Package 'rSPACE'

December 15, 2014

Type Package

Title Spatially-explicit Power Analysis for Conservation and Ecology

Version 1.0

Date 2014-12-15

Author Martha Ellis, Jake Ivan, Jody Tucker, Mike Schwartz

Maintainer Martha Ellis <martha.ellis@gmail.com>

Description This package provides subroutines to run a spatially-explicit power analysis for detecting population trends through occupancy-based modeling.

License GPL (>=2)

Imports raster, RMark, ggplot2, tcltk2, sp, grid, plyr, tcltk

R topics documented:

`rSPACE-package` *Spatially-explicit Power Analysis for Conservation and Ecology*

Description

This package provides subroutines necessary to run a simple, spatially-explicit population simulation and then conduct a power analysis for detecting population trends through occupancy-based modeling.

Details

Package:	rSPACE
Type:	Package
Version:	1.0
Date:	2013-10-15
License:	GPL (>=2)

There are many possible options for how a simulation and power analysis could be conducted. We set up this package to provide options for how to conduct both the simulations and the data analysis. We would like to help make this framework accessible, but we also acknowledge that it may not always work in situations for which the software was not originally designed. We will be working to make more of the functionality accessible as we go along and will be interested to hear how things work (or fail to work) for anyone else who is interested in using our program.

For now, we have only made the basics of the package available. We originally designed our population simulation for territorial carnivores. Given a raster habitat layer, we distribute individuals according to a number of spacing rules (e.g. territory size, percent overlap, etc). We use bivariate normal movement distributions, adjusted based on an underlying raster habitat layer, to build a probability of use layer for each individual. We then combine the individual probabilities to create a layer describing the probability of use for at least one individual for the entire landscape. The landscape is gridded into cells and used to create an encounter history at each possible grid cell.

There are a large number of parameters needed to set up this stage of the analysis. The parameters are stored as a list, which can either be entered manually by the user or via a dialog box (for details see `enter.parameters`). The function, `encounter.history`, takes the list of parameters and a raster map of habitat on the landscape to produce a single, complete encounter history file, with options to error check the building process. `create.landscapes` will create replicated encounter history files. These data will be stored as text files (.txt) that can either be used as inputs for Program MARK, etc or read back in later in the power analysis side of the simulations.

Just as there are many options for simulating a population on a landscape, there are many ways to potentially analyze each encounter history. The way our analysis is set up, we have one wrapper function `test_samples` set up to subset the encounter history files to simulate varying sampling effort. `test_samples` takes as an argument a function name that will define the analysis that is run on each potential encounter history. This function should take the encounter history and some information about that file as arguments and return a data frame with the simulation results. We have provided the original test file that we used to analyze simulated wolverine occupancy in Ellis et al. (2013) in `wolverine_analysis`. The final output of the simulations at this stage is a text file storing simulation results, based on the output of `wolverine_analysis`.

There are many options at each of these steps, but the basic process is:

1. Enter parameters: `enter.parameters`
2. Build encounter history files: `create.landscapes`
3. Analyze encounter histories: `test_samples`

The help files will focus on these three steps, and explain additional details needed at each step.

Author(s)

Martha Ellis, Jake Ivan, Jody Tucker, Mike Schwartz

Maintainer: Martha Ellis <martha.ellis@gmail.com>

References

ELLIS, MARTHA M., JACOB S. IVAN, and MICHAEL K. SCHWARTZ. "Spatially Explicit Power Analyses for Occupancy-Based Monitoring of Wolverine in the US Rocky Mountains." Conservation Biology (2013).

Examples

```
## Not run
# data(WolverineHabitat)           # Loads example habitat map
# Parameters<-enter.parameters()   # Open dialog box with default parameter values
#
# # Create encounter history replicates
# create.landscapes(n_runs=10, map=WolverineHabitat, Parameters=Parameters)
#
# # Analyze encounter histories (
# # Default: RDOccup model via RMark (Program MARK must be available))
# test_samples(folder="./rSPACE.X", Parameters)
```

`create.landscapes` *Wrapper function to create replicate simulated landscapes*

Description

Creates a series of files with replicated, complete encounter histories based on habitat and population parameters.

