

# Package ‘rLakeAnalyzer’

May 9, 2026

**Title** Lake Physics Tools

**Maintainer** Luke Winslow <lawinslow@gmail.com>

**Version** 1.11.4.1

**Author** Luke Winslow, Jordan Read, Richard Woolway, Jennifer Brentrup, Taylor Leach, Jake Zwart, Sam Albers, Doug Collinge

**Description** Standardized methods for calculating common important derived physical features of lakes including water density based based on temperature, thermal layers, thermocline depth, lake number, Wedderburn number, Schmidt stability and others.

**Depends** R (>= 2.10)

**Imports** plyr, stats, graphics, utils, grDevices

**Suggests** testthat, knitr, rmarkdown

**License** GPL (>= 2)

**Repository** CRAN

**BugReports** <https://github.com/GLEON/rLakeAnalyzer/issues>

**Date/Publication** 2019-06-09 18:45:46 UTC

**RoxygenNote** 6.0.1

**VignetteBuilder** knitr

**NeedsCompilation** no

## Contents

approx.bathy . . . . .	2
buoyancy.freq . . . . .	4
center.buoyancy . . . . .	5
depth.filter . . . . .	6
drop.datetime . . . . .	6
epi.temperature . . . . .	7
get.datetime . . . . .	7
get.offsets . . . . .	8
hypo.temperature . . . . .	9

internal.energy . . . . .	9
lake.number . . . . .	10
lake.number.plot . . . . .	11
latesummer . . . . .	12
layer.density . . . . .	13
layer.temperature . . . . .	14
load.bathy . . . . .	15
load.ts . . . . .	15
meta.depths . . . . .	16
rLakeAnalyzer . . . . .	17
schmidt.plot . . . . .	18
schmidt.stability . . . . .	19
thermo.depth . . . . .	20
ts.buoyancy.freq . . . . .	21
ts.center.buoyancy . . . . .	22
ts.internal.energy . . . . .	23
ts.lake.number . . . . .	24
ts.layer.temperature . . . . .	25
ts.meta.depths . . . . .	26
ts.schmidt.stability . . . . .	27
ts.thermo.depth . . . . .	28
ts.uStar . . . . .	29
ts.wedderburn.number . . . . .	30
uStar . . . . .	31
water.density . . . . .	32
wedderburn.number . . . . .	33
whole.lake.temperature . . . . .	34
wtr.heat.map . . . . .	35
wtr.heatmap.layers . . . . .	35
wtr.layer . . . . .	36
wtr.lineseries . . . . .	37
wtr.plot.temp . . . . .	38

<b>Index</b>	<b>40</b>
--------------	-----------

---

approx.bathy	<i>Estimate hypsography curve</i>
--------------	-----------------------------------

---

## Description

Estimates a depth-area curve for a lake using lake surface area, maximum depth and mean depth. Two methods for estimating the curve are available; 'cone' assumes the lake is shaped as a cone and requires only surface area and maximum depth; "voldev" uses the volume development (Vd) parameter from Hakanson (1981) and Johansson et al. (2007). Vd is a dimensionless parameter that describes lake basin shape in relation to the volume of cone whose base area and height equal the surface area and maximum lake depth, it is estimated as  $Vd = Z_{mean}/Z_{max}$  (Hakanson et al. 2000). Method "voldev" requires lake surface area, mean and maximum depth. Depths at which the area is estimated can be set by as a numeric vector or as a regularly spaced sequence.

**Usage**

```
approx.bathy(Zmax, lkeArea, Zmean = NULL, method = "cone", zinterval = 1,
  depths = seq(0, Zmax, by = zinterval))
```

**Arguments**

Zmax	a single value of the maximum depth of the lake (in m)
lkeArea	a single value of the surface area of the lake (in m <sup>2</sup> )
Zmean	a single value of the mean depth of the lake (in m)
method	specifies the method used to estimate depth-area relationship, can be "cone"(default) or "voldev". Method "voldev" requires Zmean. See notes for details.
zinterval	a single value defining the depth interval at which volumes should be calculated, default is 1 m.
depths	a numeric vector of depths (in m) at which areas are estimated. If not specified depths is regularly spaced sequence of values with the interval set by zinterval.

**Value**

a dataframe which defines the lake area for each depth. Columns are depths (m) and Area.at.z (m<sup>2</sup>). Area at 0 m should equal the user entered lkeArea.

**References**

Hakanson, L. (1981). On lake bottom dynamics - the energy- topography factor. Canadian Journal of Earth Sciences, 18, 899-909. Johansson, H., A. A. Brolin, and L. Hakanson. 2007. New approaches to the modelling of lake basin morphometry. Environ. Model. Assess. 12: 213-228.

**Examples**

```
Voldev.ex = approx.bathy(Zmax = 25, Zmean = 12, lkeArea = 39400000, method = "voldev")
Voldevshallow.ex = approx.bathy(Zmax = 25, Zmean = 6, lkeArea = 39400000, method = "voldev")
Cone.ex = approx.bathy(Zmax = 25, lkeArea = 39400000, method = "cone")

# plot depth-area curves
plot(Cone.ex$depths ~ Cone.ex$Area.at.z, xlab = "Area (m^3)", ylab = "Depth (m)")
points(Voldev.ex$depths ~ Voldev.ex$Area.at.z, col = "red")
points(Voldevshallow.ex$depths ~ Voldevshallow.ex$Area.at.z, col = "blue")
```

---

buoyancy.freq                      *Calculates buoyancy frequency.*

---

### Description

Calculate the buoyancy frequency (Brunt-Vaisala frequency) for a temperature profile.

### Usage

```
buoyancy.freq(wtr, depths)
```

### Arguments

wtr                      a numeric vector of water temperature in degrees C  
depths                   a numeric vector corresponding to the depths (in m) of the wtr measurements

### Value

Returns a vector of buoyancy frequency in units  $\text{sec}^{-2}$ . Return value has attribute "depths" which define buoyancy frequency depths (which differ from supplied depths).

### See Also

thermo.depth, ts.buoyancy.freq

### Examples

```
# A vector of water temperatures
wtr = c(22.51, 22.42, 22.4, 22.4, 22.4, 22.36, 22.3, 22.21, 22.11, 21.23, 16.42,
15.15, 14.24, 13.35, 10.94, 10.43, 10.36, 9.94, 9.45, 9.1, 8.91, 8.58, 8.43)

#A vector defining the depths
depths = c(0, 0.5, 1, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20)

b.f = buoyancy.freq(wtr, depths)

plot(b.f, attr(b.f, 'depths'), type='b',
ylab='Depth', xlab='Buoyancy Frequency', ylim=c(max(depths), min(depths)))
```

---

center.buoyancy	<i>Calculates the center of buoyancy.</i>
-----------------	---

---

### Description

Calculate the center of buoyancy using buoyancy frequency with a center of mass analysis. Brunt-Vaisala frequency is used for a temperature profile. Negative values for N2 are set to 0 (as they represent transient instabilities or sensor calibration issues) for this calculation.

