Package: pemisc (via r-universe)

September 8, 2024

Type Package

Title Miscellaneous Utilities Developed by the Predictive Ecology Group

Description Miscellaneous utilities developed by the Predictive Ecology Group (<http://predictiveecology.org>).

URL http://pemisc.predictiveecology.org, https://github.com/PredictiveEcology/pemisc

Date 2023-09-27

Version 0.0.4.9011

Depends R (>= 4.1)

Imports amc (>= 0.1.6.9000), data.table, fasterize, methods, parallel, parallelly, peutils, raster, RCurl, reproducible, Require, sf, stats, tools, utils

Suggests igraph, knitr, roxygen2, rmarkdown, terra, testthat

Remotes achubaty/amc@development, PredictiveEcology/peutils@development

Encoding UTF-8

Language en-CA

License GPL-3

VignetteBuilder knitr, rmarkdown

BugReports https://github.com/PredictiveEcology/pemisc/issues

ByteCompile yes

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Repository https://predictiveecology.r-universe.dev

RemoteUrl https://github.com/PredictiveEcology/pemisc

RemoteRef development

RemoteSha 022b049c5d98f010bef84457ee6558f50cc11a84

Contents

pemisc-package . . . 2
availableMemory . . . 2
biomassCalculation . . . 3
createPrjFile . . . 4
factorValues2 . . . 5
getLocalArgsFor . . . 5
ghPkgName . . . 6
identifyVectorArgs . . . 6
isGitHubPkg . . . 7
makeClusterRandom . . . 8
makeIpsForNetworkCluster . . . 8
makeOptimalCluster . . . 9
Map2 . . . 10
MapOrDoCall . . . 11
messageDF . . . 11
normalizeStack . . . 12
numActiveThreads . . . 12
optimalClusterNumGeneralized . . . 13
pkgDepsGraph . . . 14
prepFireCanada . . . 15
rasterToMatch . . . 16
reproducibilityReceipt . . . 16
termsInData . . . 17

Index **18**

`pemisc-package` `pemisc` *package*

Description

Miscellaneous utilities developed by the Predictive Ecology Group (https://predictiveecology.org).

`availableMemory` *Report the estimated amount of available memory in the OS*

Description

This reports the 'available' memory from a system call: `free` on Linux, `vm_stat` on macOS, or `wmic` on Windows. If neither is installed on the system, returns `NULL`.

Usage

```
availableMemory()
```

Value

Numeric of class "object_size", so it can be reported in any units with format, e.g., `format(availableMemory(), unit = "GB")`.

See Also

`man free` for description of available memory estimation.

`biomassCalculation` *Calculate above ground biomass for Canadian tree species*

Description

Based on DBH or DBH and Height.

Usage

```
biomassCalculation(species, DBH, includeHeight, height, equationSource)

## S4 method for signature 'character,numeric,logical,numeric,character'
biomassCalculation(species, DBH, includeHeight, height, equationSource)

## S4 method for signature 'character,numeric,missing,numeric,character'
biomassCalculation(species, DBH, height, equationSource)

## S4 method for signature 'character,numeric,logical,numeric,missing'
biomassCalculation(species, DBH, includeHeight, height)

## S4 method for signature 'character,numeric,missing,numeric,missing'
biomassCalculation(species, DBH, height)

## S4 method for signature 'character,numeric,missing,missing,character'
biomassCalculation(species, DBH, equationSource)

## S4 method for signature 'character,numeric,missing,missing,missing'
biomassCalculation(species, DBH)
```

Arguments

`species`	Character string giving the species name.
`DBH`	Numeric. The tree's diameter at breast height (DBH, cm).
`includeHeight`	Logical. Whether the biomass is calculated based on DBH and height. If `TRUE`, height must be provided. Default `FALSE`.
`height`	Numeric. The tree's height (m).
`equationSource`	Character. Determine the sources of equations. Currently, this function has two options, i.e., `"Lambert2005"` and `"Ung2008"`. Default, `"Lambert2005"`.

Value

Biomass (kg) and `missedSpecies` list that was not calculated.

