

Package ‘microclima’

September 3, 2018

Type Package

Title microclimate modelling with R

Version 0.1.0

Maintainer Ilya M. D. Maclean <i.m.d.maclean@exeter.ac.uk>

Description The functions in the microclima package contain tools for modelling the mechanistic processes that govern fine-scale variation in temperature. It includes tools for determining local variation in temperature arising from variation in radiation, wind speed, altitude, surface albedo, coastal influences and cold air drainage. A series of functions for determining the fine-scale topographic and vegetation effects on wind speed and radiation are also provided. It also includes tools for deriving canopy cover, leaf architecture, surface albedo and cold air drainage basins from digital elevation data and aerial imagery.

License GPL-2

Encoding UTF-8

LazyData true

Remotes mrke/NicheMapR

Imports raster (>= 2.5),
rgdal (>= 0.8),
elevatr (>= 0.1.4),
stringr (>= 1.2),
sp (>= 1.2),
dplyr (>= 0.7),
RNCEP (>= 1.0.8),
zoo (>= 1.8),
devtools

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

Suggests knitr,
NicheMapR,
rmarkdown,
testthat

VignetteBuilder knitr

URL <https://github.com/ilyamaclean/microclima>

BugReports <http://github.com/microclima/issues>

R topics documented:

aerial_image	3
airmasscoef	4
albedo	4
albedo2	5
albedo_adjust	7
albedo_reflected	7
arrayspline	8
basindelin	9
basindelin_big	10
basinmerge	11
basins100m	11
cadconditions	12
canopy	13
cfc	14
coastalTps	15
dailyprecipNCEP	16
demworld	16
difprop	17
difrad	18
dnirad	18
dtm100m	19
dtm1km	19
dtm1m	20
dtr	20
fitmicro	21
flowacc	22
get_dem	23
get_NCEP	24
horizonangle	25
hourlyNCEP	26
hourlytemp	27
humidityconvert	28
huss	29
if_raster	29
invs	30
is_raster	31
julday	31
lai	32
lai_adjust	33
landsearatio	34
lapserate	34
latlongfromraster	35
leaf_geometry	35
longwavetopo	36
longwaveveg	37
mean_slope	39
mesofitdata	40
microclimaforNMR	40
microfitdata	42
microvars	43

modis	43
netlong100m	44
netlong1m	44
netshort100m	44
netshort1m	45
pcad	45
pres	46
runmicro	47
shortwavetopo	49
shortwaveveg	52
siflat	55
skyviewtopo	55
skyviewveg	56
solalt	57
solarindex	58
solartime	60
solazi	60
suntimes	61
tas	62
temp100	62
veg_hgt	63
wind1m	63
wind2010	64
windcoef	64
windheight	65

Index 66

aerial_image	<i>A 1 m resolution aerial image.</i>
--------------	---------------------------------------

Description

A dataset containing image reflectance values for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

aerial_image

Format

An array with 1000 rows, 1000 columns and 4 layers:

- 1** blue band reflectance values (430 to 490 nm), in the range 0 to 255
- 2** green band reflectance values (535 to 585 nm), in the range 0 to 255
- 3** red band reflectance values (610 to 660 nm), in the range 0 to 255
- 4** near-infrared band reflectance values (835 to 885 nm), in the range 0 to 255

Source

<https://www.bluesky-world.com/>

airmasscoef	<i>Calculates the airmass coefficient</i>
-------------	---

Description

airmasscoef is used to calculate, for a given location and time, the direct optical path length of a solar beam through the atmosphere of the Earth, expressed as a ratio relative to the path length vertically upwards.

Usage

```
airmasscoef(localtime, lat, long, julian, merid = 0, dst = 0)
```

Arguments

localtime	local time (decimal hour, 24 hour clock).
lat	latitude of the location for which the airmass coefficient is required (decimal degrees, -ve south of equator).
long	longitude of the location for which the airmass coefficient is required (decimal degrees, -ve west of Greenwich meridian).
julian	a numeric value representing the Julian day as returned by <code>julday()</code> .
merid	an optional numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Value

the airmass coefficient, i.e. the direct optical path length of a solar beam through the Earth's atmosphere, expressed as a ratio relative to the path length vertically upwards for a given location and time.

Examples

```
# airmass coefficient at noon on 21 June 2010, Porthleven, Cornwall
jd <- julday (2010, 6, 21) # Julian day
airmasscoef(12, 50.08, -5.31, jd)
```

albedo	<i>Calculates surface albedo</i>
--------	----------------------------------

Description

albedo is used to calculate surface albedo.

Usage

```
albedo(blue, green, red, nir, maxval = 255, bluerange = c(430, 490),
  greenrange = c(535, 585), redrange = c(610, 660), nirrange = c(835,
  885))
```

Arguments

blue	a raster object, two-dimensional array or matrix of reflectance values in the blue spectral band (0 to max.val).
green	a raster object, two-dimensional array or matrix of reflectance values in the green spectral band (0 to max.val).
red	a raster object, two-dimensional array or matrix of reflectance values in the red spectral band (0 to max.val).
nir	a raster object, two-dimensional array or matrix of reflectance values in the near-infrared spectral band (0 to max.val).
maxval	a single numeric value representing the maximum reflectance in any of the spectral bands.
bluerange	an optional numeric vector of length 2 giving the range (minimum, maximum) of wavelength values captured by the blue spectral band sensor (nm).
greenrange	an optional numeric vector of length 2 giving the range (minimum, maximum) of wavelength values captured by the green spectral band sensor (nm).
redrange	an optional numeric vector of length 2 giving the range (minimum, maximum) of wavelength values captured by the red spectral band sensor (nm).
nirrange	an optional numeric vector of length 2 giving the range (minimum, maximum) of wavelength values captured by the near-infrared spectral band sensor (nm).

Details

The function assumes that image reflectance has been captured using four spectral bands. If blue is a raster object, then a raster object is returned.

Value

a raster object or two-dimensional array of numeric values representing surface albedo (range 0 to 1).

See Also

Function [albedo_adjust\(\)](#) for adjusted albedo for image brightness and contrast.

Examples

```
alb <- albedo(aerial_image[,1], aerial_image[,2], aerial_image[,3],
             aerial_image[,4])
plot(if_raster(alb, dtm1m), main = "Albedo", col = gray(0:255/255))
```

albedo2

Partitions surface albedo between ground and canopy albedo

Description

albedo2 is used to calculate either bare ground or canopy albedo from surface albedo.

Usage

```
albedo2(alb, fr, ground = TRUE)
```

Arguments

alb	a raster object, two-dimensional array or matrix of surface albedo values (range 0 - 1) derived using <code>albedo()</code> or <code>albedo_adjust()</code> .
fr	a raster object, two-dimensional array or matrix of fractional canopy cover as returned by <code>canopy()</code> .
ground	a logical value indicating whether to return ground albedo (TRUE) or canopy albedo (FALSE).

Details

If alb is a raster object, a raster object is returned. For calculation of net radiation, both ground and canopy albedo may be needed. Areas with high canopy cover typically have lower albedo values than areas with low canopy cover and mean values for these can be derived from parts of the image with very high or very low canopy cover. It is assumed that albedo of the image as returned by `albedo()` is a function of both, weighted by canopy cover, such that canopy albedos are closer to the image-derived albedos in areas of high canopy cover and ground albedos closer to the image-derived albedo in areas with low canopy cover.

Value

If ground is TRUE, a raster object or a two-dimensional array of numeric values representing ground albedo (range 0 to 1).

If ground is FALSE, a raster object or two-dimensional array of numeric values representing canopy albedo (range 0 to 1).

Examples

```
# =====
# Calculate image albedo
# =====
alb <- albedo(aerial_image[,1], aerial_image[,2], aerial_image[,3],
             aerial_image[,4])
# =====
# Calculate canopy cover
# =====
l <- lai(aerial_image[,3], aerial_image[,4])
x <- leaf_geometry(veg_hgt)
fr <- canopy(l, x)
# =====
# Calculate and plot ground and canopy albedo
# =====
ag <- albedo2(alb, fr)
ac <- albedo2(alb, fr, ground = FALSE)
par(mfrow=c(2, 1))
plot(if_raster(ag, dtm1m), main = "Ground albedo", col = gray(0:255/255))
plot(if_raster(ac, dtm1m), main = "Canopy albedo", col = gray(0:255/255))
```

albedo_adjust	<i>Adjusts albedo to correct for image brightness and contrast</i>
---------------	--

Description

albedo_adjust is used to correct albedo derived from aerial imagery for image brightness and contrast using MODIS data.

Usage

```
albedo_adjust(alb_image, alb_modis)
```

Arguments

alb_image	a raster object with numeric albedo values derived from high-resolution aerial imagery, as derived by albedo() and converted to a raster object.
alb_modis	a raster object with numeric albedo values derived from MODIS imagery covering the same extent as alb_image (range 0 to 1). If the extent of the alb_modis is greater than that of alb_image, alb_modis is cropped.

Value

a raster object with numeric values representing the adjusted surface albedo values (range 0 to 1).

Examples

```
alb <- albedo(aerial_image[,1], aerial_image[,2], aerial_image[,3],
             aerial_image[,4])
img <- if_raster(alb, dtm1m)
mds <- raster(modis, xmn = 169000, xmx = 170000, ymn = 12000, ymx = 13000)
alb2 <- albedo_adjust(img, mds)
par(mfrow=c(2, 1))
plot(img, main = "Raw albedo", col = gray(0:255/255))
plot(alb2, main = "Adjusted albedo", col = gray(0:255/255))
```

albedo_reflected	<i>Calculates the mean albedo of surfaces surrounding each location</i>
------------------	---

Description

albedo_reflected is used to calculate mean albedo of surfaces surrounding a location from which radiation is reflected.

Usage

```
albedo_reflected(alb, e = extent(alb))
```

Arguments

- alb** a raster object, two-dimensional array or matrix of surface albedo values (range 0 - 1) derived using `albedo()` or `albedo_adjust()`.
- e** an optional extent object indicating the geographic extent of alb.

Details

A small proportion of radiation received at any given location is in the form of reflected radiation, and this function permits the albedo of surrounding surfaces to be calculated. An inverse distance-weighting is applied (range 0 to 1). If alb is a raster object, then a raster object is returned and e can be determined from alb.

Value

a raster object, two-dimensional array or matrix of values representing the mean albedo of surfaces surrounding each pixel of a two-dimension albedo array (range 0 - 1).

Examples

```
alb <- albedo(aerial_image[, , 1], aerial_image[, , 2], aerial_image[, , 3],
             aerial_image[, , 4])
alb <- alb[901:1000, 901:1000]
e <- extent(c(169900, 170000, 12900, 13000))
rt <- raster(e, res = 1)
r_alb <- albedo_reflected(alb, e)
par(mfrow = c(2, 1))
plot(if_raster(alb, rt), main = "Surface albedo", col = gray(0:255/255))
plot(if_raster(r_alb, rt), main = "Albedo of surrounding surfaces",
     col = gray(0:255/255))
```

arrayspline

*Applies a spline function to an array of values***Description**

arrayspline is used to derive e.g. hourly climate data from e.g. daily values.

Usage

```
arrayspline(a, tme, nfact = 24, out = NA)
```

Arguments

- a** a three-dimensional array (row, column, time)
- tme** an object of class POSIXct of times for a. I.e. `length(tme) = dim(a)[3]`
- nfact** indicates the time interval for which outputs are required. E.g to derive hourly from daily data `nfact = 24`, or derive six-hourly from daily data `nfact = 4`
- out** an optional character vector indicating the time for which the output is required. Format must be as for tme.

Details

arrayspline uses the Forsythe, Malcolm and Moler method of splining, as specified by "fmm" in [spline\(\)](#). If `a[i, j,]` is a vector of NAs, then the corresponding vector in the output array is also a vector of NAs. It is assumed that all spatial data within `a` have equivalent times. I.e. the time of both `a[1, 1, 1]` and `a[1000, 1000, 1]` is identical and equal to `tme[1]`.

Value

If `out` is unspecified, a three dimensional array of size `c(a[1], a[2], (length(tme) - 1) * nfact + 1)` is returned. If `out` is specified, a three dimensional array of size `c(a[1], a[2], length(out))` is returned.

Examples

```
tme <- as.POSIXct(c(0:364) * 24 * 3600, origin="2010-01-01", tz = "GMT")
h <- arrayspline(huss, tme, out = "2010-05-01 11:00")
plot(raster(h, template = dtm1km),
     main = "Specific humidity 2010-05-01 11:00")
```

basindelin

Delineates hydrological basins

Description

basindelin uses digital elevation data to delineate hydrological or cold-air drainage basins.

Usage

```
basindelin(dem)
```

Arguments

`dem` a raster object, two-dimensional array or matrix of elevations.

Details

If `dem` is a raster object, a raster object is returned. This function is used to delineate cold-air drainage basins. It iteratively identifies the lowest elevation pixel of `dem`, and assigns any of the eight adjoining pixels to the same basin if higher. The process is repeated on all assigned adjoining pixels until no further higher pixels are found. The next lowest unassigned pixel is then identified, a new basin identity assigned and the processes repeated until all pixels are assigned to a basin. Relative to heuristic algorithms, it is slow and run time increases exponentially with size of `dem`. However, in contrast to many such algorithms, all basins are correctly separated by boundaries >0 . With large datasets, with $> 160,000$ pixels, calculations will be slow and [basindelin_big\(\)](#) should be used instead.

Value

a raster object, two-dimensional array or matrix with individual basins numbered sequentially as integers.

See Also

[basindelin_big\(\)](#) for working with large datasets.