### Usage

```
center.buoyancy(wtr, depths)
```

### Arguments

wtr	a numeric vector of water temperature in degrees C
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements

### Value

Returns a value for the center of buoyancy.

### See Also

buoyancy.freq, ts.buoyancy.freq, center.buoyancy

### Examples

```
# A vector of water temperatures
wtr = c(22.51, 22.42, 22.4, 22.4, 22.4, 22.36, 22.3, 22.21, 22.11, 21.23, 16.42,
15.15, 14.24, 13.35, 10.94, 10.43, 10.36, 9.94, 9.45, 9.1, 8.91, 8.58, 8.43)

#A vector defining the depths
depths = c(0, 0.5, 1, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20)

c.b = center.buoyancy(wtr, depths)
```

---

depth.filter	<i>Data filter to remove soak, heave and upcast</i>
--------------	---

---

### Description

- Soak period: water profiling instruments typically require a soak period where you let the instrument rest submerged at the surface. While it is "soaking" it is collecting data. We don't want that data
- Upcast versus downcast: typically instruments are turned on before you put them in the water and turn them off once you pull them out. The data consequence of that is that you collect both the "downcast" and the "upcast". In some case the upcast is of interest but usually it isn't. And because we would prefer increasing depth data it is better to remove an upcast if it is present.
- Heave: when lowering the instrument in rough weather a boat will heave side to side. Sometimes it will heave enough that you get small data groupings where the decreases a little while the boat heaves then go down. The overall trend is still down but those slight upticks in depth cause problems for our algorithm.

### Usage

```
depth.filter(z0, run_length = 20, index = FALSE)
```

### Arguments

z0	depth vector
run_length	Length of run upon which to start the soak removal
index	Logical: Should the function return an index value or actual value?

### Value

index values of z0 of filtered data. Will return a warning if the function removed more than 10

---

drop.datetime	<i>Find and drop the datetime column from the datatable</i>
---------------	---

---

### Description

Liberally looks for a datetime column and drops it, returning a data.frame with only water temperature. Errors if datetime column is ambiguous. Warns if there is no match.

### Usage

```
drop.datetime(data, error = FALSE)
```

**Arguments**

data            data arg  
 error           defaults to FALSE

**Value**

A data.frame with only the data, after datetime has been dropped

---

*epi.temperature*            *Get volumetrically averaged epilimnion temp*

---

**Description**

Calculates volumetrically weighted average epilimnetic temperature using the supplied water temperature timeseries. If the lake is not stratified, the bottom of the epilimnion is calculated as the full depth of the lake.

**Usage**

`epi.temperature(wtr, depths, bthA, bthD)`

**Arguments**

wtr            a numeric vector of water temperature in degrees C.  
 depths        a numeric vector corresponding to the depths (in m) of the wtr measurements  
 bthA          a numeric vector of cross sectional areas (m<sup>2</sup>) corresponding to bthD depths  
 bthD          a numeric vector of depths (m) which correspond to areal measures in bthA

**See Also**

[hypo.temperature](#) [whole.lake.temperature](#)

---

*get.datetime*            *Search for and return the datetime column from a ts data.frame*

---

**Description**

Warns if unavailable then returns NULL.

**Usage**

`get.datetime(data, error = FALSE)`

**Arguments**

data            data arg  
 error           defaults to FALSE

---

`get.offsets`*Gets depths from data frame containing profile info.*

---

### Description

Extracts the depth information from a data frame containing multi-depth observation data. Relies on the format of the header to get information and may fail if your file format is incorrect. Please follow 'VAR\_##.#' format, where ##.# is the depth of data for that column. VAR is typically 'wtr' to indicate water temperature.

### Usage

```
get.offsets(data)
```

### Arguments

`data` Data frame returned from [load.ts](#).

### Value

A numeric vector of depth values. Should be the `ncol(data) - 1` in length as the first column contains date/time data.

### See Also

[load.ts](#)

### Examples

```
#Get the path for the package example file included
exampleFilePath <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")

#Load
sparkling.temp = load.ts(exampleFilePath)

#get the lake depths associated with each column
depths = get.offsets(sparkling.temp)

print(depths)
```

---

hypo.temperature	<i>Get volumetrically averaged hypolimnion temp</i>
------------------	---

---

**Description**

Calculates volumetrically weighted average hypolimnetic temperature using the supplied water temperature timeseries. If the lake is not stratified, an NA value is returned.

**Usage**

```
hypo.temperature(wtr, depths, bthA, bthD)
```

**Arguments**

wtr	a numeric vector of water temperature in degrees C.
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
bthA	a numeric vector of cross sectional areas (m <sup>2</sup> ) corresponding to bthD depths
bthD	a numeric vector of depths (m) which correspond to areal measures in bthA

**See Also**

[epi.temperature](#), [whole.lake.temperature](#)

---

internal.energy	<i>Internal energy function (Joules)</i>
-----------------	--

---

**Description**

Calculates the internal energy of the water column with temperature and hypsography

Internal energy is the thermal energy in the water column, which is calculated by multiplying the specific heat of water (J kg<sup>-1</sup> K<sup>-1</sup>) by the temperature and mass of the water in the lake.

**Usage**

```
internal.energy(wtr, depths, bthA, bthD)
```

**Arguments**

wtr	a numeric vector of water temperature in degrees C
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
bthA	a numeric vector of cross sectional areas (m <sup>2</sup> ) corresponding to bthD depths
bthD	a numeric vector of depths (m) which correspond to areal measures in bthA

**Value**

internal energy in Joules m<sup>-2</sup>. (Currently not vectorized..)

**Examples**

```
bthA <-c(1000,900,864,820,200,10)
bthD <-c(0,2.3,2.5,4.2,5.8,7)

wtr <-c(28,27,26.4,26,25.4,24,23.3)
depths <-c(0,1,2,3,4,5,6)

cat('Internal Energy for input is: ')
cat(internal.energy(wtr, depths, bthA, bthD))
```

---

lake.number

*Calculate Lake Number*

---

**Description**

The Lake Number, defined by Imberger and Patterson (1990), has been used to describe processes relevant to the internal mixing of lakes induced by wind forcings. Lower values of Lake Number represent a higher potential for increased diapycnal mixing, which increases the vertical flux of mass and energy across the metalimnion through the action of non-linear internal waves. Lake Number is a dimensionless index.

