it with `cf = c(1, 0.2777778, 1)` to convert the wind speed to m/s.

INTA.csv Hourly weather station records near Mendoza, Argentina (latitude -33.00513, longitude -68.86469, 927 m, sensors at 2 m) on 9 February 2016, in local time (America/Argentina/Mendoza). Columns: datetime (yyyy/mm/dd HH:MM), temp (Celsius), RH (%), pp (rain, mm), radiation (W/m2) and wind (m/s).

L7.MTL.txt Metadata of the Landsat 7 scene of `L7_Talca` (15 February 2013).

LC82320832016040LGN00_band2.tif to _band11.tif, LC82320832016040LGN00_MTL.txt Landsat 8 Level-1 digital numbers (bands 2 to 7, 10 and 11, with pre-Collection file names) of path 232, row 83, 9 February 2016, near Mendoza, and its metadata. Read them with `loadImage(sat = "L8")`.

L8_C2 Folder with the Collection 2 Level-2 product of the same scene, on the same grid: surface reflectance (SR_B2 to SR_B7), surface temperature (ST_B10), quality band (QA_PIXEL) and metadata (MTL). Read it with `loadImageSR()` and mask clouds with `cfmask()`.

Examples

```
list.files(system.file("extdata", package = "water"))
```

`waterOptions` *Global options for water package*

Description

Sets or shows the global options of the water package. Inspired by rasterOptions() of the raster package (Robert J. Hijmans).

Usage

```
waterOptions(
  overwrite,
  writeResults,
  outputFolder,
  SRTMrepo,
  autoAoi,
  default = FALSE
)
```

Arguments

`overwrite`	Logical. If TRUE and writeResults is TRUE, the final outputs overwrite files of earlier sessions; a file written earlier in the same session is never overwritten (an object may still read it): a new name is used, with a warning (so, in one session, EB.tif keeps the first result). If FALSE, a time stamp is appended to the file name (name_YYYYmmddHHMMSS.tif).
`writeResults`	Logical. If TRUE (default FALSE) results are written to disk as GeoTIFF files: the final outputs (EB from METRIC.EB(), ET24 from ET24h(), DEM from getDEM() and prepareSRTMdata()) to outputFolder, and the intermediate results to terra's temporary files, with unique names. This is slower, but avoids out-of-memory problems with large images. Values are stored as 32-bit floats. The temporary files are deleted when the R session ends (they can be large for full scenes).
`outputFolder`	Name of a folder to save files, relative to the working directory.
`SRTMrepo`	A folder where SRTM grids are stored. Currently not used by prepareSRTMdata(); use its path argument
`autoAoi`	Deprecated and ignored, with a warning. An object called aoi in the workspace is no longer used: pass aoi to loadImage(), loadImageSR(), albedo() or the METRIC functions. See `createAoi`
`default`	Logical. If TRUE will revert all options to default values

Value

list of the current options (invisibly). If no arguments are provided the options are printed.

Author(s)

Guillermo Federico Olmedo

References

Robert J. Hijmans (2015). raster: Geographic Data Analysis and Modeling. R package version 2.4-18. https://CRAN.R-project.org/package=raster

See Also

Other support functions: `checkSRTMgrids()`, `createAoi()`, `getDEM()`, `prepareSRTMdata()`

Examples

```
# show the options
waterOptions()
```

`writeRaster.waterLSEB` *writeRaster method for waterLSEB objects*

Description

Writes the energy balance layers (`x$EB`) of a METRIC.EB() result to a file.

Arguments

`x`	waterLSEB object, from METRIC.EB().
`filename`	output file name, e.g. "EB.tif"
`...`	additional arguments passed to terra::writeRaster(), e.g. overwrite

Value

The SpatRaster written to the file (x$EB), invisibly, as returned by terra::writeRaster().

