Package 'RNCEP'

May 7, 2026

Type Package

Title Obtain, Organize, and Visualize NCEP Weather Data

Version 1.0.11

Date 2025-01-22

Maintainer Michael U. Kemp <mukemp+RNCEP@gmail.com>

Description Contains functions to retrieve, organize, and visualize weather data from the NCEP/NCAR Reanalysis (<https://psl.noaa.gov/data/gridded/data.ncep.reanalysis.html>) and NCEP/DOE Reanalysis II (<https://psl.noaa.gov/data/gridded/data.ncep.reanalysis2.html>) datasets. Data are queried via the Internet and may be obtained for a specified spatial and temporal extent or interpolated to a point in space and time. We also provide functions to visualize these weather data on a map. There are also functions to simulate flight trajectories according to specified behavior using either NCEP wind data or data specified by the user.

License GPL (>= 2)

Imports RColorBrewer, abind, fields, tgp, tcltk, graphics, sp

LazyLoad yes

LazyData true

Depends R (>= 2.10), maps

URL https://psl.noaa.gov/data/gridded/index.html

https://sites.google.com/site/michaelukemp/home

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-23 14:00:03 UTC

Author Michael U. Kemp [aut, cre],
E. Emiel van Loon [ths],
Judy Shamoun-Baranes [ths],
Willem Bouten [ths]

Contents

`RNCEP-package` *This package of functions retrieves, organizes, and visualizes weather data from either the NCEP/NCAR Reanalysis or NCEP/DOE Reanalysis II datasets*

Description

This package contains functions to...

1. Query data from these two NCEP datasets for a specified range of space and time, maintaining both the spatial and temporal structure of the data
2. Remove any unwanted time intervals of the returned data
3. Temporally aggregate the data and apply any function to the subsets (i.e. calculate user-defined climatic variables)
4. Create a contour map from the data
5. Query data from these two NCEP datasets interpolated to a particular point in time and space
6. Visualize these interpolated data as points on a map using color to represent the interpolated value
7. Perform trajectory simulations according to specified behavior using wind data from NCEP or data specified by the user.

Details

Package:	RNCEP
Type:	Package
Version:	1.0.11
Date:	2025-01-22
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., Emiel van Loon, E., Shamoun-Baranes, J., Bouten, W., 2012. RNCEP: global weather and climate data at your fingertips. Methods in Ecology and Evolution (3), 65-70., DOI: 10.1111/j.2041-210X.2011.00138.x

For more information on flow-assistance and NCEP.flight see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., McLaren, J. D., Dokter, A. M., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. Journal of Theoretical Biology. (308), 56-67. DOI 10.1016/j.jtbi.2012.05.026

To cite the NCEP/NCAR Reanalysis dataset use:

Kalnay et al. (1996), The NCEP/NCAR 40-year reanalysis project, Bull. Amer. Meteor. Soc., 77, 437-470

To cite the NCEP/DOE Reanalysis II dataset use:

Kanamitsu et al. (2002), NCEP-DOE AMIP-II Reanalysis (R-2). Bull. Amer. Meteor. Soc., 83, 1631-1643

Please acknowledge the use of NCEP data in any publications by including text such as, "NCEP Reanalysis data provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site at https://psl.noaa.gov/". They would also appreciate a copy of any publications using NCEP data.

Examples

```
## Not run:
library(RNCEP)
## Retrieve the temperature from a particular pressure level for
## a specified spatial and temporal extent
wx.extent <- NCEP.gather(variable='air', level=850,
    months.minmax=c(8,9), years.minmax=c(2000,2001),
    lat.southnorth=c(50,55), lon.westeast=c(0,5),
    reanalysis2 = FALSE, return.units = TRUE)
```

```
## Retrive the temperature from a particular pressure level
## interpolated in space and time
wx.interp <- NCEP.interp(variable='air', level=850, lat=55.1,
    lon=11.3, dt='2006-10-12 17:23:12')

## Simulate a flight trajectory using NCEP wind data
flight <- NCEP.flight(beg.loc=c(58.00,7.00),
  end.loc=c(53.00,7.00), begin.dt='2007-10-01 18:00:00',
  flow.assist='NCEP.Tailwind', fa.args=list(airspeed=12),
  path='loxodrome', calibrate.dir=FALSE, calibrate.alt=FALSE,
  cutoff=0, when2stop='latitude', levels2consider=c(850,925),
  hours=12, evaluation.interval=60, id=1, land.if.bad=FALSE,
  reanalysis2 = FALSE, query=TRUE)

## End(Not run)
```

---

gull *GPS data points from a lesser black-backed gull*

Description

This data set contains a collection of GPS coordinates obtained between June 2008 and July 2009. The GPS tag, developed by the University of Amsterdam, was worn by a lesser black-backed gull (Larus fuscus).

Usage

```
gull
```

Format

A data.frame containing 22115 observations of the varibles latitude, longitude, altitude, and date-time.

NCEP.aggregate *Temporally Aggregate Weather Data*

Description

This function temporally aggregates weather data from the NCEP/NCAR Reanalysis or NCEP/DOE Reanalysis II datasets as returned by NCEP.gather. The spatial structure of the data is retained. Data can be aggregated by any combination of year, month, day, and hour.

Usage

```
NCEP.aggregate(wx.data, YEARS = TRUE, MONTHS = TRUE, DAYS = TRUE,
    HOURS = TRUE, fxn = "%>0")
```

Arguments

wx.data	the 3-D array of weather data as returned by NCEP.gather to be aggregated
YEARS	Logical. Should the years portion of the datetime be retained in the aggregation?
MONTHS	Logical. Should the months portion of the datetime be retained in the aggregation?
DAYS	Logical. Should the days portion of the datetime be retained in the aggregation?
HOURS	Logical. Should the hours portion of the datetime be retained in the aggregation?
fxn	A scalar function to be applied to all aggregated subsets of the data.

Details

Each latitude and longitude combination in the array is subset according to the logical datetime arguements in the function call, and fxn is applied to the subsets. fxn can be an internal R function such as mean, sum, or sd, or it can be a function created by the user.

The default setting of fxn (i.e. "%>0") calculates the percentage of observations in each subset greater than zero.

To calculate some variables, sequential aggregations may be needed. For instance, if the user wants to calculate monthly averages of maximum daily relative humidity, two aggregations would be needed. In the first aggregation, the user would find the maximum relative humidity per day. A second aggregation would then be required to find the monthly average of those maximum daily values. See the examples below for a demonstration.

Value

A three-dimensional array (or 2-D if all layers are completely aggregated) containing the same latitude and longitude ranges and intervals as the input data specified in wx.data. The names of the aggregated time components will be replaced with "XX" or "XXXX" in the output array.

Author(s)

Michael U. Kemp < mukemp+RNCEP@gmail.com >

References

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
#######################################################
## In the first example, we use the internal R function
## 'mean' to calculate average temperatures per
## month and year
```

```
## First gather temperature data from a spatial extent
## for January and February from 2000-2001.
wx.extent <- NCEP.gather(variable='air', level=850,
    months.minmax=c(1,2), years.minmax=c(2000,2001),
    lat.southnorth=c(50, 55), lon.westeast=c(0, 5),
    reanalysis2=FALSE, return.units=TRUE)

## Now calculate the average temperature per month and year ##
wx.ag <- NCEP.aggregate(wx.data=wx.extent, YEARS=TRUE, MONTHS=TRUE,
    DAYS=FALSE, HOURS=FALSE, fxn='mean')

## Notice that aggregated time components have been replaced
## with "XX" or "XXXX" ##
dimnames(wx.ag)[[3]][1]

#########################################################
## In the second example, we create our own function to
## calculate the percentage of observations at each grid
## point with a temperature less than -5 degrees Celsius ##

## First create the function ##
## Note: temperature is in degrees Kelvin in NCEP database ##
COLD <- function(x){
    return(length(which(x < 273.15-5))/(length(x) - sum(is.na(x))))
    }

## Now calculate the percentage of occuence of temperatures
## less than -5 degrees Celsius per month and year ##
wx.cold <- NCEP.aggregate(wx.data=wx.extent, YEARS=TRUE, MONTHS=TRUE,
    DAYS=FALSE, HOURS=FALSE, fxn='COLD')


#########################################################
## As explained in the Details section above,
## calculating some variables requires sequential aggregations ##
## Here we calculate the monthly mean of daily average
## relative humidity.

## First gather relative humidity data from near the surface
## for a spatial extent for October through November
## from 2001-2002.
wx.extent <- NCEP.gather(variable='rhum.sig995', level='surface',
    months.minmax=c(10,11), years.minmax=c(2001,2002),
    lat.southnorth=c(50, 55), lon.westeast=c(0, 5),
    reanalysis2=FALSE, return.units=FALSE)

## First calculate maximum daily relative humidity ##
wx.ag <- NCEP.aggregate(wx.data=wx.extent,
    HOURS=FALSE, fxn='max')

## Then calculate the monthly average of those daily maximums ##
wx.ag2 <- NCEP.aggregate(wx.data=wx.ag,
    DAYS=FALSE, fxn='mean')
```

```
############################################################
## Data that have been aggregated may then be visualized
## or exported in various formats ##

#########################################
## Visualize the aggregated temperatures ##
NCEP.vis.area(wx.ag2, title.args=
    list(main='Monthly average of daily maximum relative humidity
    \n October 2000'), image.plot.args=
    list(legend.args=list(text='%',
    cex=1.15, padj=-1, adj=-.25)))

############################################################
## Export a layer of the data to a format that can then be
## imported into ArcGIS ##

## Convert the first layer of the aggregated array
## to a data.frame ##
wx.df <- NCEP.array2df(wx.ag[,,1])

## Specify that the data.frame is a spatial object
library(sp)
coordinates(wx.df) <- ~longitude+latitude
gridded(wx.df) <- TRUE
proj4string(wx.df) <- CRS('+proj=longlat + datum=WGS84')

## Save the data in .asc format
write.asciigrid(wx.df, fname='wx.asc')
## Note: Data will be written to your working directory ##

## The resulting .asc file can be imported into ArcMap
## using ArcMap's "ASCII to Raster" tool in the "Conversion Tools"
## section of the ArcToolbox. ##

## End(Not run)
```

`NCEP.Airspeed` *Calculate flow-assistance according to equation 'Airspeed'*

Description

This function calculates flow-assistance according to equation Airspeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation Airspeed.

Usage

```
NCEP.Airspeed(u, v, direction, airspeed,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating a preferred direction of movement in degrees from North.
airspeed	The airspeed (i.e. speed relative to the flow) of the animal in meters per second.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation Airspeed. Equation Airspeed stipulates that the animal fully compensates for any lateral drift, altering its heading and groundspeed (i.e. speed relative to the fixed Earth), in order to maintain its direction. If, with the given airspeed, the animal is incapable of maintaining its direction, the equation produces no real solution.

