Package ‘GenEst’

June 8, 2020

Title Generalized Mortality Estimator

Version 1.4.4

Date 2020-06-06

Description Command-line and 'shiny' GUI implementation of the GenEst models for estimating bird and bat mortality at wind and solar power facilities, following Dalthorp, et al. (2018) <doi:10.3133/tm7A2>.

Depends R (>= 3.5.0)

License CC0

Encoding UTF-8

LazyData true

Imports corpus, DT, gsl, gtools, hellno, htmltools, htmlwidgets (>= 1.5), lubridate, MASS, matrixStats, mvtnorm, Rcpp, shiny (>= 1.4.0), shinyjs, survival

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, shinytest, testthat

VignetteBuilder knitr

LinkingTo Rcpp

NeedsCompilation yes

Author Daniel Dalthorp [aut, cre],
Juniper Simonis [aut],
Lisa Madsen [aut],
Manuela Huso [aut],
Paul Rabie [aut],
Jeffrey Mintz [aut],
Robert Wolpert [aut],
Jared Studyvin [aut],
Franzi Korner-Nievergelt [aut]

Maintainer Daniel Dalthorp <`ddalthorp@usgs.gov`>

Repository CRAN

Date/Publication 2020-06-08 18:20:03 UTC

R topics documented:

`aicc` *Generic S3 function for summarizing AICc*

Description

Extract AICc values from `pkm`, `pkmSet`, `pkmSetSize`, `cpm`, `cpmSet`, and `cpmSetSize`.

Usage

```
aicc(x, ...)
```

Arguments

`x`	Model or list of models to extract AICc values from.
`...`	further arguments passed to or from other methods

Value

list of models sorted by AICc

`aicc.cpm` *Extract AIC and AICc for a carcass persistence model*

Description

S3 function for generating AIC for `cpm` objects

Usage

```
## S3 method for class 'cpm'
aicc(x, ...)
```

Arguments

`x`	Carcass persistence model (`cpm` objects)
`...`	further arguments passed to or from other methods

Value

AIC, AICc vector

Examples

```
  data(wind_RP)
  mod <- cpm(formula_l = l ~ Season, formula_s = s ~ Season,
           data = wind_RP$CP, left = "LastPresent", right = "FirstAbsent")
 aicc(mod)
```

`aicc.cpmSet` *Create the AICc tables for a set of carcass persistence models*

Description

S3 function to generate model comparison tables based on AICc values for a set of CP models generated by `cpmSet`

Usage

```
## S3 method for class 'cpmSet'
aicc(x, ..., quiet = FALSE, app = FALSE)
```

Arguments

`x`	Set of carcass persistence models fit to the same observations
`...`	further arguments passed to or from other methods
`quiet`	Logical indicating if messages should be printed
`app`	Logical indicating if the table should have the app model names

Value

AICc table

Examples

```
  data(wind_RP)
  mod <- cpmSet(formula_l = l ~ Season * Visibility, formula_s = s ~ Season,
           data = wind_RP$CP, left = "LastPresent", right = "FirstAbsent")
 aicc(mod)
```

`aicc.cpmSetSize` *Create the AICc tables for a list of sets of searcher efficiency models*

Description

S3 function to generate model comparison tables for lists of of sets of CP models of class `cpmSetSize`

Usage

```
## S3 method for class 'cpmSetSize'
aicc(x, ...)
```

Arguments

`x`	List of sets of CP models fit to the same observations
`...`	further arguments passed to or from other methods

Value

AICc table

Examples

```
cpmods <- cpm(formula_l = l ~ Visibility, data = wind_RP$CP,
  left = "LastPresent", right = "FirstAbsent", sizeCol = "Size",
  allCombos = TRUE)
aicc(cpmods)
```

`aicc.pkm` *extract AICc value from pkm object*

Description

extract AICc value from pkm object

Usage

```
## S3 method for class 'pkm'
aicc(x, ...)
```

Arguments

`x`	object of class `pkm`
`...`	further arguments passed to or from other methods

Value

Data frame with the formulas for p and k and the AICc of the model

---

`aicc.pkmSet` *Create the AICc tables for a set of searcher efficiency models*

Description

Generates model comparison tables based on AICc values for a set of pk models generated by `pkmSet`

Usage

```
## S3 method for class 'pkmSet'
aicc(x, ..., quiet = FALSE, app = FALSE)
```

Arguments

`x`	Set of searcher efficiency models fit to the same observations
`...`	further arguments passed to or from other methods
`quiet`	Logical indicating if messages should be printed
`app`	Logical indicating if the table should have the app model names

Value

AICc table

Examples

```
  data(wind_RP)
  mod <- pkmSet(formula_p = p ~ Season, formula_k = k ~ Season, data = wind_RP$SE)
 aicc(mod)
```

`aicc.pkmSetSize` *Create the AICc tables for a list of sets of searcher efficiency models*

Description

Generates model comparison tables based on AICc values for a set of pk models generated by `pkm` with `allCombos = TRUE` and a non-`NULL` `sizeCol`.

