Package ‘PointedSDMs’

June 15, 2022

Type Package

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

Version 1.0.6

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.


Depends R (>= 3.5), stats, methods, ggplot2, inlabru (>= 2.5)

Imports R6, sp (>= 1.4-5), raster, R.devices, blockCV

Suggests testthat (>= 3.0.0), USAboundaries, sf, sn, INLA (>= 21.08.31), rasterVis, ggmap, ggpolyline, RColorBrewer, cowplot, knitr, kableExtra, rmarkdown, covr

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

RoxygenNote 7.2.0

License GPL (>= 3)

VignetteBuilder knitr

Config/testthat/edition 3
LazyData true
Encoding UTF-8
NeedsCompilation no
Author Philip Mostert [aut, cre],
Bob O'hara [aut]
Repository CRAN
Date/Publication 2022-06-15 08:20:12 UTC

R topics documented:

Altitude ......................................................... 3
BBA ................................................................. 3
BBS ................................................................. 3
blockedCV .......................................................... 4
blockedCV-class .................................................. 5
bruSDM-class ...................................................... 5
bruSDM_predict-class ............................................ 5
changeCoords ..................................................... 6
checkCoords ...................................................... 6
checkVar ......................................................... 7
data2ENV .......................................................... 7
dataOrganize ..................................................... 8
dataSDM ........................................................... 10
dataSet ............................................................ 24
datasetOut ......................................................... 26
datasetOut-class .................................................. 27
eBird ............................................................... 27
eBird_caerulescens .............................................. 27
eBird_fusca ....................................................... 28
eBird_magnolia .................................................... 28
elev_raster ....................................................... 28
Forest .............................................................. 28
Gbif ............................................................... 29
intModel ........................................................... 29
makeLhoods ....................................................... 32
nameChanger ..................................................... 33
NLCD_canopy_raster ............................................ 34
NPP ............................................................... 34
Parks ............................................................. 34
plot.bruSDM_predict ............................................ 35
predict.bruSDM .................................................. 36
print.blockedCV .................................................. 38
print.bruSDM ..................................................... 39
print.bruSDM_predict ............................................ 39
print.datasetOut ................................................. 40
region ............................................................ 40
Altitude

SpatialPixelsDataFrame containing Altitude covariate

Description

SpatialPixelsDataFrame containing Altitude covariate

Source

https://github.com/oharar/PointedSDMs

BBA

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

http://www.pabirdatlas.psu.edu

BBS

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

blockedCV

blockedCV: run spatial blocked cross-validation on the integrated model.

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 intModel. 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

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

# Get Data
data("SolitaryTinamou")
proj <- CRS("+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)

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

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

    #Print summary
    blocked

}

## End(Not run)

---

**blockedCV-class**  
*Export class blockedCV*

---

**Description**

Export class blockedCV

---

**bruSOM-class**  
*Export bru_sdm class*

---

**Description**

Export bru_sdm class

---

**bruSOM_predict-class**  
*Export class predict_bru_sdm*

---

**Description**

Export class predict_bru_sdm
changeCoords

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

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)
```

**Method** `makeMultinom()`:

*Usage:*

```r
dataOrganize$makeMultinom(multinomVars, return, oldVars)
```
Method makeFormulas():

Usage:
dataOrganize$makeFormulas(
  spatcobs,
  speciesname,
  paresp,
  countresp,
  marks,
  marksspatial,
  spatial,
  intercept,
  temporalname,
  markintercept,
  pointcovs,
  speciesspatial
)

Method makeComponents():

Usage:
dataOrganize$makeComponents(
  spatial,
  intercepts,
  datanames,
  marks,
  speciesname,
  multinomnames,
  pointcovariates,
  covariatenames,
  covariateclass,
  marksspatial,
  marksintercept,
  temporalname,
  speciesspatial,
  numtime,
  temporalmodel,
  offsetname
)

Method makeLhoods():

Usage:
dataOrganize$makeLhoods(
  mesh,
  ips,
  paresp,
  ntrialsvar,
  markstrialssvar,
  speciesname
)
Method `clone()`: The objects of this class are cloneable with this method.

Usage:
```r
dataOrganize$clone(deep = FALSE)
```

Arguments:
- `deep` Whether to make a deep clone.

---

**dataSDM**

*R6 class for creating a dataSDM object.*

Description

A data object containing the data and the relevant information about the integrated model. The function `intModel` 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:
- `dataSDM$print()`
- `dataSDM$plot()`
- `dataSDM$addData()`
- `dataSDM$addBias()`
- `dataSDM$updateFormula()`
- `dataSDM$changeComponents()`
- `dataSDM$priorsFixed()`
- `dataSDM$specifySpatial()`
- `dataSDM$changeLink()`
- `dataSDM$spatialBlock()`
- `dataSDM$new()`
- `dataSDM$samplingBias()`

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

Usage:
```r
dataSDM$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:
dataSDM$plot(datasetNames, Species = FALSE, 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 FALSE.
Boundary      Logical: should a boundary (created using the Mesh object) be used in the plot. Defaults to TRUE. Note that the output of this function is a gg object, and so a boundary surrounding the points may be added using standard ggplot2 syntax and the gg function provided in the inlabru package.
... Not used.

