Package: fireSenseUtils (via r-universe)

October 2, 2024

Type Package

Title Utilities for Working With the 'fireSense' Group of 'SpaDES' Modules

Description Utilities for working with the 'fireSense' group of 'SpaDES' modules.

Date 2024-07-10

Version 0.0.5.9073

Depends R (>= 4.2)

Imports crayon, data.table, DEoptim, EnvStats, fastDummies, kSamples, LandR (>= 1.1.0.9066), mclust, methods, qs, quickPlot, parallelly, pemisc (>= 0.0.4.9011), pROC, purrr, reproducible (>= 2.0.12), RhpcBLASctl, sf, sp, SpaDES.tools (>= 2.0.4), stats, terra, utils

Suggests cowplot, dplyr, ggplot2, knitr, lintr, PtProcess, raster, rasterVis, RColorBrewer, Require (>= 0.3.1), remotes, rmarkdown, SpaDES.core, spatstat, spelling, testthat

Remotes PredictiveEcology/LandR@development, PredictiveEcology/pemisc@development, PredictiveEcology/SpaDES.core@development, PredictiveEcology/SpaDES.tools@development, PredictiveEcology/Require@development, PredictiveEcology/reproducible@development

VignetteBuilder knitr, rmarkdown

Encoding UTF-8

Language en-CA

LazyData true

License GPL-3

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

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

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

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

RemoteRef development

RemoteSha 492560cf40d720ba03f6131be67235ccc3c23e75

Contents

`fireSenseUtils-package` `fireSenseUtils` *package*

Description

Utilities for working with the 'fireSense' group of 'SpaDES' modules.

Author(s)

Maintainer: Eliot J B McIntire <eliot.mcintire@nrcan-rncan.gc.ca> (ORCID)

Authors:

- Jean Marchal <jean.d.marchal@gmail.com>
- Alex M Chubaty <achubaty@for-cast.ca> (ORCID)
- Ian Eddy <ian.eddy@nrcan-rncan.gc.ca> (ORCID)

See Also

Useful links:

- Report bugs at https://github.com/PredictiveEcology/fireSenseUtils/issues

---

`.objFunIgnition` *Objective function when no piecewise model is used*

Description

Objective function when no piecewise model is used

Usage

```
.objFunIgnition(params, linkinv, nll, sm, nx, mm, mod_env, offset)
```

Arguments

params	DESCRIPTION NEEDED
linkinv	the link function
nll	the log-likelihood function
sm	scaling matrix
nx	number of covariates
mm	model matrix containing data
mod_env	the environment containing params - can be a data.frame
offset	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

.objFunIgnitionPW *Function to pass to the optimizer - Piece-wise version*

Description

Function to pass to the optimizer - Piece-wise version

Usage

```
.objFunIgnitionPW(
  params,
  formula,
  linkinv,
  nll,
  sm,
  updateKnotExpr,
  nx,
  mod_env,
  offset
)
```

Arguments

params	DESCRIPTION NEEDED
formula	DESCRIPTION NEEDED
linkinv	DESCRIPTION NEEDED
nll	DESCRIPTION NEEDED
sm	DESCRIPTION NEEDED
updateKnotExpr	DESCRIPTION NEEDED
nx	DESCRIPTION NEEDED
mod_env	the environment containing params - can be a data.frame
offset	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

`.objfunSpreadFit` *Objective function for* `fireSense_spreadFit` *module*

Description

Objective function for `fireSense_spreadFit` module

Usage

```
.objfunSpreadFit(
  par,
  landscape,
  annualDTx1000,
  nonAnnualDTx1000,
  FS_formula,
  historicalFires,
  fireBufferedListDT,
  covMinMax = NULL,
  maxFireSpread = 0.28,
  minFireSize = 2,
  tests = "snll_fs",
  Nreps = 10,
  mutuallyExclusive = list(youngAge = c("class", "nf")),
  doAssertions = TRUE,
  plot.it = FALSE,
  objFunCoresInternal = 1,
  lanscape1stQuantileThresh = 0.265,
  thresh = 550,
  weighted = TRUE,
  verbose = TRUE
)
```

Arguments

`par`	parameters
`landscape`	A SpatRaster with extent, res, proj used for SpaDES.tools::spread2
`annualDTx1000`	A list of data.table class objects. Each list element is data from a single calendar year, and whose name is "yearxxxx" where xxxx is the 4 number year. The columns in the data.table must integers, that are 1000x their actual values as this function will divide by 1000.

nonAnnualDTx1000
: Like annualDTx1000, but with where each list element will be used for >1 year. The names of the list elements must be "yearxxxx_yearyyyy_yearzzzz" where the xxxx, yyyy, or zzzz represent the calendar years for which that list element should be used. The columns are variables that are used for more than 1 year.

FS_formula
: Formula, put provided as a character string, not class formula. (if it is provided as a class formula, then it invariably will have an enormous amount of data hidden in the formula environment; this is bad for DEoptim)

historicalFires
: DESCRIPTION NEEDED

fireBufferedListDT
: DESCRIPTION NEEDED

covMinMax
: This is a 2 row by multiple column data.frame indicating the minimum and maximum values of the original covariate data values. These will be used to rescale the covariates internally so that they are all between 0 and 1. It is important to not simply rescale internally here because only 1 year is run at a time; all years must be rescaled for a given covariate by the same amount.

maxFireSpread
: A value for spreadProb that is considered impossible to go above. Default 0.28, which is overly generous unless there are many non-flammable pixels (e.g., lakes).

minFireSize
: DESCRIPTION NEEDED

tests
: One or more of "mad", "adTest", "SNLL", or "SNLL_FS". Default: "mad".