Value

A data.frame containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation Airspeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.Airspeed to calculate flow-assistance ##
tst <- NCEP.Airspeed(u=-2, v=-1, direction=225, airspeed=12)
```

NCEP.array2df *Convert data from an array to a data frame*

Description

This function takes a 3-dimensional array of weather data, as returned by NCEP.gather, and converts it to a dataframe composed of latitude, longitude, datetime, and the weather data.

Usage

```
NCEP.array2df(wx.data, var.names=NULL)
```

Arguments

wx.data	either a single 3-D array of weather data, as returned by NCEP.gather, or a list of multiple 3-D arrays of different variables but with the exact same spatial and temporal dimensions and intervals.
var.names	an optional vector of names for the weather variables in the 3-D arrays to be used in the output data frame

Details

This is a function to convert the data contained in one or many arrays to a single data.frame.

The order of the names specified in var.names should correspond to the order of the list of input arrays in wx.data. Latitude, longitude, and datetime are named automatically and cannot be changed in the function call.

When converting the data from two or more 3-D arrays to a single data frame, all 3-D arrays must contain the exact same spatial and temporal dimensions and intervals.

Value

A data frame with the components of latitudes, longitude, datetime, and the weather variable(s) from the input data array(s).

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
############################################################
############################################################
## In this first example, we take data from a single 3-D array
## and arrange them in a data.frame ##
############################################################

## First query the temperature for a particular pressure level
## and datetime range ##
wx.extent <- NCEP.gather(variable='air', level=850,
    months.minmax=c(8,9), years.minmax=c(2006,2007),
    lat.southnorth=c(50,55), lon.westeast=c(0,5),
    reanalysis2 = FALSE, return.units = TRUE)

## Then convert the 3-D array to a data.frame ##
wx.df <- NCEP.array2df(wx.data=wx.extent, var.names='Temperature')

############################################################
############################################################
## In this second example, we take data from two 3-D arrays
## and arrange them in a single data.frame ##
############################################################

## Query the U (east/west) and V (north/south) wind components
## for a particular pressure level and datetime range ##
wx.uwnd <- NCEP.gather(variable='uwnd', level=850,
    months.minmax=c(8,9), years.minmax=c(2006,2007),
    lat.southnorth=c(50,55), lon.westeast=c(0,5),
    reanalysis2 = FALSE, return.units = TRUE)
wx.vwnd <- NCEP.gather(variable='vwnd', level=850,
    months.minmax=c(8,9), years.minmax=c(2006,2007),
    lat.southnorth=c(50,55), lon.westeast=c(0,5),
    reanalysis2 = FALSE, return.units = TRUE)

## Then convert the two 3-D arrays to a single data.frame ##
wx.df <- NCEP.array2df(wx.data=list(wx.uwnd, wx.vwnd),
    var.names=c('Uwind', 'Vwind'))

## End(Not run)
```

NCEP.bind *Bind Two 3-D Arrays of Weather Data Along the Prime Meridian*

Description

This function is applied automatically by NCEP.gather whenever it is needed. It binds the results from either side of the Prime Meridian.

Usage

```
NCEP.bind(data.west, data.east)
```

Arguments

data.west	a 3-D array of weather data, as returned by NCEP.gather , from the West side of the Prime Meridian
data.east	a 3-D array of weather data, as returned by NCEP.gather , from the East side of the Prime Meridian

Details

This function is applied automatically by NCEP.gather whenever it is needed.

The arrays specified in data.west and data.east must have the same latitude and datetime intervals and extents.

This function depends on the package abind

The maximum longitudinal extent of the NCEP dataset is 357.5 not 360.

Value

A 3-D array with the same latitude and datetime intervals and extent as data.west and data.east. Row names (i.e. longitudes) for data from the west of the Prime Meridian are converted from positive to negative values.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
## Using NCEP.gather(), query weather data from both sides of
## the Prime Meridian ##
## NCEP.bind() is applied automatically ##
wx <- NCEP.gather(variable='air', level=925,
    months.minmax=10, years.minmax=2003,
    lat.southnorth=c(50, 52.5), lon.westeast=c(-2.5, 2.5),
    reanalysis2=FALSE, return.units=TRUE)

## End(Not run)
```

NCEP.flight *Simulate Trajectories*

Description

This function simulates a single trajectory for any time and location by default using wind data returned by NCEP.interp. Various flight behaviors are possible.

Usage

```
NCEP.flight(beg.loc, end.loc, begin.dt, flow.assist='NCEP.Tailwind',
  fa.args=list(airspeed=12), path='loxodrome', calibrate.dir=FALSE,
  calibrate.alt=TRUE, cutoff=0, when2stop=list('latitude','longitude',50),
  levels2consider=c(850,925), hours=12, evaluation.interval=60, id=1,
  land.if.bad=FALSE, reanalysis2 = FALSE, query=TRUE)
```

Arguments

beg.loc	A numeric vector of length two giving the starting location in decimal degrees i.e. c(latitude, longitude)
end.loc	A numeric vector of length two giving the end location (i.e. goal location) in decimal degrees i.e. c(latitude, longitude)
begin.dt	A character string in the format '%Y-%m-%d %H:%M:%S' indicating the day and time at which to initiate take-off.
flow.assist	A character string giving the name of the flow-assistance function to apply, which also specifies the behavior of the animal in relation to the flow conditions. See Details below.
fa.args	A list of arguments passed to the flow-assistance function. See Details below.
path	Either a numeric value indicating the preferred direction of movement in degrees from North, or one of 'loxodrome' or 'great.circle' describing the method of calculating the preferred direction.
calibrate.dir	A logical expression indicating whether or not to re-calibrate the path to the goal at each evaluation interval.
calibrate.alt	A logigal expression indicating whether or not to re-calibrate the altitude from which to obtain wind data at each evaluation interval.
cutoff	A numeric value indicating the minimum acceptable flow-assistance to initiate the simulation (and to continue the simulation if land.if.bad = TRUE).
when2stop	A list containing at least one of 'latitude', 'longitude', or a numeric value. See Details below.
levels2consider	A vector indicating the altitudes at which to consider wind conditions. Values may be numeric and describe specific pressure levels or may be a character string describing 'surface' and/or 'gaussian'. See NCEP.interp

hours — A numeric value indicating the maximum number of hours to continue the simulation, in case the spatial goal is not reached.

evaluation.interval — A numeric value indicating the interval (in minutes) in which to reassess the situation and recalibrate (if calibrate.dir or calibrate.alt is TRUE). Also the interval at which movement statistics are calculated.

id — A character string or numeric value giving an id for the simulation.

land.if.bad — A logical expression describing whether or not the simulation should be interrupted if the flow-assistance is NA or below the cutoff.

reanalysis2 — A logical expression indicating whether wind data should come from the Reanalysis I dataset (FALSE) or from Reanalysis II (TRUE). See NCEP.interp.

query — A logical expression indicating whether or not to use NCEP.interp to retrieve wind data. See Details below.

Details

This function simulates a single trajectory according to the behavioral rules specified in flow.assist.

The argument flow.assist gives the flow-assistance equation to apply. It also determines how the animal will behave in relation to the flow conditions and thus how the animal will move. Internal options for flow.assist include "NCEP.FlowSpeed", "NCEP.NegFlowSpeed", "NCEP.Tailwind", "NCEP.Groundspeed", "NCEP.M.Groundspeed", "NCEP.Airspeed",and "NCEP.PartialSpeed". Each flow-assistance equation requires further arguements that should be given as a list to fa.args.

Users are invited to create their own flow-assistance functions using one of the existing functions as a template. Any flow-assistance function that is used in the context of NCEP.flight must contain at least the arguments u, v, and direction, as these arguments are passed directly by NCEP.flight if query == TRUE. If query == FALSE, only the argument direction is passed directly by NCEP.flight. Any other arguments are possible, and their value may be set using fa.args. As well, any flow-assistance function that is used in the context of NCEP.flight must produce a data.frame as output. This data.frame must contain at least the variables fa, forward.move, and side.move, but may also contain as many other variables as desired. Note that forward.move and side.move should give forward and sideways speeds relative to the specified direction in meters per second and should already account for the animal's own airspeed.

If calibrate.dir is TRUE, the animal will adjust its direction at each evaluation.interval if necessary to reorient to its end.loc. The new direction is calculated according to the specification of path. If calibrate.dir is FALSE and path is not numeric, the direction to the end.loc is calculated before take-off but not again during the simulation. If path is 'loxodrome', the rhumb line or constant compass direction is calculated to the end.loc. See NCEP.loxodrome. If path is 'great.circle', the angle describing the shortest path to the end.loc is calculated using earth.bear. A warning is issued if path is numeric and calibrate.dir is TRUE.

If calibrate.alt is TRUE, the animal will select at each evaluation.interval from levels2consider the most supportive flow conditions according to the specified flow.assist equation. If calibrate.alt is FALSE and the length of levels2consider is greater than one, the animal will select from levels2consider the most supportive flow conditions according to the specified flow.assist equation only when initiating the simulation and will continue using conditions at that level thereafter.

If the list passed to when2stop includes the character strings 'latitude' or 'longitude' the simulation will stop when the latitude or longitude, respectively, of the end.loc is reached. A numeric argument passed to the when2stop list indicates a distance from the end.loc in kilometers at which to stop the simulation. The simulation will end when any of the when2stop arguments are satisfied.

The argument query indicates whether or not to use NCEP.interp to retrieve wind data from the NCEP database via the Internet. This is intended to facilitate the use of flow conditions from sources other than NCEP. For example, one could use the time and location of the animal at each timestep to retrieve data from an oceanic dataset or an atmospheric dataset with higher resolution than NCEP. Alternatively, one could examine the trajectory of an animal that encounters consistent flow conditions of a given intensity for its entire journey. Since the user supplies the flow information, there is no need to query wind data from NCEP, and the function will run much faster. See the examples for a demonstration.

Value

A data.frame containing at least the flow-assistance (i.e. fa), forward.move, and side.move each in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and NCEP.flight see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Use NCEP.flight(), to simulate a flight over the North Sea
## from Norway to the Netherlands using NCEP wind data
## queried automatically from the NCEP Reanalysis datset ##
## Not run: tst <- NCEP.flight(beg.loc=c(58.00,7.00),
  end.loc=c(53.00,7.00), begin.dt='2007-10-01 18:00:00',
  flow.assist='NCEP.Tailwind', fa.args=list(airspeed=12),
  path='loxodrome', calibrate.dir=FALSE, calibrate.alt=FALSE,
  cutoff=0, when2stop='latitude', levels2consider=c(850,925),
  hours=12, evaluation.interval=60, id=1, land.if.bad=FALSE,
  reanalysis2 = FALSE)
## End(Not run)


#########################################################
```

```
## Use NCEP.flight(), to simulate a flight without
## querying the NCEP database.  In this case, we use
## constant wind conditions (i.e. u=2 and v=0). ##
tst2 <- NCEP.flight(beg.loc=c(58.00,7.00), end.loc=c(53.00,7.00),
  begin.dt='2007-10-01 18:00:00', flow.assist='NCEP.Tailwind',
  fa.args=list(u=2, v=0, airspeed=12), path='loxodrome',
  calibrate.dir=TRUE, calibrate.alt=FALSE, cutoff=-10, when2stop='latitude',
  levels2consider=850, hours=12, evaluation.interval=60,
  id=1, land.if.bad=FALSE, reanalysis2 = FALSE, query=FALSE)

#########################################################
## Use NCEP.flight(), to simulate a flight without
## querying the NCEP database.  In this case, we use
## wind conditions that depend on the animal's location ##

## First create a flow assistance function that will
## generate u and v flow components based on the animal's
## location
## Note that to use the datetime and the animal's location,
## the function MUST have the arguments lat.x, lon.x, and dt.x
Ex.FA <- function(direction, lat.x, lon.x, dt.x, airspeed){
    deg2rad = pi/180
    rad2deg = 180/pi
    ## Generate U and V flow component based on lat and lon
    u.x <- sin(lat.x*deg2rad)
    v.x <- cos(lon.x*deg2rad)
    ## Apply NCEP.Tailwind with generated flow data
    return(cbind(NCEP.Tailwind(u=u.x, v=v.x,
        direction=direction, airspeed=airspeed), u.x, v.x))
    }

## Now use the function we just created in NCEP.flight
## The location and datetime are passed to the flow-
## assistance function automatically as lat.x, lon.x, and dt.x
tst3 <- NCEP.flight(beg.loc=c(58.00,7.00), end.loc=c(53.00,7.00),
  begin.dt='2007-10-01 18:00:00', flow.assist='Ex.FA',
  fa.args=list(airspeed=12), path='loxodrome',calibrate.dir=TRUE,
  calibrate.alt=FALSE, cutoff=-10, when2stop='latitude',
  levels2consider=850, hours=12, evaluation.interval=60,
  id=1, land.if.bad=FALSE, reanalysis2 = FALSE, query=FALSE)

## Confirm that the U and V wind components were determined
## by the latitude and longitude at each timestep
sin(tst3$lat[3] * (pi/180)) == tst3$u.x[3]
cos(tst3$lon[7] * (pi/180)) == tst3$v.x[7]
```

NCEP.FlowSpeed *Calculate flow-assistance according to equation 'FlowSpeed'*

Description

This function calculates flow-assistance according to equation FlowSpeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation FlowSpeed.

