Package ‘TreeMig’

March 23, 2026

Type Package

Title Wrapper for TreeMig

Version 0.3.1

Author Daniel Scherrer

Maintainer The package maintainer `<yourself@somewhere.net>`

Description This package allowes the user to use the TreeMig model (implemented in Fortran) by providing functionalities for data preprocessing, simulation setup, and post-processing the output.
Additionally, it includes a user-
friendly graphical user interface (GUI), enhancing the accessibility and ease of use of the package.

License What license is it under?

Encoding UTF-8

LazyData true

Depends R (>= 4.3.2)

Imports raster (>= 3.6.26),
sf (>= 1.0.14),
shiny (>= 1.8.0),
ggplot2 (>= 3.4.4),
gridExtra (>= 2.3),
methods (>= 4.3.2),
rlang (>= 1.1.2),
ncdf4 (>= 1.22),
data.table (>= 1.14.8),
crayon (>= 1.5.2),
dplyr (>= 1.1.4),
tibble (>= 3.2.1),
tidyr (>= 1.3.0),
gganimate (>= 1.0.8),
latticeExtra (>= 0.6.30),
rasterVis (>= 0.51.6),
slider (>= 0.3.1),
gifski (>= 1.12.0.2),
terra (>= 1.7.55),
shinyFiles (>= 0.9.3),
callr (>= 3.7.3),
pals (>= 1.8)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

Contents

`applyStockability` *Removes non-stockable cells from the bioclimate given a stockability data.frame*

Description

Removes non-stockable cells from the bioclimate given a stockability data.frame

Usage

```
applyStockability(bioclimate, stockability, years)
```

Arguments

`bioclimate`	Any bioclimate data.frame with columns lon and lat
`stockability`	A stockability data.frame with columns lon, lat and stock. stock values of 0 will be removed from bioclimate.
`years`	One or more years that will be affected.

Value

bioclimate data.frame as.data.frame

`calculateBioclimate` *Calculate Bioclimate Data*

Description

This function calculates bioclimate data based on input data for climate, slope aspect, and bucket size. All the data should originate from the same study area and maintain the same order of cells.

Usage

```
calculateBioclimate(studyArea, climData, slopeAspectData, bucketSizeData)
```

Arguments

`studyArea`	A StudyArea object representing the study area.
`climData`	A data frame containing climate data.
`slopeAspectData`	A data frame containing slope aspect.
`bucketSizeData`	A data frame containing bucket size data.

Value

A data frame containing bioclimate data including longitude, latitude, year, Degree Days (DD), Winter Temperature (WiT), and Drought Stress (DrStr).

`calculateSlasp` *Calculate Slope-Aspect (Slasp) Data*

Description

This function calculates the Slasp values based on input slope and aspect data. The slasp value is calculated as follows: `slasp=2*cos(aspect*pi/180)*min(1,slope/60)`

Usage

```
calculateSlasp(slopeData, aspectData)
```

Arguments

`slopeData`	data.frame or data.table. Should have "lon", "lat" and "slope" columns with the slope data in degrees.
`aspectData`	data.frame or data.table. Should have "lon", "lat" and "aspect" column with the aspect data in degrees.

Value

A `data.table` with columns "aspect" and "slasp" columns appended to the input slopeData.

`CheckPrepareSpeciesSummaryData` *Loading and checking of Outputvariables*

Description

Function to load data from the `outputVariable` created in a speciefic TreeMig simulation (`simulationID`) for a subset of `years`, `species` and `sub.area`. The data gets checked if all the desired output is available and a `output.threshold` gets applied.

Usage

```
CheckPrepareSpeciesSummaryData(
  simulationID,
  species,
  output.variable,
  output.threshold,
  sub.area = NULL,
  years
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`species`	The species that should be extracted. This must be a subset of the species simulated `species.names`.
`output.threshold`	A threshold of the output variable to be applied to restrict the species pool.
`sub.area`	Either a dataframe with lon/lat Coordinates or a shapefile with polygon(s). This will remove all data points not selected in case only a df is returned. No effect on returning an array.
`years`	The years that should be extracted. This must be a subset of the years that were written out in the TreeMig simulation.
`outputVariable`	The variable to be extracted from the output data of TreeMig simulation.

Value

Either a dataframe or a multi-dimensional array containing the information.

`combineClimateData` *Combine Temperature and Precipitation Data*

Description

This function combines temperature and precipitation data. The function takes two `data.tables` as input, one containing temperature data (`tempData`) and the other containing precipitation data (`precData`). Both `data.tables` must have columns "lon", "lat", "year", and columns for temperature and precipitation values from 1 to 12. The function returns a new `data.table` with temperature and precipitation values merged together, denoted as "temp_1" to "temp_12" and "prec_1" to "prec_12".

Usage

```
combineClimateData(tempData, precData)
```

Arguments

`tempData`	data.table. Temperature data with columns "lon", "lat", "year", and "temp_1" to "temp_12".
`precData`	data.table. Precipitation data with columns "lon", "lat", "year", and "prec_1" to "prec_12".

Value

A new `data.table` with merged temperature and precipitation data.

`CoupleOtherModelToTreeMig` *CoupleOtherModelToTreeMig: Couple an External Model to a TreeMig Simulation*

Description

Provides a generic coupling framework that runs **TreeMig (TM)** together with a user-defined **other model (OM)**. The routine prepares TM, iterates through user-specified coupling time points, runs TM and OM in each sub-simulation period (SSP), glues their results together, and writes the final output files in a format compatible with the native TreeMig post-processing tools.

Usage

```
CoupleOtherModelToTreeMig(
  couplingTimePoints_v,
  realTimeStart = NULL,
  realTimeEnd = NULL,
  TMDirectory = NULL,
  TM_ctr,
  TM_OutputVars,
  TM_statefilenamepathorig = statefilenamepathorig,
  otherMod,
  Glue_OM_Results_Together = MakeOMList(),
  OM_var_data_init = NULL,
  OM_fixed_data = NULL
)
```

Arguments

`couplingTimePoints_v`	Integer vector of time points ti when the coupling between TM and OM occurs.
`realTimeStart`	REAL; optional; start time of the entire simulation in real values, for transformation of input data (e.g. 2000.5)
`TM_OutputVars`	optional; vector of strings, one or several of c("Biomass","LAI","Numbers","Seeds","BA"). can be read in with ReadTMOutput (simulationID,outputVar)

`TM_statefilenamepathorig`	string; name and path of the TreeMig initial state file param TM_simulationID;string
`otherMod`	string; name of the function containing the other model. must be provided by the user according to the definitions in `otherMod`.
`Glue_OM_Results_Together`	string;optional; name of a function to collect the results of the other model in the iterations. can be provided by the user; if not provided, the OM_var_data lists will be put into a list.
`OM_var_data_init`	list, userdefined; optional; contains initial values for variables in other model, that can change in iteration, e.g., state, output (, input)
`OM_fixed_data`	list, userdefined; optional; contains fixed information for other model, e.g., control settings, parameters, input
`TM_ctr;`	structure, control settings of TreeMig

Details

Preparation – Saves the current TM control settings and state file, creates the working directory, and loads the initial TM state. All data that the OM needs (fixed model and control parameters, variable inputs, initial state) are passed via the lists 'OM_fixed_data' and 'OM_var_data_init'. OM_var_data_init contains the variable data for OM at the beginning of the simulation.
Iteration through sub simulation periods (SSPs) (from ti to tip1) within the simulation. In each SSP,
... **TM** is run from its current state. The results are glued together (in TM_result_df_list_wholeIteration), the statefile is read into a dataframe.
... **OM** is run (by otherMod) and optionally influenced by the state, output and/or bioclimate of TreeMig at the beginning or at the end of the SSP (ti or tip1). The results are glued together (in OM_var_data_wholeIteration). The modified TM-statefile and TM-bioclimate are stored in the InitialValues file and the bioclimate file.
Results of TM and OM over the entire simulation period are stored in the lists TM_result_df_list_wholeIteration and OM_var_data_wholeIteration.
After the iteration, to be compatible with the TM framework plotting routines, the TreeMig results over the entire iteration are stored in a text file, and a control file over the entire simulation period is constructed.

Value

a list containing

`TM_result_df_list_wholeIteration`	list of dataframes, for the overall simulation TreeMig outputs for the different ou
`OM_results_wholeIteration`	list, containing the results of OM for the overall simulation

See Also

Called by: `SomeHigherLevelFunction` # replace with actual callers, if any
Calls: `ReadStateFile2Dataframe` `TreeMig::ctr$setOption` `TreeMig::ctr$saveConfig` `TreeMig::runTreeMig` `otherMod` `Glue_TM_Results_Together` `Glue_OM_Results_Together` `MakeOMList` `TailorTMStateFile2TMInputFil` `WriteDataframe2StateFile` `WriteTMOutputVars` `mpr`

createAndSetAddCtrParams *createAndSetAddCtrParams*

Description

Creates an additional TreeMig control parameter adds it with a given value to the control object ctr.

Usage

