Package ‘PointedSDMs’

July 8, 2024

Type Package

Title Fit Models Derived from Point Processes to Species Distributions using ‘inlabru’

Version 2.1.0

Maintainer Philip Mostert <philip.s.mostert@ntnu.no>

URL https://github.com/PhilipMostert/PointedSDMs

BugReports https://github.com/PhilipMostert/PointedSDMs/issues

Description Integrated species distribution modeling is a rising field in quantitative ecology thanks to significant rises in the quantity of data available, increases in computational speed and the proven benefits of using such models. Despite this, the general software to help ecologists construct such models in an easy-to-use framework is lacking. We therefore introduce the R package ‘PointedSDMs’: which provides the tools to help ecologists set up integrated models and perform inference on them. There are also functions within the package to help run spatial cross-validation for model selection, as well as generic plotting and predicting functions. An introduction to these methods is discussed in Issac, Jarzyna, Keil, Dambly, Boersch-Supan, Browning, Freeman, Golding, Guillera-Arroita, Henrys, Jarvis, Lahoz-Monfort, Pagel, Pescott, Schmucki, Simmons and O’Hara (2020) <doi:10.1016/j.tree.2019.08.006>.

Depends R (>= 4.1), stats, sf, inlabru (>= 2.8.0), R6 (>= 2.5), methods,

Imports terra, ggplot2, raster, sp (>= 1.4-5), R.devices, blockCV (>= 3.0.0)

Suggests testthat (>= 3.0.0), sn, INLA (>= 21.08.31), rasterVis, ggtmap, ggpolypath, RColorBrewer, cowplot, knitr, kableExtra, markdown, spocc, covr

Additional_repositories https://inla.r-inla-download.org/R/testing

RoxygenNote 7.3.1

License GPL (>= 3)

VignetteBuilder knitr
Config/testthat/edition 3
LazyData false
Encoding UTF-8
NeedsCompilation no
Author Philip Mostert [aut, cre],
Bob O’hara [aut]
Repository CRAN
Date/Publication 2024-07-08 13:50:06 UTC

Contents

BBA .................................................. 3
BBS .................................................. 3
BBS_ColinusVirginianus .................... 4
blockedCV ........................................ 4
blockedCV-class .............................. 5
bruSDM-class .................................. 5
bruSDM_predict-class ..................... 6
changeCoords .................................. 6
checkCoords .................................. 6
checkVar ........................................ 7
data2ENV .......................................... 7
dataOrganize ................................... 8
dataSet .......................................... 10
datasetOut ....................................... 11
datasetOut-class ......................... 13
eBird ............................................. 13
elev_raster .................................... 13
fitISDM .......................................... 14
Gbif .............................................. 15
intModel ......................................... 15
Koala ............................................. 19
makeFormulaComps ......................... 20
makeLhoods .................................... 20
modISDM-class ................................ 21
modISDM_predict-class ................... 21
modMarks-class ................................ 21
modMarks_predict-class .................. 21
modSpecies-class ............................. 22
modSpecies_predict-class ............. 22
nameChanger ................................... 22
NLCD_canopy_raster ....................... 23
Parks ............................................. 23
plot.bruSDM_predict ..................... 23
predict.bruSDM ............................... 27
print.blockedCV ............................... 32
Dataset of Setophaga caerulescens obtained from the Pennsylvania Atlas of Breeding Birds.

**Description**

Dataset of Setophaga caerulescens obtained from the Pennsylvania Atlas of Breeding Birds.

**References**

https://ebird.org/atlaspa/home

---

Dataset of Setophaga caerulescens obtained from the North American Breeding Bird survey across Pennsylvania state.

**Description**

Dataset of Setophaga caerulescens obtained from the North American Breeding Bird survey across Pennsylvania state.

**References**

https://www.pwrc.usgs.gov/bbs/
Dataset of Colinus Virginianus obtained from the North American Breeding Bird survey across Alabama state.

Description

This function is used to perform spatial blocked cross-validation with regards to model selection for the integrated model. It does so by leaving out a block of data in the full model, running a model with the remaining data, and then calculating the deviance information criteria (DIC) as a score of model fit.

Usage

`blockedCV(data, options = list())`

Arguments

- `data`: An object produced by either `startISDM` or `startSpecies`. Requires the slot function, `.SpatialBlock` to be run first in order to specify how the data in the model is blocked.

- `options`: A list of `INLA` or `inlabru` options to be used in the model. Defaults to `list()`.

Value

An object of class `blockedCV`, which is essentially a list of DIC values obtained from each iteration of the model.
Examples

```r
## Not run:
if(requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
                           responsePA = 'Present',
                           Projection = proj)

# Set up spatial block
organizedData$spatialBlock(k = 2, rows = 2, cols = 1)

# Run spatial block cross-validation
blocked <- blockedCV(organizedData)

# Print summary
blocked
}
```

## End(Not run)

---

**blockedCV-class**

*Export class blockedCV*

**Description**

Export class blockedCV

---

**bruSDM-class**

*Export bru_sdm class*

**Description**

Export bru_sdm class
checkCoords

bruSOM_predict-class  
Export class predict_bru_sdm

changeCoords

changeCoords: function used to change coordinate names.

Description
An internal function used to change the coordinate names for datasets.

Usage
changeCoords(data, oldcoords, newcoords)

Arguments
- data: A list of datasets.
- oldcoords: The old coordinate names.
- newcoords: The new coordinate names.

Value
A list of data.frame or spatial objects with the coordinate names changed.

checkCoords

checkCoords: function used to check coordinate names.

Description
An internal function used to check if all the coordinates are the same.

Usage
checkCoords(data, coords)

Arguments
- data: A list of datasets.
- coords: A vector of length 2 of the coordinate names.

Value
A logical variable.
checkVar

checkVar: Function used to check variable names.

Description
Internal function used to check if variable is in dataset.

Usage
checkVar(data, var)

Arguments
- data: A list of datasets.
- var: A variable name.

Value
A logical variable

data2ENV
data2ENV: function used to move objects from one environment to another.

Description
Internal function: used to assign objects specified in bruSDM to the dataSDM/blockedCV function environments.

Usage
data2ENV(data, env)

Arguments
- data: bruSDM data file to be used in the integrated model.
- env: Environment where the objects should be assigned.

Value
Assignment of the relevant spatial fields to the specified environment.
dataOrganize  

*R6 class to assist in reformatting the data to be used in dataSDM.*

**Description**

Internal functions used to temporarily store data and other information before adding to dataSDM.

**Methods**

**Public methods:**

- `dataOrganize$makeData()`
- `dataOrganize$makeSpecies()`
- `dataOrganize$makeMultinom()`
- `dataOrganize$makeFormulas()`
- `dataOrganize$makeComponents()`
- `dataOrganize$makeLhoods()`
- `dataOrganize$clone()`

**Method makeData():**

*Usage:*

```r
dataOrganize$makeData(
  datapoints, 
  datanames, 
  coords, 
  proj, 
  marktrialname, 
  paresp, 
  countsresp, 
  trialname, 
  speciesname, 
  marks, 
  pointcovnames, 
  markfamily, 
  temporalvar, 
  offsetname 
)
```

**Method makeSpecies():**

*Usage:*

```r
dataOrganize$makeSpecies(speciesname, repl = FALSE)
```

**Method makeMultinom():**

*Usage:*

```r
dataOrganize$makeMultinom(multinomVars, return, oldVars, repl = FALSE)
```
**Method** makeFormulas():

**Usage:**

```r
dataOrganize$makeFormulas(
  spatcovs,
  speciesname,
  paresp,
  countresp,
  marks,
  marksspatial,
  speciesintercept,
  speciesenvironment,
  spatial,
  intercept,
  temporalname,
  speciesindependent,
  markintercept,
  pointcovs,
  speciesspatial,
  biasformula,
  covariateformula
)
```

**Method** makeComponents():

**Usage:**

```r
dataOrganize$makeComponents(
  spatial,
  intercepts,
  datanames,
  marks,
  speciesname,
  multinomnames,
  pointcovariates,
  covariatenames,
  covariateclass,
  marksspatial,
  marksintercept,
  temporalname,
  speciesspatial,
  speciesenvironment,
  speciesintercept,
  numtime,
  temporalmodel,
  offsetname,
  copymodel,
  speciesindependent,
  biasformula,
  covariateformula,
  marksCopy
)```
Method `makeLhoods()`:

Usage:
dataOrganize$makeLhoods(
    mesh,
    ips,
    paresp,
    ntrialsvar,
    markstrialsvar,
    speciesname
)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
dataOrganize$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

dataSet

Internal function used to standardize datasets, as well as assign metadata.

Description

Internal function used to assist in structuring the data.

Usage

dataSet(
    datapoints,
    datanames,
    coords = c("CoordLoc1", "CoordLoc2"),
    proj,
    pointcovnames,
    paresp,
    countsresp,
    trialname,
    speciesname,
    marks,
    marktrialname,
    markfamily,
    temporalvar,
    offsetname
)
**datasetOut**

**Arguments**

- **datapoints**: A list of datasets as sf objects
- **datanames**: A vector of the names of the datasets.
- **coords**: Names of the coordinates used in the model.
- **proj**: The projection reference system used in the model.
- **pointcovnames**: Name of the point covariates used in the model.
- **paresp**: Name of the response variable used by the presence absence datasets.
- **countsresp**: Name of the response variable used by the counts data.
- **trialname**: Name of the trial variable used by the presence absence datasets.
- **speciesname**: Name of the species name variable.
- **marks**: Name of the marks considered in the model.
- **marktrialname**: Name of the trial variable used by the binomial marks.
- **markfamily**: A vector describing the distribution of the marks.
- **temporalvar**: Name of the temporal variable.
- **offsetname**: Name of the offset column in the datasets.

**Value**

A list of relevant metadata

---

**datasetOut**

*datasetOut: function that removes a dataset out of the main model, and calculates some cross-validation score.*

**Description**

This function calculates the difference in covariate values between a full integrated model and a model with one dataset left out, as well as some cross-validation score, which is used to obtain a score of the relative importance of the dataset in the full model. The score is calculated as follows:

1. Running a new model with one less dataset (from the main model) – resulting in a reduced model,
2. predicting the intensity function at the locations of the left-out dataset with the reduced model,
3. using the predicted values as an offset in a new model,
4. finding the difference between the marginal-likelihood of the main model (ie the model with all the datasets considered) and the marginal-likelihood of the offset model.

**Usage**

```r
datasetOut(model, dataset, predictions = TRUE)
```
Arguments

- **model**: Model of class modISDM run with multiple datasets.
- **dataset**: Names of the datasets to leave out. If missing, will run for all datasets used in the full model.
- **predictions**: Will new models be used for predictions. If TRUE returns marginals and bru_info in model. Defaults to TRUE.

Value

A list of inlabru models with the specified dataset left out. If predictions is FALSE, these objects will be missing their bru_info and call lists.

Examples

```r
## Not run:
if (requireNamespace('INLA')) {
  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj

  # Set model up
  organizedData <- startISDM(data, Mesh = mesh,
                              Projection = proj,
                              responsePA = 'Present')

  ## Run the model
  modelRun <- fitISDM(organizedData,
                      options = list(control.inla = list(int.strategy = 'eb')))

  # Choose dataset to leave out
  eBirdOut <- datasetOut(modelRun, dataset = 'eBird')

  # Print datasetOut summary
  eBirdOut

}
## End(Not run)
```
datasetOut-class

Export class bru_sdm_leave_one_out

Description

Export class bru_sdm_leave_one_out

eBird
data.frame object containing solitary tinamou observations from eBird

Description

data.frame object containing solitary tinamou observations from eBird

References

http://ebird.org/

elev_raster

Raster object containing the elevation across Pennsylvania state.

Description

Raster object containing the elevation across Pennsylvania state.

References

https://cran.r-project.org/package=elevatr
fitISDM: function used to run the integrated model.

Description

This function takes an intModel object and produces an inlabru model object with additional lists and meta-data added.

Usage

fitISDM(data, options = list())

Arguments

data A intModel object to be used in the integrated model.
options A list of INLA options used in the model. Defaults to list().

Value

An inlabru model with additional lists containing some more metadata attached.

Examples

## Not run:

```r
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                           Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData,
                    options = list(control.inla = list(int.strategy = 'eb')))}