Usage

```
NCEP.FlowSpeed(u, v, direction, airspeed,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating the preferred direction of movement in degrees from North. For this function, direction is required but does not change the calculation of flow-assistance or the direction of movement. The argument is included for use with NCEP.flight.
airspeed	The airspeed of the animal in meters per second. This value does not affect the calculation of flow-assistance, but does affect the forward and sideways movement of the animal.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation FlowSpeed. Equation FlowSpeed assigns the speed of the flow (e.g. wind speed) as flow-assistance, therefore it can never produce negative flow-assistance values. It assumes that the animal applies its own airspeed in the same direction as the flow (e.g. wind direction).

Value

A data.frame containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation FlowSpeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.FlowSpeed to calculate flow-assistance ##
tst <- NCEP.FlowSpeed(u=-2, v=-1, direction=225, airspeed=12)
```

NCEP.gather *Queries Weather Data*

Description

This function queries weather data over the Internet from the NCEP/NCAR Reanalysis or NCEP/DOE Reanalysis II datasets for the spatial and temporal extents specified in the function call and maintains the spatial and temporal structure of the data in a 3-D array.

Usage

```
NCEP.gather(variable, level, months.minmax, years.minmax,
    lat.southnorth, lon.westeast, reanalysis2 = FALSE,
    return.units = TRUE, status.bar=TRUE)

## NCEP.gather is a wrapper function that calls one of the
## following functions based on the value of level.
## Users should avoid using these functions directly.
NCEP.gather.gaussian(variable, months.minmax, years.minmax,
    lat.minmax, lon.minmax, reanalysis2 = FALSE,
    return.units = TRUE, increments=NULL, pb=NULL)
NCEP.gather.pressure(variable, months.minmax, years.minmax,
    lat.minmax, lon.minmax, pressure, reanalysis2 = FALSE,
    return.units = TRUE, increments=NULL, pb=NULL)
NCEP.gather.surface(variable, months.minmax, years.minmax,
    lat.minmax, lon.minmax, reanalysis2 = FALSE,
    return.units = TRUE, increments=NULL, pb=NULL)
```

Arguments

variable	Character. The name of the weather variable to be obtained. See 'Details' for possible variable names.
level	A numeric pressure level or one of either 'gaussian' or 'surface'. See 'Details'.
months.minmax	Numeric. Specifies the range of months to be obtained from each year.
years.minmax	Numeric. Specifies the range of years to be obtained.
lat.southnorth	Numeric. Specifies the range of latitudes to be obtained in the order c(southernmost, northernmost).
lat.minmax	Same as lat.southnorth.
lon.westeast	Numeric. Specifies the range of longitudes to be obtained in the order c(westernmost, easternmost).
lon.minmax	Same as lat.westeast.
reanalysis2	Logical. Should the data be obtained from the Reanalysis I dataset (default) or from Reanalysis II?
return.units	Logical. Should the units of the variable being obtained be printed after the query finishes?
pressure	Numeric. A pressure level in millibars that is assigned automatically from the value of level when needed.
status.bar	Logical. Should a status bar be shown indicating the percentage of completion?
increments	Numeric. This value, which is assigned automatically when using NCEP.gather, indicates the number of queries necessary to retrieve all data.
pb	An object of class "tkProgressBar", which is assigned automatically when using NCEP.gather.

Details

NCEP.gather is a wrapper function that applies one of NCEP.gather.gaussian, NCEP.gather.pressure, or NCEP.gather.surface depending on the value of level.

level must specify one of either 'gaussian' or 'surface' or give a numerical pressure level in millibars. Numeric pressure levels must be one of 1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, 100, 70, 50, 30, 20, 10 See 'Variable Naming Conventions' below to determine if your variable of interest is stored relative to the surface, a pressure level, or a T62 Gaussian grid. Note that variables on a T62 Gaussian grid are evenly spaced in longitude but unevenly spaced in latitude while variables from either the surface or a particular pressure level are evenly spaced in both latitude and longitude (2.5 deg. x 2.5 deg.)

Months and years in months.minmax and years.minmax must be numeric and given in the order c(minimum,maximum).

Latitude and longitude ranges should be given in decimal degrees. If the latitudes or longitudes given do not match a grid point in the NCEP dataset, the function moves to the next grid point such that the specified range is always included. Latitude should always be given in the order c(southernmost, northernmost) and longitude should always be given in the order c(westernmost, easternmost).

Latitudes below the equator are negative. Longitudes west of the Prime Meridian can be specified using either positive (i.e. 350) or negative (i.e. -10) notation.

Some variables are not in both the Reanalysis I and II datasets. If a variable is chosen that is not in the specified dataset, the other dataset will be used... with a warning.

Very large queries may cause errors due to memory limitations in R. See `Memory-limits` for more information or `memory.limit` to increase available memory. Alternatively, consider querying subsets of your total desired range. For instance, subset a very large spatial domain into several smaller regions, perform analyses on each geographical subset independently, and then combine the results. Functions in the `raster-package` may be useful for managing subsets in the temporal domain.

Note that the status bar may be hidden behind an active R window.

`variable` must be specified using one of the names found in the section 'Variable Naming Conventions' below...

Value

This function returns a three dimensional array of weather data. The three dimensions are latitude, longitude, and datetime reflected in the dimnames of the array. Datetimes are always expressed in UTC with the format `"%Y_%m_%d_%H"`.

Optionally, the units of the variable being queried are printed upon completion.

Variable Naming Conventions

VARIABLES IN REFERENCE TO A PARTICULAR PRESSURE LEVEL

'air'	Air Temperature	deg K
'hgt'	Geopotential Height	m
'rhum'	Relative Humidity	%
'shum'	Specific Humidity	kg/kg
'omega'	Omega [Vertical Velocity]	Pascal/s
'uwnd'	U-Wind Component [East/West]	m/s
'vwnd'	V-Wind Component [North/South]	m/s

VARIABLES IN REFERENCE TO THE SURFACE

'air.sig995'	Air Temperature	(Near Surface)	deg K
'lftx.sfc'	Surface Lifted Index	(At Surface)	deg K
'lftx4.sfc'	Best (4-layer) Lifted Index	(At Surface)	deg K
'omega.sig995'	Omega [Vertical Velocity]	(Near Surface)	Pascal/s
'pottmp.sig995'	Potential Temperature	(Near Surface)	deg K
'pr_wtr.eatm'	Precipitable Water	(Entire Atmosphere)	kg/m^2
'pres.sfc'	Pressure	(At Surface)	Pascals
'rhum.sig995'	Relative Humidity	(Near Surface)	%
'slp'	Sea Level Pressure	(Sea Level)	Pascals
'mslp'	Mean Sea Level Pressure	(Sea Level)	Pascals
'uwnd.sig995'	U-Wind Component [East/West]	(Near Surface)	m/s
'vwnd.sig995'	V-Wind Component [North/South]	(Near Surface)	m/s

VARIABLES IN REFERENCE TO A T62 GAUSSIAN GRID

—— These variables are forecasts valid 6 hours after the reference time ——

'air.2m'	Air Temperature	(At 2 meters)	deg K
'icec.sfc'	Ice Concentration	(At Surface)	fraction
'pevpr.sfc'	Potential Evaporation Rate	(At Surface)	W/m^2
'pres.sfc'	Pressure	(At Surface)	Pascals
'runof.sfc'	Water Runoff	(At Surface)	kg/m^2
'sfcr.sfc'	Surface Roughness	(At Surface)	m
'shum.2m'	Specific Humidity	(At 2 meters)	kg/kg
'soilw.0-10cm'	Soil Moisture	(From 0-10 cm)	fraction
'soilw.10-200cm'	Soil Moisture	(From 10-200 cm)	fraction
'skt.sfc'	Skin Temperature	(At Surface)	deg K
'tmp.0-10cm'	Temperature of 0-10 cm layer	(From 0-10 cm)	deg K
'tmp.10-200cm'	Temperature of 10-200 cm layer	(From 10-200 cm)	deg K
'tmp.300cm'	Temperature at 300 cm	(From 300 cm)	deg K
'uwnd.10m'	U-wind	(At 10 meters)	m/s
'vwnd.10m'	V-wind	(At 10 meters)	m/s
'weasd.sfc'	Water equivalent of snow depth	(At Surface)	kg/m^2

—— These variables are 6 hour hindcasts from the reference time ——

'tmax.2m'	Maximum temperature	(At 2 meters)	deg K
'tmin.2m'	Minimum temperature	(At 2 meters)	deg K

—— These variables are 6 hour averages starting at the reference time ——

'cfnlf.sfc'	Cloud forcing net longwave flux	(At Surface)	W/m^2
'cfnsf.sfc'	Cloud forcing net solar flux	(At Surface)	W/m^2
'cprat.sfc'	Convective precipitation rate	(At Surface)	Kg/m^2/s
'csdlf.sfc'	Clear sky downward longwave flux	(At Surface)	W/m^2
'csdsf.sfc'	Clear sky downward solar flux	(At Surface)	W/m^2
'dlwrf.sfc'	Downward longwave radiation flux	(At Surface)	W/m^2
'dswrf.sfc'	Downward solar radiation flux	(At Surface)	W/m^2
'dswrf.ntat'	Downward solar radiation flux	(Nominal Top of Atmosphere)	W/m^2
'gflux.sfc'	Ground heat flux	(At Surface)	W/m^2
'lhtfl.sfc'	Latent heat net flux	(At Surface)	W/m^2
'nbdsf.sfc'	Near IR beam downward solar flux	(At Surface)	W/m^2
'nddsf.sfc'	Near IR diffuse downward solar flux	(At Surface)	W/m^2
'nlwrs.sfc'	Net longwave radiation	(At Surface)	W/m^2
'nswrs.sfc'	Net shortwave radiation	(At Surface)	W/m^2
'prate.sfc'	Precipitation rate	(At Surface)	Kg/m^2/s
'shtfl.sfc'	Sensible heat net flux	(At Surface)	W/m^2
'uflx.sfc'	Momentum flux (zonal)	(At Surface)	N/m^2
'ugwd.sfc'	Zonal gravity wave stress	(At Surface)	N/m^2
'ulwrf.sfc'	Upward longwave radiation flux	(At Surface)	W/m^2
'ulwrf.ntat'	Upward longwave radiation flux	(Nominal Top of Atmosphere)	W/m^2

