Package ‘Thermimage’

May 16, 2016

Type Package

Title Functions for Handling Thermal Images

Version 2.1

Date 2016-05-07

Author Glenn J. Tattersall

Maintainer Glenn J. Tattersall <gtatters@brocku.ca>

Description A collection of functions and routines for inputting thermal image video files, plotting and converting binary raw data into estimates of temperature. First published 2015-03-26. Written primarily for research purposes in biological applications of thermal images. v1 included the base calculations for converting thermal image binary values to temperatures. v2 included additional equations for providing heat transfer calculations.

SampleScripts http://github.com/gtatters/Thermimage/heatcalc.R

License GPL-2

Depends R (>= 2.10)

Suggests

Imports stats

LazyData TRUE

URL http://github.com/gtatters/Thermimage

BugReports http://github.com/gtatters/Thermimage/issues

RoxygenNote 5.0.1

R topics documented:

`Thermimage-package` *Handles thermal image data input and conversion to temperature using established physical equations.*

Description

Assists in converting raw thermal imaging data files into temperature values.

Details

Package:	Thermimage
Type:	Package
License:	GPL-2

Primary purpose of the package is to assist with manipulating raw data extracted from thermal image files. These raw data are stored in a raw data format and require inforamtion about various environmental variables to estimate surface temperatures accurately. raw2temp is the primary function of use. Other functions included involve simple scripts for data handling.

Author(s)

Glenn J. Tattersall <gtatters@brocku.ca>

References

1. http://130.15.24.88/exiftool/forum/index.php/topic,4898.60.html

2. Minkina, W. and Dudzik, S. 2009. Infrared Thermography: Errors and Uncertainties. Wiley Press, 192 pp.

`airtconductivity` *Thermal conductivity of air.*

Description

Thermal conductivity of air. Units: W/m/K

Usage

```
airtconductivity(Ta = 20)
```

Arguments

`Ta`	Air temperature in degrees Celsius. Default value is 20.

Author(s)

Glenn J Tattersall

References

http://www.engineeringtoolbox.com/air-properties-d_156.html

Examples

```
## The function is currently defined as
function (Ta = 20)
{
    Intercept <- 0.024280952
    Slope <- 7.07143e-05
    k <- Intercept + Slope * Ta
    k
  }
# Example calculation:
Ta<-20
airtconductivity(Ta)
```

`airviscosity` *Returns air viscovisty for a given air temperature.*

Description

Returns the air viscosity value for a given, supplied air temperature (Ta). Ta should be in units of oC.

Usage

```
airviscosity(Ta = 20)
```

Arguments

`Ta`	Air temperature in degrees Celsius. Default value is 20.

Value

Kinematic viscosity of air, as a function of temperature Units: m2/s Regression for 0 to 100oC range: Intercept<-13.17380952 Slope<-0.097457143 k<-(Intercept+Slope*Ta)*1e-6 # multiply by 1e-6 to get into m2/s units

Author(s)

Glenn J Tattersall

References

http://www.engineeringtoolbox.com/air-properties-d_156.html

Examples

```
## The function is currently defined as
function (Ta = 20)
{
    Intercept <- 13.17380952
    Slope <- 0.097457143
    k <- (Intercept + Slope * Ta) * 1e-06
    k
```

```
  }
# Example calculation
Ta<-20
airviscosity(Ta)
```

---

areacone *Provides the surface are of a cone*

Description

Provides the surface area of a cone with an elliptical base. For a circular cone, simply use Radius=radius.

Usage

```
areacone(Radius, radius=Radius, hypotenuse=NULL, height, ends=1)
```

Arguments

`Radius`	The Radius of the major axis of the base of the cone.
`radius`	The radius of the minor axis of the base of the cone.
`hypotenuse`	The hypotenuse of the height of the cone (if blank, determined from radius and height)
`height`	The height of the cone (if hypotenuse is known, leave height blank)
`ends`	To include the base area in surface area calculation, set ends = 1, otherwise set ends = 0.

Details

Calculates the surface are of a cone with an elliptical base.

Author(s)

Glenn J Tattersall

Examples

```
## The function is currently defined as
function(Radius, radius=Radius, hypotenuse=NULL, height, ends=1)
{
  if(is.null(hypotenuse)){
    hypotenuse<-sqrt(height^2+Radius^2)
  }
  Area <- ends*pi*Radius*radius + pi*Radius*hypotenuse
  Area
}

# Example calculation from a measure of a bird bill.

# Typically, a bird bill will be measured by its depth (d) at the base, its width (w) at the
# base and by its overall length. The length (l) is typically measured along the length of
```

```
# the culmen, and thus is a diagonal measure along the hypotenuse of the cone.

d<-12
w<-6
l<-18
areacone(Radius=d/2, radius=w/2, hypotenuse=l, height=NULL, ends=1)


# If the perpendicular cone height (h) is instead measured, rather than the hypotenuse, then
# substitute h for height and assign hypotenuse = NULL, to obtain the same result

h<-sqrt(l^2-(d/2)^2)
areacone(Radius=d/2, radius=w/2, hypotenuse=NULL, height=h, ends=1)


# To only show surface area of the exposed surface, and exclude the oval base of the cone
# set ends=0:

areacone(Radius=d/2, radius=w/2, hypotenuse=l/2, height=NULL, ends=0)
```

`areacylinder` *Provides the surface area of a cylinder.*

Description

Provides the surface area of a cylinder, including the circular bases.

Usage

```
areacylinder(Radius, radius=Radius, height, ends = 2)
```

Arguments

`Radius`	The major radius of the base of the cylinder.
`radius`	The minor radius of the base of the cylinder. Default is to equal the major Radius in the case of a circular base.
`height`	The height of the cylinder (alternatively, the length of a horizontal cylinder)
`ends`	How many ends to include in the surface area calculation (2=both ends, 1=one end, 0=neither end)

Author(s)

Glenn J Tattersall

Examples

```
## The function is currently defined as
function(Radius, radius=Radius, height, ends=2)
{
  Area <- (Radius+radius)*pi*height + ends*pi*Radius*radius
  Area
  }
```

```
# Example calculation:

# Typically, a body part might be modelled as cylindrical if it appears to be approximately
# circular or elliptical and elongated.  By measuring the major diameter (D) and minor
# diameter (d) as well as the length or height (l), the overall surface area can be
# determined:

D<-12
d<-6
l<-18
areacylinder(Radius=D/2, radius=d/2, height=l, ends=2)


# To only show surface area of the exposed surface, and exclude the oval base of the
# cylinder, set ends=0

areacylinder(Radius=D/2, radius=d/2, height=l, ends=0)
```

`areasphere` *Provides the surface area of a sphere.*

Description

Provides the surface area of a sphere.

Usage

```
areasphere(radius)
```

Arguments

`radius`	The radius of the sphere.

Author(s)

Glenn J Tattersall

Examples

```
## The function is currently defined as
function (radius)
{
    Area <- 4 * pi * radius^2
    Area
  }

# Example calculation:
radius<-4
areasphere(radius)
```

`flip.matrix` *Flips a matrix 'left-right'. Used in re-arranging image data for plotting properly in R.*

Description

Flips a matrix 'left-right'. Used in re-arranging image data for plotting properly in R.

Usage

```
flip.matrix(x)
```

Arguments

x	A matrix corresponding to raster or image data.

Author(s)

Glenn J Tattersall

References

1. http://www.inside-r.org/packages/cran/RSEIS/docs/mirror.matrix
2. Based on similar code in package <RSEIS>

See Also

mirror.matrix rotate90.matrix rotate270.matrix rotate180.matrix

Examples

```
## The function is currently defined as
function (x)
{
    mirror.matrix(rotate180.matrix(x))
  }


par(mfrow=c(1,2),mar=c(1,1,1,1))
r<-c(1:100,rnorm(1:100)*10,1:100)
m<-matrix(r,20)
image(m, axes=FALSE)
box()
text(.5,.5,"Matrix",col="white")
mf<-flip.matrix(m)
image(mf,axes=FALSE)
box()
text(.5,.5,"Flipped",col="white")
```

flirpal — *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

forcedparameters — *Parameters required for forced convection equation.*

Description

Parameters required for forced convection equation and heat exchange estimation.

Usage

```
forcedparameters(V = 1, L = 0.1, Ta = 20, shape = "hcylinder")
```

Arguments

V	Air velocity in metres/second. Used in call to Reynolds(). Default is 0.1.
L	Characteristic dimension in metres. Default value is 0.1.
Ta	Air temperature in degrees celsius. Used in call to Reynolds(). Default is 20.
shape	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Details

Gates (2003) describes coefficients that characterise the base and exponent values used to calculate Nusselt numbers from Reynolds number as: c*Re^n. This function will return those parameters.