```
createAndSetAddCtrParams(ctr, parname, comment, left, type, value)
```

Arguments

`ctr`	structure; control settings of TreeMig
`parname`	string; name of the new control parameter. Must match exactly one of the control parameter names set by the function Init_additional_params in All_par.f90.- Currently c("descriptor","maxspc","maxhc","maxlc","totalMaxHeight","toplt","birht","ndigits") "descriptor" means a line describing the subsequent lines will be written into the control file
`comment`	string; comment up to 60 characters
`left`	logical; does the value have to be left aligned? Usually FALSE
`type`	string; type of the control parameter, one of c("logical","integer","numeric","string","NA")
`value`	type matching type; value of the control parameter of the type type

Details

Must be called after ctr <- newConfig(TMDirectory), normally after the last ctr$setOption In the control file, the corresponding line will be either #### comment #### if parname is "descriptor". or "value" ! "comment" | "parname"

Value

ctr, structure; control settings of TreeMig now including also the new control parameter

createStudyArea *Utility function to create a StudyArea object.*

Description

This function acts as a simple wrapper to the StudyArea class constructor.

Usage

```
createStudyArea(extent, gridCellSize, projection)
```

Arguments

`extent`	A numeric vector of length 4 specifying the extent of the study area.
`gridCellSize`	A numeric value representing the size of each grid cell.
`projection`	A character string specifying the projection (WKT/CRS string).

Value

An instance of the StudyArea class.

`createTMEnvironment` *Creates a directory with all the necessary files to run/prepare TreeMig*

Description

Creates a directory with all the necessary files to run/prepare TreeMig

Usage

```
createTMEnvironment(path, name = "TreeMig")
```

Arguments

`path`	character. Directory where TreeMig should be located
`name`	character. Name of TreeMig folder

`extractData` *Helper Function to Extract Data from Raster or Text File*

Description

This function extracts data from either text files or raster formats. It allows for various preprocessing steps including spatial filtering, interpolation and aggregation.

Usage

```
extractData(
  data,
  studyArea,
  type = "raster",
  years = NA,
  crs = NA,
  interpolate = FALSE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`data`	Raster or text data. Can be a filepath or an object in R.
`studyArea`	A study area for extraction.
`type`	Character string indicating type of input data: "text" or "raster".
`years`	Optional vector to select only the specified years.
`crs`	The Coordinate Reference System (CRS) of the raster if not specified in the input data itself.
`interpolate`	logical. If TRUE, bilinear interpolation will be used during extraction; if FALSE, nearest neighbor method will be used.
`aggregate`	logical. If TRUE, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

Processed data extracted for the studyArea.

Note

If the data's CRS is not provided and can't be inferred from the data itself, the function assumes it's the same as the study area's CRS.

`extractRaster` *Extract values from a raster for a given study area.*

Description

This function extracts values from a given raster (or raster file path) based on a provided study area.

Usage

```
extractRaster(
  raster,
  studyArea,
  xy = FALSE,
  rasterCRS = NA,
  timeFilter = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`raster`	SpatRaster or character. The file path of a raster or a SpatRaster for data extraction.
`studyArea`	studyArea. A study area for extraction.
`xy`	logical. If TRUE, the resulting data table will have 'lon' and 'lat' columns.
`rasterCRS`	The Coordinate Reference System (CRS) of the raster if not specified in the raster itself.
`timeFilter`	character. String to filter the time dimension of the raster.
`interpolate`	logical. If TRUE, bilinear interpolation will be used during extraction; if FALSE, nearest neighbor method will be used.
`aggregate`	logical. If TRUE, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A data.table containing extracted values. If xy=TRUE, the table will also have 'lon' and 'lat' columns.

Note

If the raster's CRS is not provided and can't be inferred from the raster itself, the function assumes it's the same as the study area's CRS.

`generateAspectData` *Generate Aspect Data for a Study Area*

Description

Generates a `data.table` with an "aspect" column for a given study area. Each cell in the study area is assigned the same aspect value defined in `aspectValue`.

Usage

```
generateAspectData(aspectValue, studyArea)
```

Arguments

`aspectValue`	numeric. The aspect value for the study area's cells.
`studyArea`	StudyArea. The study area for which the aspect data will be generated.

Value

A `data.table` with columns "lon", "lat", and "aspect".

`generateBucketSizeData` *Generate BucketSize Data for a Study Area*

Description

Generates a `data.table` with a "bucketSize" column for a given study area. Each cell in the study area is assigned the same bucketSize value defined in `bucketSizeValue`.

Usage

```
generateBucketSizeData(bucketSizeValue, studyArea)
```

Arguments

`bucketSizeValue`	numeric. The bucketSize value for the study area's cells.
`studyArea`	StudyArea. The study area for which the bucketSize data will be generated.

Value

A `data.table` with columns "lon", "lat", and "bucketSize".

`generatePrecData` *Generate Precipitation Data for a Study Area*

Description

Generates a dataset containing precipitation values for a given study area. Each cell in the study area is assigned the same precipitation value. For each subsequent year, the precipitation value is incremented by ‘yearlyIncrement‘.

Usage

```
generatePrecData(precValue, studyArea, years, yearlyIncrement = 0)
```

Arguments

`precValue`	numeric. The initial precipitation value for the study area's cells.
`studyArea`	StudyArea. The study area for which the precipitation data will be generated.
`years`	numeric vector. The range of years for which the data will be generated.
`yearlyIncrement`	numeric. The yearly increment to apply to the ‘precValue‘.

Value

A data.table with columns "lon", "lat", "year", "prec_1",...,"prec_12" containing the generated yearly precipitation values for the specified study area.

`generateSlopeData` *Generate Slope Data for a Study Area*

Description

Generates a `data.table` with a "slope" column for a given study area. Each cell in the study area is assigned the same slope value defined in `slopeValue`.

Usage

```
generateSlopeData(slopeValue, studyArea)
```

Arguments

`slopeValue`	numeric. The slope value for the study area's cells.
`studyArea`	StudyArea. The study area for which the slope data will be generated.

Value

A `data.table` with columns "lon", "lat", and "slope".

`generateStockData` *Generate Stockability Data for a Study Area*

Description

Each cell in the study area is assigned to the "stock" value of 1.

Usage

