Package ‘Thermimage’

March 24, 2017

Type Package

Title Thermal Image Analysis

Version 3.0.0

Date 2016-10-18

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 and an import function for thermal image files (v2.2.3 fixed error importing jpg to windows OS). v3. Added numerous functions for importing thermal image videos, rewriting and exporting.

License GPL-2

Depends R (>= 2.10)

SystemRequirements Exiftool (http://www.sno.phy.queensu.ca/~phil/exiftool/)

Suggests ggplot2, fields, reshape

Imports tiff, png

LazyData true

URL https://cran.r-project.org/package=Thermimage, https://github.com/gtatters/Thermimage, https://github.com/gtatters/Thermimage/blob/master/heatcalc.R

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

RoxygenNote 6.0.1

R topics documented:

Thermimage-package 3
airdensity 3
airspecificheat 4
airtconductivity 5

airviscosity . . . 6
areacone . . . 7
areacylinder . . . 8
areasphere . . . 9
cumulDiff . . . 10
diffFrame . . . 11
flip.matrix . . . 13
flirpal . . . 14
flirsettings . . . 14
forcedparameters . . . 16
frameLocates . . . 17
freeparameters . . . 19
getFrames . . . 21
getTimes . . . 23
glowbowpal . . . 24
Grashof . . . 25
grey10pal . . . 26
grey120pal . . . 26
greyredpal . . . 26
hconv . . . 26
hotironpal . . . 28
ironbowpal . . . 28
Ld . . . 28
locate.fid . . . 29
Lu . . . 30
Lw . . . 31
meanEveryN . . . 32
medicalpal . . . 33
midgreenpal . . . 34
midgreypal . . . 34
mikronprismpal . . . 34
mikroscanpal . . . 34
mirror.matrix . . . 35
nameleadzero . . . 36
Nusseltforced . . . 37
Nusseltfree . . . 38
palette.choose . . . 39
plotTherm . . . 40
Prandtl . . . 43
qabs . . . 44
qcond . . . 45
qconv . . . 46
qrad . . . 48
rainbowpal . . . 49
raw2temp . . . 50
readflirJPG . . . 52
Reynolds . . . 54
rotate180.matrix . . . 55
rotate270.matrix . . . 56
rotate90.matrix . . . 57
samp.image . . . 58
slopebypoint . . . 59

slopeEveryN . . . 60
StephBoltz . . . 61
Te . . . 62
temp2raw . . . 63
Teq . . . 66
Tground . . . 67
thermsum . . . 68
thermsumcent . . . 70
writeFlirBin . . . 72
yellowpal . . . 73

Index **74**

`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.

`airdensity` *Returns the density of air for a given air temperature.*

Description

Density of air if temperature (degrees Celsius) provided. Units: kg/m3

Usage

```
airdensity(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)
{
    Base <- 314.156
    Exponent <- (-0.981)
    p <- Base * (Ta + 273.15)^Exponent
    p
  }
```

`airspecificheat` *Specific heat capacity of air*

Description

Specific heat capacity of air if temperature (degrees Celsius) provided. Units: J/(kg*K)

Usage

```
airspecificheat(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 <- 1.003731424
    Slope1 <- 5.37909e-06
    Slope2 <- 7.30124e-07
    Slope3 <- (-1.34472e-09)
    Slope4 <- 1.23027e-12
    cp <- 1000*(Intercept + Slope1 * Ta + Slope2 * Ta^2 + Slope3 *
        Ta^3 + Slope4 * Ta^4)
    cp
  }
```

`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

See Also

`airviscosity`

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 viscosity 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)
```

`cumulDiff` *Cumulative difference sum function for use with frame by frame difference dataframe*

Description

Based on the absolute difference sum method (Lighton and Turner, 2004), this function takes a difference frame dataframe, where each column corresponds to a video frame (i+1) that has been subtracted from the previous (ith) frame. Each row corresponds to a pixel difference value.

Usage

```
cumulDiff(fdiff, extract.times, samples = 2)
```

Arguments

`fdiff`	Dataframe containing the frame by frame differences obtained from the diffFrame function. Rows corresponds to the pixel dimensions (w x h) of each frame and Columns (C-1) correpond to the number of columns, which is one fewer columns compared to the original video dataframe.
`extract.times`	A vector of times (POSIXct format) that corresponds to the actual frames from the original video file. This should be length of C.
`samples`	The number of samples over which to calculate the slope of the cumulative difference sums. Must be >= 2, as it will calculate the slope over at least two frames.

Details

Each row in fdiff corresponds to a specific pixel position in a thermal video frame. Data frames are preferred over array functions for speed and simplicity. Row numbers range from 1 through to the image dimensions (i.e. w*h = 640 * 480=307200). Image dimensions are not required, provided the row number corresponds to the same relative position.

The premise behind this is that the thermal video is either time lapse or higher speed video. If a specific pixel shows no change (0) from frame to frame, then there is no movement or temperature change. For videos of living specimens, movement artefacts will manifest as change over time at specific pixels. If there is sufficient movement, across the image space, the accumulation of small differences will provide a measure of relative activity from frame to frame.

cumulDiff takes the average, standard deviation and rootmean square of all pixels within one frame to arrive at an aggregate value for each difference frame (absolute value). Subsequently, it sums these successive data points (avg,sd,rms) across all frames, arriving at an absolute difference summation. This results in an incrementing value, of which the slope will be a semi-quantitative assessment of relative change. It also provides a clean break point when activity ceases (Lighton, 2008).

The extract.times value (POSIX) is required to provide a time index as well as to calculate the frame rate.

Value

Returns a list variable, containing raw, cumulative difference calculations and the slope calculations on a minimum of 2, preferrably every 3rd frame.

`rawdiff`	rawdiff is a table of the cumulative average, sd, and rms values
`slopediff`	slopediff is the summarised rates of change over time in the rawdiff values

Author(s)

Glenn J Tattersall

References

1. Lighton, J.R.B., and Turner, R.J. (2004). Thermolimit respirometry: an objective assessment of critical thermal maxima in two sympatric desert harvester ants, Pogonomyrmex rugosus and P. californicus. J Exp Biol 207: 1903-1913.

2. Lighton, J. R. B. (2008). Measuring metabolic rates : a manual for scientists. Oxford ; New York, Oxford University Press.

See Also

`diffFrame`

Examples