Value

A vector of length two, with values c and n.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

Gates, DM. 2003. Biophysical Ecology. Dover Publications, Mineola, New York, 611 pp.

See Also

freeparameters Nusseltforced

Examples

```
## The function is currently defined as
function (V = 1, L = 0.1, Ta = 20, shape = "hcylinder")
{
    Re <- Reynolds(V, L, airviscosity(Ta))
    if (shape == "vplate" | shape == "hplate")
        shape <- "plate"
    if (shape == "vcylinder" | shape == "hcylinder")
        shape <- "cylinder"
    if (shape == "plate") {
        c = 0.595
        n = 0.5
    }
    if (shape == "sphere") {
        c = 0.37
        n = 0.6
    }
    if (shape == "cylinder" & Re >= 0.4 & Re < 4) {
        c <- 0.891
        n = 0.33
    }
    if (shape == "cylinder" & Re >= 4 & Re < 40) {
        c <- 0.821
        n = 0.385
    }
    if (shape == "cylinder" & Re >= 40 & Re < 4000) {
        c <- 0.615
        n = 0.466
    }
    if (shape == "cylinder" & Re >= 4000 & Re < 40000) {
        c <- 0.174
        n = 0.618
    }
    if (shape == "cylinder" & Re >= 40000 & Re < 4e+05) {
        c <- 0.024
        n = 0.805
    }
    coeffs <- c(c, n)
    names(coeffs) <- c("c", "n")
    coeffs
  }
  # Example:
V<-1
L<-0.1
Ta<-20
shape="hcylinder"
forcedparameters(V, L, Ta, shape)

shape="vcylinder"
forcedparameters(V, L, Ta, shape)

shape="hplate"
forcedparameters(V, L, Ta, shape)

shape="vplate"
```

```
forcedparameters(V, L, Ta, shape)

shape="sphere"
forcedparameters(V, L, Ta, shape)
```

`freeparameters` *Parameters required for free convection equation.*

Description

Parameters required for free convection equation and heat exchange estimation.

Usage

```
freeparameters(L = 0.1, Ts = 30, Ta = 20, shape = "hcylinder")
```

Arguments

L	Characteristic dimension in metres. Default is 0.1.
Ts	Surface temperature (degrees Celsius) of object. Default is 30.
Ta	Air temperature (degrees Celsius) of environment. Defauly is 20.
shape	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder".

Details

Gates (2003) describes coefficients that characterise laminar flow patterns describing how to calculate Nusselt numbers for objects of different shapes. This function will return those parameters. At present, it only supplies coefficients for different shapes, not for laminar vs. turbulent since free convection is not often used in biological applications.

Value

A vector of length three, with values a, b, and m.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

Gates, DM. 2003. Biophysical Ecology. Dover Publications, Mineola, New York, 611 pp.

See Also

Nusseltfree forcedparameters

Examples

```
## The function is currently defined as
function (L = 0.1, Ts = 30, Ta = 20, shape = "hcylinder")
{
    a = 1
    Gr <- Grashof(L = 1, Ts = Ts, Ta = Ta)
    Pr <- Prandtl(Ta)
    if (shape == "hcylinder") {
        b <- 0.53
        m <- 0.25
    }
    if (shape == "vcylinder") {
        b <- 0.726
        m <- 0.25
    }
    if (shape == "hplate") {
        b <- 0.71
        m <- 0.25
    }
    if (shape == "vplate") {
        b <- 0.523
        m <- 0.25
    }
    if (shape == "sphere") {
        b <- 0.58
        m <- 0.25
    }
    coeffs <- c(a, b, m)
    names(coeffs) <- c("a", "b", "m")
    coeffs
  }

# Example:
L<-0.1
Ts<-30
Ta<-20
shape="hcylinder"
freeparameters(L, Ts, Ta, shape)

shape="vcylinder"
freeparameters(L, Ts, Ta, shape)

shape="hplate"
freeparameters(L, Ts, Ta, shape)

shape="vplate"
freeparameters(L, Ts, Ta, shape)

shape="sphere"
freeparameters(L, Ts, Ta, shape)
```

`glowbowpal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

---

`Grashof` *Determines the Grashof number for an object*

Description

Determines the Grashof number for an object. The Grashof number is used in calculations of heat exchange.

Usage

```
Grashof(L = 1, Ts = 20, Ta = 20)
```

Arguments

`L`	Characteristic dimension of object in metres. Usually height, depending on object shape.
`Ts`	Surface Temperature of object, in degrees Celsius.
`Ta`	Air/Ambient Temperature surrounding object, in degrees Celsius.

Details

The Grashof number is a dimensionless number describing the ability of a parcel of fluid warmer or colder than the surrounding fluid to rise against or fall with the attractive force of gravity as follows: Gr=agL^3(Ts-Ta)/v^2 where L is the characteristic dimension, usually the vertical dimension. For reference, a cylinder's characteristic L would be its height, assuming it is standing on its end Units of L should be in metres This L should be the same L as is used for the convective coefficient calculation Ts is the surface temperature Ta is the ambient temperature v2 is the kinematic viscosity squared (calculated from airviscosity(Ta))

Author(s)

Glenn J Tattersall

References

Blaxter, K. 1989. Energy Metabolism in Animals and Man. Gates, D. M. 2003. Biophysical Ecology. Dover Publications, Mineola, New York. 611 pp.

See Also

`airviscosity`

Examples

```
## The function is currently defined as
function (L = 1, Ts = 20, Ta = 20)
{
    a <- 1/273
    g <- 9.81
    Gr <- a * g * L^3 * (Ts - Ta)/v^2
    Gr
  }

# Typical values for Grashof number range from 0.016 to 4.6e+09 if Ts-Ta varies from
# 0.1 to 30oC

# Example calculation:
L<-1
Ts<-30
Ta<-20
Grashof(L, Ts, Ta)
```

`grey10pal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

---

`grey120pal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

---

`greyredpal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

hconv *Convective heat coefficient (W/m2/oC)*

Description

Calculates the convective heat coefficient for an object of known dimensions, and given various physical parameters.

Usage

```
hconv(Ts=30, Ta = 20, V = 1, L = 0.1, c = NULL, n = NULL, a = NULL, b = NULL, m = NULL,
type = "forced", shape="hcylinder")
```

Arguments

`Ts`	Surface temperature (degrees celsius). Required for free convection function call. Default value is 30.
`Ta`	Air temperature (degrees celsius). Default value is 20.
`V`	Air velocity (m/s). Default value is 1.
`L`	Characteristic dimension (m) of object. Usually the vertical dimension (i.e. height). Default value is 0.1.
`c`	coefficient used in forced convection (see Gates, 2003). Default value is NULL, typical values is 0.24)
`n`	coefficient used in forced convection (see Gates, 2003). Default value is NULL, typical value is 0.6)
`a`	coefficient used in forced convection (see Gates, 2003). Default value is NULL, typical value is 1.
`b`	coefficient used in free convection (see Gates, 2003). Default value is NULL, typical value is 0.58 for upright cylinder, 0.48 for horizontal cylinder.
`m`	coefficient used in free convection. Default is NULL. For laminar flow, m=0.25
`type`	"forced" or "free" - to calculate convection coefficient for either forced or free convection. Default value is "forced"
`shape`	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Details

Calculates the convection coefficient for heat transfer estimation. Used in conjunction with known temperature differences in order to estimate heat transfer via convection. Bates advises to use "forced" convection coefficients down to 0.1 m/s as appropriate for very low air flow rates, rather than distinguishing between "free" and "forced" convection.

Value

A value corresponding to the convection coefficient, units: W/m/oC.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

Gates, DM. 2003. Biophysical Ecology. Dover Publications, Mineola, New York, 611 pp.

See Also

qconv

Examples

```
## The function is currently defined as
function (Ts=30, Ta = 20, V = 1, L = 0.1, c = NULL, n = NULL, a = NULL, b = NULL,
    m = NULL, type = "forced", shape="hcylinder")
{
    if (V == 0)
        type <- "free"
    if (type == "forced" | type == "Forced")
        Nu <- Nusseltforced(c = c, n = n, V = V, L = L, Ta = Ta, shape="hcylinder")
    if (type == "free" | type == "Free")
       Nu <- Nusseltfree(a = a, b = b, m = m, L = L, Ts = Ts, Ta = Ta, shape="hcylinder")
    k <- airtconductivity(Ta)
    hconv <- Nu * k/L
    hconv
  }
```

---

hotironpal — *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

---

ironbowpal — *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

Ld *Estimates downward facing longwave radiation (W/m2)*

Description

Estimates downward incoming longwave radiation (W/m2) using relationship derived from Konzelmann et al. 1994.