Usage

```
create.landscapes(n_runs, map, Parameters, ... )
```

Arguments

`n_runs`	number of replicate encounter histories to produce
`map`	a raster layer with underlying habitat information
`Parameters`	list of parameter values for simulation (see `enter.parameters`)
`...`	optional options (see details)

Details

This function will produce `n_run` text files with simulated encounter histories. Each individual file does not contain identifying information about the parameters used to create it, so it will be good practice to create a folder structure to keep separate simulations separate. Default is to create a folder for the runs (./rSPACE_X) and name all encounter histories ./rSPACE_X/rSPACEx??.txt where ?? indicates the run number. If `Parameters` is not specified, `enter.parameters` will be called.

`map` defines the landscape for the population simulation. Values of `map` are used as weights for both selecting individual activity centers and scaling bivariate normal movement distributions.

There are additional arguments that can be included to specify other options, but some are more sketchy than others.

Optional argument `skipConfirm` will bypass the user-confirmation for creating a folder structure.

Optional argument `run.label` changes the folder name for the run scenario. Defaults to './rSPACE_X'

Optional argument `base.name` can be used to change the base file name for encounter history replicate files. Defaults to 'rSPACEx'

Optional argument `filter.map` can be used to specify an alternative sampling frame for the landscape. The default sampling frame consists of a rectangular grid built from the habitat information in `map` and user-specifications for `Paramters$grid_size` and `Parameters$sample.cutoff`. By providing a secondary raster layer via `filter.map`, users can restrict sampling to specific areas (e.g., to exclude inaccessible private lands or restrict sampling to a single jurisdiction such as USFS, NPS, BLM, etc) or to provide a user-specific grid for the entire landscape. The raster provided in `filter.map` will be expanded to match the extent of `map`, with non-coded regions coded 0. Raster values of 0 in `filter.map` are treated as areas to exclude from sampling. If `filter.map` consists of 0/1 values only, the default rectangular grid will be restricted to areas with values of 1 in `filter.map`. If `filter.map` contains additional values, non-zero values will be treated as indentifiers for sampling cells in a user-specified grid.

Optional argument `printN` defaults to TRUE. Logical - prints population size by run by year to a text file.

Optional argument `saveParameters` defaults to TRUE. Logical - save rData file with parameter list used in run scenario.

Value

returns list with directory location and output file names

Author(s)

Martha Ellis

References

ELLIS, MARTHA M., JACOB S. IVAN, and MICHAEL K. SCHWARTZ. "Spatially Explicit Power Analyses for Occupancy-Based Monitoring of Wolverine in the US Rocky Mountains." Conservation Biology (2014).

`encounter.history` *Produce a single encounter history file based on population parameters.*

Description

Given an underlying landscape, populates the landscape with simulated individuals based on spacing rules in `Parameters`. Calculates probability of at least one individual at every location on the landscape and accumulates probability over grid cells to determine presence/absence.

Usage

```
encounter.history(map, Parameters, ...)
```

Arguments

`Parameters`	list of parameter values for simulation
`map`	a raster layer with underlying habitat information
`...`	additional options

Details

Main simulation function called through `create.landscapes` to produce replicate, complete encounter history files.

Use `showSteps = T` as an additional option to display plots, save sampling grid, and other output as error checking.

Value

Vector with one element for each grid cell representing the encounter history at that cell. Encounter histories consist of 1/0 values indicating presence/absence of at least one individual in each cell at each visit in each year of the study period.

See Also

`create.landscapes`

`enter.parameters` *Enter parameters*

Description

Dialogue box for help setting up a parameter list

Usage

```
enter.parameters(Parameters = NULL)
```

Arguments

`Parameters`	Optional parameters to use as defaults in clicky box.

Details

Creates a minimal set of parameters necessay to run simulations and analysis. Additional arguments can be added to the outputted list as needed.

Parameter explanations

Population parameters	———————-
Initial population size	Population size at the start of the simulation
Population growth rate	Annual population multiplication rate (Lambda=N_t/N_0)
Number of years	Maximum number of years for the simulation
Number of types	Types of individuals to treat separately in the population simulation
Proportion of population	Proportion of population in each individual-type (e.g. sex ratio)
Movement parameters	———————-

Buffer distance	Minimum distance between movement centers for each type of individual
Movement radius	With `moveDistQ`, used to define bivariate normal use distribution
Proportions of movements within radius	With movement radius, used to define bivariate normal use distribution
Truncation	Limits to long-distance movements (proportion of movement distribution to all
Sampling parameters	———————-
Cell size	Area of grid cell
Minimum habitat value	Cutoff value for where individual centers should be located
Minimum sampling value	Cutoff for home much habitat must be included in cell to sample
Number of visits	Maximum number of visits to each grid cell per year

