Package ‘daRt’

July 8, 2020

Type Package

Title Read DART Model Outputs

Version 0.7.2

Author William T. J. Morrison

Maintainer William T. J. Morrison <willmorrison661@gmail.com>

Description Easily read output data from the Discrete Anisotropic Radiative Transfer (DART) model and return in a ``long'' dplyr-ready format suitable for efficient analysis.

Github https://github.com/willmorrison1/daRt

License GPL-3

Encoding UTF-8

RoxygenNote 7.0.0

Depends dplyr (>= 0.7.6),
stringr (>= 1.4.0),
tibble (>= 2.1.3),
data.table (>= 1.12.0),
foreach (>= 1.4.7),
doParallel (>= 1.0.15),
reshape2 (>= 1.4.3),
shadowtext (>= 0.0.7),
fields (>= 10.0),
ncdf4 (>= 1.17),
chron (>= 2.3),
xml2 (>= 1.2.2),
tidyr (>= 1.0.0),
parallel,
tools,
raster (>= 3.0.0)

Remotes git::https://github.com/willmorrison1/QOLfuns.git

R topics documented:

accessors 2
as.data.frame,SimulationData-method 3
deleteFiles 3
Directions-class 4
getData 4

getFiles 5
Images-class 5
imagesToDirectionsDF 5
plotDirections 6
RB3D-class 7
rb3DtoNc 7
removeRelief 7
resourceUse 8
sequenceParameters 9
SimulationData-class 9
SimulationFiles-class 10
simulationFilter 10
SimulationFilter-class 11
sunAngles 12
tappToRadiance 12
versionInfo 13
wavelengths 13

Index **14**

`accessors` *Access object information*

Description

Generic functions to access information from the objects with classes defined in this package

Usage

```
product(x)

simname(x)

fileName(x)

bands(x)

iters(x)

variables(x)

variablesRB3D(x)

typeNums(x)

imageTypes(x)

imageNums(x)
```

Arguments

x	SimulationFilter or SimulationFiles class

Examples

```
sF <- simulationFilter(product = "directions")
bands(sF)

## Not run:
#access information within SimulationFiles object
#define the simulation directory
simDir <- "C:/Users/<Username>/DART/user_data/simulations/cesbio/"
simFiles <- getFiles(simDir)
#show bands that are selected
bands(simFiles)
#show 'type numbers' that have been selected
typeNums(simFiles)

## End(Not run)
```

---

as.data.frame,SimulationData-method *as.data.frame*

Description

as.data.frame

Usage

```
## S4 method for signature 'SimulationData'
as.data.frame(x, as.tibble = TRUE)
```

Arguments

`x`	SimulationData.
`as.tibble`	Return as a tibble-type data frame?

Value

data.frame or tibble

---

deleteFiles *deleteFiles*

Description

DART input files can be very large. This function deletes those large files that are not required for post-processing of data in this package.

Usage

```
deleteFiles(x = "SimulationFiles", deleteSimulationFiles = "logical", ...)
```

Arguments

x SimulationFiles-class type object.

deleteSimulationFiles
logical A hard check that you are happy to delete the files in x, shown by fileName(x).

... `maketOutput` remove "maket.txt" output file? (bool)

Details

Delete potentially large input files

`Directions-class` *Directions data class*

Description

Directions data class that extends SimulationData-class class.

`getData` *Main function: get DART data*

Description

Main function to get data from DART simulation outputs in a friendly 'long' data format that is part of an object that extends a SimulationData-class type object

Usage

```
getData(x, sF, ...)
```

Arguments

x simulation directory or directories (character) or SimulationFiles-class object

sF SimulationFilter-class if x = `character`

getFiles *Get DART output filenames*

Description

Function for getting SimulationFiles-class type object. Useful to perform a 'dry run' of getData by exploring the files that will vary based on the contents of `x` and the configuration of `sF`.

Usage

```
getFiles(x = "character", sF = "SimulationFilter")
```

Arguments

`x`	simulation directory or directories (character)
`sF`	SimulationFilter-class object
`...`	Optional arguments of: `nCores`: number of cores to use when loading data.

Images-class *Images data class*

Description

Image data class extends SimulationData-class class.

imagesToDirectionsDF *imagesToDirectionsDF*

Description

Convert an Images-class object to a Directions-class object

Usage

```
imagesToDirectionsDF(x, fun)
```

Arguments

`x`	Images-class object
`fun`	Function to apply across each image.

Details

Aggregate images to single values

Value

data frame

`plotDirections` *plotDirections*

Description

Plot directions data as polar plot.