```
generateStockData(studyArea, years)
```

Arguments

`studyArea`	StudyArea. The study area for which the stock data will be generated.
`years`	numeric vector. The range of years for which the data will be generated.

Value

A data.table with columns "lon", "lat" and "stock", with stock initialized to 1.

`generateStudyData` *Generate study data for a given study area.*

Description

This function creates a data table containing study points with an associated value for a specified variable.

Usage

```
generateStudyData(studyArea, name, value)
```

Arguments

`studyArea`	studyArea. A study area.
`name`	character. The name of the variable (column).
`value`	numeric. The value for each study point.

Value

A data.table with columns "lon", "lat", and an additional column specified by the name parameter.

`generateTempData` *Generate Temperature Data for a Study Area*

Description

Generates a dataset containing temperature values for a given study area. Each cell in the study area is assigned the same temperature value. For each subsequent year, the temperature value is incremented by ‘yearlyIncrement‘.

Usage

```
generateTempData(tempValue, studyArea, years, yearlyIncrement = 0)
```

Arguments

`tempValue`	numeric. The initial temperature value for the study area’s cells.
`studyArea`	StudyArea. The study area for which the temperature data will be generated.
`years`	numeric vector. The range of years for which the data will be generated.
`yearlyIncrement`	numeric. The yearly increment to apply to the ‘tempValue‘.

Value

A data.table with columns "lon", "lat", "year", "temp_1",...,"temp_12" containing the generated yearly temperature values for the specified study area.

`generateYearlyStudyData` *Generate yearly study data for a given study area.*

Description

This function creates a data table containing study points for each year specified, with the associated column names and values for each of those years.

Usage

```
generateYearlyStudyData(studyArea, names, values, years, yearlyIncrements)
```

Arguments

`studyArea`	A study area
`names`	character vector. Names for the variables (columns).
`values`	numeric vector. Starting values corresponding to each name.
`years`	numeric vector. The years for which the study data should be generated.
`yearlyIncrements`	numeric vector. Yearly increments for each value.

Value

A data.table with columns "lon", "lat", "year", and additional columns specified by the names parameter.

`getAspectData` *Extract and Process Aspect Data for a Study Area*

Description

This function extracts aspect data for a specified study area and optionally processes it. The function supports two primary input types for extraction: raster and data.table. The `aspect` parameter can be either the actual data object or a file path pointing to the data.

Usage

```
getAspectData(
  aspect,
  studyArea,
  type = "raster",
  crs = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`aspect`	data source for aspect values. Either a path or actual the data object containing to aspect data
`studyArea`	studyArea. The study area for which the aspect data will be extracted.
`type`	character. Type of the aspect data source, either "raster" or "text"
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If `TRUE`, bilinear interpolation will be used during extraction; if `FALSE`, nearest neighbor method will be used.
`aggregate`	logical. If `TRUE`, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A `data.table` with columns "lon", "lat" and "aspect" containing the extracted aspect values for the specified study area.

`getAvailableOutputVariables`

Creates a vector of the names of all the available output variables based on an input ctr file.

Description

Creates a vector of the names of all the available output variables based on an input ctr file.

Usage

```
getAvailableOutputVariables(simulationID, TMDirectory = getwd())
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`TMDirectory`	optional path to the TreeMig directory. If not set getwd() is used.

Value

A vector of the names of allt eh available output variables.

`getBucketSizeData` *Extract and Process BucketSize Data for a Study Area*

Description

This function extracts bucketSize data for a specified study area and optionally processes it. The function supports two primary input types for extraction: raster and data.table. The `bucketSize` parameter can be either the actual data object or a file path pointing to the data.

Usage

```
getBucketSizeData(
  bucketSize,
  studyArea,
  type = "raster",
  crs = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`bucketSize`	data source for bucketSize values. Either a path or actual the data object containing to bucketSize data
`studyArea`	studyArea. The study area for which the bucketSize data will be extracted.
`type`	character. Type of the bucketSize data source, either "raster" or "text"
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If `TRUE`, bilinear interpolation will be used during extraction; if `FALSE`, nearest neighbor method will be used.
`aggregate`	logical. If `TRUE`, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A `data.table` with columns "lon", "lat" and "bucketsize" containing the extracted bucketSize values for the specified study area.

`getColors` *Simple functions to return some colors for the plotting*

Description

Simple functions to return some colors for the plotting

Usage

```
getColors(color.option)
```

Arguments

`color.option`	A choice of predefined color palette: `"natural"`,`"highcontrast"`,`"amazing"` or `"blue"` or a vector of custom colors.

Value

Return a vector of hex colors assosiated with the choosen color palette.

`getLegendBreaks` *Simple functions to return the breaks for the color scale*

Description

Simple functions to return the breaks for the color scale

Usage

```
getLegendBreaks(maximum)
```

Arguments

`maximum`	The maximum value of data to display in the figure.

Value

Return the number and values of the breaks used to construct a good looking legend.

`getOutputYears` *Checking the year for which output is available in a simulation.*

Description

This function uses the `simulationID` to check which years created ouput for the simulations. This data is used to make sure that no years can be selected for plotting that were not part of the simulation.

Usage

```
getOutputYears(simulationID, TMDirectory = getwd())
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`TMDirectory`	optional path to the TreeMig directory. If not set getwd() is used.

Value

A vector with the years that have output written the TreeMig simulation with the `simulationID`.

`getPrecData` *Extract and Process Precipitation Data for a Study Area*

Description

This function extracts precipitation data for a specified study area and optionally processes it. The function supports two primary input types for extraction: raster and data.table. The ‘prec‘ parameter can be either the actual data object or a file path pointing to the data. If the provided precipitation data is in millimeters, it will be converted to centimeters.

Usage

```
getPrecData(
  prec,
  studyArea,
  years = NA,
  type = "raster",
  unit = NA,
  crs = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`prec`	data source for precipitation. Either a path or actual the data object containing to precipitation data
`studyArea`	studyArea. The study area for which the precipitation data will be extracted.
`years`	numeric vector. The range of years for which the data will be extracted.
`type`	character. Type of the precipitation data source, either "raster" or "text"
`unit`	charcter. Unit of the precipitation data ("cm" or "mm"). If not provided for raster data, it tries to infer it from the data.
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If TRUE, bilinear interpolation will be used during extraction; if FALSE, nearest neighbor method will be used.
`aggregate`	logical. If TRUE, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A data.table with columns "lon", "lat", "year", "prec_1",...,"prec_12" containing the extracted yearly precipitation values for the specified study area.

`getSlopeData` *Extract and Process Slope Data for a Study Area*

Description

This function extracts slope data for a specified study area and optionally processes it. The function supports two primary input types for extraction: raster and data.table. The `slope` parameter can be either the actual data object or a file path pointing to the data.