```
# Create a vector of arbitrary frame times - these would be extracted normally using the
# locateFrames and getTimes functions

start<-as.POSIXct("2017-03-31 12:00:00")
fdiff<-data.frame(matrix(runif(307200*20, 20, 40), nrow=307200))

# add noise to pixels
for(i in 1:20){
  randpixels<-floor(runif(10000, 1,307200))
  fdiff[randpixels,i]<-fdiff[randpixels,i]*runif(1, 10, 10000)
}

extract.times<-seq(start, start+20,1)
cumulDiff(fdiff, extract.times, 2)
```

`diffFrame` *A frame difference function for subtracting adjacent frames from an imported thermal image sequence.*

Description

Works similarly to the simple diff() function, but on a data.frame. Subtracts column i from column i+1, assuming each column represents the pixel information for one frame of an imported thermal image video. Each row in the column corresponds to a pixel. Returns a data.frame of one column shorter dimension than the original data.frame.

Usage

```
diffFrame(dat, absolute = TRUE)
```

Arguments

- `dat` A data.frame of R x C dimensions, where R represents the specific pixel, ranging from 1 to w x h rows, and C represents the frame number.
- `absolute` If set to TRUE (default) the absolute difference between the value for each pixel is provided. If set to FALSE, it will return the true difference (negative/positive values).

Details

Providing a data frame of R x C dimensions, returns a data frame of R x (C-1) dimensions, where each column represents the difference between adjacent columns. Absolute or relative values are provided.

Each row in dat corresponds to a specific pixel position in a thermal video frame. Data frames are preferred over array functions for speed and simplicity. Row numbers range from 1 through to the image dimensions (i.e. w*h = 640 * 480=307200).

The premise behind this is that the thermal video is either time lapse or higher speed video. If a specific pixel shows no change (0) from frame to frame, then there is no movement or temperature change. For videos of living specimens, movement artefacts will manifest as change over time at specific pixels. If there is sufficient movement, across the image space, the accumulation of small differences will provide a measure of relative activity from frame to frame.

In combination of a cumulative summation function (cumulDiff), the diffFrame function can assess relative change in movement or activity. This makes use of a concept called the absolute difference sum method, sometimes used to simplify noisy data. See cumulDiff for further info.

Value

Returns a data frame of R x (C-1) dimensions, where each column represents the difference between adjacent columns.

Author(s)

Glenn J Tattersall

References

1. Lighton, J.R.B., and Turner, R.J. (2004). Thermolimit respirometry: an objective assessment of critical thermal maxima in two sympatric desert harvester ants, Pogonomyrmex rugosus and P. californicus. J Exp Biol 207: 1903-1913.

2. Lighton, J. R. B. (2008). Measuring metabolic rates : a manual for scientists. Oxford ; New York, Oxford University Press.

See Also

cumulDiff

Examples

```
# Create a Mandelbrot image sequence, stored in a data.frame
m = 640 # grid size
n = 480 # grid size
C = complex(real=rep(seq(-1.8,0.6, length.out=m), each=n),
imag=rep(seq(-1.2,1.2, length.out=m), n))
C = matrix(C,m,n)
Z = 0
Y = NULL

for (k in 1:20) {
  Z = Z^2+C
  ZZ<-matrix(exp(-abs(Z)), ncol=1)
  Y<-cbind(Y, ZZ)
}

# Calculate frame by frame difference
dY<-diffFrame(Y, absolute=TRUE)

# Plot the difference images
for (k in 1:19){
  x<-as.matrix(dY[,k], nrow=m)
  x[is.na(x)]<-min(x, na.rm=TRUE)
  plotTherm(x, w=m, h=n, minrangeset=min(x), maxrangeset=max(x))
  Sys.sleep(0.03)
}
```

`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

---

flirsettings *Extracts meta tag information from a FLIR JPG image*

Description

Extracts meta tag information from a FLIR JPG image using system installed Exiftool application. Use this to obtain thermal image calibration values, date/time stamps, object distance, and other parameters saved in FLIR image or video files.

Usage

```
flirsettings(imagefile, exiftoolpath = "installed", camvals = NULL)
```

Arguments

`imagefile`	Name of the FLIR JPG file to read from, as captured by the thermal camera. A character string.
`exiftoolpath`	A character string that determines whether Exiftool has been "installed" (http://www.sno.phy.queensu. or not. If Exiftool has been installed in a specific location, use to direct to the folder location.
`camvals`	A list of arguments to be passed to Exiftool as described in Exiftool documentation. A character string. Default value (recommended) is "", which will pass all possible arguments to Exiftool.

Details

The imagefile should be the original captured FLIR JPG file, not a modified JPG. This also works with FLIR video files (.seq and .fcf).

Exiftool should install on most operating systems. Consult with http://www.sno.phy.queensu.ca/~phil/exiftool/ for information on installing Exiftool. If trouble installing, download Exiftool perl scripts and set exiftoolpath to the custom folder locationto access the perl scripts that are attached with this package.

For camvals, provide a character string as described in Exiftool documentation. Set camvals="-*Emissivity", to simply return the Emissivity value. Set camvals="-*Planck*" for camera calibration constants.

Note: the Emissivity value is simply that which is stored in the file. It typically is the default value the camera is set to (0.95), but this does not mean that the true Emissivity of the surface is what is stored in the file. Similar caution is advised regarding the environmental parameters returned from the meta tags. User knowledge is required.

Value

Returns a list of camera meta tags for use in thermal imaging calculations.

Info is the basic list of camera settings.

Dates will be the date values associated with the image creation, modification etc.

Note

Requires Exiftool be installed. see http://www.sno.phy.queensu.ca/~phil/exiftool/

Author(s)

Glenn J Tattersall

References

1. http://www.sno.phy.queensu.ca/~phil/exiftool/ 2. http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/FLIR.html

Examples