Usage

```
## S3 method for class 'pkmSetSize'
aicc(x, ...)
```

Arguments

`x`	List of set of searcher efficiency models fit to the same observations
`...`	further arguments passed to or from other methods

Value

AICc table

Examples

```
  data(wind_RP)
  mod <- pkmSet(formula_p = p ~ Season, formula_k = k ~ Season, data = wind_RP$SE)
 aicc(mod)
```

`aicc.pkmSize` *Create the AICc tables for a list of sets of searcher efficiency models*

Description

Generates model comparison tables based on AICc values for a set of pk models generated by `pkm` with `allCombos` = `FALSE` and a non-`NULL` `sizeCol`.

Usage

```
## S3 method for class 'pkmSize'
aicc(x, ...)
```

Arguments

`x`	List of set of searcher efficiency models fit to the same observations
`...`	further arguments passed to or from other methods

Value

AICc table

Examples

```
  data(wind_RP)
  mod <- pkmSet(formula_p = p ~ Season, formula_k = k ~ Season, data = wind_RP$SE)
 aicc(mod)
```

`app_content` *GenEst Information*

Description

HTML generators for app information and content

`disclaimerUSGS` creates the text for the USGS disclaimer.

Usage

```
createvtext(type = "Full")

gettingStartedContent()

aboutContent()

GenEstAuthors()

GenEstGUIauthors()

GenEstLicense()

GenEstAcknowledgements()

GenEstLogos()

disclaimersContent(appType = "base")

disclaimerUSGS()

disclaimerWEST(appType)
```

Arguments

`type`	"Full" or "Short" or "Name" or "NameDate"
`appType`	"base" (for local version) or "deploy" (for hosted version)

Value

Panels and text for displaying general information about GenEst

---

`app_download_functions` *GenEst app download funtions*

Description

Handle downloading of app data and figures

Usage

```
downloadCPFig(rv)

downloadCPmod(rv, input)

downloadSEFig(rv)

downloadSEmod(rv, input)

downloadgFig(rv, sc)

downloadMres(rv, input)

downloadMFig(rv, split = TRUE)

downloadTable(filename, tablename, csvformat)

downloadgres(rv, input)
```

Arguments

`rv`	the reactive values list
`input`	list of shiny input parameters
`sc`	size class
`split`	logical indicator to use the split or not
`filename`	the name for the file writing out
`tablename`	the name of the table in the rv list
`csvformat`	format for .csv file: "" or NULL for comma-separated, 2 for semi-colon separated

Value

download handler functions

---

app_msg_functions *GenEst App Messages*

Description

lists of messages used in the app

Usage

```
msgList()

clearNotifications(msgs = msgList(), clear = TRUE)

msgModRun(msgs, modelType, clear = TRUE)

msgModDone(msgs, rv, type = "SE", clear = TRUE)

msgModPartialFail(mods, type = "SE")

msgSampleSize(mods)

msgModWarning(mods, type = "SE", rv = NULL)

msgModSENobs(rv)

msgModFail(mods, type = "SE", special = NULL)

msgSSavgFail(msgs, rv, clear = TRUE)

msgSSinputFail(msgs, rv, clear = TRUE)

msgSplitFail(type = NULL)

msgFracNote(fracNote)
```

Arguments

`msgs`	message list
`clear`	logical indicator if clearing should happen.
`modelType`	"SE", "CP", "g", or "M"
`rv`	reactive values list
`type`	"SE", "CP", "M", "split", or "g"
`mods`	Set Size list of models
`special`	indicator of a special type of message
`fracNote`	the note regarding the input

`app_output_utilities` *app utilities for formatting text, tables, figs, etc. for display*

Description

app utilities for formatting text, tables, figs, etc. for display

Usage

```
classText(rv, type = "SE")

estText(rv, type = "SE")

setNotSuspending(output, dontSuspend)

reNULL(x, toNULL)

initialOutput(rv, output)

setFigW(modelSet)

setFigH(modelSet, modType = "SE")
```

Arguments

`rv`	Reactive values list for the GenEst GUI.
`type`	Model type, either "SE" or "CP" or "g".
`output`	`output` list to have elements `dontSuspend` (re)set to having `suspendWhenHidden = FALSE`.
`dontSuspend`	Names of elements in `output` to (re)set to having `suspendWhenHidden = FALSE`.
`x`	`list` object to have elements `toNULL` reset to `NULL`.
`toNULL`	Names of elements in `x` to reset to `NULL`.
`modelSet`	Model set of class `cpmSet` or `pkmSet`.
`modType`	"SE" or "CP"

`app_panels` *app panel utility functions*

Description

functions for formatting and displaying app panels

Usage

```
dataTabPanel(dataType)

selectedDataPanel(modType)

modelOutputPanel(outType)
```

Arguments

`dataType`	Toggle control for the model type of the panel. One of "SE", "CP", "SS", "DWP", or "CO".
`modType`	Toggle control for the model type of the panel. One of "SE", "CP", or "g".
`outType`	Toggle control for the model type of the panel. One of "SEFigures", "SEEstimates", "SEModComparison", "SEModSelection", "CPFigures", "CPEstimates", "CPModComparison", "CPModSelection", "gFigures", or "gSummary".