Returns: A ggplot object.

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

  #Get Data
  data("SolitaryTinamou")
  proj <- CRS("+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')

  #Create plot of data
  organizedData$plot
}

Method addData(): Function used to add additional datasets to the dataSDM object. This function should be used if the user would like to add any additional datasets to the integrated model, but do not have the same standardized variable names as those added initially with intModel. Use ?intModel for a more comprehensive description on what each argument in this function is used for.

Usage:
dataSDM$addData(
  ..., 
  responseCounts, 
  responsePA, 
  trialsPA, 
  markNames, 
  markFamily, 
  pointCovariates, 
  trialsMarks, 
  speciesName,
temporalName, Coordinates, Offset
)

Arguments:
... The datasets to be added to the integrated model: should be either data.frame or SpatialPoints* objects, or a list of objects from these classes.
responseCounts The name of the response variable for the counts data.
responsePA The name of the response variable for the presence absence data.
trialsPA The name of the trials variable for the presence absence data.
markNames The names of the marks found in the data.
markFamily The associated distributions of the marks.
pointCovariates The additional, non-spatial covariates describing the data.
trialsMarks The name of the trials variable for the binomial marks.
speciesName The name of the species variable included in the data. Used to make a stacked species distribution model.
temporalName The name of the temporal variable in the datasets.
Coordinates A vector of length 2 describing the names of the coordinates of the data.
Offset Name of the offset column in the dataset

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

# Get Data
data("SolitaryTinamou")
proj <- CRS("+proj=longlat +ellps=WGS84")

# Only select eBird data
ebird <- SolitaryTinamou$datasets$eBird
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj

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

# Print summary
organizedData

# Add new dataset
Parks = SolitaryTinamou$datasets$Parks
organizedData$addData(Parks, responsePA = 'Present')

# Print summary
organizedData
}

**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
dataSDM$addBias(datasetNames = NULL, allPO = FALSE, biasField = NULL)
```

**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.

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

#Get Data
data("SolitaryTinamou")
proj <- CRS("+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')

#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
dataSDM$updateFormula(
    datasetName = NULL,
    Points = TRUE,
    speciesName = NULL,
    markName = NULL,
    Formula,
    allProcesses = FALSE,
    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. If FALSE, then markNames needs to be non-NULL in intModel.
speciesName Name of the species (class character) to change the formula for. Defaults to NULL.
   If NULL and speciesName is non-NULL in intModel, will update the formula for all species within the dataset. Cannot be non-NULL if speciesName is NULL in intModel.
markName Name of the mark (class character) to change the formula for. Defaults to NULL.
   If NULL and markNames is non-NULL in intModel, will update the formula for all marks within the dataset. Cannot be non-NULL if markNames is NULL in intModel.
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 \$.{addComponents} as well.
allProcesses 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: If Formula and newFormula are missing, will print out the formula for the specified processes.

Examples:

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

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

#Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
   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 dd scaling to components
organizedData$changeComponents(addComponent = 'scaling')
}
```
Method \texttt{changeComponents()}: Function to add and specify custom components to model, which are required by \texttt{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 .\texttt{updateFormula} function. If \texttt{addComponent} and \texttt{removeComponent} are both missing, the function will print out the components to be supplied to \texttt{inlabru}'s \texttt{bru} function.

