

Implementation

Fitting g across size classes

Inputs

- SE trial data, observation columns, predictor columns
- CP trial data, observation columns, predictor columns
- Search schedules
- Size class factor
- Number of iterations

Run SEmodsacrosssizes

Run ThetaSEcreateacrosssizes

Pass output to SE table function (AICtabcreateSEmods)

Pass output to SE graphics functions

Allow user to select from the model fits for each size class

Store the choice of model for each size class

Run CPmodsacrosssizes

Run ThetaCPcreateacrosssizes

Pass output to CP table function (AICtabcreateCPmods)

Pass output to CP graphics functions

Allow user to select from the model fits for each size class

Store the choice of model for each size class

Run gcreateacrosssizes

Produces an array of g values with dimensionality (Niter, 1, Nss, Ncellcombos, Nsizeclasses)

Pass output to g table function

Estimating M across splits

Inputs

Array from gcreateacrosssizes

Proportion weighted area searched (PWAS) for each size class x turbine x search schedule combination

Carcass observations, split column

Run Mhatgenerator

Produces an array of \hat{M} values with dimensionality (Niter, Nss, Nunits, Nsplitcategories, Nsizeclasses)

Condense \hat{M} array to split category-level estimates using Mhatcondense

Pass output to \hat{M} table function (Mhattable), which allows for whole-facility expansion

Pass output to \hat{M} graphics function

Functions

SEmodssacrossizes

Inputs: SE trial data, SE observation columns, SE predictors, size class column, if k should be fixed and value

Actions: Fit all possible models for each size class

Output: list (length Nsizeclasses) of lists (each length NmodelsSE) of model fits

To Do: modularize the internal model fitting to its own function

ThetaSEcreateacrossizes

Inputs: SE trial data, SE predictors, size class column, list from SEmodssacrossizes, Niterations, if k should be fixed and value

Actions: draw Niter samples from the models for each cell for each size class

Output: multidimensional array [Niterations, 2, NcellSE, NmodelsSE, Nsizeclasses]

AICtabcreateSEmods

Inputs: list from SEmodssacrossizes, selection of what to sort on (NULL, "AIC", or "AICc")

Actions: create a model output table array (length = Nsizeclasses), each table has NmodelsSE rows, sort based on input

Output: list of sorted model tables for SE

SEgraphcreate

Inputs: SE data, SE predictors, array of theta SEs, Niterations, observation columns, size class column, r (size class index), j (model index)

Actions: create parameter plots and SE decay curves and compare to cell means

Outputs: plot of parameters and SE decay curves for each cell combo (compared to cell means) within a given size class and model combination

To Do: clean up code

CPmodssacrossizes

Inputs: CP trial data, CP predictors, size class column, unit of time to use, last time present column, first time absent column

Actions: Fit all possible models for each size class

Output: list (length Nsizeclasses) of lists (each length NmodelsCP) of model fits

To Do: modularize the internal model fitting to its own function

ThetaCPcreateacrossizes

Inputs: CP trial data, CP predictors, size class column, list from CPmodssacrossizes, Niterations

Actions: draw Niter samples from each model for each cell for each size class

Output: multidimensional array [Niterations, 2, NcellCP, NmodelsCP, Nsizeclasses]

AICtabcreateCPmods

Inputs: list from CPmodssacrossizes, selection of what to sort on (NULL, "AIC", or "AICc")

Actions: create a model output table array (length Nsizeclasses), each table has NmodelsCP rows, sort based on input

Output: list of sorted model tables for CP

CPgraphscreate

Inputs: CP model list, CP data, CP predictors, array of theta CPs, Niterations, time unit choice, size class column, last time present column, first time absent column, size class, model complexity (1 = intercept only, 2 = first term only, 3 = second term only, 4 = additive, 5 = interactive), which distribution to emphasize in the plot

Actions: create a K-M survival curve and the model-fitted curve for each cell combo for a given model and size class

Outputs: survival curves for each cell combo, with all four distributions plotted