```
# Example Use
# See https://github.com/gtatters/FLIRJPGConvert/blob/master/Examples.R
```

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)
```

`frameLocates` *Find the frame read start positions in a thermal video file.*

Description

Using readBin function, find everywhere in file vector where thermal resolution info is stored: i.e. 640x480, 320x240. These positions denote where the image frame data is found in the larger video file and will facilitate extraction.

Usage

```
frameLocates(vidfile = "", w = 640, h = 480, res2fram = 15)
```

Arguments

`vidfile`	Filename or filepath (as character) of the thermal video. Should end in .seq or .fcf. Not tested comprehensively so it may only work for certain camera models and software packages.
`w`	Width resolution (pixels) of thermal camera. Can be found by using the flirsettings function.
`h`	Height resolution (pixels) of thermal camera.
`res2fram`	# res2fram = frame data stream commences 15 bytes after where the resolution info (w,h) is found

Details

FLIR cameras have built-in radiometric video saving functions. FLIR software also has similar video, or time lapse, functionality. These files are typically stored as .seq or .fcf and encode information on the thermal imaging camera model, calibration, date/time, etc. These meta-tages can be extracted using system installed software (Exiftool).

This function makes use of the readBin function in the R base package, by loading a small portion of the file as two-byte integers in little endian order. It then searches through this data vector for the two-number sequence (w,h) which corresponds to meta-tags referring to the frame width and height which are typically stored 30 bytes ahead of the image pixel data. The actual start of the header information is empirically determined by the pattern of (w,h) locations within the file.

Frame refers to the still frame that is to be extracted from the thermal video file.

The function returns a list, containing the header start (h.start) position of each frame and the frame start (f.start) where pixel data is stored in raw, binary format (at present, in 16-Bit integers).

h.start and f.start can be passed to other functions to extract the precise times of each frame (getTimes) and to extract the actual frame by frame data (getFrames).

The length of h.start and f.start should be the same. If these are blank, then the detection process has not worked and the filetype might not be supported by this function.

Value

Returns a list, containing two vectors, h.start and f.start. These should be the same length.

`h.start`	A vector containing the read position start points in the file to extract header information from each frame. Typically passed to the getTimes function.
`f.start`	A vector containing the read position start points in the file to extract raw, binary pixel data from each frame. Typically passed to the getFrames function.

Note

Requires Exiftool be installed in order to automatically determine thermal image width and height. If you know the width and height in pixels, then the frame start locations can be determined.

For information on installing Exiftool, see http://www.sno.phy.queensu.ca/~phil/exiftool/

Author(s)

Glenn J Tattersall

References

1. http://www.sno.phy.queensu.ca/~phil/exiftool/ 2. http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/FLIR.html

See Also

getFrames, getTimes, readBin

Examples

```
x<-frameLocates(vidfile = system.file("extdata", "SampleSEQ.seq", package = "Thermimage"))
x$h.start
x$f.start
```

---

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)
```

`getFrames` *Extract raw binary thermal from thermal image file.*

Description

Extracts raw binary thermal image data in integer format as a vector

Usage

```
getFrames(vidfile, framestarts, w = 640, h = 480, l = w * h, byte.length = 2)
```

Arguments

`vidfile`	Filename or filepath (as character) of the thermal video. Should end in .seq or .fcf. Not tested comprehensively so it may only work for certain camera models and software packages.
`framestarts`	A vector of integers corresponding to the actual pixel read byte start positions in the thermal video file. Acquired using the getFrames function.
`w`	Width of thermal image.
`h`	Height of thermal image
`l`	The total size (length) of pixel data corresponding to one image = width * height. User does not need to set this.
`byte.length`	Set to 2 by default. Each pixel information is encoded in two bytes (i.e. 16 bit), leading to an integer value ranging from 1 to 2^16. Pixel data are read in order in the file and converted to integer using the readBin function. User does not need to set this.

Details

This function will load into memory the raw binary pixel data from the entire thermal video file. Data are stored linearly as read in using the readBin function, but the number of frames read in can be determined by dividing the length of the vector by (w*h). Depending on the size of the video, this can become quite large.

Frame data is stored as a vector to speed calculations. Thermal video files may exceed memory capacity of some systems, so processing as arrays or dataframes is generally avoided.

As written, this is a vectorised function, so will only load in one frame is used normally. To load multiple frames from the video file, use a for-loop (usually slow) or the apply function to import (faster processing) or parallel apply functions (best).

Value

Returns a vector of integers, each item corresponding to raw pixel value. With information on thermal image width and height, the specific image can be reconstructed. To be used in conjunction with raw2temp function which will convert this raw binary value into an estimated temperature.

Note

Requires Exiftool be installed in order to automatically determine thermal image width and height. If you know the width and height in pixels, then the frame start locations can be determined.

For information on installing Exiftool, see http://www.sno.phy.queensu.ca/~phil/exiftool/

Author(s)

Glenn J Tattersall

References

1. http://www.sno.phy.queensu.ca/~phil/exiftool/ 2. http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/FLIR.html

See Also

frameLocates, getTimes, readBin, raw2temp

Examples