\textit{Usage:}
\begin{verbatim}
dataSDM$changeComponents(addComponent, removeComponent, print = TRUE)
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
\item \texttt{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 \texttt{removeComponent}.
\item \texttt{removeComponent} Component (or just the name of a component) present in the model which should be removed.
\item \texttt{print} Logical: should the updated components be printed. Defaults to \texttt{TRUE}.
\end{itemize}

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

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

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

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

}
}
\end{verbatim}

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

dataSDM$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.

Examples:

  if (requireNamespace('INLA')) {

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

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

dataSDM$specifySpatial(
  sharedSpatial = FALSE,
dataSDM

datasetName,
Species,
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`.
Species Name of which of the species’ spatial field to be specified. Requires `speciesName` to be non-NULL in `intModel`.
Mark Name of which of the mark’ spatial field to be specified. Requires `markNames` to be non-NULL 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.

Examples:

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

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

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

# Specify the shared spatial field
organizedData$speifySpatial(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:
dataSDM$changeLink(datasetName, Species, Mark, Link, ...)

Arguments:
datasetName Name of the dataset for which the link function needs to be changed.
Species Name of the species for which the link function needs to be changed.
Mark Name of the mark 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.
... Not used
Examples:
\dontrun{
  #Create data object
dataObj <- intModel(...)

  #Print link function for a process
dataObj$changeLink(Dataset = Dataset, Species = Species)

  #Change link function
dataObj$changeLink(Dataset = Dataset, Species = Species,
                   Link = "log")
}

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:
dataSDM$spatialBlock(k, rows, cols, plot = FALSE, seed = 1234, ...)

Arguments:
k Integer value reflecting the number of folds to use.
rows Integer value by which the area is divided into latitudinal bins.
cols Integer value by which the area is divided into longitudinal 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:
if (requireNamespace('INLA')) {
  #Get Data
data("SolitaryTinamou")
proj <- CRS("+proj=longlat +ellps=WGS84")
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- SolitaryTinamou$covariates$Forest

#Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c("X", "Y"),
                           spatialCovariates = Forest,
                           Projection = proj, responsePA = "Present",
                           pointsSpatial = "individual")

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

Method new():
Usage:
dataSDM$new(
  coordinates, projection, Inlamesh, initialnames, responsecounts, responsepa,
  marksnames, marksfamily, pointcovariates, trialspa, trialsmarks, speciesname,
  marksspatial, spatial, intercepts, spatialcovariates, marksintercepts, boundary,
  ips, temporal, temporalmodel, speciesspatial, offset
)

Method samplingBias():
Usage:

dataSDM$samplingBias(datasetName, Samplers)

Note

The arguments of this function may be missing (ie not provided) if they have already been specified in intModel, and do not need changing. Therefore this function is useful if there are some variable names not standardized across the datasets; this function will thus standardize the variable names to those provided initially in intModel.

Examples

```r
## ------------------------------------------------
## Method 'dataSDM$plot'
## ------------------------------------------------

if (requireNamespace('INLA')) {

  # Get Data
data("SolitaryTinamou")
  proj <- CRS("+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')

  # Create plot of data
  organizedData$plot()
}

## ------------------------------------------------
## Method 'dataSDM$addData'
## ------------------------------------------------

if (requireNamespace('INLA')) {

  # Get Data
data("SolitaryTinamou")
  proj <- CRS("+proj=longlat +ellps=WGS84")

  # Only select eBird data
ebird <- SolitaryTinamou$datasets$eBird
  mesh <- SolitaryTinamou$mesh
  mesh$crs <- proj
```
dataSDM

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

# Print summary
organizedData

# Add new dataset
Parks = SolitaryTinamou$datasets$Parks
organizedData$addData(Parks, responsePA = 'Present')

# Print summary
organizedData

}"

## Method `dataSDM$addBias`
## -----------------------------------------------

if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- CRS("+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')

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

}

## Method `dataSDM$updateFormula`
## -----------------------------------------------

if (requireNamespace('INLA')) {

# Get Data
data("SolitaryTinamou")
proj <- CRS("+proj=longlat +ellps=WGS84")
data <- SolitaryTinamou$datasets
mesh <- SolitaryTinamou$mesh
mesh$crs <- proj
Forest <- SolitaryTinamou$covariates$Forest
#Set model up
organizedData <- intModel(data = data, Mesh = mesh, Coordinates = c('X', 'Y'),
                           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')