Usage

```
getSlopeData(
  slope,
  studyArea,
  type = "raster",
  crs = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`slope`	data source for slope values. Either a path or actual the data object containing to slope data
`studyArea`	studyArea. The study area for which the slope data will be extracted.
`type`	character. Type of the slope data source, either "raster" or "text"
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If `TRUE`, bilinear interpolation will be used during extraction; if `FALSE`, nearest neighbor method will be used.
`aggregate`	logical. If `TRUE`, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A `data.table` with columns "lon", "lat" and "slope" containing the extracted slope values for the specified study area.

`getSpeciesLabs` *Simple functions to return some colors for the plotting*

Description

Simple functions to return some colors for the plotting

Usage

```
getSpeciesLabs(species)
```

Arguments

`species`	A vector of species names as used in the TreeMig model

Value

Return the species names in a format better fitted for labelling.

getStockData *Extract and Process Stock Data for a Study Area*

Description

This function extracts stock data for a specified study area and optionally processes it. Stock is set to 1 if the extracted value matches with the provided `filterValue` and `filter`. The function supports two primary input types for extraction: raster and data.table. The ‘temp‘ parameter can be either the actual data object or a file path pointing to the data.

Usage

```
getStockData(
  stock,
  studyArea,
  field,
  filter,
  filterValue,
  type = "raster",
  crs = NA,
  interpolate = FALSE,
  aggregate = TRUE,
  aggregateFun = "modal"
)
```

Arguments

`stock`	data source for stock. Either a path or actual the data object containing to stock data
`studyArea`	studyArea. The study area for which the stock data will be extracted.
`field`	character. Name of the field that is used for comparison with `filterValue`
`filter`	character. One of "in" (%in%), "notin", "eq" (==), "gt" (>), "ge" (>=), "lt" (<), "leq" (<=), "noteq" (!=)
`filterValue`	Any value, is passed to the filter
`type`	character. Type of the stock data source, either "noas04", "raster", "shape" or "text"
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If TRUE, bilinear interpolation will be used during extraction; if FALSE, nearest neighbor method will be used.
`aggregate`	logical. If TRUE, the function will aggregate raster cells if the raster’s resolution is finer than the study area’s.
`aggregateFun`	character. Function used to aggregate values. (Ignored when type is "noas04"). Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A data.table with columns "lon", "lat" and "stock" containing the filtered yearly stock values for the specified study area.

`getSwissSoilBucketSize` *Calculate Bucket Size from Swiss Soil Suitability Map*

Description

This function calculates the bucket size based on data from the Swiss Soil Suitability Map for a given study area.

Usage

```
getSwissSoilBucketSize(pathToSwissSoil, studyArea)
```

Arguments

`pathToSwissSoil`	Character string specifying the path to the Swiss Soil Suitability Map file.
`studyArea`	An object of class ‘StudyArea‘ representing the study area for data extraction.

Value

A ‘data.table‘ with the columns "lon", "lat" and "bucketsize"

`getTempData` *Extract and Process Temperature Data for a Study Area*

Description

This function extracts temperature data for a specified study area and optionally processes it. The function supports two primary input types for extraction: raster and data.table. The ‘temp‘ parameter can be either the actual data object or a file path pointing to the data.

Usage

```
getTempData(
  temp,
  studyArea,
  years,
  type = "raster",
  crs = NA,
  interpolate = TRUE,
  aggregate = TRUE,
  aggregateFun = "mean"
)
```

Arguments

`temp`	data source for temperature. Either a path or actual the data object containing to temperature data
`studyArea`	studyArea. The study area for which the temperature data will be extracted.
`years`	numeric vector. The range of years for which the data will be extracted.
`type`	character. Type of the temperature data source, either "raster" or "text"
`crs`	Coordinate reference system for the data.
`interpolate`	logical. If TRUE, bilinear interpolation will be used during extraction; if FALSE, nearest neighbor method will be used.
`aggregate`	logical. If TRUE, the function will aggregate raster cells if the raster's resolution is finer than the study area's.
`aggregateFun`	character. function used to aggregate values. Either an actual function, or for the following, their name: "mean", "max", "min", "median", "sum", "modal", "any", "all", "prod", "which.min", "which.max", "sd" (sample standard deviation) and "std" (population standard deviation)

Value

A data.table with columns "lon", "lat", "year", "temp_1",...,"temp_12" containing the extracted yearly temperature values for the specified study area.

`GetTM_OutputAndTransformCoords`

GetTM_OutputAndTransformCoords

Description

Reads the TreeMig output and transforms to the real coordinate system. Years are still in 0,1,...

Usage

```
GetTM_OutputAndTransformCoords(outputvar, ctr = TM_ctr)
```

Arguments

`outputvar`	string; which TM output (e.g. biomass, LAI, BA,...) should be read in
`ctr`	structure; current simulation control settings

Details

The latest TreeMig output for the wished variable (outputvar) is read in. The coordinates are changed from 1,2,... to the real coordinates

Value

outp dataframe; of the TM output with additionally the transformed coordinates on lat and lon. The read in coordinates are now ilat and ilon.

`GetTM_OutputAndTransformCoordsAndYears` *GetTM_OutputAndTransformCoordsAndYears*

Description

Reads the TreeMig output and transforms to the real coordinate system and years to the real years.

Usage

```
GetTM_OutputAndTransformCoordsAndYears(outputvar, ctr, realStartYear)
```

Arguments

`outputvar`	string; which TM output (e.g. biomass, LAI, BA,...) should be read in
`ctr`	structure; current simulation control settings
`realStartYear`	Numeric; current start year of the simulation

Details

The latest TreeMig output for the wished variable (outputvar) is read in. The coordinates are changed from 1,2,... to the real coordinates The years are changed from 0,1,2,... to the real years

Value

outp dataframe; of the TM output with additionally the transformed coordinates on lat and lon, and with the transformed years. The read in coordinates are now ilat and ilon. Also the years are changed to the real years.

`Glue_OM_Results_Together` *Glue_OM_Results_Together*

Description

Placeholder for userdefined function. glues the results of the other model of the different SSPs together

Usage

```
Glue_OM_Results_Together(
  ti = ti,
  OM_var_data_wholeIteration = OM_var_data_wholeIteration,
  OM_var_data = OM_var_data,
  otherStuff = list()
)
```

Arguments

`ti`	INT; first time point of the SSP (0,1,2,...)
`OM_var_data`	; all results of the other model in SSP
`otherStuff`	list; placeholder for potential other information to be stored and glued together
`numYrsPerIterStep;INT;`	nr of timesteps in this SSP
`sim_start;REAL;`	start of simulation in real years
`OM_var_data_wholeIteration;`	contains all results of the other model up to ti

Details

#' userdefined.

Value

OM_var_data_wholeIteration; contains all results of the other model up to tip1

`Glue_TM_Results_Together` *Glue_TM_Results_Together*

Description

glues the results of the TreeMig of the current SSP to those of the previous SSP

Usage

```
Glue_TM_Results_Together(
  TM_simulationID,
  ti,
  numYrsPerIterStep,
  sim_start,
  TM_result_df_list_wholeIteration,
  TM_OutputVars = "Biomass"
)
```

Arguments

`ti`	INT; first time point of the SSP (0,1,2,...)
`TM_OutputVars`	optional; vector of strings, one or several of c("Biomass","LAI","Numbers","Seeds","BA").
`TM_simulationID;string;`	to find the current TreeMig output in ReadTMOutput (simulationID,outputVar)
`numYrsPerIterStep;INT;`	nr of timesteps in this SSP
`sim_start;REAL;`	start of simulation in real years
`TM_result_df_list_wholeIteration;list`	of dataframes, each df contains the simulation results for the SSPs up to ti

Details

Reads the output file(s) for the chosen output variables, calculates the actual year, and binds the dataframes together

Value

TM_result_df_list_wholeIteration; list of dataframes, each df contains the simulation results for the SSPs up to tip1 for one output variable

`loadConfigAddLines` *loadConfigAddLines*

Description

Reads in additional lines other than the standard ones from the control file, if they exist.

Usage

```
loadConfigAddLines(controlFileName, TMDirectory, ctr)
```

Arguments

`TMDirectory`	string; path to the current TM environment
`ctr`	structure; standard control settings of TreeMig

Details

Must be called after ctr$loadConfig(TMDirectory) In the control file, the additional lines have to be either #### comment #### or value ! comment | parname , where parname must match exactly one of the control parameter names set by the function Init_additional_params in All_par.f90. Currently c("descriptor","maxspc","maxhc","maxlc","totalMaxHeight","toplt","birht","ndigits")

Value

ctr, structure; control settings of TreeMig. including also the additional control parameters in the control file if there are any exist.

`MakeOMDataFrame` *MakeOMDataFrame*

Description

appends the results of the other model at tip1 to the already existing dataframe

Usage