```
# set w to 640 and h to 480

w<-640
h<-480
f<-system.file("extdata", "SampleSEQ.seq", package = "Thermimage")
x<-frameLocates(f)

# Slow approach:
system.time({
  alldata<-matrix(nrow=w*h, ncol=length(x$f.start))
  for(i in 1:length(x$f.start)) alldata[,i]<-getFrames(f, x$f.start[i], w, h)
})
dim(alldata)

# Faster approach
alldata<-NULL
system.time(alldata<-unlist(lapply(x$f.start, getFrames, vidfile=f, w=w, h=h)))
length(alldata)/(w*h)

# Parallel approach (will not be faster on small files)
library(parallel)
alldata<-NULL
# set mc.cores to higher number to use parallel processing
system.time(alldata<-unlist(mclapply(x$f.start, getFrames, vidfile=f, mc.cores=1)))
length(alldata)/(w*h) # number of frames in video
```

getTimes *Extracts time values as POSIX from binary imported thermal video file*

Description

Extracts time values for each image frame from a thermal camera video file (.seq or .fcf). For time lapse or video capture, computer time is stored for each image frame in 3 byte chunks, denoting msec, sec, and date information.

Usage

```
getTimes(vidfile, headstarts, timestart = 448, byte.length = 2)
```

Arguments

vidfile	Filename or filepath (as character) of the thermal video. Should end in .seq or .fcf. Not tested comprehensively so it may only work for certain camera models and software packages.
headstarts	A vector of integers corresponding to the header read byte start positions in the thermal video file. Acquired using the getFrames function. The header information is where the (width, height) image information, as well as information on the camera, calibration, time of image capture, etc...are stored.
timestart	Set to 448 by default. Once the header start location has been determined with the frameLocates function, the frame times were stored in 448 bytes into the header. The user does not need to set this.
byte.length	Set to 2 by default. Each pixel information is encoded in two bytes (i.e. 16 bit), leading to an integer value ranging from 1 to 2^16. Pixel data are read in order in the file and converted to integer using the readBin function. User does not need to set this.

Details

Somewhat empirically determined, but also information provided on the exiftool website below describes where time stamp information is stored in each file. This function concatentates the 3 time stamps corresponding to msec, sec, and date into one POSIX variable that gives the actual time each image was captured.

As written, this is a vectorised function, so to extract multiple frames of data (i.e. length(headstarts)>1), use a loop or the apply function as shown in the example below.

Extracted times are used in sumamrising information about the temperature profiles of the thermal videos and can be passed to the cumulDiff function.

Extracted times can also be used to verify the frame rate of the image capture in the video.

Has not been fully tested on file types from all cameras or thermal imaging software.

Value

Returns a vector of times (POSIXct) corresponding to the frame capture times as extracted from the thermal video file.

Author(s)

Glenn J Tattersall

References

1. http://www.sno.phy.queensu.ca/~phil/exiftool/
2. http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/FLIR.html
3. http://www.silisoftware.com/tools/date.php
4. http://www.sandersonforensics.com/forum/content.php?131-A-brief-history-of-time-stamps

See Also

getFrames, frameLocates, cumulDiff

Examples

```
f<-system.file("extdata", "SampleSEQ.seq", package = "Thermimage")
x<-frameLocates(f)
getTimes(f, x$h.start)

# only returns the first frame of data, must use lapply to get all frames
# POSIX type data do not play well with lists, so try the following:

# Using lapply
extract.times<-do.call("c", lapply(x$h.start, getTimes, vidfile=f))
extract.times

# Using parallel lapply:
library(parallel)
# set mc.cores to higher number to use parallel processing:
extract.times<-do.call("c", mclapply(x$h.start, getTimes, vidfile=f, byte.length=2,
timestart=448, mc.cores=1))
extract.times
```

---

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 = 25, 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

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

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 global and longwave incoming radiation for the Greenland ice-sheet. Global and Planetary Change. 9: 143-164.

See Also

Lw

Examples

```
# 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

```
# 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

```
# 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
  }


# 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)
# box()
# text(.5,.5,"Matrix",col="white")
# mf<-mirror.matrix(m)
# image(mf,axes=FALSE)
# box()
# text(.5,.5,"Mirror",col="white")
```

`nameleadzero` *Add leading zeros to character for easy sequential naming of filenames.*

Description

Returns a character with leading zeros according to the total number of filenames to be created. Useful when exporting multiple images arising from imported video data stored as a matrix or dataframe. By providing a base root name, the function will then add leading zeroes ahead of the number suffix (counter variable), according to the no.digits requested (i.e. Img0001.png, Img0002.png,...Img9999.png). Best used inside a loop exporting images.

Usage

```
nameleadzero(filenameroot = "Img", filetype = ".png", no.digits = 5, counter = 1)
```

Arguments

`filenameroot`	Prefix or root filename, supplied as a character vector.
`filetype`	The type of file to be saved, as a character. i.e. ".png", or ".csv".
`no.digits`	The total number of digits required for the suffix portion of the complete filename. Use 2 if numbers range from 1 to 99.
`counter`	The specific counter to add to the suffix. Typically counter is a number.

Details

Although this returns a single character value with leading zeros, it could be used in a loop to create a new, incremented file name (i.e. Img0001.png, Img0002.png, Img0003.png,... Img9999.png), or wrapped in an apply function:

Value

Returns a character value.

Author(s)

Glenn J Tattersall

Examples

```
# Using for-loop
prefix<-"Img_"
filetype<-".png"
no.digits<-2
for(i in 1:10){
  f.txt<-nameleadzero(prefix, filetype, no.digits, counter=i)
  print(f.txt)
}

# Using an apply function
x<-unlist(lapply(1:10, nameleadzero, filenameroot="Img_", filetype=".png", no.digits=2))
```

x

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 = 25, 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")
```

`plotTherm` *Plot thermal image data for visualisation purposes.*

Description

A quick way to plot and visualise thermal image data using the fields package image.plot function.

