Package ‘microclima’

August 16, 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

Depends raster (>= 2.5.8),
rgdal (>= 1.2),
stringr (>= 1.2),
sp (>= 1.2),
dplyr(>= 0.7)

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

Suggests knitr,
rmarkdown,
testthat

VignetteBuilder knitr

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

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

R topics documented:

`aerial_image` *A 1 m resolution aerial image.*

Description

A dataset containing image reflectance values for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordance 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 *Calculates the airmass coefficient*

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 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).

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 *Calculates surface albedo*

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 `albedo()` or `albedo_adjust()`.
- `fr` — a raster object, two-dimensional array or matrix of fractional canopy cover as returned by `canopy()`.
- `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 are 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 — *Adjusts albedo to correct for image brightness and contrast*

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 — *Calculates the mean albedo of surfaces surrounding each location*

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 tthe corresponding vector in the output array is also a vector of NAs. It is assumed that all spatial data within a have equivelent 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 onbject 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 dtm. However, in contrast to many such algorithms, all basins are correctly seperated 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 *Merges adjoining basins*

Description

basinmerge merges adjoining basins if the height differences between the bottom of the basin and the pour point is less than 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 seperated 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 *A raster of basins numbered as integers*

Description

A raster object of basins numbered as integers for the area bounded by 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordance 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 — *Calculates whether conditions are right for cold air drainage*

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 (ºC).
`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 `julday()`
`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 0.05º 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 — *Calculates coastal effects using thin-plate spline*

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 (ºC).
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 (ºC).

Examples

```
# =========================================
# Calculate land-sea temperature difference
# =========================================
temp <- tas[,,1]
sst <- 10.665
dT <- if_raster(sst - temp, dtm1km)
# ============================
# Obtain coastal exposure data
# ============================
lsw <- landsearatios[,,7]  # upwind
lsa <- apply(landsearatios, 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))))
```

`difprop` *Calculates the diffuse fraction from incoming shortwave radiation*

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 `julday()`
`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 *A 0.05º resolution dataset of hourly diffuse radiation*

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 *A 0.05º resolution dataset of hourly direct radiation normal to the direction of the solar beam*

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	*A 100 m resolution raster object of elevation for the Lizard Peninsula, Cornwall, UK.*

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 Ordance 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	*A 1 km resolution raster object of elevation for the Lizard Peninsula, Cornwall, UK.*

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 Ordance 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` *A 1 m resolution raster object of elevation for part of the Lizard Peninsula, Cornwall, UK.*

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 Ordance 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` *A 1 km resolution dataset of diurnal temperature ranges*

Description

A spatially interpolated dataset containing diurnal temperature ranges (ºC) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordance 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 anomoly 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 `lm((temperature - reftemp) ~ netrad * windfact)` where windfact is given by `ifelse(wind > 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 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 seperate 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` *Calculates accumulated flow*

Description

`flowacc` is used by `pcad()` to calculate accumulated flow to each cold air drainage basin

Usage

```
flowacc(dem, basins)
```

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()`.

Details

Accumulated flow is expressed in terms of number of cells.

Value

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

Examples

```
# Merge basins seperated by boundary < 2m
basins <- basinmerge(dtm100m, basins100m, 2)
# Calculate accumulated flow: takes a few minutes to run
fa <- flowacc(dtm100m, basins)
# plot data (expressed as area)
plot(log(fa * 100^2), main = "Log (accumulated flow)")
```

horizonangle *Calculates the tangent of the horizon angle*

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()` (º 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")
```

hourlytemp *Derives hourly temperatures from daily data*

Description

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

Usage

```
hourlytemp(julian, 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()`.
`h`	a vector of hourly specific humidities ($kgkg^{-1}$).
`n`	a vector of hourly fractional cloud cover values (range 0 - 1).
`p`	an optional vector of hourly atmospheric pressure values (Pa).
`dni`	a vector of hourly direct radiation values normal to the solar beam (MJm^-2hr^{-1}).
`dif`	a vector of hourly diffuse radiation values (MJm^-2hr^{-1}).
`mintemp`	a vector of daily minimum temperatures (ºC).
`maxtemp`	a bector of daily maximum temperatures (ºC).
`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 (º) (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 (ºC).

