

Package ‘aiRthermo’

August 7, 2017

Type Package

Title Atmospheric Thermodynamics and Visualization

Version 1.1

Date 2017-07-11

Author Jon Sáenz, Santos J. González-Rojí, Sheila Carreno-Madinabeitia and Gabriel Ibarra-Berastegi

Maintainer Santos J. González-Rojí <santosjose.gonzalez@ehu.eus>

Description Deals with many computations related to the thermodynamics of atmospheric processes. It includes many functions designed to consider the density of air with varying degrees of water vapour in it, saturation pressures and mixing ratios, conversion of moisture indices, computation of atmospheric states of parcels subject to dry or pseudoadiabatic vertical evolutions and atmospheric instability indices that are routinely used for operational weather forecasts or meteorological diagnostics.

License GPL-3

Encoding UTF-8

Repository CRAN

NeedsCompilation yes

Date/Publication 2017-08-07 19:38:38 UTC

R topics documented:

aiRthermo-package	2
adiabatic_ascent	3
aiRthermoConstants	4
AnyAdiabaticDown	5
boltonTLCL	6
bruntVaisallaOmegaSquared	7
C2K	8
CAPE_CIN	9
densityDry	11
densityH2Ov	12
densityMoistAir	12
dewpointdepression2rh	13

e2w	14
equivalentPotentialTemperature	15
export_constants	16
export_lines	17
find_lcl	17
fixedlines	18
gamma_saturated	19
K2C	20
Kindex	20
latent_heat_H2O	21
Lindex	22
moistAdiabaticLapseRate	23
moistCp	24
moistCv	25
parcelState	26
PT2Theta	27
PTheta2T	27
PW	28
q2e	29
q2w	30
RadiosondeA	30
RadiosondeD	31
RadiosondeDavenport	32
rh2shum	33
rh2w	34
saturation_mixing_ratio	35
saturation_pressure_H2O	35
Sindex	36
stuve_diagram	37
TTdP2rh	38
TTheta2P	39
TTindex	40
virtual_temperature	41
w2q	41
w2Td	42
Index	43

Description

Deals with many computations related to the thermodynamics of atmospheric processes. It includes many functions designed to consider the density of air with varying degrees of water vapour in it, saturation pressures and mixing ratios, conversion of moisture indices, computation of atmospheric states of parcels subject to dry or pseudoadiabatic vertical evolutions and atmospheric instability indices that are routinely used for operational weather forecasts or meteorological diagnostics.

Unless otherwise explicitly noted ([boltonTLCL](#) and [stuve_diagram](#)) all parameters to functions must be provided in the International System of Units: P in Pa, T in K and w in kg/kg.

Author(s)

Jon Sáenz, Santos J. González-Rojí, Sheila Carreno-Madinabeitia and Gabriel Ibarra-Berastegi
 Maintainer: Santos J. González-Rojí <santosjose.gonzalez@ehu.eus>

Examples

```
# CAPE, CIN index
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
capeCin<-CAPE_CIN(PlowTop=98000,precoolType="adiabatic",
                  Ps=aPs,Ts=aTs,ws=aws,deltaP=5,
                  getLiftedBack=TRUE,upToTop=TRUE)
print(min(capeCin$CAPE))

pdf("stuve.pdf")
stuveA<-stuve_diagram(Pres = aPs/100,Temp=aTs-273.15)
lines(capeCin$Tl-273.15,capeCin$P1/100,col="red",lwd=2)
dev.off()

# Adiabatic Ascent
P0<-101325
T0<-273.15
w0<-0.0025
adiabEvol<-adiabatic_ascent(P0,T0,w0,50000,5)
```

adiabatic_ascent *Properties of an air parcel after adiabatic ascent*

Description

A particle located at Pstart pressure (Pa), Tstart temperature (K) and wstart mixing ratio (kg/kg) ascends (pseudo)adiabatically to Pend (Pa). The evolution is computed by numerically integrating the dT/dP ordinary differential equation (ODE) using a 4th order Runge-Kutta scheme, assuming hydrostatic equilibrium and that the particle is saturated after the Lifted Condensation Level (LCL).

Usage

```
adiabatic_ascent(Pstart, Tstart, wstart, Pend, deltaP = 1)
```

Arguments

Pstart	Initial value for pressure (Pa)
Tstart	Initial value for temperature (K)
wstart	Initial value for mixing ratio (kg/kg)
Pend	End value for pressure (Pa)
deltaP	deltaP (Pa) represents the numerical increment used for integrating the Ordinary Differential Equation (ODE) representing the vertical evolution

Value

The function returns a list that includes Tend (final value of temperature) and mixRatioEnd (mixing ratio of the air parcel at the end of the evolution).

Tend	Temperature at the end (K)
mixRatioEnd	Mixing ratio at the end (kg/kg)

Examples

```
P0<-101325
T0<-273.15
w0<-0.0025
adiabEvov<-adiabatic_ascent(P0,T0,w0,50000,5)
```

aiRthermoConstants *Thermodynamical Constansts*

Description

Frecuently used constants in atmospheric thermodynamics and in this package.

Usage

```
data(aiRthermoConstants)
```

Format

aiRthermoConstants is a vector that includes the constants used by many of the functions in package

Details

The constants stored in the vector are (in SI units): the gas constant for dry air R_d and for water vapour R_v ($\frac{J}{Kkg}$), the temperature T_0 corresponding to 0 degree Celsius, es_0 used to calculate the saturated vapour pressure (Pa), 1000 hPa in Pa (P1000), the specific heat of dry air for constant pressure c_p ($\frac{J}{Kkg}$) and for constant volume c_v ($\frac{J}{Kkg}$), acceleration due to gravity at sea level g ($\frac{m}{s^2}$), our definition of a missing value MISSING_VALUE (-99999999) and epsilon ε ($\frac{R_d}{R_v}$).

The values of the constants are taken from Bohren & Albrecht (1998), and they are also consistent with those used in Petty (2008), Erukhimova & North (2009) and Davies-Jones (2009).

References

Bohren, C.F., & Albrecht, B. A. (1998). Atmospheric thermodynamics. Atmospheric thermodynamics. Publisher: New York; Oxford: Oxford University Press, 1998. ISBN: 0195099044.

Petty, G.W. (2008). A First Course in Atmospheric Thermodynamics, Sundog Publishing, Madison.

North, G. R. , Erukhimova, T. L. (2009). Atmospheric Thermodynamics, Cambridge University Press, New York.

Davies-Jones, R. (2009). On formulas for equivalent potential temperature, Monthly Weather Review, 137,3137-3148. doi:10.1175/2009MWR2774.1.