```
MakeOMDataFrame(
  ti = ti,
  OM_var_data_wholeIteration = OM_var_data_wholeIteration,
  OM_var_data = OM_var_data,
  otherStuff = list()
)
```

Arguments

`ti`	INT; first time point of the SSP (0,1,2,...)
`OM_var_data_wholeIteration`	dataframe; contains all results of the other model up to ti
`OM_var_data`	dataframe ; all results of the other model in SSP
`otherStuff`	list; placeholder for potential other information to be stored and glued together

Details

#' userdefined.

Value

OM_var_data_wholeIteration; contains all results of the other model up to tip1

`MakeOMList` *MakeOMList*

Description

appends the results of the other model (list) at tip1 to the already existing list

Usage

```
MakeOMList(
  ti = ti,
  OM_var_data_wholeIteration = OM_var_data_wholeIteration,
  OM_var_data = OM_var_data,
  otherStuff = list()
)
```

Arguments

`ti`	INT; first time point of the SSP (0,1,2,...)
`OM_var_data`	; all results of the other model in SSP
`otherStuff`	list; placeholder for potential other information to be stored and glued together
`OM_var_data_wholeIteration;`	contains all results of the other model up to ti

Details

#' userdefined.

Value

OM_var_data_wholeIteration; contains all results of the other model up to tip1

newConfig *Creates a new empty TreeMig config (FFT support)*

Description

Creates a new empty TreeMig config (FFT support)

Usage

```
newConfig(TMDirectory)
```

Arguments

TMDirectory	character. Path to TreeMig location

Value

TreeMigConfig.

otherMod *otherMod: Model coupled to TreeMig*

Description

To be defined by user. Simulates the dynamics of another model coupled to TreeMig in an iteration step based on the current state of TreeMig and modifies the TreeMig state according the simulation result at the end of the iteration step

Usage

```
otherMod(
  ti = NULL,
  tip1 = NULL,
  tip2 = NULL,
  TM_state_ti = NULL,
  TM_state_tip1 = NULL,
  TM_input_tip1 = NULL,
  realTimeStart = NULL,
  TM_ctr = NULL,
  TM_OutputVars = NULL,
  OM_var_data = NULL,
  OM_fixed_data = NULL
)
```

Arguments

`ti`	INT; first time point of the iteration interval (0,1,2,...)
`tip1`	INT; last time point of the iteration interval and first time point of the next one (0,1,2,...)
`TM_state_ti`	dataframe; TreeMig state at ti;
`TM_state_tip1`	dataframe; TreeMig state at tip1; columns species,(REAL)year,(INT 1,2,3... coordinates) lat,lon,(REAL nr of seeds in) seedbank,(REAL nr of) newSeeds, (REAL nr of) antagonists, (INT nr of height classes) nhc , REAL nr of trees in height classes 0 to 15
`TM_input_tip1`	dataframe, format as TM_input_ti, optional; TreeMig input (bioclim) at tip1 before running the other model columns dataframe;INT: lon, lat (real coords), year (real year); REAL: DD (degree day sum), WiT (min. winter temp.), DrStr (drought stress), and optionally ...
`realTimeStart`	REAL; optional; start time of the entire simulation in real values, for transformation of input data (e.g. 2000.5)
`TM_ctr`	structure; contains the control settings of the simulation of the given coupling period
`TM_OutputVars`	optional; vector of strings, one or several of c("Biomass","LAI","Numbers","Seeds","BA"). can be read in with ReadTMOutput (simulationID,outputVar)
`OM_var_data`	list, userdefined; optional; contains information for other model changing in iteration, e.g., state, output, input
`OM_fixed_data`	list, userdefined; optional; contains fixed information for other model, e.g., control settings, parameters, input

Details

Different options for the model type:

1. continuous: between ti and tip1, e.g. ODE or time discrete with smaller time steps in between
2. time discrete: from ti to tip1
3. discrete event: only at tip1

The model must be either space-independent or defined on the exactly same grid as TreeMig and run in the same integer timesteps as TreeMig, starting with 0.
It must use as coupling variables either the TreeMig state at ti or tip1 and/or the TreeMig output, which has to be read with the function ReadTMOutput.
It must return a changed TreeMig state at tip1, optionally a changed TreeMig input (bioclimfile) If not meeting to these requirements

Value

a list containing

`TM_state_modified`	dataframe, format as TM_state_ti; state of TreeMig affected by the other model at tip1
`TM_input_modified`	dataframe, format as TM_input_tip1; optional; bioclim, affected by the other model at tip1, fo
`OM_var_data_modified`	list, content as OM_var_data; optional; contains changed state or input of other model

outputToNCDF *Converts TreeMig output to NCDF4 output*

Description

Converts TreeMig output to NCDF4 output

Usage

```
outputToNCDF(simulationID, delete = FALSE, compression = 4)
```

Arguments

simulationID	The simulationID belonging to the desired control file created by TreeMig.
delete	Flag to indicate if the text files should be deleted or not.
compression	Amount of compression to be used for the netcdf.

Value

Writes a single netCDF file to the disk

plotBioClimateMaps *Creates a map of the choosen bioclimatic variables and years.*

Description

Creates a map of the choosen bioclimatic variables and years.

Usage

```
plotBioClimateMaps(
  simulationID,
  years = "all_years",
  bioclim.var = "all",
  single.figure = FALSE,
  color.option = "highcontrast",
  animated = FALSE,
  filename = NA,
 ggplot.args = list(device = "png", scale = 3, width = NA, height = NA, units = "in",
    dpi = 300, duration = 15, limitsize = TRUE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`years`	A vector of the years to be selected (e.g., `c(1900:2000)`) or `"all_years"`. This must be a subset of the years for which the bioclimate was calculated.
`bioclim.var`	The bioclimatic variables which should be plotted across time. Accepted input are combinations of `"DD"`, `"WiT"`, `"DrStr"`, or `"all"`).
`single.figure`	A flag indicating if the maps of all chosen bioclimatic variables should be drawn in a single plot.
`color.option`	A choice of predefined color palette: `"natural"`,`"highcontrast"`,`"amazing"` or `"blue"` or a vector of custom colors.
`filename`	A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.
`ggplot.args`	A list of settings to be transfered to ggplot to adjust plot size and layout.
`amimated`	Should the figure be an amimated time series rather than one figure per selected year?

Value

All the figures are directly written to the disk. Nothing is returned within R.

---

plotBioClimateSummary *Creates a figure of the choosen bioclimatic parameters for the selected years.*

Description

Creates a figure of the choosen bioclimatic parameters for the selected years.

