Package ‘meteoForecast’

March 2, 2021

Type Package
Title Numerical Weather Predictions
Version 0.54

URL https://github.com/oscarperpinan/meteoForecast
BugReports https://github.com/oscarperpinan/meteoForecast/issues
License GPL-3
LazyData yes
Depends raster, sp, zoo, ncdf4
Imports methods, stats, utils, XML
Suggests rgdal, lattice, rasterVis
NeedsCompilation no
Author Oscar Perpinan Lamigueiro [cre, aut], Marcelo Pinho Almeida [ctb]
Maintainer Oscar Perpinan Lamigueiro <oscar.perpinan@upm.es>
Repository CRAN
Date/Publication 2021-03-02 15:10:02 UTC

R topics documented:

meteoForecast-package .................................................. 2
Forecast variables ......................................................... 2
getPoint ................................................................. 3
getRaster ............................................................... 6
options ............................................................ 10
**Description**

meteoForecast is a package to access outputs from Numerical Weather Prediction models both in raster format and as a time series for a location. Currently it works with GFS, MeteoGalicia, NAM, and RAP.

**Details**

getRaster, getRasterDay, and getRasterDays get data inside a bounding box and provide a multilayer raster data using the RasterBrick class defined in the package raster.

getPoint, getPointDays, and getPointRuns get data for a certain location and produce a time series using the zoo class.

**Author(s)**

Oscar Perpiñán, with contributions from Marcelo Almeida

**References**

https://www.meteogalicia.es/web/modelos/threddsIndex.action
https://www.ncei.noaa.gov/thredds/catalog/model-gfs-003-files/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-nam218/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-rap130/catalog.html

**See Also**

raster zoo

**Forecast variables**

**Forecast Variables available in each model.**

**Description**

The grepVar retrieves the XML file with the names, description, and labels of each variable available in the service, and searches for matches in the description field.

**Usage**

grepVar(x, service, day = Sys.Date() - 1, complete = FALSE)
getPoint

Arguments

- **x**: character string to be matched in the description field of the set of variables. Try `x = ""` and `complete = TRUE` to get the complete list of choices with the description field.
- **service**: Character, to choose from 'meteogalicia', 'gfs', 'nam', and 'rap'
- **day**: Date. Services change the variables availability over time.
- **complete**: Logical, if FALSE (default) only the name of the variables is returned. If TRUE the name, label, and description columns are provided.

Value

If `complete = TRUE` this function provides a `data.frame` with three columns, name, label, and description. Use the elements of the name column to choose a variable with the argument `var` of `getRaster` and `getPoint`.

Source

http://mandeo.meteogalicia.es/thredds/catalogos/WRF_2D/catalog.html
http://mandeo.meteogalicia.es/thredds/catalog/gfs_0p25/fmrc/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-nam218/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-rap130/catalog.html

Examples

```r
## Not run:
## Variables available recently
grepVar("cloud", service = "gfs", complete = TRUE)

## Variables available some days ago
grepVar("cloud", service = "nam",
      day = Sys.Date() - 10,
      complete = TRUE)

## You can get the complete list with x = ""
grepVar("", service = "meteogalicia", complete = TRUE)

## End(Not run)
```

g RandomForest 3

**Description**

The `getPoint` functions get outputs of the NWP models run by MeteoGalicia and NCEP (GFS, RAP, NAM) for a single location.
Usage

getCode(point, vars = "swflx", day = Sys.Date(), run = "00",
       resolution = NULL, vertical = NA, service = mfService())

getCodeDays(point, vars = "swflx", start = Sys.Date(), end,
             service = mfService(), ...)

getCodeRuns(point, var = "swflx",
           start = Sys.Date() - 1, end = Sys.Date(),
           service = mfService(), ...)