Author(s)

Yong Luo

Examples

```
## Not run:
 DBH <- seq(1, 100, 5)
 species <- c(rep("jack pine", 10), rep("black spruce", 10))
 species[1] <- "wrongSpecies"
 height <- seq(20, 40, length = 20)

 # without height information and and taking the equations from Lambert 2005
 biomass1 <- biomassCalculation(species = species, DBH = DBH)

 # with height information and and taking the equations from Lambert 2005
 biomass2 <- biomassCalculation(species = species, DBH = DBH,
                                includeHeight = TRUE, height = height)

## End(Not run)
```

`createPrjFile` *Create a* `.prj` *file*

Description

In cases where a shapefile is missing its associated `.prj` file.

Usage

```
createPrjFile(
  shpFile,
  urlForProj = "http://spatialreference.org/ref/epsg/nad83-utm-zone-11n/prj/"
)
```

Arguments

`shpFile`	The filename of a shapefile to add `.prj`
`urlForProj`	The url from which to fetch the projection, e.g., "`http://spatialreference.org/ref/epsg/nad83-ut`

factorValues2 *Faster version of* raster::factorValues()

Description

Note there is an option to remove the NAs, which will make it MUCH faster, if TRUE

Usage

```
factorValues2(x, v, layer, att, append.names, na.rm = FALSE)
```

Arguments

x	Raster* object
v	integer cell values
layer	integer > 0 indicating which layer to use (in a RasterStack or RasterBrick)
att	numeric or character. Which variable(s) in the RAT table should be used. If NULL, all variables are extracted. If using a numeric, skip the first two default columns
append.names	logical. Should names of data.frame returned by a combination of the name of the layer and the RAT variables? (can be useful for multilayer objects
na.rm	Logical. If TRUE, then the NAs will be removed, returning a possibly shorter vector

getLocalArgsFor *Find sources for arguments in arbitrary function(s)*

Description

Search among local objects (which will often be arguments passed into a function) as well as dot objects to match the formals needed by fn. If localFormalArgs is named, then it will match the formal (name of localFormalArgs) with the local object, e.g., localFormalArgs = c(x = "obj") will find the object in the local environment called "obj", and this will be found because it matches the x argument in fn.

Usage

```
getLocalArgsFor(fn, localFormalArgs, envir, dots)
```

Arguments

fn	Function name(?)
localFormalArgs	A (named) character vector or arguments to
envir	The environment in which to (???)
dots	TODO: need description

Value

List of named objects. The names are the formals in `fn`, and the objects are the values for those formals. This can easily be passed to `do.call(fn, args1)`

---

ghPkgName *Get the package name from a GitHub* `repo/package@branch` *string*

Description

Get the package name from a GitHub `repo/package@branch` string

Usage

```
ghPkgName(x)
```

Arguments

`x`	character vector of package names

Value

a named character vector

Examples

```
pkgs <- c("dplyr", "PredictiveEcology/pemisc", "PredictiveEcology/SpaDES.core@development")
ghPkgName(pkgs) ## "dplyr" "pemisc" "SpaDES.core"
```

---

identifyVectorArgs *Identify the source for arguments passed to an arbitrary function*

Description

When running arbitrary functions inside other functions, there is a common construct in R to use `....` It does not work, however, in the general case to write `do.call(fn, list(...))` because not all `fn` themselves accept `....` So this will fail if too many arguments are supplied to the `....` In the general case, we want to write: `do.call(fn, list(onlyTheArgumentsThatAreNeeded))`. This function helps to find the `onlyTheArgumentsThatAreNeeded` by determining a) what is needed by the `fn` (which can be a list of many `fn`), and b) where to find values, either in an arbitrary environment or passed in via `dots`.

Usage

```
identifyVectorArgs(fn, localFormalArgs, envir, dots)
```

Arguments

fn	A function or list of functions from which to run formalArgs
localFormalArgs	A vector of possible objects, e.g., from ls()
envir	The environment to find the objects named in localFormalArgs
dots	Generally list(...), which would be an alternative place to find localFormalArgs

Value

A list of length 2, named argsSingle and argsMulti, which can be passed to e.g., MapOrDoCall(fn, multiple = args1$argsMulti, single = args1$argsSingle)

---

isGitHubPkg *Check whether a package is one installed from GitHub*

Description

Determines whether a string that may correspond to a package name (e.g., repo/package@branch), could be a package installed from GitHub. This is determined solely by the presence of a / in the string. See example below.