Nreps
: Integer. The number of replicates, per ignition, to run.

mutuallyExclusive
: If there are any covariates, e.g,. youngAge, that should be considered mutually exclusive, i.e., "if youngAge is non-zero, should vegPC2 be set to zero", then this can be done here. A named list, where the name of the list element must be a single covariate column name in either annualDTx1000 or nonAnnualDTx1000. The list content should be a "grep" pattern with which to match column names, e.g., "vegPC". The values of all column names that match the grep value will be set to 0, whenever the name of that list element is non-zero. Default is list("youngAge" = list("vegPC")), meaning that all columns with vegPC in their name will be set to zero wherever youngAge is non-zero.

doAssertions
: Logical. If TRUE, the default, the function will test a few minor things for consistency. This should be set to FALSE for operational situations, as the assertions take some small amount of time.

plot.it
: DESCRIPTION NEEDED

objFunCoresInternal
: Internally, this function can use mcmapply to run multiple parallel spread function calls. This should only be >1L if there are spare threads. It is highly likely that there won't be. However, sometimes the DEoptim is particularly inefficient, it starts X cores, and immediately several of them are stopped inside this function because the parameters are so bad, only 2 year are attempted. Then the core will stay idle until all other cores for the DEoptim iteration are complete. Similarly, if only physical cores are used for DEoptim, the additional use of hyperthreaded cores here, internally will speed things up (i.e., this maybe could be 2L or 3L).

`lanscape1stQuantileThresh`
: A `spreadProb` value that represents a threshold for the 1st quantile of the `spreadProbs` on the landscape; if that quantile is above this number, then the `.objFunSpredFit` will bail because it is "too burny" a landscape. Default = `0.265`, meaning if only 25% of the pixels on the landscape are below this `spreadProb`, then it will bail.

`thresh`
: Threshold multiplier used in SNLL fire size (SNLL_FS) test. Default 550. Lowering the threshold value will be more restrictive, but being too restrictive will result in `DEoptim` rejecting more tests and using the "fail value" of 10000. Too high a threshold, and more years will be run and it will take longer to find values.

`weighted`
: Logical. Should empirical likelihood be weighted by log of the actual fire size? This will give large fires more influence on the SNLL.

`verbose`
: DESCRIPTION NEEDED

Value

Attempting a weighted likelihood, https://stats.stackexchange.com/questions/267464/algorithms-for-weighted
With log(fireSize) * likelihood for each fire.

`annualStackToDTx1000` *Convert list of annual SpatRaster to* `data.table`

Description

Convert list of annual SpatRaster to `data.table`

Usage

```
annualStackToDTx1000(x, whNotNA, ...)

## S3 method for class 'SpatRaster'
annualStackToDTx1000(x, whNotNA, ...)

## S3 method for class 'Raster'
annualStackToDTx1000(x, whNotNA, ...)

## S3 method for class 'list'
annualStackToDTx1000(x, whNotNA, ...)
```

Arguments

`x`
: `RasterStack` or list of rasters to convert to `data.table` and multiply by 1000 to save space

`whNotNA`
: Pixel indexes that should go through this process (i.e. not NA)

`...`
: Not currently used

Value

data.table of the SpatRaster or the list

Examples

```
library(raster)
r1 <- raster(extent(0, 10, 0, 10), vals = 1:100)
r2 <- raster(extent(0, 10, 0, 10), vals = 100:1)
r3 <- raster(extent(0, 10, 0, 10), vals = 200:101)
r4 <- raster(extent(0, 10, 0, 10), vals = 300:201)

# list of Rasters
lRast <- list(r1, r2, r3)
lRast[[1]][5] <- NA
whNotNA <- setdiff(1:ncell(r1), 5)

# unnamed -- should error
try(out1 <- annualStackToDTx1000(lRast, whNotNA))

# named
names(lRast) <- c("OneToHun", "HunToOne", "TwoHunToOneHun")
out1 <- annualStackToDTx1000(lRast, whNotNA)

# RasterStack
out2 <- annualStackToDTx1000(raster::stack(lRast), whNotNA)

# List of RasterStacks
s1 <- raster::stack(r1, r2)
names(s1) <- names(lRast)[1:2]
s2 <- raster::stack(r4, r3)
names(s2) <- c(names(lRast)[3], "ThreeHunToTwoHun")
out3 <- annualStackToDTx1000(list(s1 = s1, s2 = s2), whNotNA) ## named list required

# With duplicated names -- to remove duplicates;
#  actually, this doesn't make sense: RasterStack can't have duplicated names
names(lRast) <- c("OneToHun", "OneToHun", "TwoHunToOneHun")
out4 <- annualStackToDTx1000(raster::stack(lRast), whNotNA)
```

---

bufferIgnitionPoints *buffer ignition points to create non-ignitions for model*

Description

buffer ignition points to create non-ignitions for model

Usage

```
bufferIgnitionPoints(ignitionPoints, rtm, bufferSize)
```

Arguments

ignitionPoints	SpatialPolygonsDataFrame with year of ignition
rtm	a template raster
bufferSize	the size of the buffers

Value

a list of data.tables containing indices inside buffered area of each year's ignitions

---

bufferToArea *Create buffers around polygons based on area target for buffer*

Description

Create buffers around polygons based on area target for buffer

Usage

```
bufferToArea(
  poly,
  rasterToMatch,
  areaMultiplier,
  verb = FALSE,
  polyName = NULL,
  field = NULL,
  minSize = 500,
  cores = 1,
  ...
)

## S3 method for class 'list'
bufferToArea(
  poly,
  rasterToMatch,
  areaMultiplier = 10,
  verb = FALSE,
  polyName = NULL,
  field = NULL,
  minSize = 500,
  cores = 1,
  ...
)

## S3 method for class 'SpatialPolygons'
bufferToArea(
  poly,
```

```
  rasterToMatch,
  areaMultiplier = 10,
  verb = FALSE,
  polyName = NULL,
  field = NULL,
  minSize = 500,
  cores = 1,
  ...
)

## S3 method for class 'sf'
bufferToArea(
  poly,
  rasterToMatch,
  areaMultiplier = 10,
  verb = FALSE,
  polyName = NULL,
  field = NULL,
  minSize = 500,
  cores = 1,
  ...
)
```