Usage

```
Ld(Ta = 20, RH = 0.5, n = 0.5)
```

Arguments

`Ta`	Local air temperature (degrees Celsius), ~ 2 m above ground
`RH`	Local relative humidity (fractional value from 0 to 1)
`n`	Fractional cloud cover (fractional value from 0 to 1)

Details

By estimating the sky emissivity, from information on humidity and cloud cover, the incoming infrared radiation can be estimated using the Stephan-Boltzmann relationship: emissivity*Stephan Boltzmann constant * T^4. The effective atmospheric emissivity is determined from known cloud emissivity (0.97) and empirically determined clear sky emissivities.

Value

A value, vector of length one, corresponding to the incoming longwave radiation, units: W/m2.

Author(s)

Glenn J Tattersall

References

Konzelmann et al 1994. Parameterization of gloabl and longwave incoming radiation for the Greenland ice-sheet. Global and Planetary Change. 9: 143-164.

See Also

`Lw`

Examples

```
## The function is currently defined as
function (Ta = 20, RH = 0.5, n = 0.5)
{
    AT <- Ta + 273.15
    WVPs <- 611 * exp(17.27 * (AT - 273.15)/(AT - 36))
    WVP <- RH * WVPs
    ecs <- 0.23 + 0.443 * (WVP/AT)^(1/8)
```

```
    ecl <- 0.976
    etotal <- ecs * (1 - n^2) + ecl * n^2
    Ld <- etotal * StephBoltz() * AT^4
    Ld
  }

# Returns a value in W/m2 of the estimated incoming longwave radiation
# Example calculation:

Ta<-30
RH<-0.5
n<-0
Ld(Ta, RH, n)
```

`locate.fid` *Returns the index locations that match vector fid within data vector.*

Description

Returns the index locations that match vector fid within data vector.

Usage

```
locate.fid(fid, vect, long = TRUE)
```

Arguments

`fid`	A lookup vector, typically numeric, which can be 1 element long or greater. Typical use is 2 elements long. fid<-c(1,2). This sequence of values will be searched within the data vector, vect.
`vect`	Data vector of interest, within which fid will be searched.
`long`	Default is TRUE, will use a slower algorithm. When long=true, any length of fid can be used to search in vector. Computing time also depends on the length of fid. Caution advised when setting long = FALSE. Null values maye be returned.

Details

Returns the positions within the data vector where fid is found. Do not use this function if fid is length = 1. Use which(). If length(fid)>1, the elements of fid must be adjacent and in that specific order.

Value

An object of type integer, to be used as an index subset.

Author(s)

Glenn J. Tattersall

See Also

`match` `which`

Examples

```
# Similar to the which or match functions in package::base, except that this returns the
# index placement where variable fid occurs in data

## Define a vector
s<-c(2,3,42,38,88,33,55,99,32,56,22,48,1,2,3,5,6,7,8,9,10,12,20)
## Define what fid sequence to look for: i.e. what adjacent elements to look for in
## this order
fid<-c(22,48)
## look for all instances where 22 and 48 occur together, using locate.fid
system.time(where.locate<-locate.fid(fid,s,long=FALSE))
where.locate
## verify that locate.fid worked by subsetting s, using where.locate as index
s[where.locate]
system.time(where.locate<-locate.fid(fid,s,long=TRUE))
s[where.locate]

## longer algorithm check
### Define a vector of 100000 random numbers from 1 to 100
s<-ceiling(runif(100000, 0, 100))
## Define what fid sequence to look for: i.e. what adjacent elements to look for in
## this order
fid<-c(22,48)
system.time(where.locate<-locate.fid(fid,s,long=TRUE))
where.locate
## verify that locate.fid worked by subsetting s, using where.locate as index
s[where.locate]
```

Lu *Estimates upward facing ground radiation (W/m2)*

Description

Estimates upward facing ground radiation (W/m2), from the Stephan Boltzmann relationship and ground temperature

Usage

```
Lu(Tg = 20, Eground = 0.97)
```

Arguments

Tg	Ground temperature (degrees celsius)
Eground	Emissivity of soil or ground. Default value is 0.97.

Details

Calculates ground radiation facing upward. Assumes ground emissivity = 0.97. Terrain emissivities vary from 0.89 (sand, snow) to 0.97 (moist soil) - Blaxter, 1986

Value

A value, vector of length one, corresponding to the longwave radiation from the ground, units: W/m2.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

See Also

Ld

Examples

```
## The function is currently defined as
function (Tg = 20, Eground = 0.97)
{
    GT <- Tg + 273.15
    Lu <- Eground * StephBoltz() * (GT)^4
    Lu
  }

# Estimates ground generated longwave radiation rising up.  Units W/m2.
# Example calculation:
Tg<--30
Eground<-0.97
Lu(Tg, Eground)
```

Lw — *Estimates downward facing longwave radiation (W/m2)*

Description

Estimates downward facing longwave radiation (W/m2) using a relationship derived from Gabathuler et al 2001

Usage

```
Lw(Ta = 20, RH = 0.5, n = 0.5)
```

Arguments

Ta	Local air temperature (degrees Celsius), ~ 2 m above ground
RH	Local relative humidity (fractional value from 0 to 1)
n	Fractional cloud cover (fractional value from 0 to 1)

Details

An alternative to Ld() for estimating incoming radiation by determining an offset temperature to account for the influence of atmospheric transmission loss. The incoming infrared radiation is estimated using the Stephan-Boltzmann relationship: emissivity*Stephan Boltzmann constant*T^4

Value

A value, vector of length one, corresponding to the incoming longwave radiation, units: W/m2.

Author(s)

Glenn J Tattersall

References

Gabathuler et al 2001. Parameterization of incoming longwave radiation in high mountain environments. Physical Geography 22: 99-114

See Also

Ld

Examples

```
## The function is currently defined as
function (Ta = 20, RH = 0.5, n = 0.5)
{
    AT <- Ta + 273.15
    RH.pct <- RH * 100
    Ko <- n
    Lw <- StephBoltz() * (-21 * Ko + AT)^4 + 0.84 * RH.pct - 57
    Lw
  }

# Example calculation:
Ta<-30
RH<-0.5
n<-0
Lw(Ta, RH, n)
```

---

meanEveryN *Calculate the mean every nth data point.*

Description

meanEveryN calculates the mean of a vectorised data set (x) at N intervals. Means are calculated by centring around every nth data point in the vector. Upon running the function, it attempts to subdivide the vector into n discrete intervals. If the vector length is not fully divisible by n, then the remainder elements are forced to NA values and the final mean calculated.

The function returns a labelled matrix, with the average index as the first column and the mean over that range of data.

Usage

```
meanEveryN(x, n = 2, lag = round(n/2),showsamples=FALSE)
```

Arguments

x	numeric vector containing the data over which mean is required. Typically this is a vector of data that has been sampled at even time intervals (represented by n).
n	the sample interval over which the mean will be calculated. Default is 2 (as in every 2nd data point). At minimum this must be >1.
lag	default value is half the sample interval, n, which will ensure the calculation is centred over the new sample interval. Not tested for any other situation. Leave blank to have function operate as intended.
showsamples	default value is false. Determines whether to output a matrix where the first column contains the mean sample #. If true, the mean sample number is included with the mean calculations of the variable of interest, x. If false, then only a vector containing the mean values of x will be provided.

Details

The general purpose of this function is to assist with time based averaging a data stream typically sampled at evenly recorded time intervals common to computerised data acquisition systems. Akin to a moving average function, except that it also resamples the data.

Value

A matrix object returned

Author(s)

Glenn J. Tattersall

See Also

slopeEveryN

Examples

```
## Define a vector of 50 random numbers from 1 to 100
#s<-ceiling(runif(50, 0, 100))
#x<-seq(1,50,1)
## Calculate the mean value every 4th point
#s10<-meanEveryN(s,4)

#plot(x,s,type="l",col="red")
#lines(s10,col="black")
```

`medicalpal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

`midgreenpal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

`midgreypal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

`mikronprismpal` *Colour palette extracted from Mikron thermal camera files*

Description

A text file containing the palette information for use in thermal images

`mikroscanpal` *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

`mirror.matrix` *Mirrors a matrix upside-down. Used in re-arranging image data for plotting properly in R.*

Description

Mirrors a matrix upside-down. Used in re-arranging image data for plotting properly in R.

