Package: surveyvoi (via r-universe)

October 7, 2026

Type Package

Version 1.1.1

Title Survey Value of Information

Description Decision support tool for prioritizing sites for ecological surveys based on their potential to improve plans for conserving biodiversity (e.g. plans for establishing protected areas). Given a set of sites that could potentially be acquired for conservation management, it can be used to generate and evaluate plans for surveying additional sites. Specifically, plans for ecological surveys can be generated using various conventional approaches (e.g. maximizing expected species richness, geographic coverage, diversity of sampled environmental algorithms. After generating such survey plans, they can be evaluated using conditions) and maximizing value of information. Please note that several functions depend on the 'Gurobi' optimization software (available from <https://www.gurobi.com>). Additionally, the 'JAGS' software (available from <https://mcmc-jags.sourceforge.io/>) is required to fit hierarchical generalized linear models. For further details, see Hanson et al. (2023) <doi:10.1111/1365-2664.14309>.

Imports utils, methods, stats, parallel, progress (>= 1.2.2), assertthat (>= 0.2.0), xgboost (>= 1.7.8.1), plyr (>= 1.8.4), withr (>= 2.1.2), tibble (>= 2.1.3), scales (>= 1.0.0), doParallel (>= 1.0.15), dplyr (>= 0.8.3), vegan (>= 2.5-6), RcppAlgos (>= 2.3.6), groupdata2 (>= 1.3.0), Rcpp (>= 0.12.19), Rsymphony (>= 0.1-31),

Suggests testthat (>= 3.1.2), knitr (>= 1.20), roxygen2 (>= 6.1.0), rmarkdown (>= 1.10), tidyr (>= 1.0.0), ggplot2 (>= 3.2.1), gridExtra (>= 2.3), viridis (>= 0.5.1), PoissonBinomial (>= 1.1.1), gurobi (>= 8.1.0), Rmpfr (>= 0.8-1), runjags (>= 2.0.4.6)

Depends R (>= 4.0.0), Matrix, sf (>= 0.8.0), nloptr (>= 1.2.2.2),

LinkingTo Rcpp (>= 0.12.19), RcppEigen (>= 0.3.3.7.0), PoissonBinomial (>= 1.1.1), nloptr (>= 1.2.2.2),

License GPL-3

LazyData true

Language en-US

SystemRequirements JAGS (>= 4.3.0) (optional), fftw3 (>= 3.3), gmp (>= 6.2.1), gmpxx (>= 6.2.1), mpfr (>= 3.0.0), autoconf (>= 2.69), automake (>= 1.16.5)

URL https://prioritizr.github.io/surveyvoi/

BugReports https://github.com/prioritizr/surveyvoi/issues

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

Biarch true

Collate 'RcppExports.R' 'approx_evdsi.R' 'evdsi.R' 'internal.R' 'approx_near_optimal_survey_scheme.R' 'approx_optimal_survey_scheme.R' 'cluster.R' 'data.R' 'ilp.R' 'env_div_survey_scheme.R' 'evdci.R' 'feasible_survey_schemes.R' 'fit_hglm_occupancy_models.R' 'fit_xgb_occupancy_models.R' 'geo_cov_survey_scheme.R' 'greedy_heuristic_optimization.R' 'n_states.R' 'optimal_survey_scheme.R' 'package.R' 'prior_probability_matrix.R' 'relative_site_richness_scores.R' 'relative_site_uncertainty_scores.R' 'simulate_feature_data.R' 'simulate_site_data.R' 'validate.R' 'weighted_survey_scheme.R' 'zzz.R'

NeedsCompilation yes

Author Jeffrey O Hanson [aut, cre] (<https://orcid.org/0000-0002-4716-6134>), Iadine Chadès [aut] (<https://orcid.org/0000-0002-7442-2850>), Emma J Hudgins [aut] (<https://orcid.org/0000-0002-8402-5111>), Joseph R Bennett [aut] (<https://orcid.org/0000-0002-3901-9513>)

Maintainer Jeffrey O Hanson <jeffrey.hanson@uqconnect.edu.au>

Config/pak/sysreqs automake libfftw3-dev libgmp3-dev jags libmpfr-dev

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

Date/Publication 2025-04-14 08:42:26 UTC

RemoteUrl https://github.com/cran/surveyvoi

RemoteRef HEAD

RemoteSha e6d6362f254cf766fa2c5dd5424b3fe305bdf4f3

Contents

approx_evdsi 3
approx_near_optimal_survey_scheme 7

approx_optimal_survey_scheme 12
env_div_survey_scheme 16
evdci 18
evdsi 21
feasible_survey_schemes 25
fit_hglm_occupancy_models 27
fit_xgb_occupancy_models 31
geo_cov_survey_scheme 34
greedy_heuristic_prioritization 37
n_states 39
optimal_survey_scheme 39
prior_probability_matrix 43
relative_site_richness_scores 45
relative_site_uncertainty_scores 47
sim_data 48
simulate_feature_data 50
simulate_site_data 51
surveyvoi 53
weighted_survey_scheme 54

Index **57**

`approx_evdsi` *Approximate expected value of the decision given survey information*

Description

Calculate the *expected value of the management decision given survey information*. This metric describes the value of the management decision that is expected when the decision maker conducts a surveys a set of sites to inform the decision. To speed up the calculations, an approximation method is used.

Usage

```
approx_evdsi(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  site_survey_scheme_column,
  site_survey_cost_column,
  feature_survey_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
```

```
  feature_target_column,
  total_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
  prior_matrix = NULL,
  n_approx_replicates = 100,
  n_approx_outcomes_per_replicate = 10000,
  seed = 500
)
```

Arguments

`site_data` sf::sf() object with site data.

`feature_data` base::data.frame() object with feature data.

`site_detection_columns`
: character names of numeric columns in the argument to site_data that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.

`site_n_surveys_columns`
: character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

`site_probability_columns`
: character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.

`site_management_cost_column`
: character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

`site_survey_scheme_column`
: character name of logical (TRUE / FALSE) column in the argument to site_data that indicates which sites are selected in the scheme or not. No missing NA values are permitted. Additionally, only sites that are missing data can be selected or surveying (as per the argument to site_detection_columns).

`site_survey_cost_column`
: character name of column in the argument to site_data that contains costs for surveying each site. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

feature_survey_column
: character name of the column in the argument to feature_data that contains logical (TRUE / FALSE) values indicating if the feature will be surveyed in the planned surveys or not. Note that considering additional features will rapidly increase computational burden, and so it is only recommended to consider features that are of specific conservation interest. No missing (NA) values are permitted in this column.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

total_budget
: numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.

site_management_locked_in_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

prior_matrix
: numeric matrix containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to NULL such that prior data is calculated automatically using prior_probability_matrix().

n_approx_replicates
: integer number of replicates to use for approximating the expected value calculations. Defaults to 100.

n_approx_outcomes_per_replicate
: integer number of outcomes to use per replicate for approximation calculations. Defaults to 10000.

seed
: integer state of the random number generator for simulating outcomes when conducting the value of information analyses. Defaults to 500.

Details

This function uses approximation methods to estimate the expected value calculations. The accuracy of these calculations depend on the arguments to n_approx_replicates and n_approx_outcomes_per_replicate, and so you may need to increase these parameters for large problems.

Value

A numeric vector containing the expected values for each replicate.

See Also

prior_probability_matrix().

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
total_budget <- sum(sim_sites$management_cost) * 0.5
```

```
# create a survey scheme that samples the first two sites that
# are missing data
sim_sites$survey_site <- FALSE
sim_sites$survey_site[which(sim_sites$n1 < 0.5)[1:2]] <- TRUE

# calculate expected value of management decision given the survey
# information using approximation method
approx_ev_survey <- approx_evdsi(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_site",
  "survey_cost", "survey", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
  "target", total_budget)

# print mean value
print(mean(approx_ev_survey))
```

`approx_near_optimal_survey_scheme` *Approximately near optimal survey scheme*

Description

Find a near optimal survey scheme that maximizes value of information. This function uses the approximation method for calculating the expected value of the decision given a survey scheme, and a greedy heuristic algorithm to maximize this metric.

Usage

```
approx_near_optimal_survey_scheme(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  site_survey_cost_column,
  feature_survey_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
  feature_target_column,
  total_budget,
  survey_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
```