Arguments

poly	sf polygons or a list of sf containing polygons to buffer.
rasterToMatch	A SpatRaster with res, origin, extent, crs of desired outputted pixelID values.
areaMultiplier	Either a scalar that will buffer areaMultiplier * fireSize or a function of fireSize. Default is 1. See multiplier() for an example.
verb	Logical or numeric related to how much verbosity is printed. FALSE or 0 is none. TRUE or 1 is some. 2 is much more.
polyName	Optional character string of the polygon layer name (not the individual polygons on a sf polygon object)
field	Passed to fasterize::fasterize. If this is unique (such as polygon id), then each polygon will have its buffer calculated independently for each unique value in field
minSize	The absolute minimum size of the buffer & non-buffer together. This will be imposed after areaMultiplier.
cores	number of processor cores to use
...	passed to fasterize::fasterize

Value

A data.table (or list of data.tables if poly was a list) with 2 columns: buffer and pixelID. buffer is either 1 (the original polygon) or 0 (in the buffer).

bufferToAreaRast *create a variable sized buffer around a set of pixels belonging to the same fire ID*

Description

create a variable sized buffer around a set of pixels belonging to the same fire ID

Usage

```
bufferToAreaRast(fireIDraster, areaMultiplier, minSize, flammableRTM, verb = 1)
```

Arguments

fireIDraster	a SpatRaster with values representing distinct fires in a year
areaMultiplier	A scalar that will buffer areaMultiplier * fireSize
minSize	The absolute minimum size of the buffer & non-buffer together. This will be imposed after areaMultiplier.
flammableRTM	@template flammableRTM
verb	Logical or numeric related to how much verbosity is printed. FALSE or 0 is none. TRUE or 1 is some. 2 is much more.

Value

a data.table with fire ID, buffer status, and pixelID

buildCohortBurnHistory *Modify cohortData with burn column*

Description

Modify cohortData with burn column

Usage

```
buildCohortBurnHistory(cohortData, pixelGroupMap, firePolys, year)
```

Arguments

cohortData	either a cohortData object or list of cohortData objects named by year
pixelGroupMap	either a SpatRaster with pixelGroups or list of SpatRasters named by year
firePolys	the output of fireSenseUtils::getFirePolys with YEAR column
year	length-two vector giving temporal period used to subset firePolys. Closed interval

Value

cohortData modified with burn status

`burnClassGenerator` *Generate, Summarize, Predict Burn Classes from Covariates*

Description

Generate, Summarize, Predict Burn Classes from Covariates

Usage

```
burnClassGenerator(df, numClasses = 4:9, AUC = TRUE, plotAUC = FALSE)

burnClassSummary(mod)

burnClassPredict(mod, df)

burnProbFromClass(mod, df)
```

Arguments

`df`	A `data.frame` (or `data.table`), with covariates, including "burned" (a binary 0, 1; not burned = 0, burned = 1), e.g., `timeSinceFire`, `biomassJackPine`, etc. that will be used to find fuel classes. This set of covariates must be available both during fitting and for prediction. These must be quantitative.
`numClasses`	A vector indicating how many classes should be attempted. The function will return the number of classes that best classify the data into homogeneous groups.
`AUC`	Logical. Should the Area Under the receiver operating Curve be returned?
`plotAUC`	Logical. Should the plot of the AUC be made.
`mod`	A model of class `Mclust`, e.g,. coming from `Mclust` or `burnClassGenerator`

Details

This was inspired by reading here: https://www.datanovia.com/en/blog/types-of-clustering-methods-overview-
and here: https://www.datanovia.com/en/lessons/model-based-clustering-essentials/,
with citation here: Scrucca L., Fop M., Murphy T. B. and Raftery A. E. (2016) mclust 5: clustering, classification and density estimation using Gaussian finite mixture models, The R Journal, 8/1, pp. 205-233. https://journal.r-project.org/archive/2016/RJ-2016-021/RJ-2016-021.pdf

Value

A list with 2 elements, first the `model`, which comes from `mclust::Mclust`, and second the Area Under the Curve or AUC as an indicator of the overall goodness of fit.

The algorithm

The basic solution is to take all covariates, including the binary "not burned", "burned" (coded as 0 and 1, respectively), and do model-based clustering with the `mclust` R package. We can choose a fixed number of burn classes, or a finite range (see `numClasses` argument. This will make `numClasses` "homogeneous" groups, including whether they burned or not. From this, we can identify groups by looking at the mean values of "burned" to see what their burn tendency is as a "homogeneous" group.

Categorical data

For now, it is recommended to convert categorical data to dummy variables, 0 and 1. E.g., For land cover, wetland class can be converted to a column "wetland" with 1 for data points that are wetlands and 0 for non-wetland.

How much data to include

This has not been tested yet; however, I believe that having a relatively similar number of "burned" and "unburned" pixels (within 3x either way), is probably a good idea. In other words, if there are 100,000 burned data points, there should be between 30,000 and 300,000 unburned data points. If there are already buffers around the burned polygons that include unburned pixels, then these buffers can be used as part of the unburned content.

Author(s)

Eliot McIntire

Examples

```
## Not run:
#################################
# Use own data; here is a generated set for reprex
library("data.table")
N <- 1e5
DT <- list()
for (i in c("train", "test")) {
  DT[[i]] <- data.table(burned = sample(c(0, 0, 0, 1), replace = TRUE, size = N))
  set(DT[[i]], NULL, "jp", rlnorm(N, mean = 4 + 0.5 * DT[[i]]$burned, sd = 0.25))
  set(DT[[i]], NULL, "bs", rlnorm(N, mean = 4 + 0.3 * DT[[i]]$burned, sd = 0.25))
  set(DT[[i]], NULL, "ws", rlnorm(N, mean = 4 + 0.2 * DT[[i]]$burned, sd = 0.25))
  set(DT[[i]], NULL, "age", rlnorm(N, mean = 4 - 0.2 * DT[[i]]$burned, sd = 0.25))
  DT[[i]][, c("jp", "bs", "ws", "age") := lapply(.SD, function(x) x / max(x) * 1000),
    .SDcols = c("jp", "bs", "ws", "age")
  ]
  DT[[i]][, c("age") := lapply(.SD, function(x) x / max(x) * 200), .SDcols = c("age")]
  summary(DT[[i]])
  boxplot(DT[[i]]$age ~ DT[[i]]$burned)
}

bc <- burnClassGenerator(DT[["train"]], 4:8)
# Show if the model is good at predicting burn state
(bc$AUC) # area under the curve
```

```
# print summary of mean values of each burn class
(summ <- burnClassSummary(bc$model))

# predict -- add Burn Class to object
set(DT[["test"]], NULL, "burnClass", burnClassPredict(bc$model, df = DT[["test"]]))
prob <- burnProbFromClass(bc$model, DT[["test"]])

## End(Not run)
```