Examples

```
dem <- aggregate(dtm1m, 20)
basins <- basindelin (dem)
plot(basins, main = "Basins")
```

basindelin_big

Delineates hydrological basins for large datasets

Description

basindelin_big is for use with large digital elevation datasets, to delineate hydrological or cold-air drainage basins.

Usage

```
basindelin_big(dem, dirout = NA, trace = TRUE)
```

Arguments

dem	a raster object of elevations.
dirout	an optional character vector containing a single path directory for temporarily storing tiles. Deleted after use. Tilde expansion (see path.expand()) is done.
trace	a logical value indicating whether to plot and report on progress.

Details

basindelin_big divides the large dataset into tiles and then uses [basindelin\(\)](#) to delineate basins for each tile before mosaicing back together and merging basins along tile edges if not seperated by a boundary

1. If dirout is unspecified, then a directory basinsout is temporarily created within the working directory. If trace is TRUE (the default) then progress is tracked during three stages: (1) the basins of each tile are plotted, (2) basins after mosaicing, but prior to merging are plotted and (3) on each merge iteration, the number of basins to merge is printed and processed basin is plotted.

Value

a raster object with individual basins numbered sequentially as integers.

See Also

[basindelin\(\)](#) for working with smaller datasets.

Examples

```
basins <- basindelin_big(dtm1m)
plot(basins, main = "Basins")
```

basinmerge	<i>Merges adjoining basins</i>
------------	--------------------------------

Description

basinmerge merges adjoining basins if the height differences between the bottom of the basin and the pour point is less than that specified by boundary.

Usage

```
basinmerge(dem, basins, boundary)
```

Arguments

dem	a raster object, two-dimensional array or matrix of elevations.
basins	a raster object, two-dimensional array or matrix with basins numbered as integers as returned by basindelin() .
boundary	a single numeric value. Basins separated by boundaries below this height are merged (should have same units as dtm).

Details

If dem is a raster object, then a raster object is returned. If the differences in height between the pour-point and bottom of the basin is less than that specified by boundary the basin is merged with basin to which water or air would pour.

Value

a raster object, two-dimensional array or matrix with basins numbered as integers.

Examples

```
basins2 <- basinmerge(dtm100m, basins100m, 1)
par(mfrow=c(1, 2))
plot(basins100m, main = "Basins")
plot(basins2, main = "Merged basins")
```

basins100m	<i>A raster of basins numbered as integers</i>
------------	--

Description

A raster object of basins numbered as integers for the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700) as returned by [basindelin\(\)](#).

Usage

```
basins100m
```

Format

A raster object with 187 rows and 214 columns.

cadconditions	<i>Calculates whether conditions are right for cold air drainage</i>
---------------	--

Description

cadconditions determines whether wind speed, humidity and cloud cover are such that cold air drainage is likely to occur

Usage

```
cadconditions(h, tc, n, p = 100346.13, wind, startjul, lat, long,
  starttime = 0, hourint = 1, windthresh = 4.5, emthresh = 0.5,
  tz = 0, dst = 0, con = TRUE)
```

Arguments

h	a vector of hourly specific humidities ($kgkg^{-1}$).
tc	a single numeric value, raster object, two-dimensional array or matrix of temperatures ($^{\circ}C$).
n	a vector of hourly fractional cloud cover values (range 0 - 1).
p	an optional vector of hourly atmospheric pressure values (Pa).
wind	a vector of wind speed values at one metre above the ground (ms^{-1})
startjul	julian day of first observation as returned by <code>julday()</code>
lat	latitude of the location for which cold air drainage conditions are required (decimal degrees, -ve south of equator).
long	longitude of the location for which cold air drainage conditions are required (decimal degrees, -ve west of Greenwich meridian).
starttime	the hour of the first observation (decimal, 0-23).
hourint	the interval (in hours) between successive observations
windthresh	an optional threshold value of wind speed below which cold air conditions can occur (ms^{-1})
emthresh	an optional threshold value of emissivity below which cold air conditions can occur (range 0 - 1)
tz	an optional numeric value specifying the time zones expressed as hours different from GMT (-ve to west).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).
con	an optional logical value indicating whether or not to allow cold air drainage conditions to occur only if conditions are right for three or more consecutive hours. Ignored if hourint != 1.

Details

`cadconditions` uses a time series of wind and emissivity data to determine whether cold air drainage conditions are likely to occur and returns a binary vector of the same length as `em` and `wind` indicating whether conditions occur (1) or not (0). They are assumed to occur at night or within three hours of dawn only and when both `em` and `wind` are below the values specified by `windthresh` and `emthresh`. If no start time is specified it is assumed that first index of `em` and `wind` occurs at midnight on the date specified by `startjul`. If no `hourint` is provided, the time interval between indices of `em` and `wind` are assumed to be hourly. If `con` is `TRUE` and `hourint` is 1 (the default), cold air drainage conditions are assumed to persist only when conditions are right for three consecutive hours or more. The first index of the output, for which prior conditions cannot be assessed is set to 1 if conditions are right, irrespective of prior conditions. The second index is set to one only if conditions are right in both that hour and the preceding hour. If `con` is `FALSE` or `hourint != 1` prior conditions are ignored.

Value

a vector of binary values indicating whether cold air drainage conditions occur (1) or not (0)

Examples

```
# =====
# Mean daily climate for Lizard, Cornwall in 2010
# =====
h <- apply(huss, 3, mean, na.rm = TRUE)
p <- apply(pres, 3, mean, na.rm = TRUE)
tmin <- apply((tas - dtr), 3, mean, na.rm = TRUE)[2:364]
tmax <- apply((tas + dtr), 3, mean, na.rm = TRUE)[2:364]
# =====
# hourly climate 2nd Jan to 30 Dec 2010
# =====
h <- spline(h, n = 8737)$y[13:8724]
p <- spline(p, n = 8737)$y[13:8724]
n <- apply(cfc[,13:8724], 3, mean)
rdni <- apply(dnirad[,13:8724], 3, mean)
rdif <- apply(difrad[,13:8724], 3, mean)
jd <- julday(2010, 2, 1)
jd <- c(jd:(jd+362))
tc <- hourlytemp(jd, h, n, p, rdni, rdif, tmin, tmax, 50.05, -5.19)
ws10m <- spline(wind2010$wind10m, n = 8755)$y[25:8736]
ws1m <- windheight(ws10m, 10, 1)
# =====
# Calculate whether cold air drainage, persists and plot proportion
# =====
startjul <- julday(2010,1,2)
hist(cadconditions(h, tc, n, p, ws1m, startjul, 50.05, -5.19),
     main = "", xlab = "Cold air drainage conditions (1 = Y)")
```

canopy

Calculates canopy cover

Description

`canopy` is used to calculate fractional canopy cover.

Usage

```
canopy(l, x)
```

Arguments

- l** a raster object, two-dimensional array or matrix of leaf area index values as returned by `lai()`.
- x** a raster object, two-dimensional array of numeric values representing the ratio of vertical to horizontal projections of leaf foliage as returned by `leaf_geometry()`.

Details

Canopy cover calculated by this function is defined as 1 - the proportion of isotropic radiation transmitted through the canopy. If `l` is a raster object, a raster object is returned.

Value

a raster object or a two-dimensional array of numeric values representing fractional canopy cover estimated as the proportion of isotropic radiation transmitted through the canopy.

Examples

```
l <- lai(aerial_image[, , 3], aerial_image[, , 4])
l <- if_raster(l, dtm1m) # convert to raster
x <- leaf_geometry(veg_hgt)
fr <- canopy(l, x)
plot(fr, main = "Fractional canopy cover")
```

cfc

A 0.05° resolution dataset of hourly fractional cloud cover

Description

A dataset containing hourly fractional cloud cover values in 2010 for the area bounded by -5.40, -5.00, 49.90, 50.15 (xmin, xmax, ymin, ymax) (CRS: +init=epsg:4326).

Usage

```
cfc
```

Format

An array with 5 rows, 8 columns and 8670 hourly values

Source

<http://www.cmsaf.eu/>

coastalTps	<i>Calculates coastal effects using thin-plate spline</i>
------------	---

Description

coastalTps uses thin-plate spline interpolation to estimate the effects of coastal buffering of land-temperatures by the sea.

Usage

```
coastalTps(dT, lsw, lsa)
```

Arguments

dT	a coarse-resolution raster of sea - land temperatures (°C).
lsw	a fine-resolution raster of coastal exposure upwind, as produced by invls() .
lsa	a fine-resolution raster of mean coastal exposure in all directions.

Value

a fine-resolution raster of sea - land temperature differences (°C).

Examples

```
# =====
# Calculate land-sea temperature difference
# =====
temp <- tas[, , 1]
sst <- 10.665
dT <- if_raster(sst - temp, dtm1km)
# =====
# Obtain coastal exposure data
# =====
lsw <- landsearatio[, , 7] # upwind
lsa <- apply(landsearatio, c(1, 2), mean) # mean, all directions
lsw <- if_raster(lsw, dtm100m)
lsa <- if_raster(lsa, dtm100m)
# =====
# Calculate coastal effects using thin-plate spline and plot
# =====
dTf <- coastalTps(dT, lsw, lsa)
par(mfrow = c(2, 1))
plot(sst - dT, main = expression(paste("Temperature ", (~degree~C))))
plot(sst - dTf, main = expression(paste("Temperature ", (~degree~C))))
```

dailyprecipNCEP	<i>Obtain daily precipitation from NCEP</i>
-----------------	---

Description

Obtain daily precipitation from NCEP

Usage

```
dailyprecipNCEP(lat, long, tme, reanalysis2 = TRUE)
```

Arguments

lat	the latitude of the location for which data are required
long	the longitude of the location for which data are required
tme	a POSIXlt object covering the duration for which data are required. Intervals should be daily. Hourly data are returned irrespective of the time interval of tme.
reanalysis2	Logical. Should data be obtained from the Reanalysis II dataset (default) or from Reanalysis I (data prior to 1979).

Value

a vector of daily precipitations (mm / day) for the period covered by tme

Examples

```
tme <- as.POSIXlt(c(1:15) * 24 * 3600, origin = "2015-01-15", tz = 'UTC')
dailyprecipNCEP(50, -5, tme)
```

demworld	<i>A 2.5° resolution dataset of elevations</i>
----------	--

Description

A dataset global dataset elevation in metres (CRS: +init=epsg:4326)

Usage

```
demworld
```

Format

A raster object with 72 rows and 144 columns.

Source

<http://www.cmsaf.eu/>

difprop	<i>Calculates the diffuse fraction from incoming shortwave radiation</i>
---------	--

Description

difprop calculates proportion of incoming shortwave radiation that is diffuse radiation using the method of Skartveit et al. (1998) Solar Energy, 63: 173-183.

Usage

```
difprop(rad, julian, localtime, lat, long, hourly = FALSE, watts = TRUE,
        merid = 0, dst = 0)
```

Arguments

rad	a vector of incoming shortwave radiation values (either $MJm^{-2}hr^{-1}$ or Wm^{-2})
julian	the Julian day as returned by <code>julday()</code>
localtime	a single numeric value representing local time (decimal hour, 24 hour clock)
lat	a single numeric value representing the latitude of the location for which partitioned radiation is required (decimal degrees, -ve south of equator).
long	a single numeric value representing the longitude of the location for which partitioned radiation is required (decimal degrees, -ve west of Greenwich meridian).
hourly	species whether values of rad are hourly (see details).
watts	a logical value indicating whether the units of rad are Wm^{-2} (TRUE) or $MJm^{-2}hr^{-1}$ (FALSE).
merid	an optional single numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours in 24 hour clock) (+1 for BST, +0 for GMT).

Details

The method assumes the environment is snow free. Both overall cloud cover and heterogeneity in cloud cover affect the diffuse fraction. Breaks in an extensive cloud deck may primarily enhance the beam irradiance, whereas scattered clouds may enhance the diffuse irradiance and leave the beam irradiance unaffected. In consequence, if hourly data are available, an index is applied to detect the presence of such variable/inhomogeneous clouds, based on variability in radiation for each hour in question and values in the preceding and deciding hour. If hourly data are unavailable, an average variability is determined from radiation intensity.

Value

a vector of diffuse fractions (either $MJm^{-2}hr^{-1}$ or Wm^{-2}).

Examples

```
rad <- c(5:42) / 0.036 # typical values of radiation in W/m^2
jd <- julday(2017, 6, 21) # julian day
dfr <- difprop(rad, jd, 12, 50, -5)
plot(dfr ~ rad, type = "l", lwd = 2, xlab = "Incoming shortwave radiation",
     ylab = "Diffuse fraction")
```

difrad	<i>A 0.05° resolution dataset of hourly diffuse radiation</i>
--------	---

Description

A dataset containing hourly diffuse radiation values in 2010 ($MJm^{-2}hr^{-1}$) for the area bounded by -5.40, -5.00, 49.90, 50.15 (xmin, xmax, ymin, ymax) (CRS: +init=epsg:4326).

Usage

difrad

Format

An array with 5 rows, 8 columns and 8670 hourly values

Source

<http://www.cmsaf.eu/>

dnirad	<i>A 0.05° resolution dataset of hourly direct radiation normal to the direction of the solar beam</i>
--------	--

Description

A dataset containing hourly direct radiation values in 2010 ($MJm^{-2}hr^{-1}$) for the area bounded by -5.40, -5.00, 49.90, 50.15 (xmin, xmax, ymin, ymax) (CRS: +init=epsg:4326).