```
  site_survey_locked_out_column = NULL,
  prior_matrix = NULL,
  n_approx_replicates = 100,
  n_approx_outcomes_per_replicate = 10000,
  seed = 500,
  n_threads = 1,
  verbose = FALSE
)
```

Arguments

`site_data` sf::sf() object with site data.

`feature_data` base::data.frame() object with feature data.

`site_detection_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.

`site_n_surveys_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

`site_probability_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (`NA`) values are permitted in these columns.

`site_management_cost_column`
: `character` name of column in the argument to `site_data` that contains costs for managing each site for conservation. This column should have `numeric` values that are equal to or greater than zero. No missing (`NA`) values are permitted in this column.

`site_survey_cost_column`
: `character` name of column in the argument to `site_data` that contains costs for surveying each site. This column should have `numeric` values that are equal to or greater than zero. No missing (`NA`) values are permitted in this column.

`feature_survey_column`
: `character` name of the column in the argument to `feature_data` that contains `logical` (`TRUE` / `FALSE`) values indicating if the feature will be surveyed in the planned surveys or not. Note that considering additional features will rapidly increase computational burden, and so it is only recommended to consider features that are of specific conservation interest. No missing (`NA`) values are permitted in this column.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

total_budget
: numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.

survey_budget
: numeric maximum expenditure permitted for conducting surveys.

site_management_locked_in_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

site_survey_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out (TRUE) from being selected for future surveys or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites will never be considered for future surveys (e.g. because they are too costly to survey, or have a low chance of containing the target species). Defaults to NULL such that no sites are locked out.

prior_matrix
: numeric matrix containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to NULL such that prior data is calculated automatically using prior_probability_matrix().

n_approx_replicates
: integer number of replicates to use for approximating the expected value calculations. Defaults to 100.

n_approx_outcomes_per_replicate
: integer number of outcomes to use per replicate for approximation calculations. Defaults to 10000.

seed
: integer state of the random number generator for simulating outcomes when conducting the value of information analyses. Defaults to 500.

n_threads
: integer number of threads to use for computation.

verbose
: logical indicating if information should be printed during processing. Defaults to FALSE.

Details

Ideally, the brute-force algorithm would be used to identify the optimal survey scheme. Unfortunately, it is not feasible to apply the brute-force to large problems because it can take an incredibly long time to complete. In such cases, it may be desirable to obtain a "relatively good" survey scheme and the greedy heuristic algorithm is provided for such cases. The greedy heuristic algorithm – unlike the brute force algorithm – is not guaranteed to identify an optimal solution – or even a "relatively good solution" for that matter – though greedy heuristic algorithms tend to deliver solutions that are 15\ greedy algorithms is implemented as:

1. Initialize an empty *list of survey scheme solutions*, and an empty *list of approximate expected values*.
2. Calculate the expected value of current information.
3. Add a survey scheme with no sites selected for surveying to the *list of survey scheme solutions*, and add the expected value of current information to the *list of approximate expected values*.
4. Set the *current survey solution* as the survey scheme with no sites selected for surveying.
5. For each remaining candidate site that has not been selected for a survey, generate a new candidate survey scheme with each candidate site added to the current survey solution.
6. Calculate the approximate expected value of each new candidate survey scheme. If the cost of a given candidate survey scheme exceeds the survey budget, then store a missing NA value instead. Also if the the cost of a given candidate survey scheme plus the management costs of locked in planning units exceeds the total budget, then a store a missing value NA value too.

7. If all of the new candidate survey schemes are associated with missing NA values – because they all exceed the survey budget – then go to step 12.
8. Calculate the cost effectiveness of each new candidate survey scheme. This calculated as the difference between the approximate expected value of a given new candidate survey scheme and that of the *current survey solution*, and dividing this difference by the the cost of the newly selected candidate site.
9. Find the new candidate survey scheme that is associated with the highest cost-effectiveness value, ignoring any missing NA values. This new candidate survey scheme is now set as the *current survey scheme*.
10. Store the *current survey scheme* in the *list of survey scheme solutions* and store its approximate expected value in the *list of approximate expected values*.
11. Go to step 12.
12. Find the solution in the *list of survey scheme solutions* that has the highest expected value in the *list of approximate expected values* and return this solution.

Value

A matrix of logical (TRUE/ FALSE) values indicating if a site is selected in the scheme or not. Columns correspond to sites, and rows correspond to different schemes. If there are no ties for the best identified solution, then the the matrix will only contain a single row.

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
total_budget <- sum(sim_sites$management_cost) * 0.5

# set total budget for surveying sites for conservation
# (i.e. 40% of the cost of managing all sites)
survey_budget <- sum(sim_sites$survey_cost) * 0.4

# find survey scheme using approximate method and greedy heuristic algorithm
# (using 10 replicates so that this example completes relatively quickly)
approx_near_optimal_survey <- approx_near_optimal_survey_scheme(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_cost",
  "survey", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
```

```
  "target", total_budget, survey_budget)

# print result
print(approx_near_optimal_survey)
```

`approx_optimal_survey_scheme` *Approximately optimal survey scheme*

Description

Find the optimal survey scheme that maximizes value of information. This function uses the approximation method for calculating the expected value of the decision given a survey scheme.

Usage

```
approx_optimal_survey_scheme(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  site_survey_cost_column,
  feature_survey_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
  feature_target_column,
  total_budget,
  survey_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
  site_survey_locked_out_column = NULL,
  prior_matrix = NULL,
  n_approx_replicates = 100,
  n_approx_outcomes_per_replicate = 10000,
  seed = 500,
  n_threads = 1,
  verbose = FALSE
)
```

Arguments

`site_data`	`sf::sf()` object with site data.
`feature_data`	`base::data.frame()` object with feature data.

site_detection_columns
: character names of numeric columns in the argument to site_data that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.

site_n_surveys_columns
: character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

site_probability_columns
: character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.

site_management_cost_column
: character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

site_survey_cost_column
: character name of column in the argument to site_data that contains costs for surveying each site. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

feature_survey_column
: character name of the column in the argument to feature_data that contains logical (TRUE / FALSE) values indicating if the feature will be surveyed in the planned surveys or not. Note that considering additional features will rapidly increase computational burden, and so it is only recommended to consider features that are of specific conservation interest. No missing (NA) values are permitted in this column.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature

in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

total_budget
: numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.

survey_budget
: numeric maximum expenditure permitted for conducting surveys.

site_management_locked_in_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

site_survey_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out (TRUE) from being selected for future surveys or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites will never be considered for future surveys (e.g. because they are too costly to survey, or have a low chance of containing the target species). Defaults to NULL such that no sites are locked out.

prior_matrix
: numeric matrix containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to NULL such that prior data is calculated automatically using prior_probability_matrix().

n_approx_replicates
: integer number of replicates to use for approximating the expected value calculations. Defaults to 100.

`n_approx_outcomes_per_replicate`
: `integer` number of outcomes to use per replicate for approximation calculations. Defaults to 10000.

`seed`
: `integer` state of the random number generator for simulating outcomes when conducting the value of information analyses. Defaults to 500.

`n_threads`
: `integer` number of threads to use for computation.

`verbose`
: `logical` indicating if information should be printed during processing. Defaults to `FALSE`.

Details

The "approximately" optimal survey scheme is determined using a brute-force algorithm. Initially, all feasible (valid) survey schemes are identified given the survey costs and the survey budget (using `feasible_survey_schemes()`. Next, the expected value of each and every feasible survey scheme is approximated (using `approx_evdsi()`). Finally, the greatest expected value is identified, and all survey schemes that share this greatest expected value are returned. Due to the nature of this algorithm, it can take a very long time to complete.

Value

A `matrix` of `logical` (`TRUE`/ `FALSE`) values indicating if a site is selected in the scheme or not. Columns correspond to sites, and rows correspond to different schemes. If there is only one optimal survey scheme then the `matrix` will only contain a single row.

This matrix also has a `numeric` `"ev"` attribute that contains a matrix with the approximate expected values. Within this attribute, each row corresponds to a different survey scheme and each column corresponds to a different replicate.

Dependencies

Please note that this function requires the Gurobi optimization software (https://www.gurobi.com/) and the **gurobi** R package if different sites have different survey costs. Installation instruction are available online for Linux, Windows, and Mac OS (see https://support.gurobi.com/hc/en-us/articles/4534161999889-How-do-I-install-Gurobi-Optimizer).