Lake Number has been used, for example, to estimate the flux of oxygen across the thermocline in a small lake (Robertson and Imberger, 1994), and to explain the magnitude of the vertical flux of ammonium in a lake (Romero et al., 1998).

**Usage**

```
lake.number(bthA, bthD, uStar, St, metaT, metaB, averageHypoDense)
```

**Arguments**

bthA	a numeric vector of cross sectional areas (m <sup>2</sup> ) corresponding to bthD depths, hypsographic areas
bthD	a numeric vector of depths (m) which correspond to areal measures in bthA, hypsographic depths
uStar	a numeric array of u* (m/s), water friction velocity due to wind stress
St	a numeric array of Schmidt stability (J/m <sup>2</sup> ), as defined by Idso 1973
metaT	a numeric array of the top of the metalimnion depth (m from the surface)
metaB	a numeric array of the bottom of the metalimnion depth (m from the surface)
averageHypoDense	a numeric array of the average density of the hypolimnion (kg/m <sup>3</sup> )

**Value**

A numeric vector of Lake Number [dimensionless]

**References**

Imberger, J., Patterson, J.C., 1990. *Physical limnology*. Advances in Applied Mechanics 27, 303-475.

Idso, S.B., 1973. *On the concept of lake stability*. Limnology and Oceanography 18, 681-683.

**See Also**

[ts.lake.number](#) [wedderburn.number](#)

**Examples**

```
bthA <-c(1000,900,864,820,200,10)
bthD <-c(0,2.3,2.5,4.2,5.8,7)
uStar <-c(0.0032,0.0024)
St <-c(140,153)
metaT <-c(1.34,1.54)
metaB <-c(4.32,4.33)
averageHypoDense <-c(999.3,999.32)
cat('Lake Number for input vector is: ')
cat(lake.number( bthA, bthD, uStar, St, metaT, metaB, averageHypoDense ) )
```

---

lake.number.plot	<i>Plots time series of Lake Number</i>
------------------	---

---

**Description**

Generates a time series plot of Lake Number for appropriately formatted data. See [lake.number](#) for more details on Lake Number and reference.

**Usage**

```
lake.number.plot(wtr, wnd, wh, bth)
```

**Arguments**

wtr	Data frame of water temperature loaded with <a href="#">load.ts</a>
wnd	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
wh	A value indicating the height of the anemometer above lake surface in meters. This value must be specified, there is no default.
bth	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>

**See Also**[wtr.lineseries](#)**Examples**

```
#Get system data file paths
wtr.path <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")
bth.path <- system.file('extdata', 'Sparkling.bth', package="rLakeAnalyzer")
wnd.path <- system.file('extdata', 'Sparkling.wnd', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng Lake, Wisconsin.
wtr = load.ts(wtr.path)
wnd = load.ts(wnd.path)
bth = load.bathy(bth.path)
wh = 1 # user specified, here as 1 m.
## Not run:
#generate default plot
lake.number.plot(wtr,wnd,wh,bth)

## End(Not run)
```

---

latesummer

*Late Summer Profile*

---

**Description**

Late summer water profile taken from Quesnel Lake, British Columbia, Canada. Profile taken with Sea-Bird SBE19plus.

**depth** Depth, m

**temper** Temperature, degC

**salinity** Salinity, PSU

**oxygen** Oxygen, ml/l

**oxygen.sat** Oxygen saturation, percent saturation

**density** Density, kg/m<sup>3</sup> ...

**Usage**

```
latesummer
```

**Format**

An object of class `data.frame` with 881 rows and 6 columns.

---

layer.density	<i>Returns the average density of a layer between two depths.</i>
---------------	---

---

### Description

This function calculates the average density of a layer of water between two depths.

### Usage

```
layer.density(top, bottom, wtr, depths, bthA, bthD, sal = wtr * 0)
```

### Arguments

top	Numeric value of the depth (m) of the top of the layer from the water surface
bottom	Numeric value of the depth (m) of the bottom of the layer from the water surface
wtr	Numeric vector of water temperature in degrees C
depths	Numeric vector of depths (m) corresponding to water temperature vector
bthA	Numeric vector of water body cross sectional area (m <sup>2</sup> ) corresponding to bthD depths
bthD	Numeric vector of water body bathymetric depths (m) corresponding to areal bthA values
sal	Optional numeric vector of salinity in Practical Salinity Units corresponding to water temperature vector. If left blank, salinity is set to be zero

### Value

Numeric value of average water density for bounded layer in kg/m<sup>3</sup>

### See Also

water.density

### Examples

```
top <- 2
bottom <- 6
wtr <- c(25.2, 25.1, 24.1, 22.0, 19.8, 15.3, 12.0, 11.1)
depths <- c(0, 1, 2, 3, 4, 5, 6, 7)
bthA <- c(10000, 8900, 5000, 3500, 2000, 1000, 300, 10)
bthD <- c(0, 1, 2, 3, 4, 5, 6, 7)
layer.density(top, bottom, wtr, depths, bthA, bthD)
```

---

layer.temperature      *Returns the average temperature of a layer between two depths.*

---

### Description

This function calculates the average temperature of a layer of water between two depths.

### Usage

```
layer.temperature(top, bottom, wtr, depths, bthA, bthD)
```

### Arguments

top	Numeric value of the depth (m) of the top of the layer from the water surface
bottom	Numeric value of the depth (m) of the bottom of the layer from the water surface
wtr	Numeric vector of water temperature in degrees C
depths	Numeric vector of depths (m) corresponding to water temperature vector
bthA	Numeric vector of water body cross sectional area (m <sup>2</sup> ) corresponding to bthD depths
bthD	Numeric vector of water body bathymetric depths (m) corresponding to areal bthA values

### Value

Numeric value of average water temperature

### See Also

layer.density

### Examples

```
# Supply input data
top <- 2
bottom <- 6
wtr <- c(25.2, 25.1, 24.1, 22.0, 19.8, 15.3, 12.0, 11.1)
depths <- c(0, 1, 2, 3, 4, 5, 6, 7)
bthA <- c(10000, 8900, 5000, 3500, 2000, 1000, 300, 10)
bthD <- c(0, 1, 2, 3, 4, 5, 6, 7)

#Return the average temperature of the water column between 2 and 6 meters.
layer.temperature(top, bottom, wtr, depths, bthA, bthD)
```

---

load.bathy	<i>Import lake bathymetry data.</i>
------------	-------------------------------------

---

**Description**

Imports lake bathymetry data. Bathymetric data file must be a 2 column array where depth (in meters) and area (in meters<sup>2</sup>) information are provided in columns with headers containing the words "depths" and "areas" respectively.

**Usage**

```
load.bathy(fPath)
```

**Arguments**

fPath            File path to the bathymetry file.