# Print summary of model
modelRun
}
## End(Not run)
```
Gbif
data.frame object containing solitary tinamou observations from Gbif

Description
data.frame object containing solitary tinamou observations from Gbif

Source
https://www.gbif.org/

intModel
intModel: Function used to initialize the integrated species distribution model.

Description
This function is depreciated. Please use one of startISDM or startSpecies.

Usage
intModel(
  ..., spatialCovariates = NULL,
  Coordinates, Projection, Mesh,
  IPS = NULL, Boundary = NULL,
  speciesSpatial = "copy", speciesIndependent = FALSE,
  markNames = NULL, markFamily = NULL,
  pointCovariates = NULL, pointsIntercept = TRUE,
  marksIntercept = TRUE, speciesEffects = list(randomIntercept = FALSE, Environmental = TRUE),
  Offset = NULL, pointsSpatial = "copy", marksSpatial = TRUE,
  responseCounts = "counts", responsePA = "present",
  trialsPA = NULL, trialsMarks = NULL,
  speciesName = NULL,
temporalName = NULL,
temporalModel = list(model = "ar1"),
copyModel = list(beta = list(fixed = FALSE)),
Formulas = list(covariateFormula = NULL, biasFormula = NULL)
)

Arguments

... The datasets to be used in the model. May come as either sf, data.frame or SpatialPoints* objects, or as a list of objects with these classes. The classes of the datasets do not necessarily need to be standardized, however the variable names within them often have to be.
spatialCovariates The spatial covariates used in the model. These covariates must be measured at every location (pixel) in the study area, and must be a Raster*, SpatialPixelsDataFrame or SpatialRaster object. Can be either numeric, factor or character data.
Coordinates A vector of length 2 containing the names (class character) of the coordinate variables used in the model.
Projection The coordinate reference system used by both the spatial points and spatial covariates. Must be of class character.
Mesh An inla.mesh object required for the spatial random fields and the integration points in the model (see inla.mesh.2d from the INLA package for more details).
IPS The integration points to be used in the model (that is, the points on the map where the intensity of the model is calculated). See fm_int from the inlabru package for more details regarding these points; however defaults to NULL which will create integration points from the Mesh object.
Boundary A sf object of the study area. If not missing, this object is used to help create the integration points.
speciesSpatial Argument to specify if each species should have their own spatial effect with different hyperparameters to be estimated using INLA’s "replicate" feature, or if a the field’s should be estimated per species copied across datasets using INLA’s "copy" feature. Possible values include: 'replicate', 'copy', 'shared' or NULL if no species-specific spatial effects should be estimated.
speciesIndependent Logical argument: Should species effects be made independent of one another. Defaults to FALSE which creates effects for each species independently.
markNames A vector with the mark names (class character) to be included in the integrated model. Marks are variables which are used to describe the individual points in the model (for example, in the field of ecology the size of the species or its feeding type could be considered). Defaults to NULL, however if this argument is non-NULL, the model run will become a marked point process. The marks must be included in the same data object as the points.
markFamily A vector with the statistical families (class character) assumed for the marks. Must be the same length as markNames, and the position of the mark in the vector markName is associated with the position of the family in markFamily. Defaults to NULL which assigns each mark as "Gaussian".
pointCovariates
The non-spatial covariates to be included in the integrated model (for example, in the field of ecology the distance to the nearest road or time spent sampling could be considered). These covariates must be included in the same data object as the points.

pointsIntercept
Logical argument: should the points be modeled with intercepts. Defaults to TRUE. Note that if this argument is non-NULL and pointsIntercepts is missing, pointsIntercepts will be set to FALSE.

marksIntercept
Logical argument: should the marks be modeled with intercepts. Defaults to TRUE.

speciesEffects
List specifying if intercept terms and environments effects should be made for the species. Defaults to `list(randomIntercept = FALSE, Environmental = TRUE)`. randomIntercept may take on three values: TRUE which creates a random intercept for each species, FALSE which creates fixed intercepts for each species, of NULL which removes all species level intercepts. Note that if randomIntercept = NULL and pointsIntercept = TRUE, dataset specific intercept terms will be created.

Offset
Name of the offset variable (class character) in the datasets. Defaults to NULL; if the argument is non-NULL, the variable name needs to be standardized across datasets (but does not need to be included in all datasets). The offset variable will be transformed onto the log-scale in the integrated model.

pointsSpatial
Argument to determine whether the spatial field is shared between the datasets, or if each dataset has its own unique spatial field. The datasets may share a spatial field with INLA's "copy" feature if the argument is set to copy. May take on the values: "shared", "individual", "copy", "correlation" or NULL if no spatial field is required for the model. Defaults to "copy".

marksSpatial
Logical argument: should the marks have their own spatial field. Defaults to TRUE.

responseCounts
Name of the response variable in the counts/abundance datasets. This variable name needs to be standardized across all counts datasets used in the integrated model. Defaults to 'counts'.

responsePA
Name of the response variable (class character) in the presence absence/detection non-detection datasets. This variable name needs to be standardized across all present absence datasets. Defaults to 'present'.

trialsPA
Name of the trials response variable (class character) for the presence absence datasets. Defaults to NULL.

trialsMarks
Name of the trials response variable (class character) for the binomial marks (if included). Defaults to NULL.

speciesName
Name of the species variable name (class character). Specifying this argument turns the model into a stacked species distribution model, and calculates covariate values for the individual species, as well as a species group model in the shared spatial field. Defaults to NULL. Note that if this argument is non-NULL and pointsIntercepts is missing, pointsIntercepts will be set to FALSE.

temporalName
Name of the temporal variable (class character) in the model. This variable is required to be in all the datasets. Defaults to NULL.
temporalModel  List of model specifications given to the control.group argument in the time effect component. Defaults to list(model = 'ar1'); see control.group from the INLA package for more details.

copyModel  List of model specifications given to the hyper parameters for the "copy" model. Defaults to list(beta = list(fixed = FALSE)).

Formulas  A named list with two objects. The first one, covariateFormula, is a formula for the covariates and their transformations for the distribution part of the model. Defaults to NULL which includes all covariates specified in spatialCovariates into the model. The second, biasFormula, specifies which covariates are used for the PO datasets. Defaults to NULL which includes no covariates for the PO datasets.

Value  A specifyISDM object (class R6). Use ?specifyISDM to get a comprehensive description of the slot functions associated with this object.

Note  The idea with this function is to describe the full model: that is, all the covariates and spatial effects will appear in all the formulas for the datasets and species. If some of these terms should not be included in certain observation models in the integrated model, they can be thinned out using the .$updateFormula function. Note: the point covariate and mark terms will only be included in the formulas for where they are present in a given dataset, and so these terms do not need to be thinned out if they are not required by certain observation models.

Examples  
## Not run:
if (requireNamespace('INLA')) {

#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

#Set base model up
baseModel <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
  Projection = proj, responsePA = 'Present')

#Print summary
baseModel

#Set up model with dataset specific spatial fields
indSpat <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
  Projection = proj, pointsSpatial = 'individual', responsePA = 'Present')
Koala

# Model with offset variable
offSet <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                   Projection = proj, Offset = 'area', responsePA = 'Present')

# Assume area as a mark
markModel <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                       Projection = proj, markNames = 'area', markFamily = 'gamma',
                       responsePA = 'Present')

}#

## End(Not run)

Koala

Dataset of Eucalyptus globulus (common name: blue gum) sightings collected across the Koala conservation reserve on Phillip island (Australia) between 1993 and 2004. Two marks are considered from this dataset: "koala" which describes the number of koala visits to each tree, and "food" which is some index of the palatability of the leaves.

Description

Dataset of Eucalyptus globulus (common name: blue gum) sightings collected across the Koala conservation reserve on Phillip island (Australia) between 1993 and 2004. Two marks are considered from this dataset: "koala" which describes the number of koala visits to each tree, and "food" which is some index of the palatability of the leaves.

Format

eucTrees is a data.frame object (1284 observations; 6 variables) with the following columns:

- **E** Latitude of the observation.
- **N** Longitude of the observation.
- **FOOD** Some value of the palatability of the leaves of the trees.
- **koala** The number of koala sightings per tree.
- **nitrogen** Total nitrogen (mg.g-1 DM).
- **dbh** Diameter at breast hight of the trees (cm).

References

### makeFormulaComps

**Description**

An internal function used to make the formula components required for inlabru.

**Usage**

```r
makeFormulaComps(form, species, speciesnames, type)
```

**Arguments**

- `form` The formula which needs to be changed into a component.
- `species` Logical indicating if species occur in the model.
- `speciesnames` The names of the species occurring in the model.
- `type` What type of component is being created: one for the covariate formula or the bias formula.

**Value**

A vector of components required for inlabru.

### makeLhoods

**Description**

Function to make the datasets into likelihoods.

**Usage**

```r
makeLhoods(
  data, formula, family, mesh, ips, paresp, ntrialsvar, markstrialsvar, speciesname, speciesindex, samplers, pointcova = NULL
)
```
### modISDM-class

**Arguments**

- **data** A list of sf objects containing the datasets for which likelihoods need to be constructed.
- **formula** A list of formulas to add to the likelihoods.
- **family** A list of vectors containing the families within each dataset.
- **mesh** An inla.mesh object.
- **ips** Integration points used.
- **paresp** The response variable name for the presence absence datasets.
- **ntrialsvar** The trials variable name for the presence absence datasets.
- **markstrialsvvar** The trial variable name for the binomial marks.
- **speciesname** The name of the species variable used.
- **speciesindex** A vector containing the numeric index of where the species occurs in the data
- **samplers** A list of integration domains for the datasets.
- **pointcovs** A vector of the point covariates used in the model.

---

### Description

Export modISDM class

---

### modISDM_predict-class

**Description**

Export class predict.modISDM

---

### modMarks-class

**Description**

Export modMarks class
nameChanger

### modMarks_predict-class

Export class `predict_modMarks`

### Description

Export class `predict_modMarks`

### modSpecies-class

Export `modSpecies` class

### Description

Export `modSpecies` class

### modSpecies_predict-class

Export class `predict_modSpecies`

### Description

Export class `predict_modSpecies`

### nameChanger

**nameChanger**: function to change a variable name.

### Description

An internal function used to change the name of a variable.

### Usage

`nameChanger(data, oldName, newName)`

### Arguments

- **data**: A list of datasets.
- **oldName**: The old variable name.
- **newName**: The new variable name.

### Value

A list of data.frame or spatial objects with the name of the variable changes.
### NLCD_canopy_raster

**Description**

Raster object containing the canopy cover across Pennsylvania state.

**References**

https://cran.r-project.org/package=FedData

---

### Parks
data.frame object containing solitary tinamou observations from Parks

**Description**

data.frame object containing solitary tinamou observations from Parks

**Source**

https://www.gbif.org/

---

### plot.bruSDM_predict

Generic plot function for predict_bru_sdm.

**Description**

Plot for predict_bru_sdm
Plot for modISDM_predict
Plot for modMarks_predict
Plot for modSpecies_predict
### S3 method for class 'bruSDM_predict'
```r
plot(
  x,
  whattoplot = c("mean"),
  cols = NULL,
  layout = NULL,
  colourLow = NULL,
  colourHigh = NULL,
  plot = TRUE,
  ...
)
```

### S3 method for class 'modISDM_predict'
```r
plot(x, variable = "mean", plot = TRUE, ...)
```

### S3 method for class 'modMarks_predict'
```r
plot(x, variable = "mean", plot = TRUE, ...)
```

### S3 method for class 'modSpecies_predict'
```r
plot(x, variable = "mean", plot = TRUE, ...)
```

#### Arguments

- **x**: A modSpecies_predict object.
- **whattoplot**: One of the following statistics to plot: "mean", "sd", "q0.025", "median", "q0.975", "smin", "smax", "cv", "var"
- **cols**: Number of columns required for the plotting. Used by inlabru’s multiplot function.
- **layout**: Layout of the plots. Used by inlabru’s multiplot function.
- **colourLow**: Colour for the low values in the predictions (see `?scale_colour_gradient` from ggplot2). Defaults to NULL. If non-NULL, colourHigh is required.
- **colourHigh**: Colour for the high values in the predictions (see `?scale_colour_gradient` from ggplot2). Defaults to NULL. If non-NULL, colourLow is required.
- **plot**: Should the plots be printed, defaults to TRUE. If FALSE will produce a list of ggplot objects.
- **variable**: One of the following statistics to plot: "mean", "sd", "q0.025", "median", "q0.975", "smin", "smax", "cv", "var"

#### Value

A ggplot2 object.
A ggplot2 object.
A ggplot2 object.
A ggplot2 object.
Examples

```r
## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                           Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb')))  

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

# Make generic plot of predictions
plot(predictions, colourHigh = 'red', colourLow = 'orange')
}

## End(Not run)
```

## Not run:
```r
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                           Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb')))  

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

# Make generic plot of predictions
plot(predictions, colourHigh = 'red', colourLow = 'orange')
```
## End(Not run)

## Not run:

```r
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- startMarks(data, Mesh = mesh,
   Projection = proj, responsePA = 'Present',
   markNames = 'speciesName',
   markFamily = 'multinomial')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb',
   diagonal = 1)))

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

# Make generic plot of predictions
plot(predictions)
}
```

## End(Not run)

## Not run:

```r
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
   Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb')))
#Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

#Make generic plot of predictions
plot(predictions, colourHigh = 'red', colourLow = 'orange')

}  # End(Not run)

## End(Not run)

---

**Description**

Predict function for the object produced by `fitISDM`. Should act identically to `inlabru`'s generic predict function if wanted, but has additional arguments to help predict certain components created by the model. This is needed since `intModel` creates variable names which might not be directly apparent to the user.

Predict function for the object produced by `fitISDM`. Should act identically to `inlabru`'s generic predict function if wanted, but has additional arguments to help predict certain components created by the model. This is needed since `startISDM` creates variable names which might not be directly apparent to the user.

Predict function for the object produced by `fitISDM`. Should act identically to `inlabru`'s generic predict function if wanted, but has additional arguments to help predict certain components created by the model. This is needed since `startMarks` creates variable names which might not be directly apparent to the user.

Predict function for the object produced by `fitISDM`. Should act identically to `inlabru`'s generic predict function if wanted, but has additional arguments to help predict certain components created by the model. This is needed since `startSpecies` creates variable names which might not be directly apparent to the user.