`app_server` *The GenEst server definition function*

Description

This suite of functions defines the server-side program for the GenEst user interface (UI). See the "GenEst Graphic User Interface" vignette for a more complete detailing of the codebase underlying the GenEst UI.

`GenEstServer`: main server function expressed within the application.

Usage

```
GenEstServer(input, output, session)

reaction(eventName)

reactionMessageRun(eventName)

reactionMessageDone(eventName)

eventReaction(eventName, rv, input, output, session)
```

Arguments

`input`	`input` list for the GenEst GUI.
`output`	`output` list for the GenEst GUI.
`session`	Environment for the GenEst GUI.

`eventName`	Character name of the event. One of "clear_all", "file_SE", "file_SE_clear", "file_CP", "file_CP_clear", "file_SS", "file_SS_clear", "file_DWP", "file_DWP_clear", "file_CO", "file_CO_clear", "class", "obsSE", "predsSE", "run_SE", "run_SE_clear", "outSEclass", "outSEp", "outSEk", "ltp", "fta", "predsCP", "run_CP", "run_CP_clear", "outCPclass", "outCPdist", "outCPl", "outCPs", "run_M", "run_M_clear", "split_M", "split_M_clear", "transpose_split", "run_g", "run_g_clear", or "outgclass".
`rv`	reactive variable

Details

`GenEstServer` is used as the main server function, and is therefore included in the `server.R` script of the app. This function is not used in a standard R function sense, in that it does not return a value and is not used on its own to have side effects. The code of the function has two parts:

1. preamble that defines all the necessary variables and options
2. `observeEvent` calls, one for each event in the application. Each call to `observeEvent` includes the `eventExpr` (event expression) as the first argument and the `handlerExpr` (handler expression) as the second argument, which is an evaluated (via `eval`) block of code returned from `reaction` for the specific event, as well as any other control switch arguments needed (such as `ignoreNULL`).

`app_ui` *Create the GenEst User Interface HTML*

Description

This suite of functions in `app_ui.R` create the HTML code underlying the GenEst user interface (UI). See the "GenEst Graphic User Interface" vignette for a more complete detailing of the codebase underlying the GenEst UI.

`GenEstUI`: whole application. Calls `dataInputPanel`, `analysisPanel`, and `helpPanel`.

Usage

```
GenEstUI(appType = "base")

dataInputPanel()

dataInputSidebar()

loadedDataPanel()

analysisPanel()

GeneralInputsPanel()

GeneralInputSidebar()
```

```
SEPanel()

SESidebar()

SEMainPanel()

CPPanel()

CPSidebar()

CPMainPanel()

MPanel()

MSidebar()

MMainPanel()

gPanel()

gSidebar()

gMainPanel()

helpPanel(appType = "base")

gettingStartedPanel()

downloadsPanel()

aboutPanel()

disclaimersPanel(appType = "base")
```

Arguments

`appType`	Toggle control for the app, `"base"` for local versions or `"deploy"` for hosted version. Currently only differentiates the disclaimer text.

Details

Currently there are few differences between the local and deployed versions of GenEst, and the appType toggle is only included as an argument for functions that can produce different versions of the HTML. At this point, the only content that is different is the disclaimer text on the Help panel.

Value

Each function returns a string of HTML code, either as a `"shiny.tag.list"` object (in the case of `GenEstUI`) or a `"shiny.tag"` object (in the case of the other functions).

`GenEstUI`: Full GenEst user interface.

`app_ui_utilities` *HTML parameters*

Description

utility functions for simple HTML tasks

Usage

```
navbar()

style(...)

ol(...)

ul(...)

li(...)

b(...)

u(...)

small(...)

big(text = NULL)

center(text = NULL)

GenEstInlineCSS(...)

GenEstShinyJS(...)

cButtonStyle(buttonType = "single")
```

Arguments

`...`	attributes and children of the element
`text`	text to wrap in the tag
`buttonType`	"single" (for clearing a single component) or "all" (for clearing everything).

`app_utilities` *app utilities*

Description

utility functions for simple app rv management

Usage

```
initialReactiveValues()

reVal(rv, toReVal)

setkNeed(rv)

updateColNames_size(rv)

selectData(data, cols)

modNameSplit(modNames, pos)

prepSizeclassText(sizeclasses)

modNamePaste(parts, type = "SE", tab = FALSE)

plotNA(type = "model")

updateSizeclasses(data, sizeCol)

pickSizeclass(sizeclasses, choice)

updatesizeCol(sizeCol, colNames_size)
```

Arguments

`rv`	Reactive values list for the GenEst GUI, created by `initialReactiveValues`, which calls `reactiveValues`
`toReVal`	Names of elements in `rv` to reset to their factory setting (as defined by `initialReactiveValues`).
`data`	data table
`cols`	column names to select
`modNames`	names of the model to be split off
`pos`	position in the name to split off
`sizeclasses`	names of the carcass classes
`parts`	the component parts of the model's name
`type`	"SE" or "CP"

`tab`	logical for if it's the table output for CP
`sizeCol`	carcass class column name
`choice`	carcass class chosen
`colNames_size`	updated vector of size column names in all needed tables

---

`app_widgets` *Create and manage widgets for data input, function execution, data output*

Description

This is a generalized function for creating a data input widget used in the GenEst GUI, based on the data type (`dataType`). Included within the widget is a conditional panel that allows removal of the specific data file (and clearing of all downstream models) once it has been loaded.