Value

A parameter list with the following items

N Initial population size

lmda Population growth rate

n_yrs Maximum number of years in the simulation

MFratio Proportion of population by type of individual

buffer Distance between individual movement centers for each type of individual

moveDist Percent of time within movement radius for each type of individual

moveDistQ Movement radius for each type of individual

maxDistQ Proportion of movement distribution to allow

grid_size Area of each grid cell

habitat.cutoff Minimum value of `map` where individual movement centers can be located.

sample.cutoff Cutoff of `map` values above `habitat.cutoff` required for a grid cell to be included in sampling

n_visits Maximum number of visits per year

detP Initial detection probability = 1 (perfect detection to established availability)

Note

Please direct all complaints about the damn clicky box to Mike Schwartz

Author(s)

Martha Ellis, Jody Tucker, Mike Schwartz

`findPower` *Predict power from an rSPACE simulation and analysis*

Description

Uses a loess smoother to predict either the number of grid cells needed to achieve a given level of power or the expected power from a specific sample size based on rSPACE output.

Usage

```
findPower(folder, data, CI = 0.95, pwr = 0.8, n_grid=NULL)
```

Arguments

folder	Full path for rSPACE folder containing both simulation and analysis output
data	Dataframe with previously loaded results
CI	Significance level
pwr	If specified, desired power
n_grid	If specified, number of grid cells used

Details

Data must be provided either by specifying a folder within which to look for a results file or providing previously loaded results.

Depending on whether pwr or n_grid are specified, findPower will return the opposite. If both pwr and n_grid, n_grid will be ignored

Value

Either the number of cells required to reach a given power or the power expected from a sample of a given number of cells.

Author(s)

Martha Ellis

---

getResults *Plot power curves and return analysis data*

Description

Load data and plot power curves for representing each subset in a specified rSPACE folder.

Usage

```
getResults(folder, CI = 0.95, returnData = 1, plot=T)
```

Arguments

folder	Full path for rSPACE folder containing both simulation and analysis output
CI	Significance value for one-tailed test
returnData	Option to return contents of results file as a data.frame.
plot	Option to plot power curves [Logical, default=T]

Value

If returnData=0, getResults returns nothing
returnData=1, the complete output for each analyses are returned
returnData=2, output of the analyses summarized by sampling intensity is returned.

Author(s)

Martha Ellis

IrregularYrs *Possible sampling frame for irregular visits across years*

Description

For use in resampling, a data.frame with 223 observations of visiting schedules over a 10 year period. Each row represents a sampling cell, with 1/0 in each year representing whether the cell was visited or not. For use in mimicking the implementation of a sampling design where logistics, funding, or site access (weather, permits, etc) may interfer with planned regular sampling.

Usage

```
data(IrregularYrs)
```

Format

A data frame with 223 observations of 10 variables.

- `Y0` Indicator of whether cell was visited at start of study
- `Y1` Indicator of whether cell was visited in year 1
- `Y2` Indicator of whether cell was visited in year 2
- `Y3` Indicator of whether cell was visited in year 3
- `Y4` Indicator of whether cell was visited in year 4
- `Y5` Indicator of whether cell was visited in year 5
- `Y6` Indicator of whether cell was visited in year 6
- `Y7` Indicator of whether cell was visited in year 7
- `Y8` Indicator of whether cell was visited in year 8
- `Y9` Indicator of whether cell was visited in year 9

Details

The IrregularYrs sampling option was developed to simulate sampling that does not follow a regular schedule (i.e. every year, or alternate year sampling) which is a reality of many monitoring studies. Long term monitoring programs to assess trend often face logistic, environmental, or financial challenges that may unexpectedly cause deviation from the planned sampling schedule (i.e. a heavy snow year precludes access to a portion of the sampling grid). The IrregularYrs option can be used to investigate how such irregular sampling may affect statistical power to detect trend. This example data is based on systematic sampling conducted by U.S Forest Service Sierra Nevada carnivore monitoring program designed to detect declines in occupancy for the southern Sierra Nevada fisher population (described in detail in Zielinski et al. 2013).