Examples

```
#Define the Rd
data(aiRthermoConstants)
Rd <- aiRthermoConstants['Rd']

#Define gravity
data(aiRthermoConstants)
g <- aiRthermoConstants['g']
```

AnyAdiabaticDown

Adiabatic Downwards Evolution

Description

Calculation of the state of an air parcel subject to an adiabatic downwards evolution, taking into account the initial conditions of the parcel (Pstart, Tstart, wstart, wcstart).

Usage

```
AnyAdiabaticDown(Pstart, Tstart, wstart, wcstart, Pend, deltaP)
```

Arguments

Pstart	Initial pressure value (Pa)
Tstart	Initial temperature value (K)
wstart	Initial mixing ratio value (kg/kg)
wcstart	Initial mixing ratio value for the condensates (kg/kg)
Pend	Final pressure value (Pa)
deltaP	Pressure step used for the calculation. It must be a positive value (Pa)

Details

In this case, we start from a parcel at pressure p_{start} (Pa), temperature t_{start} (K) and mixing ratio w_{start} (kg/kg), with potentially some condensates w_{cstart} (kg/kg). The latent heat (L) used during the evolution depends on the Temperature (T). It is computed as described by [latent_heat_H20](#). As the parcel goes down it could evaporate the condensates or, if no condensates are available anymore, it will go down according to a dry adiabatic evolution by means of a dry adiabatic process until the level P_{end} . At this point, it will have a temperature T_{end} , mixing ratio (vapour) W_{end} and W_{cend} (may be still some condensates could be left) using steps of pressure dP (always positive).

Value

This function returns a list including the following values:

Tend	Temperature at the end (K)
Wend	Mixing ratio of water vapour at the end (kg/kg)
Wcend	Mixing ratio of condensed water at the end (kg/kg)

Examples

```
AnyAdiabaticDown(50000,227,8.5e-5,0.005,101325,5)
AnyAdiabaticDown(70000,237,4e-4,0.005,101325,5)
```

 boltonTLCL

Find the Temperature at the Lifting Condensation Level (LCL)

Description

This function is used to calculate the Temperature at the Lifting Condensation Level (LCL) using Bolton's approximation instead of integrating the Ordinary Differential Equation (ODE) upwards

Usage

```
boltonTLCL(TempCelsius, rh, consts = export_constants())
```

Arguments

TempCelsius	Temperature in degrees Celsius
rh	Relative humidity (%)
consts	Includes the frequently used constants in thermodynamics defined in <i>aiRthermoConstants</i> .

Value

This function calculates an approximation of the temperature in degrees Celsius corresponding to the LCL.

References

Bolton, D. (1980). The computation of equivalent potential temperature, Monthly Weather Review 108, 1046-1053. doi:10.1175/1520-0493(1980)108<1046:TCOEPT>2.0.CO;2.

Examples

```
T0=273.15
rh=66.25489
boltonTLCL(T0,rh)
```

bruntVaisallaOmegaSquared

Brunt-Vaisalla (angular) frequency

Description

Brunt-Vaisalla (angular) frequency (s^{-2}) considering hydrostatic equilibrium. P is used as a vertical level.

Usage

```
bruntVaisallaOmegaSquared(Ps, Ts, ws, consts = export_constants())
```

Arguments

Ps	A vector with pressure values (Pa)
Ts	A vector with temperature values (K)
ws	A vector with mixing ratio values (kg/kg)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary. The constants g and Rd are used.

Details

The angular frequency (squared, (s^{-2})) is returned in order to avoid complex numbers.

Value

omega_sq

Note

For stable atmospheres, should be positive at every level. Ps, Ts and ws are 1D arrays

See Also

[PT2Theta](#) and [densityMoistAir](#) are used inside bruntVaisallaOmegaSquared.

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
bruntVaisallaOmegaSquared(dPs,dTs,dws)
```

C2K

From Celsius to Kelvin degrees

Description

This function makes the transformation from Celsius to Kelvin degrees.

Usage

```
C2K(Tc, consts = export_constants())
```

Arguments

Tc	A vector of temperatures in degrees Celsius
consts	This function uses the T_0 constant, corresponding to 0 degree Celsius expressed in K (273.15 K).

Value

A vector of temperatures in Kelvin degrees is returned

See Also

[aiRthermoConstants](#) and [K2C](#)

Examples

```
data(RadiosondeD)
dTs<-RadiosondeD[,3]
C2K(dTs)
```

CAPE_CIN *calculation of CAPE and CIN*

Description

Taking into account the data obtained in a radiosonde, and after defining the initial values of the parcel, this function calculates the values of CAPE and CIN for the sounding.

Usage