Usage

dnirad

Format

An array with 5 rows, 8 columns and 8670 hourly values

Source

<http://www.cmsaf.eu/>

dtm100m	<i>A 100 m resolution raster object of elevation for the Lizard Peninsula, Cornwall, UK.</i>
---------	--

Description

A raster object containing elevation in metres with sea coded as NA for the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

dtm100m

Format

A raster object with 187 rows and 214 columns.

Source

<http://www.tellusgb.ac.uk/>

dtm1km	<i>A 1 km resolution raster object of elevation for the Lizard Peninsula, Cornwall, UK.</i>
--------	---

Description

A raster object containing elevation in metres with sea coded as NA for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

dtm1km

Format

A raster object with 19 rows and 22 columns.

Source

<http://www.tellusgb.ac.uk/>

dtm1m	<i>A 1 m resolution raster object of elevation for part of the Lizard Peninsula, Cornwall, UK.</i>
-------	--

Description

A raster object containing elevation in metres with sea coded as NA for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

dtm1m

Format

A raster object with 1000 rows and 1000 columns.

Source

<http://www.tellusgb.ac.uk/>

dtr	<i>A 1 km resolution dataset of diurnal temperature ranges</i>
-----	--

Description

A spatially interpolated dataset containing diurnal temperature ranges (°C) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

dtr

Format

An array with 19 rows, 22 columns and 365 daily values

Source

<https://www.metoffice.gov.uk/>

fitmicro

*Fits micro- or mesoclimate model***Description**

fitmicro is used to fit a micro- or mesoclimate model using field temperature readings, and estimates of reference temperature, net radiation and wind speed at the locations of those readings.

Usage

```
fitmicro(microfitdata, alldata = FALSE, windthresh = NA, iter = 999)
```

Arguments

microfitdata	a data.frame with at least the following columns (see, for example, microfit data): temperature microclimate temperature readings reftemp Reference (e.g. coarse-scale or weather station) temperatures wind Wind speeds netrad Net radiation values
alldata	an optional logical value indicating whether to fit the model using all data (TRUE) or using a randomization procedure (FALSE). See details.
windthresh	an optional single numeric value indicating the threshold wind speed above which an alternative linear relationship between net radiation the microclimate temperature anomaly is fitted. See details.
iter	a single integer specifying the iterations to perform during randomization. Ignored if alldata = TRUE.

Details

If modelling mesoclimate, it is assumed that altitudinal, coastal and cold-air drainage effects have already been accounted for in the calculation of reftemp. It is therefore assumed that the most important energy fluxes determining near-surface temperature are those due to the radiation flux and convection that occurs at the surface-atmosphere boundary. Heat fluxes into the soil and latent heat exchange are considered to be small and proportional to the net radiation flux, and the heat capacity of the vegetation is considered to be small so that, compared to the time-scale of the model, surface temperature rapidly reach equilibrium. In consequence, the difference between the near-ground temperature and the ambient temperature is a linear function of netrad. The gradient of this linear relationship is a measure of the thermal coupling of the surface to the atmosphere. If this relationship is applied to vegetation, assuming the canopy to act like a surface, while air density and the specific heat of air at constant pressure are constant, the slope varies as a function of a wind speed factor, such that different slope values are assumed under high and low wind conditions. Hence, fitmicro fits a linear model of the form $\text{lm}((\text{temperature} - \text{reftemp}) \sim \text{netrad} * \text{windfact})$ where windfact is given by $\text{ifelse}(\text{wind} > \text{windthresh}, 1, 0)$. If all data is FALSE, random subsets of the data are selected and the analyses repeated iter times to reduce the effects of temporal autocorrelation. Parameter estimates are derived as the median of all runs. If no value is provided for windthresh, it is derived by iteratively trying out different values, and selecting that which yields the best fit. The gradient of the relationship is also dependent on vegetation structure, and in some circumstances it may therefore be advisable to fit separate models for each vegetation type.

Value

a data.frame with the following columns:

Estimate parameter estimates and windthresh

Std.Dev Standard deviation of parameter estimates

P Two-tailed p-value

Examples

```
fitmicro(microfitdata)
fitmicro(mesofitdata, alldata = TRUE)
```

flowacc	<i>Calculates accumulated flow</i>
---------	------------------------------------

Description

flowacc is used by [pcad\(\)](#) to calculate accumulated flow to each cold air drainage basin

Usage

```
flowacc(dem)
```

Arguments

dem a raster object, two-dimensional array or matrix of elevations.

Details

Accumulated flow is expressed in terms of number of cells.

Value

a raster object, two-dimensional array or matrix of accumulated flow.

Examples

```
fa <- flowacc(dtm100m)
plot(fa, main = 'Accumulated flow')
```

get_dem	<i>Gets raster of elevations</i>
---------	----------------------------------

Description

Several web services provide access to raster elevations. This function provides access to the Registry of Open Data on AWS to retrieve a raster object of elevations.

Usage

```
get_dem(r = NA, lat, long, resolution = 30, zmin = 0)
```

Arguments

r	optional raster object covering the location for which elevations are required
lat	the latitude (in decimal degrees) of the location for which elevations are required. Not used if r is a raster.
long	the longitude (in decimal degrees) of the location for which elevations are required. Not used if r is a raster
resolution	the resolution (in metres) of the required dem (see details).
zmin	assumed sea-level height. Values below this are set to zmin.

Details

Currently, `get_dem` utilises the Amazon Web Services (AWS) (<https://aws.amazon.com/public-datasets/terrain/>) to retrieve dems. The spatial resolution of the data available vary by location (see <https://mapzen.com/documentation/terrain-tiles/data-sources/> for details). Global coverage is ~30m resolution and if the resolution of data requested is higher than that available, elevations are derived by interpolation. If r is unspecified, then lat and long are used to retrieve a 200 x 200 cell raster centred on lat and long. Bathymetry is also returned automatically from AWS, so zmin allows the user to specify a sea-level height. Elevations below zmin are set at zmin. For use with microclima the returned raster has a Mercator Coordinate Reference System if the latitude specified by lat is between 80 degrees S and 84 degrees N. In polar regions WGS 84 / NSIDC Sea Ice Polar Stereographic projections are used. X, y and Z units are all in metres. If r is specified, the Coordinate Reference System of r is used.

Value

a raster object of elevations in metres.

Examples

```
dem50m <- get_dem(dtm100m, resolution = 50)
plot(dem50m) # 50m raster, OSGB projection system

dem5m <- get_dem(lat = 49.97, long = -5.22, resolution = 5)
plot(dem5m) # 5m raster, Mercator projection system
```

get_NCEP

Obtains NCEP data required to run microclima

Description

Obtains NCEP data required to run microclima

Usage

```
get_NCEP(lat, long, tme, reanalysis2 = TRUE)
```

Arguments

lat	the latitude of the location for which data are required
long	the longitude of the location for which data are required
tme	a POSIXlt object covering the duration for which data are required. Hourly data are returned irrespective of the time interval of tme.
reanalysis2	Logical. Should data be obtained from the Reanalysis II dataset (default) or from Reanalysis I (data prior to 1979).

Details

The NCEP-NCAR and NCEP-DOE Atmospheric Model Intercomparison Project (Kanamitso et al 2002) provides estimates of a range of climate and radiation data for the period 1948 to present. Data at six-hourly intervals are available at a spatial resolution of ~ 1.8 to 2.5° . The function downloads the following datasets: mean, min and max air temperature at 2m, specific humidity and 2m, surface pressure, the u and v wind vectors at 10, the downward and upward longwave radiation fluxes at the surface and the downward shortwave flux at the surface. Data for days either side of tme are downloaded to aid with interpolation by [hourlyNCEP\(\)](#). Requires internet connection.

Value

a dataframe with the following variables: (i) obs_time: times in YYYY-MM-DD HH:MM format (UTC), (ii) Tk: mean temperatures at 2m (K), (iii) Tkmin: minimum temperatures at 2m (K) (iv) Tkmax: maximum temperatures at 2m (K), (v) sh: specific humidity at 2m (Kg / Kg), (vi) pr: surface pressure (Pa), (vii) wu u wind vector at 10m (m/s) (viii) wv: v wind vector at 10m (m/s) (ix) dlw: downward longwave radiation flux (Wm^{-2}), (x) ulw: upward longwave radiation flux (Wm^{-2}), (xi) dsw: downward shortwave radiation flux (Wm^{-2}), (xii) tcdc: Total cloud cover for the entire atmosphere.

References

Kanamitsu M, Ebisuzaki W, Woollen J, Yang SK, Hnilo JJ, Fiorino M, Potter GL (2002) Ncep-doe amip-ii reanalysis (r-2). Bulletin of the American Meteorological Society 83: 1631-1644.

See Also

[hourlyNCEP\(\)](#)

Examples

```
tme <- as.POSIXlt(c(1:15) * 24 * 3600, origin = "2015-01-15", tz = 'UTC')
head(get_NCEP(50, -5, tme))
```

horizonangle	<i>Calculates the tangent of the horizon angle</i>
--------------	--

Description

horizonangle is used to calculate the tangent of the angle to the horizon in a specified direction.

Usage

```
horizonangle(dtm, azimuth, res = 1)
```

Arguments

dtm	a raster object, two-dimensional array or matrix of elevations (m). If not a raster, orientated as if derived from a raster using is_raster() . I.e. [1, 1] is the NW corner.
azimuth	a numeric value representing the direction of the horizon as, for example, returned by solazi() (° from north).
res	a single numeric value representing the spatial resolution of dtm (m).

Details

To enable calculation of horizon angles near the edge of dtm a 100 pixel buffer is of zeros is placed around it. NAs in dtm are converted to zeros. The projection system used must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If dtm is a raster object, a raster object is returned.

Value

a raster object or two-dimensional array of numeric values representing the tangent of the angle to the horizon in a specified direction.

Examples

```
ha <- horizonangle(dtm1m, 0)
plot(ha, main = "Tangent of angle to horizon")
```

hourlyNCEP

*Interpolate NCEP data for running microclima to hourly***Description**

hourlyNCEP optionally downloads the required NCEP climate and radiation forcing data required for running microclima and interpolates 4x daily data to hourly.

Usage

```
hourlyNCEP(ncepdata = NA, lat, long, tme, reanalysis2 = TRUE)
```

Arguments

ncepdata	an optional data frame of climate variables as returned by get_ncep() .
lat	the latitude of the location for which data are required. Ignored if ncepdata provided.
long	the longitude of the location for which data are required. Ignored if ncepdata provided.
tme	a POSIXlt object covering the duration for which data are required. Ignored if ncepdata`` provided. Hourly data are returned irrespective of the time interval of tme provided.
reanalysis2	Logical. Should data be obtained from the Reanalysis II dataset (default) or from Reanalysis I (data prior to 1979). Ignored if ncepdata provided.

Details

If ncepdata is not provided, then [get_ncep\(\)](#) is called and data are downloaded from NCEP Atmospheric Model Intercomparison Project (Kanamitso et al 2002). Six-hourly data are interpolated as follows. Pressure, humidity and the u and v wind vectors are converted to hourly using spline interpolation. Wind speed and direction and then calculated and adjusted to give values at 1 m using [windheight\(\)](#). The diffuse radiation proportion is calculated using [difprop\(\)](#), and hourly obtained by adjusting for the direction of the solar beam and airmass thickness using functions [siflat\(\)](#) and [airmasscoef\(\)](#). Hourly temperature are derived using [hourlytemp\(\)](#) and long-wave radiation by splining emissivity values.

Value

a dataframe with the following variables: (i) obs_time: a POSIXlt oobject of times, (ii) temperature: temperatures at 2m (°C), (iii) humidity: specific humidity at 2m (Kg / Kg), (iv) pressure: surface pressure (Pa), (v) windspeed: wind speed at 2m (metres per second), (vi) winddir: wind direction (degrees from N), (vii) emissivity: emissivity of the atmosphere (0 - 1), (viii) netlong: Net longwave radiation ($MJm^{-2}hr^{-1}$) (ix) uplong: upward longwave radiation ($MJm^{-2}hr^{-1}$), (x) downlong: downward longwave radiation ($MJm^{-2}hr^{-1}$), (xi) rad_dni: Direct radiation normal to the solar beam ($MJm^{-2}hr^{-1}$), (xii) rad_dif: Diffuse radiation ($MJm^{-2}hr^{-1}$), (xiii) szenith: the zenith angle (degrees), (ixx) cloudcover: cloud cover (

See Also

[get_NCEP\(\)](#)

Examples

```
tme <- as.POSIXlt(c(0:30) * 24 * 3600, origin = "2015-01-15 00:00", tz = "UTC")
# NB takes a while to download data
hdata<- hourlyNCEP(NA, 50, -5, tme)
head(hdata)
plot(temperature ~ as.POSIXct(obs_time), data = hdata, type = "l", xlab = "Month")
plot(rad_dif ~ as.POSIXct(obs_time), data = hdata, type = "l", xlab = "Month")
```

hourlytemp

*Derives hourly temperatures from daily data***Description**

hourlytemp is used to derive hourly temperatures from daily maxima and minima.