Usage

```
plotBioClimateSummary(
  simulationID,
  years = "all_years",
  bioclim.var = "all",
  sub.area = NULL,
  averaging.intervall = 10,
  quant.boundries = NULL,
  filename = NA,
 ggplot.args = list(device = "png", scale = 1, width = NA, height = NA, units = "in",
    dpi = 300, limitsize = TRUE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`years`	A vector of the years to be selected (e.g., `c(1900:2000)`) or `"all_years"`. This must be a subset of the years for which the bioclimate was calculated.

`bioclim.var`	The bioclimatic variables which should be plotted across time. Accepted input are combinations of `"DD"`, `"WiT"`, `"DrStr"`, or `"all"`).
`sub.area`	Either a dataframe with lon/lat Coordinates or a `"SpatialPolygonsDataFrame"` with polygon(s). In case there are several polygons one figure per polygon will be drawn.
`averaging.intervall`	The time window used for the moving average.
`quant.boundries`	The lower and upper boundry for the shaded area in quantiles. e.g. `c(0.025,0.975)`
`filename`	A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.
`ggplot.args`	A list of settings to be transfered to ggplot to adjust plot size and layout.

Value

All the figures are directly written to the disk. Nothing is returned within R.

`plotSpeciesAlongGradients` *Creating figures of variables along envrionmental or geographic gradients*

Description

Creating figures of variables along envrionmental or geographic gradients

Usage

```
plotSpeciesAlongGradients(
  simulationID,
  species = "all_species",
  years = "all_years",
  output.variable = "all_variables",
  plot.type = "stacked_area",
  gradient = "latitude",
  custom.raster = NULL,
  custom.name = "custom.name",
  bins = 25,
  se = FALSE,
  grid.arrange = FALSE,
  output.threshold = 0,
  sub.area = NULL,
  animated = FALSE,
  filename = NA,
 ggplot.args = list(device = "png", scale = 1, width = 3, height = 3, units = "in", dpi
   = 300, duration = 15, limitsize = FALSE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`species`	A vector with the names of the desired species (e.g., `c("Abies_alba", "Picea_abies")`) or `"all_species"`. This must be a subset of the species included in the selected simulationID.
`years`	A vector of the years to be selected (e.g., `c(1900,2000)`) or `"all_years"`. This must be a subset of the years that were exported from the TreeMig simulation.
`output.variable`	The output variable for which the data should be plotted. Must be a selection of output variables created by TreeMig (`"SeedAntagonists"`, `"BasalArea"`, `"Biomass"`, `"Ingrowth"`, `"LAI"`, `"NPP"`, `"StemNumber"`, `"Pollen"`, `"Seeds"`, `"HeightClass_0"`, `"HeightStruct_01"`, `"HeightStruct_02"`, `"HeightStruct_03"`, `"HeightStruct_04"`, `"HeightStruct_05"`, `"HeightStruct_06"`, `"HeightStruct_07"` , `"HeightStruct_08"` , `"HeightStruct_09"`, `"HeightStruct_10"`, `"HeightStruct_11"`, `"HeightStruct_12"`, `"HeightStruct_13"`, `"HeightStruct_14"`, `"HeightStruct_15"` or `"all_variables"`. The function will plot one Figure per variable unless the flag. grid.arrange is set to TRUE
`plot.type`	Type of the plot, can be either line or stacked_area
`gradient`	Must either be `"latitude"`, `"longitude"` or `"custom"`
`custom.raster`	A raster layer from which the information can be sampled. This is only used if gradient is `"custom"`.
`custom.name`	A string with the name of the variable provided in the raster layer. This is mainly used for plot labels.
`bins`	Number of bins used to construct the figures.
`se`	Flag indicating if the 95% standard error around the lines should be shown.
`grid.arrange`	A flag to determine if a big grid with all selected output variables should be created.
`output.threshold`	A threshold excluding species which never reach a certain biomass from the species comparison figure.
`sub.area`	A `"SpatialPolygonsDataFrame"` with polygon(s). In case there are several polygons one figure per polygon will be drawn.
`filename`	A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.
`ggplot.args`	A list of arguments to be given to the ggsave function. This allows to adjust the width, height, scaling, dpi and output format. (e.g., `list(device = "png",scale = 1, width = 8, height = 8, units = "in", dpi = 300, limitsize = TRUE)`)
`amimated`	Should the figure be an amimated time series rather than one figure per selected year?

Value

All the figures are directly written to the disk. Nothing is returned within R.

plotSpeciesLimits — *Creating figures of treelines along envrionmental gradients*

Description

Creating figures of treelines along envrionmental gradients

Usage

```
plotSpeciesLimits(
  simulationID,
  species = "all_species",
  years = "all_years",
  th.type = "Biomass",
  threshold.lines = "leading",
  gradient = "latitude",
  custom.raster = NULL,
  custom.name = "Elevation",
  bins = 100,
  averaging.intervall = 10,
  sub.area = NULL,
  filename = NA,
 ggplot.args = list(device = "png", scale = 1, width = 10, height = 5, units = "in", dpi
    = 300, duration = 15, limitsize = TRUE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`species`	A vector with the names of the desired species (e.g., `c("Abies_alba", "Picea_abies")`) or `"all_species"`. This must be a subset of the species included in the selected simulationID.
`years`	A vector of the years to be selected (e.g., `c(1900,2000)`) or `"all_years"`. This must be a subset of the years that were exported from the TreeMig simulation.
`th.type`	Three methods are available to determine the species-specific treeline: `"Biomass"` This threshold is based on the decline of Biomass with increasing elevation and selects the treeline at 5% of the maximum observed biomass per species. `"TreeHeight"` This threshold is based on the highest elevation where we still observe trees with a height > 3m. `"Presence"` This threshold is based simply on the presence of the species in therefore also includes seedlings. This rather reflects the highest elevation a species can currently establish rather than the traditonal treeline.
`threshold.lines`	Which end of the distribution of the species along the gradient should be marked? This can either be `"leading"`, `"trailing"`.
`gradient`	Must either be `"latitude"`, `"longitude"` or `"custom"`
`custom.raster`	A raster layer from which the information can be sampled. This is only used if gradient is `"custom"`.

`custom.name`	A string with the name of the variable provided in the raster layer. This is mainly used for plot labels.
`bins`	Number of bins used to construct the figures.
`averaging.intervall`	The time window used for the moving average.
`sub.area`	A `"SpatialPolygonsDataFrame"` with polygon(s). In case there are several polygons one figure per polygon will be drawn.
`filename`	A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.
`ggplot.args`	A list of arguments to be given to the ggsave function. This allows to adjust the width, height, scaling, dpi and output format. (e.g., `list(device = "png",scale = 1, width = 8, height = 8, units = "in", dpi = 300, limitsize = TRUE))`

Value

All the figures are directly written to the disk. Nothing is returned within R.

---

plotSpeciesMaps *Creating maps of the spatial and temporal distributiion of the output variables of species as well as their comparison.*

Description

Creating maps of the spatial and temporal distributiion of the output variables of species as well as their comparison.

Usage

```
plotSpeciesMaps(
  simulationID,
  output.variable = "Biomass",
  species = "all_species",
  years = "all_years",
  species.comparison = TRUE,
  output.threshold = 0,
  filename = NA,
  color.option = "amazing",
  animated = FALSE,
 ggplot.args = list(device = "png", scale = 1, width = NA, height = NA, units = "in",
    dpi = 300, duration = 15, limitsize = FALSE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.

`output.variable`
: The output variable for which the data should be plotted. Must be a selection of output variables created by TreeMig (`"SeedAntagonists"`, `"BasalArea"`, `"Biomass"`, `"Ingrowth"`, `"LAI"`, `"NPP"`, `"StemNumber"`, `"Pollen"`, `"Seeds"`, `"HeightClass_0"`, `"HeightStruct_01"`, `"HeightStruct_02"`, `"HeightStruct_03"`, `"HeightStruct_04"`, `"HeightStruct_05"`, `"HeightStruct_06"`, `"HeightStruct_07"` , `"HeightStruct_08"` , `"HeightStruct_09"`, `"HeightStruct_10"`, `"HeightStruct_11"`, `"HeightStruct_12"`, `"HeightStruct_13"`, `"HeightStruct_14"`, `"HeightStruct_15"` or `"all_variables"`.

`species`
: A vector with the names of the desired species (e.g., `c("Abies_alba", "Picea_abies")`) or `"all_species"`. This must be a subset of the species included in the selected simulationID.

`years`
: A vector of the years to be selected (e.g., `c(1900,2000)`) or `"all_years"`. This must be a subset of the years that were exported from the TreeMig simulation.