## Usage

```r
## S3 method for class 'bruSDM'
predict(
  object,
  data = NULL,
  formula = NULL,
  mesh = NULL,
  mask = NULL,
  temporal = FALSE,
  covariates = NULL,
  spatial = FALSE,
  intercepts = FALSE,
)```
datasets = NULL,
species = NULL,
marks = NULL,
biasfield = FALSE,
biasnames = NULL,
predictor = FALSE,
fun = "linear",
format = "sf",
...)

## S3 method for class 'modISDM'
predict(
  object,
data = NULL,
formula = NULL,
mesh = NULL,
mask = NULL,
covariates = NULL,
spatial = FALSE,
intercepts = FALSE,
datasets = NULL,
bias = FALSE,
biasnames = NULL,
predictor = FALSE,
fun = "linear",
...)

## S3 method for class 'modMarks'
predict(
  object,
data = NULL,
formula = NULL,
mesh = NULL,
mask = NULL,
covariates = NULL,
spatial = FALSE,
intercepts = FALSE,
datasets = NULL,
mats = NULL,
bias = FALSE,
biasnames = NULL,
predictor = FALSE,
fun = "linear",
...)

# S3 method for class 'modSpecies'
predict(
  object,
  data = NULL,
  formula = NULL,
  mesh = NULL,
  mask = NULL,
  covariates = NULL,
  spatial = FALSE,
  intercepts = FALSE,
  datasets = NULL,
  species = NULL,
  bias = FALSE,
  biasnames = NULL,
  predictor = FALSE,
  fun = "linear",
  ...
)

Arguments

- **object**: A `modSpecies` object.
- **data**: Data containing points of the map with which to predict on. May be `NULL` if one of `mesh` or `mask` is `NULL`.
- **formula**: Formula to predict. May be `NULL` if other arguments: `covariates`, `spatial`, `intercepts` are not `NULL`.
- **mesh**: An `inla.mesh` object.
- **mask**: A mask of the study background. Defaults to `NULL`.
- **temporal**: Make predictions for the temporal component of the model.
- **covariates**: Name of covariates to predict.
- **spatial**: Logical: include spatial effects in prediction. Defaults to `FALSE`.
- **intercepts**: Logical: include intercept terms in prediction. Defaults to `FALSE`.
- **datasets**: Names of the datasets to include intercept and spatial term.
- **species**: Names of the species to predict. Default of `NULL` results in all species being predicted.
- **marks**: Names of the marks to include intercept and spatial term.
- **biasfield**: Logical include bias field in prediction. Defaults to `FALSE`.
- **biasnames**: Names of the datasets to include bias term. Defaults to `NULL`. Note: the chosen dataset needs to be run with a bias field first; this can be done using `.addBias` with the object produced by `intModel`.
- **predictor**: Should all terms (except the bias terms) included in the linear predictor be used in the predictions. Defaults to `FALSE`.
- **fun**: Function used to predict. Set to 'linear' if effects on the linear scale are desired.
format
Class of the data for which to predict on. Must be one of 'sp', 'sf' or 'terra'.
Defaults to 'sf'.

... Additional arguments used by the inlabru predict function.

bias Logical include bias field in prediction. Defaults to FALSE.

Details
Predict for bru_sdm
Predict for modISDM
Predict for modMarks
Predict for modSpecies

Value
A list of inlabru predict objects.
A list of inlabru predict objects.
A list of inlabru predict objects.
A list of inlabru predict objects.

Examples

## Not run:

if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                          Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb'))) 

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

}

## End(Not run)

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
                           Projection = proj,
                           responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb')))

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')

}

## End(Not run)

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
data <- lapply(data, function(x) {x$mark = runif(nrow(x));x})
mesh$crs <- proj

# Set model up
organizedData <- startMarks(data, Mesh = mesh, markNames = 'mark',
                        markFamily = 'gaussian',
                        Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb',
                               diagonal = 1)))

# Predict spatial field on linear scale
predictions <- predict(modelRun, mesh = mesh, marks = 'mark', fun = 'linear')

}

## End(Not run)
## Not run:

if (requireNamespace('INLA')) {
  
  # Get data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj

  # Set model up
  organizedData <- startSpecies(data, Mesh = mesh, speciesName = 'speciesName',
                                 Projection = proj, responsePA = 'Present')

  ## Run the model
  modelRun <- fitISDM(organizedData, options = list(control.inla = list(int.strategy = 'eb',
                                                       diagonal = 1)))

  # Predict spatial field on linear scale
  predictions <- predict(modelRun, mesh = mesh, spatial = TRUE, fun = 'linear')
}

## End(Not run)

---

print.blockedCV  

*Print function for blockedCV.*

### Description

Print for blockedCV

### Usage

```r
## S3 method for class 'blockedCV'
print(x, ...)
```

### Arguments

- `x`  
  A blockedCV object.

- `...`  
  Unused argument.
**print.bruSDM**

---

**Generic print function for bruSDM.**

**Description**

Print method for bru_sdm

**Usage**

```r
## S3 method for class 'bruSDM'
print(x, ...)
```

**Arguments**

- **x** bruSDM object.
- **...** Un used argument.

---

**print.bruSDM_predict**

---

**Generic print function for bru_sdm_predict.**

**Description**

Generic print function for bru_sdm_predict.

**Usage**

```r
## S3 method for class 'bruSDM_predict'
print(x, ...)
```

**Arguments**

- **x** bruSDM_predict object
- **...** Not used.
**print.datasetOut**  
*Generic print function for datasetOut.*

**Description**
Print for bru_sdm_leave_one_out

**Usage**
```r
## S3 method for class 'datasetOut'
print(x, ...)
```

**Arguments**
- `x` datasetOut object.
- `...` Unused argument.

**print.modISDM**  
*Generic print function for modISDM.*

**Description**
Print method for modISDM

**Usage**
```r
## S3 method for class 'modISDM'
print(x, ...)
```

**Arguments**
- `x` modISDM object.
- `...` Not used.
print.modMarks

Generic print function for modMarks.

Description
Print method for modMarks

Usage
## S3 method for class 'modMarks'
print(x, ...)

Arguments
x modMarks object.
...
Not used.

print.modSpecies

Generic print function for modSpecies.

Description
Print method for modSpecies

Usage
## S3 method for class 'modSpecies'
print(x, ...)

Arguments
x modSpecies object.
...
Not used.
reduceComps

reduceComps: Reduce the components of the model.

Description
reduceComps: Reduce the components of the model.

Usage
reduceComps(componentsOld, pointsCopy, biasCopy, datasetName, reducedTerms)

Arguments
componentsOld The old components in the model.
pointsCopy Logical: is the spatial model for the points copy.
biasCopy Logical: is the bias model copy.
datasetName Name of the dataset of interest.
reducedTerms Terms to include in the new components.

Value
New components.

region
sf object containing the boundary region for solitary tinamouc

Description
sf object containing the boundary region for solitary tinamouc

Source
https://github.com/oharar/PointedSDMs
removeFormula

removeFormula: Function to remove term from a formula.

Description

formulaChanger: Internal function used to remove formula terms.

Usage

removeFormula(formulaRemove, oldFormula)

Arguments

formulaRemove The formula with all the components to remove.
oldFormula The formula which needs to change.

runModel

runModel: function used to run the integrated model. Note that this function is depreciated, and will be removed in a later version of the package.

Description

This function takes a intModel object and produces an inlabru model object with additional lists and meta-data added.

Usage

runModel(data, options = list())

Arguments

data A intModel object to be used in the integrated model.
options A list of INLA options used in the model. Defaults to list().

Value

An inlabru model with additional lists containing some more metadata attached.
Examples

```r
## Not run:

if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                          Projection = proj, responsePA = 'Present')

## Run the model
modelRun <- runModel(organizedData,
                      options = list(control.inla = list(int.strategy = 'eb')))

# Print summary of model
modelRun
}

## End(Not run)
```

---

**SetophagaData**

_List of all data objects used for the Setophaga vignette._

---

**Description**

List of all data objects used for the Setophaga vignette.

---

**SolitaryTinamou**

_List of all data objects used for the solitary tinamou vignette._

---

**Description**

List of all data objects used for the solitary tinamou vignette.
SolTinCovariates

**Description**

spatRaster object containing covariate values

**Source**

https://github.com/oharar/PointedSDMs

---

**specifyISDM**

*R6 class for creating a startISDM object.*

**Description**

A data object containing the data and the relevant information about the integrated model. The function `startISDM` acts as a wrapper in creating one of these objects. The output of this object has additional functions within the object which allow for further specification and customization of the integrated model.

**Methods**

**Public methods:**

- `specifyISDM$help()`
- `specifyISDM$print()`
- `specifyISDM$plot()`
- `specifyISDM$addBias()`
- `specifyISDM$updateFormula()`
- `specifyISDM$changeComponents()`
- `specifyISDM$priorsFixed()`
- `specifyISDM$specifySpatial()`
- `specifyISDM$changeLink()`
- `specifyISDM$spatialBlock()`
- `specifyISDM$addSamplers()`
- `specifyISDM$specifyRandom()`
- `specifyISDM$new()`
- `specifyISDM$samplingBias()`

**Method** `help()`: Function to provide documentation for a specifyISDM object.

**Usage:**

```r
specifyISDM$help(...)```

**Arguments:**
...
Not used

**Returns:** Documentation.

**Method print():** Prints the datasets, their data type and the number of observations, as well as the marks and their respective families.

**Usage:**
specifyISDM$print(...)

**Arguments:**
...
Not used.

**Method plot():** Makes a plot of the points surrounded by the boundary of the region where they were collected. The points may either be plotted based on which dataset they come from, or which species group they are part of (if speciesName is non-NULL in `intModel`).

**Usage:**
specifyISDM$plot(datasetNames, Boundary = TRUE, ...)

**Arguments:**

- **datasetNames** Name of the datasets to plot. If this argument is missing, the function will plot all the data available to the model.
- **Boundary** Logical: should a boundary (created using the `Mesh` object) be used in the plot. Defaults to TRUE.
...
Not used.

**Returns:** A ggplot object.

**Examples:**
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    library(ggplot2)
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj

    #Set model up
    organizedData <- startISDM(data, Mesh = mesh,
                              Projection = proj,
                              responsePA = 'Present')

    #Create plot of data
    organizedData$plot()
  }
}
**Method** addBias(): Function used to add additional spatial fields (called bias fields) to a selected dataset present in the integrated model. Bias fields are typically used to account for sampling biases in opportunistic citizen science data in the absence of any covariate to do such.

**Usage:**

```r
specifyISDM$addBias(
  datasetNames = NULL,
  allPO = FALSE,
  biasField = NULL,
  copyModel = TRUE,
  shareModel = FALSE,
  temporalModel = list(model = "ar1")
)
```

**Arguments:**

- `datasetNames` A vector of dataset names (class character) for which a bias field needs to be added to. If NULL (default), then allPO has to be TRUE.
- `allPO` Logical: should a bias field be added to all datasets classified as presence only in the integrated model. Defaults to FALSE.
- `biasField` An `inla.spde` object used to describe the bias field. Defaults to NULL which uses `inla.spde2.matern` to create a Matern model for the field.
- `copyModel` Create copy models for all the of the datasets specified with either `datasetNames` or `allPO`. The first dataset in the vector will have its own spatial effect, and the other datasets will "copy" the effect with shared hyperparameters. Defaults to TRUE.
- `shareModel` Share a bias field across the datasets specified with `datasetNames`. Defaults to FALSE.
- `temporalModel` List of model specifications given to the control.group argument in the time effect component. Defaults to `list(model = 'ar1')`; see `control.group` from the **INLA** package for more details. `temporalName` needs to be specified in `inModel` prior.

**Returns:** A bias field to the model.

**Examples:**

```r
\dontrun{
if (requireNamespace('INLA')) {

  #Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj

  #Set model up
  organizedData <- startISDM(data, Mesh = mesh,
                              Projection = proj,
                              responsePA = 'Present')

  #Add bias field to eBird records
  organizedData$addBias(datasetNames = 'eBird')
}
```

Method `updateFormula()`: Function used to update the formula for a selected observation model. The function is designed to work similarly to the generic update formula, and should be used to thin terms out of a process from the full model specified in `intModel`. The function also allows the user to add their own formula to the model, such that they can include non-linear components in the model. The function can also be used to print out the formula for a process by not specifying the `Formula` or `newFormula` arguments.

**Usage:**

```r
specifyISDM$updateFormula(
  datasetName = NULL,
  Formula,
  newFormula,
  processLevel = FALSE
)
```

**Arguments:**

- `datasetName` Name of the dataset (class character) for which the formula needs to be changed.
- `Formula` An updated formula to give to the process. The syntax provided for the formula in this argument should be identical to the formula specification as in base R. Should be used to thin terms out of a formula but could be used to add terms as well. If adding new terms not specified in `intModel`, remember to add the associated component using `.changeComponents` as well.
- `newFormula` Completely change the formula for a process – primarily used to add non-linear components into the formula. Note: all terms need to be correctly specified here.
- `processLevel` Logical argument: if `TRUE` changes the formulas for all of the processes in a dataset. Defaults to `FALSE`.