Usage

```
hourlytemp(julian, em = NA, h, n, p = 100346.13, dni, dif, mintemp, maxtemp,
  lat, long, merid = 0, tz = 0, dst = 0)
```

Arguments

julian	vector of julian days expressed as integers for every day for which mintemp and maxtemp are provided, as returned by function julday() .
em	an optional vector of hourly emissivities. If not provided, calculated from h, n and p.
h	a vector of hourly specific humidities ($kgkg^{-1}$). Ignored if em provided.
n	a vector of hourly fractional cloud cover values (range 0 - 1). Ignored if em provided.
p	an optional vector of hourly atmospheric pressure values (Pa). Ignored if em provided.
dni	a vector of hourly direct radiation values normal to the solar beam ($MJm^{-2}hr^{-1}$).
dif	a vector of hourly diffuse radiation values ($MJm^{-2}hr^{-1}$).
mintemp	a vector of daily minimum temperatures ($^{\circ}C$).
maxtemp	a vector of daily maximum temperatures ($^{\circ}C$).
lat	a single numeric value representing the latitude of the location for which hourly temperatures are required (decimal degrees, -ve south of equator).
long	a single numeric value representing the longitude of the location for which hourly temperatures are required (decimal degrees, -ve west of Greenwich meridian).
merid	an optional numeric value representing the longitude of the local time zone meridian ($^{\circ}$) (0 for UK time).
tz	an optional single numeric value or vector of values specifying the time zones expressed as hours different from GMT for each day (-ve to west).
dst	a single numeric value or vector of values representing the local summer time adjustment for each day (hours, e.g. +1 for BST).

Details

A warning is returned if any of following conditions are not met. (1) h, n, dif and dct differ in length. (2) julian, mintemp and maxtemp differ in length. (3) E.g. h / 24 is not an integer. (4) the length of e.g. h is not equal to the length of e.g. mintemp x 24.

Value

a vector of hourly temperatures (°C).

Examples

```
jd <- julday(2010, 5, c(1,2))
ht <- hourlytemp(jd, NA, microvars$humidity[1:48], microvars$cloudcover[1:48],
  microvars$pressure[1:48], dni = microvars$dni[1:48],
  dif = microvars$dif[1:48], c(7.5, 7.2), c(14.6, 15.2),
  49.968, -5.216)
plot(ht ~ c(0:47), type="l", xlab = "Hour", ylab = "Temperature",
  main = paste("tmins:", 7.2, 7.5, "tmaxs:", 14.6, 15.2))
```

humidityconvert

Converts between different measures of humidity

Description

humidityconvert is used to convert between different measures of humidity, namely relative, absolute or specific. Vapour pressure is also returned.

Usage

```
humidityconvert(h, intype = "relative", tc = 20, p = 101300)
```

Arguments

h	humidity value(s). Units as follows: specific humidity ($kg\,kg^{-1}$), absolute humidity ($kg\,m^{-3}$), relative humidity (%), vapour pressure (kPa).
intype	a character string description of the humidity type of h. One of "relative", "absolute" or "specific".
tc	A numeric value specifying the temperature (°C).
p	An optional numeric value specifying the atmospheric pressure (Pa).

Details

This function converts between vapour pressure and specific, relative and absolute humidity, based on sea-level pressure and temperature. It returns a list of relative, absolute and specific humidity and vapour pressure. If intype is unspecified, then h is assumed to be relative humidity. If p is unspecified, pressure assumed to be 101300, a typical value for sea-level pressure. If one or more of the relative humidity values exceeds 100% a warning is given.

Value

a list of numeric humidity values with the following components:

relative relative humidity (%).

absolute absolute humidity (kgm^{-3}).

specific specific humidity ($kgkg^{-1}$).

vapour_pressure vapour pressure (kPa).

Examples

```
humidityconvert(90, 'relative', 20)
humidityconvert(0.01555486, 'absolute', 20)
humidityconvert(0.01292172, 'specific', 20)
```

huss	<i>A 1 km resolution dataset of interpolated daily specific humidity values</i>
------	---

Description

A dataset containing daily specific humidity values ($kgkg^{-1}$) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

```
huss
```

Format

An array with 19 rows, 22 columns and 365 daily values.

Source

<https://eip.ceh.ac.uk/chess/>

if_raster	<i>Flexible conversion to raster object</i>
-----------	---

Description

if_raster is used to permit flexibility in the use of rasters, matrices or arrays in many functions.

Usage

```
if_raster(x, r)
```

Arguments

x	an R object
r	an R object

Value

if *r* is a raster, *x* is converted to a raster with the same attributes as *r*, otherwise returns *x*

Examples

```
r <- is_raster(dtm100m)
r1 <- if_raster(r, dtm100m)
r2 <- if_raster(r, r)
class(r1) # is a RasterLayer
class(r2) # is a matrix
```

invls	<i>Calculates land to sea ratio in upwind direction</i>
-------	---

Description

invls is used to calculate an inverse distance² weighted ratio of land to sea in a specified upwind direction.

Usage

```
invls(landsea, e, direction, maxdist = 2e+05)
```

Arguments

landsea	A raster object with NAs (representing sea) or any non-NA value (representing land). The object should have a larger extent than that for which land-sea ratio values are needed, as the calculation requires land / sea coverage to be assessed upwind outside the target area.
e	an extent object indicating the region for which land-sea ratios are required.
direction	a single numeric value specifying the direction (decimal degrees) from which the wind is blowing.
maxdist	The maximum distance (in the same units as landsea) over which land-sea ratios should be calculated. If the maximum distance is greater than the extent of landsea, then the area outside landsea is ignored.

Details

This function calculates a coefficient of the ratio of land to sea pixels in a specified upwind direction, across all elements of a raster object, weighted using an inverse distance squared function, such that nearby pixels have a greater influence on the coefficient. It returns a raster object representing values ranging between zero (all upwind pixels sea) to one (all upwind pixels land).

Value

a raster object with distance-weighted proportions of upwind land pixels

Examples

```
ls1 <- invls(dtm100m, extent(dtm1m), 180)
ls2 <- invls(dtm100m, extent(dtm1m), 270)
par(mfrow=c(2,1))
plot(ls1, main = "Land to sea weighting, southerly wind")
plot(ls2, main = "Land to sea weighting, westerly wind")
```

is_raster

*Checks whether object is a raster and returns a matrix if yes***Description**

is_raster is used to permit flexibility in the use of rasters, matrices or arrays in many functions.

Usage

```
is_raster(r)
```

Arguments

r an R object

Value

if r is a raster, returns a matrix containing all values of r, otherwise returns r

Examples

```
r <- is_raster(dtm100m)
class(dtm100m) # is a RasterLayer
class(r) # is a matrix
plot(r) # not a raster
plot(raster(r)) # converts to raster
```

julday

*Calculates the astronomical Julian day***Description**

julian is used to calculate the astronomical Julian day (days since since January 1, 4713 BCE at noon UTC) from a given year, month and day.

Usage

```
julday(year, month, day, hour = 12, min = 0, sec = 0, tz = 0)
```

Arguments

year	year (AD).
month	month in numeric form (1-12).
day	days of the month (1-31).
hour	hours (decimal, 0-23).
min	minutes (decimal, 0-59).
sec	seconds (decimal, 0-59).
tz	an optional numeric value specifying the time zones expressed as hours different from GMT (-ve to west).

Value

Julian Day. I.e. the number of days since January 1, 4713 BCE at noon UTC.

Examples

```
jd1 <- julday(2010, 1, 31)
jd2 <- julday(2010, 1, 31, 11, 0, 0)
jd1 - jd2
```

lai	<i>Calculates Leaf Area Index</i>
-----	-----------------------------------

Description

lai is used to calculate the total one-sided area of leaf tissue per unit ground surface area from the Normalized Difference Vegetation Index.

Usage

```
lai(red, nir, maxlai = 20)
```

Arguments

red	a raster object, two-dimensional array or matrix of reflectance values in the red spectral band.
nir	a raster object, two-dimensional array or matrix of reflectance values in the near-infrared spectral band.
maxlai	an optional single numeric value representing the likely upper limit of Leaf Area Index to which all values are capped.

Details

If red is a raster object, a raster object is returned. This function has been calibrated using data derived from a small area of Cornwall only. It is strongly recommended that locally calibrated values are obtained.

Value

a raster object or two-dimensional array of numeric values representing the total one-sided area of leaf tissue per unit ground surface area.

See Also

function `lai_adjust()` for calculated leaf area index values at specified heights above the ground.

Examples

```
leaf <- lai(aerial_image[,3], aerial_image[,4])
plot(if_raster(leaf, dtm1m), main = "Leaf area index")
```

lai_adjust	<i>Calculates Leaf Area Index for specified height above ground</i>
------------	---

Description

`lai_adjust` is used to adjust the total one-sided area of leaf tissue per unit ground surface area to derive values at a specified height above the ground.

Usage

```
lai_adjust(l, veght, hgt = 0.05)
```

Arguments

<code>l</code>	raster object, two-dimensional array or matrix of leaf area index values as returned by <code>lai()</code> .
<code>veght</code>	a raster object, two-dimensional array or matrix of vegetation heights (m).
<code>hgt</code>	a numeric value representing the height above the ground for which Leaf Area Index is required (m).

Details

If `l` is a raster object, a raster object is returned. Temperatures are often required for a specified height above the ground, and in short vegetation, the leaf area can be substantially less at this height than at ground level. This function enables the user to estimate leaf area for a specified height about the ground.

Value

a raster object or a two-dimensional area of numeric values representing the Leaf Area Index values for a specified height above the ground

Examples

```
l <- lai(aerial_image[,3], aerial_image[,4])
la<-lai_adjust(l, veg_hgt)
par(mfrow=c(2, 1))
plot(if_raster(l, dtm1m), main = "Leaf area index")
plot(if_raster(la, dtm1m), main = "Adjusted leaf area index")
```

landsearatio	<i>Inverse-distance weighted land-sea ratios in 36 directions</i>
--------------	---

Description

A 100m resolution dataset of nverse-distance weighted land-sea ratios, as produced by function `invls()` in each of 36 directions (0°, 10°.350°) for the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

```
landsearatio
```

Format

An array with 187 rows, 214 columns and 36 directional values.

lapserate	<i>Calculates the moist adiabatic lapse rate</i>
-----------	--

Description

lapserate is used to calculate changes in temperature with height.

Usage

```
lapserate(tc, h, p = 101300)
```

Arguments

tc	a single numeric value, raster object, two-dimensional array or matrix of temperature (°C).
h	a single numeric value, raster object, two-dimensional array or matrix of specific humidity ($kg\,kg^{-1}$).
p	an optional single numeric value, raster object, two-dimensional array or matrix of atmospheric pressure (Pa).

Details

if tc is a raster, a raster object is returned. This function calculates the theoretical lapse rate. Environmental lapse rates can vary due to winds.

Value

the lapse rate (m^{-1}).

Examples

```
lapserate(20, 0) * 1000 # dry lapse rate per km
h <- humidityconvert(100, intype = "relative", 20)
lapserate(20, h$specific) * 1000 # lapse rate per km when relative humidity is 100%
```

latlongfromraster	<i>Derives latitude and longitude of centre of raster object</i>
-------------------	--

Description

latlongfromraster is used to calculate the latitude and longitude of the centre of a raster object.

Usage

```
latlongfromraster(r)
```

Arguments

r a raster object with the coordinate reference system defined by [crs\(\)](#)

Value

a data.frame with the latitude and longitude of the centre of the raster

Examples

```
latlongfromraster(dtm1m)
latlongfromraster(dtm100m)
```

leaf_geometry	<i>Calculates leaf orientation</i>
---------------	------------------------------------

Description

leaf_geometry is used to calculates the ratio of vertical to horizontal projections of leaf foliage.

Usage

```
leaf_geometry(veghgt, maxx = 20)
```

Arguments

veghgt a raster object, two-dimensional array or matrix of vegetation heights (m).
maxx a theoretical upper limit for the ratio of vertical to horizontal projections of leaf foliage, to which all values are capped.

Details

Under vegetated canopies, canopy transmission not only decreases with canopy cover but is also affected by leaf structure. At low solar angles, radiation is lower when leaves are more vertically oriented and leaf_geometry is hence used to calculate an approximate factor indicating the degree to which vegetation is vertically orientated based on the premise that shorter vegetation is more likely to have more vertically orientated leaves. If vegght is a raster object, a raster object is returned. This function has been calibrated using data derived from a small area of Cornwall only. It is strongly recommended that locally calibrated values units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system.

Value

a raster object or a two-dimensional array of numeric values representing the ratio of vertical to horizontal projections of leaf foliage. The output tends towards zero as vegetation is more vertically orientated and maxx as it is more horizontally orientated.

Examples

```
x <- leaf_geometry(veg_hgt)
plot(x, main = "Leaf geometry")
```

longwavetopo	<i>calculates net longwave radiation above canopy</i>
--------------	---

Description

longwavetopo is used to calculate a high-resolution dataset of the net longwave radiation flux density emitted from the Earth, ignoring canopy effects.

Usage

```
longwavetopo(h, tc, p = 101300, n, svf = 1)
```

Arguments

h	a single numeric value, raster object, two-dimensional array or matrix of specific humidities ($kg\,kg^{-1}$).
tc	a single numeric value, raster object, two-dimensional array or matrix of temperatures ($^{\circ}C$).
p	an optional single numeric value, raster object, two-dimensional array or matrix of sea-level pressures (Pa).
n	a single numeric value, raster object, two-dimensional array or matrix of fractional cloud cover values (range 0 - 1).
svf	an optional single value, raster object, two-dimensional array or matrix of values representing the proportion of isotropic radiation received by a partially obscured surface relative to the full hemisphere, as returned by skyviewtopo() .

Details

if svf is a raster object, a raster object is returned. If no values for p are provided, a default value of 101300 Pa, typical of sea-level pressure, is assumed. If single values of h, tc, p and n are given, and svf is an array or matrix, then the entire area is assumed to have the same values of h, tc, p and n. If no value for svf is provided then the entire hemisphere is assumed to be in view. If single values of h, tc, p and n are given, and no value of 'svf' is provided, a single value is returned, and it is assumed that the entire hemisphere is in view.