Author(s)

María Victoria Munafó

See Also

Other LSEB objects related functions: `plot.waterLSEB()`, `plotAnchors()`, `print.waterLSEB()`, `summary.waterLSEB()`

Examples

```
MTL <- system.file("extdata", "L7.MTL.txt", package = "water")
WeatherStation <- read.WSdata(WSdata = system.file("extdata", "apples.csv", package = "water"),
  date.format = "%d/%m/%Y", lat = -35.42222, long = -71.38639, elev = 201, height = 2.2,
  cf = c(1, 0.2777778, 1), tz = "America/Santiago", MTL = MTL)
image.DN <- terra::rast(L7_Talca)[1:60, 1:80, drop = FALSE]
Energy.Balance <- METRIC.EB(image.DN = image.DN, WeatherStation = WeatherStation, MTL = MTL,
                            sat = "L7", thermalband = image.DN$Thermal1, plain = TRUE)
# writes the layers of the energy balance as a GeoTIFF file
writeRaster(Energy.Balance, tempfile(fileext = ".tif"))
```

Index

∗ **ET functions**
Crad, 20
dailyET, 22
dailyETseries, 24
ET24h, 26
ETperiod, 28
hourlyET, 32
∗ **LSEB objects related functions**
plot.waterLSEB, 58
plotAnchors, 60
print.waterLSEB, 62
summary.waterLSEB, 71
writeRaster.waterLSEB, 78
∗ **METRIC model functions**
anchors.control, 4
calcH.control, 12
METRIC.EB, 41
METRIC.G, 46
METRIC.Rn, 49
∗ **Weather station related functions**
getDataWS, 30
plot.waterWeatherStation, 59
print.waterWeatherStation, 63
read.WSdata, 64
read.WSdata2, 66
∗ **datasets**
DEM_Talca, 25
L7_Talca, 37
water-extdata, 76
∗ **net radiation related functions**
albedo, 3
incLWradiation, 34
incSWradiation, 35
LAI, 37
METRICtopo, 52
netRadiation, 55
outLWradiation, 56
soilHeatFlux, 68
solarAngles, 70
surfaceTemperature, 73
SWtransmissivity, 75
∗ **remote sensing support functions**
calcRadiance, 14
calcSR, 15
calcTOAr, 17
cfmask, 18
loadImage, 39
loadImageSR, 40
∗ **sensible heat flux functions**
calcAnchors, 6
calcH, 9
momentumRoughnessLength, 53
∗ **support functions**
checkSRTMgrids, 19
createAoi, 21
getDEM, 31
prepareSRTMdata, 61
waterOptions, 77

albedo, 3, *44*, *48*, *51*
albedo(), *35, 36*, *39*, *52*, *56, 57*, *69*, *71*, *74*, *76*
anchors.control, 4, *7*, *44*
anchors.control(), *13*, *46*, *49*, *51*
apples.csv *(*water-extdata*)*, 76

calcAnchors, *5*, 6, *43*
calcAnchors(), *11*, *54*
calcH, 9, *13*, *26*, *44*
calcH(), *8*, *54*
calcH.control, *10*, 12, *44*, *54*
calcH.control(), *5*, *46*, *49*, *51*
calcRadiance, 14, *38*
calcRadiance(), *16*, *18, 19*, *40, 41*
calcSR, 15
calcSR(), *14*, *18, 19*, *40, 41*
calcTOAr, 17
calcTOAr(), *14*, *16*, *19*, *40, 41*
cfmask, 18
cfmask(), *14*, *16*, *18*, *40, 41*

checkSRTMgrids, 19, *31*, *61*
checkSRTMgrids(), *22*, *31*, *61*, *78*
Crad, 20, *26*, *45*
Crad(), *24, 25*, *27*, *29*, *33*
createAoi, 21, *77*
createAoi(), *20*, *31*, *61*, *78*

dailyET, 22, *24*
dailyET(), *21*, *25*, *27*, *29*, *33*
dailyETseries, 24, *29*
dailyETseries(), *21*, *24*, *27*, *29*, *33*
DEM_Talca, 25