**Value**

data.frame of depth and area for given lake.

**See Also**

[load.ts](#)

**Examples**

```
#Get the path for the package example file included
exampleFilePath <- system.file('extdata', 'Sparkling.bth', package="rLakeAnalyzer")

#Load and plot the hypsometric curve
sparkling.bathy = load.bathy(exampleFilePath)

#If successful, there will be two colums. "depths", and "areas".
plot(sparkling.bathy$areas, sparkling.bathy$depths, type='l', ylim=c(20,0),
     ylab='Depths (m)', xlab='Areas (m^2)')
```

---

load.ts	<i>Load timeseries from properly formatted text file.</i>
---------	---

---

**Description**

A convenience function to load timeseries data into R based on the standardized format used by Lake Analyzer.

Timeseries files must follow a common format. The first column must have the label 'datetime' and be of the format *yyyy-mm-dd HH:MM:SS* (ISO 8601 without the "T" delimiter). The second can be skipped if not using sub-minute data.

**Usage**

```
load.ts(fPath, tz = "GMT")
```

**Arguments**

fPath	The file path as a string.
tz	Timezone string to be supplied to <a href="#">as.POSIXct</a> . Defaults to GMT (UTC-0). This often can be left to the default unless timezone support is specifically required.

**Value**

A data frame in the required format for use with other rLakeAnalyzer timeseries functions.

**See Also**

For dataloading [ts.meta.depths](#),  
 For analyzing timeseries data, see [ts.meta.depths](#), [ts.thermo.depth](#), [ts.schmidt.stability](#),  
[ts.lake.number](#).

**Examples**

```
#Get the path for the package example file included
exampleFilePath <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")

#Load
sparkling.temp = load.ts(exampleFilePath)

#calculate and plot the thermocline depth
t.d = ts.thermo.depth(sparkling.temp)

plot(t.d$datetime, t.d$thermo.depth, type='l', ylab='Thermocline Depth (m)', xlab='Date')
```

---

 meta.depths

---

*Calculate the Top and Bottom Depths of the Metalimnion*


---

**Description**

Calculates the top and bottom depths of the metalimnion in a stratified lake. The metalimnion is defined as the water stratum in a stratified lake with the steepest thermal gradient and is demarcated by the bottom of the epilimnion and top of the hypolimnion.

**Usage**

```
meta.depths(wtr, depths, slope = 0.1, seasonal = TRUE, mixed.cutoff = 1)
```

**Arguments**

wtr	a numeric vector of water temperature in degrees C
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
slope	a numeric vector corresponding to the minimum slope
seasonal	a logical value indicating whether the seasonal thermocline should be returned. This is fed to thermo.depth, which is used as the starting point. The seasonal thermocline is defined as the deepest density gradient found in the profile. If FALSE, the depth of the maximum density gradient is used as the starting point.
mixed.cutoff	A cutoff (deg C) where below this threshold, thermo.depth and meta.depths are not calculated (NaN is returned). Defaults to 1 deg C.

**Value**

A numeric vector of the top and bottom metalimnion depths in meters. Returns the bottom depth if no distinct metalimnion top and bottom found.

**References**

Wetzel, R. G. 2001. Limnology: Lake and River Ecosystems, 3rd ed. Academic Press.

**See Also**

[ts.meta.depths](#), [thermo.depth](#)

**Examples**

```
wtr = c(22.51, 22.42, 22.4, 22.4, 22.4, 22.36, 22.3, 22.21, 22.11, 21.23, 16.42,
15.15, 14.24, 13.35, 10.94, 10.43, 10.36, 9.94, 9.45, 9.1, 8.91, 8.58, 8.43)

depths = c(0, 0.5, 1, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20)

m.d = meta.depths(wtr, depths, slope=0.1, seasonal=FALSE)
cat('The top depth of the metalimnion is:', m.d[1])
cat('The bottom depth of the metalimnion is:', m.d[2])
```

**Description**

Standardized methods for calculating common important derived physical features of lakes including water density based on temperature, thermal layers, thermocline depth, lake number, Wedderburn number, Schmidt stability and others.

---

schmidt.plot	<i>Creates a time series plot of Schmidt's stability</i>
--------------	--

---

### Description

Generates a time series of Schmidt's stability where each value represents water column stability for each time step of data. See [schmidt.stability](#) for more details and reference.

### Usage

```
schmidt.plot(wtr, bth)
```

### Arguments

wtr	Data frame of water temperature loaded with <a href="#">load.ts</a>
bth	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>

### References

See [schmidt.stability](#)

### See Also

[schmidt.stability](#)

### Examples

```
# Get system data file paths
wtr.path <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")
bth.path <- system.file('extdata', 'Sparkling.bth', package="rLakeAnalyzer")

# Load data for example lake, Sparkling Lake, Wisconsin.
wtr = load.ts(wtr.path)
bth = load.bathy(bth.path)

## Not run:
# Generate default plot
schmidt.plot(wtr,bth)

## End(Not run)
```

---

schmidt.stability      *Calculate the Schmidt stability*

---

### Description

Schmidt stability, or the resistance to mechanical mixing due to the potential energy inherent in the stratification of the water column.

Schmidt stability was first defined by Schmidt (1928) and later modified by Hutchinson (1957). This stability index was formalized by Idso (1973) to reduce the effects of lake volume on the calculation (resulting in a mixing energy requirement per unit area).

### Usage

```
schmidt.stability(wtr, depths, bthA, bthD, sal = 0)
```

### Arguments

wtr	a numeric vector of water temperature in degrees C
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
bthA	a numeric vector of cross sectional areas (m <sup>2</sup> ) corresponding to bthD depths
bthD	a numeric vector of depths (m) which correspond to areal measures in bthA
sal	a numeric vector of salinity in Practical Salinity Scale units

### Value

a numeric vector of Schmidt stability (J/m<sup>2</sup>)

### References

Schmidt, W., 1928. *Ueber Temperatur and Stabilitaetsverhaeltnisse von Seen*. Geographiska Annaler 10, 145-177.

Hutchinson, G.E., 1957. *A Treatise on Limnology*, vol. 1. John Wiley & Sons, Inc., New York.

Idso, S.B., 1973. *On the concept of lake stability*. Limnology and Oceanography 18, 681-683.

### See Also

[ts.schmidt.stability](#) [lake.number](#) [wedderburn.number](#)

### Examples

```
bthA <-c(1000,900,864,820,200,10)
bthD <-c(0,2.3,2.5,4.2,5.8,7)

wtr <-c(28,27,26.4,26,25.4,24,23.3)
depths <-c(0,1,2,3,4,5,6)
```

```
cat('Schmidt stability for input is: ')
cat(schmidt.stability(wtr, depths, bthA, bthD))
```

---

thermo.depth	<i>Calculate depth of the thermocline from a temperature profile.</i>
--------------	---

---

### Description

This function calculates the location of the thermocline from a temperature profile. It uses a special technique to estimate where the thermocline lies even between two temperature measurement depths, giving a potentially finer-scale estimate than usual techniques.