Examples

```
jd <- julday(2010, 5 , c(1,2))
ht <- hourlytemp(jd, 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 ($kgkg^{-1}$), absolute humidity (kgm^{-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 (ºC).
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 — *A 1 km resolution dataset of interpolated daily specific humidity values*

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 Ordance 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 — *Flexible conversion to raster object*

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 *Calculates land to sea ratio in upwind direction*

Description

invls is used to calculate an inverse distance^2 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 *Calculates Leaf Area Index*

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 — *Calculates Leaf Area Index for specified height above ground*

Description

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

Usage

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

Arguments

`l`	raster object, two-dimensional array or matrix of leaf area index values as returned by `lai()`.
`veghgt`	a raster object, two-dimensional array or matrix of vegetation heights (m).
`hgt`	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 gorund 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")
```

landsearatios — *Inverse-distance weighted land-sea ratios in 36 directions*

Description

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

Usage

```
landsearatios
```

Format

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

`lapserate` *Calculates the moist adiabatic lapse rate*

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 (ºC).
`h`	a single numeric value, raster object, two-dimensional array or matrix of specific humidity ($kgkg^{-1}$).
`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` *Derives latitude and longitude of centre of raster object*

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` *Calculates leaf orientation*

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 `veghgt` 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` *calculates net longwave radiation above canopy*

Description

`longwavetopo` is used to calculate a high-resolution dataset of the net longwave radiation flux density emmited 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 ($kgkg^{-1}$).
`tc`	a single numeric value, raster object, two-dimensional array or matrix of temperatures (ºC).
`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 pr 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 humidy from other meaures 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 emmited 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 ($kgkg^{-1}$).
`tc`	a single numeric value, raster object, two-dimensional array or matrix of temperatures (ºC).

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 humidy from other meaures 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 *Calculates the mean slope to the horizon*

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 10º 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 (º).

Examples

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

mesofitdata	*2010 Data for fitting mesoclimate model.*

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 (ºC)

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 (ºC)

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

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

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 (ºC)

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

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 Ordance 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` *A 100 m resolution matrix of net longwave radiation*

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 Ordance Survey GB Grid Reference system (CRS: +init=epsg:27700), , as produced by `longwavetopo()`

Usage

```
netlong100m
```

Format

A matrix with 187 rows and 214 columns.

`netlong1m` *A one metre resolution matrix of net longwave radiation*

Description

A dataset containing net longwave radiation values (MJm^-2hr^{-1}) for 2010-05-24 11:00 across the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordance 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` *A 100 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 160000, 181400, 11300, 30000 (xmin, xmax, ymin, ymax) using the Ordance 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 Ordance 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 (ºC).
`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 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 (ºC).

if `out = "pflow"` the accumulated flow logairthmically 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` *A 1 km resolution dataset of interpolated daily sea-level pressure values*

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 Ordance 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 — *Runs micro- or mesoclimate model*

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 fitmicro()
netrad	a raster object, two-dimensional array or matrix of downscaled net radiation as produced by shortwaveveg() - longwaveveg() or shortwavetopo() - longwavetopo().
wind	a raster object, two-dimensional array or matrix of downscaled wind speed, as produced by reference wind speed x the output of windcoef().

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 anomolies from reference temperature, normally in ºC, 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(landsearatios[,,28], dtm100m)  # upwind
lsa <- if_raster(apply(landsearatios, 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 ot 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 (º). 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 (º). 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" (anistopic 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 seperate 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 (MJ m^-2 hr^-1).

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

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

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

If component is "iso", a raster object or two-dimensional array of numeric values representing isotropic diffuse shortwave radiation (MJ m^-2 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, l)
```

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 ot 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	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 (º). 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 (º). 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 seperate 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 paramaters
# =========================
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")
```

skyviewtopo — *calculates the proportion of sky in view*

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 10º 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 10º 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 of 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 `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).

Value

a numeric value representing the solar altitude (º).

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 — *Calculates the solar index*

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 (º). 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 (º). If an array or matrix, then orientated as if derived using `is_raster()`. 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 `julday()`.
`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 `is_raster()`. 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 *Calculates the solar time*

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, 5ºW
```

---

solazi *Calculates the solar azimuth*

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 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).

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 *Calculates sunrise, sunset and daylength*

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 julday().
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 appoximate 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	*A 1km resolution dataset of daily sea-level temperature*

Description

A dataset containing daily sea-level temperature (ºC) in 2010 for the area bounded by 160000, 182000, 11000, 30000 (xmin, xmax, ymin, ymax) using the Ordance 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	*A 100 m resolution array of hourly reference temperatures.*

Description

A 100 m resolution three-dimensional array of hourly reference temperatures (ºC) for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordance 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 — *A 1 m resolution raster object of vegetation height.*

Description

A dataset containing the vegetation height (m) for the area bounded by 169000, 170000, 12000, 13000 (xmin, xmax, ymin, ymax) using the Ordance 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 — *A one metre resolution matrix of windspeed*

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 Ordance 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 — *Six-hourly wind speed at 10 m for 2010.*

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 — *Calculates wind shelter coefficient*

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 `is_raster()`. I.e. `[1, 1]` is the NW corner.
`direction`	a single numeric value specifying the direction from which the wind is blowing (º).
`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 uo is lower than 0.2 and a warning is given. If uo is below ~0.08 then the logairthmic height profile cannot be used, and uo 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