Usage

```
plotTherm(bindata, templookup = NULL, w, h, minrangeset = 20, maxrangeset = 40, trans="I",
main = NULL, thermal.palette = flirpal)
```

Arguments

`bindata`	An integer vector of raw binary thermal information (usually) extracted from a thermal video or image using the getFrames or readflirJPG functions to be

converted to temperature and summarised. Instead, this can be a vector of temperature values (numeric); if so, then templookup should be set to NULL or ignored.

`templookup` A vector of temperatures converted using the raw2temp function, corresponding to the conversion from raw binary thermal information to calibrated temperature estimates. Typically will be vector of numbers 2^16 long, for a 16-bit camera. Default is NULL, which assumes that dat has already been converted to temperature.

`w` Width resolution (pixels) of thermal camera. Can be found by using the flirsettings function.

`h` Height resolution (pixels) of thermal camera. Can be found by using the flirsettings function.

`minrangeset` The minimum temperature to scale the raster plot z (temperature) value to.

`maxrangeset` The maximum temperature to scale the raster plot z (temperature) value to.

`trans` Transformation to apply to image matrix. Default is I, the identity matrix, which will plot the image without transformation. Options are mirror.matrix, rotate90.matrix, rotate270.matrix, rotate180.matrix, flip.matrix.

`main` Title to plot on image. Default is NULL.

`thermal.palette`
Palette to use for the thermal image plot. Default is ironbowpal (FLIR standard prism palette). See examples in the palette.choose() function, or provide a custom palette.

Experience has shown that it is challenging to set the scale bar to align nicely with the rasterised image, so the user is left to explore the image.plot() function on their own. It may help to set the plot area size first to get nicely aligned image and scale bars. The following option has worked in testing: par(pin=c(6,4.5))

Details

This function is a simplified wrapper to call the image.plot function in the fields package. Not all options are implemented, but default ones are shown here.

Value

Provides a rasterised plot based on a vector of data from a thermal image file.

Author(s)

Glenn J Tattersall

References

Douglas Nychka, Reinhard Furrer, John Paige and Stephan Sain (2015). "fields: Tools for spatial data." doi: 10.5065/D6W957CT (URL: http://doi.org/10.5065/D6W957CT), R package version 8.10, <URL: www.image.ucar.edu/fields>.

Examples

```
# Create a Mandelbrot image sequence, stored in a data.frame
m = 640 # grid size
```

```
n = 480 # grid size
C = complex(real=rep(seq(-1.8,0.6, length.out=m), each=n ),
imag=rep(seq(-1.2,1.2, length.out=m),n))
C = matrix(C,m,n)
Z = 0
Y = NULL

for (k in 1:20) {
  Z = Z^2+C
  ZZ<-matrix(exp(-abs(Z)), ncol=1)
  Y<-cbind(Y, ZZ)
}

par(pin=c(6,4.5))
# Plot the difference images
for (k in 1:19){
  x<-as.matrix(Y[,k], nrow=m)
  x[is.na(x)]<-min(x, na.rm=TRUE)
  plotTherm(x, w=m, h=n, minrangeset=min(x), maxrangeset=max(x))
  Sys.sleep(0.03)
}

# set w to 640 and h to 480
w<-640
h<-480
f<-system.file("extdata", "SampleSEQ.seq", package = "Thermimage")
x<-frameLocates(f)
suppressWarnings(templookup<-raw2temp(1:65535))
alldata<-unlist(lapply(x$f.start, getFrames, vidfile=f, w=w, h=h))
alldata<-matrix(alldata, nrow=w*h, byrow=FALSE)
alltemperature<-templookup[alldata]
alltemperature<-unname(matrix(alltemperature, nrow=w*h, byrow=FALSE))

# Plot
plotTherm(alldata[,2], templookup=templookup, w=w, h=h,  minrangeset=min(alldata),
          maxrangeset=max(alldata))


# Plot all frames using binary data with templookup
x<-apply(alldata, 2, plotTherm, templookup=templookup, w=w, h=h, minrangeset=20, maxrangeset=40)

# Plot all frames using converted temperature data
x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
maxrangeset=max(alltemperature), thermal.palette=flirpal)

# Try other palettes:
#x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
#maxrangeset=max(alltemperature), thermal.palette=rainbowpal)

#x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
#maxrangeset=max(alltemperature), thermal.palette=midgreypal)

#x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
#maxrangeset=max(alltemperature), thermal.palette=midgreenpal)

#x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
#maxrangeset=max(alltemperature), thermal.palette=greyredpal)
```

```
#x<-apply(alltemperature, 2, plotTherm, w=w, h=h, minrangeset=min(alltemperature),
#maxrangeset=max(alltemperature), thermal.palette=hotironpal)
```

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

```
# 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 degrees C. 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"))

# Create a templookup vector - faster calculations when working with huge binary data files
# suppressWarnings remove the NaN warning that results from the low values falling outside the
# range of temperatures relevant

suppressWarnings(templookup<-raw2temp(raw=1:65535))
r<-floor(runif(10000000, 16000,25000)) # create a long vector of raw binary values

# calculate temperature using the lookup vector:
system.time(templookup[r]) # 0.109 seconds

# calculate temperature using the raw2temp function on the raw vector:
system.time(raw2temp(r)) # 0.248 seconds
```

---

readflirJPG *Reads an image from a FLIR JPG file into an integer array.*

Description

Reads an image from a FLIR JPG file into an integer matrix, w pixels wide x h pixels high, depending on image size.