Value

a single numeric value, raster object, two-dimensional array or matrix of values representing net longwave radiation (MJ per metre squared per hour).

See Also

The function `longwaveveg()` returns the net longwave radiation under vegetation. The function `humidityconvert()` can be used to derive specific humidity from other measures of humidity.

Examples

```
# =====
# Extract data for 2010-05-24 11:00
# =====
h <- huss[, ,144]
p <- pres[, ,144]
tc <- tas[, ,144] + dtr[, ,144]
n <- cfc[, ,3444]
sv <- skyviewtopo(dtm100m)
# =====
# Resample to 100m resolution
# =====
hr <- if_raster(h, dtm1km)
tr <- if_raster(tc, dtm1km)
pr <- if_raster(p, dtm1km)
nr <- raster(n, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
crs(nr) <- '+init=epsg:4326'
nr <- projectRaster(nr, crs = '+init=epsg:27700')
hr <- resample(hr, dtm100m)
tr <- resample(tr, dtm100m)
pr <- resample(pr, dtm100m)
nr <- resample(nr, dtm100m)
# =====
# Calculate and plot net longwave radiation
# =====
netlong100m <- longwavetopo(hr, tr, pr, nr, sv)
netlong100m <- mask(netlong100m, dtm100m)
plot(netlong100m, main = "Net longwave radiation")
```

longwaveveg

Calculates net longwave radiation below canopy

Description

longwaveveg is used to calculate a high-resolution dataset of the net longwave radiation flux density emitted from the Earth, accounting for canopy effects.

Usage

```
longwaveveg(h, tc, p = 101300, n, x, fr, svv = 1, albc = 0.23)
```

Arguments

- | | |
|----|--|
| h | a single numeric value, raster object, two-dimensional array or matrix of specific humidities ($kg\,kg^{-1}$). |
| tc | a single numeric value, raster object, two-dimensional array or matrix of temperatures ($^{\circ}C$). |

p	an optional single numeric value, raster object, two-dimensional array or matrix of sea-level pressures (Pa).
n	a single numeric value, raster object, two-dimensional array or matrix of fractional cloud cover (range 0 - 1).
x	a raster object, two-dimensional array or matrix of numeric values representing the ratio of vertical to horizontal projections of leaf foliage as returned by leaf_geometry() .
fr	a raster object, two-dimensional array or matrix of fractional canopy cover as returned by canopy() .
svv	an optional raster object, two-dimensional array or matrix of values representing the proportion of isotropic radiation received by a surface partially obscured by topography relative to the full hemisphere underneath vegetation as returned by skyviewveg() .
albc	an optional single value, raster object, two-dimensional array or matrix of values representing the albedo(s) of the vegetated canopy as returned by albedo2() .

Details

If svv is a raster object, a raster object is returned. If no values for p are provided, a default value of 101300 Pa, typical of sea-level pressure, is assumed. If no value for albc is provided, then the entire area is assumed to have a default value of 0.23, typical of well-watered grass. If single values of h, tc, p, n or albc are given, then the entire area is assumed to have the same values. If no value for svv is provided then the entire hemisphere is assumed to be in view.

Value

a single numeric value, raster object or two-dimensional array of values representing net longwave radiation (MJ per metre squared per hour).

See Also

The function [longwavetopo\(\)](#) returns the net longwave radiation above vegetation. The function [humidityconvert\(\)](#) can be used to derive specific humidity from other measures of humidity.

Examples

```
# =====
# Extract data for 2010-05-24 11:00
# =====
h <- microvars$humidity[564]
p <- microvars$pressure[564]
n <- microvars$cloudcover[264]
tcr <- raster(temp100[,564], xmn = 169000, xmx = 170000, ymn = 12000, ymx = 13000)
tc <- resample(tcr, dtm1m) # Resample temperature raster to 1m
# =====
# calculate input variables
# =====
x <- leaf_geometry(veg_hgt)
l <- lai(aerial_image[,3], aerial_image[,4])
l <- lai_adjust(l, veg_hgt)
svv <- skyviewveg(dtm1m, l, x)
fr <- canopy(l, x)
alb <- albedo(aerial_image[,1], aerial_image[,2], aerial_image[,3],
```

```

      aerial_image[,4])
albc <- albedo2(alb, fr, ground = FALSE)
# =====
# calculate and plot longwave radiation
# =====
netlong1m <- longwaveveg(h, tc, p, n, x, fr, svv, albc)
nlr <- mask(netlong1m, dtm1m)
plot(nlr, main = "Net longwave radiation")

```

mean_slope	<i>Calculates the mean slope to the horizon</i>
------------	---

Description

mean_slope is used to calculates the mean slope to the horizon in all directions.

Usage

```
mean_slope(dtm, steps = 36, res = 1)
```

Arguments

dtm	a raster object, two-dimensional array or matrix of elevations (m).
steps	an optional integer. The mean slope is calculated from the horizon angle in specified directions. Steps defines the total number of directions used. If the default 36 is specified, then the horizon angle is calculated at 10° intervals.
res	a single numeric value representing the spatial resolution of dtm (m).

Details

If dtm is a raster object, a raster object is returned. The projection system associated with dtm must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system.

Value

a raster object or a two-dimensional array of the mean slope angle to the horizon in all directions (°).

Examples

```

ms <- mean_slope(dtm100m, res = 100)
plot(ms, main = "Mean slope to horizon")

```

mesofitdata	<i>2010 Data for fitting mesoclimate model.</i>
-------------	---

Description

A dataset containing the hourly temperature logger data and net radiation and wind data at each logger location in May 2010.

Usage

```
mesofitdata
```

Format

A data frame with 65772 rows and 7 variables:

obs_time The time of the logger recording, in DD/MM/YY HH:MM:SS format

temperature The mean hourly logger temperature reading ($^{\circ}\text{C}$)

reftemp The predicted reference temperature at the logger location as derived from coarse-scale data after adjusted for elevation, cold air drainage and coastal effects ($^{\circ}\text{C}$)

wind The predicted wind speed at the logger locations (ms^{-1})

netrad The predicted net radiation at the logger locations ($\text{MJm}^{-2}\text{hr}^{-1}$)

microclimaforNMR	<i>Extract NCEP data and run microclima for use with NicheMapR</i>
------------------	--

Description

Extract NCEP data and run microclima for use with NicheMapR

Usage

```
microclimaforNMR(lat, long, dstart, dfinish, l, x, hourlydata = NA,
  dailyprecip = NA, dem = NA, albr = 0.15, resolution = 30, zmin = 0,
  slope = NA, aspect = NA, windthresh = 4.5, emthresh = 0.78,
  reanalysis2 = TRUE)
```

Arguments

lat	the latitude (in decimal degrees) of the location for point data are required.
long	the longitude (in decimal degrees) of the location for point data are required.
dstart	start date as character of time period required in format DD/MM/YYYY
dfinish	end date as character of time period required in format DD/MM/YYYY
l	a single numeric value of the leaf area index for location (see details).
x	a single numeric value representing the ratio of vertical to horizontal projections of leaf foliage for the location (see details).

hourlydata	an optional data frame of ourly weather and radiation data as returned by function <code>hourlyNCEP()</code> . Function called if data not provided.
dailyprecip	an optional data frame of ourly weather and radiation data as returned by function <code>dailyprecipNCEP()</code> . Function called if data not provided.
dem	optional raster object of elevations covering the location for which point data are required. If not provided, one is downloaded from the registry of Open Data on AWS.
albr	albedo of ground surface from which radiation is reflected (see details)
resolution	the resolution (in metres) of the dem used for downscaling (see details).
zmin	assumed sea-level height. Values below this are set to zmin (see details).
slope	an optional slope value for the location in decimal degrees. If not specified, then the slope is calculated from the retrieved dem.
aspect	an optional aspect value for the location in decimal degrees. If not specified, then the slope is calculated from the retrieved dem.
windthresh	an optional threshold wind value (m /s) above which cold air drainage is assumed not to occur
emthresh	an optional threshold emissivity value above which cold air drainage is assumed not to occur
reanalysis2	Logical. Should data be obtained from the Reanalysis II dataset (default) or from Reanalysis I (data prior to 1979).

Details

The package NicheMapR (Kearney & Porter 2016), includes a suite of programs for mechanistic modelling of heat and mass exchange and provides a time-series of hourly estimates of above- and below-ground conditions. It has the significant advantage over microclima in that it works entirely from first principles does not require measurements of temperature for calibration, but does not explicitly model elevation and cold-air drainage effects. The slope, aspect and canopy effects must be specified by the user, and their effect on radiation are more simplistically modelled than by microclima.

Value

a list with the following objects:

- (1) `hourlydata`: a data frame of hourly climate data from NCEP, as returned by `hourlyNCEP()`.
- (2) `hourlyradwind`: A data frame with seven columns: (i) `swrad`: total downward shortwave radiation as a function of slope, aspect and canopy shading ($MJm^{-2}hr^{-1}$) as returned by `shortwaveveg()`. (ii) `skyviewfact`: a topographic sky view correction factor as returned by `skyviewveg()`. (iii) `canopyfact`: a canopy shading factor weighted by the ratio of direct to diffuse radiation. (iv) `whselt`: a ground level wind shelter coefficient as returned by `windcoef()`. (v) `windspeed`: wind speed (m /s) at 1 m, corrected for topographic sheltering. (vi) `slope`: the slope (decimal degrees) of the location specified by `lat` and `long`. Either the same as that input, or determined from `dem`. (vii) `aspect`: the aspect (decimal degrees) of the location specified by `lat` and `long`. Either the same as that input, or determined from `dem`.
- (3) `tref`: a dataframe with five columns (i) `tref` the reference temperature, derived from NCEP data. (ii) `elev` the elevation of the location. (iii) `elevncep`: the mean elevation of the 2.5° grid cell corresponding to that from which NCEP data were derived. (iv) `telev` the expected difference in temperature due to elevation, calculated by applying `lapse_rate()`. (v) `tcad` the expected difference in temperature due to cold air drainage effects, determined using `pcad()` and `cadconditions()`.
- (4) `dailyprecip` a vector of daily rainfalls derived from NCEP.

References

Kearney MR, Porter WP (2016) NicheMapR – an R package for biophysical modelling: the microclimate model. *Ecography* 40: 664-674.

See Also

Function `get_dem()` for retrieving digital elevation data and `hourlyNCEP()` for retrieving and processing hourly data from NCEP.

Examples

```
mnr <- microclimaforNMR(50, -5.2, '15/01/2015', '15/02/2015', 1, 1)
head(mnr$hourlydata)
head(mnr$hourlymicrolyradwind)
head(mnr$tref)
head(mnr$dailyprecip)
```

microfitdata

May 2010 Data for fitting microclimate model.

Description

A dataset containing the hourly temperature logger data and net radiation and wind data at each logger location in May 2010.

Usage

```
microfitdata
```

Format

A data frame with 11761 rows and 5 variables:

obs_time The time of the logger recording, in DD/MM/YY HH:MM:SS format

temperature The mean hourly logger temperature reading (°C)

reftemp The predicted reference temperature at the logger location as output by the mesoclimate model (°C)

wind The predicted wind speed at the logger locations (ms^{-1})

netrad The predicted net radiation at the logger locations ($MJm^{-2}hr^{-1}$)

microvars

*Climate variables for May 2010.***Description**

A dataset containing hourly coarse-resolution climate variables in May 2010.

Usage

microvars

Format

A data frame with 744 rows and 9 variables:

dni Direct Normal Radiation ($MJm^{-2}hr^{-1}$)

dif Diffuse Radiation ($MJm^{-2}hr^{-1}$)

humidity Specific humidity ($kgkg^{-1}$)

pressure Sea-level pressure (Pa)

cloudcover Fractional cloud cover

year Year (AD)

month Numeric month

day Day of month

hour Hour of day

modis

*A ~500 m resolution dataset of MODIS-derived albedo values***Description**

A dataset ground surface albedo for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700) derived from Moderate Resolution Imaging Spectroradiometer (MODIS) imagery

Usage

modis

Format

A matrix with 18 rows and 36 columns.

Source

<https://lpdaac.usgs.gov/>

netlong100m	<i>A 100 m resolution matrix of net longwave radiation</i>
-------------	--

Description

A dataset containing net longwave radiation values ($MJm^{-2}hr^{-1}$) for 2010-05-24 11:00 GMT across the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), as produced by [longwvetopo\(\)](#)

Usage

```
netlong100m
```

Format

A matrix with 187 rows and 214 columns.

netlong1m	<i>A one metre resolution matrix of net longwave radiation</i>
-----------	--

Description

A dataset containing net longwave radiation values ($MJm^{-2}hr^{-1}$) for 2010-05-24 11:00 across the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), as produced by [longwaveveg\(\)](#)

Usage

```
netlong1m
```

Format

A matrix with 1000 rows and 1000 columns.

netshort100m	<i>A 100 metre resolution matrix of net shortwave radiation</i>
--------------	---

Description

A dataset containing net shortwave radiation values ($MJm^{-2}hr^{-1}$) for 2010-05-24 11:00 GMT across the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), as produced by [shortwavetopo\(\)](#)

Usage

```
netshort100m
```

Format

A matrix with 187 rows and 214 columns.

netshort1m

A one metre resolution matrix of net shortwave radiation

Description

A dataset containing net shortwave radiation values ($MJm^{-2}hr^{-1}$) for 2010-05-24 11:00 GMT across the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), as produced by [shortwaveveg\(\)](#)

Usage

```
netshort1m
```

Format

A matrix with 1000 rows and 1000 columns.

pcad

Calculates cold air drainage potential

Description

pcad calculates the expected temperature differences resulting from cold air drainage.