Usage

```
isGitHubPkg(x)
```

Arguments

x	character vector of package names

Value

a named logical vector

Examples

```
pkgs <- c("dplyr", "PredictiveEcology/pemisc", "PredictiveEcology/SpaDES.core@development")
isGitHubPkg(pkgs) ## FALSE TRUE TRUE
```

makeClusterRandom makeForkCluster *with random seed set*

Description

This will set different random seeds on the clusters (not the default) with makeForkCluster. It also defaults to creating a logfile with message of where it is.

Usage

```
makeClusterRandom(
  ...,
  type = "SOCK",
  iseed = NULL,
  libraries = NULL,
  objects = NULL,
  envir = parent.frame()
)

makeForkClusterRandom(..., iseed = NULL)

makeSockClusterRandom(..., iseed = NULL)
```

Arguments

...	passed to makeCluster, e.g.,
type	One of the supported types: see 'Details'.
iseed	passed to clusterSetRNGStream
libraries	A character vector of libraries to load in the SOCK cluster. This is ignored if a "FORK" cluster
objects	a character string of objects that are required inside the SOCK cluster. Ignored if type != "SOCK"
envir	Required if objects is passed. The environment where objects are found.

makeIpsForNetworkCluster

Create IP addresses for network cluster

Description

makeIpsForNetworkCluster is a simple wrapper around makeIps.

Usage

```
makeIpsForNetworkCluster(
  ipStart = "10.20.0",
  ipEnd = c(68, 97, 189, 213, 220, 58, 106, 184, 217),
  availableCores = c(50, 50, 50, 50, 50, 50, 23, 23, 23),
  availableRAM = c(950, 500, 500, 500, 500, 500, 245, 245, 245),
  nProcess = 8,
  proc = "cores",
  internalProcesses = 10,
  sizeGbEachProcess = 35,
  localHostEndIp = 68
)

makeIps(machines, ipStart, proc, nProcess, sizeGbEachProcess)
```

Arguments

`ipStart`	Network address prefix (i.e., the first, second, and third triplets of the IP address)
`ipEnd`	Host IP address identifier (i.e., the final triplet of the IP address)
`availableCores`	the number of available threads on each machine.
`availableRAM`	the available RAM on each machine in GB
`nProcess`	the number of processes
`proc`	one of `"cores"` or `"ram"`, describing the limiting factor of the cluster computations
`internalProcesses`	DESCRIPTION NEEDED
`sizeGbEachProcess`	the size in GB of each process
`localHostEndIp`	the address in `ipEnd` corresponding to local host
`machines`	`data.frame` of compute node information containing the following columns: `ipEnd`, `availableCores`, `availableRam`

Value

A vector of IP addresses associated with each machine in the network cluster.

`makeOptimalCluster` *Create a parallel fork cluster*

Description

Given the size of a problem, it may not be useful to create a cluster. This will make a fork cluster (so Linux only).

Usage

```
makeOptimalCluster(
  useParallel = getOption("pemisc.useParallel", FALSE),
  MBper = 500,
  maxNumClusters = parallelly::availableCores(constraints = "connections"),
  assumeHyperThreads = FALSE,
  ...
)
```

Arguments

useParallel	Logical or numeric. If FALSE, returns NULL. If numeric, then will return a cluster object with this many cores, up to maxNumClusters
MBper	Numeric. Passed to memRequiredMB in optimalClusterNum()
maxNumClusters	Numeric or Integer. The theoretical upper limit for number of nodes to use with the cluster.
assumeHyperThreads	Logical. If TRUE, then it will more efficiently divide the maxNumClusters by useParallel, so that there is a lower number of cores used. This calculation may not be the ideal balance. A message will indicate the change from maxNumClusters, if there is one.
...	Passed to makeForkClusterRandom. Only relevant for iseed.

Map2 Map *and* parallel::clusterMap *together*

Description

This will send to Map or clusterMap, depending on whether cl is provided. Because they use different argument names for the main function to call, leave that argument unnamed.

Usage

```
Map2(f, ..., cl = NULL)
```

Arguments

f	passed as f to Map or fun to clusterMap
...	passed to Map or clusterMap
cl	A cluster object, passed to clusterMap

Examples

```
## Not run:
a <- 1:5
Map2(a, f = function(x) x)

## End(Not run)
```

MapOrDoCall — Map/lapply *all in one*

Description

Usually run after identifyVectorArgs which will separate the arguments into vectors of values for a call to Map, and arguments that have only one value (passed to MoreArgs in Map). If all are single length arguments, then it will pass to lapply. If a cl is provided and is non-NULL, then it will pass all arguments to clusterMap or clusterApply.