Usage

```
mirror.matrix(x)
```

Arguments

x	A matrix corresponding to raster or image data.

Value

Returns a matrix

Author(s)

Glenn J Tattersall

See Also

flip.matrix rotate90.matrix rotate270.matrix rotate180.matrix

Examples

```
## The function is currently defined as
function (x)
{
    xx <- as.data.frame(x)
    xx <- rev(xx)
    xx <- as.matrix(xx)
    xx
  }


# flir<-palette.choose("flir")
# par(mfrow=c(2,1),mar=c(1,1,1,1))
# r<-c(1:100,rnorm(1:100)*10,1:100)
# m<-matrix(r,50)
# image(m, axes=FALSE, col=flir)
# box()
# text(.5,.5,"Matrix",col="white")
# mf<-mirror.matrix(m)
# image(mf,axes=FALSE,col=flir)
# box()
# text(.5,.5,"Mirror",col="white")
```

`Nusseltforced` *Nusselt number for forced convection.*

Description

Nusselt number for forced convection. Used in estimating convective heat loss. Typical values of c and n are 0.24 and 0.6, respectively. This function sets c and n to NULL to force shape calculation checks.

Usage

```
Nusseltforced(c = NULL, n = NULL, V = 1, L = 0.1, Ta = 20, shape="hcylinder")
```

Arguments

`c`	coefficient used in calculating Nusselt number. Default is NULL
`n`	coefficient used in calculating Nusselt number. Default is NULL
`V`	Air velocity in metres/second. Used in call to Reynolds(). Default value is 1.
`L`	Characteristic dimension in metres. Default value is 0.1.
`Ta`	Air temperature in degrees celsius. Used in call to Reynolds().
`shape`	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Author(s)

Glenn J Tattersall

References

Gates, DM. 2003. Biophysical Ecology. Dover Publications, Mineola, New York, 611 pp. Blaxter, K. 1989. Energy Metabolism in Animals and Man

Examples

```
## The function is currently defined as
function (c = NULL, n = NULL, V = 1, L = 0.1, Ta = 20, shape="hcylinder")
{
    Nu <- c * Reynolds(V, L, Ta)^n
    Nu
  }

# Example
# Usually called from the hconv() or qconv() functions
V<-1
L<-0.1
Ta<-20
shape="hcylinder"

Nu<-Nusseltforced(V=V, L=L, Ta=Ta, shape=shape)
```

`Nusseltfree` *Nusselt number for free convection.*

Description

Nusselt number for free convection. Used in calculating heat loss by convection.

Usage

```
Nusseltfree(a=NULL, b = NULL, m = NULL, L = 0.1, Ts = 20, Ta = 20, shape="hcylinder")
```

Arguments

a	Coefficient used in calculating Nu. a is normally 1, except for turbulent flow.
b	Coefficient used in calculating Nu. b is 0.58 for upright cylinders, 0.48 for horizontal cylinders.
m	Coefficient used in calculating Nu. m=0.25 for laminar flow.
L	Characteristic dimension in metres.
Ts	Surface temperature in degrees celsius. Used in call to Grashof() function.
Ta	Air temperature in degrees celsius. Used in call to Grashof() function.
shape	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Author(s)

Glenn J Tattersall

References

Blaxter, K. 1989. Energy Metabolism in Animals and Man Gates, DM. 2003. Biophysical Ecology. Dover Publications, Mineola, New York, 611 pp.

Examples

```
## The function is currently defined as
function (a=NULL, b = NULL, m = NULL, L = 0.1, Ts = 20, Ta = 20)
{
    Nu <- b * (Grashof(L, Ts, Ta)*Prandtl(Ta)^a)^m
    Nu
  }

# Nusselt number for free convection
# Example calculation:

a<-1
b<-0.58
m<-0.25
L<-1
Ts<-30
Ta<-20
```

```
Nusseltfree(a,b,m,L,Ts,Ta)


# Free convection is higher when surface temperatures are elevated.  This is the effect
# that free convection predicts: greater molecular energy of air surrounding a warmer surface
# leading to air currents over top of a warm surface.

Ts<-40
Nusseltfree(a,b,m,L,Ts,Ta)
```

`palette.choose` *Choose a colour palette for gradient filling thermal image files.*

Description

Choose from among three the following colour palettes: flir, glowblow, grey120, grey10, greyred, hotiron, ironbow, medical, midgreen, midgrey, mikronprism, mikroscan, rain, and yellow.

Usage

```
palette.choose(colscheme)
```

Arguments

`colscheme`	A colour palette from the following: flir, glowblow, grey, grey10, greyred, hotiron, ironbow, medical, midgreen, midgrey, mikronprism, mikroscan, rain, and yellow.

Details

Colscheme is a character description drawn from the following list: ("flir", "glowblow", "grey120", "grey10", "greyred", "hotiron", "ironbow", "medical", "midgreen", "midgrey", "mikronprism", "mikroscan", "rain", "yellow")

palnames<-c("flir", "glowblow", "grey120", "grey10", "greyred", "hotiron", "ironbow", "medical", "midgreen", "midgrey", "mikronprism", "mikroscan", "rainbowpal", "yellowpal")

where "flir" is palnames[1], "rain" is palnames[13]

Value

Returns a palette to be used in various graphics functions where 'col=palette' is requested. The palette vector is formatted for use as gradient fills in plotting functions.

Author(s)

Glenn J. Tattersall

Examples

```
###### Example #####
palnames<-c("flir", "ironbow", "mikronprism", "glowbow", "grey120", "grey10", "greyred",
"hotiron",  "medical", "midgreen", "midgrey", "mikroscan", "yellowpal", "rainbowpal")
palnames<-as.matrix(palnames)

pals<-apply(as.matrix(palnames),1,palette.choose)
# add palnames to a list to call in image function below

par(mfrow=c(4,1),mar=c(1,0.3,1,0.3))
r<-c(1:500)
m<-matrix(r,500)

## Show palettes
image(m, axes=FALSE, col=flirpal, main="Flir Standard Palette")
image(m, axes=FALSE, col=ironbowpal, main="Ironbow Palette")
# smaller palette for faster plotting
image(m, axes=FALSE, col=mikronprismpal, main="Mikron Prism Palette")
image(m, axes=FALSE, col=glowbowpal, main="Glowbow Palette")
image(m, axes=FALSE, col=grey120pal, main="Grey120 Palette")
image(m, axes=FALSE, col=grey10pal, main="Grey10 Palette")
image(m, axes=FALSE, col=greyredpal, main="Greyred Palette")
image(m, axes=FALSE, col=hotironpal, main="Hotiron Palette")
image(m, axes=FALSE, col=medicalpal, main="Medical Palette")
image(m, axes=FALSE, col=midgreypal, main="Midgrey Palette")
image(m, axes=FALSE, col=mikroscanpal, main="Mikroscan Palette")
image(m, axes=FALSE, col=rainbowpal, main="Rainbow Palette")
image(m, axes=FALSE, col=yellowpal, main="Yellow Palette")

# Palettes can be run in reverse
par(mfrow=c(2,1),mar=c(1,0.3,1,0.3))
image(m, axes=FALSE, col=flirpal, main="Flir Standard Palette")
image(m, axes=FALSE, col=rev(flirpal), main="Reverse Flir Standard Palette")
```

`Prandtl` *Returns the Prandtl number*

Description

Returns the Prandtl number

Usage

```
Prandtl(Ta = 20)
```

Arguments

`Ta`	Air temperature in degrees Celsius. Default value is 20.

Details

Returns the Prandlt number

Author(s)

Glenn J Tattersall

References

Blaxter, K. 1989. Energy Metabolism in Animals and Man Gates, D. M. 2003. Biophysical Ecology. Dover Publications, Mineola, New York. 611 pp.

Examples

```
## The function is currently defined as
function (Ta = 20)
{
    Temp <- c(-50, -25, 0, 20, 40, 60, 80, 90, 100)
    Pr <- c(0.725, 0.72, 0.715, 0.713, 0.711, 0.709, 0.707, 0.7055,
        0.703)
    lm <- lm(Pr ~ poly(Temp, 6))
    Prandtl <- predict(lm, newdata = data.frame(Temp = Ta), interval = "none")
    names(Prandtl) <- NULL
    Prandtl
  }

# Example:
Ta<-30
Prandtl(Ta)
```

---

qabs *Estimates the absorbed solar and infrared radiation (W/m2)*

Description

Estimates the absorbed solar radiation and infrared radiation (W/m2) of an object using known physical relationships.