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
```

```
total_budget <- sum(sim_sites$management_cost) * 0.5

# set total budget for surveying sites for conservation
# (i.e. 40% of the cost of surveying all sites)
survey_budget <- sum(sim_sites$survey_cost) * 0.4

## Not run:
# find optimal survey scheme using approximate method
# (using 10 replicates so that this example completes relatively quickly)
approx_opt_survey <- approx_optimal_survey_scheme(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_cost",
  "survey", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
  "target", total_budget, survey_budget)

# print result
print(approx_opt_survey)

## End(Not run)
```

`env_div_survey_scheme` *Environmental diversity survey scheme*

Description

Generate a survey scheme by maximizing the diversity of environmental conditions that are surveyed.

Usage

```
env_div_survey_scheme(
  site_data,
  cost_column,
  survey_budget,
  env_vars_columns,
  method = "mahalanobis",
  locked_in_column = NULL,
  locked_out_column = NULL,
  exclude_locked_out = FALSE,
  solver = "auto",
  verbose = FALSE
)
```

Arguments

`site_data`	`sf::sf()` object containing the candidate survey sites.

cost_column
: character name of the column in the argument to the argument to site_data that contains the cost for surveying each site. No missing (NA) values are permitted.

survey_budget
: numeric vector of maximum budgets for the survey schemes. No missing (NA) values are permitted.

env_vars_columns
: character vector names of the columns in the argument to site_data that contain numeric environmental variables. No missing (NA) values are permitted.

method
: character name of the distance metric to use for calculating environmental dissimilarity scores. See vegan::vegdist() documentation the method parameter for other available distance metrics and more information. No missing (NA) values are permitted. Defaults to "mahalanobis" for Mahalanobis distances.

locked_in_column
: character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked into the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked in.

locked_out_column
: character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked out of the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked out.

exclude_locked_out
: logical should locked out planning units be entirely excluded from the optimization process? Defaults to FALSE.

solver
: character name of the optimization solver to use for generating survey schemes. Available options include: "Rsymphony", "gurobi" and "auto". The "auto" method will use the Gurobi optimization software if it is available; otherwise, it will use the SYMPHONY software via the **Rsymphony** package. Defaults to "auto".

verbose
: logical indicating if information should be printed while generating survey scheme(s). Defaults to FALSE.

Details

The integer programming formulation of the environmental diversity reserve selection problem (Faith & Walker 1996) is used to generate survey schemes.

Value

A matrix of logical (TRUE/ FALSE) values indicating if a site is selected in a scheme or not. Columns correspond to sites, and rows correspond to different schemes.

Solver

This function can use the **Rsymphony** package and the Gurobi optimization software to generate survey schemes. Although the **Rsymphony** package is easier to install because it is freely available

on the The Comprehensive R Archive Network (CRAN), it is strongly recommended to install the Gurobi optimization software and the **gurobi** R package because it can generate survey schemes much faster. Note that special academic licenses are available at no cost. Installation instructions are available online for Linux, Windows, and Mac OS operating systems.

References

Faith DP & Walker PA (1996) Environmental diversity: on the best-possible use of surrogate data for assessing the relative biodiversity of sets of areas. *Biodiversity & Conservation*, **5**, 399–415.

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data
 x <- sf::st_as_sf(
   tibble::tibble(x = rnorm(4), y = rnorm(4),
                  v1 = c(0.1, 0.2, 0.3, 10), # environmental axis 1
                  v2 = c(0.1, 0.2, 0.3, 10), # environmental axis 2
                  cost = rep(1, 4)),
    coords = c("x", "y"))

# plot the sites' environmental conditions
plot(x[, c("v1", "v2")], pch = 16, cex = 3)

# generate scheme with a budget of 2
s <- env_div_survey_scheme(x, "cost", 2, c("v1", "v2"), "mahalanobis")

# print scheme
print(s)

# plot scheme
x$scheme <- c(s)
plot(x[, "scheme"], pch = 16, cex = 3)
```

---

evdci *Expected value of the decision given current information*

Description

Calculate the *expected value of the management decision given current information*. This metric describes the value of the management decision that is expected when the decision maker is limited to existing biodiversity data (i.e. survey data and environmental niche models).

Usage

```
evdci(
  site_data,
```

```
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
  feature_target_column,
  total_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
  prior_matrix = NULL
)
```

Arguments

site_data	sf::sf() object with site data.
feature_data	base::data.frame() object with feature data.
site_detection_columns	character names of numeric columns in the argument to site_data that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.
site_n_surveys_columns	character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.
site_probability_columns	character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.
site_management_cost_column	character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.
feature_survey_sensitivity_column	character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

total_budget
: numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.

site_management_locked_in_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column
: character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

prior_matrix
: numeric matrix containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to NULL such that prior data is calculated automatically using prior_probability_matrix().

Details

This function calculates the expected value and does not use approximation methods. As such, this function can only be applied to very small problems.

Value

A numeric value.

See Also

prior_probability_matrix().

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
total_budget <- sum(sim_sites$management_cost) * 0.5

# calculate expected value of management decision given current information
# using exact method
ev_current <- evdci(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
  "target", total_budget)

# print exact value
print(ev_current)
```

---

evdsi *Expected value of the decision given survey information*

Description

Calculate the *expected value of the management decision given survey information*. This metric describes the value of the management decision that is expected when the decision maker surveys a set of sites to help inform the decision.

Usage

```
evdsi(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  site_survey_scheme_column,
  site_survey_cost_column,
  feature_survey_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
  feature_target_column,
  total_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
  prior_matrix = NULL
)
```

Arguments

site_data	sf::sf() object with site data.
feature_data	base::data.frame() object with feature data.
site_detection_columns	character names of numeric columns in the argument to site_data that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.
site_n_surveys_columns	character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.
site_probability_columns	character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.
site_management_cost_column	character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric val-

ues that are equal to or greater than zero. No missing (NA) values are permitted in this column.

site_survey_scheme_column
: character name of logical (TRUE / FALSE) column in the argument to site_data that indicates which sites are selected in the scheme or not. No missing NA values are permitted. Additionally, only sites that are missing data can be selected or surveying (as per the argument to site_detection_columns).

site_survey_cost_column
: character name of column in the argument to site_data that contains costs for surveying each site. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

feature_survey_column
: character name of the column in the argument to feature_data that contains logical (TRUE / FALSE) values indicating if the feature will be surveyed in the planned surveys or not. Note that considering additional features will rapidly increase computational burden, and so it is only recommended to consider features that are of specific conservation interest. No missing (NA) values are permitted in this column.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of

each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

total_budget numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.

site_management_locked_in_column
character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column
character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

prior_matrix numeric matrix containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to NULL such that prior data is calculated automatically using prior_probability_matrix().

Details

This function calculates the expected value and does not use approximation methods. As such, this function can only be applied to very small problems.

Value

A numeric value.

See Also

prior_probability_matrix().

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
```

```
total_budget <- sum(sim_sites$management_cost) * 0.5

# create a survey scheme that samples the first two sites that
# are missing data
sim_sites$survey_site <- FALSE
sim_sites$survey_site[which(sim_sites$n1 < 0.5)[1:2]] <- TRUE

# calculate expected value of management decision given the survey
# information using exact method
ev_survey <- evdsi(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_site",
  "survey_cost", "survey", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
  "target", total_budget)

# print value
print(ev_survey)
```

`feasible_survey_schemes`

Find all feasible survey schemes

Description

Generate a `matrix` representing all possible different survey schemes given survey costs and a fixed budget.

Usage

```
feasible_survey_schemes(
  site_data,
  cost_column,
  survey_budget,
  locked_in_column = NULL,
  locked_out_column = NULL,
  verbose = FALSE
)
```

Arguments

`site_data`	`sf::sf()` object containing the candidate survey sites.
`cost_column`	`character` name of the column in the argument to the argument to `site_data` that contains the cost for surveying each site. No missing (`NA`) values are permitted.
`survey_budget`	`numeric` the maximum possible expenditure permitted for conducting surveys.

locked_in_column
: character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked into the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked in.

locked_out_column
: character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked out of the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked out.

verbose
: logical indicating if information should be printed while searching for feasible schemes. Defaults to FALSE.

Value

A matrix where each row corresponds to a different survey scheme, and each column corresponds to a different planning unit. Cell values are logical (TRUE / FALSE) indicating if a given site is selected in a given survey scheme.

Dependencies

Please note that this function requires the Gurobi optimization software (https://www.gurobi.com/) and the **gurobi** R package if different sites have different survey costs. Installation instruction are available online for Linux, Windows, and Mac OS (see https://support.gurobi.com/hc/en-us/articles/4534161999889-How-do-I-install-Gurobi-Optimizer).

Examples

```
## Not run:
# set seed for reproducibility
set.seed(123)

# simulate data
x <- sf::st_as_sf(tibble::tibble(x = rnorm(4), y = rnorm(4),
                                 cost = c(100, 200, 0.2, 1)),
                  coords = c("x", "y"))

# print data
print(x)

# plot site locations
plot(st_geometry(x), pch = 16, cex = 3)

# generate all feasible schemes given a budget of 4
s <- feasible_survey_schemes(x, "cost", survey_budget = 4)

# print schemes
print(s)

# plot first scheme
x$scheme_1 <- s[1, ]
```

```
plot(x[, "scheme_1"], pch = 16, cex = 3)

## End(Not run)
```

`fit_hglm_occupancy_models` *Fit hierarchical generalized linear models to predict occupancy*

Description

Estimate probability of occupancy for a set of features in a set of planning units. Models are fitted as hierarchical generalized linear models that account for for imperfect detection (following Royle & Link 2006) using JAGS (via `runjags::run.jags()`). To limit over-fitting, covariate coefficients are sampled using a Laplace prior distribution (equivalent to L1 regularization used in machine learning contexts) (Park & Casella 2008).