`calcYoungAge` *Iteratively calculate* `youngAge` *column in FS covariates*

Description

Iteratively calculate youngAge column in FS covariates

Usage

```
calcYoungAge(
  years,
  annualCovariates,
  standAgeMap,
  fireBufferedListDT,
  cutoffForYoungAge = 15
)
```

Arguments

`years`	the years over which to iterate
`annualCovariates`	list of data.table objects with pixelID
`standAgeMap`	template `SpatRaster`
`fireBufferedListDT`	data.table containing non-annual burn and buffer pixelIDs
`cutoffForYoungAge`	Numeric. Default is 15. This is the age below which the pixel is considered "young" –> `youngAge` column will be 1 if `age <= 15`

Value

a raster layer with unified standAge and time-since-disturbance values

castCohortData *preparing covariates for fitting modules*

Description

preparing covariates for fitting modules

Usage

```
castCohortData(
  cohortData,
  pixelGroupMap,
  lcc,
  ageMap = NULL,
  missingLCC,
  year = NULL,
  cutoffForYoungAge = 15
)
```

Arguments

cohortData	A data.table with columns: pixelGroup, ecoregionGroup, speciesCode, and optionally age, B, mortality, aNPPAct, and sumB.
pixelGroupMap	A RasterLayer with pixel values equal to a pixel group number that corresponds exactly to pixelGroup column in cohortData.
lcc	data.table of dummified landcover
ageMap	a stand age map to assign ages to non-forest LCC used during predict
missingLCC	LCC class to assign forested pixels absent from cohortData must be a character matching a nonForestedLCC group, e.g. 'nonForest_highFlam'
year	numeric representing the year represented by cohortData
cutoffForYoungAge	Numeric. Default is 15. This is the age below which the pixel is considered "young" –> youngAge column will be 1 if age <= 15

Value

a trimmed cohortData with wide-layout and rows for every pixel in lcc

---

`chk_duplicatedStartPixels` *Data checks and assertions for* `spreadFitRun`

Description

Data checks and assertions for `spreadFitRun`

Usage

```
chk_duplicatedStartPixels(cells, size)

.doDataChecks(moduleName, envir, attribs, fml)
```

Arguments

`cells`	DESCRIPTION NEEDED
`size`	DESCRIPTION NEEDED
`moduleName`	DESCRIPTION NEEDED
`envir`	DESCRIPTION NEEDED
`attribs`	DESCRIPTION NEEDED
`fml`	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

DESCRIPTION NEEDED

`cleanUpSpreadFirePoints` *Ensure fire points are located on flammable pixels inside a fire polygon Intended to be run using* `Map`

Description

Ensure fire points are located on flammable pixels inside a fire polygon Intended to be run using `Map`

Usage

```
cleanUpSpreadFirePoints(firePoints, bufferDT, flammableRTM)
```

Arguments

firePoints	a sf points object representing annual ignitions
bufferDT	a data.table of burned cells, output from bufferToArea
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data

Value

a list of harmonized points and polygons

---

climateRasterToDataTable

Converts stacks of climate rasters to data.table and optionally subsets to index

---

Description

Converts stacks of climate rasters to data.table and optionally subsets to index

Usage

```
climateRasterToDataTable(historicalClimateRasters, Index = NULL)
```

Arguments

historicalClimateRasters	named list of SpatRaster objects
Index	optional list of data.table objects named by fireYear and containing fire buffer indices

Value

a long-layout data.table of climate values in each pixel and year

cohortsToFuelClasses *Classify* pixelGroups *by flammability*

Description

Classify pixelGroups by flammability

Usage

```
cohortsToFuelClasses(
  cohortData,
  pixelGroupMap,
  flammableRTM,
  landcoverDT = NULL,
  sppEquiv,
  sppEquivCol,
  cutoffForYoungAge,
  fuelClassCol = "FuelClass"
)
```

Arguments

cohortData	A data.table with columns: pixelGroup, ecoregionGroup, speciesCode, and optionally age, B, mortality, aNPPAct, and sumB.
pixelGroupMap	A RasterLayer with pixel values equal to a pixel group number that corresponds exactly to pixelGroup column in cohortData.
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
landcoverDT	Optional table of nonforest landcovers and pixel indices. It will override pixel values in cohortData, if supplied.
sppEquiv	table with species name equivalencies between the kNN and final naming formats. See data("sppEquivalencies_CA", "LandR"). Only necessary if mixedType == 2. If not provided and mixedType == 2, will attempt to use data("sppEquivalencies_CA", "LandR").
sppEquivCol	the column name to use from sppEquiv.
cutoffForYoungAge	age at and below which pixels are considered 'young'
fuelClassCol	the column in sppEquiv that describes unique fuel classes

Value

a SpatRaster of biomass by fuel class as determined by fuelClassCol and cohortData

compareMDC *Download and prepare fire data from National Fire Database*

Description

Download and prepare fire data from National Fire Database

Usage

```
compareMDC(
  historicalMDC,
  projectedMDC,
  flammableRTM = NULL,
  Ylimits = c(80, 220),
  firstHistoricalYear = 2001,
  firstProjectedYear = 2011
)
```

Arguments

historicalMDC	raster stack of historical MDC
projectedMDC	raster stack of projected MDC
flammableRTM	an optional raster of flammable pixels to subset data
Ylimits	the upper and lower MDC range for the plot
firstHistoricalYear	the earliest year of historical data
firstProjectedYear	the earliest year of projected data