Usage

```
qabs(Ta = 20, Tg = NULL, RH = 0.5, E = 0.96, rho = 0.1, cloud = 0, SE = 100)
```

Arguments

Ta	Air temperature (degrees Celsius). Default value is 20. Used to estimate ground temperature if Tg is unavailable.
Tg	Ground temperature (degrees Celsius). Default value is NULL, but a measured Tg can be substituted or estimated with other functions.
RH	Relative humidity (fraction 0 to 1). Default value is 0.5. Used in call to Ld() to determine incoming radiation.
E	Emissivity (fraction 0 to 1) of the object absorbing longwave radiation. According to Kirschoff's law, emissivity = absorptivity. Absorptivity is multiplied by the average of the incoming longwave radiation to estimate absorbed radiation.

rho	Reflectivity (fraction 0 to 1) of the object absorbing solar radiation. Used to modify absorbed solar energy. Default is 0.1.
cloud	Fractional cloud cover (fraction from 0 to 1). Used in call to Ld() to determine incoming radiation. Default is 0.
SE	Solar energy (W/m2), usually measured. Default is 100.

Details

Total solar radiation must be supplied at this stage. The calculation here provides the worst case scenario since since no profile/angle metrics are yet taken into account. The animal could change orientation to/away from solar beam.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

See Also

Ld Lu Ld qrad

Examples

```
## The function is currently defined as
function (Ta = 25, Tg = NULL, RH = 0.5, E = 0.96, rho = 0.1,
    cloud = 0, SE = 100)
{
    if (length(SE) == 1)
        SE <- rep(SE, length(Ta))
    if (is.null(Tg))
        Tg <- Tg(Ta, SE)
    Ld <- Ld(Ta, RH = RH, n = cloud)
    Lu <- Lu(Tg)
    IR <- E * (Lu + Ld)/2
    qabs <- (1 - rho) * SE + IR
    qabs
  }

# Example:
Ta<-25
Tg<-30
RH<-0.5
E<-0.96
rho<-0.1
cloud=0
SE<-100
qabs(Ta, Tg, RH, E, rho, cloud, SE)

# If Tg is unknown it can be set to NULL, and the qabs function will estimate Tg from
# an empirical relationship of Tg vs Ta and SE from the Tground() function
```

```
qabs(Ta, Tg=NULL, RH, E, rho, cloud, SE)
```

---

qcond *Estimates the area specific heat transfer by conduction (W/m2)*

Description

Estimates the area specific heat transfer by conduction (W/m2). Positive

Usage

```
qcond(Ts = 30, Tc = 20, ktiss = 0.502, x = 1)
```

Arguments

Ts	Surface temperature (degrees Celsius). Default value is 30.
Tc	Contact temperature (degrees Celsius), usually ground temperature. Default value is 20.
ktiss	Thermal conductivity of tissue (W/m/oC).
x	Distance over which heat is conducted. Default value is 1 m (unrealistic, but easier for converting)

Details

Usually conductive heat transfer is ignored given little surface area will be in contact with the ground, but this is included for functionality.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

See Also

qrad qconv

Examples

```
## The function is currently defined as
function (Ts = 30, Tc = 20, ktiss = 0.502, x = 1)
{
    qcond <- ktiss * (Tc - Ts)/x
    qcond
  }
```

qconv — *Estimates the area specific heat transfer by convection (W/m2)*

Description

Estimates heat transfer by convective heat exchange, using the heat transfer coefficient estimate, surface temperature, and air temperature. Positive value = heat gain from air to object. Negative value = heat loss from object to air.

Usage

```
qconv(Ts = 30, Ta = 20, V = 1, L = 0.1, c = NULL, n =NULL, a=NULL, b = NULL, m = NULL,
type = "forced", shape="hcylinder")
```

Arguments

Ts	Surface temperature (degrees celsius). Default value is 30.
Ta	Air temperature (degrees celsius). Default value is 20.
V	Air velocity (m/s). Default value is 1.
L	Characteristic dimension (m) of object. Usually the vertical dimension (i.e. height). Default value is 0.1.
c	coefficient used in forced convection (see Blaxter, 1986, default value is 0.24)
n	coefficient used in forced convection (see Blaxter, 1986, default value is 0.6)
a	coefficient used in free convection (see Gates, 2003. default value is 1)
b	coefficient used in free convection (0.58 upright cylinder, 0.48 flat cylinder, default value is 0.58)
m	coefficient used in free convection (0.25 laminar flow, default value is 0.25)
type	"forced" or "free" - to calculate convection coefficient for either forced or free convection. Default value is "forced"
shape	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Details

Estimates an area specific rate of heat transfer (W/m2), where a negative value depicts heat loss from surface to air, while positive value depicts heat gain from air to surface. Uses the gradient in temperature (Ta minus Ts) multiplied by a convection coefficient to estimate heat transfer from a surface. Designed for estimating steady state heat exchange from animal surfaces using thermal images.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

See Also

hconv

Examples

```
## The function is currently defined as
function (Ts = 30, Ta = 20, V = 1, L = 0.1, c = NULL, n = NULL, a=NULL,
    b = NULL, m = NULL, type = "forced", shape="hcylinder")
{
    qconv <- (Ta - Ts) * hconv(Ta = 20, V = 1, L = 0.1, c = NULL, n = NULL, a=NULL,
    b = NULL, m = NULL, type = "forced", shape="hcylinder")
    qconv
  }

# Example:
Ts<-30
Ta<-20
V<-1
L<-0.1
type="forced"
shape="hcylinder"

qconv(Ts=Ts, Ta=Ta, V=V, L=L, type=type, shape=shape)

qconv(Ts=Ts, Ta=Ta, V=V, L=L, type=type, shape="sphere")
```

---

qrad — *Estimates the area specific heat transfer by radiation (W/m2)*

Description

Estimates heat transfer by radiation (W/m2), using the absorbed radiation estimate from qabs() minus emitted radiation from the object surface (determined from thermal image surface temperature estimates). Positive value = heat gain from environment to object. Negative value = heat loss from object to environment.

Usage

```
qrad(Ts = 30, Ta = 25, Tg = NULL, RH = 0.5, E = 0.96, rho = 0.1, cloud = 0, SE = 0)
```

Arguments

Ts	Surface temperature (degrees Celsius) of the object. Default value is 30.
Ta	Air temperature (degrees Celsius), or effective atmospheric temperature. Default value is 25.
Tg	Ground temperature (degrees Celsius) to estimate longwave ground radiation. Default value is NULL, since Tg can be estimated from Ta unless otherwise measured.
RH	Relative humidity (fraction 0 to 1). Default value is 0.5. Used in call to Ld() to determine incoming radiation.

E	Emissivity (fraction 0 to 1) of the object absorbing longwave radiation. According to Kirschoff's law, emissivity = absorptivity. Absorptivity is multiplied by the average of the incoming longwave radiation to estimate absorbed radiation.
rho	Reflectivity (fraction 0 to 1) of the object absorbing solar radiation. Used to modify absorbed solar energy. Default is 0.1.
cloud	Fractional cloud cover (fraction from 0 to 1). Used in call to Ld() to determine incoming radiation. Default is 0.
SE	Solar energy (W/m2), usually measured. Default is 100.

Details

Total solar radiation must be supplied at this stage. The calculation here provides the worst case scenario since since no profile/angle metrics are yet taken into account. The animal could change orientation to/away from solar beam.

Author(s)

Glenn J Tattersall

References

Blaxter, 1986. Energy metabolism in animals and man. Cambridge University Press, Cambridge, UK, 340 pp.

See Also

Ld Lu Ld qabs

Examples

```
## The function is currently defined as
function (Ts = 30, Ta = 25, Tg = NULL, RH = 0.5, E = 0.96, rho = 0.1,
    cloud = 0, SE = 0)
{
    qrad <- qabs(Ta = Ta, Tg = Tg, RH = RH, E = E, rho = rho,
        cloud = cloud, SE = SE) - E * StephBoltz() * (Ts + 273.15)^4
    qrad
  }

# Example:
Ts<-30
Ta<-25
Tg<-28
RH<-0.5
E<-0.96
rho<-0.1
cloud<-0
SE<-100
# qrad should result in a positive gain of heat:
qrad(Ts, Ta, Tg, RH, E, rho, cloud, SE)

# if rho is elevated (i.e. doubles reflectance of solar energy), heat exchange by
# radiation is reduced
rho<-0.2
```

```
qrad(Ts, Ta, Tg, RH, E, rho, cloud, SE)

# But if solar energy = 0, under similar conditions, qrad is negative:
SE<-0
qrad(Ts, Ta, Tg, RH, E, rho, cloud, SE)
```

---

rainbowpal *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

---

raw2temp *Converts raw thermal data into temperature (oC)*

Description

Converts a raw value obtained from binary thermal image video file into estimated temperature using standard equations used in infrared thermography.