Usage

```
fit_hglm_occupancy_models(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_env_vars_columns,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  jags_n_samples = rep(10000, length(site_detection_columns)),
  jags_n_burnin = rep(1000, length(site_detection_columns)),
  jags_n_thin = rep(100, length(site_detection_columns)),
  jags_n_adapt = rep(1000, length(site_detection_columns)),
  jags_n_chains = rep(4, length(site_detection_columns)),
  n_folds = rep(5, length(site_detection_columns)),
  n_threads = 1,
  seed = 500,
  verbose = FALSE
)
```

Arguments

`site_data` `sf::sf()` object with site data.

`feature_data` `base::data.frame()` object with feature data.

`site_detection_columns`
`character` names of `numeric` columns in the argument to `site_data` that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.

site_n_surveys_columns
: character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

site_env_vars_columns
: character names of columns in the argument to site_data that contain environmental information for fitting updated occupancy models based on possible survey outcomes. Each column should correspond to a different environmental variable, and contain numeric, factor, or character data. No missing (NA) values are permitted in these columns.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

jags_n_samples
: integer number of sample to generate per chain for MCMC analyses. See documentation for the sample parameter in runjags::run.jags() for more information). Defaults to 10,000 for each feature.

jags_n_burnin
: integer number of warm up iterations for MCMC analyses. See documentation for the burnin parameter in runjags::run.jags() for more information). Defaults to 10,000 for each feature.

jags_n_thin
: integer number of thinning iterations for MCMC analyses. See documentation for the thin parameter in runjags::run.jags() for more information). Defaults to 100 for each feature.

jags_n_adapt
: integer number of adapting iterations for MCMC analyses. See documentation for the adapt parameter in runjags::run.jags() for more information). Defaults to 1,000 for each feature.

jags_n_chains
: integer total number of chains for MCMC analyses. See documentation for the n.chains parameter in runjags::run.jags() for more information). Defaults to 4 for each fold for each feature.

n_folds
: numeric number of folds to split the training data into when fitting models for each feature. Defaults to 5 for each feature.

n_threads
: integer number of threads to use for parameter tuning. Defaults to 1.

seed
: integer initial random number generator state for model fitting. Defaults to 500.

verbose
: logical indicating if information should be printed during computations. Defaults to FALSE.

Details

This function (i) prepares the data for model fitting, (ii) fits the models, and (iii) assesses the performance of the models. These analyses are performed separately for each feature. For a given feature:

1. The data are prepared for model fitting by partitioning the data using k-fold cross-validation (set via argument to `n_folds`). The training and evaluation folds are constructed in such a manner as to ensure that each training and evaluation fold contains at least one presence and one absence observation.
2. A model for fit separately for each fold (see `inst/jags/model.jags` for model code). To assess convergence, the multi-variate potential scale reduction factor (MPSRF) statistic is calculated for each model.
3. The performance of the cross-validation models is evaluated. Specifically, the TSS, sensitivity, and specificity statistics are calculated (if relevant, weighted by the argument to `site_weights_data`). These performance values are calculated using the models' training and evaluation folds. To assess convergence, the maximum MPSRF statistic for the models fit for each feature is calculated.

Value

A `list` object containing:

models `list` of `list` objects containing the models.

predictions `tibble::tibble()` object containing predictions for each feature.

performance `tibble::tibble()` object containing the performance of the best models for each feature. It contains the following columns:

feature name of the feature.

max_mpsrf maximum multi-variate potential scale reduction factor (MPSRF) value for the models. A MPSRF value less than 1.05 means that all coefficients in a given model have converged, and so a value less than 1.05 in this column means that all the models fit for a given feature have successfully converged.

train_tss_mean mean TSS statistic for models calculated using training data in cross-validation.

train_tss_std standard deviation in TSS statistics for models calculated using training data in cross-validation.

train_sensitivity_mean mean sensitivity statistic for models calculated using training data in cross-validation.

train_sensitivity_std standard deviation in sensitivity statistics for models calculated using training data in cross-validation.

train_specificity_mean mean specificity statistic for models calculated using training data in cross-validation.

train_specificity_std standard deviation in specificity statistics for models calculated using training data in cross-validation.

test_tss_mean mean TSS statistic for models calculated using test data in cross-validation.

test_tss_std standard deviation in TSS statistics for models calculated using test data in cross-validation.

test_sensitivity_mean mean sensitivity statistic for models calculated using test data in cross-validation.

test_sensitivity_std standard deviation in sensitivity statistics for models calculated using test data in cross-validation.

test_specificity_mean mean specificity statistic for models calculated using test data in cross-validation.

test_specificity_std standard deviation in specificity statistics for models calculated using test data in cross-validation.

Dependencies

This function requires the JAGS software to be installed. For information on installing the JAGS software, please consult the documentation for the **rjags** package.

References

Park T & Casella G (2008) The Bayesian lasso. *Journal of the American Statistical Association*, 103: 681–686.

Royle JA & Link WA (2006) Generalized site occupancy models allowing for false positive and false negative errors. *Ecology*, 87: 835–841.

Examples

```
## Not run:
# set seeds for reproducibility
set.seed(123)

# simulate data for 200 sites, 2 features, and 3 environmental variables
site_data <- simulate_site_data(n_sites = 30, n_features = 2, prop = 0.1)
feature_data <- simulate_feature_data(n_features = 2, prop = 1)

# print JAGS model code
cat(readLines(system.file("jags", "model.jags", package = "surveyvoi")),
    sep = "\n")

# fit models
# note that we use a small number of MCMC iterations so that the example
# finishes quickly, you probably want to use the defaults for real work
results <- fit_hglm_occupancy_models(
   site_data, feature_data,
   c("f1", "f2"), c("n1", "n2"), c("e1", "e2", "e3"),
   "survey_sensitivity", "survey_specificity",
   n_folds = rep(5, 2),
   jags_n_samples = rep(250, 2), jags_n_burnin = rep(250, 2),
   jags_n_thin = rep(1, 2), jags_n_adapt = rep(100, 2),
   n_threads = 1)

# print model predictions
print(results$predictions)

# print model performance
```

```
print(results$performance, width = Inf)

## End(Not run)
```

`fit_xgb_occupancy_models` *Fit boosted regression tree models to predict occupancy*

Description

Estimate probability of occupancy for a set of features in a set of planning units. Models are fitted using gradient boosted trees (via `xgboost::xgb.train()`).

Usage

```
fit_xgb_occupancy_models(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_env_vars_columns,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  xgb_tuning_parameters,
  xgb_early_stopping_rounds = rep(20, length(site_detection_columns)),
  xgb_n_rounds = rep(100, length(site_detection_columns)),
  n_folds = rep(5, length(site_detection_columns)),
  n_threads = 1,
  seed = 500,
  verbose = FALSE
)
```

Arguments

`site_data`	`sf::sf()` object with site data.
`feature_data`	`base::data.frame()` object with feature data.
`site_detection_columns`	`character` names of `numeric` columns in the argument to `site_data` that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.
`site_n_surveys_columns`	`character` names of `numeric` columns in the argument to `site_data` that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

site_env_vars_columns
: character names of columns in the argument to site_data that contain environmental information for fitting updated occupancy models based on possible survey outcomes. Each column should correspond to a different environmental variable, and contain numeric, factor, or character data. No missing (NA) values are permitted in these columns.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

xgb_tuning_parameters
: list object containing the candidate parameter values for fitting models. Valid parameters include: "max_depth", "eta", "lambda", "min_child_weight", "subsample", "colsample_by_tree", "objective". See documentation for the params argument in xgboost::xgb.train() for more information.

xgb_early_stopping_rounds
: numeric model rounds for parameter tuning. See xgboost::xgboost() for more information. Defaults to 10 for each feature.

xgb_n_rounds
: numeric model rounds for model fitting See xgboost::xgboost() for more information. Defaults to 100 for each feature.

n_folds
: numeric number of folds to split the training data into when fitting models for each feature. Defaults to 5 for each feature.

n_threads
: integer number of threads to use for parameter tuning. Defaults to 1.

seed
: integer initial random number generator state for model fitting. Defaults to 500.

verbose
: logical indicating if information should be printed during computations. Defaults to FALSE.