To Do: clean up code

gcreateacrosssizes:

Inputs: CP trial data, SE trial data, Search Schedule data, CP predictors, SE predictors, array of CP thetas, array of SE thetas, list from CPmodsacrosssizes, CP models to use for each size, SE models to use for each size

Actions: calculate g (using gvec) for each search schedule x cell combination x size class

Output: multidimensional array of g values [Niter, 1, Nss, Ncellcombos, Nclasses]

gtablecreate

Inputs: g array, confidence interval width

Action: summarize the g iterations according to each size class, search schedule, and cell combo

Output: table of mean (with CI) gs

Mhatgenerator

Inputs: carcass observations, PWAS for each size class x turbine x search schedule combo, size class column, split column, unit column, search schedule column, seed for random number generator, CP predictors, SE predictors, CP trial data, SE trial data, g array from gcreateacrosssizes

Action: calculate \hat{M} values

-For each size class r (NOTE: r replaced q due to $q()$ being a function in R)

-For each split category l

-For each turbine k

-For each Search Schedule j

-For each cell i

-subset the inputs to X_{ijklr} (length 1), a_{jkr} (length 1), g_{ijr} (length Niter)

-Draw $\tilde{X}_{ijklr} \sim \text{Bin}\left(\frac{X_{ijklr}}{g_{ijr}}, g_{ijr}\right)$

-Calculate $\tilde{M}_{ijklr} = \text{round}\left(\frac{\tilde{X}_{ijklr}}{g_{ijr}}\right)$

-Sum \tilde{M} across all cells within the search schedule

-Calculate $\hat{M}_{jklr} = \frac{\sum_{i=1}^{N_{\text{cells}}} \tilde{M}_{ijklr}}{a_{jkr}}$

Output: multidimensional array of \hat{M} [Niter, Nss, Nunits, Nsplitcategories, Nsizeclasses]

Mhatcondense

Inputs: Mhataray

Action: condense across search schedules, units, and size classes

Output: matrix of Mhat values: rows = Niterations, columns = Nsplitcategories

Mhattable

Inputs: condensed (to split categories) Mhat, fraction of facility searched, confidence interval width

Action: summarize the Mhat iterations according to the split confidence intervals, for searched and whole

Output: table of mean (with CI) mortalities for the searched area and the whole facility

Mhatgraph

Inputs: condensed (to split categories) Mhat, fraction of facility searched, which split to plot

Action: create a distribution of simulated searched-area and whole-facility Mhats for each split category

Output: distributions of whole-facility Mhat for a given split category

To do: clean up the x axis situation

factorcombinations

Inputs: predictor variables, dataset

Action: creates a factor combination table for a CP or SE analysis

Output: factor combination table

to-do: generalize (base on make_egDat function)

crossmodelcells

Inputs: CP predictors, SE predictors, CP trial data, SE trial data

Action: creates a factor combination table across the CP and SE analyses

Output: factor table applicable to both analyses

to-do: generalize (base on make_egDat function)

pkfunction

Inputs: number of trials with no detection for each carcass, trial on which each carcass was found, parameters, number of parameters associated with p, groups, maximum number of misses for a carcass, combined p-k model matrix, if k should be fixed and value

Action: function is optimized over using optim()

Output: negative log likelihood of the observations given the parameters

gvec

Inputs: specific Search schedule, specific CP theta [Niterations, 2], specific CP distribution, specific SE theta [Niterations, 2]

Action: calculate g for a specific set of CP and SE parameters and a specific search schedule

Output: simulation of g values [Niterations, 1]

ppersist

Inputs: specific CP distribution, CP parameters, arrival times, search intervals

Action: calculates the probability that a carcass that arrives in the interval between t_arrive0 and t_arrive1 persists until t_search, using exact integrals

Output: probability of persistence to detection for each interval

logit

Inputs: single numeric value

Action: computes the logit

Output: logit of the single numeric value

alogit

Inputs: single numeric value

Action: computes anti-logit

Output: anti-logit of the single numeric value