### Usage

```
thermo.depth(wtr, depths, Smin = 0.1, seasonal = TRUE, index = FALSE,
             mixed.cutoff = 1)
```

### Arguments

wtr	a numeric vector of water temperature in degrees C
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
Smin	Optional parameter defining minimum density gradient for thermocline
seasonal	a logical value indicating whether the seasonal thermocline should be returned. This is fed to thermo.depth, which is used as the starting point. The seasonal thermocline is defined as the deepest density gradient found in the profile. If FALSE, the depth of the maximum density gradient is used as the starting point.
index	Boolean value indicated if index of the thermocline depth, instead of the depth value, should be returned.
mixed.cutoff	A cutoff (deg C) where below this threshold, thermo.depth and meta.depths are not calculated (NaN is returned). Defaults to 1 deg C.

### Value

Depth of thermocline. If no thermocline found, value is NaN.

### See Also

[ts.thermo.depth](#), [water.density](#)

**Examples**

```
# A vector of water temperatures
wtr = c(22.51, 22.42, 22.4, 22.4, 22.4, 22.36, 22.3, 22.21, 22.11, 21.23, 16.42,
15.15, 14.24, 13.35, 10.94, 10.43, 10.36, 9.94, 9.45, 9.1, 8.91, 8.58, 8.43)

#A vector defining the depths
depths = c(0, 0.5, 1, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20)

t.d = thermo.depth(wtr, depths, seasonal=FALSE)

cat('The thermocline depth is:', t.d)
```

---

ts.buoyancy.freq	<i>Calculate the buoyancy (Brunt-Vaisala) frequency for a temperature profile.</i>
------------------	--

---

**Description**

Function for simplifying the calculation of buoyancy frequency. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

**Usage**

```
ts.buoyancy.freq(wtr, at.thermo = TRUE, na.rm = FALSE, ...)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a>
at.thermo	Boolean indicating if only buoyancy frequency at the thermocline should be returned. If false, full profile is returned.
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will likely return an NA value. If true, best effort will be made to calculate indices despite NA values.
...	Additional parameters will be passed on to thermo.depth function when extracting buoyancy frequency at only the thermocline. Common parameters to supply would be seasonal and slope.

**Value**

Returns a data frame with the timeseries of buoyancy frequency in units  $\text{sec}^{-2}$ . Includes a 'date-time' column.

**References**

Imberger, J., Patterson, J.C., 1990. *Physical limnology*. Advances in Applied Mechanics 27, 353-370.

**See Also**

buoyancy.freq

**Examples**

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng Lake, Wisconsin.
sp.wtr = load.ts(wtr.path)

N2 = ts.buoyancy.freq(sp.wtr, seasonal=FALSE)
SN2 = ts.buoyancy.freq(sp.wtr, seasonal=TRUE)

plot(N2, type='l', ylab='Buoyancy Frequency', xlab='Date')
lines(SN2, col='red')
```

---

ts.center.buoyancy      *Calculates the center of buoyancy for multiple temperature profiles.*

---

**Description**

Function for simplifying the calculation of the center of buoyancy. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

**Usage**

```
ts.center.buoyancy(wtr, na.rm = FALSE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a>
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.

**Value**

Returns a data frame with the timeseries of the center of buoyancy frequency. Includes a 'datetime' column.

**References**

Imberger, J., Patterson, J.C., 1990. *Physical limnology*. Advances in Applied Mechanics 27, 353-370.

**See Also**

center.buoyancy, load.bathy, load.ts

**Examples**

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")

#Load data for example lake, Sparkling Lake, Wisconsin.
sp.wtr = load.ts(wtr.path)

#calculate and plot the thermocline depth
t.d = ts.thermo.depth(sp.wtr)

center.N2 = ts.center.buoyancy(sp.wtr)

plot(center.N2, type='l', ylab='Depth (m)', xlab='Date', ylim=c(19,0), lwd = 1.5)
lines(t.d, type='l', col='red', lwd = 1.5)
legend(x = t.d[3,1], y = .25,
       c('center of buoyancy', 'thermocline depth'),
       lty=c(1,1),
       lwd=c(1.5,1.5),col=c("black","red"), bty = "n")
```

---

ts.internal.energy      *Calculate physical indices for a timeseries.*

---

**Description**

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

**Usage**

```
ts.internal.energy(wtr, bathy, na.rm = FALSE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns datetime, wtr_##.# where ##.# is depth in meters.
bathy	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.

**Value**

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.lake.number](#), [ts.meta.depths](#), [ts.schmidt.stability](#), [ts.thermo.depth](#), [ts.uStar](#)

---

ts.lake.number	<i>Calculate physical indices for a timeseries.</i>
----------------	---

---

**Description**

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

**Usage**

```
ts.lake.number(wtr, wnd, wnd.height, bathy, seasonal = TRUE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns datetime, wtr_##.# where ##.# is depth in meters.
wnd	A data frame of wind speeds (in m/s). Loaded using <a href="#">load.ts</a>
wnd.height	Height of the anemometer above the lake surface in meters
bathy	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
seasonal	Boolean indicating if seasonal thermocline should be used in calculation.

**Value**

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.internal.energy](#), [ts.meta.depths](#), [ts.schmidt.stability](#), [ts.thermo.depth](#), [ts.uStar](#)

---

`ts.layer.temperature`    *Calculate volume-weighted average water temperature across a range of depths for a timeseries.*

---

**Description**

Function for simplifying the calculation of Wedderburn Number. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

**Usage**

```
ts.layer.temperature(wtr, top, bottom, bathy, na.rm = FALSE)
```

**Arguments**

<code>wtr</code>	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a>
<code>top</code>	Either a single numeric depth value to be used across the entire timeseries, or a vector of same length as the timeseries (e.g., <code>nrow(wtr)</code> ). This is useful when calculating a time-varying layer average, like average epilimnion temperature.
<code>bottom</code>	Either a single numeric depth value to be used across the entire timeseries, or a vector of same length as the timeseries (e.g., <code>nrow(wtr)</code> ). This is useful when calculating a time-varying layer average, like average epilimnion temperature.
<code>bathy</code>	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
<code>na.rm</code>	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.

**Value**

Returns a data frame with the timeseries of the average layer temperature. Includes 'datetime' and 'layer.temp' columns.