Usage

```
plotDirections(
  azimuth,
  zenith,
  value,
  azimuthOffsetVal = 0,
  outerRadius = max(zenith) + max(zenith) * 0.01,
  zenithLabPch = 20,
  zenithLabCol = "darkgrey",
  zenithLabCex = 1,
  brks = seq(min(value), max(value), length.out = 10),
  cols = c("dark grey", colorRampPalette(c("purple", "blue3", "yellow",
    "red"))(length(brks) - 3), "firebrick4"),
  ...
)
```

Arguments

`azimuth`	Numeric. Azimuth angle with DART conventions
`zenith`	Numeric. Zenith angle with DART conventions
`value`	Numeric. Values associated with the given azimuth and zenith angles
`azimuthOffsetVal`	Numeric. Scene offset (degrees) as shown in the DART GUI.
`outerRadius`	Numeric. Maximum radius (degrees) of polar plot
`zenithLabPch`	Numeric. Pch for zenith label.
`zenithLabCol`	Character. Colour for zenith label.
`zenithLabCex`	Numeric. Cex for zenith label.
`brks`	Numeric. Breaks for colour palette e.g. seq(0, 1, by = 0.1). Optional.
`cols`	Character. Colours for given breaks. Optional.
`...`	Additional options passed to points() when drawing directions points.

Examples

```
#Inputs are DART oriented directions (as seen in the DART files and \link{Directions-class})
plotDirections(azimuth = rep(225, 10),
               zenith = seq(0, 90, length.out = 10),
               value = 1:10)
#Output plot uses 'upward' directions from ground, where e.g.:
  0deg (270deg) azimuth faces north (west)
  0deg (90deg) zenith faces upward (horizon)
```

`RB3D-class` *RB3D class*

Description

RB3D (Radiative Budget 3D) class that extends SimulationData-class class.

`rb3DtoNc` *rb3DtoNc*

Description

DART radiative budget .bin files can be very large. This function replaces all .bin files with .nc files, which can be compressed and are faster to read.

Usage

```
rb3DtoNc(x = "SimulationFiles", ...)
```

Arguments

`x`	SimulationFiles-class type object.
`ncCompressionFactor`	Compression factor (0 - 9) for writing ncdf files (see ncdf4 package)

Details

Convert radiative budget .bin to .nc

Value

SimulationFiles-class type object.

`removeRelief` *removeRelief*

Description

Remove underlying orography from a RB3D-class dataset using a digital elevation model (DEM) of class `RasterLayer` that is georeferenced to RB3D-class.

Usage

```
removeRelief(x = "RB3D", DEM = "RasterLayer", ...)
```

Arguments

`x`	RB3D-class type object.
`DSM`	`RasterLayer` type object with height above ground level (m) and - preferably - a finer
`BOAextrapolation`	Character. When the 3D radiative budget is height-adjusted, the BOA layer is no longer plane-parallel with the ground. How to make the BOA layer plane-parallel with the ground? One of "extrapolate" or "clip". Extrapolate: the highest BOA cell with a recorded value is the new BOA layer. Other cells in this horizontal layer may be empty and are filled using values from lower vertical layers (most accurate, most cells, most memory). Clip: the first BOA cell where all cells in its horizontal layer have a recorded value is the new BOA layer. All cells above this layer are removed. (Least accurate, least cells, least memory).
`‘maxUndergroundCells‘`	Integer. How many cells below the "ground" should be kept? I.e. the 3D RB array will be offset with Z=0 as the new ground level, and Z=-maxUndergroundCells as the lowest elevation to keep. Cells below -maxUndergroundCells are removed as this saves a lot of memory. If there is lots of small-scale variation in topography then this parameter should be relaxed at the expense of array size and memory usage.

Details

Remove underlying orography

`resourceUse` *ResourceUse*

Description

Return a data frame with information on the resource use for a SimulationFiles-class type object

Usage

```
resourceUse(x = "SimulationFiles")
```

Arguments

`x`	SimulationFiles-class type object

Details

Return resource use

`sequenceParameters` *sequenceParameters*

Description

Return a data frame where rows describe a parameter (parametre*) for a simulation (simName).

Usage

```
sequenceParameters(x)
```

Arguments

`SimulationFiles-class`
or SimulationData-class class object or character string of simulation directories

Details

Get data frame of all sequence parameters

Value

data frame

`SimulationData-class` *Generic SimulationData class*

Description

Generic SimulationData class that extends to data classes for specific DART products

Slots

`data` data.frame.

See Also

Images-class Directions-class RB3D-class

SimulationFiles-class *SimulationFiles class*

Description

An S4 class to represent the files within a simulation or simulations. Created using the getFiles method. Specific files within the class are modified by the object with class SimulationFilter-class

Usage

```
baseDir(x)

simulationFilter(x) <- value
```

Slots

`simulationFilter` contains SimulationFilter-class object

`files` a data.frame, with each row describing the file

`sequenceInfoList` a list, with each list element showing the variable permutation(s) within this specific simulation sequence.