Usage

```
raw2temp(raw, E = 1, OD = 1, RTemp = 20, ATemp = RTemp, IRWTemp = RTemp, IRT = 1,
RH = 50, PR1 = 21106.77, PB = 1501, PF = 1, PO = -7340, PR2 = 0.012545258)
```

Arguments

raw	A/D bit signal from FLIR file. FLIR .seq files and .fcf files store data in a 16-bit encoded value. This means it can range from 0 up to 65535. This is referred to as the raw value. The raw value is actually what the sensor detects which is related to the radiance hitting the sensor. At the factory, each sensor has been calibrated against a blackbody radiation source so calibration values to conver the raw signal into the expected temperature of a blackbody radiator are provided. Since the sensors do not pick up all wavelengths of light, the calibration can be estimated usinga limited version of Planck's law. But the blackbody calibration is still critical to this.
E	Emissivity - default 1, should be ~0.95 to 0.97 depending on object of interest. Determined by user.
OD	Object distance from thermal camera in metres
RTemp	Apparent reflected temperature (oC) of the enrivonment impinging on the object of interest - one value from FLIR file (oC), default 20C.
ATemp	Atmospheric temperature (oC) for infrared tranmission loss - one value from FLIR file (oC) - default value is set to be equal to the reflected temperature. Transmission loss is a function of absolute humidity in the air.

IRWTemp	Infrared Window Temperature (oC). Default is set to be equivalent to reflected temp (oC).
IRT	Infrared Window transmission - default is set to 1.0. Likely ~0.95-0.97. Should be empirically determined. Germanium windows with anti-reflective coating typically have IRTs ~0.95-0.97.
RH	Relative humidity expressed as percent. Default value is 50.
PR1	PlanckR1 - a calibration constant for FLIR cameras
PB	PlanckB - a calibration constant for FLIR cameras
PF	PlanckF - a calibration constant for FLIR cameras
PO	PlanckO - a calibration constant for FLIR cameras
PR2	PlanckR2 - a calibration constant for FLIR cameras

Details

Note: PR1, PR2, PB, PF, and PO are specific to each camera and result from the calibration at factory of the camera's Raw data signal recording from a blackbody radiation source. Sample calibration constants for three different cameras (FLIR SC660 with 24x18 degree lens, FLIR T300 with 25x19 degree lens, FLIR T300 with 2xtelephoto.

Calibration Constants by cameras: SC660, T300(25o), T300(25o with telephoto)

Constant	FLIR SC660	FLIR T300	FLIR T300(t)
PR1:	21106.77	14364.633	14906.216
PB:	1501	1385.4	1396.5
PF:	1	1	1
PO:	-7340	-5753	-7261
PR2:	0.012545258	0.010603162	0.010956882

PR1: PlanckR1 calibration constant PB: PlanckB calibration constant PF: PlanckF calibration constant PO: PlanckO calibration constant PR2: PlanckR2 calibration constant

The calibration constants allow for the raw digital signal conversion to and from the predicted radiance of a blackbody, using the standard equation:

temperature<-PB/log(PR1/(PR2*(raw+PO))+PF)-273.15

Also used in calculations for transmission loss are the following constants:

ATA1: Atmospheric Trans Alpha 1 0.006569

ATA2: Atmospheric Trans Alpha 2 0.012620

ATB1: Atmospheric Trans Beta 1 -0.002276

ATB2: Atmospheric Trans Beta 2 -0.006670

ATX: Atmospheric Trans X 1.900000

Value

Returns numeric value in oC. Can handle vector or matrix objects

Warning

Raw values need to be greater than Planck0 constant

Author(s)

Glenn J. Tattersall

References

1. http://130.15.24.88/exiftool/forum/index.php/topic,4898.60.html

2. Minkina, W. and Dudzik, S. 2009. Infrared Thermography: Errors and Uncertainties. Wiley Press, 192 pp.

See Also

temp2raw,

Examples

```
# General Usage:
# raw2temp(raw,E,OD,RTemp,ATemp,IRWTemp,IRT,RH,PR1,PB,PF,PO,PR2)
###
# Example with all settings at default/blackbody levels:
raw2temp(18109,1,0,20,20,20,1,50,PR1=21106.77,PB=1501,PF=1,PO=-7340,PR2=0.012545258)

# Example with emissivity=0.95, distance=1m, window transmission=0.96, all temperatures=20C,
# 50 RH:

raw2temp(18109,0.95,1,20,20,20,0.96,50)
# Note: default calibration constants for the FLIR camera will be used if you leave out the
# calibration data

# Vector example
r<-17000:25000
t1.0<-raw2temp(r,1,0,20,20,20,0.96,50)
t0.9<-raw2temp(r,0.9,0,20,20,20,0.96,50)

dev.off()
plot(r,t1.0,type="l",col="red")
lines(r,t0.9,col="black")
legend("topleft", bty = "n", c("E=1.0", "E=0.9"), lty=c(1,1), col=c("red", "black"))
```

Reynolds *Calculates the Reynolds number.*

Description

Calculates the Reynolds number, a unitless measure.

Usage

```
Reynolds(V, L, v)
```

Arguments

V	Air velocity in m/s

L	The characteristic dimension, usually the vertical dimension. For reference, a cylinder's characteristic L would be its height, assuming it is standing on its end This L should be the same L as is used for the convective coefficient calculation

v	The kinematic viscosity returned from function airviscosity (Ta).

Author(s)

Glenn J Tattersall

References

Blaxter, K. 1989. Energy Metabolism in Animals and Man Gates, D. M. 2003. Biophysical Ecology. Dover Publications, Mineola, New York. 611 pp.

Examples

```
## The function is currently defined as
function (V, L, v)
{
  v<-airviscosity(Ta)
  Re<-V*L/v
  }

# Typical values for Reynolds numbers range from 6.6 to 6.6e+5

# Example calculation:
V<-1
L<-1
Ta<-20
v<-airviscosity(Ta)
Reynolds(V, L, v)
```

---

rotate180.matrix   *Rotate a matrix by 180 degrees. Used for adjusting image plotting in R.*

Description

Rotate a matrix by 180 degrees. Used for adjusting image plotting in R.

Usage

```
rotate180.matrix(x)
```

Arguments

x	A matrix corresponding to raster or image data.

Value

Returns a matrix

Author(s)

Glenn J Tattersall

References

1. http://www.inside-r.org/packages/cran/RSEIS/docs/mirror.matrix
2. Based on similar code in package <RSEIS>

See Also

flip.matrix mirror.matrix rotate90.matrix rotate270.matrix

Examples

```
## The function is currently defined as
function (x)
{
    xx <- rev(x)
    dim(xx) <- dim(x)
    xx
  }


# flir<-palette.choose("flir")
# set.seed(5)
# par(mfrow=c(1,2),mar=c(1,1,1,1))
# r<-c(1:100,rnorm(1:100)*10,1:100)
# m<-matrix(r,50)
# image(m, axes=FALSE, col=flir)
# box()
# text(.5,.5,"Matrix",col="white")
# mf<-rotate180.matrix(m)
# image(mf,axes=FALSE,col=flir)
# box()
# text(.5,.5,"Rotate180",col="white")
```

---

rotate270.matrix *Rotate a matrix by 270 degrees counterclockwise (or 90 degree clockwise). Used for adjusting image plotting in R.*

Description

Rotate a matrix by 270 degrees counterclockwise (or 90 degree clockwise). Used for adjusting image plotting in R.

Usage

```
rotate270.matrix(x)
```

Arguments

x	A matrix corresponding to raster or image data.