```
CAPE_CIN(Ps, Ts, ws, deltaP = 5, P0 = NA, T0 = NA, w0 = NA, PlowTop = NA,
precoolType = "none", doLog = 0, getLiftedBack = FALSE, upToTop = TRUE,
checkBuoyancy = 0)
```

Arguments

Ps	Pressures (Pa) defining the sounding
Ts	Temperatures (K) defining the sounding
ws	Mixing ratios (kg/kg) defining the sounding
deltaP	The width (Pa) of the layers used in the calculation of the numerical solution for the vertical evolution. A default value of 5 Pa is used. It must be positive.
P0	The initial pressure (Pa) for the parcel that is lifted (may be the lowest level of the sounding). Missing value is used by default.
T0	The initial temperature (K) of the parcel being lifted. Missing value is used by default.
w0	The initial mixing ratio (kg/kg) of the parcel being lifted.
PlowTop	If some layers must be averaged in the bottom of the sounding this argument provides the pressure (Pa) at the top of the layer that must be averaged in the bottom of the sounding. NA is used by default.
precoolType	If requested, an adiabatic or an isobaric precooling of the initial parcel is performed. "none" is used by default, but "adiabatic" and "isobaric" are also accepted.
doLog	Use logarithmic vertical interpolation between sounding levels if doLog=1. The default value is doLog=0.
getLiftedBack	TRUE/FALSE requests that the evolution of the lifted particle until the top level of the soundig is returned as a set of vectors for P, T and w (fields Pl, Tl and wl respectively). FALSE is used by default.
upToTop	TRUE(FALSE) requests that the lifted particle continues(stops) after the first crossing with the ambient sounding (EL) (until the sounding finishes). If TRUE, remaining negative areas above are accumulated into CIN only if the parcel becomes buoyant again in upper levels depending on the setting of <i>checkBuoyancy</i> . TRUE is used by default.
checkBuoyancy	If <i>checkBuoyancy</i> is TRUE, the computation of CAPE and CIN proceed to the top of the sounding if <i>upToTop</i> is TRUE if CAPE is larger than CIN while the parcel passes non-buoyant regions. The default value is FALSE.

Details

CAPE and CIN (J/kg) are calculated from a sounding given by 1D arrays for pressure P_s (Pa), for temperature T_s (K) and for mixing ratio w_s (kg/kg).

If $P_0/T_0/w_0$ are provided, no low vertical averaging is done and these values are used as initial points for the parcel. Missing value is used by default for these arguments.

This function returns some error codes in field *outCode* in the return value if the computation of CAPE and CIN failed.

Value

Returns:

<code>airStart</code>	The real starting variable of the air parcel. It is a vector with 6 elements: P (Pa), Temp (K), w (kg/kg), theta (K), Tvirtual (K) and wsat (kg/kg). The values are computed depending on the input arguments.
<code>cape</code>	CAPE index (J/kg)
<code>cin</code>	CIN index (J/kg) as a negative number
<code>apLCL</code>	Variables of the air parcel at the Lifting Condensation Level (LCL). It is returned as a vector with 6 elements: P (Pa), Temp (K), w (kg/kg), theta (K), virtualT (K) and wsat (kg/kg)
<code>apLFC</code>	Variables of the Level of Free Convection (LFC). If LFC is found, it is returned as a vector with six elements: P (Pa), Temp (K), w (kg/kg), theta (K), virtualT (K) and wsat (kg/kg)
<code>apEL</code>	End Level (EL). If EL is found, it is returned as a vector with six elements: P (Pa), Temp (K), w (kg/kg), theta (K), virtualT (K) and wsat (kg/kg)
<code>gotLCL</code>	TRUE/FALSE whether the LCL has been found or not
<code>gotLFC</code>	TRUE/FALSE whether the LFC has been found or not
<code>gotEL</code>	TRUE/FALSE whether the EL has been found or not
<code>P1</code>	Pressure (Pa) at every step of the lifted particle during its evolution. If requested by using <code>getLiftedBack==TRUE</code> , every step until the end of the radiosonde is returned.
<code>T1</code>	Temp (K) at every step of the lifted particle during its evolution. If requested by using <code>getLiftedBack==TRUE</code> , every step until the end of the radiosonde is returned.
<code>w1</code>	Mixing-ratio of the lifted particle during its evolution. If requested by using <code>getLiftedBack==TRUE</code> , every step until the end of the radiosonde is returned.
<code>Olifted</code>	Number of elements in P1/T1/w1
<code>upToTop</code>	Process the whole sounding even after finding the first "EL level"
<code>outCode</code>	The error code returned by the C routine that computes CAPE/CIN. If 0, everything has been OK!

Examples

```

data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
capeCin<-CAPE_CIN(PlowTop=98000,precoolType="adiabatic",
                  Ps=aPs,Ts=aTs,ws=aws,dolog=0,deltaP=5,
                  getLiftedBack=TRUE,upToTop=TRUE)
print(min(capeCin$Tl))

pdf("stuve.pdf")
stuveA<-stuve_diagram(Pres = aPs/100,Temp=aTs-273.15)
lines(capeCin$Tl-273.15,capeCin$P1/100,col="red",lwd=2)
dev.off()

```

densityDry

*Density of Dry Air***Description**

From pressure P (Pa) and temperature Temp (K), this function calculates the density of dry air in kg/m^3

Usage

```
densityDry(P, Temp, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
Temp	A vector with temperature values (K)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

A vector with density of dry air values is returned (kg/m^3)

Examples

```

data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTts<-C2K(RadiosondeD[,3])
densityDry(dPs,dTs)

```

densityH20v *Density of water vapour*

Description

From pressure of water vapour Pw (Pa) and temperature Temp (K), this function calculates density of water vapour (kg/m^3)

Usage

```
densityH20v(Pw, Temp, consts = export_constants())
```

Arguments

Pw A vector with pressure water vapour values (Pa)
 Temp A vector with temperature values (K)
 consts The constants defined in *aiRthermoConstants* data are necessary.

Value

A vector with density of water vapour values is returned (kg/m^3)

See Also

[q2e](#) and [w2q](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
h2oe<-q2e(dPs,w2q(dws))
densityH20v(h2oe,dTs)
```

densityMoistAir *Density of Moist Air*

Description

From pressure P (Pa) temperature Temp (K) and mixing ratio (kg/kg), this function calculates the density of moist air (kg/m^3)

Usage

```
densityMoistAir(P, Temp, w, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
Temp	A vector with temperature values (K)
w	A vector with mixing ratio values (kg/kg)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

A vector with density of moist air values is returned (kg/m^3)

See Also

[virtual_temperature](#)

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
densityMoistAir(aPs,aTs,aws)
```

dewpointdepression2rh *Relative Humidity from the dew point depression*

Description

This function calculates the relative humidity (%) from the dew point depression (K)

Usage

```
dewpointdepression2rh(P, Temp, dpd, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
Temp	A vector with temperature values (K)
dpd	A vector with dew point depression values (K)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

A vector with relative humidity (%)

See Also

[saturation_mixing_ratio](#) and [saturation_pressure_H2O](#)

Examples

```

data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dTds=w2Td(dPs,dws)
dDPDs=dTs-dTds
dewpointdepression2rh(dPs,dTs,dDPDs)

```

e2w

Compute Mixing Ratio from partial pressure of water vapour

Description

This function calculates the mixing ratio (kg/kg) from the partial vapour pressure of water vapour (Pa)

Usage

```
e2w(eh2o, P, consts = export_constants())
```

Arguments

eh2o	A vector with partial pressure of water vapour (Pa)
P	A vector with pressure (Pa) values
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

A vector with mixing ratio values

Examples

```

#Partial pressure of water vapour
data(RadiosondeA)
dPs<-RadiosondeA[,1]*100
dws<-RadiosondeA[,6]/1000
eh2o<-q2e(dPs,w2q(dws))
#Pressure
e2w(eh2o,dPs)

```

equivalentPotentialTemperature
Equivalent Potential Temperature

Description

This function calculates the equivalent potential temperature (K), following the techniques used in Davies-Jones (2009).

Usage

```
equivalentPotentialTemperature(P, Temp, w, TLCL, consts = export_constants())
```

Arguments

P	The pressure (Pa) of the air parcel
Temp	The temperature (K) of the parcel
w	The mixing ratio (kg/kg) of the parcel
TLCL	The temperature (K) at the Lifting Condensation Level (LCL)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns the value of the equivalent potential temp (K)

References

Davies-Jones, R. (2009). On formulas for equivalent potential temperature. *Monthly Weather Review*, 137(9), 3137-3148.