**Returns:** An updated formula.

**Examples:**

```r
\dontrun{
  if (requireNamespace('INLA')) {

    # Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file('extdata/SolitaryTinamouCovariates.tif',
      package = "PointedSDMs"))$Forest

    # Set model up
  }
}
```
organizedData <- startISDM(data, Mesh = mesh,  
    spatialCovariates = Forest,  
    Projection = proj,  
    responsePA = 'Present',  
    pointsSpatial = 'individual')

#Remove Forest from eBird  
organizedData$updateFormula(datasetName = 'eBird', Formula = ~ . - Forest)

#Add some scaling to Forest for Parks  
organizedData$updateFormula('Parks', newFormula = ~ I(. + (Forest + 1e-6) * scaling))

#Now add scaling to components  
organizedData$changeComponents(addComponent = 'scaling')

Method changeComponents(): Function to add and specify custom components to model, which are required by inlabru. The main purpose of the function is to re-specify or completely change components already in the model, however the user can also add completely new components to the model as well. In this case, the components need to be added to the correct formulas in the model using the .$updateFormula function. If addComponent and removeComponent are both missing, the function will print out the components to be supplied to inlabru's bru function.

Usage:

specifyISDM$changeComponents(addComponent, removeComponent, print = TRUE)

Arguments:

addComponent Component to add to the integrated model. Note that if the user is re-specifying a component already present in the model, they do not need to remove the old component using removeComponent.

removeComponent Component (or just the name of a component) present in the model which should be removed.

print Logical: should the updated components be printed. Defaults to TRUE.

Returns: An updated components list.

Examples:

\donttrun{

if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(  

```r
system.file(
  'extdata/SolitaryTinamouCovariates.tif',
  package = "PointedSDMs")$Forest

#Set model up
organizedData <- startISDM(data, Mesh = mesh,
    spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present')

#Remove Forest from components
organizedData$changeComponents(removeComponent = 'Forest')

}

}]

Method `priorsFixed()`: Function to change priors for the fixed (and possibly random) effects of the model.

Usage:
```r
specifyISDM$priorsFixed(
  Effect,
  datasetName = NULL,
  mean.linear = 0,
  prec.linear = 0.001
)
```  
Arguments:

- **Effect**: Name of the fixed effect covariate to change the prior for. Can take on 'intercept', which will change the specification for an intercept (specified by one of species or datasetName).
- **datasetName**: Name of the dataset for which the prior of the intercept should change (if fixedEffect = 'intercept'). Defaults to NULL which will change the prior effect of the intercepts for all the datasets in the model.
- **mean.linear**: Mean value for the prior of the fixed effect. Defaults to 0.
- **prec.linear**: Precision value for the prior of the fixed effect. Defaults to 0.001.

Returns: New priors for the fixed effects.

Examples:
```r
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
```
specifyISDM

```
#Specify ISDM
system.file('extdata/SolitaryTinamouCovariates.tif',
package = "PointedSDMs")$Forest

#Set model up
organizedData <- startISDM(data, Mesh = mesh,
spatialCovariates = Forest,
Projection = proj, responsePA = 'Present',
pointsSpatial = 'individual')

#Add prior to Forest
organizedData$priorsFixed(Effect = 'Forest', mean.linear = 2, prec.linear = 0.1)
```

**Method** `specifySpatial()`: Function to specify random fields in the model using penalizing complexity (PC) priors for the parameters.

**Usage:**
```r
specifyISDM$specifySpatial(
  sharedSpatial = FALSE,
  datasetName, 
  Bias, 
  PC = TRUE,
  Remove = FALSE,
  ...
)
```

**Arguments:**
- `sharedSpatial` Logical: specify the shared spatial field in the model. Requires `pointsSpatial == 'shared'` in `intModel`. Defaults to FALSE.
- `datasetName` Name of which of the datasets' spatial fields to be specified. Requires `pointsSpatial = 'individual'` in `intModel`.
- `Bias` Name of the dataset for which the bias field to be specified.
- `PC` Logical: should the Matern model be specified with pc priors. Defaults to TRUE, which uses `inla.spde2.pcmatern` to specify the model; otherwise uses `inla.spde2.matern`.
- `Remove` Logical: should the chosen spatial field be removed. Requires one of `sharedSpatial`, `species`, `mark` or `bias` to be non-missing, which chooses which field to remove.
- `...` Additional arguments used by INLA's `inla.spde2.pcmatern` or `inla.spde2.matern` function, dependent on the value of `PC`.

**Returns:** A new model for the spatial effects.

**Examples:**
```r
\dontrun{
  if (requireNamespace('INLA')) {
    #Get Data
```
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file(
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

#Set model up
organizedData <- startISDM(data, Mesh = mesh,
  spatialCovariates = Forest,
  Projection = proj, responsePA = 'Present')

#Specify the shared spatial field
organizedData$specifySpatial(sharedSpatial = TRUE,
  prior.range = c(1,0.001),
  prior.sigma = c(1,0.001))

Method changeLink(): Function used to change the link function for a given process.

Usage:
specifyISDM$changeLink(datasetName, Link)

Arguments:
datasetName Name of the dataset for which the link function needs to be changed.
Link Name of the link function to add to the process. If missing, will print the link function of
  the specified dataset.

Returns: A new link function for a process.

Examples:
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file(
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

    #Set model up
    organizedData <- startISDM(data, Mesh = mesh,
          spatialCovariates = Forest,
          Projection = proj, responsePA = 'Present')

    #Specify the shared spatial field
    organizedData$specifySpatial(sharedSpatial = TRUE,
          prior.range = c(1,0.001),
          prior.sigma = c(1,0.001))

  }
}
# Set model up
organizedData <- startISDM(data, Mesh = mesh,
spatialCovariates = Forest,
Projection = proj, responsePA = 'Present')

# Specify the shared spatial field
organizedData$changeLink('Parks', 'logit')

Method `spatialBlock()`: Function to spatially block the datasets, which will then be used for model cross-validation with `blockedCV`. See the `spatialBlock` function from `blockCV` for how the spatial blocking works and for further details on the function’s arguments.

Usage:
specifyISDM$spatialBlock(k, rows_cols, plot = FALSE, seed = 1234, ...)

Arguments:
- k: Integer value reflecting the number of folds to use.
- rows_cols: Integer value by which the area is divided into longitudinal and latitudinal bins.
- plot: Plot the cross-validation folds as well as the points across the boundary. Defaults to FALSE.
- seed: Seed used by `blockCV`’s `spatialBlock` to make the spatial blocking reproducible across different models. Defaults to 1234.
- ...: Additional arguments used by `blockCV`’s `spatialBlock`.

Returns: If plot = TRUE, a plot of the grid.

Examples:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(system.file('extdata/SolitaryTinamouCovariates.tif',
package = "PointedSDMs"))$Forest

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
spatialCovariates = Forest,
Projection = proj, responsePA = 'Present',

pointsSpatial = 'individual')

#Specify the spatial block
organizedData$spatialBlock(k = 2, rows = 2, cols = 1, plot = FALSE)

}

Method addSamplers(): Function to add an integration domain for the PO datasets.

Usage:
specifyISDM$addSamplers(datasetName, Samplers)

Arguments:
datasetName Name of the dataset for the samplers.
Samplers A Spatial* object representing the integration domain.

Returns: New samplers for a process.

Method specifyRandom(): Function to specify the models and priors for the random effects included in the model.

Usage:
specifyISDM$specifyRandom(
  temporalModel = list(model = "ar1"),
  copyModel = list(beta = list(fixed = FALSE)),
  copyBias = list(beta = list(fixed = FALSE))
)

Arguments:
temporalModel List of model specifications given to the control.group argument in the time effect component. Defaults to list(model = 'ar1'); see control.group from the INLA package for more details.
copyModel List of model specifications given to the hyper parameters for the "copy" model. Defaults to list(beta = list(fixed = FALSE)).
copyBias List of model specifications given to the hyper parameters for the "copy" bias model. Defaults to list(beta = list(fixed = FALSE)).

Returns: An updated component list.

Method new():

Usage:
specifyISDM$new(
  data,
  projection,
  Inlamesh,
  initialnames,
  responsecounts,
  responsepa,
  pointcovariates,
  trialspa,
Method samplingBias():

Usage:

```r
specifyISDM$samplingBias(datasetName, Samplers)
```

Examples

```r
## Method `specifyISDM$plot`
## ------------------------------------------------

# Not run:
if (requireNamespace('INLA')) {

# Get Data
library(ggplot2)
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
                           Projection = proj,
                           responsePA = 'Present')

# Create plot of data
organizedData$plot()

} # End(Not run)

## Method `specifyISDM$addBias`
## ------------------------------------------------

# Not run:
if (requireNamespace('INLA')) {
```
# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
                           Projection = proj,
                           responsePA = 'Present')

# Add bias field to eBird records
organizedData$addBias(datasetNames = 'eBird')

## End(Not run)
## ------------------------------------------------
## Method `specifyISDM$updateFormula`
## ------------------------------------------------
## Not run:
if (requireNamespace(‘INLA’)) {
  # Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(system.file(‘extdata/SolitaryTinamouCovariates.tif’, package = “PointedSDMs”))$Forest

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
                           spatialCovariates = Forest,
                           Projection = proj,
                           responsePA = 'Present',
                           pointsSpatial = 'individual')

# Remove Forest from eBird
organizedData$updateFormula(datasetName = 'eBird', Formula = ~ . - Forest)

# Add some scaling to Forest for Parks
organizedData$updateFormula('Parks', newFormula = ~ I(. + (Forest + 1e-6) * scaling))

# Now add scaling to components
organizedData$changeComponents(addComponent = 'scaling')
}
## End(Not run)

## Method `specifyISDM$changeComponents`
## ------------------------------------------------

## Not run:
if (requireNamespace('INLA')) {
#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file(  
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

#Set model up
organizedData <- startISDM(data, Mesh = mesh,
  spatialCovariates = Forest,
  Projection = proj, responsePA = 'Present')

#Remove Forest from components
organizedData$changeComponents(addComponent = 'Forest')
}
## End(Not run)

## Method `specifyISDM$priorsFixed`
## ------------------------------------------------

## Not run:
if (requireNamespace('INLA')) {
#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(  

system.file('extdata/SolitaryTinamouCovariates.tif',
package = "PointedSDMs")$Forest

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
spatialCovariates = Forest,
Projection = proj, responsePA = 'Present',
pointsSpatial = 'individual')

# Add prior to Forest
organizedData$priorsFixed(Effect = 'Forest',
mean.linear = 2, prec.linear = 0.1)

}

## End(Not run)

## Method `specifyISDM$specifySpatial`
## ----------------------------------

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
system.file('extdata/SolitaryTinamouCovariates.tif',
package = "PointedSDMs")$Forest)

# Set model up
organizedData <- startISDM(data, Mesh = mesh,
spatialCovariates = Forest,
Projection = proj, responsePA = 'Present')

# Specify the shared spatial field
organizedData$specifySpatial(sharedSpatial = TRUE,
  prior.range = c(1, 0.001),
  prior.sigma = c(1, 0.001))

}

## End(Not run)

## Method `specifyISDM$changeLink`
## Not run:
if (requireNamespace('INLA')) {
  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file( 'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs")$Forest

  # Set model up
  organizedData <- startISDM(data, Mesh = mesh,
    spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present')

  # Specify the shared spatial field
  organizedData$changeLink('Parks', 'logit')
}

## End(Not run)

## Method `specifyISDM$spatialBlock`
## -----------------------------------

if (requireNamespace('INLA')) {
  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file( 'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs")$Forest

  # Set model up
  organizedData <- startISDM(data, Mesh = mesh,
    spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present',
    pointsSpatial = 'individual')

#Specify the spatial block
organizedData$spatialBlock(k = 2, rows = 2, cols = 1, plot = FALSE)
}

## specifyMarks

R6 class for creating a specifyMarks object.

### Description

A data object containing the data and the relevant information about the integrated model. The function `startMarks` acts as a wrapper in creating one of these objects. The output of this object has additional functions within the object which allow for further specification and customization of the integrated model.

### Methods

**Public methods:**

- `specifyMarks$help()`
- `specifyMarks$print()`
- `specifyMarks$plot()`
- `specifyMarks$addBias()`
- `specifyMarks$updateFormula()`
- `specifyMarks$changeComponents()`
- `specifyMarks$priorsFixed()`
- `specifyMarks.specifySpatial()`
- `specifyMarks$changeLink()`
- `specifyMarks$spatialBlock()`
- `specifyMarks$addSamplers()`
- `specifyMarks$specifyRandom()`
- `specifyMarks$new()`
- `specifyMarks$samplingBias()`

**Method help():** Function to provide documentation for a specifyMarks object.

*Usage:*

`specifyMarks$help()`

*Arguments:*

... Not used

*Returns: * Documentation.

**Method print():** Prints the datasets, their data type and the number of observations, as well as the marks and their respective families.
specifyMarks

Usage:
specifyMarks$print(...)

Arguments:
... Not used.

Method plot(): Makes a plot of the points surrounded by the boundary of the region where they were collected. The points may either be plotted based on which dataset they come from, or which species group they are part of (if speciesName is non-NULL in intModel).

Usage:
specifyMarks$plot(datasetNames, Boundary = TRUE, ...)

Arguments:
datasetNames Name of the datasets to plot. If this argument is missing, the function will plot all the data available to the model.
 Boundary Logical: should a boundary (created using the Mesh object) be used in the plot.
   Defaults to TRUE.
... Not used.
Returns: A ggplot object.
Examples:
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    library(ggplot2)
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj

    #Set organizedData up
    organizedData <- startMarks(data, Mesh = mesh,
      Projection = proj, responsePA = 'Present',
      markNames = 'speciesName',
      markFamily = 'multinomial')

    #Create plot of data
    organizedData$plot()
  }
}

Method addBias(): Function used to add additional spatial fields (called bias fields) to a selected dataset present in the integrated model. Bias fields are typically used to account for sampling biases in opportunistic citizen science data in the absence of any covariate to do such.

Usage:
specifyMarks$addBias(
  datasetNames = NULL,
  allPO = FALSE,
  biasField = NULL,
  copyModel = TRUE,
  shareModel = FALSE,
  temporalModel = list(model = "ar1")
)

Arguments:

`datasetNames` A vector of dataset names (class character) for which a bias field needs to be added to. If NULL (default), then `allPO` has to be TRUE.

`allPO` Logical: should a bias field be added to all datasets classified as presence only in the integrated model. Defaults to FALSE.

`biasField` An inla.spde object used to describe the bias field. Defaults to NULL which uses `inla.spde2.matern` to create a Matern model for the field.

`copyModel` Create copy models for all the of the datasets specified with either `datasetNames` or `allPO`. The first dataset in the vector will have its own spatial effect, and the other datasets will "copy" the effect with shared hyperparameters. Defaults to TRUE.

`shareModel` Share a bias field across the datasets specified with `datasetNames`. Defaults to FALSE.

`temporalModel` List of model specifications given to the control.group argument in the time effect component. Defaults to list(model = 'ar1'); see `control.group` from the INLA package for more details. `temporalName` needs to be specified in intModel prior.

Returns: A bias field to the model.

Examples:

```r
if (requireNamespace('INLA')) {
  #Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj

  #Set model up
  organizedData <- startMarks(data, Mesh = mesh,
                              Projection = proj, responsePA = 'Present',
                              markNames = 'speciesName',
                              markFamily = 'multinomial')

  #Add bias field to eBird records
  organizedData$addBias(datasetNames = 'eBird')
}
```
**Method** updateFormula(): Function used to update the formula for a selected observation model. The function is designed to work similarly to the generic update formula, and should be used to thin terms out of a process from the full model specified in `intModel`. The function also allows the user to add their own formula to the model, such that they can include non-linear components in the model. The function can also be used to print out the formula for a process by not specifying the `Formula` or `newFormula` arguments.

**Usage:**

```r
specifyMarks$updateFormula(
  datasetName = NULL,
  Points = TRUE,
  Mark = NULL,
  Formula,
  newFormula
)
```

**Arguments:**

- **datasetName** Name of the dataset (class character) for which the formula needs to be changed.
- **Points** Logical: should the formula be changed for the points (or otherwise, a marked process). Defaults to TRUE.
- **Mark** Name of the mark (class character) to change the formula for. Defaults to NULL.
- **Formula** An updated formula to give to the process. The syntax provided for the formula in this argument should be identical to the formula specification as in base R. Should be used to thin terms out of a formula but could be used to add terms as well. If adding new terms not specified in `intModel`, remember to add the associated component using `.changeComponents` as well.
- **newFormula** Completely change the formula for a process – primarily used to add non-linear components into the formula. Note: all terms need to be correctly specified here.

**Returns:** If `Formula` and `newFormula` are missing, will print out the formula for the specified processes.

**Examples:**

```r
\dontrun{
  if (requireNamespace('INLA')) {
    #Get Data
data("SolitaryTinamou")
    proj <- '+proj=longlat +ellps=WGS84'
data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- SolitaryTinamou$covariates$Forest

    #Set model up
    organizedData <- startMarks(data, Mesh = mesh,
                               Projection = proj, responsePA = 'Present',
                               markNames = 'speciesName',
                               markFamily = 'multinomial')
  }
}
```
Method changeComponents(): Function to add and specify custom components to model, which are required by inlabru. The main purpose of the function is to re-specify or completely change components already in the model, however the user can also add completely new components to the model as well. In this case, the components need to be added to the correct formulas in the model using the .$updateFormula function. If addComponent and removeComponent are both missing, the function will print out the components to be supplied to inlabru’s bru function.

Usage:
specifyMarks$changeComponents(addComponent, removeComponent, print = TRUE)

Arguments:
addComponent Component to add to the integrated model. Note that if the user is re-specifying a component already present in the model, they do not need to remove the old component using removeComponent.
removeComponent Component (or just the name of a component) present in the model which should be removed.
print Logical: should the updated components be printed. Defaults to TRUE.

Examples:
\dontrun{
  if (requireNamespace('INLA')) {

    # Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- SolitaryTinamou$covariates$Forest

    # Set model up
    organizedData <- startMarks(data, Mesh = mesh,
                                Projection = proj, responsePA = 'Present',
                                markNames = 'speciesName',
                                markFamily = 'multinomial')

    # Remove Forest from components
    organizedData$changeComponents(removeComponent = 'Forest')

  }
}
Method \texttt{priorsFixed()}: Function to change priors for the fixed (and possibly random) effects of the model.

Usage:
\begin{verbatim}
specifyMarks$priorsFixed(
  Effect, 
  datasetName = NULL, 
  mean.linear = 0, 
  prec.linear = 0.001
)
\end{verbatim}

Arguments:
\begin{description}
\item[Effect] Name of the fixed effect covariate to change the prior for. Can take on 'intercept', which will change the specification for an intercept (specified by one of \texttt{species} or \texttt{datasetName}).
\item[datasetName] Name of the dataset for which the prior of the intercept should change (if fixed-Effect = 'intercept'). Defaults to NULL which will change the prior effect of the intercepts for all the datasets in the model.
\item[mean.linear] Mean value for the prior of the fixed effect. Defaults to 0.
\item[prec.linear] Precision value for the prior of the fixed effect. Defaults to 0.001.
\end{description}

Returns: New priors for the fixed effects.

Examples:
\begin{verbatim}
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file(
        'extdata/SolitaryTinamouCovariates.tif',
        package = "PointedSDMs")$Forest

    #Set model up
    organizedData <- startMarks(data, Mesh = mesh, marksIntercept = FALSE,
      Projection = proj, responsePA = 'Present',
      markNames = 'speciesName',
      markFamily = 'multinomial')

    #Add prior to Forest
    organizedData$priorsFixed(Effect = 'Intercept', mean.linear = 2, prec.linear = 0.1)
  }
}
\end{verbatim}
**Method** `specifySpatial()`: Function to specify random fields in the model using penalizing complexity (PC) priors for the parameters.

**Usage:**
```r
specifyMarks$ specifySpatial(
    sharedSpatial = FALSE,
    datasetName,
    Mark,
    Bias,
    PC = TRUE,
    Remove = FALSE,
    ...
)
```

**Arguments:**
- `sharedSpatial` Logical: specify the shared spatial field in the model. Requires `pointsSpatial == 'shared'` in `intModel`. Defaults to FALSE.
- `datasetName` Name of which of the datasets’ spatial fields to be specified. Requires `pointsSpatial == 'individual'` in `intModel`.
- `Mark` Name of the marks to specify the spatial field for. If TRUE changes the spatial effect for all marks.
- `Bias` Name of the dataset for which the bias field to be specified.
- `PC` Logical: should the Matern model be specified with pc priors. Defaults to TRUE, which uses `inla.spde2.pcmatern` to specify the model; otherwise uses `inla.spde2.matern`.
- `Remove` Logical: should the chosen spatial field be removed. Requires one of `sharedSpatial`, `species`, `mark` or `bias` to be non-missing, which chooses which field to remove.
- `...` Additional arguments used by INLA’s `inla.spde2.pcmatern` or `inla.spde2.matern` function, dependent on the value of `PC`.

**Returns:** A new model for the spatial effects.

**Examples:**
```r
\dontrun{
  if (requireNamespace('INLA')) {

    # Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file(
        'extdata/SolitaryTinamouCovariates.tif',
        package = "PointedSDMs"))$Forest

    # Set model up
    organizedData <- startMarks(data, Mesh = mesh,
                                Projection = proj, responsePA = 'Present',
                                ...)
markNames = 'speciesName',
markFamily = 'multinomial')

#Specify the shared spatial field
organizedData$specifySpatial(sharedSpatial = TRUE,
   prior.range = c(1,0.001),
   prior.sigma = c(1,0.001))

Method \texttt{changeLink()}: Function used to change the link function for a given process.

\textbf{Usage:}
\texttt{specifyMarks$changeLink(datasetName, Mark, Link, ...)}

\textbf{Arguments:}
\texttt{datasetName} Name of the dataset for which the link function needs to be changed.
\texttt{Mark} Name of the mark for which the link function needs to be changed.
\texttt{Link} Name of the link function to add to the process. If missing, will print the link function of
   the specified dataset.
\texttt{...} Not used
\texttt{Species} Name of the species for which the link function needs to be changed.

\textbf{Examples:}
\texttt{\dontrun{
if (requireNamespace('INLA')) {

#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
   system.file(
      'extdata/SolitaryTinamouCovariates.tif',
      package = "PointedSDMs"))$Forest

#Set model up
organizedData <- startMarks(data, Mesh = mesh,
   Projection = proj, responsePA = 'Present',
   markNames = 'speciesName',
   markFamily = 'multinomial')

#Specify the shared spatial field
organizedData$changeLink(datasetName = 'Parks',
   Mark = 'speciesName',
   Link = 'logit')
}}
Method **spatialBlock()**: Function to spatially block the datasets, which will then be used for model cross-validation with `blockCV`. See the `spatialBlock` function from `blockCV` for how the spatial blocking works and for further details on the function’s arguments.

**Usage:**
```r
specifyMarks$spatialBlock(k, rows_cols, plot = FALSE, seed = 1234, ...)
```

**Arguments:**
- `k` Integer value reflecting the number of folds to use.
- `rows_cols` Integer value by which the area is divided into longitudinal and latitudinal bins.
- `plot` Plot the cross-validation folds as well as the points across the boundary. Defaults to `FALSE`.
- `seed` Seed used by `blockCV`’s `spatialBlock` to make the spatial blocking reproducible across different models. Defaults to 1234.
- `...` Additional arguments used by `blockCV`'s `spatialBlock`.

**Examples:**
```r
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- SolitaryTinamou$covariates$Forest

    #Set model up
    organizedData <- startMarks(data, Mesh = mesh,
                                Projection = proj, responsePA = 'Present',
                                markNames = 'speciesName',
                                markFamily = 'multinomial')

    #Specify the spatial block
    organizedData$spatialBlock(k = 2, rows = 2, cols = 1, plot = FALSE)

  }
}
```

Method **addSamplers()**: Function to add an integration domain for the PO datasets.

**Usage:**
```r
specifyMarks$addSamplers(datasetName, Samplers)
```
**specifyMarks**

Arguments:

- **datasetName**  Name of the dataset for the samplers.
- **samplers** A Spatial object representing the integration domain.

Method `specifyRandom()`: Function to specify the models and priors for the random effects included in the model.

Usage:

```r
specifyMarks$specifyRandom(
    temporalModel = list(model = "ar1"),
    copyModel = list(beta = list(fixed = FALSE)),
    copyBias = list(beta = list(fixed = FALSE))
)
```

Arguments:

- **temporalModel** List of model specifications given to the control.group argument in the time effect component. Defaults to `list(model = 'ar1')`; see `control.group` from the **INLA** package for more details.
- **copyModel** List of model specifications given to the hyper parameters for the "copy" model. Defaults to `list(beta = list(fixed = FALSE))`.
- **copyBias** List of model specifications given to the hyper parameters for the "copy" bias model. Defaults to `list(beta = list(fixed = FALSE))`.

Returns: An updated component list.