'uswrf.sfc'	Upward solar radiation flux	(At Surface)	W/m^2
'uswrf.ntat'	Upward solar radiation flux	(Nominal Top of Atmosphere)	W/m^2
'vbdsf.sfc'	Visable beam downward solar flux	(At Surface)	W/m^2
'vddsf.sfc'	Visable diffuse downward solar flux	(At Surface)	W/m^2
'vflx.sfc'	Momentum flux (meridional)	(At Surface)	N/m^2
'vgwd.sfc'	Meridional gravity wave stress	(At Surface)	N/m^2

—— These variables are 6 hour averages starting at the reference time ——

'csulf.ntat'	Clear Sky Upward Longwave Flux	(Nominal Top of Atmosphere)	W/m^2
'csusf.ntat'	Clear Sky Upward Solar Flux	(Nominal Top of Atmosphere)	W/m^2
'dswrf.ntat'	Downward Solar Radiation Flux	(Nominal Top of Atmosphere)	W/m^2
'pres.hcb'	Pressure	(High Cloud Bottom)	Pascals
'pres.hct'	Pressure	(High Cloud Top)	Pascals
'pres.lcb'	Pressure	(Low Cloud Bottom)	Pascals
'pres.lct'	Pressure	(Low Cloud Top)	Pascals
'pres.mcb'	Pressure	(Middle Cloud Bottom)	Pascals
'pres.mct'	Pressure	(Middle Cloud Top)	Pascals
'tcdc.eatm'	Total Cloud Cover	(Entire Atmosphere)	%
'ulwrf.ntat'	Upward Longwave Radiation Flux	(Nominal Top of Atmosphere)	W/m^2
'uswrf.ntat'	Upward Solar Radiation Flux	(Nominal Top of Atmosphere)	W/m^2

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M.U., van Loon, E.E., Shamoun-Baranes, J.,and Bouten, W. (2011). RNCEP: global weather and climate data at your fingertips. Methods in Ecology and Evolution, DOI:10.1111/j.2041-210X.2011.00138.x.

To cite the NCEP/NCAR Reanalysis dataset use:

Kalnay et al. (1996), The NCEP/NCAR 40-year reanalysis project, Bull. Amer. Meteor. Soc., 77, 437-470

To cite the NCEP/DOE Reanalysis II dataset use:

Kanamitsu et al. (2002), NCEP-DOE AMIP-II Reanalysis (R-2). Bull. Amer. Meteor. Soc., 83, 1631-1643

Please acknowledge the use of NCEP data in any publications by including text such as, "NCEP Reanalysis data provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site at https://psl.noaa.gov/". They would also appreciate a copy of any publications using NCEP data.

Examples

```
## Not run:
library(RNCEP)
```

```
## Query the temperature from a particular pressure level ##
wx.extent1 <- NCEP.gather(variable='air', level=850,
    months.minmax=c(9,10), years.minmax=c(1996,1997),
    lat.southnorth=c(50,55), lon.westeast=c(5,10),
    reanalysis2 = FALSE, return.units = TRUE)

## Query the temperature at 2 meters altitude with reference to
## the surface
wx.extent2 <- NCEP.gather(variable='air.sig995', level='surface',
    months.minmax=c(2,3), years.minmax=c(2000,2001),
    lat.southnorth=c(50,55), lon.westeast=c(0,5),
    reanalysis2 = FALSE, return.units = TRUE)

## Query the temperature at 2 meters altitude with reference to
## a T62 Gaussian grid
wx.extent3 <- NCEP.gather(variable='air.2m', level='gaussian',
    months.minmax=c(4,5), years.minmax=c(2006,2007),
    lat.southnorth=c(32,35), lon.westeast=c(-35,-32),
    reanalysis2 = FALSE, return.units = TRUE)

## Note that the dimnames of the data array indicate the
## latitudes, longitudes, and datetimes of the data. ##
dimnames(wx.extent1)
## Therefore, the latitudes, longitudes, and datetimes
## can be called. ##
dimnames(wx.extent1)[[1]] ## latitudes
dimnames(wx.extent1)[[2]] ## longitudes
dimnames(wx.extent1)[[3]] ## datetimes

#####################################################################
#####################################################################
#####################################################################
## THERE ARE MANY OPTIONS FOR CREATING DIFFERENT R OBJECTS
## AND/OR FOR EXPORTING THESE WEATHER DATA.
## HERE ARE A FEW EXAMPLES

###########################################################
###########################################################
## The data array may be saved directly as an R object ##
save(wx.extent, file='wx_extent.Rdata')
## And then later recalled ##
load(file='wx_extent.Rdata')


#################################################################
#################################################################
## Another option is to create a raster object from the array ##
## For more info, see package raster
library(raster)

## Using the data from a query above ##
## First create a stacked raster object using the first
## layer (i.e. datetime) of the weather data array ##
```

```
## Notice the offset of 1.25 degrees (1/2 the spatial resolution)
## to describe the limits of the bounding box not the points
ras <- stack(raster(wx.extent1[,,1], crs="+proj=longlat +datum=WGS84",
    xmn=min(as.numeric(dimnames(wx.extent1)[[2]])) - 1.25,
    xmx=max(as.numeric(dimnames(wx.extent1)[[2]])) + 1.25,
    ymn=min(as.numeric(dimnames(wx.extent1)[[1]])) - 1.25,
    ymx=max(as.numeric(dimnames(wx.extent1)[[1]])) + 1.25))

## Then add each subsequent layer to the raster stack ##
for(i in 2:length(dimnames(wx.extent1)[[3]])){
    ras <- addLayer(ras, raster(wx.extent1[,,i],
        crs="+proj=longlat +datum=WGS84",
        xmn=min(as.numeric(dimnames(wx.extent1)[[2]])) - 1.25,
        xmx=max(as.numeric(dimnames(wx.extent1)[[2]])) + 1.25,
        ymn=min(as.numeric(dimnames(wx.extent1)[[1]])) - 1.25,
        ymx=max(as.numeric(dimnames(wx.extent1)[[1]])) + 1.25))
    }

#########################################################
## Optionally, export a layer from the raster stack to
## a format that can be imported by Esri's ArcGIS products ##

## First, select a layer from the raster stack ##
ras1 <- raster(ras, layer=1)

## Then write the data from that layer to a .bil file ##
writeRaster(ras, filename='ras_example.bil', format="EHdr")
## The file will be saved in your current working directory ##

## The resulting file can be imported into ArcGIS ##
## by using the "Raster to Other Format" tool in the
## "To Raster" section of the ArcToolbox.

#############################################################
## NOTE: Weather data obtained from a Gaussian grid must
## first be resampled onto a regular grid !!! ##
## Here we use the interp.loess() function
## from the tgp package

## Using data from a T62 Gaussian grid queried above
## Interpolate the data from the first layer (i.e. datetime)
## onto a regular grid ##
library(tgp)
wx.reg <- interp.loess(x=rep(as.numeric(dimnames(wx.extent3)[[2]]),
        each=length(dimnames(wx.extent3)[[1]])),
    y=rep(as.numeric(dimnames(wx.extent3)[[1]]),
         length(dimnames(wx.extent3)[[2]])),
    z=as.vector(wx.extent3[,,1]), span=0.6,
    gridlen=c(length(dimnames(wx.extent3)[[2]]),
        length(dimnames(wx.extent3)[[1]])))

## Create a stacked raster object from the first layer
## (i.e. datetime) after interpolation ##
```

```
## Again, notice the offset (1/2 the resolution) ##
## Also note that the matrix (i.e. wx.reg$z) must be flipped
## along the y axis and transposed ##
## This is required b/c of the interpolation performed above ##
ras <- stack(raster(t(wx.reg$z[,length(wx.reg$y):1]),
    crs="+proj=longlat +datum=WGS84",
    xmn=min(as.numeric(wx.reg$x)) - abs(diff(wx.reg$x)[1]/2),
    xmx=max(as.numeric(wx.reg$x)) + abs(diff(wx.reg$x)[1]/2),
    ymn=min(as.numeric(wx.reg$y)) - abs(diff(wx.reg$y)[1]/2),
    ymx=max(as.numeric(wx.reg$y)) + abs(diff(wx.reg$y)[1]/2)))

## Add each subsequent layer in the array to the raster stack ##
## after interpolating onto a regular grid ##

for(i in 2:length(dimnames(wx.extent3)[[3]])){

  ## Interpolate
  t.wx.reg <- interp.loess(x=rep(as.numeric(dimnames(wx.extent3)[[2]]),
      each=length(dimnames(wx.extent3)[[1]])),
      y=rep(as.numeric(dimnames(wx.extent3)[[1]]),
      length(dimnames(wx.extent3)[[2]])),
      z=as.vector(wx.extent3[,,i]), span=0.6,
      gridlen=c(length(dimnames(wx.extent3)[[2]]),
      length(dimnames(wx.extent3)[[1]])))

  ## Note the offset ##
  ## Note flipping the matrix along the y axis and transposing ##
  ## Add layer to stack
  ras <- addLayer(ras, raster(t(t.wx.reg$z[,length(t.wx.reg$y):1]),
    crs="+proj=longlat +datum=WGS84",
    xmn=min(as.numeric(t.wx.reg$x)) - abs(diff(t.wx.reg$x)[1]/2),
    xmx=max(as.numeric(t.wx.reg$x)) + abs(diff(t.wx.reg$x)[1]/2),
    ymn=min(as.numeric(t.wx.reg$y)) - abs(diff(t.wx.reg$y)[1]/2),
    ymx=max(as.numeric(t.wx.reg$y)) + abs(diff(t.wx.reg$y)[1]/2)))
  }

#####################################################################
#####################################################################
## Another option is to create a Spatial object
## using the sp package
## Again, data from a Gaussian grid may require special attention
## as the grid points are unevenly spaced
library(sp)

## Using the data from a query above
## Convert the array to a data.frame ##
wx.df <- NCEP.array2df(wx.extent2)

## Specify that the data.frame is a spatial object
library(sp)
coordinates(wx.df) <- ~longitude+latitude
gridded(wx.df) <- TRUE
proj4string(wx.df) <- CRS('+proj=longlat + datum=WGS84')
```

```
#########################################################
## A Spatial object of a single datetime (i.e. layer) can
## be written to .asc, a format that may then be
## imported into ArcGIS.

## First, convert the first layer of the array to a data.frame ##
wx.df <- NCEP.array2df(wx.extent2[,,1])

## Specify that the data.frame is a spatial object
library(sp)
coordinates(wx.df) <- ~longitude+latitude
gridded(wx.df) <- TRUE
proj4string(wx.df) <- CRS('+proj=longlat + datum=WGS84')

## Save the data in .asc format
write.asciigrid(wx.df, fname='wx.asc')
## Note: Data will be written to your working directory ##

## The resulting .asc file can be imported into ArcMap
## using ArcMap's "ASCII to Raster" tool in the "Conversion Tools"
## section of the ArcToolbox. ##

#################################################################
## There are still more options for writing these data to files ##
## See e.g. writeMat() in the R.matlab package for writing Matlab files
## Also see the RSAGA package for GIS functionality in R
## One could even write the data array back to NetCDF
## (see packages RNetCDF and ncdf)

## End(Not run)
```

NCEP.Groundspeed *Calculate flow-assistance according to equation 'Groundspeed'*

Description

This function calculates flow-assistance according to equation Groundspeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation Groundspeed.