**See Also**

`layer.temperature`

## Examples

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")
bathy.path <- system.file('extdata', 'Sparkling.bth', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng lake, in Wisconsin.
sp.wtr = load.ts(wtr.path)
sp.bathy = load.bathy(bathy.path)

l.t = ts.layer.temperature(sp.wtr, 0, 18, sp.bathy)
plot(l.t$datetime, l.t$layer.temp, type='l',
      ylab='Volumetrically averaged lake temperature', xlab='Date')
```

---

ts.meta.depths	<i>Calculate physical indices for a timeseries.</i>
----------------	---

---

## Description

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

## Usage

```
ts.meta.depths(wtr, slope = 0.1, na.rm = FALSE, ...)
```

## Arguments

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns datetime, wtr_##.# where ##.# is depth in meters.
slope	The minimum density gradient (kg/m <sup>3</sup> /m) that can be called the thermocline
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.
...	Additional parameters passed to underlying base function (e.g., index=TRUE for thermo.depth)

## Value

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.internal.energy](#), [ts.lake.number](#), [ts.schmidt.stability](#), [ts.thermo.depth](#), [ts.uStar](#)

**Examples**

```
#Get the path for the package example file included
exampleFilePath <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")

#Load
sparkling.temp = load.ts(exampleFilePath)

#calculate and plot the metalimnion depths
m.d = ts.meta.depths(sparkling.temp)

plot(m.d$datetime, m.d$top, type='l', ylab='Meta Depths (m)', xlab='Date', col='blue')
lines(m.d$datetime, m.d$bottom, col='red')
```

---

ts.schmidt.stability *Calculate physical indices for a timeseries.*

---

**Description**

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

**Usage**

```
ts.schmidt.stability(wtr, bathy, na.rm = FALSE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns datetime, wtr_##.# where ##.# is depth in meters.
bathy	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.

**Value**

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.internal.energy](#), [ts.lake.number](#), [ts.meta.depths](#), [ts.thermo.depth](#), [ts.uStar](#)

---

ts.thermo.depth	<i>Calculate physical indices for a timeseries.</i>
-----------------	---

---

**Description**

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

**Usage**

```
ts.thermo.depth(wtr, Smin = 0.1, na.rm = FALSE, ...)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns datetime, wtr_##.# where ##.# is depth in meters.
Smin	The minimum density gradient cutoff (kg/m <sup>3</sup> /m) defining the metalimion
na.rm	Boolean indicated if step-by-step removal of NA's should be tried. If false, a timestep with any NA values will return an NA value. If true, best effort will be made to calculate indices despite NA values.
...	Additional parameters passed to underlying base function (e.g., index=TRUE for thermo.depth)

**Value**

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.internal.energy](#), [ts.lake.number](#), [ts.meta.depths](#), [ts.schmidt.stability](#), [ts.uStar](#)

**Examples**

```
#Get the path for the package example file included
exampleFilePath <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")

#Load
sparkling.temp = load.ts(exampleFilePath)

#calculate and plot the thermocline depth
t.d = ts.thermo.depth(sparkling.temp)

plot(t.d$date, t.d$thermo.depth, type='l', ylab='Thermocline Depth (m)', xlab='Date')
```

---

ts.uStar

*Calculate physical indices for a timeseries.*


---

**Description**

Functions for simplifying the calculation of physical indices for a timeseries of observation data. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

These are wrapper functions that accept a timeseries of data and call the core physical metric functions (like [schmidt.stability](#)) on each timestep.

**Usage**

```
ts.uStar(wtr, wnd, wnd.height, bathy, seasonal = TRUE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a> . Must have columns <code>datetime</code> , <code>wtr_##.#</code> where <code>##.#</code> is depth in meters.
wnd	A data frame of wind speeds (in m/s). Loaded using <a href="#">load.ts</a>
wnd.height	Height of the anemometer above the lake surface in meters
bathy	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
seasonal	Boolean indicating if seasonal thermocline should be used in calculation.

**Value**

Returns a data frame with the timeseries of calculated derivatives. All include a 'datetime' column, but derivative columns differ between functions.

**See Also**

For loading input data [load.ts](#), [load.bathy](#).

For the underlying functions operating at each timestep [meta.depths](#), [thermo.depth](#), [schmidt.stability](#), [lake.number](#), [internal.energy](#).

Other Timeseries functions for r Lake Analyzer: [ts.internal.energy](#), [ts.lake.number](#), [ts.meta.depths](#), [ts.schmidt.stability](#), [ts.thermo.depth](#)

---

ts.wedderburn.number *Calculate Wedderburn number for a timeseries.*

---

**Description**

Function for simplifying the calculation of Wedderburn Number. Can usually be called directly on data loaded directly using [load.ts](#) and [load.bathy](#).

**Usage**

```
ts.wedderburn.number(wtr, wnd, wnd.height, bathy, Ao, seasonal = TRUE)
```

**Arguments**

wtr	A data frame of water temperatures (in Celsius). Loaded using <a href="#">load.ts</a>
wnd	A data frame of wind speeds (in m/s). Loaded using <a href="#">load.ts</a>
wnd.height	Height of the anemometer above the lake surface in meters
bathy	A data frame containing hypsometric data. Loaded using <a href="#">load.bathy</a>
Ao	Numeric value for the water body surface area (m <sup>2</sup> ) at zero meters depth
seasonal	Boolean indicating if seasonal thermocline should be used in calculation.

**Value**

Returns a data frame with the timeseries of Wedderburn number. Includes a 'datetime' column.

**References**

Imberger, J., Patterson, J.C., 1990. *Physical limnology*. Advances in Applied Mechanics 27, 353-370.

**See Also**

wedderburn.number, ts.lake.number

## Examples

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")
wnd.path <- system.file('extdata', 'Sparkling.daily.wnd', package="rLakeAnalyzer")
bathy.path <- system.file('extdata', 'Sparkling.bth', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng lake, in Wisconsin.
sp.wtr = load.ts(wtr.path)
sp.wnd = load.ts(wnd.path)
sp.bathy = load.bathy(bathy.path)

sp.area = 64e4 #Area of Sparkling lake in m^2
wnd.height = 2 #Height of Sparkling lake anemometer

w.n = ts.wedderburn.number(sp.wtr, sp.wnd, wnd.height, sp.bathy, sp.area)
plot(w.n$date, w.n$wedderburn.number, type='l', ylab='Wedderburn Number', xlab='Date')
```

---

uStar

*Calculates the water friction velocity, uStar*


---

## Description

uStar is the water friction velocity due to wind stress at the lake surface, it is calculated following the methods of Imberger (1985) as a function of the shear stress of air (Fischer et al., 1979), drag coefficient for momentum (Hicks, 1972), and a dimensionless constant (von Karman constant) that describes the logarithmic velocity profile at the air-water interface

## Usage