Usage

```
dataInputWidget(dataType)

dataDownloadWidget(set)

modelInputWidget(inType)

widgetMaker(Condition, Name, Fun, Label, Args)

modelRunWidget(modType)

preTextMaker(modType)

modelOutputWidget(modType)

splitButtonWidget()

modelSelectionWidget(mods, modType)

modelSelectionWidgetHeader(mods)

modelSelectionWidgetRow(mods, modType, sci)

kFixedWidget(sizeclasses)

kFixedWidgetHeader(sizeclasses)

kFixedWidgetRow(sizeclasses, sci)
```

Arguments

`dataType`	Toggle control for the model type of the widget. One of "SE", "CP", "SS", "DWP", or "CO".
`set`	Name of data set. One of "RP", "RPbat", "cleared", "powerTower", "PV", "trough", "mock"
`inType`	Toggle control for the input type of the widget. One of "nsim", "CL", "class", "obsSE", "predsSE", "kFixed", "ltp", "fta", "predsCP", "dist", "xID", "frac", "DWPCol", "COdate", "gSearchInterval", or "gSearchMax".
`Condition`	Condition under which the widget is present to the user.
`Name`	Name (id) of the widget created.
`Fun`	Function name (as character) used to create the widget.
`Label`	Label presented to the user in the UI for the widget.
`Args`	List of any additional arguments to be passed to the widget creating function.
`modType`	Toggle control for the model type of the widget. One of "SE", "CP", "M", or "g".
`mods`	Model Set Size object (from the reactive values list).
`sci`	Name of carcass class element
`sizeclasses`	Vector of carcass class names (from the reactive values list).

Value

HTML for the data input widget.

`averageSS` *Tabulate an average search schedule from a multi-unit SS data table*

Description

Given a multi-unit Search Schedule data table, produce an average search schedule for use in generic detection probability estimation.

Usage

```
averageSS(data_SS, SSdate = NULL)
```

Arguments

`data_SS`	a multi-unit SS data table, for which the average interval will be tabulated. It is assumed that `data_SS` is properly formatted, with a column of search dates and a column of 1s and 0s for each unit indicating whether the unit was searched on the given date). Other columns are optional, but optional columns should not all contain at least on value that is not a 1 or 0.
`SSdate`	Column name for the date searched data (optional). if no `SSdate` is provided, `data_SS` will be parsed to extract the dates automatically. If there is more than one column with dates, then an error will be thrown and the user will be required to provide the name of the desired dates column.

Value

vector of the average search schedule

Examples

```
data(mock)
avgSS <- averageSS(mock$SS)
```

`calcg` *Calculate detection probability for given SE and CP parameters and search schedule.*

Description

Calculate detection probability (g) given SE and CP parameters and a search schedule.

The g given by `calcg` is a generic aggregate detection probability and represents the probability of detecting a carcass that arrives at a (uniform) random time during the time spanned by the search schedule for the the given SE and CP parameters. This differs from the GenEst estimation of g when the purpose is to estimate total mortality (M), in which case the detection probability varies with carcass arrival interval and is difficult to summarize statistically. `calcg` provides a useful "big picture" summary of detection probability, but would be difficult to work with for estimating M with precision.

Usage

```
calcg(days, param_SE, param_CP, dist)
```

Arguments

`days`	Search schedule (vector of days searched)
`param_SE`	numeric array of searcher efficiency parameters (p and k); must have the name number of rows as the `param_CP`.
`param_CP`	numeric array of carcass persistence parameters (a and b) must have the name number of rows as the `param_SE`.
`dist`	distribution for the CP model

Value

a vector of detection probabilities for each

---

calcRate	*Estimate the number of fatalities in each search interval throughout the monitoring period.*

Description

A carcass that is observed in a given search may have arrived at any time prior to that search, so carcass discovery time is often not a reliable estimate of carcass arrival time. For each observed carcass, `calcRate` takes into account the estimated probability of arrival in each possible arrival interval, adjusts by detection probability, and sums to estimate the estimated number of carcass arrivals in every search interval.

Usage

```
calcRate(M, Aj, days = NULL, searches_carcass = NULL, data_SS = NULL)
```

Arguments

`M`	Numeric array (ncarc x nsim) of estimated number of fatalities by observed carcass and simulation rep
`Aj`	Integer array (ncarc x nsim) of simulated arrival intervals for each observed carcass. Arrival intervals are given as integers j, indicating that the given carcass (indexed by row) arrived in the jth search interval in the given simulation rep (indexed by column). Arrival interval indices (j) are relative to indexed carcasses' search schedules.
`days`	Vector of all dates that at least one unit was searched. Format is the number of days since the first search. For example, days = c(0, 7, 14, 28, 35) for a simple 7-day search schedule in which searches were conducted every once per week on the same day for 5 weeks. Not all units need be searched on every search date.
`searches_carcass`	An ncarc x length(days) array of 0s and 1s to indicate searches in which the indexed carcass could have been found. For example, row i = `c(1,0,1,0,1)` indicates that the search schedule for the location (unit) where carcass i was found would be `days[c(1,3,5)]`.
`data_SS`	`prepSS` object that contains formatted data for calculating splits. Optional argument. Alternatively, user may provide `days` and `searches_carcass`.