Usage

```
NCEP.Groundspeed(u, v, direction, groundspeed,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.

v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating the preferred direction of movement in degrees from North.
groundspeed	The desired groundspeed (i.e. speed relative to the fixed Earth) of the animal in meters per second.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation Groundspeed. Equation Groundspeed stipulates that an animal maintains the specified groundspeed in the given direction by altering its airspeed and heading. Note that, as a result of these assumptions, flow-assistance degrades from optimum as the tailwind component increases beyond the specified groundspeed and forward movement remains constant at the speed specified by groundspeed.

Value

A data.frame containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation Groundspeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.Groundspeed to calculate flow-assistance ##
tst <- NCEP.Groundspeed(u=-2, v=-1, direction=225, groundspeed=12)
```

NCEP.interp *Interpolates Weather Data to a point in space and time*

Description

This function queries a weather variable via the Internet from the NCEP/NCAR Reanalysis or NCEP/DOE Reanalysis II datasets and subsequently calculates values at desired locations in space and time by interpolation.

Usage

```
 NCEP.interp(variable, level, lat, lon, dt, reanalysis2 = FALSE,
    interpolate.space = TRUE, interpolate.time = TRUE,
    keep.unpacking.info = FALSE, return.units = TRUE,
    interp = 'linear', p = 1, status.bar=TRUE)

## NCEP.interp is a wrapper function that calls one of the
## following functions based on the value of level.
## Users should avoid using these functions directly.

NCEP.interp.gaussian(variable, lat, lon, dt, reanalysis2 = FALSE,
    interpolate.space = TRUE, interpolate.time = TRUE,
    keep.unpacking.info = FALSE, return.units = TRUE,
    interp = 'linear', p = 1, status.bar=TRUE)
NCEP.interp.pressure(variable, lat, lon, dt, pressure,
    reanalysis2 = FALSE, interpolate.space = TRUE,
    interpolate.time = TRUE, keep.unpacking.info = FALSE,
    return.units = TRUE, interp = 'linear', p = 1, status.bar=TRUE)
NCEP.interp.surface(variable, lat, lon, dt, reanalysis2 = FALSE,
    interpolate.space = TRUE, interpolate.time = TRUE,
    keep.unpacking.info = FALSE, return.units = TRUE,
    interp = 'linear', p = 1, status.bar=TRUE)
```

Arguments

variable	Character. The name of the weather variable to be obtained. See 'Variable Naming Conventions' below for possible variable names.
level	A numeric pressure level or one of either 'gaussian' or 'surface'. See 'Details'.
lat	Numeric. The latitude to which the weather variable should be interpolated.
lon	Numeric. The longitude to which the weather variable should be interpolated.
dt	Character. The datetime (specified in UTC) to which the weather variable should be interpolated. Must use the format "%Y-%m-%d %H:%M:%S".
reanalysis2	Logical. Should the data be obtained from the Reanalysis II dataset or from Reanalysis I(default)?
interpolate.space	Logical. Should interpolation be done in space?

`interpolate.time`
: Logical. Should interpolation be done in time?

`keep.unpacking.info`
: Logical. Should the information needed to unpack the data be used for all queries in the function call?

`return.units`
: Logical. Should the units of the variable being obtained be printed after the query finishes?

`interp`
: Method of interpolation. One of 'linear' (default) or 'IDW'. See 'Details'.

`p`
: A positive real number. The power parameter controlling interpolation. Only matters when `interp` is 'IDW'. See 'Details'.

`pressure`
: Numeric. A pressure level in millibars that is assigned automatically from the value of `level` when needed.

`status.bar`
: Logical. Should a status bar be shown indicating the percentage of completion?

Details

`NCEP.interp` is a wrapper function that applies one of `NCEP.interp.gaussian`, `NCEP.interp.pressure`, or `NCEP.interp.surface` depending on the value of `level`.

`level` must specify one of either 'gaussian' or 'surface' or give a numerical pressure level in millibars. Numeric pressure levels must be one of 1000, 925, 850, 700, 600, 500, 400, 300, 250, 200, 150, 100, 70, 50, 30, 20, 10. See 'Variable Naming Conventions' below to determine if your variable of interest is stored relative to the surface, a pressure level, or a T62 Gaussian grid. Note that variables on a T62 Gaussian grid are evenly spaced in longitude but unevenly spaced in latitude while variables from either the surface or a particular pressure level are evenly spaced in both latitude and longitude (2.5 deg. x 2.5 deg.).

All arguments except `keep.unpacking.info`, `return.units`, and `status.bar` can be vectors. The remaining arguments are recycled to the length of the longest arguement.

`lat` and `lon` should be given in decimal degrees. Latitudes south of the equator should be negative. Longitudes west of the Prime Meridian can be specified using either positive (i.e. 350) or negative (i.e. -10) notation.

All interpolation is performed assuming a spherical (rather than a planar) grid.

When `interp` is 'IDW', 2-D spatial interpolation is done using inverse distance weighting followed by a 1-D linear interpolation in time. When `interp` is 'linear', the function performs a trilinear interpolation in latitude, longitude, and time. If `interpolate.space` or `interpolate.time` is `FALSE`, the function performs 'nearest neighbor' interpolation and returns data from the grid point closest in space or time, respectively. The numerical value of `p` controls the degree of smoothing in the interpolation only when `interp` is 'IDW'. Greater values of `p` assign greater influence to values closest to the interpolated point. For $0 < p < 1$ peaks over the interpolated point remain smooth. As `p` increases beyond 1, the peaks become sharper.

Variables in these datasets on a T62 Gaussian grid describe conditions over an interval of time rather than at a particular point in time. (see 'Variable Naming Conventions' below) As such, it is not appropriate to perform temporal interpolation on these varaibles. Therefore, `NCEP.interp` automatically sets `interpolate.time` to `FALSE` when querying one of these variables, and returns the data corresponding to the interval within which the specified datetime falls. Spatial interpolation is still performed as long as `interpolate.space` is `TRUE`.

Unpacking information is unique to each variable and dataset. Therefore, `keep.unpacking.info` can be made `TRUE` as long as only one variable from one dataset (i.e. Reanalysis I or II) is queried in a single function call, even for multiple times and locations. `keep.unpacking.info` will be made `FALSE`, if necessary, with a warning.

The function will run faster when `keep.unpacking.info` is `TRUE`.

The robust alternative to `NCEP.interp` is applied automatically. These robust functions are applied when interpolation requires data from two different years or from both sides of the Prime Meridian.

Some variables are not in both the Reanalysis I and II datasets. If a variable is chosen that is not in the specified dataset, the other dataset will be used... with a warning.

The function also returns, as an attribute, standard deviation calculated on all data used in the interpolation. This provides an indication of the precision of an interpolated result described in the same units as the interpolated variable. Smaller values indicate that there is less variability among the points used in interpolation. Standard deviation is only calcuated on the points used in the interpolation. Therefore, if `interpolate.time` and `interpolate.space` are both `TRUE`, standard deviation is calculated on eight points.
If `interpolate.time` is `FALSE` and `interpolate.space` is `TRUE`, standard deviation is calculated on four points. If `interpolate.time` is `TRUE` and `interpolate.space` is `FALSE`, standard deviation is calculated on only two points. If `interpolate.time` and `interpolate.space` are both `FALSE`, standard deviation is not calculated and `NA` is returned. This measure of precision is the same irrespective of `interp`.

Note that the status bar may be hidden behind an active R window.

`variable` must be specified using one of the names found in the section 'Variable Naming Conventions' below...

Value

A vector of interpolated results with the associated standard deviation of the points used to perform the interpolation as an attribute.

Optionally, the units of the variable being queried are printed when the function completes.

Variable Naming Conventions

VARIABLES IN REFERENCE TO A PARTICULAR PRESSURE LEVEL

'air'	Air Temperature	deg K
'hgt'	Geopotential Height	m
'rhum'	Relative Humidity	%
'shum'	Specific Humidity	kg/kg
'omega'	Omega [Vertical Velocity]	Pascal/s
'uwnd'	U-Wind Component [East/West]	m/s
'vwnd'	V-Wind Component [North/South]	m/s

VARIABLES IN REFERENCE TO THE SURFACE

'air.sig995'	Air Temperature	(Near Surface)	deg K
'lftx.sfc'	Surface Lifted Index	(At Surface)	deg K

'lftx4.sfc'	Best (4-layer) Lifted Index	(At Surface)	deg K
'omega.sig995'	Omega [Vertical Velocity]	(Near Surface)	Pascal/s
'pottmp.sig995'	Potential Temperature	(Near Surface)	deg K
'pr_wtr.eatm'	Precipitable Water	(Entire Atmosphere)	kg/m^2
'pres.sfc'	Pressure	(At Surface)	Pascals
'rhum.sig995'	Relative Humidity	(Near Surface)	%
'slp'	Sea Level Pressure	(Sea Level)	Pascals
'mslp'	Mean Sea Level Pressure	(Sea Level)	Pascals
'uwnd.sig995'	U-Wind Component [East/West]	(Near Surface)	m/s
'vwnd.sig995'	V-Wind Component [North/South]	(Near Surface)	m/s

VARIABLES IN REFERENCE TO A T62 GAUSSIAN GRID

—— These variables are forecasts valid 6 hours after the reference time ——

'air.2m'	Air Temperature	(At 2 meters)	deg K
'icec.sfc'	Ice Concentration	(At Surface)	fraction
'pevpr.sfc'	Potential Evaporation Rate	(At Surface)	W/m^2
'pres.sfc'	Pressure	(At Surface)	Pascals
'runof.sfc'	Water Runoff	(At Surface)	kg/m^2
'sfcr.sfc'	Surface Roughness	(At Surface)	m
'shum.2m'	Specific Humidity	(At 2 meters)	kg/kg
'soilw.0-10cm'	Soil Moisture	(From 0-10 cm)	fraction
'soilw.10-200cm'	Soil Moisture	(From 10-200 cm)	fraction
'skt.sfc'	Skin Temperature	(At Surface)	deg K
'tmp.0-10cm'	Temperature of 0-10 cm layer	(From 0-10 cm)	deg K
'tmp.10-200cm'	Temperature of 10-200 cm layer	(From 10-200 cm)	deg K
'tmp.300cm'	Temperature at 300 cm	(From 300 cm)	deg K
'uwnd.10m'	U-wind	(At 10 meters)	m/s
'vwnd.10m'	V-wind	(At 10 meters)	m/s
'weasd.sfc'	Water equivalent of snow depth	(At Surface)	kg/m^2

—— These variables are 6 hour hindcasts from the reference time ——

'tmax.2m'	Maximum temperature	(At 2 meters)	deg K
'tmin.2m'	Minimum temperature	(At 2 meters)	deg K

—— These variables are 6 hour averages starting at the reference time ——

'cfnlf.sfc'	Cloud forcing net longwave flux	(At Surface)	W/m^2
'cfnsf.sfc'	Cloud forcing net solar flux	(At Surface)	W/m^2
'cprat.sfc'	Convective precipitation rate	(At Surface)	Kg/m^2/s
'csdlf.sfc'	Clear sky downward longwave flux	(At Surface)	W/m^2
'csdsf.sfc'	Clear sky downward solar flux	(At Surface)	W/m^2
'dlwrf.sfc'	Downward longwave radiation flux	(At Surface)	W/m^2
'dswrf.sfc'	Downward solar radiation flux	(At Surface)	W/m^2
'dswrf.ntat'	Downward solar radiation flux	(Nominal Top of Atmosphere)	W/m^2
'gflux.sfc'	Ground heat flux	(At Surface)	W/m^2