```
uStar(wndSpeed, wndHeight, averageEpiDense)
```

## Arguments

wndSpeed	a numeric vector of wind speed in m s <sup>-1</sup>
wndHeight	a numeric vector of wind measurement height in m
averageEpiDense	a numeric vector of epilimnion density in kg m <sup>-3</sup>

## Value

a numeric vector of uStar

## References

Hicks, B.B., 1972. *A procedure for the formulation of bulk transfer coefficients over water bodies of different sizes*. *Boundary-Layer Meteorology* 3: 201-213.

Amorocho, J., DeVries, J.J., 1980. *A new evaluation of the wind stress coefficient over water surfaces*. *Journal of Geophysical Research* 85: 433-442.

Fischer, H.B., List, E.J., Koh, R.C.Y., Imberger, J., Brooks, N.H., 1979. *Mixing in inland and coastal waters*. Academic Press.

Imberger, J., 1985. *The diurnal mixed layer*. *Limnology and Oceanography* 30: 737-770.

## See Also

[ts.uStar layer.density](#)

## Examples

```
wndSpeed <- c(5.1,6.3,6.3,5.2,7,7.2)
wndHeight <-2
averageEpiDense <- c(14,15,14.2,13,12,12)

cat('uStar for input vector is: ')
cat(uStar(wndSpeed,wndHeight,averageEpiDense))
```

---

water.density

*Estimate Water Density*

---

## Description

Density of water from temperature and salinity

## Usage

```
water.density(wtr, sal = wtr * 0)
```

## Arguments

wtr                    a numeric vector of water temperature in degrees Celsius  
sal                    a numeric vector of salinity in Practical Salinity Scale units

## Value

A numeric vector of water densities in kg/m<sup>3</sup>.

## References

- Martin, J.L., McCutcheon, S.C., 1999. *Hydrodynamics and Transport for Water Quality Modeling*. Lewis Publications, Boca Raton, FL, 794pp.
- Millero, F.J., Poisson, A., 1981. *International one-atmosphere equation of state of seawater*. UNESCO Technical Papers in Marine Science. No. 36.

## Examples

```
#Plot water density for water between 1 and 30 deg C
dens = water.density(1:30)
plot(1:30, dens, xlab="Temp(deg C)", ylab="Density(kg/m^3)")
```

---

wedderburn.number      *Calculates Wedderburn Number for a lake.*

---

## Description

Wedderburn Number (Wn) is a dimensionless parameter measuring the balance between wind stress and bouyancy force and is used to estimate the amount of upwelling occurring in a lake. When Wn is much greater than 1, the bouyancy force is much greater than the wind stress and therefore there is a strong vertical stratification with little horizontal variation in the stratification. When Wn is much less than 1, the wind stress is much greater than the bouyancy force and upwelling is likely occurring at the upwind end of the lake. When Wn is near 1, the bouyance force and wind stress are nearly equal and horizontal mixing is considered important

## Usage

```
wedderburn.number(delta_rho, metaT, uSt, Ao, AvHyp_rho)
```

## Arguments

delta_rho	Numeric value for the water density difference between the epilimnion and hypolimnion (kg/m <sup>3</sup> )
metaT	Numeric value for the thickness of the water body's surface layer (m)
uSt	Numeric value for the water friction velocity due to wind stress (m/s)
Ao	Numeric value for the water body surface area (m <sup>2</sup> ) at zero meters depth
AvHyp_rho	Numeric value for the average water density of the hypolimnion layer (kg/m <sup>3</sup> )

## Value

The dimensionless numeric value of Wedderburn Number

## References

- Imberger, J., Patterson, J.C., 1990. *Physical limnology*. Advances in Applied Mechanics 27, 353-370.

**See Also**

[ts.wedderburn.number lake.number](#)

**Examples**

```
delta_rho <- c(3.1,1.5)
metaT <- c(5.5,2.4)
uSt <- c(0.0028,0.0032)
Ao <- c(80300,120000)
AvHyp_rho <- c(999.31,999.1)
wedderburn.number(delta_rho, metaT, uSt, Ao, AvHyp_rho)
```

---

whole.lake.temperature

*Get volumetrically averaged whole lake temperature*

---

**Description**

Calculates volumetrically weighted average whole lake temperature using the supplied water temperature timeseries.

**Usage**

```
whole.lake.temperature(wtr, depths, bthA, bthD)
```

**Arguments**

wtr	a numeric vector of water temperature in degrees C.
depths	a numeric vector corresponding to the depths (in m) of the wtr measurements
bthA	a numeric vector of cross sectional areas (m <sup>2</sup> ) corresponding to bthD depths
bthD	a numeric vector of depths (m) which correspond to areal measures in bthA

**See Also**

[hypo.temperature](#), [epi.temperature](#)

---

wtr.heat.map	<i>Plots a heat-map of water temperature.</i>
--------------	---

---

### Description

This creates a simple, default heatmap of water temperature.

### Usage

```
wtr.heat.map(wtr, ...)
```

### Arguments

wtr	Data frame of water temperature loaded with <a href="#">load.ts</a> .
...	Additional parameters supplied to <a href="#">filled.contour</a> to modify defaults. Common examples include <code>zlim</code> and <code>plot.title</code> .

### See Also

[load.ts](#)

### Examples

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.daily.wtr', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng Lake, Wisconsin.
sp.wtr = load.ts(wtr.path)

#Plot default figure
wtr.heat.map(sp.wtr)

#Change defaults supplied to filled.contour
wtr.heat.map(sp.wtr, zlim=c(0,15), plot.title="Sparkling Water Temp (C)")
```

---

wtr.heatmap.layers	<i>Plots water temperature heatmap with major limnetic layers indicated</i>
--------------------	---

---

### Description

This creates a heat map of water temperature similar to [wtr.heat.map](#) with additional lines drawn to denote the thermocline, and the top and bottom of the metalimnion as calculated using [ts.meta.depths](#) and [thermo.depth](#).

### Usage

```
wtr.heatmap.layers(wtr, ...)
```

**Arguments**

wtr                    Data frame of water temperature loaded with `load.ts`.

...                    Additional parameters supplied to `filled.contour` to modify defaults. Common examples include `zlim` and `plot.title`.

**Note**

This plot cannot be used in customized multi-panel figures using `layout` as `layout` is already used in the `filled.contour` plotting function.