Details

This function (i) prepares the data for model fitting, (ii) calibrates the tuning parameters for model fitting (see xgboost::xgb.train() for details on tuning parameters), (iii) generate predictions using the best found tuning parameters, and (iv) assess the performance of the best supported models. These analyses are performed separately for each feature. For a given feature:

1. The data are prepared for model fitting by partitioning the data using k-fold cross-validation (set via argument to n_folds). The training and evaluation folds are constructed in such a manner as to ensure that each training and evaluation fold contains at least one presence and one absence observation.

2. A grid search method is used to tune the model parameters. The candidate values for each parameter (specified via `parameters`) are used to generate a full set of parameter combinations, and these parameter combinations are subsequently used for tuning the models. To account for unbalanced datasets, the `scale_pos_weight` `xgboost::xgboost()` parameter is calculated as the mean value across each of the training folds (i.e. number of absence divided by number of presences per feature). For a given parameter combination, models are fit using k-fold cross- validation (via `xgboost::xgb.cv()`) – using the previously mentioned training and evaluation folds – and the True Skill Statistic (TSS) calculated using the data held out from each fold is used to quantify the performance (i.e. `"test_tss_mean"` column in output). These models are also fitted using the `early_stopping_rounds` parameter to reduce time-spent tuning models. If relevant, they are also fitted using the supplied weights (per by the argument to `site_weights_data`). After exploring the full set of parameter combinations, the best parameter combination is identified, and the associated parameter values and models are stored for later use.
3. The cross-validation models associated with the best parameter combination are used to generate predict the average probability that the feature occupies each site. These predictions include sites that have been surveyed before, and also sites that have not been surveyed before.
4. The performance of the cross-validation models is evaluated. Specifically, the TSS, sensitivity, and specificity statistics are calculated (if relevant, weighted by the argument to `site_weights_data`). These performance values are calculated using the models' training and evaluation folds.

Value

A `list` object containing:

parameters `list` of `list` objects containing the best tuning parameters for each feature.

predictions `tibble::tibble()` object containing predictions for each feature.

performance `tibble::tibble()` object containing the performance of the best models for each feature. It contains the following columns:

- **feature** name of the feature.
- **train_tss_mean** mean TSS statistic for models calculated using training data in cross-validation.
- **train_tss_std** standard deviation in TSS statistics for models calculated using training data in cross-validation.
- **train_sensitivity_mean** mean sensitivity statistic for models calculated using training data in cross-validation.
- **train_sensitivity_std** standard deviation in sensitivity statistics for models calculated using training data in cross-validation.
- **train_specificity_mean** mean specificity statistic for models calculated using training data in cross-validation.
- **train_specificity_std** standard deviation in specificity statistics for models calculated using training data in cross-validation.
- **test_tss_mean** mean TSS statistic for models calculated using test data in cross-validation.
- **test_tss_std** standard deviation in TSS statistics for models calculated using test data in cross-validation.

test_sensitivity_mean mean sensitivity statistic for models calculated using test data in cross-validation.

test_sensitivity_std standard deviation in sensitivity statistics for models calculated using test data in cross-validation.

test_specificity_mean mean specificity statistic for models calculated using test data in cross-validation.

test_specificity_std standard deviation in specificity statistics for models calculated using test data in cross-validation.

Examples

```
## Not run:
# set seeds for reproducibility
set.seed(123)

# simulate data for 30 sites, 2 features, and 3 environmental variables
site_data <- simulate_site_data(
  n_sites = 30, n_features = 2, n_env_vars = 3, prop = 0.1)
feature_data <- simulate_feature_data(n_features = 2, prop = 1)

# create list of possible tuning parameters for modeling
parameters <- list(eta = seq(0.1, 0.5, length.out = 3),
                   lambda = 10 ^ seq(-1.0, 0.0, length.out = 3),
                   objective = "binary:logistic")

# fit models
# note that we use 10 random search iterations here so that the example
# finishes quickly, you would probably want something like 1000+
results <- fit_xgb_occupancy_models(
   site_data, feature_data,
   c("f1", "f2"), c("n1", "n2"), c("e1", "e2", "e3"),
   "survey_sensitivity", "survey_specificity",
   n_folds = rep(5, 2), xgb_early_stopping_rounds = rep(100, 2),
   xgb_tuning_parameters = parameters, n_threads = 1)

# print best found model parameters
print(results$parameters)

# print model predictions
print(results$predictions)

# print model performance
print(results$performance, width = Inf)

## End(Not run)
```

---

geo_cov_survey_scheme *Geographic coverage survey scheme*

Description

Generate a survey scheme by maximizing the geographic coverage of surveys.

Usage

```
geo_cov_survey_scheme(
  site_data,
  cost_column,
  survey_budget,
  locked_in_column = NULL,
  locked_out_column = NULL,
  exclude_locked_out = FALSE,
  solver = "auto",
  verbose = FALSE
)
```

Arguments

site_data	sf::sf() object containing the candidate survey sites.
cost_column	character name of the column in the argument to the argument to site_data that contains the cost for surveying each site. No missing (NA) values are permitted.
survey_budget	numeric vector of maximum budgets for the survey schemes. No missing (NA) values are permitted.
locked_in_column	character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked into the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked in.
locked_out_column	character (optional) name of the column in the argument to site_data that contains logical (TRUE/ FALSE) values indicating if certain sites should be locked out of the survey scheme. No missing (NA) values are permitted. Defaults to NULL such that no sites are locked out.
exclude_locked_out	logical should locked out planning units be entirely excluded from the optimization process? Defaults to FALSE.
solver	character name of the optimization solver to use for generating survey schemes. Available options include: "Rsymphony", "gurobi" and "auto". The "auto" method will use the Gurobi optimization software if it is available; otherwise, it will use the SYMPHONY software via the **Rsymphony** package. Defaults to "auto".
verbose	logical indicating if information should be printed while generating survey scheme(s). Defaults to FALSE.

Details

The integer programming formulation of the *p*-Median problem (Daskin & Maass 2015) is used to generate survey schemes.

Value

A `matrix` of `logical` (`TRUE`/ `FALSE`) values indicating if a site is selected in a scheme or not. Columns correspond to sites, and rows correspond to different schemes.

Solver

This function can use the **Rsymphony** package and the Gurobi optimization software to generate survey schemes. Although the **Rsymphony** package is easier to install because it is freely available on the The Comprehensive R Archive Network (CRAN), it is strongly recommended to install the Gurobi optimization software and the **gurobi** R package because it can generate survey schemes much faster. Note that special academic licenses are available at no cost. Installation instructions are available online for Linux, Windows, and Mac OS operating systems.

References

Daskin MS & Maass KL (2015) The p-median problem. In *Location Science* (pp. 21-45). Springer, Cham.

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data
 x <- sf::st_as_sf(
   tibble::tibble(x = rnorm(4), y = rnorm(4),
                  v1 = c(0.1, 0.2, 0.3, 10), # environmental axis 1
                  v2 = c(0.1, 0.2, 0.3, 10), # environmental axis 2
                  cost = rep(1, 4)),
    coords = c("x", "y"))

# plot the sites' locations
plot(st_geometry(x), pch = 16, cex = 3)

# generate scheme with a budget of 2
s <- geo_cov_survey_scheme(x, "cost", 2)

# print scheme
print(s)

# plot scheme
x$scheme <- c(s)
plot(x[, "scheme"], pch = 16, cex = 3)
```

greedy_heuristic_prioritization
Greedy heuristic prioritization

Description

Generate a prioritization for protected area establishment.

Usage

```
greedy_heuristic_prioritization(
  site_data,
  feature_data,
  site_probability_columns,
  site_management_cost_column,
  feature_target_column,
  total_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL
)
```

Arguments

site_data	sf::sf() object with site data.
feature_data	base::data.frame() object with feature data.
site_probability_columns	character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.
site_management_cost_column	character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.
feature_target_column	character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.
total_budget	numeric maximum expenditure permitted for conducting surveys and managing sites for conservation.
site_management_locked_in_column	character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked in for

(TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to NULL such that no sites are locked in.

site_management_locked_out_column

character name of the column in the argument to site_data that contains logical (TRUE / FALSE) values indicating which sites should be locked out for (TRUE) being managed for conservation or (FALSE) not. No missing (NA) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to NULL such that no sites are locked out.

Details

The prioritization is generated using a greedy heuristic algorithm. The objective function for this algorithm is calculated by: (i) estimating the probability that each species meets its target, and (ii) calculating the sum of these probabilities. Note that this function underpins the value of information calculations because it is used to assess a potential management decision given updated information on the presence of particular species in particular sites.

Value

A list containing the following elements:

x logical vector indicating if each site is selected for protection or not.

objval numeric value denoting the objective value for the prioritization.

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
 # (i.e. 50% of the cost of managing all sites)
total_budget <- sum(sim_sites$management_cost) * 0.5

# generate reserve selection prioritization
results <- greedy_heuristic_prioritization(
  sim_sites, sim_features,
  c("p1", "p2", "p3"),
  "management_cost",
  "target",
  total_budget
```

```
)

# print results
print(results)
```

`n_states` *Number of states*

Description

Calculate the total number of presence/absence states for a given number of sites and features.