See Also

[PT2Theta](#) and [moistCp](#)

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aP0<-aPs[1]
aT0<-C2K(RadiosondeA[1,3])
aw0<-RadiosondeA[1,6]/1000
deltaP=1
Na=length(aPs)
Ptop=aPs[Na]
fndlcl=find_lcl(Ptop,aP0,aT0,aw0,deltaP)
TLCL=fndlcl$Tlcl
equivalentPotentialTemperature(aP0,aT0,aw0,TLCL)
```

export_constants	<i>Export the constants</i>
------------------	-----------------------------

Description

This function exports to R the constants frequently used in the C part of *aiRthermo* for consistency.

Usage

```
export_constants()
```

Details

The constants stored in the vector are (in SI units): the gas constant for dry air R_d and for water vapour R_v ($\frac{J}{Kkg}$), the temperature T_0 corresponding to 0 degree Celsius, e_{s0} used to calculate the saturated vapour pressure (Pa), 1000 hPa in Pa (P1000), the specific heat of dry air for constant pressure c_p ($\frac{J}{Kkg}$) and for constant volume c_v ($\frac{J}{Kkg}$), acceleration due to gravity at sea level g ($\frac{m}{s^2}$), our definition of a missing value MISSING_VALUE (-99999999) and epsilon ε ($\frac{R_d}{R_v}$).

Constants are taken from Bohren & Albrecht (1998), and they are also consistent with those used in Petty (2008), Erukhimova & North (2009) and Davies-Jones (2009).

References

Bohren, C.F., & Albrecht, B. A. (1998). Atmospheric thermodynamics. Atmospheric thermodynamics. Publisher: New York; Oxford: Oxford University Press, 1998. ISBN: 0195099044.

Petty, G.W. (2008). A First Course in Atmospheric Thermodynamics, Sundog Publishing, Madison.

North, G. R. , Erukhimova, T. L. (2009). Atmospheric Thermodynamics, Cambridge University Press, New York.

Davies-Jones, R. (2009). On formulas for equivalent potential temperature, Monthly Weather Review, 137,3137-3148. doi:10.1175/2009MWR2774.1.

See Also

[aiRthermoConstants](#)

Examples

```
aiRthermoConstants<-export_constants()
```

export_lines	<i>Export the lines for the thermodynamic diagram</i>
--------------	---

Description

This function exports the *fixedlines* for Stüve Diagram. It includes the data for plotting the pseudo-adiabatic (adiabat_*_T), dry adiabatic (theta_*_T) and constant mixing ratio lines (wsat_*_T).

Usage

```
export_lines()
```

See Also

[fixedlines](#)

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
stuveA<-stuve_diagram(Pres = aPs/100,Temp=aTs-273.15)
```

find_lcl	<i>Calculation of the Lifted Condensation Level (LCL)</i>
----------	---

Description

For a particle with initial conditions P_0 , T_0 and w_0 , this function performs an adiabatic vertical evolution until it gets saturated at most when Ptop is reached.

Usage

```
find_lcl(Ptop, P0, T0, w0, deltaP)
```

Arguments

Ptop	Maximun level pressure selected (Pa)
P0	Initial value of pressure (Pa)
T0	Initial value of temperature (K)
w0	Initial value of mixing ratio (kg/kg)
deltaP	The width (Pa) of the layers used in the calculation of the numerical solution for the vertical evolution. A default value of 5 Pa is used.

Value

Returns a list including the following values:

<code>Plcl</code>	The pressure at LCL (Pa)
<code>Tlcl</code>	The temperature at LCL (K)
<code>wlcl</code>	The mixing ratio at LCL (kg/kg)
<code>thetalcl</code>	The potential temperature at LCL (K)
<code>gotit</code>	0 or 1 whether the particle arrived or not to saturation (LCL) before arriving to <code>Ptop</code>

Examples

```
Ptop=50000
P0=101325
T0=273.15
w0=0.0025
deltaP=5
rh=100*w0/saturation_mixing_ratio(P0,T0,export_constants())
fndlcl=find_lcl(Ptop,P0,T0,w0,deltaP)
```

fixedlines

Data for plotting the lines of the thermodynamic (STUVE) diagram

Description

The vectors included in the list are: both components of the pseudoadiabatic lines (`adiabatic_x_T`, and `adiabatic_y_T`), labels of the pseudoadiabatic lines (`adiabatic_z_T`), both components of the dry adiabatic lines (`theta_x_T` and `theta_y_T`), both components of the constant mixing ratio lines (`wsat_x_T` and `wsat_y_T`) and their labels (`wsat_z_T`). The X components are provided in Celsius and the Y components in hPa.

Usage

```
data(fixedlines)
```

Details

The pseudoadiabatic lines were calculated by the authors for this R-package following pseudoadiabatic evolutions from 1050 hPa.

The dry adiabatic lines were obtained using the functions in *aiRthermo* for different initial conditions and for a fixed set of initial potential temperatures. A similar procedure was applied on the calculation of the constant mixing ratio lines, starting from different values of saturation mixing ratio.

Source

The data were calculated by the authors for this R-package

See Also[export_lines](#)**Examples**

```
data(fixedlines)
```

gamma_saturated	<i>Saturated Adiabatic Gamma</i>
-----------------	----------------------------------

Description

Saturated adiabat at the points of the sounding as computed internally, considering hydrostatic balance and as $\frac{dT}{dP}$ (in pressure levels) (K/Pa)

Usage

```
gamma_saturated(Ps, Temps)
```

Arguments

Ps	A vector with pressure values (Pa)
Temps	A vector with temperature values (K)

Value

This function returns the vertical derivative $\Gamma_s = \left. \frac{dT}{dP} \right|_s$ for a saturated adiabatic evolution

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
gamma_saturated(aPs,aTs)
```

K2C

From Kelvin to Celsius degrees

Description

This function makes the transformation from Kelvin degrees to Celsius

Usage

```
K2C(Tk, consts = export_constants())
```

Arguments

Tk A vector of temperatures in Kelvin degrees
consts This function uses the T_0 constant corresponding to 0 degree Celsius as K

Value

This function returns a vector of temperatures in Celsius degrees

See Also

[aiRthermoConstants](#) and [C2K](#)

Examples

```
data(RadiosondeD)  
dT_s<-RadiosondeD[,3]  
K2C(C2K(dTs))
```

Kindex*K Instability Index*

Description

This function calculates the K instability index (Celsius) from a sounding given by the measured arrays pressure Ps (Pa) temperature Ts (K) and mixing ratio ws (kg/kg).

Usage