Usage

```
readflirJPG(imagefile, exiftoolpath = "installed")
```

Arguments

`imagefile` Name of the FLIR JPG file to read from, as captured by the thermal camera. A character string.

`exiftoolpath` A character string that determines whether Exiftool has been "installed" (http://www.sno.phy.queensu. or not. If Exiftool has been installed in a specific location, use to direct to the folder location.

Details

Only tested on a select number of FLIR JPGs. Usage depends on functionality provided by Exiftool.

Exiftool should install on most operating systems. Consult with http://www.sno.phy.queensu.ca/~phil/exiftool/ for information on installing Exiftool. If trouble installing, download Exiftool perl scripts and set exiftoolpath to the custom folder locationto access the perl scripts that are attached with this package. The version of Exiftool installed with Thermimage package has not been tested on all OS.

v 2.2.3: updated to fix a problem calling shell commands requiring folder write access on a windows OS (thanks to John Al-Alawneh)

Value

Returns a matrix of integer values, corresponding the calibrated raw thermal image radiance values. Can be converted to temperature estimates using the raw2temp() function.

Author(s)

Glenn J Tattersall

References

1. Exiftool Command line tool: http://www.sno.phy.queensu.ca/~phil/exiftool/

2. Simon Urbanek (2013). tiff: Read and write TIFF images. R package version 0.1-5. https://CRAN.R-project.org/package=tiff

3. Simon Urbanek (2013). png: Read and write PNG images. R package version 0.1-7. https://CRAN.R-project.org/package=png

See Also

`temp2raw` `raw2temp`

Examples

```
# Examples
# See https://github.com/gtatters/FLIRJPGConvert/blob/master/Examples.R
```

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
  }


# 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)
# box()
# text(.5,.5,"Matrix")
# mf<-rotate180.matrix(m)
```

```
# image(mf,axes=FALSE)
# 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))
  }



 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)
box()
text(.5,.5,"Matrix",col="white")
mf<-rotate270.matrix(m)
image(mf,axes=FALSE)
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))
  }

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)
box()
text(.5,.5,"Matrix",col="white")
mf<-rotate90.matrix(m)
image(mf,axes=FALSE)
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

The 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

```
# 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"))
```

`Teq` *Estimates equivalent temperature.*

Description

Estimates equivalent black-body temperature of an object. Analagous to other measures of operative temperature

Usage

```
Teq(Ts = 30, Ta = 25, Tg = NULL, RH = 0.5, E = 0.96, rho = 0.1, cloud = 0, SE = 0, V = 1,
L = 0.1, type = "forced")
```

Arguments

`Ts`	Surface temperature (degrees Celsius). Default value is 30. Not used in this calculation but kept for similar structure to other functions in package.
`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. Used in estimating long wave radiation from the ground.
`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.
`type`	"forced" or "free" - to calculate convection coefficient for either forced or free convection. Default value is "forced"

Author(s)

Glenn J Tattersall

References

Mahoney, S.A. and King, J. R. (1977). The use of the equivalent black-body temperautre in the thermal energetics of small birds. J Thermal Biol. 2: 115-120

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, type = "forced")
{
    if (type == "forced")
        k <- 0.7 * 310
    if (type == "free")
        k <- 310
    rr <- airdensity(Ta) * airspecificheat(Ta)/(4 * E * StephBoltz() *
        (Ta + 273.15)^3)
    ra <- k * (L/V)^0.5
    re <- 1/(1/ra + 1/rr)
    Rni <- qabs(Ta = Ta, Tg = Tg, RH = RH, E = E, rho = rho,
        cloud = cloud, SE = SE) - StephBoltz() * E * (Ta + 273.15)^4
    Teq <- Ta + Rni * re/(airdensity(Ta) * airspecificheat(Ta))
    Teq
  }
```

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

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

`thermsum` *Return summary of thermal image data.*

Description

Provides typical summary data (min, max, mean, sd, median) of a vector of raw binary thermal encoded data. If templookup is not provided, the summary info is conducted on the data provided. If a templookup vector is provided (see Examples in raw2temp function), the dat values are converted to temperature before summary information is extracted.

Usage

```
thermsum(dat, templookup = NULL)
```

Arguments

`dat` An integer vector of raw binary thermal information (usually) extracted from a thermal video or image using the getFrames or readflirJPG functions to be converted to temperature and summarised. Instead, this can be a vector of temperature values (numeric); if so, then templookup should be set to NULL or ignored.

`templookup` A vector of temperatures converted using the raw2temp function, corresponding to the conversion from raw binary thermal information to calibrated temperature estimates. Typically will be vector of numbers 2^16 long, for a 16-bit camera. Default is NULL, which assumes that dat has already been converted to temperature.

Details

A simple summary function for thermal imaging data to allow for extraction of basic statistical data from a thermal image dataset. If dat is supplied as an integer vector of raw binary values, then templookup should be supplied to use as an indexing function.

Using raw2temp(1:65535) will produce a vector of temperatures that correspond to the indexed integers 1:65535. This method of calculation can be faster on large video files. The default settings for raw2temp() will not be appropriate, and all camera settings should be used according to calibration constants.

If dat is supplied as a vector of temperatures, then templookup must be left blank or NULL as the default. Summary information will be calculated on the dat variable assuming it is properly calibrated temperature values.

As written, this is a vectorised function, so will only calculate summary on the vector provided. To perform thermal summaries on multiple frames from the raw binary video data, use a for-loop (usually slow) or the apply function to process (faster processing) or parallel apply functions (best).

Value

Returns a named vector: Mintemp, Maxtemp, Meantemp, SDtemp, and Mediantemp

Warning

This function simply calculates summary data, and does not detect objects in the image frame. Use only as rapid way to extract thermal information. This is not a replacement for doing analysis by hand, and may only be useful for objects that are stationary and remain within the image frame over time.

Author(s)

Glenn J Tattersall

See Also

raw2temp, thermsumcent

Examples