ET24h, 26, *29*, *45*
ET24h(), *21*, *24, 25*, *29*, *33*
ETperiod, *24*, 28
ETperiod(), *21*, *24, 25*, *27*, *33*
extdata *(*water-extdata*)*, 76

getDataWS, 30, *32*
getDataWS(), *59*, *63*, *66*, *68*
getDEM, 31
getDEM(), *20*, *22*, *61*, *78*

hourlyET, *11*, *22, 23*, 32
hourlyET(), *21*, *24, 25*, *27*, *29*

incLWradiation, 34
incLWradiation(), *4*, *36*, *39*, *52*, *56, 57*, *69*, *71*, *74*, *76*
incSWradiation, 35
incSWradiation(), *4*, *35*, *39*, *52*, *56, 57*, *69*, *71*, *74*, *76*
INTA.csv *(*water-extdata*)*, 76

L7_Talca, 37
LAI, 37, *43, 44*, *48*, *50, 51*
LAI(), *4*, *35, 36*, *52*, *56, 57*, *69*, *71*, *74*, *76*
loadImage, 39
loadImage(), *14*, *16*, *18, 19*, *41*
loadImageSR, *15*, 40
loadImageSR(), *14*, *16*, *18, 19*, *40*

METRIC.EB, 41
METRIC.EB(), *5*, *13*, *49*, *51*
METRIC.G, 46
METRIC.G(), *5*, *13*, *46*, *51*
METRIC.Rn, 49
METRIC.Rn(), *5*, *13*, *46*, *49*
METRICtopo, 52
METRICtopo(), *4*, *35, 36*, *39*, *56, 57*, *69*, *71*, *74*, *76*
momentumRoughnessLength, *43*, 53
momentumRoughnessLength(), *8*, *11*

netRadiation, *10*, *26*, 55
netRadiation(), *4*, *35, 36*, *39*, *52*, *57*, *69*, *71*, *74*, *76*

OlsonNames, *65*, *67*
outLWradiation, 56
outLWradiation(), *4*, *35, 36*, *39*, *52*, *56*, *69*, *71*, *74*, *76*

plot.waterLSEB, 58
plot.waterLSEB(), *60*, *62*, *72*, *78*
plot.waterWeatherStation, 59
plot.waterWeatherStation(), *30*, *63*, *66*, *68*
plotAnchors, 60
plotAnchors(), *58*, *62*, *72*, *78*
prepareSRTMdata, 61
prepareSRTMdata(), *20*, *22*, *31*, *78*
print.waterLSEB, 62
print.waterLSEB(), *58*, *60*, *72*, *78*
print.waterWeatherStation, 63
print.waterWeatherStation(), *30*, *59*, *66*, *68*

read.WSdata, *15*, *20*, *23, 24*, *30*, *34*, *36*, *59*, *63*, 64, *68*, *70*
read.WSdata(), *30*, *59*, *63*, *68*
read.WSdata2, 66, *66*
read.WSdata2(), *30*, *59*, *63*, *66*

soilHeatFlux, *10*, *26*, *43*, *47*, 68
soilHeatFlux(), *4*, *35, 36*, *39*, *52*, *56, 57*, *71*, *74*, *76*
solarAngles, 70
solarAngles(), *4*, *35, 36*, *39*, *52*, *56, 57*, *69*, *74*, *76*
summary.waterLSEB, *45*, *62*, 71
summary.waterLSEB(), *58*, *60*, *62*, *78*
surfaceTemperature, *6*, *10*, *26*, *43*, *48*, *50*, 73
surfaceTemperature(), *4*, *35, 36*, *39*, *52*, *56, 57*, *69*, *71*, *76*
SWtransmissivity, *16*, *20*, *34*, *36*, *44*, *48*, *50*, 75
SWtransmissivity(), *4*, *35, 36*, *39*, *52*, *56, 57*, *69*, *71*, *74*

SWtrasmisivity *(SWtransmissivity)*, 75

water-extdata, 76
waterOptions, 77
waterOptions(), *20*, *22*, *31*, *61*
writeRaster,waterLSEB,character-method
(writeRaster.waterLSEB), 78
writeRaster.waterLSEB, *58*, *60*, *62*, *72*, 78