Value

Numeric array (nsim x nsearch) of estimated fatalities in each search interval. NOTE: The search at time t = 0 does not correspond to an interval, and all carcasses found at that time are assumed to have arrived prior to the monitoring period and are not included in mortality estimates so `nsearch` = `length(days) -1`.

`calcSplits` *Estimate the number of fatalities by up to two splitting covariates*

Description

Total mortality can be split into sub-categories, according to various splitting covariates such as species, visibility class, season, site, unit, etc. Given the carcass search data, estimated mortalities, and splitting covariates, `calcSplits()` gives the "splits" or summaries the estimated mortalities by levels of the splitting covariates. For example, user may specify `"season"` and `"species"` as splitting variables to see estimated mortalities by season and species. Input would be arrays of estimated mortalities and arrival intervals when `ncarc` carcass have been discovered and uncertainty in mortality estimates is captured via simulation with `nsim` simulation draws.

Usage

```
calcSplits(M, split_CO = NULL, data_CO = NULL, split_SS = NULL,
  data_SS = NULL, split_time = NULL, ...)
```

Arguments

`M`	`estM` object, containing numeric array (ncarc x nsim) of estimated mortalities and other pieces
`split_CO`	Character vector of names of splitting covariates to be found in the `data_CO` data frame. No more than two `split_CO` variables are allowed. Use `split_CO = NULL` if no CO splits are desired.
`data_CO`	data frame that summarizes the carcass search data and must include columns specified by the `split_CO` arg. Each row includes search and discovery parameters associated with a single observed carcass. Columns include carcass ID, carcass discovery date, unit, and any number of covariates. `data_CO` is required if and only if `split_CO` is non-NULL.
`split_SS`	Character string giving the name of a splitting covariate in the `data_SS` list, with `data_SS[[split_SS]]` describing characteristics of the search intervals (e.g., "season"). Note that `length(data_SS[[split_SS]]` must equal `length(data_SS$days) -1` because no inference is made about carcass arrivals prior to time t = 0, and the "interval" prior to t = 0 is not taken as a "search interval." If no `split_SS` split is desired, use `split_SS = NULL`.
`data_SS`	Search schedule data
`split_time`	Numeric vector that defines time intervals for splits. Times must be numeric, strictly increasing, and span the monitoring period [0, `max(data_SS$days)`]. If no `split_time` is desired, use `split_time = NULL`. If `split_time` is NULL and `split_SS` is not NULL, `data_SS` is required.
`...`	arguments to be passed down

Details

Arrival intervals (`Aj`) are given as integers, j, that indicate which search interval the given carcass (indexed by row) arrived in the given simulation draw (indexed by column). Arrival interval indices (j) are relative to indexed carcasses' search schedules.

No more than two splitting variables (`split_CO`, `split_SS`, and `split_time`) in total may be used. `split_CO` variables describe qualitative characteristics of the observed carcasses or where they were found. Some examples include searcher (DHD, JPS, MMH), carcass size (S, M, L), species, age (fresh/dry or immature/mature), unit, visibility class (easy, moderate, difficult), etc.

`split_SS` variables describe characteristics of the search intervals, such as season (spring, summer, fall, winter) or treatment (pre- or post-minimization). Each search interval is assigned a level of the `split_SS` variable. For example, for a search schedule with 5 searches (including a search at t = 0), and the `split_SS` variable would have values for each of the 4 search intervals. The levels of the `split_SS` must be in contiguous blocks. For example, `season = c("S","S","F","F")` would be acceptable, but `season = c("S","F","S","F")` would not be.

`split_time` variables are numeric vectors that split the monitoring period into distinct time intervals. For example, `split_time = c(0,30,60,90,120)` would split the 120 monitoring period into 30-day intervals, and `calcSplits()` would return mortality estimates for each of the intervals.

Value

An object of class `splitFull` is returned. If one splitting covariate is given, then the output will be an array of estimated mortality in each level of the splitting covariate, with one row for each covariate level and one column for each simulation draw. If two splitting covariates are given, output will be a list of arrays. Each array gives the estimated mortalities for one level of the second splitting covariate and all levels of the first splitting covariate.

Objects of class `splitFull` have attributes `vars` (which gives the name of the splitting covariate(s)) and `type` (which specifies whether the covariate(s) are of type `split_CO`, `split_SS`, or `split_time`). A summary of a resulting `splitFull` object is returned from the S3 function `summary(splits,CL = 0.90,...)`, which gives the mean and a 5-number summary for each level of each covariate. The 5-number summary includes the alpha/2, 0.25, 0.5, 0.75, and 1 - alpha/2 quantiles, where alpha = 1 - CL. A graph summarizing the results can be drawn using `plot(splits,CL,...)`, which gives a graphical representation of the `summary`.