}
if (requireNamespace('INLA')) {

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

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

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

## ------------------------------------------------
## Method `dataSDM$specifySpatial`
## ------------------------------------------------

if (requireNamespace('INLA')) {

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

# Set model up
organizedData <- intModel(data, Mesh = mesh, Coordinates = c('X', 'Y'),
                           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 `dataSDM$changeLink`
## ------------------------------------------------
## Not run:

```r
# Create data object
dataObj <- intModel(...)

# Print link function for a process
dataObj$changeLink(Dataset = Dataset, Species = Species)

# Change link function
dataObj$changeLink(Dataset = Dataset, Species = Species,
                   Link = "log")
```

```r
## End(Not run)

## Method `dataSDM$spatialBlock`

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

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

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

---

**dataSet**

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

**Description**

Internal function used to assist in structuring the data.
DataSet

Usage

dataSet(
  datapoints,  # A list of datasets as either data.frame or SpatialPoints 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.
)

Arguments

datapoints  A list of datasets as either data.frame or SpatialPoints 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

datasetOut(model, dataset, predictions = TRUE)

Arguments

model            Model of class bru_sdm 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 <- CRS("+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')))

# 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/

eBird_caerulescens  
Dataset of setophaga caerulescens across Pennsylvania state.

**Description**

Dataset of setophaga caerulescens across Pennsylvania state.

**References**
https://ebird.org/home.
**eBird_fusca**  
*Dataset of setophaga fusca across Pennsylvania state.*

**Description**  
Dataset of setophaga fusca across Pennsylvania state.

**References**  
[https://ebird.org/home](https://ebird.org/home).

---

**eBird_magnolia**  
*Dataset of setophaga magnolia across Pennsylvania state.*

**Description**  
Dataset of setophaga magnolia across Pennsylvania state.

**References**  
[https://ebird.org/home](https://ebird.org/home).

---

**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](https://cran.r-project.org/package=elevatr)

---

**Forest**  
*SpatialPixelsDataFrame containing Forest covariate*

**Description**  
SpatialPixelsDataFrame containing Forest covariate

**Source**  
[https://github.com/oharar/PointedSDMs](https://github.com/oharar/PointedSDMs)
**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 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 ?dataSDM for a comprehensive description of these public methods).

**Usage**

```r
intModel(  
  ...,  
  spatialCovariates = NULL,  
  Coordinates,  
  Projection,  
  Boundary = NULL,  
  Mesh,  
  IPS = NULL,  
  speciesSpatial = TRUE,  
  markNames = NULL,  
  markFamily = NULL,  
  pointCovariates = NULL,  
  pointsIntercept = TRUE,  
  marksIntercept = TRUE,  
  Offset = NULL,  
  pointsSpatial = "shared",  
  marksSpatial = TRUE,  
  responseCounts = "counts",  
)```
responsePA = "present",
trialsPA = NULL,
trialsMarks = NULL,
speciesName = NULL,
temporalName = NULL,
temporalModel = list(model = "ar1")
)

Arguments

... The datasets to be used in the model. May come as either 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* or SpatialPixelsDataFrame object. Can be either numeric or factor\code{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 CRS (see CRS from the sp package for more details).

Boundary

##NOT USED YET.

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 ipoints from the inlabru package for more details regarding these points; however defaults to NULL which will create integration points from the Mesh object.

speciesSpatial

Logical argument: should the species have their own individual spatial fields in the integrated model. Defaults to TRUE.

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.
intModel

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

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. May take on the values: "shared", "individual" or NULL if no spatial field is required for the model. Defaults to "shared".

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.

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.

Value

A dataSDM object (class R6). Use ?dataSDM 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')) {
    #Get Data
    data("SolitaryTinamou")
    proj <- CRS("+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')

    #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')
}
```

---

### makeLhoods

**makeLhoods: function to make likelihoods.**

**Description**

Function to make the datasets into likelihoods.

**Usage**

```r
makeLhoods(data,
```

```r
...```
nameChanger

```r
formula,
family,
mesh,
ips,
paresp,
ntrialsvar,
markstrialsvar,
speciesname,
speciesindex
)
```

**Arguments**

- `data`: A list of sp 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.
- `markstrialsvar`: 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.