Usage

```
n_states(n_sites, n_features)
```

Arguments

`n_sites`	`numeric` number of sites.
`n_features`	`numeric` number of features.

Value

A `numeric` value.

Examples

```
# calculate number of states for 3 sites and 2 features
n_states(n_sites = 2, n_features = 3)
```

`optimal_survey_scheme` *Optimal survey scheme*

Description

Find the optimal survey scheme that maximizes value of information. This function uses the exact method for calculating the expected value of the decision given a survey scheme.

Usage

```
optimal_survey_scheme(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  site_management_cost_column,
  site_survey_cost_column,
  feature_survey_column,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column,
  feature_target_column,
  total_budget,
  survey_budget,
  site_management_locked_in_column = NULL,
  site_management_locked_out_column = NULL,
  site_survey_locked_out_column = NULL,
  prior_matrix = NULL,
  n_threads = 1,
  verbose = FALSE
)
```

Arguments

`site_data` `sf::sf()` object with site data.

`feature_data` `base::data.frame()` object with feature data.

`site_detection_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.

`site_n_surveys_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.

`site_probability_columns`
: `character` names of `numeric` columns in the argument to `site_data` that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (`NA`) values are permitted in these columns.

site_management_cost_column
: character name of column in the argument to site_data that contains costs for managing each site for conservation. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

site_survey_cost_column
: character name of column in the argument to site_data that contains costs for surveying each site. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

feature_survey_column
: character name of the column in the argument to feature_data that contains logical (TRUE / FALSE) values indicating if the feature will be surveyed in the planned surveys or not. Note that considering additional features will rapidly increase computational burden, and so it is only recommended to consider features that are of specific conservation interest. No missing (NA) values are permitted in this column.

feature_survey_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_survey_specificity_column
: character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_target_column
: character name of the column in the argument to feature_data that contains the $target$ values used to parametrize the conservation benefit of managing of each feature. This column should have numeric values that are equal to or greater than zero. No missing (NA) values are permitted in this column.

`total_budget` `numeric` maximum expenditure permitted for conducting surveys and managing sites for conservation.

`survey_budget` `numeric` maximum expenditure permitted for conducting surveys.

`site_management_locked_in_column`
`character` name of the column in the argument to `site_data` that contains `logical` (`TRUE` / `FALSE`) values indicating which sites should be locked in for (`TRUE`) being managed for conservation or (`FALSE`) not. No missing (`NA`) values are permitted in this column. This is useful if some sites have already been earmarked for conservation, or if some sites are already being managed for conservation. Defaults to `NULL` such that no sites are locked in.

`site_management_locked_out_column`
`character` name of the column in the argument to `site_data` that contains `logical` (`TRUE` / `FALSE`) values indicating which sites should be locked out for (`TRUE`) being managed for conservation or (`FALSE`) not. No missing (`NA`) values are permitted in this column. This is useful if some sites could potentially be surveyed to improve model predictions even if they cannot be managed for conservation. Defaults to `NULL` such that no sites are locked out.

`site_survey_locked_out_column`
`character` name of the column in the argument to `site_data` that contains `logical` (`TRUE` / `FALSE`) values indicating which sites should be locked out (`TRUE`) from being selected for future surveys or (`FALSE`) not. No missing (`NA`) values are permitted in this column. This is useful if some sites will never be considered for future surveys (e.g. because they are too costly to survey, or have a low chance of containing the target species). Defaults to `NULL` such that no sites are locked out.

`prior_matrix` `numeric` `matrix` containing the prior probability of each feature occupying each site. Rows correspond to features, and columns correspond to sites. Defaults to `NULL` such that prior data is calculated automatically using `prior_probability_matrix()`.

`n_threads` `integer` number of threads to use for computation.

`verbose` `logical` indicating if information should be printed during processing. Defaults to `FALSE`.

Details

The optimal survey scheme is determined using a brute-force algorithm. Initially, all feasible (valid) survey schemes are identified given the survey costs and the survey budget (using `feasible_survey_schemes()`. Next, the expected value of each and every feasible survey scheme is computed (using `evdsi()`). Finally, the greatest expected value is identified, and all survey schemes that share this greatest expected value are returned. Due to the nature of this algorithm, it can take a very long time to complete.

Value

A `matrix` of `logical` (`TRUE`/ `FALSE`) values indicating if a site is selected in the scheme or not. Columns correspond to sites, and rows correspond to different schemes. If there is only one optimal survey scheme then the `matrix` will only contain a single row. This matrix also has a `numeric` `"ev"` attribute that contains the expected value of each scheme.

Dependencies

Please note that this function requires the Gurobi optimization software (https://www.gurobi.com/) and the **gurobi** R package if different sites have different survey costs. Installation instruction are available online for Linux, Windows, and Mac OS (see https://support.gurobi.com/hc/en-us/articles/4534161999889-How-do-I-install-Gurobi-Optimizer).

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# set total budget for managing sites for conservation
# (i.e. 50% of the cost of managing all sites)
total_budget <- sum(sim_sites$management_cost) * 0.5

# set total budget for surveying sites for conservation
# (i.e. 40% of the cost of managing all sites)
survey_budget <- sum(sim_sites$survey_cost) * 0.4

## Not run:
# find optimal survey scheme using exact method
opt_survey <- optimal_survey_scheme(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "management_cost", "survey_cost",
  "survey", "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity",
  "target", total_budget, survey_budget)

# print result
print(opt_survey)

## End(Not run)
```

---

prior_probability_matrix

Prior probability matrix

Description

Create prior probability matrix for the value of information analysis.

Usage

```
prior_probability_matrix(
  site_data,
  feature_data,
  site_detection_columns,
  site_n_surveys_columns,
  site_probability_columns,
  feature_survey_sensitivity_column,
  feature_survey_specificity_column,
  feature_model_sensitivity_column,
  feature_model_specificity_column
)
```

Arguments

site_data	sf::sf() object with site data.
feature_data	base::data.frame() object with feature data.
site_detection_columns	character names of numeric columns in the argument to site_data that contain the proportion of surveys conducted within each site that detected each feature. Each column should correspond to a different feature, and contain a proportion value (between zero and one). If a site has not previously been surveyed, a value of zero should be used.
site_n_surveys_columns	character names of numeric columns in the argument to site_data that contain the total number of surveys conducted for each each feature within each site. Each column should correspond to a different feature, and contain a non-negative integer number (e.g. 0, 1, 2, 3). If a site has not previously been surveyed, a value of zero should be used.
site_probability_columns	character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.
feature_survey_sensitivity_column	character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting a presence of each feature in a given site (i.e. the sensitivity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.
feature_survey_specificity_column	character name of the column in the argument to feature_data that contains probability of future surveys correctly detecting an absence of each feature in a given site (i.e. the specificity of the survey methodology). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column.

feature_model_sensitivity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting a presence of each feature in a given site (i.e. the sensitivity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

feature_model_specificity_column
: character name of the column in the argument to feature_data that contains probability of the initial models correctly predicting an absence of each feature in a given site (i.e. the specificity of the models). This column should have numeric values that are between zero and one. No missing (NA) values are permitted in this column. This should ideally be calculated using fit_xgb_occupancy_models() or fit_hglm_occupancy_models().

Value

A matrix object containing the prior probabilities of each feature occupying each site. Each row corresponds to a different feature and each column corresponds to a different site.

Examples

```
# set seeds for reproducibility
set.seed(123)

# load example site data
data(sim_sites)
print(sim_sites)

# load example feature data
data(sim_features)
print(sim_features)

# calculate prior probability matrix
prior_matrix <- prior_probability_matrix(
  sim_sites, sim_features,
  c("f1", "f2", "f3"), c("n1", "n2", "n3"), c("p1", "p2", "p3"),
  "survey_sensitivity", "survey_specificity",
  "model_sensitivity", "model_specificity")

# preview prior probability matrix
print(prior_matrix)
```

---

relative_site_richness_scores

Relative site richness scores

Description

Calculate relative site richness scores. Sites with greater scores are predicted to be more likely to contain more species. Note that these scores are relative to each other and scores calculated using different matrices cannot be compared to each other.

Usage