```
Kindex(Ps, Ts, ws, doLog = 0)
```

Arguments

Ps	A vector with pressure values (Pa) measured by the radiosonde.
Ts	A vector with temperature values (K) measured by the radiosonde.
ws	A vector with mixing ratio values (kg/kg) measured by the radiosonde.
doLog	Use logarithmic vertical interpolation between sounding levels. The default value is 0.

Details

If needed levels (850, 700 and 500 hPa) are not found in the input sounding (without extrapolation), the function returns -99999999. Use/do not use logarithmic interpolation in pressure (if needed because mandatory levels such as 700 hPa or 500 hPa are not given in the sounding) when finding the requested levels.

Value

This function returns the K index

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
aK<-Kindex(aPs,aTs,aws,0)

data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTd<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dK<-Kindex(dPs,dTd,dws,0)
```

 latent_heat_H2O

Latent heat of vaporization or sublimation of water

Description

This function calculates the latent heat of vaporization or sublimation of water depending as a function of temperature. It uses a polynomial approximation over water or ice.

Usage

```
latent_heat_H2O(Temps)
```

Arguments

Temps	A vector with temperature values (K)
-------	--------------------------------------

Details

Taking into account the observed values in tables from Rogers and Yau (1989) and Feistel and Wagner (2006), a polynomial model is used to calculate the latent heat at different temperatures

Value

This function returns the latent heat of vaporization or sublimation of water

References

Rogers, R. R., and Yau, M. K. (1989). A Short Course in Cloud Physics, 3rd Edition, Pergamon Press, Oxford.

Feistel, R. and Wagner, W. (2006). A new equation of state for H₂O ice Ih, Journal of Physical and Chemical Reference Data 35 1021-1047. doi:10.1063/1.2183324.

Examples

```
data(RadiosondeA)
aTs<-C2K(RadiosondeA[,3])
latent_heat_H2O(aTs)
```

 LIindex

Lifted index

Description

This function calculates the instability parameter Lifted index (Celsius) from pressure, temperature and mixing ratio values described by a vertical sounding.

Usage

```
LIindex(Ps, Ts, ws, Psurface, deltaP, PWIDTH, doLog = 0)
```

Arguments

Ps	Pressure (Pa) of the sounding
Ts	Temperature (K) of the sounding
ws	Mixing ratio (kg/kg) of the sounding
Psurface	Surface pressure (Pa). If not available, the first level of the sounding can be used.
deltaP	The width (Pa) of the layers used in the numerical solution of the vertical evolution (integration of the ODE). A default value of 5 Pa is used. It must be positive.
PWIDTH	PWIDTH represents the width (Pa) of the lower layer that will be averaged for P, T and w in order to calculate a "mixed-layer" average parcel that will be used for the vertical evolution. Typically 5000-10000 Pa are used.
doLog	Use logarithmic vertical interpolation between sounding levels if doLog=1. It is not used by default (doLog=0).

Details

If the 500 hPa needed level is not exactly found in the input sounding, logarithmic/linear vertical interpolation is run to get the corresponding T/w from the $P_s/T_s/w_s$ depending on the value of $doLog$ 0/1.

The evolution of the lifted particle is computed by integrating the dT/dP ordinary differential equation (applying the Runge-Kutta 4th order method), that represents the vertical adiabatic evolution from the initial condition to 500 hPa using a pressure step ΔP (Pa). The vertical adiabatic evolution is either dry (before saturation) or pseudoadiabatic at every vertical step with a correction for moisture in c_p using the value of the mixing ratio (c_{pm} as in Tsonis, eq 7.11).

If the sounding does not enclose the needed level of 500 hPa and the interpolation fails, the function returns -99999999.

Value

This function returns the LI index (Celsius)

References

Tsonis, A. A. (2002). An Introduction to Atmospheric Thermodynamics, Cambridge University Press, Cambridge. Eq. 7.11.

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
LIindex(aPs,aTs,aws,max(aPs),5,2500,0)
```

moistAdiabaticLapseRate

Moist Adiabatic Lapse Rate

Description

This function calculates the moist adiabatic lapse rate according to a provided mixing ratio (kg/kg). (Tsonis, eq 7.29)

Usage

```
moistAdiabaticLapseRate(w, consts = export_constants())
```

Arguments

w A vector with mixing ratio values (kg/kg)
consts The constants defined in *aiRthermoConstants* data are necessary.

Value

This function returns a vector with the moist adiabatic lapse rate (dry adiabatic lapse rate with correction of c_p due to the water vapour in moist air).

References

Tsonis, A. A. (2002). An Introduction to Atmospheric Thermodynamics, Cambridge University Press, Cambridge. Eq. 7.29.

Examples

```
data(RadiosondeA)
aws<-RadiosondeA[,6]/1000
moistAdiabaticLapseRate(aws)
```

moistCp	<i>Moist Cp</i>
---------	-----------------

Description

This function corrects the value of dry c_p due to the existence of water vapour according to equation 7.11 from Tsonis (2002).

Usage

```
moistCp(w, consts = export_constants())
```

Arguments

w	A vector with mixing ratio values (kg/kg)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns the value of dry c_p corrected by the mixing ratio.

References

Tsonis, A. A. (2002). An Introduction to Atmospheric Thermodynamics, Cambridge University Press, Cambridge. Eq. 7.11.

See Also

[w2q](#) and [moistCv](#)

Examples

```
data(RadiosondeD)
dws<-RadiosondeD[,6]/1000
moistCp(dws)
```

moistCv	<i>Moist cv value</i>
---------	-----------------------

Description

This function is similar to [moistCp](#) but for c_v . In this case, it is the value of c_v corrected due to the existence of water vapour (equation 7.12) from Tsonis (2002).

Usage

```
moistCv(w, consts = export_constants())
```

Arguments

w	A vector with mixing ratio values (kg/kg)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns the value of c_v corrected due to the existence of water vapour

References

Tsonis, A. A. (2002). An Introduction to Atmospheric Thermodynamics, Cam- 563 bridge University Press, Cambridge. Eq. 7.12.

See Also

[w2q](#) and [moistCp](#)

Examples

```
data(RadiosondeD)
dws<-RadiosondeD[,6]/1000
moistCv(dws)
```

parcelState	<i>State of a parcel</i>
-------------	--------------------------

Description

The function calculates the state of a parcel for easier computations

Usage