Value

a ggplot object

Examples

```
## Not run:
compareMDC(
  historicalMDC = simOutPreamble$historicalClimateRasters$MDC,
  projectedMDC = simOutPreamble$projectedClimateRasters$MDC,
  flammableRTM = fSsimDataPrep$flammableRTM
)

## End(Not run)
```

dtReplaceNAwith0 *Replace* NA*s in a* data.table *with zeros*

Description

Replace NAs in a data.table with zeros

Usage

```
dtReplaceNAwith0(DT, colsToUse = NULL)
```

Arguments

DT	DESCRIPTION NEEDED
colsToUse	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

extractSpecial *Extract the elements of the special terms, i.e. the variable and the knot value*

Description

Extract the elements of the special terms, i.e. the variable and the knot value

Usage

```
extractSpecial(v, k)
```

Arguments

v	DESCRIPTION NEEDED
k	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

`getFirePoints_NFDB` *Get Fire* `SpatialPoints` *from Canadian Fire Database*

Description

Get Fire `SpatialPoints` from Canadian Fire Database

Usage

```
getFirePoints_NFDB(
  url = NULL,
  studyArea = NULL,
  rasterToMatch = NULL,
  redownloadIn = 1,
  years = 1991:2017,
  fireSizeColName = "SIZE_HA",
  NFDB_pointPath
)
```

Arguments

`url`	Passed to `prepInputs`
`studyArea`	A `SpatialPolygons*` object used as the principle study region, passed to `prepInputs`.
`rasterToMatch`	A `RasterLayer` objects to use as the template for all subsequent raster operations (i.e., the one used throughout the simulation).
`redownloadIn`	Numeric Time in YEARS that we tolerate the data to be "old" i.e. 0.5 would mean "redownload data older than 6 months"
`years`	Numeric vector of consecutive years to fetch.
`fireSizeColName`	Character describing the name of the column containing fire size information.
`NFDB_pointPath`	Passed to `destinationPath` in `prepInputs`

Value

A `sf` spatial points object.

getFirePoints_NFDB_V2 *Get Fire* SpatialPoints *from Canadian Fire Database*

Description

Get Fire SpatialPoints from Canadian Fire Database

Usage

```
getFirePoints_NFDB_V2(
  url = NULL,
  studyArea = NULL,
  redownloadIn = 1,
  years = 1991:2017,
  fireSizeColName = "SIZE_HA",
  NFDB_pointPath = NULL,
  plot = FALSE
)
```

Arguments

url	Passed to prepInputs
studyArea	A SpatialPolygons* object used as the principle study region, passed to prepInputs.
redownloadIn	Numeric Time in YEARS that we tolerate the data to be "old" i.e. 0.5 would mean "re-download data older than 6 months"
years	Numeric vector of consecutive years to fetch.
fireSizeColName	Character describing the name of the column containing fire size information.
NFDB_pointPath	Passed to destinationPath in prepInputs
plot	logical indicating whether to produce plot of fire points. Default FALSE.

Value

A sf spatial points object.

getFirePolygons *Download and prepare fire data from National Fire Database*

Description

Download and prepare fire data from National Fire Database

Usage

```
getFirePolygons(years, useInnerCache = FALSE, ...)
```

Arguments

years	years to filter fire polygons by
useInnerCache	logical indicating whether to cache the prepInputs call
...	additional arguments passed to prepInputs()

Value

list of fire polygons by year

---

harmonizeBufferAndPoints

Cleaning up the polygon points

Description

Mostly this is about 2 things:

1. remove fires that were so small that they take less than 1 pixel so they are not in the buff object but are in the cent object.
2. the centroid cell is in a buffer or otherwise nonburnable cell (e.g., water). For 1) remove these from the centroid data. For 2) this function will search in the neighbourhood for the next closest pixel that has at least 7 available neighbours that can burn. If not, remove these.

Usage

```
harmonizeBufferAndPoints(cent, buff, ras, idCol = "FIRE_ID")
```

Arguments

cent	List of points as SpatialPointsDataFrame
buff	List of data.table objects with 3 columns, "buffer" which is 1 (in the fire) or 0 (in a buffer), ids which are the fire ids which MUST match the ids in the cent.
ras	The raster that created the pixelIDs in the buff.
idCol	The column name as a character string with the fire ids. Defaults to "FIRE_ID".

harmonizeFireData *Outer wrapper on spread fire polygon data munging that does several things:*

1. *ensure buffered fires are entirely in studyArea*
2. *ensure every fire has a corresponding ignition point, and vice versa*
3. *ensure these points are flammable*

Description

Outer wrapper on spread fire polygon data munging that does several things:

1. ensure buffered fires are entirely in studyArea
2. ensure every fire has a corresponding ignition point, and vice versa
3. ensure these points are flammable

Usage

```
harmonizeFireData(
  firePolys,
  flammableRTM,
  spreadFirePoints,
  areaMultiplier,
  minSize,
  pointsIDcolumn = "FIRE_ID"
)
```

Arguments

`firePolys`	the semi-processed fire polys, with field matching pointsIDcolumn
`flammableRTM`	a `rasterToMatch` with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
`spreadFirePoints`	the ignition points corresponding to `firePolys`
`areaMultiplier`	Either a scalar that will buffer `areaMultiplier * fireSize` or a function of `fireSize`. See `?fireSenseUtils::bufferToArea`.
`minSize`	an alternative to areaMultiplier, typically used when fires are small
`pointsIDcolumn`	the name of the column denoting fire ids in both spreadFirePoints and firePolys

logistic4p *Four- and five-parameter logistic functions*

Description

Four- and five-parameter logistic functions

Usage

```
logistic4p(x, par)

logistic5p(x, par)

logistic3p(x, par, par1 = 0.1)

logistic2p(x, par, par1 = 0.1, par4 = 0.5)
```

Arguments

x	DESCRIPTION NEEDED
par	DESCRIPTION NEEDED
par1	DESCRIPTION NEEDED
par4	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

makeFireIDs *identify each year's individual fires and buffer them accordingly*

Description

identify each year's individual fires and buffer them accordingly

Usage

```
makeFireIDs(
  year,
  fireRaster,
  flammableRTM,
  bufferForFireRaster,
  areaMultiplier,
  minSize = 5000,
  verb = 1
)
```

Arguments

year	numeric fire year
fireRaster	a SpatRaster with values representing fire years
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
bufferForFireRaster	buffer size used to group discrete patches of burned pixels as belonging to the same fire
areaMultiplier	A scalar that will buffer areaMultiplier * fireSize
minSize	The absolute minimum size of the buffer & non-buffer together. This will be imposed after areaMultiplier.
verb	Logical or numeric related to how much verbosity is printed. FALSE or 0 is none. TRUE or 1 is some. 2 is much more.