**See Also**

[wtr.heatmap](#) [load.ts](#) [ts.meta.depths](#) [ts.thermo.depth](#)

**Examples**

```
#Get the path for the package example file included
wtr.path <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng Lake, Wisconsin.
wtr = load.ts(wtr.path)

# generate default plot
## Not run:
wtr.heatmap.layers(wtr)

## End(Not run)
```

---

wtr.layer

*Exploration of lake water column layers*

---

**Description**

Extract water column parameters of a given parameter from a profile using the split-and-merge algorithm. The cline is defined as the midpoint of the layer of water where the physical property change in the greatest over a small difference. The exact cline depends on the specification of measure. For example if temperature is specified, then we can expect cline to output the thermocline.

**Usage**

```
wtr.layer(data, depth, measure, thres = 0.1, z0 = 2.5, zmax = 150,
  nseg = "unconstrained")
```

**Arguments**

data	data supplied as a bare (unquoted) value
depth	depth in metres; should be an increasing vector; supplied as a bare (unquoted) value
measure	parameter measured in the water column profile; supplied as a bare (unquoted) value
thres	error norm; defaults to 0.1
z0	initial depth in metres. Defaults to 2.5m
zmax	maximum depth in metres: defaults to 150m
nseg	optional parameter to define the number of segments a priori; defaults to an unconstrained approach whereby the algorithm determines segmentations by minimizing the error norm over each segment

**Value**

a dataframes with a list column. This includes: nseg (number of segments), mld (mix layer depth), cline (the midpoint of the segment connecting inflection points that has the maximum slope; thermocline for temperature measures) and segments calculated by the sm algorithm.

**References**

Thomson, R. and I. Fine. 2003. Estimating Mixed Layer Depth from Oceanic Profile Data. *Journal of Atmospheric and Oceanic Technology*. 20(2), 319-329.

**Examples**

```
data("latesummer")
df1 <- wtr.layer(depth=latesummer$depth, measure = latesummer$temper)
df1$mld
df1$segments

wtr.layer(data = latesummer, depth=depth, measure = temper, nseg=4)
```

---

wtr.lineseries

*Creates a line based plot of temperature profile time series*


---

**Description**

A non-heat map approach to visualizing a water temperature profile useful for identify temperature trends over time at discrete depths and diagnosing issues with data.

**Usage**

```
wtr.lineseries(wtr, ylab = "Temperature C", ...)
```

**Arguments**

wtr                    Data frame of water temperature loaded with [load.ts](#).  
ylab                   y axis title  
...                    Additional parameters supplied to the plot function

**See Also**

See [load.ts](#) and [wtr.heat.map](#)

**Examples**

```
exampleFilePath <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")  
wtr= load.ts(exampleFilePath)  
## Not run:  
wtr.lineseries(wtr, ylab = "Temperature C")  
  
## End(Not run)
```

---

wtr.plot.temp	<i>Creates a time series plot of the thermocline and top and bottom of the metalimnion</i>
---------------	--

---

**Description**

A line based plot of calculated depths of the thermocline, and top and bottom of the metalimnion from a temperature profile time series.

**Usage**

```
wtr.plot.temp(wtr, ...)
```

**Arguments**

wtr                    Data frame of water temperature loaded with [load.ts](#).  
...                    Additional paramters supplied to [ts.meta.depths](#) and [ts.thermo.depth](#)

**See Also**

[load.ts](#) and [wtr.lineseries](#)

### **Examples**

```
wtr.path <- system.file('extdata', 'Sparkling.wtr', package="rLakeAnalyzer")

#Load data for example lake, Sparkilng Lake, Wisconsin.
wtr = load.ts(wtr.path)

## Not run:
wtr.plot.temp(wtr)

## End(Not run)
```

# Index

## \* arith

buoyancy.freq, 4  
center.buoyancy, 5  
schmidt.stability, 19  
ts.buoyancy.freq, 21  
ts.center.buoyancy, 22  
ts.layer.temperature, 25  
ts.wedderburn.number, 30  
water.density, 32  
wedderburn.number, 33

## \* datasets

latesummer, 12

## \* file

load.bathy, 15  
load.ts, 15

## \* hplot

lake.number.plot, 11  
schmidt.plot, 18  
wtr.heat.map, 35  
wtr.heatmap.layers, 35  
wtr.lineseries, 37  
wtr.plot.temp, 38

## \* manip

get.offsets, 8  
lake.number, 10  
layer.density, 13  
layer.temperature, 14  
meta.depths, 16  
thermo.depth, 20  
ts.internal.energy, 23  
ts.lake.number, 24  
ts.meta.depths, 26  
ts.schmidt.stability, 27  
ts.thermo.depth, 28  
ts.uStar, 29  
uStar, 31

approx.bathy, 2  
as.POSIXct, 16

buoyancy.freq, 4

center.buoyancy, 5

depth.filter, 6

drop.datetime, 6

epi.temperature, 7, 9, 34

filled.contour, 35, 36

get.datetime, 7

get.offsets, 8

hypo.temperature, 7, 9, 34

internal.energy, 9, 24, 25, 27–30

lake.number, 10, 11, 19, 24, 25, 27–30, 34

lake.number.plot, 11

latesummer, 12

layer.density, 13, 32

layer.temperature, 14

layout, 36

load.bathy, 11, 15, 18, 21–30

load.ts, 8, 11, 15, 15, 18, 21–30, 35, 36, 38

meta.depths, 16, 24, 25, 27–30

rLakeAnalyzer, 17

rLakeAnalyzer-package (rLakeAnalyzer),  
17

schmidt.plot, 18

schmidt.stability, 18, 19, 23–30

thermo.depth, 17, 20, 24, 25, 27–30, 35

ts.buoyancy.freq, 21

ts.center.buoyancy, 22

ts.internal.energy, 23, 25, 27–30

ts.lake.number, 11, 16, 24, 24, 27–30

ts.layer.temperature, 25

ts.meta.depths, [16](#), [17](#), [24](#), [25](#), [26](#), [28–30](#),  
[35](#), [36](#), [38](#)  
ts.schmidt.stability, [16](#), [19](#), [24](#), [25](#), [27](#),  
[27](#), [29](#), [30](#)  
ts.thermo.depth, [16](#), [20](#), [24](#), [25](#), [27](#), [28](#), [28](#),  
[30](#), [36](#), [38](#)  
ts.uStar, [24](#), [25](#), [27–29](#), [29](#), [32](#)  
ts.wedderburn.number, [30](#), [34](#)

uStar, [31](#)

water.density, [32](#)  
wedderburn.number, [11](#), [19](#), [33](#)  
whole.lake.temperature, [7](#), [9](#), [34](#)  
wtr.heat.map, [35](#), [35](#), [36](#), [38](#)  
wtr.heatmap.layers, [35](#)  
wtr.layer, [36](#)  
wtr.lineseries, [12](#), [37](#), [38](#)  
wtr.plot.temp, [38](#)