```
# set w to 640 and h to 480
w<-640
h<-480
f<-system.file("extdata", "SampleSEQ.seq", package = "Thermimage")
x<-frameLocates(f)
suppressWarnings(templookup<-raw2temp(1:65535))
alldata<-unlist(lapply(x$f.start, getFrames, vidfile=f, w=w, h=h))
alldata<-matrix(alldata, nrow=w*h, byrow=TRUE)

# Summary on one image or frame of data
thermsum(alldata[,1], templookup)

# Summary on multi-frame seq file
tsum<-data.frame(t(apply(alldata, 2, thermsum, templookup)))
tsum

# Randomly generated data
alldata<-floor(runif(w*h*40, 17000, 25000))
alldata<-matrix(alldata, nrow=w*h)

# depending on the size of alldata, directly calculating temperature can slow down processing
# For a 40 frame file:
system.time(alltemperature<-raw2temp(alldata))

# But summary calculations using raw binary with lookup are slightly slower than
# using numeric temperatures:

# Perform calculations on the raw binary but supply the templookup vector
system.time(tsum<-data.frame(t(apply(alldata, 2, thermsum, templookup))))

# Perform calculations on the converted temperature values
system.time(tsum<-data.frame(t(apply(alltemperature, 2, thermsum))))
tsum
```

`thermsumcent` *Summary thermal calculations on a centrall located region of interest from a thermal image dataset*

Description

Similarly to the thermsum except this provides thermal summary data on a central region of interest, commonly used in thermal imaging. The size of the region is a rectangular region corresponding to a fraction of the total image area set by boxsize.

Usage

```
thermsumcent(dat, templookup = NULL, w = 640, h = 480, boxsize = 0.05)
```

Arguments

`dat`	An integer vector of raw binary thermal information (usually) extracted from a thermal video or image using the getFrames or readflirJPG functions to be converted to temperature and summarised. Instead, this can be a vector of temperature values (numeric); if so, then templookup should be set to NULL or ignored.
`templookup`	A vector of temperatures converted using the raw2temp function, corresponding to the conversion from raw binary thermal information to calibrated temperature estimates. Typically will be vector of numbers 2^16 long, for a 16-bit camera. Default is NULL, which assumes that dat has already been converted to temperature.
`w`	Width resolution (pixels) of thermal camera. Can be found by using the flirsettings function.
`h`	Height resolution (pixels) of thermal camera. Can be found by using the flirsettings function.
`boxsize`	Fractional area of the desired rectangular region of interest. Default is set to 0.05. Dimensions of the region will depend on w and h dimensions.

Details

A simple summary function for thermal imaging data to allow for extraction of basic statistical data from a thermal image dataset. If dat is supplied as an integer vector of raw binary values, then templookup should be supplied to use as an indexing function.

Using raw2temp(1:65535) will produce a vector of temperatures that correspond to the indexed integers 1:65535. This method of calculation can be faster on large video files. The default settings for raw2temp() will not be appropriate, and all camera settings should be used according to calibration constants.

If dat is supplied as a vector of temperatures, then templookup must be left blank or NULL as the default. Summary information will be calculated on the dat variable assuming it is properly calibrated temperature values.

As written, this is a vectorised function, so will only calculate summary on the vector provided. To perform thermal summaries on multiple frames from the raw binary video data, use a for-loop (usually slow) or the apply function to process (faster processing) or parallel apply functions (best).

Similar to thermsum, except this assesses only the centrally located region of interest in the image frame centre.

Value

Returns a named vector: CentrePoint, CentreBoxMin, CentreBoxMax, CentreBoxMean, CentreBoxSD, CentreBoxMedian)

Warning

This function simply calculates summary data, and does not detect objects in the image frame. Use only as rapid way to extract thermal information. This is not a replacement for doing analysis by hand, and may only be useful for objects that are stationary and remain within the image frame over time.

Author(s)

Glenn J Tattersall

See Also

raw2temp, thermsum

Examples

```
# set w to 640 and h to 480

w<-640
h<-480
f<-system.file("extdata", "SampleSEQ.seq", package = "Thermimage")
x<-frameLocates(f)
suppressWarnings(templookup<-raw2temp(1:65535))
alldata<-unlist(lapply(x$f.start, getFrames, vidfile=f, w=w, h=h))
alldata<-matrix(alldata, nrow=w*h, byrow=TRUE)

# Summary on one image or frame of data
thermsumcent(alldata[,1], templookup)

# Summary on multi-frame seq file
tsum<-data.frame(t(apply(alldata, 2, thermsumcent, templookup)))
tsum

# Randomly generated data
alldata<-floor(runif(w*h*50, 17000, 25000))
alldata<-matrix(alldata, nrow=w*h)

# depending on the size of alldata, directly calculating temperature can slow down processing
# For a 50 frame file:
system.time(alltemperature<-raw2temp(alldata))

# But summary calculations using raw binary with lookup are slightly slower than
# using numeric temperatures:

# Perform calculations on the raw binary but supply the templookup vector
system.time(tsum<-data.frame(t(apply(alldata, 2, thermsumcent, templookup))))