Value

a data.table with fire ID, buffer status, and pixelID

---

makeLandcoverDT *Create landcoverDT object to classify and track non-forest lcc*

Description

Create landcoverDT object to classify and track non-forest lcc

Usage

```
makeLandcoverDT(rstLCC, flammableRTM, forestedLCC, nonForestedLCCGroups)
```

Arguments

rstLCC	landcover raster
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
forestedLCC	vector of values representing forested landcover classes in rstLCC
nonForestedLCCGroups	a named list of non-forested flammable landcover groups

Value

a data.table with columns for pixelID and binary presence of landcover

makeLociList *Convert a list of* SpatialPointsDataFrame *object to a list of* data.table *objects*

Description

Must supply a raster so that points can be converted to the cells on a raster. It is assumed that the sizeCol is accurate. If not, it should be recalculated before this function call.

Usage

```
makeLociList(
  ras,
  pts,
  idsCol = "FIRE_ID",
  dateCol = "YEAR",
  sizeCol = "POLY_HA",
  sizeColUnits = "ha"
)
```

Arguments

ras	A raster that will be the template for cells (pixel ids)
pts	A list of sf point objects
idsCol	Character string identifying column name in pts that has unique id per event (i.e., fire)
dateCol	Character string identifying column name in pts that has year
sizeCol	Character string identifying column name in pts that has size of individual event. Can be in hectares or metres squared. Should set sizeColUnits
sizeColUnits	Character string. Either "ha" or "m2".

Value

A list of data.table objects, each with 4 columns, "size" (in pixels), "date", "ids" from idsCol, and "cells", which are the pixel indices of the pts points.

makeMutuallyExclusive *guarantees mutually exclusive values in a data table*

Description

guarantees mutually exclusive values in a data table

Usage

```
makeMutuallyExclusive(dt, mutuallyExclusiveCols = list(youngAge = c("vegPC")))
```

Arguments

dt	a data.table with columns that should be mutually exclusive
mutuallyExclusiveCols	A named list, where the name of the list element must be a single covariate column name in dt. The list content should be a "grep" pattern with which to match column names, e.g., "vegPC". The values of all column names that match the grep value will be set to 0, whenever the name of that list element is non-zero. Default is list("youngAge" = list("vegPC")), meaning that all columns with vegPC in their name will be set to zero wherever youngAge is non-zero.

Value

a data.table with relevant columns made mutually exclusive

makeRastersFromCD *Put* cohortData *back into a* SpatRaster *with some extra details*

Description

Put cohortData back into a SpatRaster with some extra details

Usage

```
makeRastersFromCD(class, cohortData, flammableRTM, pixelGroupMap)
```

Arguments

class	fuelClass from sppEquiv
cohortData	A data.table with columns: pixelGroup, ecoregionGroup, speciesCode, and optionally age, B, mortality, aNPPAct, and sumB.
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
pixelGroupMap	A RasterLayer with pixel values equal to a pixel group number that corresponds exactly to pixelGroup column in cohortData.

Value

a SpatRaster with values equal to `class` biomass (B)

`makeTSD` *preparing a time since disturbance map from stand age and fire data*

Description

preparing a time since disturbance map from stand age and fire data

Usage

```
makeTSD(
  year,
  firePolys = NULL,
  fireRaster = NULL,
  standAgeMap,
  lcc,
  cutoffForYoungAge = 15
)
```

Arguments

`year`	the year represented by `standAge`
`firePolys`	list of `spatialPolygon` objects comprising annual fires. fireRaster will supersede firePolys if provided
`fireRaster`	a `RasterLayer` with values representing fire years
`standAgeMap`	initial stand age map
`lcc`	`data.table` with landcover values - `landcoverDT`
`cutoffForYoungAge`	Numeric. Default is 15. This is the age below which the pixel is considered "young" –> `youngAge` column will be 1 if `age <= 15`

Value

a `SpatRaster` with values representing time since disturbance

multiplier *multiplier*

Description

DESCRIPTION NEEDED

Usage

```
multiplier(size, minSize = 1000, baseMultiplier = 5)
```

Arguments

size	DESCRIPTION NEEDED
minSize	DESCRIPTION NEEDED
baseMultiplier	DESCRIPTION NEEDED

objNlminb objNlminb

Description

Wrapper around `stats::nlminb`

Usage

```
objNlminb(x, objective, lower, upper, control, hvPW, ...)
```

Arguments

x	DESCRIPTION NEEDED
objective	objective function
lower	lower bounds on coefficients
upper	upper bounds on coefficients
control	DESCRIPTION NEEDED
hvPW	logical indicating whether the formula is piece-wise #IE added
...	additional arguments passed to objective function

Value

DESCRIPTION NEEDED

oom *Order of Magnitude*

Description

Order of Magnitude

Usage

```
oom(x)
```

Arguments

x	a numeric

Value

the order of magnitude

plotBurnSummary *Plot burn summary*

Description

Create plot with subplots showing: a) area burned; b) number of fires; c) mean fire size.

Usage

```
plotBurnSummary(studyAreaName, climateScenario, outputDir, Nreps)
```

Arguments

studyAreaName	character string giving the study area name
climateScenario	character string specifying the name of a CIMP6 climate scenario, including SSP, formatted as in ClimateNA, using underscores as separator (e.g., 'CanESM5_SSP370').
outputDir	Path specifying the directory to which outputs figures/objects should be saved.
Nreps	the number of simulation replicates/run used to produce summary figures. NOTE: mclapply is used internally, so you should set options(mc.cores = nReps) to take advantage of parallel processing.