Method `new()`:

Usage:

```r
specifyMarks$new(
    data,
    coordinates,
    projection,
    Inlamesh,
    initialnames,
    responsecounts,
    responsepa,
    marksnames,
    marksfamily,
    pointcovariates,
    trialspa,
    trialsmarks,
    marksspatial,
    spatial,
    intercepts,
    spatialcovariates,
    marksintercepts,
    boundary,
    ips,
    temporal,
    temporalmodel,
)```
Method `samplingBias()`:

Usage:
```
specifyMarks$samplingBias(datasetName, Samplers)
```

Examples

```r
## Method `specifyMarks$plot`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
library(ggplot2)
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

# Set organizedData up
organizedData <- startMarks(data, Mesh = mesh,
                            Projection = proj, responsePA = 'Present',
                            markNames = 'speciesName',
                            markFamily = 'multinomial')

# Create plot of data
organizedData$plot()
}

## Method `specifyMarks$addBias`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
```
```r
mesh$crs <- proj

# Set model up
organizedData <- startMarks(data, Mesh = mesh,
                              Projection = proj, responsePA = 'Present',
                              markNames = 'speciesName',
                              markFamily = 'multinomial')

# Add bias field to eBird records
organizedData$addBias(datasetNames = 'eBird')
}

## End (Not run)

## Method `specifyMarks$updateFormula`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- SolitaryTinamou$covariates$Forest

# Set model up
organizedData <- startMarks(data, Mesh = mesh,
                              Projection = proj, responsePA = 'Present',
                              markNames = 'speciesName',
                              markFamily = 'multinomial')

# Remove Forest from eBird
organizedData$updateFormula(datasetName = 'eBird', Mark = 'speciesName', Formula = ~ . - Forest)
}

## End (Not run)

## Method `specifyMarks$changeComponents`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
```
```r
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- SolitaryTinamou$covariates$Forest

# Set model up
organizedData <- startMarks(data, Mesh = mesh,
    Projection = proj, responsePA = 'Present',
    markNames = 'speciesName',
    markFamily = 'multinomial')

# Remove Forest from components
organizedData$changeComponents(removeComponent = 'Forest')
}

## End(Not run)

## Method `specifyMarks$priorsFixed`
## ------------------------------------------

## Not run:
if (requireNamespace('INLA')) {
    # Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(system.file(
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

    # Set model up
    organizedData <- startMarks(data, Mesh = mesh, marksIntercept = FALSE,
        Projection = proj, responsePA = 'Present',
        markNames = 'speciesName',
        markFamily = 'multinomial')

    # Add prior to Forest
    organizedData$priorsFixed(Effect = 'Intercept', mean.linear = 2, prec.linear = 0.1)
}

## End(Not run)
```
### Method `specifyMarks$specifySpatial`

```
## Not run:
if (requireNamespace('INLA')) {
  
  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file(  
      'extdata/SolitaryTinamouCovariates.tif',
      package = "PointedSDMs")$Forest
  )

  # Set model up
  organizedData <- startMarks(data, Mesh = mesh,  
    Projection = proj, responsePA = 'Present',
    markNames = 'speciesName',
    markFamily = 'multinomial')

  # Specify the shared spatial field
  organizedData$specifySpatial(sharedSpatial = TRUE,  
    prior.range = c(1,0.001),
    prior.sigma = c(1,0.001))
}
```

### Method `specifyMarks$changeLink`

```
## Not run:
if (requireNamespace('INLA')) {
  
  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file(  
      'extdata/SolitaryTinamouCovariates.tif',
      package = "PointedSDMs")$Forest
  )

  # Set model up
  organizedData <- startMarks(data, Mesh = mesh,  
    Projection = proj, responsePA = 'Present',
    markNames = 'speciesName',
    markFamily = 'multinomial')
```
#Set model up
organizedData <- startMarks(data, Mesh = mesh,
                                           Projection = proj, responsePA = 'Present',
                                           markNames = 'speciesName',
                                           markFamily = 'multinomial')

#Specify the shared spatial field
organizedData$changeLink(datasetName = 'Parks',
                                           Mark = 'speciesName',
                                           Link = 'logit')

}

## End(Not run)

## Method `specifyMarks$spatialBlock`

## Not run:
if (requireNamespace('INLA')) {

#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- SolitaryTinamou$covariates$Forest

#Set model up
organizedData <- startMarks(data, Mesh = mesh,
                                           Projection = proj, responsePA = 'Present',
                                           markNames = 'speciesName',
                                           markFamily = 'multinomial')

#Specify the spatial block
organizedData$spatialBlock(k = 2, rows = 2, cols = 1, plot = FALSE)
}

## End(Not run)

---

**specifySpecies**  
*R6 class for creating a startSpecies object.*
Description

A data object containing the data and the relevant information about the integrated model. The function `startSpecies` acts as a wrapper in creating one of these objects. The output of this object has additional functions within the object which allow for further specification and customization of the integrated model.

Methods

Public methods:

- `specifySpecies$help()`
- `specifySpecies$print()`
- `specifySpecies$plot()`
- `specifySpecies$addBias()`
- `specifySpecies$updateFormula()`
- `specifySpecies$changeComponents()`
- `specifySpecies$priorsFixed()`
- `specifySpecies$specifySpatial()`
- `specifySpecies$changeLink()`
- `specifySpecies$spatialBlock()`
- `specifySpecies$addSamplers()`
- `specifySpecies$specifyRandom()`
- `specifySpecies$new()`
- `specifySpecies$samplingBias()`

Method `help()`: Function to provide documentation for a `specifySpecies` object.

Usage:
```r
specifySpecies$help(...)  
```

Arguments:

... Not used

Returns: Documentation.

Method `print()`: Prints the datasets, their data type and the number of observations, as well as the marks and their respective families.

Usage:
```r
specifySpecies$print(...)  
```

Arguments:

... Not used.

Method `plot()`: Makes a plot of the points surrounded by the boundary of the region where they were collected.

Usage:
```r
specifySpecies$plot(datasetNames, Species = TRUE, Boundary = TRUE, ...)  
```

Arguments:
datasetNames  Name of the datasets to plot. If this argument is missing, the function will plot all the data available to the model.
Species  Logical: should the points be plotted based on the species name. Defaults to TRUE.
Boundary  Logical: should a boundary (created using the Mesh object) be used in the plot. Defaults to TRUE.
...  Not used.

Returns: A ggplot object.

Examples:
\dontrun{
  if (requireNamespace('INLA')) {
    #Get Data
    library(ggplot2)
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj

    #Set model up
    organizedData <- startSpecies(data, Mesh = mesh,
                               speciesName = 'speciesName',
                               Projection = proj,
                               responsePA = 'Present')

    #Create plot of data
    organizedData$plot()
  }
}

Method addBias(): Function used to add additional spatial fields (called bias fields) to a selected dataset present in the integrated model. Bias fields are typically used to account for sampling biases in opportunistic citizen science data in the absence of any covariate to do such.

Usage:
specifySpecies$addBias(
  datasetNames = NULL,
  allPO = FALSE,
  biasField = NULL,
  copyModel = TRUE,
  shareModel = FALSE,
  temporalModel = list(model = "ar1")
)

Arguments:
datasetNames  A vector of dataset names (class character) for which a bias field needs to be added to. If NULL (default), then allPO has to be TRUE.
allPO Logical: should a bias field be added to all datasets classified as presence only in the integrated model. Defaults to FALSE.

biasField An inla.spde object used to describe the bias field. Defaults to NULL which uses inla.spde2.matern to create a Matern model for the field.

copyModel Create copy models for all the of the datasets specified with either datasetNames or allPO. The first dataset in the vector will have its own spatial effect, and the other datasets will "copy" the effect with shared hyperparameters. Defaults to TRUE.

shareModel Share a bias field across the datasets specified with datasetNames. Defaults to FALSE.

temporalModel List of model specifications given to the control.group argument in the time effect component. Defaults to list(model = 'ar1'); see control.group from the INLA package for more details. temporalName needs to be specified in intModel prior.

Returns: A bias field to the model.

Examples:
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj

    #Set model up
    organizedData <- startSpecies(data, Mesh = mesh,
                                  speciesName = 'speciesName',
                                  Projection = proj,
                                  responsePA = 'Present')

    #Add bias field to eBird records
    organizedData$addBias(datasetNames = 'eBird')
  }
}

Method updateFormula(): Function used to update the formula for a selected observation model. The function is designed to work similarly to the generic update formula, and should be used to thin terms out of a process from the full model specified in intModel. The function also allows the user to add their own formula to the model, such that they can include non-linear components in the model. The function can also be used to print out the formula for a process by not specifying the Formula or newFormula arguments.

Usage:
specifySpecies$updateFormula(
  datasetName = NULL,
  speciesName = NULL,
  Formula,
specifySpecies

processLevel = FALSE,
newFormula
)

Arguments:

- **datasetName**: Name of the dataset (class character) for which the formula needs to be changed.
- **speciesName**: Name of the species for which to change a formula for. Defaults to NULL which changes the formula for all species present in datasetName.
- **Formula**: An updated formula to give to the process. The syntax provided for the formula in this argument should be identical to the formula specification as in base R. Should be used to thin terms out of a formula but could be used to add terms as well. If adding new terms not specified in intModel, remember to add the associated component using \$changeComponents as well.
- **processLevel**: Logical argument: if TRUE changes the formulas for all of the processes in a dataset. Defaults to FALSE.
- **newFormula**: Completely change the formula for a process – primarily used to add non-linear components into the formula. Note: all terms need to be correctly specified here.

Returns: An updated formula.

Examples:

```r
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file(
        'extdata/SolitaryTinamouCovariates.tif',
        package = "PointedSDMs"))$Forest

    #Set model up
    organizedData <- startSpecies(data, Mesh = mesh, speciesName = 'speciesName',
                                   spatialCovariates = Forest,
                                   Projection = proj, responsePA = 'Present',
                                   pointsSpatial = 'individual')

    #Remove Forest from eBird
    organizedData$updateFormula(datasetName = 'eBird', Formula = ~ . - Forest)

    #Add some scaling to Forest for Parks
    organizedData$updateFormula(datasetName = 'Parks', newFormula = ~ I(. + (Forest+1e-6)*scaling))

    #Now add scaling to components
    organizedData$changeComponents(addComponent = 'scaling')
  }
}
```


**Method** changeComponents(): Function to add and specify custom components to model, which are required by *inlabru*. The main purpose of the function is to re-specify or completely change components already in the model, however the user can also add completely new components to the model as well. In this case, the components need to be added to the correct formulas in the model using the `$updateFormula` function. If `addComponent` and `removeComponent` are both missing, the function will print out the components to be supplied to *inlabru*’s bru function.

**Usage:**
specifySpecies$changeComponents(addComponent, removeComponent, print = TRUE)

**Arguments:**
- `addComponent` Component to add to the integrated model. Note that if the user is re-specifying a component already present in the model, they do not need to remove the old component using `removeComponent`.
- `removeComponent` Component (or just the name of a component) present in the model which should be removed.
- `print` Logical: should the updated components be printed. Defaults to TRUE.

**Returns:** An updated components list.

**Examples:**

```r
dontrun{
  if (requireNamespace('INLA')) {

    # Get Data
data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file('extdata/SolitaryTinamouCovariates.tif',
        package = "PointedSDMs"))$Forest

    # Set model up
    organizedData <- startSpecies(data, Mesh = mesh,
      speciesName = 'speciesName',
      spatialCovariates = Forest,
      Projection = proj,
      responsePA = 'Present')

    # Remove Forest from components
    organizedData$changeComponents(removeComponent = 'speciesSpatial')
  }
}
```
Method priorsFixed(): Function to change priors for the fixed (and possibly random) effects of the model.

Usage:
```r
specifySpecies$priorsFixed(
  Effect,
  Species = NULL,
  datasetName = NULL,
  mean.linear = 0,
  prec.linear = 0.001
)
```

Arguments:
- **Effect**: Name of the fixed effect covariate to change the prior for. Can take on 'intercept', which will change the specification for an intercept (specified by one of species or datasetName).
- **Species**: Name of the species (class character) for which the prior should change. Defaults to NULL which will change the prior for all species added to the model.
- **datasetName**: Name of the dataset for which the prior of the intercept should change (if fixed-Effect = 'intercept'). Defaults to NULL which will change the prior effect of the intercepts for all the datasets in the model.
- **mean.linear**: Mean value for the prior of the fixed effect. Defaults to 0.
- **prec.linear**: Precision value for the prior of the fixed effect. Defaults to 0.001.

Returns: New priors for the fixed effects.

Examples:
```r
\dontrun{
  if (requireNamespace('INLA')) {
    #Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file('extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