```
relative_site_richness_scores(site_data, site_probability_columns)
```

Arguments

site_data	sf::sf() object with site data.
site_probability_columns	character names of numeric columns in the argument to site_data that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (NA) values are permitted in these columns.

Details

The relative site richness scores are calculated using the following procedure:

1. Let J denote the set of sites (indexed by j), I denote the set of features (indexed by i), and x_{ij} denote the modeled probability of feature $i \in I$ occurring in sites $j \in J$.
2. Next, we will sum the values for each site: $y_j = \sum_{i \in I} x_{ij}$.
3. Finally, we will linearly rescale the y_j values between 0.01 and 1 to produce the scores.

Value

A numeric vector of richness scores. Note that these values are automatically rescaled between 0.01 and 1.

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data for 3 features and 4 planning units
x <- tibble::tibble(x = rnorm(4), y = rnorm(4),
                    p1 = c(0.095, 0.032, 0.5, 0.924),
                    p2 = c(0.023, 0.014, 0.4, 0.919),
                    p3 = c(0.075, 0.046, 0.9, 0.977))
x <- sf::st_as_sf(x, coords = c("x", "y"))

# print data,
# we can see that the fourth site has the highest modeled probabilities of
# occupancy across all species
print(x)
```

```
# plot sites' occupancy probabilities
plot(x[, c("p1", "p2", "p3")], pch = 16, cex = 3)

# calculate scores
s <- relative_site_richness_scores(x, c("p1", "p2", "p3"))

# print scores,
# we can see that site 4 has the highest richness score
print(s)

# plot sites' richness scores
x$s <- s
plot(x[, c("s")], pch = 16, cex = 3)
```

`relative_site_uncertainty_scores` *Relative site uncertainty scores*

Description

Calculate scores to describe the overall uncertainty of modeled species' occupancy predictions for each site. Sites with greater scores are associated with greater uncertainty. Note that these scores are relative to each other and uncertainty values calculated using different matrices cannot be compared to each other.

Usage

```
relative_site_uncertainty_scores(site_data, site_probability_columns)
```

Arguments

`site_data`	`sf::sf()` object with site data.
`site_probability_columns`	`character` names of `numeric` columns in the argument to `site_data` that contain modeled probabilities of occupancy for each feature in each site. Each column should correspond to a different feature, and contain probability data (values between zero and one). No missing (`NA`) values are permitted in these columns.

Details

The relative site uncertainty scores are calculated as joint Shannon's entropy statistics. Since we assume that species occur independently of each other, we can calculate these statistics separately for each species in each site and then sum together the statistics for species in the same site:

1. Let J denote the set of sites (indexed by j), I denote the set of features (indexed by i), and x_{ij} denote the modeled probability of feature $i \in I$ occurring in sites $j \in J$.

2. Next, we will calculate the Shannon's entropy statistic for each species in each site: $y_{ij} = -\big((x_i j log_2 x_{ij}) + (1 - x_i j log_2 1 - x_{ij})\big)$
3. Finally, we will sum the entropy statistics together for each site: $s_j = \sum_{i \in I} y_{ij}$

Value

A `numeric` vector of uncertainty scores. Note that these values are automatically rescaled between 0.01 and 1.

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data for 3 features and 5 sites
x <- tibble::tibble(x = rnorm(5), y = rnorm(5),
                    p1 = c(0.5, 0, 1, 0, 1),
                    p2 = c(0.5, 0.5, 1, 0, 1),
                    p3 = c(0.5, 0.5, 0.5, 0, 1))
x <- sf::st_as_sf(x, coords = c("x", "y"))

# print data,
# we can see that site (row) 3 has the least certain predictions
# because it has many values close to 0.5
print(x)

# plot sites' occupancy probabilities
plot(x[, c("p1", "p2", "p3")], pch = 16, cex = 3)

# calculate scores
s <- relative_site_uncertainty_scores(x, c("p1", "p2", "p3"))

# print scores,
# we can see that site 3 has the highest uncertainty score
print(s)

# plot sites' uncertainty scores
x$s <- s
plot(x[, c("s")], pch = 16, cex = 3)
```

`sim_data` *Simulated datasets*

Description

Simulated data for prioritizing sites for ecological surveys.

Usage

```
data(sim_features)

data(sim_sites)
```

Format

sim_sites `sf::sf()` object.

sim_features `tibble::tibble()` object.

.

Details

The simulated datasets provide data for six sites and three features. The sites can potentially acquired for protected area establishment. However, existing information on the spatial distribution of the features is incomplete. Only some of the sites have existing ecological survey data. To help inform management decisions, species distribution models have been fitted to predict the probability of each species occupying each site.

`sim_sites` This object describes the sites and contains the following data: cost of surveying the sites (`survey_cost` column), cost of acquiring sites for conservation (`management_cost` column), results from previous ecological surveys (`f1`, `f2`, `f3` columns), previous survey effort (`n1`, `n2`, `n3` columns), environmental conditions of the sites (`e1`, `e2` columns), and modeled probability of the features occupying the sites (`p1`, `p2`, `p3` columns).

`sim_features` This object describes the features and contains the following data: the name of each feature (`name` column), whether each feature should be considered in future surveys (`survey` column), the sensitivity and specificity of the survey methodology for each the sensitivity and specificity of the species distribution model for each feature (`model_sensitivity`, `model_specificity` columns), and the representation target thresholds for each feature (`target` column).

See Also

These datasets were simulated using `simulate_feature_data()` and `simulate_site_data()`.

Examples

```
# load data
data(sim_sites, sim_features)

# print feature data
print(sim_features, width = Inf)
# print site data
print(sim_sites, width = Inf)
```

`simulate_feature_data` *Simulate feature data*

Description

Simulate feature data for developing simulated survey schemes.

Usage

```
simulate_feature_data(n_features, proportion_of_survey_features = 1)
```

Arguments

`n_features`	`integer` number of features.
`proportion_of_survey_features`	`numeric` proportion of features that will be examined in the new surveys. Values must be between zero and one. Defaults to 1 such that all features should be surveyed.

Value

A `tibble::tibble()` object. It contains the following data:

`name character` name of each feature.

`survey logical` (`TRUE` / `FALSE`) values indicating if each feature should be examined in surveys or not.

`survey_sensitivity numeric` sensitivity (true positive rate) of the survey methodology for each features.

`survey_specificity numeric` specificity (true negative rate) of the survey methodology for each features.

`model_sensitivity numeric` specificity (true positive rate) of the occupancy models for each features.

`model_specificity numeric` specificity (true negative rate) of the occupancy models for each features.

`target numeric` target values used to parametrize the conservation benefit of managing of each feature (defaults to 1).

See Also

`simulate_site_data()`

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data
d <- simulate_feature_data(n_features = 5,
                           proportion_of_survey_features = 0.5)
# print data
print(d, width = Inf)
```

`simulate_site_data` *Simulate site data*

Description

Simulate site data for developing simulated survey schemes.

Usage

```
simulate_site_data(
  n_sites,
  n_features,
  proportion_of_sites_missing_data,
  n_env_vars = 3,
  survey_cost_intensity = 20,
  survey_cost_scale = 5,
  management_cost_intensity = 100,
  management_cost_scale = 30,
  max_number_surveys_per_site = 5,
  output_probabilities = TRUE
)
```

Arguments

`n_sites`	`integer` number of sites.
`n_features`	`integer` number of features.
`proportion_of_sites_missing_data`	`numeric` proportion of sites that do not have existing presence/absence data. Values must be between zero and one.
`n_env_vars`	`integer` number of environmental variables for simulating feature distributions. Defaults to 3.
`survey_cost_intensity`	`numeric` intensity of the costs of surveying sites. Larger values correspond to larger costs on average. Defaults to 20.

survey_cost_scale
: numeric value corresponding to the spatial homogeneity of the survey costs. Defaults to 5.

management_cost_intensity
: numeric intensity of the costs of average cost of managing sites for conservation. Defaults to 100.

management_cost_scale
: numeric value corresponding to the spatial homogeneity of the survey costs. Defaults to 30.

max_number_surveys_per_site
: integer maximum number of surveys per site in the simulated data. Defaults to 5.

output_probabilities
: logical value indicating if probability values of occupancy should be output or not. Defaults to TRUE.

Value

A sf::sf() object with site data. The "management_cost" column contains the site protection costs, and the "survey_cost" column contains the costs for surveying each site. Additionally, columns that start with (i) "f" (e.g. "f1") contain the proportion of times that each feature was detected in each site, (ii) "n" (e.g. "n1") contain the number of of surveys for each feature within each site, (iii) "p" (e.g. "p1") contain prior probability data, and (iv) "e" (e.g. "e1") contain environmental data. Note that columns that contain the same integer value (excepting environmental data columns) correspond to the same feature (e.g. "d1", "n1", "p1" contain data that correspond to the same feature).

