This document outlines the GenEst package and application.

**GenEst App**

The GenEst app is structured as a classic shiny app, with two scripts: one covering the user interface (ui.R) and the other covering the server-side operations (server.R). In addition, we currently have a helper script (genestfunctions.R) that houses the GenEst functions (also outlined below), although that script will eventually be replaced via conversion of GenEst into an R package.

Shiny apps are structured to allow reactive programming (think of it as an active conversation) between the UI and the server via inputs (user-generated objects) and outputs (server-generated objects). Reactive programming is also supported by the use of reactive values objects within the server.

*Server-Side*

The server.R script contains the code that takes the inputs provided by the user, runs code, and returns outputs to the user.

Aside from preliminaries, the server code is held within the generic function returned by the server.R function. Within the generic function, the majority of the code constrained to only be executed once the user performs certain actions in the UI (which updates a specific input), which prevents code from running prematurely. The only code within the function that is not held back is the code that creates the disclaimer and the code that initializes the reactive values object.

The following outline of the server-side code is structured as

Section

User action

Server executions

Data Input

Search Efficiency Data File is uploaded

SE data file is read into R

Column names from the SE data are pulled in and updated in model input drop downs

SE data table is rendered for output

Carcass Persistence Data File is uploaded

CP data file is read into R

Column names from the CP data are pulled in and updated in model input drop downs

CP data table is rendered for output

Search Schedule Data File is uploaded

SS data file is read into R

Column names from the SS data are pulled in and updated in model input drop downs

SS data table is rendered for output

SS translation table is rendered for output

Carcass Observations Data File is uploaded

CO data file is read into R

Column names from the CO data are pulled in and updated in model input drop downs

CO data table is rendered for output

Meta Data File is uploaded

MD data file is read into R

MD data table is rendered for output

Search Efficiency Analysis

At least one SE observation column is selected

SE data table for analysis is rendered for output

Run SE Model button is pushed

The SE models are run using the inputted values

The SE theta is created using the model output and the inputted values

AIC tables are generated for each size class

The model and size class options are updated in the output drop downs

The specific AIC table and figure outputs are cleared (render NULL)

Generate AIC Table button is pushed

The AIC table for the user-inputted size class is rendered for output

Generate SE Figure button is pushed

The SE figure for the user-inputted size class and model is rendered for output

Populate Options button is pushed

As long as the SE model has been run, drop downs for model selection are provided for each size class

Carcass Persistence Analysis

The last time observed and first time absent CP observation columns are selected

CP data table for analysis is rendered for output

Run CP Model button is pushed

The CP models are run using the inputted values

The CP theta is created using the model output and the inputted values

AIC tables are generated for each size class

The model and size class options are updated in the output drop downs

The specific AIC table and figure outputs are cleared (render NULL)

Generate AIC Table button is pushed

The AIC table for the user-inputted size class is rendered for output

Generate CP Figure button is pushed

The CP figure for the user-inputted size class and model is rendered for output

Populate Options button is pushed

As long as the CP model has been run, drop downs for model selection are provided for each size class

Detection Probability Estimation

Estimate Detection Probability button is pushed

Detection probability is estimated for each size class

Detection probability table is rendered for output

Fatality Estimation

Estimate Total Carcasses button is pushed

Total number of carcasses is estimated for each user-inputted split category

Total carcass table and figure are rendered for output

*User Interface*

The ui.R script contains the code that generates the web document (written in HTML) that creates the user interface. The ui.R script consists of a single call to the navbarPage function, which creates the UI definition used by shiny to create the app. navbarPage creates a page with a top level navigation bar that allows toggling among tabs. Subsequent embedded calls to tabsetPanel create further articulation of the UI by allowing tabbed pages to occur on tabs of higher level pages.

The following outline of the UI code is structured according to the tabs from the main page on downward.

Home

A simple splash page, contains only the GenEst logo.

Data Input

Sidebar: file inputs

Main panel: tabset panel with further tabbing

Search Efficiency

Shows SE data

Carcass Persistence

Shows CP data

Search Schedule

Shows SS data

Carcass Observation

Shows CO data

Meta Data

Shows MD

Analyses

A tab page broken into tabs

General Inputs

Sidebar: general model inputs

Main panel: empty

Search Efficiency

Sidebar: model inputs

Main panel: tabset panel with further tabbing

Data

Shows data being modeled

Model Table

Shows AIC table for a given size class

Figure

Produces a figure for a given size class and selected model

Model Selection

Used to select a model for each size class

Options can only be populated once the model has been run

Carcass Persistence

Sidebar: model inputs

Main panel: tabset panel with further tabbing

Data

Shows data being modeled

Model Table

Shows AIC table for a given size class

Figure

Produces a figure for a given size class and selected model

Model Selection

Used to select a model for each size class

Options can only be populated once the model has been run

Detection Probability

Sidebar: model run button, search schedule translation table

Main panel: where the detection probability table is output

Fatality Estimation

Sidebar: model inputs

Main panel: tabset panel with further tabbing

Data

Shows data being modeled

Table

Produces the fatality table

Figure

Produces the fatality figure

About

GenEst logo and additional details about the app.