    #Set model up
organizedData <- startSpecies(data, Mesh = mesh,
    speciesName = 'speciesName',
    spatialCovariates = Forest,
  )
}
```
Projection = proj, responsePA = 'Present',
pointsSpatial = 'individual')

#Add prior to Forest
organizedData$priorsFixed(Effect = 'Forest', mean.linear = 2, prec.linear = 0.1)

Method specifySpatial(): Function to specify random fields in the model using penalizing complexity (PC) priors for the parameters.

Usage:
specifySpecies$specifySpatial(
  sharedSpatial = FALSE,
  datasetName, 
  Species,
  Bias, 
  PC = TRUE,
  Remove = FALSE,
  ...
)

Arguments:
sharedSpatial Logical: specify the shared spatial field in the model. Requires pointsSpatial == 'shared' in intModel. Defaults to FALSE.
datasetName Name of which of the datasets’ spatial fields to be specified. Requires pointsSpatial = 'individual' in intModel.
Species Name of the species to change the spatial effect for. If TRUE then changes the spatial effect for the shared species field.
Bias Name of the dataset for which the bias field to be specified.
PC Logical: should the Matern model be specified with pc priors. Defaults to TRUE, which uses inla.spde2.pcmatern to specify the model; otherwise uses inla.spde2.matern.
Remove Logical: should the chosen spatial field be removed. Requires one of sharedSpatial, species, mark or bias to be non-missing, which chooses which field to remove.
... Additional arguments used by INLA’s inla.spde2.pcmatern or inla.spde2.matern function, dependent on the value of PC.

Returns: A new model for the spatial effects.

Examples:
\dontrun{
  if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
  
  
...}
Forest <- terra::rast(
  system.file('extdata/SolitaryTinamouCovariates.tif',
               package = "PointedSDMs")$Forest
)

#Set model up
organizedData <- startSpecies(data, Mesh = mesh,
                               speciesName = 'speciesName',
                               spatialCovariates = Forest,
                               Projection = proj, responsePA = 'Present')

#Specify the shared spatial field
organizedData$specifySpatial(sharedSpatial = TRUE, PC = TRUE,
                              prior.range = c(1,0.001),
                              prior.sigma = c(1,0.001))

Method changeLink(): Function used to change the link function for a given process.

Usage:
specifySpecies$changeLink(datasetName, Link, ...)

Arguments:
datasetName Name of the dataset for which the link function needs to be changed.
Link Name of the link function to add to the process. If missing, will print the link function of
       the specified dataset.

Returns: A new link function for a process.

Examples:
\donttrun{
  if (requireNamespace('INLA')) {
    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
      system.file('extdata/SolitaryTinamouCovariates.tif',
                   package = "PointedSDMs")$Forest
    )
    #Set model up
    organizedData <- startSpecies(data, Mesh = mesh,
                                   speciesName = 'speciesName',
                                   spatialCovariates = Forest,
                                   Projection = proj, responsePA = 'Present')

    #Specify the shared spatial field
    organizedData$specifySpatial(sharedSpatial = TRUE, PC = TRUE,
                                  prior.range = c(1,0.001),
                                  prior.sigma = c(1,0.001))
}
}
spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present')

    #Specify the shared spatial field
    organizedData$changeLink('Parks', 'logit')

} }

Method spatialBlock(): Function to spatially block the datasets, which will then be used for model cross-validation with blockedCV. See the spatialBlock function from blockCV for how the spatial blocking works and for further details on the function's arguments.

Usage:
    specifySpecies$spatialBlock(k, rows_cols, plot = FALSE, seed = 1234, ...)

Arguments:
    k  Integer value reflecting the number of folds to use.
    rows_cols  Integer value by which the area is divided into longitudinal and latitudinal bins.
    plot  Plot the cross-validation folds as well as the points across the boundary. Defaults to FALSE.
    seed  Seed used by blockCV’s spatialBlock to make the spatial blocking reproducible across different models. Defaults to 1234.
    ...  Additional arguments used by blockCV’s spatialBlock.

Returns:  If plot = TRUE, a plot of the grid.

Examples:
    \dontrun{
    if (requireNamespace('INLA')) {

    #Get Data
    data("SolitaryTinamou")
    proj <- "+proj=longlat +ellps=WGS84"
    data <- SolitaryTinamou$datasets
    mesh <- SolitaryTinamou$mesh
    mesh$crs <- proj
    Forest <- terra::rast(
        system.file( 'extdata/SolitaryTinamouCovariates.tif',
            package = "PointedSDMs"))$Forest

    #Set model up
    organizedData <- startSpecies(data, Mesh = mesh,
        speciesName = 'speciesName',
        spatialCovariates = Forest,
        Projection = proj, responsePA = 'Present',
        pointsSpatial = 'individual')

} }
Method addSamplers(): Function to add an integration domain for the PO datasets.

Usage:
```r
specifySpecies$addSamplers(datasetName, Samplers)
```

Arguments:
- **datasetName**: Name of the dataset for the samplers.
- **Samplers**: A `Spatial*` object representing the integration domain.

Returns: New samplers for a process.

Method specifyRandom(): Function to specify the models and priors for the random effects included in the model.

Usage:
```r
specifySpecies$specifyRandom(
  temporalModel = list(model = "ar1"),
  copyModel = list(beta = list(fixed = FALSE)),
  copyBias = list(b = list(fixed = FALSE)),
  speciesCopy = list(beta = list(fixed = FALSE)),
  speciesIntercepts = list(prior = "loggamma", param = c(1, 5e-05)),
  speciesGroup = list(model = "iid", hyper = list(prec = list(prior = "loggamma", param = c(1, 5e-05))))
)
```

Arguments:
- **temporalModel**: List of model specifications given to the control.group argument in the time effect component. Defaults to `list(model = 'ar1')`; see `control.group` from the `INLA` package for more details.
- **copyModel**: List of model specifications given to the hyper parameters for the "copy" model. Defaults to `list(beta = list(fixed = FALSE))`.
- **copyBias**: List of model specifications given to the hyper parameters for the "copy" bias model. Defaults to `list(b = list(fixed = FALSE))`.
- **speciesCopy**: List of model specifications given to the hyper parameters for the species "copy" model. Defaults to `list(beta = list(fixed = FALSE))`.
- **speciesIntercepts**: Prior distribution for precision parameter for the random species intercept term. Defaults to INLA’s default choice.
- **speciesGroup**: Prior distribution for the precision parameter for the iid group model. Defaults to INLA’s default.

Method new():

Usage:
specifySpecies$new(
    data, projection, Inlamesh, initialnames, responsecounts, responsepa, pointcovariates, speciesintercept, trialspa, spatial, intercepts, spatialcovariates, boundary, ips, temporal, temporalmodel, offset, copymodel, formulas, speciesindependent, speciesname, speciesenvironment, speciesspatial
)

**Method** `samplingBias()`:

**Usage:**

```r
specifySpecies$samplingBias(datasetName, Samplers)
```

**Examples**

```r
## Method `specifySpecies$plot`

```
specifySpecies

Projection = proj,
responsePA = 'Present')

#Create plot of data
organizedData$plot()

}# End(Not run)

## Method `specifySpecies$addBias'

## Not run:
if (requireNamespace('INLA')) {

#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

#Set model up
organizedData <- startSpecies(data, Mesh = mesh,
       speciesName = 'speciesName',
       Projection = proj,
       responsePA = 'Present')

#Add bias field to eBird records
organizedData$addBias(datasetNames = 'eBird')

}# End(Not run)

## Method `specifySpecies$updateFormula'

## Not run:
if (requireNamespace('INLA')) {

#Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
       system.file('extdata/SolitaryTinamouCovariates.tif',


SpecifySpecies

package = "PointedSDMs")$Forest

# Set model up
organizedData <- startSpecies(data, Mesh = mesh, speciesName = 'speciesName',
                                spatialCovariates = Forest,
                                Projection = proj, responsePA = 'Present',
                                pointsSpatial = 'individual')

# Remove Forest from eBird
organizedData$updateFormula(datasetName = 'eBird', Formula = ~ . - Forest)

# Add some scaling to Forest for Parks
organizedData$updateFormula(datasetName = 'Parks', newFormula = ~ I(. + (Forest + 1e-6) * scaling))

# Now add scaling to components
organizedData$changeComponents(addComponent = 'scaling')

} # End(Not run)

## Method `specifySpecies$changeComponents`

## Not run:

if (requireNamespace('INLA')) {

  # Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file( 'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs")$Forest

  # Set model up
  organizedData <- startSpecies(data, Mesh = mesh,
                                speciesName = 'speciesName',
                                spatialCovariates = Forest,
                                Projection = proj, responsePA = 'Present')

  # Remove Forest from components
  organizedData$changeComponents(addComponent = 'scaling')

}
## End(Not run)

## Method `specifySpecies$priorsFixed`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file(
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs"))$Forest

# Set model up
organizedData <- startSpecies(data, Mesh = mesh,
   speciesName = 'speciesName',
   spatialCovariates = Forest,
   Projection = proj, responsePA = 'Present',
   pointsSpatial = 'individual')

# Add prior to Forest
organizedData$priorsFixed(Effect = 'Forest', mean.linear = 2, prec.linear = 0.1)
}

## End(Not run)

## Method `specifySpecies$specifySpatial`

## Not run:
if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- terra::rast(
  system.file(
    'extdata/SolitaryTinamouCovariates.tif',
    package = "PointedSDMs")$Forest,
specifySpecies

package = "PointedSDMs")$Forest

# Set model up
organizedData <- startSpecies(data, Mesh = mesh,
    speciesName = 'speciesName',
    spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present')

# Specify the shared spatial field
organizedData$specifySpatial(sharedSpatial = TRUE, PC = TRUE,
    prior.range = c(1,0.001),
    prior.sigma = c(1,0.001))

}
```
## Not run:
if (requireNamespace('INLA')) {
  #Get Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
  Forest <- terra::rast(
    system.file('extdata/SolitaryTinamouCovariates.tif',
      package = "PointedSDMs"))$Forest

  #Set model up
  organizedData <- startSpecies(data, Mesh = mesh,
    speciesName = 'speciesName',
    spatialCovariates = Forest,
    Projection = proj, responsePA = 'Present',
    pointsSpatial = 'individual')

  #Specify the spatial block
  organizedData$spatialBlock(k = 2, rows = 2, cols = 1, plot = FALSE)
}
## End(Not run)
```

---

**startISDM**  
*Function used to initialize the integrated species distribution model.*

**Description**

This function is used to create an object containing all the data, metadata and relevant components required for the integrated species distribution model and **INLA** to work. As a result, the arguments associated with this function are predominantly related to describing variable names within the datasets that are relevant, and arguments related to what terms should be included in the formula for the integrated model. The output of this function is an R6 object, and so there are a variety of public methods within the output of this function which can be used to further specify the model (see **?specifyISDM** or **$help()** for a comprehensive description of these public methods).

**Usage**

```r
startISDM(
  ...,
  spatialCovariates = NULL,
)```
startISDM

```r
Projection,
Mesh,
IPS = NULL,
Boundary = NULL,
pointCovariates = NULL,
Offset = NULL,
pointsIntercept = TRUE,
pointsSpatial = "copy",
responseCounts = "counts",
responsePA = "present",
trialsPA = NULL,
temporalName = NULL,
Formulas = list(covariateFormula = NULL, biasFormula = NULL)
)
```

**Arguments**

... The datasets to be used in the model. Must come as either sf objects, or as a list of named sf objects.

`spatialCovariates`  
The spatial covariates used in the model. These covariates must be measured at every location (pixel) in the study area, and must be a SpatialRaster object. Can be either numeric, factor or character data. Defaults to NULL which includes no spatial effects in the model.

`Projection`  
The coordinate reference system used by both the spatial points and spatial covariates. Must be of class character.

`Mesh`  
An inla.mesh object required for the spatial random fields and the integration points in the model (see inla.mesh.2d from the INLA package for more details).

`IPS`  
The integration points to be used in the model (that is, the points on the map where the intensity of the model is calculated). See fm_int from the inlabru package for more details regarding these points; however defaults to NULL which will create integration points from the Mesh and Boundary objects.

`Boundary`  
A sf object of the study area. If not missing, this object is used to help create the integration points.

`pointCovariates`  
The non-spatial covariates to be included in the integrated model (for example, in the field of ecology the distance to the nearest road or time spent sampling could be considered). These covariates must be included in the same data object as the points, and do not necessarily need to be present in all datasets.

`Offset`  
Name of the offset variable (class character) in the datasets. Defaults to NULL; if the argument is non-NULL, the variable name needs to be standardized across datasets (but does not need to be included in all datasets). The offset variable will be transformed onto the log-scale in the integrated model.

`pointsIntercept`  
Logical argument: should the points be modeled with intercepts. Defaults to TRUE.
 pointsSpatial  Argument to determine whether the spatial field is shared between the datasets, or if each dataset has its own unique spatial field. The datasets may share a spatial field with INLA's "copy" feature if the argument is set to "copy". May take on the values: "shared", "individual", "copy" or NULL if no spatial field is required for the model. Defaults to "copy".

 responseCounts  Name of the response variable in the counts/abundance datasets. This variable name needs to be standardized across all counts datasets used in the integrated model. Defaults to 'counts'.

 responsePA  Name of the response variable (class character) in the presence absence/detection non-detection datasets. This variable name needs to be standardized across all present absence datasets. Defaults to 'present'.

 trialsPA  Name of the trials response variable (class character) for the presence absence datasets. Defaults to NULL.

 temporalName  Name of the temporal variable (class character) in the model. This variable is required to be in all the datasets. Defaults to NULL.

 Formulas  A named list with two objects. The first one, covariateFormula, is a formula for the covariates and their transformations for the distribution part of the model. Defaults to NULL which includes all covariates specified in spatialCovariates into the model. The second, biasFormula, specifies which covariates are used for the PO datasets. Defaults to NULL which includes no covariates for the PO datasets.

 Value

 A specifyISDM object (class R6). Use ?specifyISDM to get a comprehensive description of the slot functions associated with this object.

 Note

 The idea with this function is to describe the full model: that is, all the covariates and spatial effects will appear in all the formulas for the datasets and species. If some of these terms should not be included in certain observation models in the integrated model, they can be thinned out using the .$updateFormula function. Note: the point covariate will only be included in the formulas for where they are present in a given dataset, and so these terms do not need to be thinned out if they are not required by certain observation models.

 Examples

 ## Not run:
 if (requireNamespace('INLA')) {

 #Get Data
data("SolitaryTinamou")
proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

 #Set base model up
baseModel <- startISDM(data, Mesh = mesh, 
                      Projection = proj, responsePA = 'Present')

# Print summary
baseModel

# Set up model with dataset specific spatial fields
indSpat <- startISDM(data, Mesh = mesh, 
                      Projection = proj, pointsSpatial = 'individual', responsePA = 'Present')

# Model with offset variable
offSet <- startISDM(data, Mesh = mesh, 
                      Projection = proj, Offset = 'area', responsePA = 'Present')

# Add own formula
quadModel <- startISDM(data, Mesh = mesh, 
                        Projection = proj, pointsSpatial = 'copy', responsePA = 'Present', 
                        Formulas = list(covariateFormula = ~ NPP + I(NPP^2)))

})

## End(Not run)
pointCovariates = NULL,
pointsIntercept = TRUE,
marksIntercept = TRUE,
Offset = NULL,
pointsSpatial = "copy",
responseCounts = "counts",
responsePA = "present",
trialsPA = NULL,
trialsMarks = NULL,
temporalName = NULL,
Formulas = list(covariateFormula = NULL, biasFormula = NULL)
)

Arguments

The datasets to be used in the model. Must come as either sf objects, or as a list of named sf objects.

spatialCovariates
The spatial covariates used in the model. These covariates must be measured at every location (pixel) in the study area, and must be a SpatialRaster object. Can be either numeric, factor or character data. Defaults to NULL which includes no spatial effects in the model.