Examples

```
model_SE <- pkm(p ~ 1, k ~ 1, data = wind_RPbat$SE)
model_CP <- cpm(l ~ 1, s ~ 1, data = wind_RPbat$CP, dist = "weibull",
  left = "LastPresent", right = "FirstAbsent")
Mhat <- estM(nsim = 1000, data_CO = wind_RPbat$CO,
  data_SS = wind_RPbat$SS, data_DWP = wind_RPbat$DWP,
  model_SE = model_SE, model_CP = model_CP,
  unitCol = "Turbine", COdate = "DateFound")

M_spp <- calcSplits(M = Mhat, split_CO = "Species",
  data_CO = wind_RPbat$CO)
summary(M_spp)
plot(M_spp)
```

`calcTsplit` *Estimate the number of fatalities by time interval*

Description

`calcTsplit()` is a lower-level function that requires the output of `calcRate` as input. See `calcSplits` for a more powerful, convenient, and flexible alternative.

Usage

```
calcTsplit(rate, days, tsplit)
```

Arguments

`rate`	Array (nsim x nsearch) of arrival rates as number of fatalities per search interval. Typically, `rate` will be the return value of the `calcRate` function.
`days`	A vector of times representing search dates when at least one unit was searched. Times are formatted as number of days since the first search, e.g., c(0, 7, 14, 28, 35) would indicate a schedule in at least one unit was searched every 7 days.
`tsplit`	A vector of times that splits the monitoring period into a set of time intervals for which `calcTsplit` will estimate the number of fatalities. For example, if `tsplit = c(0,14,19,35)`, then `calcTsplit` estimates the number of fatalities occuring in interval (0, 14], (14, 19], and (19, 35]. Times in `tsplit` must be increasing and between 0 and max(days), inclusive.

Value

A numeric array with dimensions `dim = c(length(tsplit) -1,nsim)` giving the estimated number of fatalities that occured in each time interval.

`checkComponents` *Check for model components*

Description

Check if all component terms and interactions are included in a formula. Terms are automatically alphabatized within interactions.

Usage

```
checkComponents(formula)
```

Arguments

`formula`	A formula object

Value

a logical regarding complete set of terms and interactions

`checkDate` *Checks whether a vector of data can be interpreted as dates*

Description

Checks whether the dates are in a standard format and sensible. If so, function returns the dates converted to ISO 8601 yyyy-mm-dd format. Acceptable formats are yyyy-mm-dd, yyyy/mm/dd, mm/dd/yyyy, and dd/mm/yyyy. If format is mm/dd/yyyy or dd/mm/yyyy, the dates must be interpretable unambiguously. Also, dates must be later than 1900-01-01. This additional check provides some protection against common data entry errors like entering a year as 0217 or 1017 instead of 2017.

Usage

```
checkDate(testdate)
```

Arguments

`testdate`	Date(s) to check and format.

Value

dates formatted as yyyy-mm-dd (if possible) or NULL (if some value is not interpretable as a date after 1900-01-01).

Examples

```
checkDate("02/20/2018")
checkDate("10/08/2018")
```

`checkSpecificModelCP` *Error check a specific model selection for a CP plot*

Description

Make sure it's available and good, update the name for usage

Usage

```
checkSpecificModelCP(modelSet, specificModel)
```

Arguments

`modelSet`	cp model set of class cpmSet
`specificModel`	the name of the specific model for the plot

Value

updated name of the model to use

`checkSpecificModelSE` *Error check a specific model selection for an SE plot*

Description

Make sure it's available and good, update the name for usage

Usage

```
checkSpecificModelSE(modelSet, specificModel)
```

Arguments

`modelSet`	pk model set of class pkmSet
`specificModel`	the name of the specific model for the plot

Value

updated name of the model to use

`combinePreds` *Combine predictors*

Description

Create a table of combinations of factor levels for the the predictors in a searcher efficiency or carcass persistence analysis. This is a utility function called by `pkm0` and `cpm0`.

Usage

```
combinePreds(preds, data)
```

Arguments

`preds`	Vector of character strings with the names of predictor variables included in the model.
`data`	Searcher efficiency or carcass persistence data frame with columns for each predictor and rows corresponding to carcasses in the field trials.

Value

Data frame with columns for each predictor in `preds` and rows for each factor level combination among the predictors. In addition there is a column with `CellNames`, which are the combinations of predictor levels separated by periods (.).

`combinePredsAcrossModels` *Combine predictors across models*

Description

Create a table of factor combinations of predictors in given searcher efficiency and carcass persistence models. This is a utility function called by estgGeneric and governs the cells for which detection probabilities are calculated.

Usage

```
combinePredsAcrossModels(preds_CP, preds_SE, data_CP, data_SE)
```

Arguments

`preds_CP`	Character vector with names of carcass persistence predictors.
`preds_SE`	Character vector with names of searcher efficiency predictors.
`data_CP`	data frame with columns for each predictor and rows corresponding to carcasses in the field trials.
`data_SE`	data frame with columns for each predictor and rows corresponding to carcasses in the field trials.