```
parcelState(Press, Temp, w = 0, consts = export_constants())
```

Arguments

Press	Value of pressure (Pa) of the parcel
Temp	Value of temperature (K) of the parcel
w	Value of mixing ratio (kg/kg) of the parcel
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a list including the following values:

pressure	Pressure value (Pa)
temperature	Temperature value (K)
mixingratio	Mixing ratio value (kg/kg)
theta	Potential temperature value (K)
virtualTemp	Virtual temperature value (K)
saturationMixingRatio	Saturation mixing ratio value (kg/kg)

See Also

[PT2Theta](#), [virtual_temperature](#) and [saturation_mixing_ratio](#)

Examples

```
parcelState(101325, 273.15, 0.2)
```

PT2Theta	<i>Potential Temperature from pressure and temperature</i>
----------	--

Description

This function calculates the potential temperature from given temperature and pressure

Usage

```
PT2Theta(P, Temp, w = 0, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
Temp	A vector with temperature values (K)
w	A vector with mixing ratio values (kg/kg). Default value 0.
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with potential temperature. Mixing ratio is only used to correct the value of c_p , not to calculate a moist adiabatic evolution.

See Also

[moistCp](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dT_s<-C2K(RadiosondeD[,3])
dThetas=PT2Theta(dPs,dTs)
```

PTheta2T	<i>Temperature from pressure and potential temperature</i>
----------	--

Description

This function calculates the temperature from a given pressure and potential temperature

Usage

```
PTheta2T(P, Theta, w = 0, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa).
Theta	A vector with potential temperature (K).
w	A vector with mixing ratio values (kg/kg). Default value 0.
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with temperatures (K).

See Also

[moistCp](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dThetas=PT2Theta(dPs,dTs)
PTheta2T(dPs,dThetas)
```

 PW

Vertically integrated water vapour column

Description

This function calculates the vertically integrated water vapour column integrating in pressure vertical coordinates

Usage

```
PW(w, PRES, Psurf, consts = export_constants())
```

Arguments

w	A vector with mixing ratio values of a sounding (kg/kg)
PRES	A vector with pressure values of a sounding (Pa)
Psurf	Is the mean sea level pressure at the place of a sounding (Pa)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns the vertically integrated water vapour column.

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dws<-RadiosondeD[,6]/1000
PW(dws, dPs, dPs[1])
```

q2e

Partial Vapour Pressure

Description

This function calculates the partial vapour pressure from specific humidity

Usage

```
q2e(P, q, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
q	A vector with specific humidity values (kg/kg)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns the value of the partial vapour pressure (Pa)

Examples

```
# Get partial pressure of water vapour
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dws<-RadiosondeD[,6]/1000
h2oe<-q2e(dPs, w2q(dws))
```

q2w *Water vapour mixing Ratio to specific humidity*

Description

This function calculates the water vapour mixing ratio (kg/kg) from specific humidity (kg/kg).

Usage

```
q2w(q)
```

Arguments

q A vector with specific humidity values (kg/kg)

Value

This function returns a vector with mixing ratio values in kg/kg

Examples

```
data(RadiosondeD)
dws<-RadiosondeD[,6]/1000
q2w(w2q(dws))
```

RadiosondeA *Radiosonde A*

Description

Contains the information measured by a sounding in Santander (Station 08023) in 2010, June 16th at 12:00 UTC. It was not a really unstable day but a great amount of (large scale) precipitation was measured.

Usage

```
data("RadiosondeA")
```

Format

A data frame with 74 observations on the following 11 variables.

V1 a vector with pressure values (hPa)

V2 a vector with height (m)

V3 a vector with temperature values (Celsius)

V4 a vector with dew point temperature values (Celsius)

- V5 a vector with relative humidity values (%)
- V6 a vector with mixing ratio values (g/kg)
- V7 a vector with wind direction values (degrees)
- V8 a vector with wind speed values (knots)
- V9 a vector with potential temperature (K)
- V10 a vector with equivalent potential temperature (K)
- V11 a vector with virtual potential temperature (K)

See Also

[RadiosondeD](#) and [RadiosondeDavenport](#)

Examples

```
data(RadiosondeA)
#Calculate the pressure in Pa
RadiosondeA$V1*100

#Calculate the temperature in K
C2K(RadiosondeA$V3)
```

RadiosondeD

Radiosonde D

Description

Contains the information measured by a sounding in Barcelona (station 05190) in 2013, August 7th at 12:00 UTC. According to the university of Wyoming, the CAPE was higher than 3000 J/kg and a great amount of (convective) precipitation was measured.

Usage

```
data("RadiosondeD")
```

Format

A data frame with 70 observations on the following 11 variables.

- V1 a vector with pressure values (hPa)
- V2 a vector with height (m)
- V3 a vector with temperature values (Celsius)
- V4 a vector with dew point temperature values (Celsius)
- V5 a vector with relative humidity values (%)
- V6 a vector with mixing ratio values (g/kg)
- V7 a vector with wind direction values (degrees)

V8 a vector with wind speed values (knots)
V9 a vector with potential temperature (K)
V10 a vector with equivalent potential temperature (K)
V11 a vector with virtual potential temperature (K)

See Also

[RadiosondeA](#) and [RadiosondeDavenport](#)

Examples

```
data(RadiosondeD)
#Calculate the pressure in Pa
RadiosondeD$V1*100

#Calculate the temperature in K
C2K(RadiosondeD$V3)
```

RadiosondeDavenport *Radiosonde Davenport*

Description

Contains the information measured by a sounding in Davenport (station 74455) in 1997, June 21st at 00:00 UTC. That day was a very unstable situation.

Usage

```
data("RadiosondeDavenport")
```

Format

A data frame with 67 observations on the following 11 variables.

V1 a vector with pressure values (hPa)
V2 a vector with height (m)
V3 a vector with temperature values (Celsius)
V4 a vector with dew point temperature values (Celsius)
V5 a vector with relative humidity values (%)
V6 a vector with mixing ratio values (g/kg)
V7 a vector with wind direction values (degrees)
V8 a vector with wind speed values (knots)
V9 a vector with potential temperature (K)
V10 a vector with equivalent potential temperature (K)
V11 a vector with virtual potential temperature (K)

See Also

[RadiosondeA](#) and [RadiosondeD](#)

Examples

```
data(RadiosondeDavenport)
#Calculate the pressure in Pa
RadiosondeDavenport$V1*100

#Calculate the temperature in K
C2K(RadiosondeDavenport$V3)
```

 rh2shum

Specific Humidity from relative humidity

Description

This function calculates the specific humidity from a given relative humidity

Usage

```
rh2shum(P, Temp, rh, consts = export_constants())
```

Arguments

P	A vector with pressure values (Pa)
Temp	A vector with temperature values (Kelvin)
rh	A vector with relative humidity values (%)
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with specific humidity (kg/kg)

See Also

[rh2shum](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTds<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dTds<-w2Td(dPs,dws)
rhs<-TTdP2rh(dTs,dTds,dPs)
rh2shum(dPs,dTs,rhs)
```

`rh2w`*Mixing Ratio from relative humidity*

Description

This function gets the mixing ratio (kg/kg) from a given relative humidity (%), pressure (Pa) and temperature (K).