Arguments

point Coordinates of the location. It can be a SpatialPoints or a numeric of length 2 (lon, lat).
var, vars Character. The name of the variables to retrieve. Use grepVar to know what variables are available in each service. getetCodeRuns only works with one variable.
day Date or character
run Character. The meteogalicia service executes the model at 00UTC and 12UTC. Therefore run can be '00' or '12'. With GFS and NAM run can be '00', '06', '12', and '18'. The RAP service is run every hour.
start Date or character. First day of the time period to retrieve.
end Date or character. Last day of the time period to retrieve.
resolution Numeric. Resolution in kilometers of the raster. Valid choices are 4, 12, and 36. It is only used with service = 'meteogalicia'.
vertical Numeric. Vertical coordinate for variables with several levels. Its default value is NA, meaning that only the first level will be retained.
service Character, which service to use, 'meteogalicia', 'gfs', 'nam', or 'rap'.
... Additional arguments for getetCode

Details

These functions download data from the MeteoGalicia and NCEP (GFS, RAP, NAM) servers using the NetCDF Subset Service. The result is returned as a zoo time series object, with one or more csv files stored in the temporary folder (as defined by tempdir()).

Value

getCode and getetCodeDays produce a zoo time series with a column for each variable included in vars.
The time series returned by getetCode starts at 01UTC of day if run = '00' or 13UTC if run = '12'. It spans over 4 days (96 hours) if run = '00' or 84 hours if run = '12'.
The time series returned by `getPointDays` starts at 01UTC of `start` and finishes at 00UTC of `end + 1`. Each day comprised in the time period is constructed with the forecast outputs corresponding to the 00UTC run of that day. Therefore, only the first 24 values obtained with `getPoint` are used for each day.

The time series returned by `getPointRuns` starts at 01UTC of `start` and finishes at 00UTC of `end + 1`. It has 4 columns, named "D3_00", "D2_00", "D1_00" and "D0_00". The column "D3_00" corresponds to the forecast results produced 3 days before the time stamp of each row, and so on.

**Author(s)**

Oscar Perpiñán Lamigueiro with contributions from Marcelo Almeida

**References**

http://mandeo.meteogalicia.es/thredds/catalogos/WRF_2D/catalog.html
http://mandeo.meteogalicia.es/thredds/catalog/gfs_0p25/fmrc/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-nam218/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-rap130/catalog.html

**See Also**

`getRaster`

**Examples**

```r
## Not run:
## If some of the next examples do not work, try using a different date. Check availability for each service with the links included in the references section.

testDay <- Sys.Date() - 1

## temperature (Kelvin) forecast from meteogalicia

tempK <- getPoint(c(0, 40), vars = 'temp', day = testDay)

## Cell does not coincide exactly with request

attr(tempK, 'lat')
attr(tempK, 'lon')

## Units conversion

tempC <- tempK - 273

library(lattice)

## Beware: the x-axis labels display time using your local timezone.

Sys.timezone()

## Use Sys.setenv(TZ = 'UTC') to produce graphics with the timezone of the objects provided by meteoforecast.

xyplot(tempC)

## Multiple variables

vars <- getPoint(c(0, 40), vars = c('swflx', 'temp'), day = testDay)

xyplot(vars)
```
## Vertical coordinates

tempK1000 <- getPoint(c(0,40),
                      vars = "Temperature_surface",
                      day = testDay,
                      service = "gfs", vertical = 1000)

## Time sequence

radDays <- getPointDays(c(0, 40),
                        start = testDay - 3,
                        end = testDay)

xyplot(radDays)

## Variability between runs

radRuns <- getPointRuns(c(0, 40),
                        start = testDay - 3,
                        end = testDay)

xyplot(radRuns, superpose = TRUE)

## variability around the average

radAv <- rowMeans(radRuns)

radVar <- sweep(radRuns, 1, radAv)

xyplot(radVar, superpose = TRUE)

## End(Not run)

---

getRaster  NWP forecasts for a region

description

The getRaster* functions get outputs of the NWP models for a region.