Usage

```
pcad(dem, basins, fa, tc, h, p = 101300, out = "cadp")
```

Arguments

dem	a raster object of elevation (m).
basins	a raster object, two-dimensional array or matrix with basins numbered as integers as returned by basindelin() .
fa	a raster object of accumulated flow, as returned by flowacc()
tc	a single numeric value, raster object, two-dimensional array or matrix of values with the dimensions as dem of temperature (°C).
h	a single numeric value, raster object, two-dimensional array or matrix of values with the dimensions as dem of specific humidity ($kgkg^{-1}$).
p	an optional single numeric value, raster object, two-dimensional array or matrix of values with the dimensions as dem of atmospheric pressure (Pa).
out	specifies the type of output to provide (see details). Possible values are "cadp", "tempdif" and "pflow".

Details

To derive expected temperature differences, `pcad` calculates the difference in elevation between each point and the highest in the basin and multiplies this by the lapse rate. Accumulated flow is calculated using `flowacc()`, logarithmically transformed and expressed as proportion of the maximum in each basin as an indication of each locations potential to experience cold air drainage. Cold air flow is possible over shallow boundaries, basins is best derived using `basinmerge()`. Warning: function is quite slow on large datasets.

Value

a raster object:

if `out = "tempdif"` the expected temperature difference (°C).

if `out = "pflow"` the accumulated flow logarithmically transformed expressed as a proportion of the maximum in each basin.

if `out = "cadp"` **(the default)** the expected temperature difference x the proportion of accumulated flow.

Examples

```
basins <- basinmerge(dtm100m, basins100m, 2)
h <- humidityconvert(50, intype = "relative", 20)$specific
fa <- flowacc(dtm100m, basins)
cp1 <- pcad(dtm100m, basins, fa, 20, h)
cp2 <- pcad(dtm100m, basins, fa, 20, h, out = "tempdif")
cp3 <- pcad(dtm100m, basins, fa, 20, h, out = "pflow")
par(mfrow=c(1, 3))
plot(cp3, main = "Accumulated flow proportion")
plot(cp2, main = "Expected temperature difference")
plot(cp1, main = "Cold air drainage potential")
```

pres	<i>A 1 km resolution dataset of interpolated daily sea-level pressure values</i>
------	--

Description

A dataset containing daily sea-level pressure values (Pa) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

```
pres
```

Format

An array with 19 rows, 22 columns and 365 daily values

Source

<https://eip.ceh.ac.uk/chess/>

runmicro	<i>Runs micro- or mesoclimate model</i>
----------	---

Description

microrun produces a high-resolution dataset of downscaled temperatures for one time interval

Usage

```
runmicro(params, netrad, wind)
```

Arguments

params	a data.frame of parameter estimates as produced by <code>fitmicro()</code>
netrad	a raster object, two-dimensional array or matrix of downscaled net radiation as produced by <code>shortwaveveg()</code> - <code>longwaveveg()</code> or <code>shortwavetopo()</code> - <code>longwavetopo()</code> .
wind	a raster object, two-dimensional array or matrix of downscaled wind speed, as produced by reference wind speed x the output of <code>windcoef()</code> .

Details

If netrad is a raster object, a raster object is returned. If modelling mesoclimate, it is assumed that altitudinal, coastal and cold-air drainage effects have already been accounted for in the calculation of reference temperature (see example). It is assumed that the most important energy fluxes determining near-surface temperature are those due to the radiation flux and convection that occurs at the surface-atmosphere boundary. Heat fluxes into the soil and latent heat exchange are considered to be small and proportional to the net radiation flux, and the heat capacity of the vegetation is considered to be small so that, compared to the time-scale of the model, surface temperature rapidly reach equilibrium. In consequence, the difference between the near-ground temperature and the ambient temperature is a linear function of netrad. The gradient of this linear relationship is a measure of the thermal coupling of the surface to the atmosphere. If this relationship is applied to vegetation, assuming the canopy to act like a surface, while air density and the specific heat of air at constant pressure are constant, the slope varies as a function of a wind speed factor, such that different slope values are assumed under high and low wind conditions.

Value

a raster object, two-dimensional array or matrix of temperature anomalies from reference temperature, normally in °C, but units depend on those used in `fitmicro()`.

Examples

```
# =====
# Run microclimate model for 2010-05-24 11:00 (one of the warmest hours)
# =====
params <- fitmicro(microfitdata)
netrad <- netshort1m - netlong1m
tempanom <- runmicro(params, netrad, wind1m)
reftemp <- raster(temp100[, , 564])
extent(reftemp) <- extent(dtm1m)
reftemp <- resample(reftemp, dtm1m)
temps <- tempanom + getValues(reftemp, format = "matrix")
```

```

plot(if_raster(temps, dtm1m), main =
      expression(paste("Temperature ", (~degree~C))))

# =====
# Run mesoclimate model for 2010-05-01 11:00 from first principles
# =====

# -----
# Resample raster function
# -----
resampleraster <- function(a, ro) {
  r <- raster(a)
  extent(r) <- c(-5.40, -5.00, 49.90, 50.15)
  crs(r) <- "+init=epsg:4326"
  r <- projectRaster(r, crs = "+init=epsg:27700")
  r <- resample(r, ro)
  as.matrix(r)
}

# -----
# Resample raster: 24 hours
# -----
get24 <- function(a) {
  ao <- array(NA, dim = c(dim(dtm1km)[1:2], 24))
  for (i in 1:24) {
    ai <- a[, , 2880 + i]
    ao[, , i] <- resampleraster(ai, dtm1km)
  }
  ao
}

# -----
# Derive hourly temperatures
# -----
tmax <- tas[, , 121] + dtr[, , 121] / 2
tmin <- tas[, , 121] - dtr[, , 121] / 2
tme <- as.POSIXct(c(0:364) * 24 * 3600, origin="2010-01-01", tz = "GMT")
out <- as.POSIXct(c(0:23) * 3600, origin="2010-05-01", tz = "GMT")
h <- arrayspline(huss, tme, out = out)
p <- arrayspline(pres, tme, out = out)
n <- get24(cfc)
dni <- get24(dnirad)
dif <- get24(difrad)
jd <- julday(2010, 5, 1)
tc <- h[, , 1] * NA
lr <- h[, , 1] * NA # Also calculates lapse rate
for (i in 1:19) {
  for (j in 1:22) {
    if (is.na(tmax[i, j]) == F)
    {
      ht <- hourlytemp(jd, h[i, j, ], n[i, j, ], p[i, j, ], dni[i, j, ],
                        dif[i, j, ], tmin[i, j], tmax[i, j], 50.02, -5.20)
      tc[i, j] <- ht[12]
      lr[i, j] <- lapserate(tc[i, j], h[i, j, 12], p[i, j, 12])
    }
  }
}

```



```

# -----
# Calculate coastal effects
# -----
sst <- 10.771
dT <- if_raster(sst - tc, dtm1km)
lsw <- if_raster(landsearatio[,28], dtm100m) # upwind
lsa <- if_raster(apply(landsearatio, c(1, 2), mean), dtm100m) # mean, all directions
dTf <- coastalTps(dT, lsw, lsa)

# -----
# Calculate altitudinal effects
# -----
lrr <- if_raster(lr, dtm1km)
lrr <- resample(lrr, dtm100m)
tc <- sst - dTf + lrr * dtm100m

# -----
# Downscale radiation
# -----
dni <- resampleraster(dnirad[,2891], dtm100m)
dif <- resampleraster(difrad[,2891], dtm100m)
n <- resampleraster(cfc[,2891], dtm100m)
h <- resample(if_raster(h[,12], dtm1km), dtm100m)
p <- resample(if_raster(p[,12], dtm1km), dtm100m)
sv <- skyviewtopo(dtm100m)
netshort <- shortwavetopo(dni, dif, jd, 11, dtm = dtm100m, svf = sv)
netlong <- longwavetopo(h, tc, p, n, sv)
netrad <- netshort - netlong

# -----
# Downscale wind
# -----
ws <- array(windheight(wind2010$wind10m, 10, 1), dim = c(1, 1, 8760))
wh <- arrayspline(ws, as.POSIXct(wind2010$obs_time), 6, "2010-05-01 11:00")
ws <- windcoef(dtm100m, 270, res = 100) * wh

# -----
# Fit and run model
# -----
params <- fitmicro(mesofitdata)
anom <- runmicro(params, netrad, ws)
tc <- tc + anom
plot(mask(tc, dtm100m), main =
      expression(paste("Mesoclimate temperature ", (~degree~C))))

```

shortwavetopo

Downscales shortwave radiation accounting for topographic effects

Description

shortwavetopo is used to downscale components of the flux density of shortwave radiation received at the surface of the Earth using a high-resolution digital elevation dataset, ignoring canopy effects.

Usage

```
shortwavetopo(dni, dif, julian, localtime, lat = NA, long = NA,
  dtm = array(0, dim = c(1, 1)), slope = NA, aspect = NA, svf = 1,
  alb = 0.23, albr = 0.23, ha = 0, res = 100, merid = 0, dst = 0,
  shadow = TRUE, component = "sw")
```

Arguments

dni	a single numeric value, raster object, two-dimensional array or matrix of coarse-resolution direct radiation perpendicular to the solar beam ($MJm^{-2}hr^{-1}$).
dif	a single numeric value, raster object, two-dimensional array or matrix of diffuse radiation horizontal of the surface ($MJm^{-2}hr^{-1}$).
julian	a single integer representing the Julian as returned by julday() .
localtime	a single numeric value representing local time (decimal hour, 24 hour clock).
lat	a single numeric value representing the mean latitude of the location for which downscaled radiation is required (decimal degrees, -ve south of equator).
long	a single numeric value representing the mean longitude of the location for which downscaled radiation is required (decimal degrees, -ve west of Greenwich meridian).
dtm	an optional raster object, two-dimensional array or matrix of elevations (m), orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
slope	a single value, raster object, two-dimensional array or matrix of slopes (°). If an array or matrix, then orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
aspect	a single value, raster object, two-dimensional array or matrix of aspects (°). If an array or matrix, then orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
svf	an optional single value, raster object, two-dimensional array or matrix of values representing the proportion of isotropic radiation received by a partially obscured surface relative to the full hemisphere as returned by skyviewtopo() .
alb	an optional single value, raster object, two-dimensional array or matrix of surface albedo(s) (range 0 - 1) derived using albedo() or albedo_adjust() .
albr	an optional single value, raster object, two-dimensional array or matrix of values of albedo(s) of adjacent surfaces (range 0 - 1) as returned by albedo_reflected() .
ha	an optional raster object, two-dimensional array or matrix of values representing the mean slope to the horizon (decimal degrees) of surrounding surfaces from which radiation is reflected for each cell of dtm as returned by mean_slope() .
res	a single numeric value representing the spatial resolution of dtm (m).
merid	an optional single numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours in 24 hour clock) (+1 for BST, +0 for GMT).
shadow	an optional logical value indicating whether topographic shading should be considered (False = No, True = Yes).
component	an optional character string of the component of radiation to be returned. One of "sw" (net shortwave radiation, i.e. accounting for albedo), "sw2" (total incoming shortwave radiation), "dir" (direct), "dif" (diffuse), "iso" (isotropic diffuse), "ani" (anisotropic diffuse), "ref" (reflected).

Details

If slope is unspecified, and dtm is a raster, slope and aspect are calculated from the raster. If slope is unspecified, and dtm is not a raster, the slope and aspect are set to zero. If lat is unspecified, and dtm is a raster with a coordinate reference system defined, lat and long are calculated from the raster. If lat is unspecified, and dtm is not a raster, or a raster without a coordinate reference system defined, an error is returned. If dtm is specified, then the projection system used must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If dtm is a raster object, a raster object is returned. If dtm is a raster object, a raster object is returned. If dni or dif are raster objects, two-dimensional arrays or matrices, then it is assumed that they have been derived from coarse-resolution data by interpolation, and have the same extent as dtm. If no value for ha is provided, the mean slope to the horizon is assumed to be 0. If no value for svv is provided, then the entire hemisphere is assumed to be in view. If no value for svf is provided, then the entire hemisphere is assumed to be in view. If values of alb and albr are not specified, then a default value of 0.23, typical of well-watered grass is assumed. If single values of alb and albr are given, then the entire area is assumed to have the same albedo. If dtm is specified, then the projection system used must be such that the units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If no value for dtm is provided, radiation is downscaled by deriving values on the inclined surfaces specified in slope and aspect and topographic shadowing is ignored. If single values are provided for slope and aspect single values of components of shortwave radiation for an inclined surface are returned. Only single values of lat and long are taken as inputs. If dtm covers a large extent, the dtm is best divided into blocks and separate calculations performed on each block. Since horizon angles, topographic shading and sky view correction factors may be influenced by locations beyond the extent of dtm, it is best to ensure dtm covers a larger extent than that for which radiation values are needed, and to ensure sub-divided blocks overlap in extent. Calculations are faster if values for all inputs are provided.

Value

If component is "sw", a raster object or two-dimensional array of numeric values representing net shortwave radiation ($\text{MJ m}^{-2} \text{hr}^{-1}$).

If component is "sw2", a raster object or two-dimensional array of numeric values representing total incoming shortwave radiation ($\text{MJ m}^{-2} \text{hr}^{-1}$).

If component is "dir", a raster object or two-dimensional array of numeric values representing direct shortwave radiation ($\text{MJ m}^{-2} \text{hr}^{-1}$).