Value

Returns a matrix

Author(s)

Glenn J Tattersall

References

1. http://www.inside-r.org/packages/cran/RSEIS/docs/mirror.matrix
2. Based on similar code in package <RSEIS>

See Also

flip.matrix mirror.matrix rotate90.matrix rotate180.matrix

Examples

```
## The function is currently defined as
function (x)
{
    mirror.matrix(t(x))
  }



 flir<-palette.choose("flir")
 set.seed(5)
 par(mfrow=c(1,2),mar=c(1,1,1,1))
 r<-c(1:100,rnorm(1:100)*10,1:100)
 m<-matrix(r,50)
 image(m, axes=FALSE, col=flir)
 box()
 text(.5,.5,"Matrix",col="white")
 mf<-rotate270.matrix(m)
 image(mf,axes=FALSE,col=flir)
 box()
 text(.5,.5,"Rotate270",col="white")
```

`rotate90.matrix` *Rotate a matrix by 90 degrees counterclockwise (270 degrees clockwise). Used for adjusting image plotting in R.*

Description

Rotate a matrix by 90 degrees counterclockwise (270 degrees clockwise). Used for adjusting image plotting in R.

Usage

```
rotate90.matrix(x)
```

Arguments

x	A matrix corresponding to raster or image data.

Value

Returns a matrix.

Author(s)

Glenn J. Tattersall

References

1. http://www.inside-r.org/packages/cran/RSEIS/docs/mirror.matrix
2. Based on similar code in package <RSEIS>

See Also

`flip.matrix` `mirror.matrix` `rotate270.matrix` `rotate180.matrix`

Examples

```
## The function is currently defined as
function (x)
{
    t(mirror.matrix(x))
  }


flir<-palette.choose("flir")
set.seed(5)
par(mfrow=c(1,2),mar=c(1,1,1,1))
r<-c(1:100,rnorm(1:100)*10,1:100)
m<-matrix(r,50)
image(m, axes=FALSE, col=flir)
box()
text(.5,.5,"Matrix",col="white")
mf<-rotate90.matrix(m)
```

```
image(mf,axes=FALSE,col=flir)
box()
text(.5,.5,"Rotate90",col="white")
```

---

samp.image *A sample thermal image to demonstrate thermal colour palette use.*

Description

A sample thermal image to demonstrate thermal colour palette use.

Usage

```
data("samp.image")
```

Format

A sample thermal image to demonstrate thermal colour palette use. The format is: num [1:480, 1:640] 23.2 23.2 23.4 23.3 23.3 ...

Examples

```
###### Example #####
palnames<-c("flir", "ironbow", "mikronprism", "glowbow", "grey120", "grey10", "greyred",
"hotiron",  "medical", "midgreen", "midgrey", "mikroscan", "yellowpal", "rainbowpal")

m<-rotate90.matrix(samp.image)
par(mfrow=c(2,1),mar=c(0.3,2,1,2))

## Show palettes
image(m, axes=FALSE, useRaster=TRUE, col=flirpal, main="Flir Standard Palette")
image(m, axes=FALSE, useRaster=TRUE, col=ironbowpal, main="Ironbow Palette")
# smaller palette for faster plotting
image(m, axes=FALSE, useRaster=TRUE, col=mikronprismpal, main="Mikron Prism Palette")
image(m, axes=FALSE, useRaster=TRUE, col=glowbowpal, main="Glowbow Palette")
image(m, axes=FALSE, useRaster=TRUE, col=grey120pal, main="Grey120 Palette")
image(m, axes=FALSE, useRaster=TRUE, col=grey10pal, main="Grey10 Palette")
image(m, axes=FALSE, useRaster=TRUE, col=greyredpal, main="Greyred Palette")
image(m, axes=FALSE, useRaster=TRUE, col=hotironpal, main="Hotiron Palette")
image(m, axes=FALSE, useRaster=TRUE, col=medicalpal, main="Medical Palette")
image(m, axes=FALSE, useRaster=TRUE, col=midgreypal, main="Midgrey Palette")
image(m, axes=FALSE, useRaster=TRUE, col=mikroscanpal, main="Mikroscan Palette")
image(m, axes=FALSE, useRaster=TRUE, col=rainbowpal, main="Rainbow Palette")
image(m, axes=FALSE, useRaster=TRUE, col=yellowpal, main="Yellow Palette")
```

slopebypoint *Returns the slope from linear regression with x values as equally spaced 1:length*

Description

Returns the slope from linear regression with x values as equally spaced 1:length

Usage

```
slopebypoint(data)
```

Arguments

data	Returns the slope from linear regression with x values as equally spaced 1:length

Details

Returns the slope (i.e. localised tangent) from linear regression with x values as equally spaced 1:length. The usefulness of this function is to reduce a time series type of data collected at equal time intervals.

N=number of data points over which to calculate the slope.

Value

An object of type numeric.

Author(s)

Glenn J. Tattersall

See Also

lm

Examples

```
## Define a vector of 50 random numbers from 1 to 100
y<-ceiling(runif(50, 0, 100))
# Calculate the slope with respect to the index values (i.e. 1 to 50)
# instead of an x axis, this will provide a slope value of y vs. index
s<-slopebypoint(y)
s

# same as if typing:
lm(y~seq(0,length(y)-1,1))
```

`slopeEveryN` *Calculate the slope every nth data point.*

Description

slopeEveryN calculates the slope of a vectorised data set (x) at N intervals. Slopes are calculated using the lm() function centred around every nth data point in the vector. Upon running the function, it attempts to subdivide the vector into n discrete intervals. If the vector length is not fully divisible by n, then the remainder elements are forced to NA values and the final slope calculated.

The function returns a labelled matrix, with the average index as the first column and the slope over that range of data. Units for slope then are technically in un

Usage

```
slopeEveryN(x, n = 2, lag = round(n/2))
```

Arguments

`x`	numeric vector containing the data over which slope is required. Typically this is a vector of data that has been sampled at even time intervals (represented by n).
`n`	the sample interval over which the slope will be calculated. Default is 2 (as in every 2nd data point). At minimum this must be >1.
`lag`	default value is half the sample interval, n, which will ensure the calculation is centred over the new sample interval. Not tested for any other situation. Leave blank to have function operate as intended.

Details

he general purpose of this function is to provide a moving average of a data stream typically sampled at evenly recorded time intervals common computerised data acquisition systems. Akin to a moving average function, except that it also resamples the data.

Value

A matrix object returned

Author(s)

Glenn J. Tattersall

See Also

`slopebypoint`

Examples

```
## Define a vector of 50 random numbers from 1 to 100
s<-ceiling(runif(50, 0, 100))
x<-seq(1,50,1)
# Calculate the slope value every 4th point
s10<-slopeEveryN(s,4)

plot(x,s,type="l",col="red")
lines(s10,col="black")
```

StephBoltz *The Stephan Boltzman constant.*

Description

The Stephan Boltzman constant. Units: W/m^2/K^4

Usage

```
StephBoltz()
```

Author(s)

Glenn J Tattersall

Examples

```
## The function is currently defined as
function ()
{
    s <- 5.67e-08
    s
  }

# Example
# This is simply the Stephan Boltzmann constant, saves having to remember the exact value
# and it allows easier coding.  To call it, type:

StephBoltz()
```

Te *Operative temperature estimate.*

Description

Operative temperature (degrees Celsius) is a measure of the effective temperature an object/animal will be given a specific radiative and convective environment. Basal heat production and evaporative heat loss are assumed to balance each other out.

Usage

```
Te(Ts=30, Ta=25, Tg=NULL, RH=0.5, E=0.96, rho=0.1, cloud=0, SE=0, V=1,
L=0.1, c=NULL, n=NULL, a=NULL, b=NULL, m=NULL, type="forced", shape="hcylinder")
```

Arguments

Ts	Surface temperature (degrees Celsius). Default value is 30. Used in free convection calculation.
Ta	Air temperature (degrees Celsius). Default value is 20. Used to estimate ground temperature if Tg is unavailable.
Tg	Ground temperature (degrees Celsius). Default value is NULL, but a measured Tg can be substituted or estimated with other functions.
RH	Relative humidity (fraction 0 to 1). Default value is 0.5. Used in call to Ld() to determine incoming radiation.
E	Emissivity (fraction 0 to 1) of the object absorbing longwave radiation. According to Kirschoff's law, emissivity = absorptivity. Absorptivity is multiplied by the average of the incoming longwave radiation to estimate absorbed radiation.
rho	Reflectivity (fraction 0 to 1) of the object absorbing solar radiation. Used to modify absorbed solar energy. Default is 0.1.
cloud	Fractional cloud cover (fraction from 0 to 1). Used in call to Ld() to determine incoming radiation. Default is 0.
SE	Solar energy (W/m2), usually measured. Default is 100.
V	Air velocity (m/s). Default value is 1.
L	Characteristic dimension (m) of object. Usually the vertical dimension (i.e. height). Default value is 1.
c	coefficient used in forced convection (see Blaxter, 1986, default value is 0.24)
n	coefficient used in forced convection (see Blaxter, 1986, default value is 0.6)
a	coefficient used in free convection (see Gates, 2003, default value is 1)
b	coefficient used in free convection (0.58 upright cylinder, 0.48 flat cylinder, default value is 0.58)
m	coefficient used in free convection (0.25 laminar flow, default value is 0.25)
type	"forced" or "free" - to calculate convection coefficient for either forced or free convection. Default value is "forced"
shape	"sphere", "hplate", "vplate", "hcylinder", "vcylinder" to denote shape and orientation. h=horizontal, v=vertical. Default shape is "hcylinder"

Details

Estimates operative temperature according to calculations in Gates (2003) and Angiletta ()

Author(s)

Glenn J Tattersall

References

Angiletta, M. J. 2009. Thermal Adaptation: A Theoretical and Empirical Synthesis. Oxford University Press, Oxford, UK, 304 pp. Gates, D.M. 2003. Biophysical Ecology. Courier Corporation, 656 pp.