'lhtfl.sfc'	Latent heat net flux	(At Surface)	W/m^2
'nbdsf.sfc'	Near IR beam downward solar flux	(At Surface)	W/m^2
'nddsf.sfc'	Near IR diffuse downward solar flux	(At Surface)	W/m^2
'nlwrs.sfc'	Net longwave radiation	(At Surface)	W/m^2
'nswrs.sfc'	Net shortwave radiation	(At Surface)	W/m^2
'prate.sfc'	Precipitation rate	(At Surface)	Kg/m^2/s
'shtfl.sfc'	Sensible heat net flux	(At Surface)	W/m^2
'uflx.sfc'	Momentum flux (zonal)	(At Surface)	N/m^2
'ugwd.sfc'	Zonal gravity wave stress	(At Surface)	N/m^2
'ulwrf.sfc'	Upward longwave radiation flux	(At Surface)	W/m^2
'ulwrf.ntat'	Upward longwave radiation flux	(Nominal Top of Atmosphere)	W/m^2
'uswrf.sfc'	Upward solar radiation flux	(At Surface)	W/m^2
'uswrf.ntat'	Upward solar radiation flux	(Nominal Top of Atmosphere)	W/m^2
'vbdsf.sfc'	Visable beam downward solar flux	(At Surface)	W/m^2
'vddsf.sfc'	Visable diffuse downward solar flux	(At Surface)	W/m^2
'vflx.sfc'	Momentum flux (meridional)	(At Surface)	N/m^2
'vgwd.sfc'	Meridional gravity wave stress	(At Surface)	N/m^2

—— These variables are 6 hour averages starting at the reference time ——

'csulf.ntat'	Clear Sky Upward Longwave Flux	(Nominal Top of Atmosphere)	W/m^2
'csusf.ntat'	Clear Sky Upward Solar Flux	(Nominal Top of Atmosphere)	W/m^2
'dswrf.ntat'	Downward Solar Radiation Flux	(Nominal Top of Atmosphere)	W/m^2
'pres.hcb'	Pressure	(High Cloud Bottom)	Pascals
'pres.hct'	Pressure	(High Cloud Top)	Pascals
'pres.lcb'	Pressure	(Low Cloud Bottom)	Pascals
'pres.lct'	Pressure	(Low Cloud Top)	Pascals
'pres.mcb'	Pressure	(Middle Cloud Bottom)	Pascals
'pres.mct'	Pressure	(Middle Cloud Top)	Pascals
'tcdc.eatm'	Total Cloud Cover	(Entire Atmosphere)	%
'ulwrf.ntat'	Upward Longwave Radiation Flux	(Nominal Top of Atmosphere)	W/m^2
'uswrf.ntat'	Upward Solar Radiation Flux	(Nominal Top of Atmosphere)	W/m^2

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M.U., van Loon, E.E., Shamoun-Baranes, J.,and Bouten, W. (2011). RNCEP: global weather and climate data at your fingertips. Methods in Ecology and Evolution, DOI:10.1111/j.2041-210X.2011.00138.x.

To cite the NCEP/NCAR Reanalysis dataset use:

Kalnay et al. (1996), The NCEP/NCAR 40-year reanalysis project, Bull. Amer. Meteor. Soc., 77, 437-470

To cite the NCEP/DOE Reanalysis II dataset use:

Kanamitsu et al. (2002), NCEP-DOE AMIP-II Reanalysis (R-2). Bull. Amer. Meteor. Soc., 83, 1631-1643

Please acknowledge the use of NCEP data in any publications by including text such as, "NCEP Reanalysis data provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site at https://psl.noaa.gov/". They would also appreciate a copy of any publications using NCEP data.

Examples

```
## Not run:
library(RNCEP)
#################################################################
#################################################################
## The function can be applied to interpolate a single variable
## to a single point in space and time ##
## Interpolate temperature from the 850 mb pressure level ##
wx.interp <- NCEP.interp(variable='air', level=850, lat=55.1,
    lon=11.3, dt='2006-10-12 17:23:12',
    interp='linear')

## Interpolate precipitable water (for the entire atmosphere, but
## described in reference to the surface)
wx.interp <- NCEP.interp(variable='pr_wtr.eatm', level='surface',
    lat=55.1, lon=11.3, dt='2006-10-12 17:23:12',
    interp='linear')

## Interpolate specific humidity (at the surface, but in
## reference to a T62 Gaussian grid) using the IDW interpolation
wx.interp <- NCEP.interp(variable='shum.2m', level='gaussian',
    lat=55.1, lon=11.3, dt='2006-10-12 17:23:12',
    interp='IDW', p=1)

#####################################################################
#####################################################################
## The function can also be applied to interpolate several variables,
## locations, datetimes, and/or methods of interpolation in a single
## function call ##
## Interpolate temperature from the 850 and 700 mb pressure levels ##
## for the same time and location ##
wx.interp <- NCEP.interp(variable='air', level=c(850,700), lat=55.1,
    lon=11.3, dt='2006-10-12 17:23:12',
    interp='linear')

## Interpolate temperature and relative humidity from the 1000 mb
## pressure level ##
wx.interp <- NCEP.interp(variable=c('air','rhum'), level=1000,
    lat=55.1, lon=11.3, dt='2006-10-12 17:23:12', interp='linear')

## Interpolate temperature and relative humidity
## from the 1000 and 700 mb pressure levels, respectively
## for the same datetime ##
wx.interp <- NCEP.interp(variable=c('air','rhum'),
```

```
    level=c(1000,700), lat=55.1, lon=11.3,
    dt='2006-10-12 17:23:12', interp='linear')

## Interpolate temperature and relative humidity
## from the 1000 and 700 mb pressure levels, respectively
## for different datetimes ##
wx.interp <- NCEP.interp(variable=c('air','rhum'), level=c(1000,700), lat=55.1,
    lon=11.3, dt=c('2006-10-12 17:23:12', '2006-10-12 18:05:31'),
    interp='linear')

## Interpolate geopotential height using 'linear', 'IDW', and
## 'nearest neighbor' interpolation ##
wx.interp <- NCEP.interp(variable='hgt', level=700, lat=55.1,
    lon=11.3, dt='2006-10-12 17:23:12',
    interp=c('linear','IDW','IDW'),
    interpolate.space=c(TRUE,TRUE,FALSE))

#################################################################
#################################################################
## Alternatively the function can be applied to interpolate a
## weather variable to multiple datetime and point locations
## in a single function call ##

## In this example, we use datetime and locational data obtained
## from a GPS device attached to a lesser black-backed gull.
## We interpolate wind information to to each point in the dataset
data(gull)

## Take a subset of the data based on the datetime of
## the measurement ##
ss <- subset(gull, format(gull$datetime, "%Y-%m-%d %H:%M:%S") >=
    "2008-09-19 16:00:00" & format(gull$datetime,
    "%Y-%m-%d %H:%M:%S") <= "2008-09-19 19:30:00")


## Now collect wind information for each point in the subset ##
uwind <- NCEP.interp(variable='uwnd', level=925,
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=TRUE, keep.unpacking.info=TRUE)
vwind <- NCEP.interp(variable='vwnd', level=925,
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=TRUE, keep.unpacking.info=TRUE)


## Now calculate the tailwind component from the U and V
## wind components assuming that the bird's preferred
## direction is 225 degrees
tailwind <- (sqrt(uwind^2 + vwind^2)*cos(((atan2(uwind,vwind)*
    (180/pi))-225)*(pi/180)))

## Now visualize the subset of the GPS track using color
## to indicate the tailwind speed ##
NCEP.vis.points(wx=tailwind, lats=ss$latitude, lons=ss$longitude,
```

```
    cols=rev(heat.colors(64)),
    title.args=list(main='Lesser black-backed gull'),
    image.plot.args=list(legend.args=list(text='Tailwind m/s',
    adj=0, padj=-2, cex=1.15)),
    map.args=list(xlim=c(-7,4), ylim=c(40,50)))

## End(Not run)
```

NCEP.loxodrome *Calculate the loxodrome angle between two points on Earth.*

Description

This function calculates the loxodrome angle (i.e rhumb line or constant compass heading) between two points on a sphere.

Usage

```
NCEP.loxodrome(lat1,lat2,lon1,lon2)
```

Arguments

lat1	A numeric value giving the starting latitude in decimal degrees.
lat2	A numeric value giving the ending latitude in decimal degrees.
lon1	A numeric value giving the starting longitude in decimal degrees.
lon2	A numeric value giving the ending longitude in decimal degrees.

Details

This function calculates the loxodrome angle (i.e. rhumb line or constant compass heading) between two points on a sphere. Output is given in degrees from north.

Value

A numeric value indicating the loxodrome angle between the two input points in degrees from north.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

https://en.wikipedia.org/wiki/Rhumb_line

Examples

```
library(RNCEP)
## Using NCEP.loxodrome ##
NCEP.loxodrome(lat1=45,lat2=40,lon1=4,lon2=5)
```

NCEP.M.Groundspeed *Calculate flow-assistance according to equation 'M.Groundspeed'*

Description

This function calculates flow-assistance according to equation M.Groundspeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation M.Groundspeed.

Usage

```
NCEP.M.Groundspeed(u, v, direction, p.airspeed,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating the preferred direction of movement in degrees from North.
p.airspeed	A numeric value indicating the animal's **preferred** speed relative to the flow (i.e. airspeed) in meters per second. See Details.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation M.Groundspeed. Equation M.Groundspeed stipulates that an animal maintains a speed relative to the fixed Earth (i.e. groundspeed) in the specified `direction` equal to its preferred airspeed (i.e. `p.airspeed`) plus the component of the flow parallel to the preferred `direction` by altering its actual airspeed and heading. Note that, the actual airspeed that a animal must exhibit, following these assumptions, can be very different from its preferred airspeed.

Value

A `data.frame` containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming `u`, `v`, and `airspeed` were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation M.Groundspeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.M.Groundspeed to calculate flow-assistance ##
tst <- NCEP.M.Groundspeed(u=-2, v=-1, direction=225, p.airspeed=12)
```

NCEP.NegFlowSpeed *Calculate flow-assistance according to equation 'NegFlowSpeed'*

Description

This function calculates flow-assistance according to equation NegFlowSpeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation NegFlowSpeed.

Usage

```
NCEP.NegFlowSpeed(u, v, direction, airspeed,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating the preferred direction of movement in degrees from North. For this function, direction is required but does not change the calculation of flow-assistance or the direction of movement. The argument is included for use with NCEP.flight.

airspeed	The animal's speed relative to the flow in meters per second. This value does not affect the calculation of flow-assistance, but does affect the forward and sideways movement of the animal.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation NegFlowSpeed. Equation NegFlowSpeed considers one minus the speed of the flow (e.g. wind speed) as flow-assistance, therefore it can never produce positive flow-assistance values. It assumes that the animal applies its own airspeed in the opposite direction of the flow (e.g. wind direction).

Value

A data.frame containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation NegFlowSpeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.NegFlowSpeed to calculate flow-assistance ##
tst <- NCEP.NegFlowSpeed(u=-2, v=-1, direction=225, airspeed=12)
```

`NCEP.PartialSpeed` *Calculate flow-assistance according to equation 'PartialSpeed'*

Description

This function calculates flow-assistance according to equation PartialSpeed and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation PartialSpeed.

Usage

```
NCEP.PartialSpeed(u, v, direction, airspeed, f=0.5,...)
```

Arguments

`u`	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
`v`	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
`direction`	A numeric value indicating the preferred direction of movement in degrees from North.
`airspeed`	The animal's speed relative to the flow in meters per second.
`f`	A numeric value between zero and one describing the proportion of the lateral component of the flow for which the animal will compensate.
`...`	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation PartialSpeed. Equation PartialSpeed stipulates that the animal compensates for a proportion (specified by `f`) of the flow lateral to the preferred `direction` by altering its heading and speed relative to the fixed Earth (i.e. groundspeed). If, with its given `airspeed`, the animal is incapable of compensating for the specified proportion of the lateral component of the flow, the equation produces no real solution.