If component is "dif", a raster object or two-dimensional array of numeric values representing diffuse shortwave radiation ($\text{MJ m}^{-2} \text{hr}^{-1}$).

If component is "iso", a raster object or two-dimensional array of numeric values representing isotropic diffuse shortwave radiation ($\text{MJ m}^{-2} \text{hr}^{-1}$).

If component is unspecified, then the default "sw" is returned.

See Also

Function [shortwaveveg\(\)](#) returns net shortwave radiation below a canopy.

Examples

```
# =====
# Extract data for 2010-05-24 11:00
# =====
dni <-dnirad[, ,3444]
```

```

dif <- difrad[, , 3444]
# =====
# Resample to 100m resolution
# =====
dnir <- raster(dni, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
difr <- raster(dif, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
crs(dnir) <- '+init=epsg:4326'
crs(difr) <- '+init=epsg:4326'
dnir <- projectRaster(dnir, crs = "+init=epsg:27700")
difr <- projectRaster(difr, crs = "+init=epsg:27700")
dni <- resample(dnir, dtm100m)
dif <- resample(difr, dtm100m)
sv <- skyviewtopo(dtm100m)
jd <- julday(2010, 5, 24)
ha <- mean_slope(dtm100m)
# =====
# Calculate and plot net shortwave radiation for 2010-05-24 11:00
# =====
netshort100m <- shortwavetopo(dni, dif, jd, 11, dtm = dtm100m,
                             svf = sv, ha = ha)
plot(mask(netshort100m, dtm100m),
     main = "Net shortwave radiation")

```

shortwaveveg

Downscales net shortwave radiation accounting for topography and vegetation

Description

shortwaveveg is used to downscale the flux density of shortwave radiation received at the surface of the Earth, accounting for both topographic and canopy effects.

Usage

```

shortwaveveg(dni, dif, julian, localtime, lat = NA, long = NA,
             dtm = array(0, dim = c(1, 1)), slope = NA, aspect = NA, svv = 1,
             albg = 0.23, fr, albr = 0.23, ha = 0, res = 1, merid = 0, dst = 0,
             shadow = TRUE, x, 1)

```

Arguments

dni	a single numeric value, raster object, two-dimensional array or matrix of coarse-resolution direct radiation perpendicular to the solar beam ($MJm^{-2}hr^{-1}$).
dif	a single numeric value, raster object, two-dimensional array or matrix of coarse-resolution diffuse radiation horizontal at the surface ($MJm^{-2}hr^{-1}$).
julian	a single integer representing the Julian as returned by <code>julday()</code> .
localtime	a single numeric value representing local time (decimal hour, 24 hour clock).
lat	an optional single numeric value representing the mean latitude of the location for which downscaled radiation is required (decimal degrees, -ve south of equator).

long	an optional single numeric value representing the mean longitude of the location for which downscaled radiation is required (decimal degrees, -ve west of Greenwich meridian).
dtm	an optional raster object, two-dimensional array or matrix of elevations (m), orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
slope	an optional single value, raster object, two-dimensional array or matrix of slopes (°). If an array or matrix, then orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
aspect	an optional single value, raster object, two-dimensional array or matrix of aspects (°). If an array or matrix, then orientated as if derived using is_raster() . I.e. [1, 1] is the NW corner.
svv	an optional raster object, two-dimensional array or matrix of values representing the proportion of isotropic radiation received by a surface partially obscured by topography relative to the full hemisphere underneath vegetation as returned by skyviewveg() .
albg	an optional single value, raster object, two-dimensional array or matrix of values representing the albedo(s) of the ground as returned by albedo2() .
fr	a raster object, two-dimensional array or matrix of fractional canopy cover as returned by canopy() .
albr	albr an optional single value, raster object, two-dimensional array or matrix of values representing the albedo(s) of adjacent surfaces as returned by albedo_reflected() .
ha	an optional raster object, two-dimensional array or matrix of values representing the mean slope to the horizon (decimal degrees) of surrounding surfaces from which radiation is reflected for each cell of dtm as returned by mean_slope() .
res	a single numeric value representing the spatial resolution of dtm (m).
merid	an optional single numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours in 24 hour clock) (+1 for BST, +0 for GMT).
shadow	an optional logical value indicating whether topographic shading should be considered (False = No, True = Yes).
x	a raster object, two-dimensional array or matrix of numeric values representing the ratio of vertical to horizontal projections of leaf foliage as returned by leaf_geometry() .
l	a raster object, two-dimensional array or matrix of leaf area index values as returned by lai() .

Details

If slope is unspecified, and dtm is a raster, slope and aspect are calculated from the raster. If slope is unspecified, and dtm is not a raster, the slope and aspect are set to zero. If lat is unspecified, and dtm is a raster with a coordinate reference system defined, lat and long are calculated from the raster. If lat is unspecified, and dtm is not a raster, or a raster without a coordinate reference system defined, an error is returned. If dtm is specified, then the projection system used must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If dtm is a raster object, a raster object is returned. If dtm is a raster object, a raster object is returned. If dni or dif are raster objects, two-dimensional arrays or matrices, then it is assumed that they have been derived from coarse-resolution data by interpolation, and have the same extent as dtm. If no value for ha is provided,

the mean slope to the horizon is assumed to be 0. If no value for `svv` is provided, then the entire hemisphere is assumed to be in view. If values of `albg` and `albr` are not specified, then a default value of 0.23, typical of well-watered grass is assumed. If single values of `albg` and `albr` are given, then the entire area is assumed to have the same albedo. If `dtm` is specified, then the projection system used must be such that the units of `x`, `y` and `z` are identical. Use `projectRaster()` to convert the projection to a Universal Transverse Mercator type projection system. If no value for `dtm` is provided, radiation is downscaled by deriving values on the inclined surfaces specified in `slope` and `aspect` and topographic shadowing is ignored. If single values are provided for `slope` and `aspect`, the entire extent covered by `fr` is assumed to have the same `slope` and `aspect`. Only single values of `lat` and `long` are taken as inputs. If `dtm` covers a large extent, the `dtm` is best divided into blocks and separate calculations performed on each block. Since horizon angles, topographic shading and sky view correction factors may be influenced by locations beyond the extent of `dtm`, it is best to ensure `dtm` covers a larger extent than that for which radiation values are needed, and to ensure sub-divided blocks overlap in extent. Calculations are faster if values for all inputs are provided.

Value

a raster object, two-dimensional array of numeric values representing net shortwave radiation (MJ per metre squared per hour). The raster package function `terrain()` can be used to derive slopes and aspects from `dtm` (see example).

See Also

Function `shortwavetopo()` returns net shortwave radiation, or components thereof, above the canopy.

Examples

```
# =====
# Extract data for 2010-05-24 11:00
# =====
dni <- microvars$dni[564]
dif <- microvars$dif[564]
# =====
# Calculate input parameters
# =====
x <- leaf_geometry(veg_hgt)
l <- lai(aerial_image[,3], aerial_image[,4])
l <- lai_adjust(l, veg_hgt)
fr <- canopy(l, x)
alb <- albedo(aerial_image[,1], aerial_image[,2], aerial_image[,3],
              aerial_image[,4])
albg <- albedo2(alb, fr)
sv <- skyviewveg(dtm1m, l, x)
jd <- julday(2010, 5, 24)
ha <- mean_slope(dtm1m)
# =====
# Calculate and plot net shortwave radiation for 2010-05-24 11:00
# =====
netshort1m <- shortwaveveg(dni, dif, jd, 11, dtm = dtm1m, svv = sv, albg = albg,
                           fr = fr, ha = ha, x = x, l = l)
plot(mask(netshort1m, dtm1m), main = "Net shortwave radiation")
```

siflat	<i>Calculates the solar index for a flat surface</i>
--------	--

Description

Calculates the solar index for a flat surface

Usage

```
siflat(localtime, lat, long, julian, merid = 0, dst = 0)
```

Arguments

localtime	local time (decimal hour, 24 hour clock).
lat	latitude of the location for which the solar altitude is required (decimal degrees, -ve south of the equator).
long	longitude of the location for which the solar altitude is required (decimal degrees, -ve west of Greenwich meridian).
julian	Julian day expressed as an integer as returned by <code>julday()</code> .
merid	an optional numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Value

a numeric value representing the solar altitude (°).

Examples

```
# solar index at noon on 21 June 2010, Porthleven, Cornwall
jd <- julday(2010, 6, 21) # Julian day
siflat(12, 50.08, -5.31, jd)
```

skyviewtopo	<i>calculates the proportion of sky in view</i>
-------------	---

Description

skyviewtopo is used to calculate a coefficient to correct for the proportion of sky in view when calculating net shortwave or longwave radiation above the canopy.

Usage

```
skyviewtopo(dtm, steps = 36, res = 100)
```

Arguments

dtm	dtm a raster object, two-dimensional array or matrix of elevations (m).
steps	an optional integer. The sky view is calculated from the horizon angle in specified directions. Steps defines the total number of directions used. If the default 36 is specified, then the horizon angle is calculated at 10° intervals.
res	a single numeric value representing the spatial resolution of dtm (m).

Details

If a proportion of the sky of partially obscured, then the isotropic radiation flux received by a surface can be determined by integrating the single direction radiation flux over the proportion of sky in view. This function returns the integrated flux over the proportion of sky in view expressed as a proportion of the integrated flux over the entire hemisphere.

If dtm is a raster object, a raster object is returned. The projection system associated with dtm must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system.

Value

a raster object or two-dimension array of values representing the proportion of isotropic radiation received by a partially obscured surface relative to the full hemisphere.

See Also

The function [skyviewveg\(\)](#) calculates a sky view correction factor underneath vegetation.

Examples

```
sv <- skyviewtopo(dtm100m)
plot(sv, main = "Sky view factor")
```

skyviewveg

Calculates a sky view correction factor underneath vegetation

Description

skyviewveg is used to calculate a coefficient to correct for the proportion of sky obscured by topography when calculating net shortwave or longwave radiation above the canopy.

Usage

```
skyviewveg(dtm, l, x, steps = 36, res = 1)
```

Arguments

dtm	a raster object, two-dimensional array or matrix of elevations (m).
l	a raster object, two-dimensional array or matrix of leaf area index values as returned by lai() .
x	a raster object, two-dimensional array of numeric values representing the ratio of vertical to horizontal projections of leaf foliage as returned by leaf_geometry() .

steps	an optional integer. The sky view is calculated from the horizon angle in specified directions. Steps defines the total number of directions used. If the default 36 is specified, then the horizon angle is calculated at 10° intervals.
res	a single numeric value representing the spatial resolution of dtm (m).

Details

If dtm is a raster object, a raster object is returned. The projection system associated with dtm must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If a proportion of the sky is partially obscured, then the isotropic radiation flux received by a surface underneath canopy can be determined by integrating the single direction radiation transmission over the proportion of sky in view. This function returns a computationally efficient approximation of the integrated transmission over the proportion of sky in view expressed as a proportion of the integrated transmission over the entire hemisphere.

Value

a raster object or a two-dimensional array of numeric values representing the proportion of isotropic radiation received by a surface partially obscured by topography relative to the full hemisphere underneath vegetation.

See Also

The function [skyviewtopo\(\)](#) calculates a sky view correction factor above vegetation.

Examples

```
l <- lai(aerial_image[, , 3], aerial_image[, , 4])
x <- leaf_geometry(veg_hgt)
sv <- skyviewveg(dtm1m, l, x)
plot(sv, main = "Sky view factor")
```

solalt

Calculates the solar altitude

Description

solalt is used to calculate the solar altitude at any given location from the local time.

Usage

```
solalt(localtime, lat, long, julian, merid = 0, dst = 0)
```

Arguments

localtime	local time (decimal hour, 24 hour clock).
lat	latitude of the location for which the solar altitude is required (decimal degrees, -ve south of the equator).
long	longitude of the location for which the solar altitude is required (decimal degrees, -ve west of Greenwich meridian).

julian	Julian day expressed as an integer as returned by <code>julday()</code> .
merid	an optional numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Value

a numeric value representing the solar altitude (°).

Examples

```
# solar altitude at noon on 21 June 2010, Porthleven, Cornwall
jd <- julday (2010, 6, 21) # Julian day
solalt(12, 50.08, -5.31, jd)
```

solarindex	<i>Calculates the solar index</i>
------------	-----------------------------------

Description

`solarindex` is used to calculate the proportion of direct beam radiation incident on an inclined surface at a specified time and location.

Usage

```
solarindex(slope = NA, aspect, localtime, lat = NA, long, julian,
  dtm = array(0, dim = c(1, 1)), res = 1, merid = 0, dst = 0,
  shadow = TRUE)
```

Arguments

slope	a single value, raster object, two-dimensional array or matrix of slopes (°). If an array or matrix, then orientated as if derived using <code>is_raster()</code> . I.e. [1, 1] is the NW corner.
aspect	a single value, raster object, two-dimensional array or matrix of aspects (°). If an array or matrix, then orientated as if derived using <code>is_raster()</code> . I.e. [1, 1] is the NW corner.
localtime	a single numeric value representing local time (decimal hour, 24 hour clock).
lat	a single numeric value representing the mean latitude of the location for which the solar index is required (decimal degrees, -ve south of the equator).
long	a single numeric value representing the mean longitude of the location for which the solar index is required (decimal degrees, -ve west of Greenwich meridian).
julian	a single integer representing the Julian day as returned by <code>julday()</code> .
dtm	an optional raster object, two-dimensional array or matrix of elevations (m). If not a raster, orientated as if derived from a raster using <code>is_raster()</code> . I.e. [1, 1] is the NW corner.
res	a single numeric value representing the spatial resolution of dtm (m).

merid	an optional single numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional single numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).
shadow	an optional logical value indicating whether topographic shading should be considered (TRUE = Yes, FALSE = No).