Usage

```
rh2w(P, Temp, rh, consts = export_constants())
```

Arguments

<code>P</code>	A vector with pressure values in Pa
<code>Temp</code>	A vector with temperature values in Kelvin
<code>rh</code>	A vector with relative humidity values in (%)
<code>consts</code>	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with mixing ratio values (kg/kg).

See Also

[saturation_mixing_ratio](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dTds<-w2Td(dPs,dws)
rhs<-TTdP2rh(dTs,dTds,dPs)
wfromrh<-rh2w(dPs,dTs,rhs)
```

saturation_mixing_ratio
Saturation Mixing Ratio

Description

This function calculates the saturation mixing ratio from a given temperature and pressure

Usage

```
saturation_mixing_ratio(P, Temp, consts = export_constants())
```

Arguments

P	A vector with pressure values in Pa
Temp	A vector with temperature values in Kelvin
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with saturation mixing ratio values (kg/kg)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
saturation_mixing_ratio(dPs,dTs)
```

saturation_pressure_H2O
Saturation Pressure

Description

This function returns the saturation pressure (Pa) from a given array of temperatures (K). It uses approximate equations 5.67 and 5.70 in Bohren Albrecht, 1998.

Usage

```
saturation_pressure_H2O(Temps)
```

Arguments

Temps	A vector with temperature values in Kelvin
-------	--

Details

Saturation pressure of water vapour e_s is computed over ice/water depending whether the temperature is over/under 273.15 K (0 degree Celsius) at every element of the array.

Value

This function returns a vector with saturation pressure values (Pa)

References

Bohren, C.F., & Albrecht, B. A. (1998). Atmospheric thermodynamics. Atmospheric thermodynamics. Publisher: New York; Oxford: Oxford University Press, 1998. ISBN: 0195099044. Equations 5.67 and 5.70.

Examples

```
data(RadiosondeA)
aTs<-C2K(RadiosondeA[,3])
esats<-saturation_pressure_H20(aTs)
```

 Sindex

Showalter Instability Index

Description

This function computes Showalter instability index (Celsius) from given parameters from a vertical sounding pressure (Pa), temperature (K) and mixing ratio (kg/kg).

Usage

```
Sindex(Ps, Ts, ws, deltaP, doLog = 0)
```

Arguments

Ps	A vector with pressure values in Pa
Ts	A vector with temperature values in Kelvin
ws	A vector with mixing ratio values in kg/kg
deltaP	The width (Pa) of the layers used in the numerical solution of the vertical evolution.
doLog	Use logarithmic vertical interpolation between sounding levels. A default value is 0.

Details

If the needed levels (850 hPa or 500 hPa) are not exactly found in the input sounding, logarithmic/linear vertical interpolation is run to get the corresponding T/w from the Ps/Ts/ws depending on the value of doLog (TRUE or FALSE).

The evolution of the lifted particle is computed by integrating the dT/dP ordinary differential equation (applying the Runge Kutta 4th order method) that represents the vertical adiabatic evolution from 850 hPa to 500 hPa using a pressure step dP > 0 (Pa). The vertical adiabatic evolution is either dry (before saturation) or pseudoadiabatic at every vertical step with a correction for moisture in the specific heat at constant pressure c_p during the dry steps (as in Tsonis, eq 7.11).

If the sounding does not enclose the needed levels and the interpolation fails, the function returns -99999999

Value

This function returns the Showalter instability index (Celsius)

References

Djuric D. (1994). Weather Analysis, Prentice Hall, New Jersey.

See Also

[Lindex](#)

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
S<-Sindex(aPs,aTs,aws,5,0)
```

stuve_diagram

Thermodynamic (STUVE) Diagram

Description

This function generates an Stüve diagram

Usage

```
stuve_diagram(Pres, Temp, TempD = NA, XLIM = c(-80, 45), YLIM = c(1050, 100))
```

Arguments

Pres	A vector with pressure values in hPa.
Temp	A vector with temperature values in Celsius .
TempD	An optional vector with dew point temperatures in Celsius. The default value is NA.
XLIM	X axis limit in Celsius. Default value is c(-80, 45)
YLIM	Y axis limit in hPa. Default value is c(1050, 100)

Details

It is possible to add extra lines and to save as a pdf, jpeg or png (see examples).

Value

The result is a plot object.

Examples

```
data(RadiosondeA)
aPs<-RadiosondeA[,1]*100
aTs<-C2K(RadiosondeA[,3])
aws<-RadiosondeA[,6]/1000
capeCin<-CAPE_CIN(PlowTop=98000,precoolType="adiabatic",
                  Ps=aPs,Ts=aTs,ws=aws,dolog=0,deltaP=5,
                  getLiftedBack=TRUE,upToTop=TRUE)

#How to add a line to the plot
stuveA<-stuve_diagram(Pres = aPs/100,Temp=aTs-273.15)
lines(capeCin$Tl-273.15,capeCin$P1/100,col="red",lwd=2)
```

TTdP2rh	<i>Relative Humidity from temperature, pressure and dew point temperature</i>
---------	---

Description

This function calculates the relative humidity from given temperature, dew point temperature and pressure.

Usage

```
TTdP2rh(Temp, Td, P, consts = export_constants())
```

Arguments

Temp	A vector with temperature values in Kelvin
Td	A vector with dew point temperature in Kelvin
P	A vector with pressure values in Pa
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with relative humidity values

See Also

[saturation_mixing_ratio](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dTds<-w2Td(dPs,dws)
rhs<-TTdP2rh(dTs,dTds,dPs)
```

TTheta2P

Pressure from temperature and potential temperature

Description

This function calculates the pressure from given potential temperature and temperature, assuming a dry adiabatic evolution (mixing ratio is only used to correct the values of c_p).