Usage

```
MapOrDoCall(fn, multiple, single, useCache = FALSE, cl = NULL)
```

Arguments

fn	The function that will be run via Map/clusterMap.
multiple	This a list the arguments that Map will cycle over.
single	Passed to MoreArgs in the mapply function.
useCache	Logical indicating whether to use the cache.
cl	A cluster object or NULL.

See Also

identifyVectorArgs

messageDF — *Use message to print a clean, rectangular data structure*

Description

Sends to message, but in a structured way so that a data.frame-like can be cleanly sent to messaging.

Usage

```
messageDF(df, round, colour = NULL)
```

Arguments

df	A data.frame, data.table, matrix
round	An optional numeric to pass to round
colour	An optional colour to use from crayon

normalizeStack *Normalize each layer of a* RasterStack

Description

Rescales the values of of each RasterLayer between [0,1].

Usage

```
normalizeStack(x)
```

Arguments

x	A RasterStack object.

Author(s)

Tati Micheletti

numActiveThreads *Count number of active threads*

Description

This uses ps -ef so only works on unix-alikes. It will search for the percent CPU use and select only those above 40

Usage

```
numActiveThreads(pattern = "--slave", minCPU = 50)
```

Arguments

pattern	Character string that will be matched to the ps call
minCPU	A numeric indicating what percent is the minimum to be considered "active"

Value

A numeric of the number of active threads that match the pattern

Author(s)

Eliot McIntire

Examples

```
## Not run:
 ## Determine how many threads are used in each remote machine in a cluster
  cores = "localhost" # put other machine names here
  uniqueCores <- unique(cores)
  cl <- future::makeClusterPSOCK(uniqueCores, revtunnel = TRUE)
  clusterExport(cl, "numActiveThreads")
  out <- clusterEvalQ(cl, {
    numActiveThreads()
  })
  names(out) <- uniqueCores
  unlist(out)
  stopCluster(cl)


## End(Not run)
```

---

optimalClusterNumGeneralized

Determine the number of nodes to use in a new cluster

Description

Optimally determine the number of cores to use to set up a new cluster, based on:

1. the number of cores available (see note);
2. the amount of free memory available on the local machine;
3. the number of cores requested vs. the number available, such that if requesting more cores than available, the number of cores used will be adjusted to be a multiple of the number of cores needed, so jobs can be run in approximately-even-sized batches. (E.g., if 16 cores available but need 50, the time taken to run 3 batches of 16 plus a single batch of 2 – i.e., 4 batches total – is the same as running 4 batches of 13.)

Usage

```
optimalClusterNumGeneralized(
  memRequiredMB = 500,
  maxNumClusters = parallelly::availableCores(constraints = "connections"),
  NumCoresAvailable = parallelly::availableCores(constraints = "connections"),
  availMem = pemisc::availableMemory()/1e+06
)

optimalClusterNum(
  memRequiredMB = 500,
  maxNumClusters = parallelly::availableCores(constraints = "connections")
)
```

Arguments

memRequiredMB	The amount of memory needed in MB
maxNumClusters	The number of nodes needed (requested)
NumCoresAvailable	The number of cores available on the local machine (see note).
availMem	The amount of free memory (RAM) available to use.

Value

integer specifying the number of cores

Note

R hardcodes the maximum number of socket connections it can use (currently set to 128 in R 4.1). Three of these are reserved for the main R process, so practically speaking, a user can create *at most* 125 connections e.g., when creating a cluster. See https://github.com/HenrikBengtsson/Wishlist-for-R/issues/28.

We limit this a bit further here just in case the user already has open connections.

---

pkgDepsGraph *Build the pkg dependency graph*

Description

Uses **igraph** and Require::pkgDep.

Usage

```
pkgDepsGraph(
  pkgs = c("LandR", "pemisc", "map", "SpaDES", "SpaDES.tools", "SpaDES.core",
    "SpaDES.addins", "SpaDES.shiny", "reproducible", "quickPlot"),
  plot.it = TRUE
)
```

Arguments

pkgs	A character vector of package names. Default is c("LandR", "pemisc", "map", "SpaDES", "SpaDES.tools", "SpaDES.core", "SpaDES.addins", "SpaDES.shiny", "reproducible", "quickPlot")
plot.it	Logical. If TRUE, it will plot the igraph.