Details

If slope is unspecified, and dtm is a raster, slope and aspect are calculated from the raster. If slope is unspecified, and dtm is not a raster, the slope and aspect are set to zero. If lat is unspecified, and dtm is a raster with a coordinate reference system defined, lat and long are calculated from the raster. If lat is unspecified, and dtm is not a raster, or a raster without a coordinate reference system defined, an error is returned. If dtm is specified, then the projection system used must be such that units of x, y and z are identical. Use [projectRaster\(\)](#) to convert the projection to a Universal Transverse Mercator type projection system. If dtm is a raster object, a raster object is returned.

Value

If shadow is TRUE, a raster object or a two-dimensional array of numeric values representing the proportion of direct beam radiation incident on an inclined surface, accounting for topographic shading.

If shadow is FALSE, a raster object or a two-dimensional array of numeric values representing the proportion of direct beam radiation incident on an inclined surface, not accounting for topographic shading.

If no dtm is provided, a vector, array or single numeric value of the proportion of direct beam radiation incident on the inclined surfaces specified by slope and aspect, and topographic shading is ignored.

See Also

the raster package function [terrain\(\)](#) can be used to derive slopes and aspects from dtm (see example).

Examples

```
jd <- julday(2010, 6, 21) # Julian day
# slope, aspect, lat & long calculated from raster
si1 <- solarindex(localtime = 8, julian = jd, dtm = dtm1m)
si2 <- solarindex(localtime = 8, julian = jd, dtm = dtm1m, shadow = FALSE)
par(mfrow = c(2, 1))
plot(si1, main = "Solar index with topographic shadowing")
plot(si2, main = "Solar index without topographic shadowing")
ll <- latlongfromraster(dtm1m)
solarindex(0, 0, 8, lat = ll$lat, long = ll$long, jd)
```

solartime	<i>Calculates the solar time</i>
-----------	----------------------------------

Description

solartime is used to calculate the solar time. I.e. the time that would be measured by a sundial.

Usage

```
solartime(localtime, long, julian, merid = 0, dst = 0)
```

Arguments

localtime	local time (decimal hour, 24 hour clock).
long	longitude of the location for which the solar time is required (decimal degrees, -ve west of Greenwich meridian).
julian	Julian day expressed as an integer as returned by julday() .
merid	an optional numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Details

‘solartime’ accounts for two factors: firstly, east or west component of the analemma, namely the angular offset of the Sun from its mean position on the celestial sphere as viewed from Earth due the eccentricity of the Earth’s orbit and the obliquity due to tilt of the Earth’s rotational axis. These two factors have different wavelengths, amplitudes and phases, that vary over geological timescales. The equations used here are those derived by Milne.

Value

the solar time. I.e. the times that would be measured by a sundial (hours).

Examples

```
jd <- julday (2010, 6, 21) # Julian day
solartime(12, -5, jd) # solartime at noon on 21 June 2010, 5°W
```

solazi	<i>Calculates the solar azimuth</i>
--------	-------------------------------------

Description

solazi is used to calculate the solar azimuth at any given location from the local time.

Usage

```
solazi(localtime, lat, long, julian, merid = 0, dst = 0)
```

Arguments

localtime	local time (decimal hour, 24 hour clock).
lat	latitude of the location for which the solar azimuth is required (decimal degrees, -ve south of the equator).
long	longitude of the location for which the solar azimuth is required (decimal degrees, -ve west of Greenwich meridian).
julian	Julian day expressed as an integer as returned by <code>julday()</code> .
merid	an optional numeric value representing the longitude (decimal degrees) of the local time zone meridian (0 for UK time).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Value

a numeric value representing the solar azimuth (decimal degrees).

Examples

```
# solar azimuth at noon on 21 June 2010, Porthleven, Cornwall, UK
jd <- julday (2010, 6, 21) # Julian day
solazi(12, 50.08, -5.31, jd)
```

suntimes	<i>Calculates sunrise, sunset and daylength</i>
----------	---

Description

suntimes is used to calculate, sunrise, sunset and daylength on any given data at a specified location.

Usage

```
suntimes(julian, lat, long, tz = 0, dst = 0)
```

Arguments

julian	the Julian day as returned by <code>julday()</code> .
lat	latitude of the location for which suntime is required (decimal degrees, -ve south of equator).
long	longitude of the location for which suntime is required (decimal degrees, -ve west of Greenwich meridian).
tz	an optional numeric value specifying the time zones expressed as hours different from GMT (-ve to west).
dst	an optional numeric value representing the local summer time adjustment (hours, e.g. +1 for BST).

Details

if the sun is above or below the horizon for 24 hours on any days, a warning is given, and the sunrise and sunset returned are the approximate times at which that the sun is closest to the horizon.

Value

a data.frame with three components:
 sunrise a vector of sunrise (hours in 24 hour clock).
 sunrise a vector of sunset (hours in 24 hour clock).
 sunrise a vector of daylengths (hours).

Examples

```
jd <- julday(2018, 1, 16)
suntimes(jd, 50.17, -5.12) # Cornwall, UK
suntimes(jd, 78.22, 15.64, 1) # Longyearbyen, Svalbad
suntimes(-54.94, -67.61, -3) # Puerto Williams, Cabo de Hornos, Chile
```

tas	<i>A 1km resolution dataset of daily sea-level temperature</i>
-----	--

Description

A dataset containing daily sea-level temperature (°C) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700).

Usage

```
tas
```

Format

An array with 19 rows, 22 columns and 365 daily values

Source

<https://eip.ceh.ac.uk/chess/>

temp100	<i>A 100 m resolution array of hourly reference temperatures.</i>
---------	---

Description

A 100 m resolution three-dimensional array of hourly reference temperatures (°C) for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), for May 2010 as output by `runmicro()`.

Usage

```
temp100
```

Format

An array with 10 rows, 10 columns and 744 hourly values.

veg_hgt	<i>A 1 m resolution raster object of vegetation height.</i>
---------	---

Description

A dataset containing the vegetation height (m) for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700). Data were derived from a digital terrain and digital surface model.

Usage

veg_hgt

Format

A matrix with 1000 rows and 1000 columns.

Source

<http://www.tellusgb.ac.uk/>

wind1m	<i>A one metre resolution matrix of windspeed</i>
--------	---

Description

A dataset containing wind speed values (ms^{-1}) for 2010-05-24 11:00 GMT across the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordnance Survey GB Grid Reference system (CRS: +init=epsg:27700), as produced with `windcoef()`

Usage

wind1m

Format

A matrix with 1000 rows and 1000 columns.

wind2010	<i>Six-hourly wind speed at 10 m for 2010.</i>
----------	--

Description

A dataset containing six-hourly wind speed in 2010 estimated 10 m above the ground.

Usage

```
wind2010
```

Format

A data frame with 1460 rows and 2 variables:

obs_time Time at date in yyyy-mm-dd HH:MM format

wind10m Wind speed 10m above the ground (ms^{-1})

Source

<http://www.ncep.noaa.gov/>

windcoef	<i>Calculates wind shelter coefficient</i>
----------	--

Description

windcoef is used to apply a topographic shelter coefficient to wind data.

Usage

```
windcoef(dsm, direction, hgt = 1, res = 1)
```

Arguments

dsm	raster object, two-dimensional array or matrix of elevations (m) derived either from a digital terrain or digital surface model, and orientated as if derived using <code>is_raster()</code> . I.e. [1, 1] is the NW corner.
direction	a single numeric value specifying the direction from which the wind is blowing (°).
hgt	a single numeric value specifying the height (m) at which wind speed is derived or measured. The wind speeds returned are also for this height, and account for the fact topography affords less shelter to wind at greater heights.
res	a single numeric value specifying the the resolution (m) of dsm.

Details

If dsm is a raster object, then a raster object is returned. If elevations are derived from a digital terrain model, then the sheltering effects of vegetation are ignored. If derived from a digital surface model, then the sheltering effects of vegetation are accounted for. If res is unspecified dtm is assumed to have a resolution of one m.

Value

a raster object, or two-dimensional array of shelter coefficients. E.g. a shelter coefficient of 0.5 indicates that wind speed at that location is 0.5 times that measured at an unobscured location.

See Also

The function `windheight()` converts measured wind heights to a standard reference height.

Examples

```
dsm <- dtm1m + veg_hgt
wc <- windcoef(dsm, 0)
plot(mask(wc, dtm1m), main = "Northerly wind shelter coefficient")
```

windheight

Applies height correction to wind speed measurements

Description

windheight is used to to apply a height correction to wind speed measured at a specified height above ground level to obtain estimates of wind speed at a desired height above the ground.

Usage

```
windheight(ui, zi, zo)
```

Arguments

ui	numeric value(s) of measured wind speed (ms^{-1}) at height zi (m).
zi	a numeric value idicating the height (m) above the ground at which ui was measured.
zo	a numeric value indicating the height (m) above ground level at which wind speed is desired.

Details

Thus function assumes a logarithmic height profile to convert wind speeds. It performs innacurately when u_0 is lower than 0.2 and a warning is given. If u_0 is below ~ 0.08 then the logairthmic height profile cannot be used, and u_0 is converted to 0.1 and a warning given.

Value

numeric values(s) of wind speed at height specified by zo (ms^{-1}).

See Also

The function `windcoef()` calculates a topographic or vegetation sheltering effect.

Examples

```
windheight(3, 10, 1) # good
windheight(3, 10, 0.15) # performs poorly. Warning given
windheight(3, 10, 0.05) # cannot calculate. ui converted and warning given
```

Index

*Topic **datasets**

- aerial_image, 3
- basins100m, 11
- cfc, 14
- demworld, 16
- difrad, 18
- dnirad, 18
- dtm100m, 19
- dtm1km, 19
- dtm1m, 20
- dtr, 20
- huss, 29
- landsearatio, 34
- mesofitdata, 40
- microfitdata, 42
- microvars, 43
- modis, 43
- netlong100m, 44
- netlong1m, 44
- netshort100m, 44
- netshort1m, 45
- pres, 46
- tas, 62
- temp100, 62
- veg_hgt, 63
- wind1m, 63
- wind2010, 64

- aerial_image, 3
- airmasscoef, 4
- airmasscoef(), 26
- albedo, 4
- albedo(), 6–8, 50
- albedo2, 5
- albedo2(), 38, 53
- albedo_adjust, 7
- albedo_adjust(), 5, 6, 8, 50
- albedo_reflected, 7
- albedo_reflected(), 50, 53
- arrayspline, 8

- basindelin, 9
- basindelin(), 10, 11, 45
- basindelin_big, 10
- basindelin_big(), 9, 10
- basinmerge, 11
- basinmerge(), 46
- basins100m, 11

- cadconditions, 12
- cadconditions(), 41
- canopy, 13
- canopy(), 6, 38, 53
- cfc, 14
- coastalTps, 15
- crs(), 35

- dailyprecipNCEP, 16
- dailyprecipNCEP(), 41
- demworld, 16
- difprop, 17
- difprop(), 26
- difrad, 18
- dnirad, 18
- dtm100m, 19
- dtm1km, 19
- dtm1m, 20
- dtr, 20

- fitmicro, 21
- fitmicro(), 47
- flowacc, 22
- flowacc(), 45, 46

- get_dem, 23
- get_dem(), 42
- get_NCEP, 24
- get_NCEP(), 26
- get_ncep(), 26

- horizonangle, 25
- hourlyNCEP, 26
- hourlyNCEP(), 24, 41, 42
- hourlytemp, 27
- hourlytemp(), 26
- humidityconvert, 28
- humidityconvert(), 37, 38
- huss, 29

if_raster, 29
invs, 30
invs(), 15, 34
is_raster, 31
is_raster(), 25, 50, 53, 58, 64

julday, 31
julday(), 4, 12, 17, 27, 50, 52, 55, 58, 60, 61

lai, 32
lai(), 14, 33, 53, 56
lai_adjust, 33
lai_adjust(), 33
landsearations, 34
lapserate, 34
lapserate(), 41
latlongfromraster, 35
leaf_geometry, 35
leaf_geometry(), 14, 38, 53, 56
longwavetopo, 36
longwavetopo(), 38, 44, 47
longwaveveg, 37
longwaveveg(), 37, 44, 47

mean_slope, 39
mean_slope(), 50, 53
mesofitdata, 40
microclimaforNMR, 40
microfitdata, 42
microvars, 43
modis, 43

netlong100m, 44
netlong1m, 44
netshort100m, 44
netshort1m, 45

path.expand(), 10
pcad, 45
pcad(), 22, 41
pres, 46
projectRaster(), 25, 35, 39, 51, 53, 54, 56, 57, 59

runmicro, 47
runmicro(), 62

shortwavetopo, 49
shortwavetopo(), 44, 47, 54
shortwaveveg, 52
shortwaveveg(), 41, 45, 47, 51
siflat, 55
siflat(), 26
skyviewtopo, 55
skyviewtopo(), 36, 50, 57
skyviewveg, 56
skyviewveg(), 38, 41, 53, 56
solalt, 57
solarindex, 58
solartime, 60
solazi, 60
solazi(), 25
spline(), 9
suntimes, 61

tas, 62
temp100, 62
terrain(), 54, 59

veg_hgt, 63

wind1m, 63
wind2010, 64
windcoef, 64
windcoef(), 41, 47, 63, 65
windheight, 65
windheight(), 26, 65