Usage

```
TTheta2P(Temp, Theta, w = 0, consts = export_constants())
```

Arguments

Temp	A vector with temperature values in Kelvin
Theta	A vector with potential temperature values in Kelvin
w	Initial value of mixing ratio (kg/kg). By default 0.
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with pressure values

Examples

```

data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
dTds<-w2Td(dPs,dws)
dThetas<-PT2Theta(dPs,dTs)
TTheta2P(dTs,dThetas)

```

TTindex

Total-Totals Instability Index

Description

Total-Totals instability index (Celsius) from parameters (1D arrays) Ps (pressure, Pa) Ts (temperature, Kelvin) and ws (mixing ratio, kg/kg) obtained from a vertical sounding.

Usage

```
TTindex(Ps, Ts, ws, doLog = 0)
```

Arguments

Ps	A vector with pressure values in Pa
Ts	A vector with temperature values in Kelvin
ws	A vector with mixing ratio values in kg/kg
doLog	Use logarithmic vertical interpolation between sounding levels. A default value is 0.

Details

If the needed levels (850 hPa or 500 hPa) are not exactly found in the input sounding, logarithmic/linear vertical interpolation is run depending on the value of doLog (TRUE or FALSE)

If the sounding does not enclose the needed levels and the interpolation fails, the function returns -99999999.

Value

This function returns the Total-Totals instability index (Celsius)

Examples

```

data(RadiosondeDavenport)
aPs<-RadiosondeDavenport[,1]*100
aTs<-C2K(RadiosondeDavenport[,3])
aws<-RadiosondeDavenport[,6]/1000
aTT<-TTindex(aPs,aTs,aws,0)

```

virtual_temperature *Virtual Temperature*

Description

This function calculates the virtual temperature from given pressure and mixing ratio.

Usage

```
virtual_temperature(P, Temp, w, consts = export_constants())
```

Arguments

P	A vector with pressure values in Pa
Temp	A vector with temperature values in Kelvin
w	A vector with mixing ratio values in kg/kg
consts	The constants defined in <i>aiRthermoConstants</i> data are necessary.

Value

This function returns a vector with virtual temperature values

See Also

[q2e](#)

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dTs<-C2K(RadiosondeD[,3])
dws<-RadiosondeD[,6]/1000
virtual_temperature(dPs,dTs,dws)
```

w2q *Specific Humidity from mixing ratio*

Description

This function calculates the specific humidity from a given Water mixing ratio.

Usage

```
w2q(w)
```

Arguments

w A vector with mixing ratio values in kg/kg

Value

The function returns a vector with the specific humidity

Examples

```
data(RadiosondeD)
dws<-RadiosondeD[,6]/1000
w2q(dws)
```

w2Td *Dew point temperature from mixing ratio*

Description

This function calculates the dew point temperature from given mixing ratio and pressure, following the APPROXIMATE expression 5.68 in Bohren and Albrech (1998).

Usage

```
w2Td(P, w, consts = export_constants())
```

Arguments

P A vector with pressure values in Pa
w A vector with mixing ratio (kg/kg)
consts The constants defined in *aiRthermoConstants* data are necessary.

Value

This function returns a vector with the dew point temperature

References

Bohren, C.F., & Albrecht, B. A. (1998). Atmospheric thermodynamics. Atmospheric thermodynamics. Publisher: New York; Oxford: Oxford University Press, 1998. ISBN: 0195099044. Equation 5.68.

Examples

```
data(RadiosondeD)
dPs<-RadiosondeD[,1]*100
dws<-RadiosondeD[,6]/1000
w2Td(dPs, dws)
```

Index

*Topic **Datasets**

- aiRthermoConstants, 4
- fixedlines, 18
- RadiosondeA, 30
- RadiosondeD, 31
- RadiosondeDavenport, 32

*Topic **Functions**

- adiabatic_ascent, 3
- AnyAdiabaticDown, 5
- boltonTLCL, 6
- bruntVaisallaOmegaSquared, 7
- C2K, 8
- CAPE_CIN, 9
- densityDry, 11
- densityH2Ov, 12
- densityMoistAir, 12
- dewpointdepression2rh, 13
- e2w, 14
- equivalentPotentialTemperature, 15
- export_constants, 16
- export_lines, 17
- find_lcl, 17
- gamma_saturated, 19
- K2C, 20
- Kindex, 20
- latent_heat_H2O, 21
- LIindex, 22
- moistAdiabaticLapseRate, 23
- moistCp, 24
- moistCv, 25
- parcelState, 26
- PT2Theta, 27
- PTheta2T, 27
- PW, 28
- q2e, 29
- q2w, 30
- rh2shum, 33
- rh2w, 34
- saturation_mixing_ratio, 35

- saturation_pressure_H2O, 35
- Sindex, 36
- stuve_diagram, 37
- TTdP2rh, 38
- TTheta2P, 39
- TTindex, 40
- virtual_temperature, 41
- w2q, 41
- w2Td, 42

*Topic **Package**

- aiRthermo-package, 2

- adiabatic_ascent, 3
- aiRthermo (aiRthermo-package), 2
- aiRthermo-package, 2
- aiRthermoConstants, 4, 8, 16, 20
- AnyAdiabaticDown, 5

- boltonTLCL, 3, 6
- bruntVaisallaOmegaSquared, 7

- C2K, 8, 20
- CAPE_CIN, 9

- densityDry, 11
- densityH2Ov, 12
- densityMoistAir, 7, 12
- dewpointdepression2rh, 13

- e2w, 14
- equivalentPotentialTemperature, 15
- export_constants, 16
- export_lines, 17, 19

- find_lcl, 17
- fixedlines, 17, 18

- gamma_saturated, 19

- K2C, 8, 20
- Kindex, 20

latent_heat_H2O, [6](#), [21](#)
LIindex, [22](#), [37](#)

moistAdiabaticLapseRate, [23](#)
moistCp, [15](#), [24](#), [25](#), [27](#), [28](#)
moistCv, [24](#), [25](#)

parcelState, [26](#)
PT2Theta, [7](#), [15](#), [26](#), [27](#)
PTheta2T, [27](#)
PW, [28](#)

q2e, [12](#), [29](#), [41](#)
q2w, [30](#)

RadiosondeA, [30](#), [32](#), [33](#)
RadiosondeD, [31](#), [31](#), [33](#)
RadiosondeDavenport, [31](#), [32](#), [32](#)
rh2shum, [33](#), [33](#)
rh2w, [34](#)

saturation_mixing_ratio, [13](#), [26](#), [34](#), [35](#),
[39](#)
saturation_pressure_H2O, [13](#), [35](#)
Sindex, [36](#)
stuve_diagram, [3](#), [37](#)

TTdP2rh, [38](#)
TTheta2P, [39](#)
TTindex, [40](#)

virtual_temperature, [13](#), [26](#), [41](#)

w2q, [12](#), [24](#), [25](#), [41](#)
w2Td, [42](#)