**Description**

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

**Usage**

```r
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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLCD_canopy_raster</td>
<td><em>Raster object containing the canopy cover across Pennsylvania state.</em></td>
<td><a href="https://cran.r-project.org/package=FedData">https://cran.r-project.org/package=FedData</a></td>
</tr>
<tr>
<td>NPP</td>
<td><em>SpatialPixelsDataFrame containing NPP covariate</em></td>
<td><a href="https://github.com/oharar/PointedSDMs">https://github.com/oharar/PointedSDMs</a></td>
</tr>
<tr>
<td>Parks</td>
<td><em>data.frame object containing solitary tinamou observations from Parks</em></td>
<td><a href="https://www.gbif.org/">https://www.gbif.org/</a></td>
</tr>
</tbody>
</table>
plot.bruSDM_predict

Generic plot function for predict_bru_sdm.

Description

Plot for predict_bru_sdm

Usage

## S3 method for class 'bruSDM_predict'
plot(
  x,
  whattoplot = c("mean"),
  cols = NULL,
  layout = NULL,
  colourLow = NULL,
  colourHigh = NULL,
  plot = TRUE,
  ...
)

Arguments

x
A bruSDM_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.

...Argument not used

Value

A ggplot2 object.
Examples

```r
## Not run:

if (requireNamespace('INLA')) {
  #Get Data
data("SolitaryTinamou")
  proj <- CRS("+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')))

  #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)
```

predict.bruSDM

Generic predict function for bru_SD objects.

Description

Predict function for the object produced by `runModel`. 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.

Usage

```r
## S3 method for class 'bruSDM'
predict(
  object,
  data = NULL,
  formula = NULL,
  mesh = NULL,
  mask = NULL,
```
Arguments

- **object**: A bru_sdm objects.
- **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.
- **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 run in the linear predictor be included in the predictions. Defaults to FALSE.
- **fun**: Function used to predict. Set to 'linear' if effects on the linear scale are desired.
- **...**: Additional arguments used by the inlabru predict function.

Details

Predict for bru_sdm

Value

A list of inlabru predict objects.
Examples

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

# Get data
data("SolitaryTinamou")
proj <- CRS("+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')))

# 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.
- **...**: Unused 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.

### region

*SpatialPolygons object containing the boundary region for solitary tinamouc*

**Description**

SpatialPolygons object containing the boundary region for solitary tinamouc

**Source**

[https://github.com/oharar/PointedSDMs](https://github.com/oharar/PointedSDMs)

### runModel

*runModel: 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**

```r
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 <- CRS("+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)
```

---

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

---

**Description**

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

---

**summary bruSDM**  
Generic summary function for bruSDM.

---

**Description**

Summary for bru_sdm
Usage

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

Arguments

- `object` bruSDM object.
- `...` Un used argument
Index

* data
  Altitude, 3
  BBA, 3
  BBS, 3
eBird, 27
eBird_caerulescens, 27
eBird_fusca, 28
eBird_magnolia, 28
elev_raster, 28
  Forest, 28
  Gbif, 29
  NLCD_canopy_raster, 34
  NPP, 34
  Parks, 34
  region, 40
  SolitaryTinamou, 41

Altitude, 3
BBA, 3
BBS, 3
blockedCV, 4, 18
blockedCV-class, 5
bru, 15
bruSDM-class, 5
bruSDM_predict-class, 5
changeCoords, 6
checkCoords, 6
checkVar, 7
control.group, 31
CRS, 30
data2ENV, 7
dataOrganize, 8
dataSDM, 10, 31
dataSet, 24
dataSetOut, 26
dataSetOut-class, 27
eBird, 27
eBird_caerulescens, 27
eBird_fusca, 28
eBird_magnolia, 28
elev_raster, 28
Forest, 28
Gbif, 29
gg, 11
inla.mesh.2d, 30
inla.spde2.matern, 13, 17
inla.spde2.pcmatern, 17
intModel, 4, 10, 11, 13, 14, 17, 20, 29, 36, 37
ipoints, 30
makeLhoods, 32
nameChanger, 33
NLCD_canopy_raster, 34
NPP, 34
plot.bruSDM_predict, 35
predict.bruSDM, 36
print.blockedCV, 38
print.bruSDM, 39
print.bruSDM_predict, 39
print.datasetOut, 40
region, 40
runModel, 36, 40
SolitaryTinamou, 41
spatialBlock, 18
summary.bruSDM, 41