`species.comparison`
: A flag indicating if a single figure including all selected species and years should be drawn. This can be reduced by a minimal biomass requirement for inclusion(see `bm.threshold`). In this case no individual figures are drawn.

`output.threshold`
: A threshold excluding species which never reach a certain biomass from the species comparison figure.

`filename`
: A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.

`color.option`
: A choice of predefined color palette: `"natural"`,`"highcontrast"`,`"amazing"` or `"blue"` or a vector of custom colors.

`ggplot.args`
: A list of arguments to be given to the ggsave function. This allows to adjust the width, height, scaling, dpi and output format. For animations it also sets the duration of the animation. (e.g., `list(device = "png",scale = 1, width = 8, height = 8, units = "in", dpi = 300, limitsize = TRUE)`)

`amimated`
: Should the figure be an amimated time series rather than one figure per selected year?

Value

All the figures are directly written to the disk. Nothing is returned within R.

`plotSpeciesSummary` *Creating maps of the spatial and temporal distributiion of the biomass of species as well as their comparison.*

Description

Creating maps of the spatial and temporal distributiion of the biomass of species as well as their comparison.

Usage

```
plotSpeciesSummary(
  simulationID,
  species = "all_species",
  years = "all_years",
  output.variable = "all_variables",
  plot.type = "stacked_area",
  output.threshold = 0,
  sub.area = NULL,
  grid.arrange = FALSE,
  quant.boundries = NULL,
  filename = NA,
 ggplot.args = list(device = "png", scale = 1, width = 10, height = 5, units = "in", dpi
    = 300, limitsize = TRUE)
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`species`	A vector with the names of the desired species (e.g., `c("Abies_alba", "Picea_abies")`) or `"all_species"`. This must be a subset of the species included in the selected simulationID.
`years`	A vector of the years to be selected (e.g., `c(1900,2000)`) or `"all_years"`. This must be a subset of the years that were exported from the TreeMig simulation.
`output.variable`	The output variable for which the data should be plotted. Must be a selection of output variables created by TreeMig (`"SeedAntagonists"`, `"BasalArea"`, `"Biomass"`, `"Ingrowth"`, `"LAI"`, `"NPP"`, `"StemNumber"`, `"Pollen"`, `"Seeds"`, `"HeightClass_0"`, `"HeightStruct_01"`, `"HeightStruct_02"`, `"HeightStruct_03"`, `"HeightStruct_04"`, `"HeightStruct_05"`, `"HeightStruct_06"`, `"HeightStruct_07"` , `"HeightStruct_08"` , `"HeightStruct_09"`, `"HeightStruct_10"`, `"HeightStruct_11"`, `"HeightStruct_12"`, `"HeightStruct_13"`, `"HeightStruct_14"`, `"HeightStruct_15"` or `"all_variables"`. The function will plot one Figure per variable unless the flag. grid.arrange is set to TRUE
`plot.type`	Type of the plot, can be either line or stacked_area
`output.threshold`	A threshold excluding species which never reach a certain biomass from the species comparison figure.
`sub.area`	Either a dataframe with lon/lat Coordinates or a `"SpatialPolygonsDataFrame"` with polygon(s). In case there are several polygons one figure per polygon will be drawn.
`grid.arrange`	A flag to determine if a big grid with all selected output variables should be created.
`quant.boundries`	The lower and upper boundry for the shaded area in quantiles. e.g. `c(0.025,0.975)`. If NULL, no quantile area will be drawn.
`filename`	A custom name for the file without file extension. If NA the file will be named generically based on a rule set defined within the function.
`ggplot.args`	A list of arguments to be given to the ggsave function. This allows to adjust the width, height, scaling, dpi and output format. (e.g., `list(device =`

"png",scale = 1, width = 8, height = 8, units = "in", dpi = 300, limitsize = TRUE))

Value

All the figures are directly written to the disk. Nothing is returned within R.

`readCtr` *Reading the control file for a specific* `simulationID`.

Description

Reading the control file for a specific `simulationID`.

Usage

```
readCtr(simulationID, TMDirectory = getwd())
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`TMDirectory`	optional path to the TreeMig directory. If not set getwd() is used.

Value

The control file with all the information used in the TreeMig simulation with the `simulationID`.

`readOutputVariable` *Returning the data of a* `output.variable` *created in a speciefic TreeMig simulation (*`simulationID`*) for a subset of* `years` *and* `species`*. The data can either be sourced from a netCDF file (*`ncdf`*) or from the text files (*`txt`*). The return format can either be a multi-dimensional array (*`array`*) or as a dataframe (*`df`*).*

Description

Returning the data of a `output.variable` created in a speciefic TreeMig simulation (`simulationID`) for a subset of `years` and `species`. The data can either be sourced from a netCDF file (`ncdf`) or from the text files (`txt`). The return format can either be a multi-dimensional array (`array`) or as a dataframe (`df`).

Usage

```
readOutputVariable(
  simulationID,
  species,
  output.variable,
  years,
  sub.area = NULL,
  preferred.source = "ncdf",
  return.format = "array"
)
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`species`	The species that should be extracted. This must be a subset of the species simulated `species.names`.
`output.variable`	The variable to be extracted from the output data of TreeMig simulation with the `simulationID`.
`years`	The years that should be extracted. This must be a subset of the years that were written out in the TreeMig simulation.
`sub.area`	Either a dataframe with lon/lat Coordinates or a shapefile with polygon(s). This will remove all data points not selected in case only a df is returned. No effect on returning an array.
`preferred.source`	The desired source to pull the data from. This can either be `ncdf` or the `txt` files. If one is not available the other one is used. This simply helps to be more efficient depending on what is desired.
`return.format`	The format in which the data should be returned. This can either be a dataframe (`df`) with columns (lon, lat, year, species, var) or a multi-dimensional array (`array`) based on the ncdf arrangement. This mostly depends if the data is used for spatial statistics (e.g., extracting a shapefile or points). In that case the array is more efficient to be put into a raster brick. While for plotting with ggplot the dataframe is more efficient.

Value

Either a dataframe or a multi-dimensional array containing the information.

`readSpeciesNames` *Returning the names of all the species of a* `simulationID`.

Description

This function uses the `simulationID` to check which species were run in the simulation. This data is used to make sure that no species can be selected for plotting that were not part of the simulation.

Usage

```
readSpeciesNames(simulationID, TMDirectory = getwd())
```

Arguments

`simulationID`	The simulationID belonging to the desired TreeMig simulation.
`TMDirectory`	optional path to the TreeMig directory. If not set getwd() is used.

Value

A vector with the species names for all species run in the TreeMig simulation with the `simulationID`.

`ReadStateFile2Dataframe` *ReadStateFile2Dataframe*

Description

Reads from a TM state file to a data frame

Usage

```
ReadStateFile2Dataframe(newestStatefileNamePath)
```

Arguments

`newestStatefileNamePath;string;`	path and name of state file to be read in

Details

Reads the state file of TM and writes it to the data frame in the correct format

Value

state_df; dataframe of TM state file

`ReadTMOutput` *ReadTMOutput*

Description

Reads from a TM output file to a data frame

Usage

```
ReadTMOutput(simulationID, outputVar)
```

Arguments

`simulationID;string;`	simulation id, to get the correct TreeMig output
`outputVar;`	string; TM output variable to be read, one of c("Biomass","LAI","Numbers","Seeds","BA").

Details

Writes the data frame in TM_result_df_list_wholeIteration containing TM output variables to a TM output file in the correct format

Value

thisresult_df; dataframe of TM output for the wished variable

ReplaceOldOMResults *ReplaceOldOMResults*

Description

Replace old results in OM_var_data_wholeIteration

Usage