Setting `f` to zero specifies no compensation (cf. `link{NCEP.Tailwind}`), while setting `f` to one specifies complete compensation (cf. `link{NCEP.Airspeed}`.

Value

A `data.frame` containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement

('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation PartialSpeed see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.PartialSpeed to calculate flow-assistance ##
tst <- NCEP.PartialSpeed(u=-2, v=-1, direction=225, airspeed=12)
```

NCEP.restrict *Temporally Filters Weather Data*

Description

This function removes unwanted datetime intervals (i.e. layers) of weather data from the NCEP/NCAR Reanalysis or NCEP/DOE Reanalysis II data array as returned by NCEP.gather. The spatial structure of the data is retained.

Usage

```
NCEP.restrict(wx.data, years2remove = NULL, months2remove = NULL,
    days2remove = NULL, hours2remove = NULL, other2remove = NULL,
    set2na = TRUE)
```

Arguments

wx.data	A 3-D weather dataset as returned by NCEP.gather
years2remove	Numeric. Specifies which years should be removed from the dataset.
months2remove	Numeric. Specifies which months should be removed from the dataset.
days2remove	Numeric. Specifies which days of the month should be removed from the dataset.
hours2remove	Numeric. Specifies which hours of the day should be removed from the dataset.

other2remove	Character. Specifies any specific combinations of year, month, day, and hour to remove from the dataset.
set2na	Logical. Should the data matching the year, month, day, or hour specified in the function call be set to NA (default) or should they be removed completely from the dataset.

Details

other2remove is for specific combinations of year, month, day, and hour and must be given in the format "%Y-%m-%d %H". If NCEP.aggregate has been applied, some datetime components will need to be replaced with "XX" or "XXXX" when specifying other2remove. Use dimnames to determine how to specify an aggregated datetime.

If set2na is anything other than TRUE or FALSE, the function replaces items to be removed with the value of set2na.

Value

This function returns a three dimensional array (or a 2-D matrix if all but a single timestep is removed) of weather data. The three dimensions are latitude, longitude, and datetime reflected in the dimnames of the output array.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
## First query the U component of the wind from the 850mb
## pressure level
uwnd <- NCEP.gather(variable='uwnd', level=850,
    months.minmax=c(5,7), years.minmax=c(2000,2001),
    lat.southnorth=c(50,55), lon.westeast=c(0,5))

## Then remove all observations except those made at midnight from
## the first half of either May or July
uwnd.r <- NCEP.restrict(wx.data=uwnd, hours2remove=c(6,12,18),
    days2remove=seq(17,31), months2remove=6, set2na=FALSE)

## Then remove the observation from 1 May 2000 at midnight ##
uwnd.r2 <- NCEP.restrict(wx.data=uwnd.r,
    other2remove="2000-05-01 00", set2na=FALSE)

## End(Not run)
```

NCEP.Tailwind *Calculate flow-assistance according to equation 'Tailwind'*

Description

This function calculates flow-assistance according to equation Tailwind and determines the speed of forward and sideways movement if an animal behaves according to the rules of equation Tailwind.

Usage

```
NCEP.Tailwind(u, v, direction, airspeed=NA,...)
```

Arguments

u	A numeric value indicating the U (i.e. zonal or east/west) flow component in meters per second, toward east being positive. Values must describe the direction into which the flow is moving.
v	A numeric value indicating the V (i.e. meridional or north/south) flow component in meters per second, toward north being positive. Values must describe the direction into which the flow is moving.
direction	A numeric value indicating the preferred direction of movement in degrees from North.
airspeed	The animal's speed relative to the flow in meters per second. This value does not affect the calculation of flow-assistance, but does affect the animal's forward and sideways movement.
...	Any extra arguments passed to the flow-assistance equation.

Details

This function calculates flow-assistance and forward and sideways movement according to equation Tailwind. Equation Tailwind considers flow-assistance to be the component of the flow moving parallel to the specified direction, with negative values indicating flows against the specified direction.

Value

A data.frame containing flow-assistance ('fa'), the animal's forward speed ('forward.move' which includes the animal's own airspeed), the animal's sideways speed ('side.move' which includes the animal's own airspeed), the component of the flow parallel to preferred direction of movement('tailwind'), the component of the flow perpendicular to the preferred direction of movement ('sidewind'), the animal's speed relative to the flow ('airspeed'), and the animal's speed relative to the fixed Earth ('groundspeed') each in meters per second, presuming u, v, and airspeed were given in meters per second.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

For more information on flow-assistance and equation Tailwind see:

Kemp, M.U., Shamoun-Baranes, J., van Loon, E. E., and Bouten, W. 2012. Quantifying flow-assistance and implications for movement research. – Journal of Theoretical Biology. In prep.

Examples

```
library(RNCEP)
## Using NCEP.Tailwind to calculate flow-assistance ##
tst <- NCEP.Tailwind(u=-2, v=-1, direction=225, airspeed=12)
```

NCEP.track2kml *Plot a track in Google Earth*

Description

This function creates a .kml file from a time series of point locations (e.g. as returned by NCEP.flight or measured with a GPS device) that can be viewed as a track in Google Earth.

Usage

```
NCEP.track2kml(latitude, longitude, datetime, altitude=NULL,
  col.variable=NULL, col.scheme=NULL, point.alpha=255,
  line.color='goldenrod', line.alpha=255, size.variable=NULL,
  point.names=NULL, data.variables=NULL, output.filename='track',
  descriptive.filename=NULL)
```

Arguments

latitude	A numeric vector of latitudes in decimal degrees
longitude	A numeric vector of longitudes in decimal degrees
datetime	A character vector of datetime increments in the format "%Y-%M-%D %H:%M:%S".
altitude	An optional vector of altitudes in meters.
col.variable	An optional numeric vector upon which the color of each point should be based
col.scheme	A character description of the color scheme to use in coloring the points. Several options, see Details.
point.alpha	A numeric vector of length one indicating the transparency of all points on a scale from 0 (transparent) to 255 (opaque)
line.color	An character expression (any of colors or hexadecimal notation), or numeric indicating the color of the line connecting the point locations.

`line.alpha` A numeric vector of length one indicating the transparency of the line connecting the point locations on a scale from 0 (transparent) to 255 (opaque)

`size.variable` An optional numeric vector upon which the size of the points should be based

`point.names` An optional character vector of containing names for each point

`data.variables` An optional data.frame containing any descriptor information for each point. See Details.

`output.filename`
A character expression giving the name of the resulting output file. This should not include the .kml extension.

`descriptive.filename`
The name of the object in the .kml file. Defaults to `output.filename`.

Details

There are several options for specifying `col.scheme`. A single color may be specified for all points (using e.g. `'red'`, `2`, or `"#FF0000"`), specific colors may be given for each point (e.g. `c('red','blue','green', etc.)` or `c('#FF0000','#0000FF','#00FF00', etc.)`), or the function can automatically assign colors, according to the values in `clo.variable`, if `col.scheme` is any color palette in `display.brewer.all` or one of the recognizezd R color palettes (i.e. `'rainbow'`, `'heat.colors'`, `'terrain.colors'`, `'topo.colors'`, `'cm.colors'`, or `'bpy.colors'`). An alpha value (0-255) may be supplied to `point.alpha` to adjust transparencies.

In the output .kml file, each point along the track contains a table of values. By default, this table contains the latitude, longitude, and datetime. Variables contained in the `data.frame` described in `data.variables` will also be included in this table.

The altitude associated with each point, i.e. those passed to the `altitude` argument, should be supplied in meters. For an example of how to assign general altitudes from pressure levels, see the Examples below.

Depending on `col.scheme`, this function may require **RColorBrewer** or **sp**.

Value

This function returns no data. It creates a .kml file in the current working directory.

Author(s)

Michael U. Kemp <<mukemp+RNCEP@gmail.com>>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

See Also

`NCEP.flight`

Examples

```
## Not run:
## Load the gull dataset ##
data(gull)

## Create a subset of the full dataset ##
g <- gull[1:100,]

## Create a .kml file from a portion of the GPS track ##
NCEP.track2kml(latitude=g$latitude, longitude=g$longitude,
    datetime=as.character(g$datetime), altitude=g$altitude,
    col.variable=g$altitude, col.scheme='heat.colors',
    point.alpha=255, line.color='goldenrod', line.alpha=255,
size.variable=NULL, point.names=NULL,
    data.variables=data.frame(g$altitude),
output.filename='track', descriptive.filename=NULL)

## End(Not run)
```

`NCEP.uv.revert` *Reverts speed and direction to U and V components.*

Description

This function calculates U (i.e. zonal or east/west) and V (i.e. meridional or north/south) components from a specified speed and direction.

Usage

```
NCEP.uv.revert(spd, dir, radians=FALSE)
```

Arguments

`spd`	A numeric value indicating speed.
`dir`	A numeric value indicating direction in degrees from north or radians if `radians` is TRUE.
`radians`	A logical indicating whether `dir` is given in degrees from north (FALSE) or radians (TRUE).

Details

This function calculates U (i.e. zonal or east/west) and V (i.e. meridional or north/south) components from a specified speed and direction. U and V components describe the direction into which movement occurs with east and north being positive, respectively. If directions are given in radians, `radians` must be set to `TRUE`.

Value

A `data.frame` containing U and V components given in the same units as `speed`.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
library(RNCEP)
## Using NCEP.uv.revert ##
NCEP.uv.revert(spd=12, dir=225, radians=FALSE)
```

NCEP.vis.area *Visualize Weather Data on a Map*

Description

This function creates a filled contour map from weather data. It visualizes data from a single layer (i.e. timestep) of a data array as returned by NCEP.gather or a single aggregated layer as returned by NCEP.aggregate

Usage