Value

Data frame with columns for each predictor in `preds` and rows for each factor level combination among the predictors. In addition there are column with `CellNames`, `CellNames_CP`, and `CellNames_SE`, which are the combinations of predictor levels for all predictors, CP predictors, and SE predictors (respectively), separated by periods (`.`).

`countCarcs` *Count the minimum number of carcasses in the cells*

Description

Count the minimum number of carcasses in all of the cells within a `_SetSize` model complex

Usage

```
countCarcs(mods)
```

Arguments

`mods`	model output from the `_SetSize` version of a function

Value

the minimum number of carcasses in the cells

`CO_DWP` *Associate CO carcasses with appropriate DWP values (by unit and carcass class)*

Description

Calculate the conditional probability of observing a carcass at search oi as a function arrival interval (assuming carcass is not removed by scavengers before the time of the final search)

Usage

```
CO_DWP(dwpsim, data_CO, unitCol, sizeCol = NULL)
```

Arguments

`dwpsim`	`rdwp` object
`data_CO`	data frame with results from carcass surveys
`unitCol`	name of the unit column in data_CO (required)
`sizeCol`	name of the carcass class column in data_CO (optional).

Value

numeric DWP array

`CPcols` *Produce a named vector of standard CP plot colors*

Description

Produce a named vector of standard CP plot colors

Usage

```
CPcols()
```

`CPdistOptions` *Produce the options for the distributions in the CP model*

Description

Simply make the named list for the disributions in the CP model

Usage

```
CPdistOptions()
```

Value

list with named elements of the distributions

`cpLogLik` *Calculate the negative log-likelihood of a carcass persistence model*

Description

The function used to calculate the negative-loglikelihood of a given carcass persistence model (`cpm`) with a given data set

Usage

```
cpLogLik(t1, t2, beta, nbeta_l, cellByCarc, cellMM, dataMM, dist)
```

Arguments

`t1`	last times observed present
`t2`	first times observed absent
`beta`	Parameters to be optimized.
`nbeta_l`	Number of parameters associated with l.
`cellByCarc`	Which cell each observation belongs to.
`cellMM`	Combined model matrix.
`dataMM`	Combined model matrix expanded to the data.
`dist`	Name of distribution.

Value

Negative log likelihood of the observations, given the parameters.

`cpm` *Fit cp carcass persistence models*

Description

Carcass persistence is modeled as survival function where the one or both parameter(s) can depend on any number of covariates. Format and usage parallel that of common `R` functions such as `lm`, `glm`, and `gam`. However, the input data (`data`) are structured differently to accommodate the survival model approach (see "Details"), and model formulas may be entered for both `l` ("location") and `s` ("scale").

Usage

```
cpm(formula_l, formula_s = NULL, data, left, right, dist = "weibull",
  allCombos = FALSE, sizeCol = NULL, CL = 0.9, quiet = FALSE)

cpm0(formula_l, formula_s = NULL, data = NULL, left = NULL,
  right = NULL, dist = "weibull", CL = 0.9, quiet = FALSE)

cpmSet(formula_l, formula_s = NULL, data, left, right,
  dist = c("exponential", "weibull", "lognormal", "loglogistic"),
  CL = 0.9, quiet = FALSE)

cpmSize(formula_l, formula_s = NULL, data, left, right,
  dist = c("exponential", "weibull", "lognormal", "loglogistic"),
  sizeCol = NULL, allCombos = FALSE, CL = 0.9, quiet = FALSE)
```

Arguments

`formula_l`	Formula for location; an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Details of model specification are given under "Details".
`formula_s`	Formula for scale; an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Details of model specification are given under "Details".
`data`	Data frame with results from carcass persistence trials and any covariates included in `formula_l` or formula_s (required).
`left`	Name of columns in `data` where the time of last present observation is stored.
`right`	Name of columns in `data` where the time of first absent observation is stored.
`dist`	Distribution name ("exponential", "weibull", "loglogistic", or "lognormal")
`allCombos`	logical. If `allCombos = FALSE`, then the single model expressed by `formula_l` and `formula_s` is fit using a call to `cpm0`. If `allCombos = TRUE`, a full set of `cpm` submodels derived from combinations of the given covariates for `p` and `k` is fit. For example, submodels of `formula_l = p ~ A * B` would be `p ~ A * B`, `p ~ A + B`, `p ~ A`, `p ~ B`, and `p ~ 1`. Models for each pairing of a `p` submodel with a `k` submodel are fit via `cpmSet`, which fits each model combination using successive calls to `cpm0`, which fits a single model.
`sizeCol`	character string. The name of the column in `data` that gives the size class of the carcasses in the field trials. If `sizeCol = NULL`, then models are not segregated by size. If a `sizeCol` is provided, then separate models are fit for the `data` subsetted by `sizeCol`.
`CL`	confidence level
`quiet`	Logical indicator of whether or not to print messsages

Details

The probability of a carcass persisting to a particular time is dictated by the specific distribution chosen and its underlying location (l) and scale (s) parameters (for all models except the exponential, which only has a location parameter). Both `l` and `s` may depend on covariates such as ground cover, season, species, etc., and a separate model format (`formula_l` and `formula_s`) may be entered for each. The models are entered as they would be in the familiar `lm` or `glm` functions in R. For example, `l` might vary with `A`, `B`, and `C`, while `k` varies only with `A`. A user might then enter `p ~ A + B + C` for `formula_l` and `k ~ A` for `formula_s`. Other R conventions for defining formulas may also be used, with `A:B` for the interaction between covariates A and B and `A * B` as short-hand for `A + B + A:B`.