`sequenceInfoDf` a data frame, with each row containing one simulation, and each column a parameter ('parametre') specific to the sequence. A condensed version of sequenceInfoList.

`wavelengths` a data frame containing spectral information on each band for each simulation

`sunAngles` a data frame containing sun angles straight from simulation.properties.txt

simulationFilter *Create SimulationFilter class*

Description

Function for creating the SimulationFilter class. Define a product, then Optional arguments of: 'bands', 'variables', 'iterations', 'variablesRB3D', 'typeNums', 'imageTypes', 'imageNums'. See SimulationFilter-class for full description.

Usage

```
simulationFilter(product = "character", x, ...)
```

Arguments

`product`	One of: 'directions', 'rb3D', 'images'.
`x`	SimulationFiles-class object if product is missing.

Value

SimulationFilter type object

See Also

SimulationFilter-class

Examples

```
sF <- daRt::simulationFilter(product = "images",
                             bands = as.integer(0:2),
                             iters = c("ITER1", "ITER2"),
                             variables = "BRF",
                             imageNums = as.integer(c(5, 7)),
                             imageTypes = c("ima", "ima_transmittance"))
```

SimulationFilter-class

SimulationFilter class.

Description

SimulationFilter class.

Usage

```
product(x) <- value

iters(x) <- value

bands(x) <- value

variablesRB3D(x) <- value

variables(x) <- value

typeNums(x) <- value

imageTypes(x) <- value

imageNums(x) <- value

subDir(x)
```

Slots

`bands` integer e.g. 0 for "BAND0"

`variables` character e.g. "BRF".

`iters` character e.g. "ITERX".

`variablesRB3D` character e.g. "Irradiance".

`typeNums` character e.g. "2_Ground".

`imageTypes` character e.g. "ima".

`imageNums` integer

`product` character e.g. "directions".

See Also

`simulationFilter`

`sunAngles` *sunAngles*

Description

Get sun angles for each simulation

Usage

```
sunAngles(x = "SimulationFiles")
```

Arguments

`x`	sF SimulationFiles-class

Value

data frame

`tappToRadiance` *tappToRadiance*

Description

Convert Tapp (K) to Radiance (W m2 sr-1 um-1) using Planck function at the equivalent Band wavelength

Usage

```
tappToRadiance(x = "SimulationData")
```

Arguments

`x`	SimulationData-class type object.

Details

Convert Tapp to Radiance

Value

SimulationData-class type object.

`versionInfo` *versionInfo*

Description

Get the version used for the given simulation data

Usage

```
versionInfo(x)
```

Arguments

`x`	SimulationFiles-class object

Details

Simulation version info

`wavelengths` *wavelengths*

Description

Get full information on wavelengths for each band

Usage

```
wavelengths(x = "SimulationFiles")
```

Arguments

`x`	sF SimulationFiles-class

Value

data frame

Index

accessors, 2
as.data.frame,SimulationData-method, 3

bands *(*accessors*)*, 2
bands<- *(*SimulationFilter-class*)*, 11
baseDir *(*SimulationFiles-class*)*, 10

deleteFiles, 3
Directions-class, 4, *9*

fileName *(*accessors*)*, 2

getData, 4, *5*
getFiles, 5, *10*

imageNums *(*accessors*)*, 2
imageNums<- *(*SimulationFilter-class*)*, 11
Images-class, *5*, 5, *9*
imagesToDirectionsDF, 5
imageTypes *(*accessors*)*, 2
imageTypes<- *(*SimulationFilter-class*)*, 11
iters *(*accessors*)*, 2
iters<- *(*SimulationFilter-class*)*, 11

plotDirections, 6
product<- *(*SimulationFilter-class*)*, 11

RB3D-class, *7*, 7, *8*, *9*
rb3DtoNc, 7
removeRelief, 7
resourceUse, 8

sequenceParameters, 9
simname *(*accessors*)*, 2
SimulationData-class, *4*, *5*, *7*, *9*, 9, *12*
SimulationFiles, *2*
SimulationFiles-class, *4*, *5*, *7–10*, 10, *12*, *13*
SimulationFilter, *2*, *10*
simulationFilter, 10, *12*
SimulationFilter-class, *4*, *5*, *10*, 11
simulationFilter<- *(*SimulationFiles-class*)*, 10
subDir *(*SimulationFilter-class*)*, 11
sunAngles, 12

tappToRadiance, 12
typeNums *(*accessors*)*, 2
typeNums<- *(*SimulationFilter-class*)*, 11

variables *(*accessors*)*, 2
variables<- *(*SimulationFilter-class*)*, 11
variablesRB3D *(*accessors*)*, 2
variablesRB3D<- *(*SimulationFilter-class*)*, 11
versionInfo, 13

wavelengths, 13