**Command Line Implementation**

*Fitting SE across size classes*

Inputs

SE trial data, observation columns, predictor columns

Size class column

Number of iterations

If k should be fixed, and if so, the value

Run SEmodsetsacrosssizes

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

*Fitting CP across size classes*

Inputs

CP trial data, observation columns, predictor columns

Size class column

Number of iterations

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

*Fitting g across size classes*

Inputs

Search Schedule data

SE and CP trial data

SE and CP predictors

SE and CP thetas

List of possible CP models

SE and CP model selections

Number of iterations

Run gcreateacrosssizes

Pass output to g table function

*Estimating M across splits*

Inputs

Search Schedule data

Array from gcreateacrosssizes

Carcass observations, split column, unit column, size class column, search schedule column

Number of iterations

SE and CP trial data

SE and CP predictors

Confidence level

Fraction of facility surveyed

Run Mhatgenerator

Condense array to split category-level estimates using Mhatcondense

Pass output to table function (Mhattable)

Pass output to graphics function

**Functions**

packageLoad [**temporary function for pre-package code**]

Inputs: None

Actions: load all dependencies

Output: None

SEmod

Inputs: equation for p, equation for k, factor combination table, SE trial data, SE observation columns, observation object (two-column table of “zeros” and “found”)

Actions: Fit a single model for a single size class

Output: model fit list

SEmodset

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

Actions: Fit all possible models for a single size class

Output: lists (length NmodelsSE) of model fit lists

SEmodsetsacrosssizes

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 fit lists

ThetaSEcreateacrosssizes

Inputs: SE trial data, SE predictors, SE size class column, list from SEmodsacrosssizes, 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 SEmodsacrosssizes, 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, SE observation columns, size class column, r (size class index), j (model index), index of the cellwise model

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 cellwise model) within a given size class and model combination

CPmod

Inputs: CP trial data, survival object, predictor equation to use, distribution to use

Actions: Fit single CP model

Output: model fit

CPmodset

Inputs: CP trial data (restricted to the size class of interest), CP predictors, last time present column, first time absent column

Actions: Fit all possible models for a given size class

Output: list (length NmodelsCP) of model fits

CPmodsetsacrosssizes

Inputs: CP trial data, CP predictors, CP size class column, 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

ThetaCPcreateacrosssizes

Inputs: CP trial data, CP predictors, size class column, list from CPmodsacrosssizes, 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 CPmodsacrosssizes, 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 (the model formula to use), 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

SSveccreate

Inputs: Search Schedule data

Actions: Determines the unique search schedules across the units

Outputs: Vector of unique search schedules

SStablereate

Inputs: Search Schedule data

Actions: Determines the unique search schedules across the units, puts into translation table

Outputs: Translation table for unique search schedules

gcreateacrosssizes:

Inputs: CP trial data, SE trial data, output from SSveccreate, Niterations, CP predictors, SE predictors, array of CP thetas, array of SE thetas, list from CPmodsetsacrosssizes, 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, Nss, Ncellcombos, Nclasses]

gtablecreate

Inputs: g array, confidence level

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

Output: table of mean (with CI) gs

DWPtablecreate

Inputs: Search Schedule data

Actions: Determines the DWP for each size x search schedule x unit combination

Output: DWP table (DWP for each size x search schedule x unit combination)

Mhatgenerator

Inputs: carcass observations, search schedule data, size class column, split column, unit column, search schedule column, CP predictors, SE predictors, CP trial data, SE trial data, g array from gcreateacrosssizes

Action: calculate 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 unit k

-For the search schedule j relevant to unit k

-For each cell i

-subset the inputs to (length 1), (length 1), (length Niter)

-Draw

-Calculate

-Sum across all cells within the search schedule

-Calculate

Output: multidimensional array of [Niter, Nunits, Nsplitcategories, Nsizeclasses]

Mhatcondense

Inputs: Mhataray

Action: condense across 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 with means and confidence intervals, for searched area and whole facility, and according to the split

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

factorcombinations

Inputs: predictor variables, dataset

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

Output: factor combination table

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

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: numeric value

Action: computes the logit

Output: logit of the numeric value

alogit

Inputs: numeric value

Action: computes anti-logit

Output: anti-logit of the numeric value

ModelNamer

Inputs: equation-style model name

Action: converts to text-style model name

Output: text-style model name

ModelNameReverser

Inputs: text-style model name

Action: converts to equation-style model name

Output: equation -style model name

SEAICorder

Inputs: SE model list

Action: determines the AICc order of the models within each size class

Output: list of vectors of orders (one vector of orders per each size class)

CPAICorder

Inputs: CP model list

Action: determines the AICc order of the models within each size class

Output: list of vectors of orders (one vector of orders per each size class)