Value

list of file names corresponding to the figures and/or objects written to disk

plotCumulativeBurns *Plot cumulative burn maps*

Description

Plot cumulative burn maps

Usage

```
plotCumulativeBurns(
  studyAreaName,
  climateScenario,
  outputDir,
  Nreps,
  rasterToMatch
)
```

Arguments

studyAreaName	character string giving the study area name
climateScenario	character string specifying the name of a CIMP6 climate scenario, including SSP, formatted as in ClimateNA, using underscores as separator (e.g., 'CanESM5_SSP370').
outputDir	Path specifying the directory to which outputs figures/objects should be saved.
Nreps	the number of simulation replicates/run used to produce summary figures. NOTE: mclapply is used internally, so you should set options(mc.cores = nReps) to take advantage of parallel processing.
rasterToMatch	A RasterLayer objects to use as the template for all subsequent raster operations (i.e., the one used throughout the simulation).

Value

list of file names corresponding to the figures and/or objects written to disk

a file path corresponding to the images and/or objects written to disk

plotHistoricFires *Plot historic ignitions, escapes, and area burned*

Description

Plot historic ignitions, escapes, and area burned

Usage

```
plotHistoricFires(
  climateScenario,
  studyAreaName,
  outputDir,
  firePolys,
  ignitionPoints
)
```

Arguments

climateScenario	character string specifying the name of a CIMP6 climate scenario, including SSP, formatted as in ClimateNA, using underscores as separator (e.g., 'CanESM5_SSP370').
studyAreaName	character string giving the study area name
outputDir	Path specifying the directory to which outputs figures/objects should be saved.
firePolys	A sf spatial polygons of historic fire burn areas, from the Canadian National Fire Database.
ignitionPoints	A sf spatial points of historic fire ignitions, from the Canadian National Fire Database.

Value

list of file names corresponding to the figures and/or objects written to disk

---

predictIgnition *Predictions from ignition model*

Description

Predictions from ignition model

Usage

```
predictIgnition(
  model,
  data,
  coefs,
  rescaleFactor,
  lambdaRescaleFactor,
  linkinv
)
```

Arguments

model	formula of fitted model (sim$fireSense_IgnitionFitted[["formula"]][-2])
data	data for prediction
coefs	model coefficients (sim$fireSense_IgnitionFitted$coef)
rescaleFactor	spatial rescaling factor when predicted and fitted data are at different scales. Calculaed as: (predResolution/fitResolution)^2
lambdaRescaleFactor	If the data for fitting has been sampled for pseudo-absences, this imposes a new baseline probability of fire occurrences, hence predictions need to be adjusted. If the original fire prob. is (total no. fires)/(total no. fires + total no. absences), and the fire probability imposed by sampling is (total no. fires)/(total no. fires + no. sampled pseudo-absences), to adjust predicted values, one needs to multiply them by (total no. fires + no. sampled pseudo-absences/(total no. fires + total no. absences)
linkinv	family link function (sim$fireSense_IgnitionFitted$family$linkinv)

Value

vector of predicted values.

---

pw *Handling piecewise terms in a formula*

Description

Handling piecewise terms in a formula

Usage

```
pw(variable, knot)
```

Arguments

variable	DESCRIPTION NEEDED
knot	DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

rasterFireBufferDT *this is a wrapper to simplify caching of* lapply *with* bufferForFireRaster. *Years are iteratively processed by* makeFireID.

Description

this is a wrapper to simplify caching of lapply with bufferForFireRaster. Years are iteratively processed by makeFireID.

Usage

```
rasterFireBufferDT(
  years,
  fireRaster,
  flammableRTM,
  bufferForFireRaster,
  areaMultiplier,
  minSize = 5000,
  verb = 1,
  cores = 1
)
```

Arguments

years	numeric fire years
fireRaster	a SpatRaster with values representing fire years
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data
bufferForFireRaster	buffer size used to group discrete patches of burned pixels as belonging to the same fire
areaMultiplier	A scalar that will buffer areaMultiplier * fireSize
minSize	The absolute minimum size of the buffer & non-buffer together. This will be imposed after areaMultiplier.
verb	Logical or numeric related to how much verbosity is printed. FALSE or 0 is none. TRUE or 1 is some. 2 is much more.
cores	number of processor cores to use

Value

a list of data.tables named by year, with cols ids, buffer, and pixelID

rasterFireSpreadPoints *create a list of annual ignition points based on fire raster*

Description

create a list of annual ignition points based on fire raster

Usage

```
rasterFireSpreadPoints(fireBufferDT, flammableRTM)
```

Arguments

fireBufferDT	a data.table with columns buffer (1 = burned), id (unique fire ID), and pixelID
flammableRTM	@template flammableRTM

Value

a list of sf point objects

rbetaBetween *Generate random beta variates between 2 values and a mean*

Description

Generate random beta variates between 2 values and a mean

Usage

```
rbetaBetween(n, l, u, m, shape1, shape2 = NULL)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
l	scalar numeric for the lower bound
u	scalar numeric for the upper bound
m	scalar numeric for the mean
shape1	non-negative parameter of the Beta distribution
shape2	If provided, passed to rbeta. If not, m must be (i.e., the mean)

See Also

stats::rbeta

removeBufferedFiresOutsideRTM *remove buffered fires in fireBufferedListDT that are outside RTM*

Description

remove buffered fires in fireBufferedListDT that are outside RTM

Usage

```
removeBufferedFiresOutsideRTM(fireBufferedDT, flammableRTM)
```

Arguments

fireBufferedDT	data.table containing indices for buffered annual fires
flammableRTM	a rasterToMatch with binary values where 1 represents flammable pixels, 0 nonflammable, and NA no data

Value

fireBufferedDT excluding fires with indices (burned or unburned) outside flammableRTM

rescaleKnown2 *rescale function no.2*

Description

rescale function no.2

Usage

```
rescaleKnown2(x, minNew, maxNew, minOrig, maxOrig)
```

Arguments

x	a vector to be rescaled
minNew	the minimum of the new range
maxNew	the max of the new range
minOrig	the minimum of the original data
maxOrig	the maximum of the original data

Value

the rescaled vector

`runDEoptim` *Wrapper around* `DEoptim` *call*

Description

Does the multiple cluster connections. This will only work if ssh keys are correctly made between machines (if using multiple machines).

Usage