Source

Data expanded from dataset analyzed in Zielinski, W. J., Baldwin, J. A., Truex, R. L., Tucker, J. M., and Flebbe, P. A. (2013). Estimating trend in occupancy for the southern Sierra fisher (Martes pennanti) population. Journal of Fish and Wildlife Management, 4(1), 3-19

Examples

```
data(IrregularYrs)
IrregularYrs[sample(nrow(IrregularYrs), 10, replace=TRUE), ]
```

`test_samples` *Analyze a set of encounter histories*

Description

For a folder containing simulated encounter histories: reads in complete encounter histories, subsets data if applicable, and runs an analysis function on each subsetted encounter history. Results are accumulated into a data.frame and written into a results output file.

Usage

```
test_samples(folder, Parameters, ... )
```

Arguments

`folder`	Folder containing encounter history files (full path).
`Parameters`	list of parameter values
`...`	additional arguments

Details

The main role of `test_samples` is to read in complete encounter histories, subset them based on varying sampling effort, and then run an analysis on each "observed" encounter history (e.g. subsetted file). The actual analysis that gets applied will be determined by the function specified with `function_name`. The default function, `wolverine_analysis`, includes code from the original analysis used in Ellis et al. (2013).

Options for subsetting the encounter histories can be specified via `Parameters`. The number of visits per year to test can be specified by `Parameters$n_visit_test`, the proprotion of the grid included in the sample `Parameters$grid_sample`, the detection probability per visit `Parameters$detP_test`, and the sampling scheme by year `Parameters$alt_model`. If these objects are not in the `Parameters` list, the default values are:

- `Parameters$n_visit_test = 2:Parameters$n_visits`
- `Parameters$detP_test = c(1,0.8,0.2)`
- `Parameters$grid_sample = c(0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.75, 0.95)`
- `Parameters$alt_model = c(0,1)`

For `Parameters$alt_model`, codes will need to match analyses in the function specified by `function_name`. Using `wolverine_analysis`, `Parameters$alt_model = 0` indicates continuous sampling and `Parameters$alt_model = 1` indicates alternate year sampling.

Value

Stores simulation results based on output specified in the analysis to `sim_results.txt`. Returns time used for the function run.

Author(s)

Martha Ellis, Jake Ivan

References

ELLIS, MARTHA M., JACOB S. IVAN, and MICHAEL K. SCHWARTZ. "Spatially Explicit Power Analyses for Occupancy-Based Monitoring of Wolverine in the US Rocky Mountains." Conservation Biology (2013).

See Also

wolverine_analysis

`wolverine_analysis` *Default function for analyzing an individual encounter history*

Description

Original function designed for analyzing wolverine encounter histories in Ellis et al. 2013.

Usage

```
wolverine_analysis(n_yrs, ch = NULL, n_visit = NULL, sample_yr = 0, FPC = 1, ... )
```

Arguments

`n_yrs`	Number of years in the encounter history. Run with all else NULL to produce an initial data.frame for setting up further results.
`ch`	Character vector with encounters
`n_visit`	Number of visits per year
`sample_yr`	Code for possible alternate model formulations
`FPC`	Finite population correction
`...`	additional arguments

Details

This function provides an example of an analysis function, which will be called by `test_samples` on each individual encounter history. Should include appropriate model options for every subset of encounter histories. Returns a dataframe with results from current simulation, which will be compiled into a single output file by `test_samples`. Can include multiple model runs on the same encounter history by returning a data.frame with multiple lines.

The current function runs a robust design occupancy model through RMark with options for continuous or alternate year sampling. Trend is tested using variance.components procedure.

Value

data.frame with model results

Note

The analysis function you give `test_samples` via `function_name` will get called in two places in `test_samples`. First, it will get called by `RunAnalysis(n_yrs)` to set up a data.frame header. Then it will get called in each step of the loop over all the encounter history files and subsets using `RunAnalysis(n_yrs, ch, n_visit, sample_yr,fpc , ... )`. This is clunky, but for now, if you want to change `wolverine_analysis`, you need to make sure the function you specify will work with those two function calls.

Author(s)

Jake Ivan, Martha Ellis

References

ELLIS, MARTHA M., JACOB S. IVAN, and MICHAEL K. SCHWARTZ. "Spatially Explicit Power Analyses for Occupancy-Based Monitoring of Wolverine in the US Rocky Mountains." Conservation Biology (2013).

See Also

test_samples

Index