Value

A list of 2: dt a data.table of the dependencies, and dtGraph an igraph object that can be plotted with plot()

prepFireCanada *Download the National Burn Area Composite (Fires) in Canada*

Description

Downloads data from CWFIS Datamart at http://cwfis.cfs.nrcan.gc.ca/datamart. This runs prepInputs internally, so use can pass studyArea etc.

Usage

```
prepFireCanada(
  year,
  type = c("NBAC", "Polygon", "Point"),
  urlBase = "http://cwfis.cfs.nrcan.gc.ca/downloads/nbac/",
  ...
)
```

Arguments

year	Numeric, length 1. Which year, from 1986 to 2018 (currently) to download
type	Either "NBAC", "Polygon" or "Point" to get the National Burn Area Composite, the Polygon or the Point datasets.
urlBase	The url of the directory where the NBAC are stored. Default is the currently known url. If this url becomes stale, please notify the predictive ecology team.
...	Additional arguments.

Value

A SpatialPolygonsDataFrame plus several downloaded files, including the '.zip' archive and the extracted files. Because it is running prepInputs, checksumming is occurring too.

Examples

```
## Not run:
  # This will download 2 recent years
  library(sf)
  NBAC <- lapply(2016:2017, function(yr) a <- prepFireCanada(yr))
  Points <- prepFireCanada(yr, type = "Points", fun = "st_read")
  Polygons <- prepFireCanada(yr, type = "Polygons")

## End(Not run)
```

rasterToMatch *Extract or create a raster to match*

Description

This extracts or creates a new raster layer, whose intention is to be used as the `rasterToMatch` argument in further `prepInputs` calls.

Usage

```
rasterToMatch(x, ...)

## S4 method for signature 'Raster'
rasterToMatch(x, studyArea, ...)

## S4 method for signature 'SpatialPolygonsDataFrame'
rasterToMatch(x, studyArea, rasterToMatch, ...)
```

Arguments

`x`	A Raster Layer with correct resolution and origin.
`...`	Additional arguments
`studyArea`	A `SpatialPolygon*` object that will be sent to `postProcess`.
`rasterToMatch`	The raster to match in a `fasterize` call.

Value

A `RasterLayer` object.

reproducibilityReceipt *Deprecated functionality*

Description

Deprecated functionality

Usage

```
reproducibilityReceipt(title)
```

Arguments

`title`	Header title for the inserted details section.

termsInData *Extract terms in a quoted model statement*

Description

Similar to terms, but this is used on a quoted model and will only return unique matches in a data.

Usage

```
termsInData(model, data)
```

Arguments

model	A quoted model statement
data	A data.frame-like object with column names in which to match terms in model

Index

availableMemory, 2

biomassCalculation, 3
biomassCalculation,character,numeric,logical,missing,missing,character-method
(biomassCalculation), 3
biomassCalculation,character,numeric,logical,numeric,character-method
(biomassCalculation), 3
biomassCalculation,character,numeric,missing,missing,missing-method
(biomassCalculation), 3
biomassCalculation,character,numeric,missing,numeric,character-method
(biomassCalculation), 3
biomassCalculation,character,numeric,missing,numeric,missing-method
(biomassCalculation), 3

createPrjFile, 4

factorValues2, 5

getLocalArgsFor, 5
ghPkgName, 6

identifyVectorArgs, 6
isGitHubPkg, 7

makeClusterRandom, 8
makeForkClusterRandom
(makeClusterRandom), 8
makeIps (makeIpsForNetworkCluster), 8
makeIpsForNetworkCluster, 8
makeOptimalCluster, 9
makeSockClusterRandom
(makeClusterRandom), 8
Map2, 10
MapOrDoCall, 11
messageDF, 11

normalizeStack, 12
numActiveThreads, 12

optimalClusterNum
(optimalClusterNumGeneralized), 13
optimalClusterNum(), 10
optimalClusterNumGeneralized, 13
optimalClusterNum,numeric,missing-method
pemisc (pemisc-package), 2
pemisc-package, 2
pkgDepsGraph, 14
prepFireCanada, 15

raster::factorValues(), 5
rasterToMatch, 16
rasterToMatch,Raster-method
(rasterToMatch), 16
rasterToMatch,SpatialPolygonsDataFrame-method
(rasterToMatch), 16
reproducibilityReceipt, 16

termsInData, 17