```
runDEoptim(
  landscape,
  annualDTx1000,
  nonAnnualDTx1000,
  fireBufferedListDT,
  historicalFires,
  itermax,
  initialpop = NULL,
  NP = NULL,
  trace,
  strategy,
  cores = NULL,
  libPath = .libPaths()[1],
  logPath = tempfile(sprintf("fireSense_SpreadFit_%s_", format(Sys.time(),
    "%Y-%m-%d_%H%M%S")), fileext = ".log"),
  doObjFunAssertions = getOption("fireSenseUtils.assertions", TRUE),
  cachePath,
  iterStep = 25,
  lower,
  upper,
  mutuallyExclusive,
  FS_formula,
  objFunCoresInternal,
  covMinMax = covMinMax,
  tests = c("SNLL", "adTest"),
  maxFireSpread,
  Nreps,
  thresh = 550,
  .verbose,
  visualizeDEoptim,
  .plotSize = list(height = 1600, width = 2000)
)

DEoptimIterative(
  itermax,
  lower,
  upper,
```

```
  control,
  FS_formula,
  covMinMax,
  tests = c("SNLL", "adTest"),
  objFunCoresInternal,
  maxFireSpread,
  Nreps,
  visualizeDEoptim,
  cachePath,
  mutuallyExclusive,
  doObjFunAssertions = getOption("fireSenseUtils.assertions", TRUE),
  iterStep = 25,
  thresh = 550,
  .verbose,
  .plotSize = list(height = 1600, width = 2000)
)
```

Arguments

landscape	A RasterLayer which has the correct metadata associated with the pixelID and cells of other objects in this function call
annualDTx1000	A list of data.table objects. Each list element will be from 1 year, and it must be the same length as fireBufferedListDT and historicalFires. All covariates must be integers, and must be 1000x their actual values.
nonAnnualDTx1000	A list of data.table objects. Each list element must be named with a concatenated sequence of names from names(annualDTx1000), e.g., 1991_1992_1993. It should contain all the years in names(annualDTx1000). All covariates must be integers, and must be 1000x their actual values.
fireBufferedListDT	A list of data.table objects. It must be same length as annualDTx1000, with same names. Each element is a data.table with columns: buff...TODO: INCOMPLETE
historicalFires	DESCRIPTION NEEDED
itermax	Passed to DEoptim.control
initialpop	DESCRIPTION NEEDED
NP	DESCRIPTION NEEDED
trace	Passed to DEoptim.control
strategy	Passed to DEoptim.control
cores	A numeric (for running on localhost only) or a character vector of machine names (including possibly "localhost"), where the length of the vector indicates how many cores should be used on that machine.
libPath	A character string indicating an R package library directory. This location must exist on each machine, though the function will make sure it does internally.

logPath — A character string indicating what file to write logs to. This dirname(logPath) must exist on each machine, though the function will make sure it does internally.

doObjFunAssertions — logical indicating whether to do assertions.

cachePath — The cachePath to store cache in. Should likely be cachePath(sim)

iterStep — Integer. Must be less than itermax. This will cause DEoptim to run the itermax iterations in ceiling(itermax / iterStep) steps. At the end of each step, this function will plot, optionally, the parameter histograms (if visualizeDEoptim is TRUE)

lower — Passed to DEoptim

upper — Passed to DEoptim

mutuallyExclusive — If there are any covariates, e.g,. youngAge, that should be considered mutually exclusive, i.e., "if youngAge is non-zero, should vegPC2 be set to zero", then this can be done here. A named list, where the name of the list element must be a single covariate column name in either annualDTx1000 or nonAnnualDTx1000. The list content should be a "grep" pattern with which to match column names, e.g., "vegPC". The values of all column names that match the grep value will be set to 0, whenever the name of that list element is non-zero. Default is list("youngAge" = list("vegPC")), meaning that all columns with vegPC in their name will be set to zero wherever youngAge is non-zero.

FS_formula — Passed to DEoptim

objFunCoresInternal — DESCRIPTION NEEDED

covMinMax — Passed to fireSenseUtils::.objfunSpreadFit

tests — Passed to fireSenseUtils::.objfunSpreadFit

maxFireSpread — Passed to fireSenseUtils::.objfunSpreadFit

Nreps — Passed to fireSenseUtils::.objfunSpreadFit

thresh — Threshold multiplier used in SNLL fire size (SNLL_FS) test. Default 550.

.verbose — Passed to fireSenseUtils::.objfunSpreadFit

visualizeDEoptim — Logical. If TRUE, then histograms will be made of DEoptim outputs.

.plotSize — List specifying plot height and width, in pixels.

control — DESCRIPTION NEEDED

Value

DESCRIPTION NEEDED

`stackAndExtract` *prepare covariate table with ignition year, fuel class, climate value, and land cover*

Description

prepare covariate table with ignition year, fuel class, climate value, and land cover

Usage

```
stackAndExtract(years, fuel, LCC, climate, fires)
```

Arguments

`years`	character vector of fire years with FS notation e.g. year2002
`fuel`	raster brick of aggregated fuel classes
`LCC`	raster brick of aggregated LCC classes
`climate`	list of raster layers named by climate variable with raster layer names matching `years`
`fires`	list of spatial points representing annual ignitions

Value

a data.frame with cell numbers, ignitions, and covariates for each year

`updateStackYearNames` *Update name of layers in a climate raster stack*

Description

Update name of layers in a climate raster stack

Usage

```
updateStackYearNames(annualDataStack, desiredYears)
```

Arguments

`annualDataStack`	`RasterStack`
`desiredYears`	character

visualizeDE *Make histograms of* DEoptim *object* pars

Description

Make histograms of DEoptim object pars

Usage

```
visualizeDE(DE, cachePath)
```

Arguments

DE	An object from a DEoptim call
cachePath	A cacheRepo to pass to showCache and loadFromCache if DE is missing.

Index