See Also

qabs hconv

Examples

```
## The function is currently defined as
function (Ts=30, Ta=25, Tg=NULL, RH=0.5, E=0.96, rho=0.1, cloud=0, SE=0, V=1,
L=0.1, c=NULL, n=NULL, a=NULL, b=NULL, m=NULL, type="forced", shape="hcylinder")
{
    Te <- Ta + (qabs(Ta=Ta, Tg=Tg, RH=RH, E=E, rho=rho, cloud=cloud,
    SE=SE) - StephBoltz()*0.96*(Ta+273.15)^4) /
    (hconv(Ts=Ts, Ta=Ta, V=V, L=L, c = NULL, n = NULL, a = NULL, b = NULL, m = NULL,
    type=type, shape=shape) + 4*StephBoltz()*0.96*(Ta+273.15)^3)
    Te
  }

# Example

Ts<-40
Ta<-30
SE<-seq(0,1500,100)
Toperative<-NULL
for(rho in seq(0, 1, 0.1)){
  temp<-Te(Ts=Ts, Ta=Ta, Tg=NULL, RH=0.5, E=0.96, rho=rho, cloud=1, SE=SE, V=0.1,
           L=0.1, type="free", shape="hcylinder")
  Toperative<-cbind(Toperative, temp)
}
Toperative<-data.frame(SE=seq(0,1500,100), Toperative)
colnames(Toperative)<-c("SE", seq(0,1,0.1))
matplot(Toperative$SE, Toperative[,-1], ylim=c(30, 50), type="l", xlim=c(0,1000),
        ylab="Operative Temperature (C)", xlab="Solar Radiation (W/m2)", lty=1,
        col=flirpal[rev(seq(1,380,35))])
```

`temp2raw` *Converts temperature (oC) to raw thermal data*

Description

Inverse of the function raw2temp. Typically used when incorrect settings were used during thermal imaging analysis, and the raw values need to be extracted in order to re-calculate temperature using raw2temp. Parameters under which the temperatures were estimated should be known, since the conversion to raw will take those into account.

Usage

```
temp2raw(temp, E = 1, OD = 1, RTemp = 20, ATemp = RTemp, IRWTemp = RTemp, IRT = 1,
RH = 50, PR1 = 21106.77, PB = 1501, PF = 1, PO = -7340, PR2 = 0.012545258)
```

Arguments

`temp`	estimate temperature (oC) from an infrared thermal imaging file
`E`	Emissivity - default 1, should be ~0.95 to 0.97 depending on object of interest. Determined by user.
`OD`	Object distance from thermal camera in metres
`RTemp`	Apparent reflected temperature (oC) of the enrivonment impinging on the object of interest - one value from FLIR file (oC), default 20C.
`ATemp`	Atmospheric temperature (oC) for infrared tranmission loss - one value from FLIR file (oC) - default value is set to be equal to the reflected temperature. Transmission loss is a function of absolute humidity in the air.
`IRWTemp`	Infrared Window Temperature (oC). Default is set to be equivalent to reflected temp (oC).
`IRT`	Infrared Window transmission - default is set to 1.0. Likely ~0.95-0.97. Should be empirically determined. Germanium windows with anti-reflective coating typically have IRTs ~0.95-0.97.
`RH`	Relative humidity expressed as percent. Default value is 50.
`PR1`	PlanckR1 - a calibration constant for FLIR cameras
`PB`	PlanckB - a calibration constant for FLIR cameras
`PF`	PlanckF - a calibration constant for FLIR cameras
`PO`	PlanckO - a calibration constant for FLIR cameras
`PR2`	PlanckR2 - a calibration constant for FLIR cameras

Details

Note: PR1, PR2, PB, PF, and PO are specific to each camera and result from the calibration at factory of the camera's Raw data signal recording from a blackbody radiation source. Sample calibration constants for three different cameras (FLIR SC660 with 24x18 degree lens, FLIR T300 with 25x19 degree lens, FLIR T300 with 2xtelephoto.

Calibration Constants by cameras: SC660, T300(25o), T300(25o with telephoto)

Constant	FLIR SC660	FLIR T300	FLIR T300(t)

PR1:	21106.77	14364.633	14906.216
PB:	1501	1385.4	1396.5
PF:	1	1	1
PO:	-7340	-5753	-7261
PR2:	0.012545258	0.010603162	0.010956882

PR1: PlanckR1 calibration constant PB: PlanckB calibration constant PF: PlanckF calibration constant PO: PlanckO calibration constant PR2: PlanckR2 calibration constant

The calibration constants allow for the raw digital signal conversion to and from the predicted radiance of a blackbody, using the standard equation:

temperature<-PB/log(PR1/(PR2*(raw+PO))+PF)-273.15

Also used in calculations for transmission loss are the following constants:

ATA1: Atmospheric Trans Alpha 1 0.006569

ATA2: Atmospheric Trans Alpha 2 0.012620

ATB1: Atmospheric Trans Beta 1 -0.002276

ATB2: Atmospheric Trans Beta 2 -0.006670

ATX: Atmospheric Trans X 1.900000

Value

Returns numeric value. Can handle vector or matrix objects

Author(s)

Glenn J. Tattersall

References

1. http://130.15.24.88/exiftool/forum/index.php/topic,4898.60.html

2. Minkina, W. and Dudzik, S. 2009. Infrared Thermography: Errors and Uncertainties. Wiley Press, 192 pp.

See Also

raw2temp

Examples

```
# General Usage:
# temp2raw(temp,E,OD,RTemp,ATemp,IRWTemp,IRT,RH,PR1,PB,PF,PO,PR2)

# Example with all settings at default/blackbody levels:
temp2raw(23,1,0,20,20,20,1,50,PR1=21106.77,PB=1501,PF=1,PO=-7340,PR2=0.012545258)

# Example with emissivity=0.95, distance=1m, window transmission=0.96, all temperatures=20C,
# 50 RH:

temp2raw(23,0.95,1,20,20,20,0.96,50)
# Note: default calibration constants for my FLIR camera will be used if you leave out the
# calibration data
```

```
t<-10:50
r1.0<-temp2raw(t,1,0,20,20,20,0.96,50)
r0.9<-temp2raw(t,0.9,0,20,20,20,0.96,50)

dev.off()
plot(t,r1.0,type="l",col="red")
lines(t,r0.9,col="black")
legend("topleft", bty = "n", c("E=1.0", "E=0.9"), lty=c(1,1), col=c("red", "black"))
```

`Tground` *Estimates ground temperature from ambient temperature and solar radiation.*

Description

Estimates ground temperature from ambient temperature and solar radiation.

Usage

```
Tground(Ta = 20, SE = 100)
```

Arguments

`Ta`	Air temperature (degrees Celsius). Default is 20.
`SE`	Solar energy (radiation in W per m2). Default is 100.

Details

If ground temperature is not measured, but air temperature and solar energy are provided, ground temperature can be estimated from empirical relationships. Ground temperature is used in obtain incoming longwave radiation from the ground.

Value

Returns a vector of one, with an estimate of ground temperature.

Author(s)

Glenn J Tattersall

References

Bartlett et al. 2006. A decade of ground-air temperature tracking at emigrant pass observatory, Utah. Journal of Climate. 19: 3722-3731.

Examples

```
## The function is currently defined as
function (Ta = 20, SE = 100)
{
    Tground <- 0.0121 * SE + Ta
    names(Tground) <- "Tground"
    Tground
  }

# Example:
Ta<-25
SE<-200
Tground(Ta, SE)
```

---

yellowpal   *Colour palette extracted from FLIR thermal camera files*

Description

A text file containing the palette information for use in thermal images

Index