Projection
The coordinate reference system used by both the spatial points and spatial covariates. Must be of class character.

Mesh
An inla.mesh object required for the spatial random fields and the integration points in the model (see inla.mesh.2d from the INLA package for more details).

IPS
The integration points to be used in the model (that is, the points on the map where the intensity of the model is calculated). See fm_int from the inlabru package for more details regarding these points; however defaults to NULL which will create integration points from the Mesh and Boundary objects.

Boundary
A sf object of the study area. If not missing, this object is used to help create the integration points.

markNames
A vector with the mark names (class character) to be included in the integrated model. Marks are variables which are used to describe the individual points in the model (for example, in the field of ecology the size of the species or its feeding type could be considered). Defaults to NULL, however if this argument is non-NULL, the model run will become a marked point process. The marks must be included in the same data object as the points.

markFamily
A vector with the statistical families (class character) assumed for the marks. Must be the same length as markNames, and the position of the mark in the vector markName is associated with the position of the family in markFamily. Defaults to NULL which assigns each mark as "Gaussian".

marksSpatial
Logical argument: should the marks have their own spatial field. Defaults to TRUE.
**pointCovariates**

The non-spatial covariates to be included in the integrated model (for example, in the field of ecology the distance to the nearest road or time spent sampling could be considered). These covariates must be included in the same data object as the points.

**pointsIntercept**

Logical argument: should the points be modeled with intercepts. Defaults to TRUE. Note that if this argument is non-NULL and pointsIntercepts is missing, pointsIntercepts will be set to FALSE.

**marksIntercept**

Logical argument: should the marks be modeled with intercepts. Defaults to TRUE.

**Offset**

Name of the offset variable (class character) in the datasets. Defaults to NULL: if the argument is non-NULL, the variable name needs to be standardized across datasets (but does not need to be included in all datasets). The offset variable will be transformed onto the log-scale in the integrated model.

**pointsSpatial**

Argument to determine whether the spatial field is shared between the datasets, or if each dataset has its own unique spatial field. The datasets may share a spatial field with INLA’s “copy” feature if the argument is set to copy. May take on the values: “shared”, “individual”, “copy” or NULL if no spatial field is required for the model. Defaults to “shared”.

**responseCounts**

Name of the response variable in the counts/abundance datasets. This variable name needs to be standardized across all counts datasets used in the integrated model. Defaults to ‘counts’.

**responsePA**

Name of the response variable (class character) in the presence absence/detection non-detection datasets. This variable name needs to be standardized across all present absence datasets. Defaults to ‘present’.

**trialsPA**

Name of the trials response variable (class character) for the presence absence datasets. Defaults to NULL.

**trialsMarks**

Name of the trials response variable (class character) for the binomial marks (if included). Defaults to NULL.

**temporalName**

Name of the temporal variable (class character) in the model. This variable is required to be in all the datasets. Defaults to NULL.

**Formulas**

A named list with two objects. The first one, covariateFormula, is a formula for the covariates and their transformations for the distribution part of the model. Defaults to NULL which includes all covariates specified in spatialCovariates into the model. The second, biasFormula, specifies which covariates are used for the PO datasets. Defaults to NULL which includes no covariates for the PO datasets.

**Value**

A `specifyMarks` object (class R6). Use `?specifyMarks` or `.help()` to get a comprehensive description of the slot functions associated with this object.
Note

The idea with this function is to describe the full model: that is, all the covariates and spatial effects will appear in all the formulas for the datasets and species. If some of these terms should not be included in certain observation models in the integrated model, they can be thinned out using the .updateFormula function. Note: the point covariate and mark terms will only be included in the formulas for where they are present in a given dataset, and so these terms do not need to be thinned out if they are not required by certain observation models.

Examples

```r
if (requireNamespace('INLA')) {
  # Data
  data("SolitaryTinamou")
  proj <- "+proj=longlat +ellps=WGS84"
  data <- SolitaryTinamou$datasets
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj

  # Set base model up
  baseModel <- startMarks(data, Mesh = mesh,
                           Projection = proj, responsePA = 'Present',
                           markNames = 'speciesName',
                           markFamily = 'multinomial')
}
```

Description

This function is used to create an object containing all the data, metadata and relevant components required for the multi-species integrated species distribution model and INLA to work. As a result, the arguments associated with this function are predominantly related to describing variable names within the datasets that are relevant, and arguments related to what terms should be included in the formula for the integrated model. The output of this function is an R6 object, and so there are a variety of public methods within the output of this function which can be used to further specify the model (see ?specifySpecies or .$help() for a comprehensive description of these public methods).

Usage

```r
startSpecies(
  ..., spatialCovariates = NULL,
```
Projection,
Mesh,
speciesSpatial = "replicate",
speciesIntercept = TRUE,
speciesEnvironment = TRUE,
speciesName,
IPS = NULL,
Boundary = NULL,
pointCovariates = NULL,
Offset = NULL,
pointsIntercept = TRUE,
pointsSpatial = "copy",
responseCounts = "counts",
responsePA = "present",
trialsPA = NULL,
temporalName = NULL,
Formulas = list(covariateFormula = NULL, biasFormula = NULL)
)

Arguments

... The datasets to be used in the model. Must come as either sf objects, or as a list of named sf objects.

spatialCovariates

The spatial covariates used in the model. These covariates must be measured at every location (pixel) in the study area, and must be a SpatialRaster object. Can be either numeric, factor or character data. Defaults to NULL which includes no spatial effects in the model.

Projection The coordinate reference system used by both the spatial points and spatial covariates. Must be of class character.

Mesh An inla.mesh object required for the spatial random fields and the integration points in the model (see inla.mesh.2d from the INLA package for more details).

speciesSpatial Argument to specify if each species should have their own spatial effect with different hyperparameters to be estimated using INLA’s "replicate" feature, of if a the field’s should be estimated per species copied across datasets using INLA’s "copy" feature. Possible values include: ‘replicate’, ‘copy’, ‘shared’ or NULL if no species-specific spatial effects should be estimated.

speciesIntercept Argument to control the species intercept term. Defaults to TRUE which creates a random intercept term, FALSE creates a fixed intercept term, and NULL removes the intercept term.

speciesEnvironment Argument to control the species environmental term. Defaults to TRUE which creates species level environmental effects. To create shared effects across the species, use FALSE.
speciesName

Name of the species variable name (class character). Specifying this argument turns the model into a stacked species distribution model, and calculates covariate values for the individual species, as well as a species group model in the shared spatial field. Defaults to NULL. Note that if this argument is non-NULL and pointsIntercepts is missing, pointsIntercepts will be set to FALSE.

IPS

The integration points to be used in the model (that is, the points on the map where the intensity of the model is calculated). See fm_int from the inlabru package for more details regarding these points; however defaults to NULL which will create integration points from the Mesh object.

Boundary

A sf object of the study area. If not missing, this object is used to help create the integration points.

pointCovariates

The non-spatial covariates to be included in the integrated model (for example, in the field of ecology the distance to the nearest road or time spent sampling could be considered). These covariates must be included in the same data object as the points.

Offset

Name of the offset variable (class character) in the datasets. Defaults to NULL; if the argument is non-NULL, the variable name needs to be standardized across datasets (but does not need to be included in all datasets). The offset variable will be transformed onto the log-scale in the integrated model.

pointsIntercept

Logical argument: should the points be modeled with intercepts. Defaults to TRUE. Note that if this argument is non-NULL and pointsIntercepts is missing, pointsIntercepts will be set to FALSE.

pointsSpatial

Argument to determine whether the spatial field is shared between the datasets, or if each dataset has its own unique spatial field. The datasets may share a spatial field with INLA’s “copy” feature if the argument is set to copy. May take on the values: “shared”, “individual”, “copy”, “correlate” or NULL if no spatial field is required for the model. Defaults to "copy".

responseCounts

Name of the response variable in the counts/abundance datasets. This variable name needs to be standardized across all counts datasets used in the integrated model. Defaults to 'counts'.

responsePA

Name of the response variable (class character) in the presence absence/detection non-detection datasets. This variable name needs to be standardized across all present absence datasets. Defaults to 'present'.

trialsPA

Name of the trials response variable (class character) for the presence absence datasets. Defaults to NULL.

temporalName

Name of the temporal variable (class character) in the model. This variable is required to be in all the datasets. Defaults to NULL.

Formulas

A named list with two objects. The first one, covariateFormula, is a formula for the covariates and their transformations for the distribution part of the model. Defaults to NULL which includes all covariates specified in spatialCovariates into the model. The second, biasFormula, specifies which covariates are used for the PO datasets. Defaults to NULL which includes no covariates for the PO datasets.
Value

A `specifySpecies` object (class R6). Use `?specifySpecies` to get a comprehensive description of the slot functions associated with this object.

Note

The idea with this function is to describe the full model: that is, all the covariates and spatial effects will appear in all the formulas for the datasets and species. If some of these terms should not be included in certain observation models in the integrated model, they can be thinned out using the `.updateFormula` function. Note: the point covariate will only be included in the formulas for where they are present in a given dataset, and so these terms do not need to be thinned out if they are not required by certain observation models.

Examples

```r
## Not run:
if (requireNamespace('INLA')) {

##REDO WITH OTHER DATA

#Get Data
data("SolitaryTinamou")

proj <- "+proj=longlat +ellps=WGS84"
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

#Set base model up
baseModel <- startSpecies(data, Mesh = mesh,
                           Projection = proj, responsePA = 'Present',
                           speciesName = 'speciesName')

#Print summary
baseModel

#Set up model with dataset specific spatial fields
indSpat <- startSpecies(data, Mesh = mesh,
                        Projection = proj, pointsSpatial = 'individual',
                        responsePA = 'Present', speciesName = 'speciesName')

#Model with offset variable
offSet <- startSpecies(data, Mesh = mesh,
                       Projection = proj,
                       Offset = 'area',
                       responsePA = 'Present',
                       speciesName = 'speciesName')

#Non-random effects for the species
```
summary.bruSDM

Generic summary function for bruSDM.

Description
Summary for bru_sdm
Summary for modSpecies
Summary for modISDM
Summary for modMarks

Usage
## S3 method for class 'bruSDM'
summary(object, ...)

## S3 method for class 'modSpecies'
summary(object, ...)

## S3 method for class 'modISDM'
summary(object, ...)

## S3 method for class 'modMarks'
summary(object, ...)

Arguments
object modMarks object.
... Not used argument
Index

* data
  BBA, 3
  BBS, 3
  BBSColinusVirginianus, 4
  eBird, 13
  elev_raster, 13
  Gbif, 15
  Koala, 19
  NLCD_canopy_raster, 23
  Parks, 23
  region, 36
  SetophagaData, 38
  SolitaryTinamou, 38
  SolTinCovariates, 39

  BBA, 3
  BBS, 3
  BBSColinusVirginianus, 4
  blockedCV, 4, 47, 62, 77
  blockedCV-class, 5
  bru, 43, 58, 73
  bruSDM-class, 5
  bruSDM_predict-class, 6
  changeCoords, 6
  checkCoords, 6
  checkVar, 7
  control.group, 18, 41, 48, 56, 63, 71, 78

  data2ENV, 7
  dataOrganize, 8
  dataSet, 10
  datasetOut, 11
  datasetOut-class, 13

  eBird, 13
  elev_raster, 13

  fitISDM, 14, 27
  fm_int, 16, 85, 88, 92

  Gbif, 15
  inla.mesh.2d, 16, 85, 88, 91
  inla.spde2.matern, 41, 45, 56, 60, 71, 75
  inla.spde2.pcmatern, 45, 60, 75
  intModel, 15, 27, 29, 40, 42, 45, 55, 57, 60, 71, 75

  Koala, 19
  makeFormulaComps, 20
  makeLhoods, 20
  modISDM-class, 21
  modISDM_predict-class, 21
  modMarks-class, 21
  modMarks_predict-class, 22
  modSpecies-class, 22
  modSpecies_predict-class, 22
  nameChanger, 22
  NLCD_canopy_raster, 23

  Parks, 23
  plot.bruSDM_predict, 23
  plot.modISDM_predict
    (plot.bruSDM_predict), 23
  plot.modMarks_predict
    (plot.bruSDM_predict), 23
  plot.modSpecies_predict
    (plot.bruSDM_predict), 23
  predict.bruSDM, 27
  predict.modISDM(predict.bruSDM), 27
  predict.modMarks(predict.bruSDM), 27
  predict.modSpecies(predict.bruSDM), 27
  print.blockedCV, 32
  print.bruSDM, 33
  print.bruSDM_predict, 33
  print.datasetOut, 34
  print.modISDM, 34
  print.modMarks, 35
  print.modSpecies, 35
reduceComps, 36
region, 36
removeFormula, 37
runModel, 37

SetophagaData, 38
SolitaryTinamou, 38
SolTinCovariates, 39
spatialBlock, 47, 62, 77
specifyISDM, 18, 39, 86
specifyMarks, 54, 89
specifySpecies, 68, 93
startISDM, 4, 15, 27, 39, 84
startMarks, 27, 54, 87
startSpecies, 4, 15, 27, 69, 90
summary.bruSDM, 94
summary.modISDM (summary.bruSDM), 94
summary.modMarks (summary.bruSDM), 94
summary.modSpecies (summary.bruSDM), 94