Usage

getRaster(var = "swflx", day = Sys.Date(), run = "00",
          frames = 'complete', box, resolution = NULL,
          names, remote = TRUE, service = mfService(),
          dataDir = ".", use00H = FALSE, ...)

getRasterDays(var = "swflx", start = Sys.Date(), end,
              remote = TRUE, dataDir = ".", ...)

getRasterDay(var = "swflx", day = Sys.Date(),
             remote = TRUE, dataDir = ".", ...)

checkDays(start, end, vars, remote = FALSE,
          service = mfService(), dataDir = ".")
getRaster

Arguments

The arguments to `getRaster` are:

```r
class <- c('var', 'vars')
char <- c('day', 'run', 'start', 'end', 'frames', 'box', 'resolution', 'names', 'remote', 'service', 'use00H', 'dataDir')
```

- **var, vars**: Character. The name of the variable (or variables in `checkDays`) to retrieve. Use `grepVar` to know what variables are available in each service.
- **day**: Date or character. In `getRaster` it defines the day when the forecast was produced. In `getRasterDay` it defines the day to be forecast.
- **run**: Character. For example, the meteogalicia service executes the model at 00 UTC and 12 UTC. Therefore `run` can be '00' or '12'.
- **start**: Date or character. First day of the time period to retrieve.
- **end**: Date or character. Last day of the time period to retrieve.
- **frames**: Numeric. It defines the number of hourly forecasts (frames) to retrieve. If `frames = 'complete'`, the full set of frames is downloaded. For example, the meteogalicia service produces 96 hourly forecasts (frames) with `run='00'` and 84 frames with `run='12'`.
- **box**: The bounding box, defined using longitude and latitude values. A `Extent` or an object that can be coerced to that class with `extent`: a 2x2 matrix (first row: xmin, xmax; second row: ymin, ymax), vector (length=4; order= xmin, xmax, ymin, ymax) or list (with at least two elements, with names 'x' and 'y').
- **resolution**: Numeric. Resolution in kilometers of the raster. Valid choices are 4, 12, and 36. It is only used with `service = 'meteogalicia'`.
- **names**: Character. Names of the layers of the resulting `RasterBrick`. If missing, a predefined vector is assigned the combination of day and hour.
- **remote**: Logical. If `TRUE` (default) data is downloaded from the remote service. If `FALSE` the `RasterBrick` is produced with the files available in the local folder.
- **service**: Character, which service to use, 'meteogalicia', 'gfs', 'nam' or 'rap'.
- **use00H**: Logical. Only used when `service` is 'gfs', 'nam', or 'rap'. If `FALSE` (default), the first frame of each run or 00h "forecast" is not considered. This first frame is only produced for some variables. Therefore, with `use00H = TRUE` fewer frames that the number defined with `frames` could be obtained for some variables.
- **dataDir**: Character, path of the folder where files are stored (if `remote = 'FALSE'`)
- ...: Additional arguments. Not used in `getRaster`.

Details

`getRaster` downloads data from the MeteoGalicia and NCDC (GFS, RAP, and NAM) servers using the NetCDF Subset Service. The result is returned as a `RasterBrick` object, with one or more NetCDF files stored in the temporary folder (as defined by `tempdir()`). Each frame or layer of the `RasterBrick` corresponds to a certain hour of the forecast.

`getRasterDay` uses `getRaster` to download the results corresponding to a certain day. If the day is in the future, the most recent forecast is downloaded with `getRaster`, and the corresponding frames are extracted. If the day is in the past, `getRaster` is used to download the corresponding frames of the forecast produced that day.

`getRasterDays` uses `getRaster` to download the results cast each day comprised between `start` and `end` using the 00UTC run. Then it subsets the first 24 frames of each result, and binds them
together to produce a RasterBrick. Therefore, each frame of this RasterBrick is a forecast for an hour of the day when the forecast was cast.

checkDays explores a local folder looking for NetCDF files corresponding to a time sequence and a set of variables. It returns a Date vector comprising the days with files available for the requested variables. If remote = TRUE it only checks that start is after 2008-01-01 (first date of the archived forecasts of MeteoGalicia.)

**Value**

The getRaster* functions return a RasterBrick with a layer for each hour of the NWP forecast. The time zone of the time index of this RasterBrick, stored in its z slot (accessible with getZ) is UTC.

MeteoGalicia, NAM, and RAP use the Lambert Conic Conformal projection. GFS files use longitude-latitude coordinates.

**Author(s)**

Oscar Perpiñán with contributions from Marcelo Almeida.