```
NCEP.vis.area(wx.data, layer=1, show.pts=TRUE, draw.contours=TRUE,
    cols=heat.colors(64), transparency=.5, axis.args=NULL, map.args=NULL,
    grid.args=NULL, title.args=NULL, interp.loess.args=NULL,
    image.plot.args=NULL, contour.args=NULL, points.args=NULL)
```

Arguments

wx.data	A 3-D array of weather data as returned by NCEP.gather or NCEP.aggregate
layer	Either a numerical indication of the layer (default is the first layer) or a character expression of the datetime of a particular layer.
show.pts	Logical. Should the points at which data were obtained be plotted?
draw.contours	Logical. Should the map include contour lines?
cols	A vector of colors such as that generated by rainbow, heat.colors, topo.colors, terrain.colors, or similar functions indicating the colors of the filled contours on the map.
transparency	A numeric value between 0 and 1 indicating the transparency of the filled contours on the map.

`axis.args` A list of arguments controlling the drawing of axes. See `axis` for acceptable arguments and the examples below for a demonstration.

`map.args` A list of arguments controlling the drawing of the map. See `map` for acceptable arguments and the examples below for a demonstration.

`grid.args` A list of arguments controlling the drawing of the lat/long grid lines. See `abline` for acceptable arguments and the examples below for a demonstration.

`title.args` A list of arguments controlling the how titles and axis lables are written. See `title` for acceptable arguments and the examples below for a demonstration.

`interp.loess.args`
A list of arguments controlling the interpolation between grid points. See `interp.loess` for acceptable arguments and the examples below for a demonstration.

`image.plot.args`
A list of arguments controlling the plotting of the filled contour surface, the color-bar legend, and the legend axis and labels. See `image.plot` for acceptable arguments and the examples below for a demonstration.

`contour.args` A list of arguments controlling the drawing of contour lines. See `contour` for acceptable arguments and the examples below for a demonstration.

`points.args` A list of arguments controlling the plotting of grid points. See `points` for acceptable arguments and the examples below for a demonstration.

Details

If the specification of `layer` is not numeric, it must specify a layer by a character expression of its datetime using the format `"%Y-%m-%d %H"`. Leading zeros in the month, day, and hour components must be given (e.g. `"2006-01-01 06"`)

If the data array has been aggregated in some way, the datetime dimension may no longer contain information on one or more temporal component: year, month, day, or hour. In this case, replace missing datetime componets with `"XX"` or `"XXXX"` (e.g. `"XXXX-01-XX XX"`) when specifying a layer by a character expression of its datetime. Use `dimnames` to see the datetime dimension labels.

If the geographical area to be plotted contains no landmass, the plot will exhibit a small unfilled border.

Most of the components of a plot produced by this function can be controlled by supplying a list of arguments to the embedded function that produces the particular component of the plot. For example, the text and size of the plot's title can be controlled by specifying a list of acceptable arguments to `title.args`. Similarly, the axes, map, and grid lines are controlled by specifying a list of acceptable arguements to `axis.args`, `map.args`, and `grid.args`, respectively. Through the argument `image.plot.args` the user can control the plotting of the filled contour surface, the color-bar legend, and the color-bar's title and axis labels. A list of arguments passed to `interp.loess.args` controls the interpolation of the filled-contour surface. See the examples below for a demonstration of how to apply these different arguments.

Value

A plot is produced. No data are returned.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
## Retrieve data for a specified spatial and temporal extent ##
wx.extent <- NCEP.gather(variable = "air", level=850,
    months.minmax = 11, years.minmax = 2000,
    lat.southnorth = c(50, 60), lon.westeast = c(0, 10),
    reanalysis2 = FALSE, return.units = TRUE)

## Visualize the first layer (i.e. first timestep) of the
## variable on a map
## Note how to specify the plot's title
## Note also the adjustment of the Kernal span argument for
##  interpolation using interp.loess.args
NCEP.vis.area(wx.data=wx.extent, layer=1, show.pts=TRUE, draw.contours=TRUE,
    cols=heat.colors(64), transparency=.6, title.args=list(main="Example"),
    interp.loess.args=list(span=.75))

## Now visualize a particular layer by specifying its datetime ##
NCEP.vis.area(wx.data=wx.extent, layer='2000-11-01 18', show.pts=TRUE,
    draw.contours=TRUE, cols=terrain.colors(64), transparency=.6,
    title.args=list(main="Example: select layer by datetime"),
    interp.loess.args=list(span=0.5))

## Now produce the same graph as above ##
## This time, label the color-bar legend ##
NCEP.vis.area(wx.data=wx.extent, layer='2000-11-01 18', show.pts=TRUE,
    draw.contours=TRUE, cols=terrain.colors(64), transparency=.6,
    title.args=list(main="Example: select layer by datetime"),
    interp.loess.args=list(span=0.5),
    image.plot.args=list(legend.args=list(text='Kelvin', cex=1.15, padj=-1, adj=-.25)))

## Now produce the same graph as above ##
## This time, explicitly specify the size and location
## of the color-bar legend using the smallplot argument
## in the list of image.plot.args ##
NCEP.vis.area(wx.data=wx.extent, layer='2000-11-01 18', show.pts=TRUE,
    draw.contours=TRUE, cols=terrain.colors(64), transparency=.6,
    title.args=list(main="Example: select layer by datetime"),
    interp.loess.args=list(span=0.5),
```

```
    image.plot.args=list(legend.args=list(text='Kelvin', cex=1.15, padj=-1, adj=-.25),
        smallplot=c(0.8475, 0.875, 0.20, 0.8)))

############################################################
## This function can also show a layer after aggregation ##
############################################################
## Calculate the average hourly temperature from the data
## obtained above ##
wx.ag <- NCEP.aggregate(wx.data=wx.extent, YEARS=FALSE, MONTHS=FALSE,
    HOURS=TRUE, DAYS=FALSE, fxn='mean')

## Now plot the mean temperature at midnight ##
## Note the adjustment of axis labels
## Note also the adjustment of the point type
NCEP.vis.area(wx=wx.ag, layer=1, interp.loess.args=list(span=0.5),
    title.args=list(main='Example: aggregated layer', xlab='Long [degrees]',
    ylab='Lat [degrees]'), points.args=list(pch=19))

## Now produce the same plot as above ##
## This time, change the font size used in the
## contour labels ##
NCEP.vis.area(wx=wx.ag, layer=1, interp.loess.args=list(span=0.5),
    title.args=list(main='Example: aggregated layer', xlab='Long [degrees]',
    ylab='Lat [degrees]'), points.args=list(pch=19),
    contour.args=list(labcex=.6))

## Notice how you can plot an aggregated layer by specifying
## it explicitly ##
NCEP.vis.area(wx=wx.ag, layer="XXXX-XX-XX 18", interp.loess.args=list(span=0.5),
    title.args=list(main='Example: aggregated layer', xlab='Long [degrees]',
    ylab='Lat [degrees]'), points.args=list(pch=19),
    contour.args=list(labcex=.6))

## End(Not run)
```

`NCEP.vis.points` *Visualize Weather Data Interpolated to a Point on a Map*

Description

This function creates a map with points. The color of the points indicates the value of some variable at that point. These values can e.g. be obtained by applying the function `NCEP.interp`.

Usage

```
NCEP.vis.points(wx, lats, lons, cols=heat.colors(64),
    transparency=.5, connect=TRUE, axis.args=NULL,
    points.args=NULL, map.args=NULL, grid.args=NULL,
    title.args=NULL, image.plot.args=NULL, lines.args=NULL)
```

Arguments

wx	A vector of weather data as returned by NCEP.interp
lats	A vector of latitudes in decimal degrees indicating the locations of the points
lons	A vector of longitudes in decimal degrees indicating the locations of the points
cols	A vector of colors such as that generated by rainbow, heat.colors, topo.colors, terrain.colors, or similar functions
transparency	A numeric value between 0 and 1 indicating the transparency of the filled points on the map.
connect	Logical. Should a line be drawn connecting the points?
axis.args	A list of arguments controlling the drawing of axes. See axis for acceptable arguments and the examples below for a demonstration.
points.args	A list of arguments controlling the drawing of points. See points for acceptable arguments and the examples below for a demonstration.
map.args	A list of arguments controlling the drawing of the map. See map for acceptable arguments and the examples below for a demonstration.
grid.args	A list of arguments controlling the drawing of the lat/long grid lines. See abline for acceptable arguments and the examples below for a demonstration.
title.args	A list of arguments controlling the how titles and axis lables are written. See title for acceptable arguments and the examples below for a demonstration.
image.plot.args	A list of arguments controlling the plotting of the color-bar legend and the legend axis and labels. See image.plot for acceptable arguments and the examples below for a demonstration.
lines.args	A list of arguments controlling the drawing of the line connecting the points. See lines for acceptable arguments and the examples below for a demonstration.

Details

Most of the components of a plot produced by this function can be controlled by supplying a list of arguments to the embedded function that produces the particular component of the plot. For example, the text and size of the plot's title can be controlled by specifying a list of acceptable arguments to title.args. Similarly, the axes, map, and grid lines are controlled by specifying a list of acceptable arguements to axis.args, map.args, and grid.args, respectively. Through the argument image.plot.args the user can control the plotting of the color-bar legend and the color-bar's title and axis labels. See the examples below for a demonstration of how to apply these different arguments.

Value

A plot is produced. No data are returned.

Author(s)

Michael U. Kemp <mukemp+RNCEP@gmail.com>

References

To cite package 'RNCEP' in publications use:

Kemp, M. U., van Loon, E. E., Shamoun-Baranes, J., and Bouten, W. 2011. RNCEP:global weather and climate data at your fingertips. – Methods in Ecology and Evolution. DOI:10.1111/j.2041-210X.2011.00138.x.

Examples

```
## Not run:
library(RNCEP)
## In this example, we use datetime and locational data
## obtained from a GPS device attached to a lesser
## black-backed gull.
data(gull, package='RNCEP')

## First, visualize the entire track representing altitude
## with the point colors ##
## Note the specification of the title
## Also, note the specification of the legend label
## and adjustment of its placement
NCEP.vis.points(wx=gull$altitude, lats=gull$latitude,
    lons=gull$longitude, cols=topo.colors(64),
    title.args=list(main='Lesser black-backed gull'),
    image.plot.args=list(legend.args=list(text='Altitude',
    adj=-1, cex=1.25)))

## Take a subset of the data based on the datetime of
## the measurement ##
ss <- subset(gull, format(gull$datetime, "%Y-%m-%d %H:%M:%S") >=
    "2008-09-19 16:00:00" & format(gull$datetime,
    "%Y-%m-%d %H:%M:%S") <= "2008-09-19 19:30:00")


## Now collect cloud cover, temperature, and wind
## information for each point in the subset ##
cloud <- NCEP.interp(variable='tcdc.eatm', level='gaussian',
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=TRUE, keep.unpacking.info=TRUE)
temp <- NCEP.interp(variable='air.sig995', level='surface',
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=FALSE, keep.unpacking.info=TRUE)
uwind <- NCEP.interp(variable='uwnd', level=925,
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=TRUE, keep.unpacking.info=TRUE)
vwind <- NCEP.interp(variable='vwnd', level=925,
    lat=ss$latitude, lon=ss$longitude, dt=ss$datetime,
    reanalysis2=TRUE, keep.unpacking.info=TRUE)

## Now visualize the subset of the GPS track using color
## to indicate the cloud cover ##
## Note the adjustment to the color of the basemap
```

```
## And the setting of the map range ##
## And the explicit placement of the colorbar legend
## using the smallplot argument
NCEP.vis.points(wx=cloud, lats=ss$latitude, lons=ss$longitude,
    cols=rev(heat.colors(64)),
    map.args=list(col='darkgreen',xlim=c(-7,4), ylim=c(40,50)),
    title.args=list(main='Lesser black-backed gull'),
    image.plot.args=list(legend.args=list(text='Cloud Cover %',
        adj=-.1, padj=-.5, cex=1),
    smallplot=c(.83,.86,.15,.85)))

## Now visualize the subset of the GPS track using color
## to indicate the temperature ##
## Note the adjustment of point size
NCEP.vis.points(wx=temp, lats=ss$latitude, lons=ss$longitude,
    cols=rev(heat.colors(64)),
    points.args=list(cex=1.25),
    title.args=list(main='Lesser black-backed gull'),
    image.plot.args=list(legend.args=list(text='Kelvin',
        adj=-.4, padj=-.5, cex=1.15)),
    map.args=list(xlim=c(-7,4), ylim=c(40,50)))

## Now calculate the tailwind component from the U and V
## wind components assuming that the bird's preferred
## direction is 225 degrees
tailwind <- (sqrt(uwind^2 + vwind^2)*cos(((atan2(uwind,vwind)*
    (180/pi))-225)*(pi/180)))

## Now visualize the subset of the GPS track using color
## to indicate the tailwind speed ##
## Note the adjustment of grid and axis properties
NCEP.vis.points(wx=tailwind, lats=ss$latitude, lons=ss$longitude,
    cols=rev(heat.colors(64)),
    axis.args=list(las=2), grid.args=list(lty=1),
    title.args=list(main='Lesser black-backed gull'),
    image.plot.args=list(legend.args=list(text='Tailwind m/s',
        adj=0, padj=-2, cex=1.15)),
    map.args=list(xlim=c(-7,4), ylim=c(40,50)))

## End(Not run)
```

Index