See Also

simulate_feature_data()

Examples

```
# set seed for reproducibility
set.seed(123)

# simulate data
d <- simulate_site_data(n_sites = 10, n_features = 4, prop = 0.5)

# print data
print(d, width = Inf)

# plot cost data
plot(d[, c("survey_cost", "management_cost")], axes = TRUE, pch = 16,
     cex = 2)

# plot environmental data
plot(d[, c("e1", "e2", "e3")], axes = TRUE, pch = 16, cex = 2)

# plot feature detection data
plot(d[, c("f1", "f2", "f3", "f4")], axes = TRUE, pch = 16, cex = 2)
```

```
# plot feature survey effort
plot(d[, c("n1", "n2", "n3", "n4")], axes = TRUE, pch = 16, cex = 2)

# plot feature prior probability data
plot(d[, c("p1", "p2", "p3", "p4")], axes = TRUE, pch = 16, cex = 2)
```

---

surveyvoi *surveyvoi: Survey Value of Information*

Description

Decision support tool for prioritizing sites for ecological surveys based on their potential to improve plans for conserving biodiversity (e.g. plans for establishing protected areas). Given a set of sites that could potentially be acquired for conservation management – wherein some sites have previously been surveyed and other sites have not – it can be used to generate and evaluate plans for additional surveys. Specifically, plans for ecological surveys can be generated using various conventional approaches (e.g. maximizing expected species richness, geographic coverage, diversity of sampled environmental conditions) and by maximizing value of information. After generating plans for surveys, they can also be evaluated using value of information analysis. For further details, see Hanson et al. (2023).

Details

The package vignette provides a tutorial (accessible using the code `vignettes("surveyvoi")`). Also, please note that several functions depend on the 'Gurobi' optimization software (available from https://www.gurobi.com) and the **gurobi** R package (installation instructions available online for Linux, Windows, and Mac OS). Additionally, the JAGS software (available from https://mcmc-jags.sourceforge.io/) is required to fit hierarchical generalized linear models.

Citation

Please cite the *surveyvoi* package when using it in publications. To cite the package, please use:

Hanson, JO, McCune JL, Chadès I, Proctor CA, Hudgins EJ, & Bennett JR (2023) Optimizing ecological surveys for conservation. Journal of Applied Ecology, 60: 41–51.

Author(s)

Package authors:

- Jeffrey O. Hanson <jeffrey.hanson@uqconnect.edu.au> ORCID
- Iadine Chadès <iadine.chades@csiro.au> ORCID
- Emma J. Hudgins <emma.hudgins@mail.mcgill.ca> ORCID
- Joseph R. Bennett <joseph.bennett@carleton.ca> ORCID

References

Hanson, JO, McCune JL, Chadès I, Proctor CA, Hudgins EJ, & Bennett JR (2023) Optimizing ecological surveys for conservation. Journal of Applied Ecology, 60: 41–51.

See Also

Useful links:

- Package website (https://prioritizr.github.io/surveyvoi/)
- Source code repository (https://github.com/prioritizr/surveyvoi)
- Report bugs (https://github.com/prioritizr/surveyvoi/issues)

`weighted_survey_scheme` *Weighted survey scheme*

Description

Generate a survey scheme by selecting the set of sites with the greatest overall weight value, a maximum budget for the survey scheme.

Usage

```
weighted_survey_scheme(
  site_data,
  cost_column,
  survey_budget,
  weight_column,
  locked_in_column = NULL,
  locked_out_column = NULL,
  solver = "auto",
  verbose = FALSE
)
```

Arguments

`site_data`	`sf::sf()` object containing the candidate survey sites.
`cost_column`	`character` name of the column in the argument to the argument to `site_data` that contains the cost for surveying each site. No missing (`NA`) values are permitted.
`survey_budget`	`numeric` vector of maximum budgets for the survey schemes. No missing (`NA`) values are permitted.
`weight_column`	`character` name of the column in the argument to `site_data` with the weights for each site.

`locked_in_column`
: `character` (optional) name of the column in the argument to `site_data` that contains `logical` (`TRUE`/ `FALSE`) values indicating if certain sites should be locked into the survey scheme. No missing (`NA`) values are permitted. Defaults to `NULL` such that no sites are locked in.

`locked_out_column`
: `character` (optional) name of the column in the argument to `site_data` that contains `logical` (`TRUE`/ `FALSE`) values indicating if certain sites should be locked out of the survey scheme. No missing (`NA`) values are permitted. Defaults to `NULL` such that no sites are locked out.

`solver`
: `character` name of the optimization solver to use for generating survey schemes. Available options include: `"Rsymphony"`, `"gurobi"` and `"auto"`. The `"auto"` method will use the Gurobi optimization software if it is available; otherwise, it will use the SYMPHONY software via the **Rsymphony** package. Defaults to `"auto"`.

`verbose`
: `logical` indicating if information should be printed while generating survey scheme(s). Defaults to `FALSE`.

Details

Let J denote the set of sites (indexed by j), and let b denote the maximum budget available for surveying the sites. Next, let c_j represent the cost of surveying each site $j \in J$, and w_j denote the relative value (weight) for surveying each site $j \in J$. The set of sites with the greatest overall weight values, subject to a given budget can the be identified by solving the following integer programming problem. Here, x_j is the binary decision variable indicating each if site is selected in the survey scheme or not.

$$Maximize \sum_{j \in J} x_j w_i subject to \sum_{j \in J} x_j c_j \leq b$$

Value

A `matrix` of `logical` (`TRUE`/ `FALSE`) values indicating if a site is selected in a scheme or not. Columns correspond to sites, and rows correspond to different schemes.

Solver

This function can use the **Rsymphony** package and the Gurobi optimization software to generate survey schemes. Although the **Rsymphony** package is easier to install because it is freely available on the The Comprehensive R Archive Network (CRAN), it is strongly recommended to install the Gurobi optimization software and the **gurobi** R package because it can generate survey schemes much faster. Note that special academic licenses are available at no cost. Installation instructions are available online for Linux, Windows, and Mac OS operating systems.

Examples

```
# set seed for reproducibility
set.seed(123)
```

```
# simulate data
x <- sf::st_as_sf(
  tibble::tibble(x = rnorm(4), y = rnorm(4),
                 w = c(0.01, 10, 8, 1),
                 cost = c(1, 1, 1, 1)),
  coords = c("x", "y"))

# plot site' locations and color by weight values
plot(x[, "w"], pch = 16, cex = 3)

# generate scheme without any sites locked in
s <- weighted_survey_scheme(x, cost_column = "cost", survey_budget = 2,
                            weight_column = "w")

# print solution
print(s)

# plot solution
x$s <- c(s)
plot(x[, "s"], pch = 16, cex = 3)
```

Index

∗ **datasets**
    sim_data, 48

approx_evdsi, 3
approx_evdsi(), *15*
approx_near_optimal_survey_scheme, 7
approx_optimal_survey_scheme, 12

base::data.frame(), *4*, *8*, *12*, *19*, *22*, *27*, *31*, *37*, *40*, *44*

env_div_survey_scheme, 16
evdci, 18
evdsi, 21
evdsi(), *42*

feasible_survey_schemes, 25
feasible_survey_schemes(), *15*, *42*
fit_hglm_occupancy_models, 27
fit_hglm_occupancy_models(), *5*, *9*, *14*, *20*, *23*, *41*, *45*
fit_xgb_occupancy_models, 31
fit_xgb_occupancy_models(), *5*, *9*, *14*, *20*, *23*, *41*, *45*

geo_cov_survey_scheme, 34
greedy_heuristic_prioritization, 37

n_states, 39

optimal_survey_scheme, 39

prior_probability_matrix, 43
prior_probability_matrix(), *6*, *10*, *14*, *20*, *21*, *24*, *42*

relative_site_richness_scores, 45
relative_site_uncertainty_scores, 47
runjags::run.jags(), *27*, *28*

sf::sf(), *4*, *8*, *12*, *16*, *19*, *22*, *25*, *27*, *31*, *35*, *37*, *40*, *44*, *46*, *47*, *49*, *52*, *54*
sim_data, 48
sim_features *(*sim_data*)*, 48
sim_sites *(*sim_data*)*, 48
simulate_feature_data, 50
simulate_feature_data(), *49*, *52*
simulate_site_data, 51
simulate_site_data(), *49*, *50*
surveyvoi, 53
surveyvoi-package *(*surveyvoi*)*, 53

tibble::tibble(), *29*, *33*, *49*, *50*

vegan::vegdist(), *17*

weighted_survey_scheme, 54

xgboost::xgb.cv(), *33*
xgboost::xgb.train(), *31*, *32*
xgboost::xgboost(), *32*, *33*