# Perform calculations on the converted temperature values
system.time(tsum<-data.frame(t(apply(alltemperature, 2, thermsumcent))))
tsum
```

`writeFlirBin` *Saves thermal image data to a binary file*

Description

Saves thermal image data to a binary file. This function serves to allow thermal images that have been imported into R to be exported to a raw, 32-bit real format that can then be imported and analysed in ImageJ.

Usage

```
writeFlirBin(bindata, templookup, w, h, Interval, rootname)
```

Arguments

`bindata`	Vector of raw binary data imported from a thermal image file, using the getFrames function. Each value corresponds to the raw binary sensor value for each pixel. Should be supplied as a vector, not a dataframe or matrix.
`templookup`	A vector of values from 1:65535 (2^16) that serves as a rapid means to convert the above bindata into calibrated temperature data for each pixel. This makes use of the raw2temp function. This value must be supplied and properly calibrated, otherwise the conversion will not be correct. Default is set to NULL. If calibrated temperature data is supplised as bindata, then templookup should be set to NULL.
`w`	Width resolution (pixels) of thermal camera. Can be found by using the flirsettings function.
`h`	Height resolution (pixels) of thermal camera. Can be found by using the flirsettings function.
`Interval`	Time inverval (in seconds = 1 / Frame rate) of the thermal video file. Used for encoding in filename.
`rootname`	Root name (character) for saving the binary file

Details

This function exports raw binary information from the getFrames function in a 32-bit real file format (4 bytes). This file format can be relatively easily imported into ImageJ using the Import-Raw option, choose 32-bit Real, set your image width and height and # of frames. Little endian and hyperstack options must be enabled during import.

The file naming takes the rootname and appends image width, height, number of frames, and image interval, appending .raw to the end to make ImageJ import easier.

If rootname = 'Thermvid', w=640, h=480, number of frames=100, and image interval is 0.0333 seconds, the file name will be saved as:

'Thermvid_W640_H480_F100_I0.0333.raw'

Value

Returns nothing, but saves a new file to the current working director.

Warning

This function has not been fully tested with all possible video/camera combinations. Users are advised to compare the exported values in ImageJ on sample images to standard FLIR software values before proceeding with analysis.

Author(s)

Glenn J Tattersall

See Also

raw2temp, getFrames, readBin, writeBin

Examples

```
bindata<-floor(runif(307200, 17000, 25000))
templookup<-raw2temp(bindata)
w<-640
h<-480
Interval<-0.03
f.root<-"Thermalvid"

# Usage:
# writeFlirBin(bindata, templookup=templookup,  w=w, h=h, Interval=Interval, rootname=f.root)
```

---

yellowpal *Colour palette extracted from FLIR thermal camera files*

Description

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

Index

∗Topic **\textasciitildekwd1**
nameleadzero, 36
thermsumcent, 70
∗Topic **\textasciitildekwd2**
nameleadzero, 36
thermsumcent, 70
∗Topic **datasets**
flirpal, 14
glowbowpal, 24
grey10pal, 26
grey120pal, 26
greyredpal, 26
hotironpal, 28
ironbowpal, 28
medicalpal, 33
midgreenpal, 34
midgreypal, 34
mikronprismpal, 34
mikroscanpal, 34
rainbowpal, 49
samp.image, 58
yellowpal, 73

airdensity, 3
airspecificheat, 4
airconductivity, 5
airviscosity, *5*, 6, *25*
areacone, 7
areacylinder, 8
areasphere, 9

cumulDiff, 10, *12*, *24*

diffFrame, *11*, 11

flip.matrix, 13, *35*, *55–57*
flirpal, 14
flirsettings, 14
forcedparameters, 16, *20*
frameLocates, 17, *22*, *24*
freeparameters, *16*, 19

getFrames, *19*, 21, *24*, *73*
getTimes, *19*, *22*, 23
glowbowpal, 24
Grashof, 25
grey10pal, 26
grey120pal, 26
greyredpal, 26

hconv, 26, *47*, *63*
hotironpal, 28

ironbowpal, 28

Ld, 28, *31*, *32*, *44*, *49*
lm, *59*
locate.fid, 29
Lu, 30, *44*, *49*
Lw, *29*, 31

match, *30*
meanEveryN, 32
medicalpal, 33
midgreenpal, 34
midgreypal, 34
mikronprismpal, 34
mikroscanpal, 34
mirror.matrix, *14*, 35, *55–57*

nameleadzero, 36
Nusseltforced, *16*, 37
Nusseltfree, *20*, 38

palette.choose, 39
plotTherm, 40
Prandtl, 43

qabs, 44, *49*, *63*
qcond, 45
qconv, *27*, *46*, 46
qrad, *44*, *46*, 48

rainbowpal, 49
raw2temp, *22*, 50, *53*, *65*, *69*, *71*, *73*
readBin, *19*, *22*, *73*
readflirJPG, 52
Reynolds, 54
rotate180.matrix, *14*, *35*, 55, *56*, *57*
rotate270.matrix, *14*, *35*, *55*, 56, *57*

rotate90.matrix, *14*, *35*, *55*, *56*, 57

samp.image, 58
slopebypoint, 59, *60*
slopeEveryN, *33*, 60
StephBoltz, 61

Te, 62
temp2raw, *51*, *53*, 63
Teq, 66
Tground, 67
Thermimage *(*Thermimage-package*)*, 3
Thermimage-package, 3
thermsum, 68, *71*
thermsumcent, *69*, 70

which, *30*
writeBin, *73*
writeFlirBin, 72

yellowpal, 73