**References**

http://mandeo.meteogalicia.es/thredds/catalogos/WRF_2D/catalog.html
http://mandeo.meteogalicia.es/thredds/catalog/gfs_0p25/fmrc/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-nam218/catalog.html
https://www.ncei.noaa.gov/thredds/catalog/model-rap130/catalog.html

**Examples**

```r
## Not run:

## If some of the next examples do not work, try using a different date. Check availability for each service with the links included in the references section.

testDay <- Sys.Date() - 1

## Retrieve raster data
wrf <- getRaster(’temp’, day = testDay)

## Display results with rasterVis
library(rasterVis)
levelplot(wrf, layers = 10:19)

hovmoller(wrf)

## Using box and frames specification
mfExtent(’gfs’)
```
cloudGFS <- getRaster('Temperature_surface',
  day = testDay,
  box = c(-30, 30, 30, 50),
  service = 'gfs')

levelplot(cloudGFS, layout = c(1, 1))

mfExtent('nam')
cloudNAM <- getRaster('Temperature_surface',
  day = testDay,
  box = c(-100, -80, 30, 50),
  frames = 10,
  service = 'nam')

mfExtent('rap')
cloudRAP <- getRaster('Temperature_surface',
  day = testDay,
  box = c(-100, -80, 30, 50),
  frames = 10,
  service = 'rap')

## Day sequence of cloud cover
wrfDays <- getRasterDays('cft',
  start = testDay - 3,
  end = testDay + 2,
  box = c(-2, 35, 2, 40))

levelplot(wrfDays, layers = 10:19)

## animation
levelplot(wrfDays, layout = c(1, 1), par.settings = BTCTheme)

## Hövmoller graphic
hovmoller(wrfDays, par.settings = BTCTheme, contour = TRUE, cuts = 10)

NAMDays <- getRasterDays('Temperature_surface',
  start = testDay - 3,
  end = testDay,
  box = c(-100, -80, 30, 50),
  service = 'nam')

## Extract data at some locations
st <- data.frame(name=c('Almeria', 'Granada', 'Huelva', 'Malaga', 'Caceres'),
                  elev=c(42, 702, 38, 29, 448))

coordinates(st) <- cbind(c(-2.46, -3.60, -6.94, -4.42, -6.37),
                         c(36.84, 37.18, 37.26, 36.63, 39.47))
proj4string(st) <- '+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0'

## Extract values for some locations
options <- extract(wrf, st)
vals <- zoo(t(vals), getZ(wrf))
names(vals) <- st$name

xyplot(vals)

## End(Not run)

---

options  

Options and Internal Variables

Description

Functions to get or set options, and to access internal parameters of the package.

Usage

getMFOption(name = NULL)
setMFOption(name, value)
mfService(service = NULL)
mfExtent(service, resolution = 12)
mfProj4(service, resolution = 12)

Arguments

name  Character, name of the option to get or set.
value  Character, value of the option to be changed.
service  Character, name of the service ('meteogalicia', 'gfs', 'nam', 'rap').
resolution  Numeric, value of the resolution (in kilometers). Only useful if service = 'meteogalicia'

Details

Use getMFOption to list the options of the package. Only one option, service, is available with this version. With setMFOption the option defined with name can be modified.

mfService, a wrapper around getMFOption and setMFOption, displays the default service if used without arguments. It modifies the default service to the value of its argument.

mfExtent and mfProj4 provides the extent and the proj4 string of the corresponding service.

Author(s)

Oscar Perpiñán Lamigueiro
options

Examples

mfService()

mfExtent('meteogalicia', 36)

mfExtent('nam')

mfProj4('rap')
Index

* datasets
  Forecast variables, 2

* package
  meteoForecast-package, 2

* raster
  getRaster, 6

* spatial
  getPoint, 3
  getRaster, 6

* time series
  getPoint, 3

checkDays (getRaster), 6

Extent, 7
extent, 7

Forecast variables, 2

getMFOption (options), 10
getPoint, 3, 3, 4
getPointDays (getPoint), 3
getPointRuns (getPoint), 3
getRaster, 3, 5, 6
getRasterDay (getRaster), 6
getRasterDays (getRaster), 6
getz, 8
grepVar, 4, 7
grepVar (Forecast variables), 2

meteoForecast (meteoForecast-package), 2
meteoForecast-package, 2

mfExtent (options), 10
mfProj4 (options), 10
mfService (options), 10

options, 10

raster, 2

setMFOption (options), 10

zoo, 2