```
ReplaceOldOMResults(
  ti = ti,
  OM_var_data_wholeIteration = OM_var_data_wholeIteration,
  OM_var_data = OM_var_data,
  otherStuff = list()
)
```

Arguments

`ti`	INT; first time point of the SSP (0,1,2,...)
`OM_var_data_wholeIteration`	dataframe; contains all results of the other model
`OM_var_data`	dataframe ; all results of the other model in SSP
`otherStuff`	list; placeholder for potential other information to be stored and glued together

Details

#'Replaces all results (variable data) in the overall OM result file by the current results (variable data). As alternative to appending the results.

Value

OM_var_data_wholeIteration; contains all results of the other model up to tip1

runTreeMig *Run the TreeMig simulation*

Description

Saves the config to the TreeMig directory and runs the TreeMig program

Usage

```
runTreeMig(simulationID, config, opSystem = tolower(Sys.info()[["sysname"]]))
```

Arguments

`simulationID`	character. ID of the simulation
`config`	`TreeMigConfig`. A TreeMig configuration
`opSystem`	`string`. Specifies the operating system. Either 'windows' or 'unix'

Value

TreeMig result

`runTreeMigShiny` *Run the TreeMig Shiny Application*

Description

Run the TreeMig Shiny Application

Usage

```
runTreeMigShiny(
  launch.browser = TRUE,
  port = getOption("shiny.port"),
  defaultFiles = NULL
)
```

Arguments

`launch.browser`	If true, the system's default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only. This value of this parameter can also be a function to call with the application's URL.
`port`	The TCP port that the application should listen on. If the port is not specified, and the shiny.port option is set (with options(shiny.port = XX)), then that port will be used. Otherwise, use a random port between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.
`defaultFiles`	A named list of file paths. These files are automatically added to the Input Files in the shiny application.

`sampleBioclimate` *This function will create a bioclimate that is in the time range of the simulation (read from the control settings). If some simulation years are missing in the provided bioclimate, they will be sampled from the bioclimate. The years that should be used for sampling can be defined with sampleRangeBefore and sampleRangeAfter.*

Description

This function will create a bioclimate that is in the time range of the simulation (read from the control settings). If some simulation years are missing in the provided bioclimate, they will be sampled from the bioclimate. The years that should be used for sampling can be defined with sampleRangeBefore and sampleRangeAfter.

Usage

```
sampleBioclimate(BCLM, ctr, sampleRangeBefore, sampleRangeAfter)
```

Arguments

`BCLM`	data.frame. A bioclimate
`ctr`	The TreeMig config
`sampleRangeBefore`	vector. Years that should be sampled for previous years
`sampleRangeAfter`	vector. Years that should be sampled for subsequent years

Value

data.table. Bioclimate

`SearchAndReplaceStars` *SearchAndReplaceStars*

Description

searches several *** directly after a numeric number in the row, interpretes them as wrong value for antagonists and sets the value to 10.0

Usage

```
SearchAndReplaceStars(dt)
```

Arguments

`state_df;`	dataframe of TM state file

Details

Sometimes by the TreeMig program undefined values for antagonists are written out as a string of several "*" directly after the number before. In these cases, this column is lost and all other columns shift one position to the left, causing problems in their interpretation. E.g., nhc, the number of heigth classes, is wrong. Occurs only in old TreeMig binaries. To still be able to use them, this mending function is used.

Value

state_df; dataframe of TM state file, where *** are interpreted as own column and set to 10.0

`setBioclimate`	*Prepares all the files required for TreeMig and defines the paths accordingly in the TreeMigConfig. This includes writing bioclimate and stock data to the TreeMig E directory, writing species parameter file for selected species. If the option readInitialStateFromStateFile is TRUE a state file (stateFile) must be specified.*

Description

Prepares all the files required for TreeMig and defines the paths accordingly in the TreeMigConfig. This includes writing bioclimate and stock data to the TreeMig E directory, writing species parameter file for selected species. If the option readInitialStateFromStateFile is TRUE a state file (stateFile) must be specified.

Usage

```
setBioclimate(config, bioClim, output_name)
```

Arguments

`config`	The TreeMig config
`bioClim`	data.frame or file name. Calculated with getBioclimate()
`output_name`	character. The output name of the bclm

`setCtrOptions`	*Changes control variables according to study area. This will only change the TreeMigConfig variable*

Description

Changes control variables according to study area. This will only change the TreeMigConfig variable

Usage

```
setCtrOptions(config, studyArea, unitOfInputData = 1000)
```

Arguments

`config`	`TreeMigConfig`. A TreeMig configuration
`studyArea`	`studyArea`. Study area, for data extraction
`unitOfInputData`	integer. One unit of input data (stock, bioClim) corresponds to x m

`StudyArea-class` *StudyArea Class*

Description

A reference class to define a study area using raster data. It provides methods to initialize and retrieve information about the study area.

`TailorTMStateFile2TMInputFile` *TailorTMStateFile2TMInputFile: Cutting away cells in statefile that are not in bioclim file*

Description

It might be that in the coupling the bioclim file is changed. To have a statefile that fits the current bioclim file, it is cut down to the cells of the bioclim file in the respective year

Usage

```
TailorTMStateFile2TMInputFile(TM_state_tip1_modified, TM_input_modified, ctr)
```

Arguments

`TM_state_tip1_modified`	dataframe; TreeMig state at tip1 after running the other model; columns species,year,lat,lon,(REAL nr of) seeds, (REAL nr of) of antagonists,nhc (INT nr of height classes) , REAL nr of trees in height classes 0 to 15
`TM_input_modified`	dataframe, optional; TreeMig input (bioclim) at tip1 after running the other model
`ctr;`	???, control settings of TreeMig

`validateEach` *Function used to validate input variables*

Description

Function used to validate input variables

Usage

```
validateEach(var, ...)
```

Arguments

`var`	a list of variables
`...`	validation requirements

`WriteDataframe2StateFile` *WriteDataframe2StateFile*

Description

Writes the TM state dataframe to a TM state file

Usage

```
WriteDataframe2StateFile(state_df, newestStatefileNamePath)
```

Arguments

`newestStatefileNamePath;string;`	path and name of state file to be writtec
`state_df;`	dataframe of TM state file

Details

Writes the TM state dataframe to a TM state file at the correct location and in the correct format

`writeSpecParsFileForSelectedSpecies` *Creates species pars file for selected species*

Description

Creates species pars file for selected species

Usage

```
writeSpecParsFileForSelectedSpecies(
  specSelection,
  specAll,
  outParseFile,
  specParsFileAll,
  ctr
)
```

Arguments

`specSelection`	character vector. Vector with species names, "Standard" selects all species from specAll
`specAll`	character vector. Vector with all species names, "Standard" selects all given species from pars file
`outParseFile`	character. File where species pars should be written to
`specParsFileAll`	character. File with species data (Name,kDMax, kHMax, kAMax etc.)
`ctr`	TreeMigConfig. A TreeMig config

WriteTMOutputVars *WriteTMOutputVars*

Description

TM output data frame -> TM output file

Usage

```
WriteTMOutputVars(
  TM_OutputVars,
  simulationID,
  TM_result_df_list_wholeIteration
)
```

Arguments

`simulationID;string;`	simulation id, to put the current TreeMig output to the correct folder
`TM_result_df_list_wholeIteration;list`	of dataframes, each df contains the simulation results for the SSPs up to ti
`TM_OutputVars;`	vector of strings, one or several of c("Biomass","LAI","Numbers","Seeds","BA").

Details

Writes the data frame in TM_result_df_list_wholeIteration containing TM output variables to a TM output file in the correct format