Carcass persistence `data` must be entered in a data frame with data in each row giving the fate of a single carcass in the trials. There must be a column for each of the last time the carcass was observed present and the first time the carcass was observed absent (or NA if the carcass was always present). Additional columns with values for categorical covariates (e.g., visibility = E, M, or D) may also be included.

Value

an object of an object of class `cpm`, `cpmSet`, `cpmSize`, or `cpmSetSize`.

`cpm0()` returns a `cpm` object, which is a description of a single, fitted pk model. Due to the large number and complexity of components of a`cpm` model, only a subset of them is printed automatically; the rest can be viewed/accessed via the `$` operator if desired. These are described in detail in the '`cpm` Components' section.

`cpmSet()` returns a list of `cpm` objects, one for each of the submodels, as described with parameter `allCombos = TRUE`.

`cpmSize()` returns a list of `cpmSet` objects (one for each 'size') if `allCombos = T`, or a list of `cpm` objects (one for each 'size') if `allCombos = T`

`cpm` returns a `cpm`, `cpmSet`, `cpmSize`, or `cpmSetSize` object:

- `cpm` object if `allCombos = FALSE, sizeCol = NULL`
- `cpmSet` object if `allCombos = TRUE, sizeCol = NULL`
- `cpmSize` object if `allCombos = FALSE, sizeCol != NULL`
- `cpmSetSize` object if `allCombos = TRUE, sizeCol != NULL`

`cpm` Components

The following components of a `cpm` object are displayed automatically:

`call` the function call to fit the model

`formula_l` the model formula for the `p` parameter

`formula_s` the model formula for the `k` parameter

`distribution` distribution used

`predictors` list of covariates of `l` and/or `s`

`AICc` the AIC value as corrected for small sample size

`convergence` convergence status of the numerical optimization to find the maximum likelihood estimates of `p` and `k`. A value of `0` indicates that the model was fit successfully. For help in deciphering other values, see `optim`.

`cell_ls` summary statistics for estimated cellwise `l` and `s`, including the medians and upper & lower bounds on CIs for each parameter, indexed by cell (or combination of covariate levels).

`cell_ab` summary statistics for estimated cellwise `pda` and `pdb`, including the medians and upper & lower bounds on CIs for each parameter, indexed by cell (or combination of covariate levels).

`cell_desc` Descriptive statistics for estimated cellwise median persistence time and rI for search intervals of 1, 3, 7 14, and 28 days, where rI is the probability of that carcass that arrives at a uniform random time in within a search interval of I days persists until the first search after arrival.

The following components are not printed automatically but can be accessed via the `$` operator:

`data` the data used to fit the model

`betahat_l` parameter estimates for the terms in the regression model for for `l`

`betahat_s` parameter estimates for the terms in the regression model for for `s`. If dist = "exponential", `s` is set at 1 and not calculated.

`varbeta` the variance-covariance matrix of the estimators for `c(betahat_l, betahat_s`.

`cellMM_l` a cellwise model (design) matrix for covariate structure of `l_formula` `cellMM_s` a cellwise model(design) matrix for covariate structure of `s_formula`

`levels_l` all levels of each covariate of `l`

`levels_s` all levels of each covariate of `s`

`nbeta_l` number of parameters fit for `l`

`nbeta_s` number of parameters fit for `s`

`cells` cell structure of the cp-model, i.e., combinations of all levels for each covariate of `p` and `k`. For example, if `covar1` has levels `"a"`, `"b"`, and `"c"`, and `covar2` has levels `"X"` and `"Y"`, then the cells would consist of `a.X`, `a.Y`, `b.X`, `b.Y`, `c.X`, and `c.Y`.

`ncell` total number of cells

`predictors_l` list of covariates of `l`

`predictors_s` list of covariates of `s`

`observations` observations used to fit the model

`carcCells` the cell to which each carcass belongs

`AIC` the AIC value for the fitted model

`CL` the input `CL`

Advanced

`cpmSize` may also be used to fit a single model for each size class if `allCombos = FALSE`. To do so, `formula_l`, `formula_s`, and `dist` be named lists with names matching the sizes listed in `unique(data[,sizeCol])`. The return value is then a list of `cpm` objects, one for each size.

Examples

```
head(data(wind_RP))
mod1 <- cpm(formula_l = l ~ Season, formula_s = s ~ 1, data = wind_RP$CP,
  left = "LastPresent", right = "FirstAbsent")
class(mod1)
mod2 <- cpm(formula_l = l ~ Season, formula_s = s ~ 1, data = wind_RP$CP,
  left = "LastPresent", right = "FirstAbsent", allCombos = TRUE)
class(mod2)
names(mod2)
class(mod2[[1]])
mod3 <- cpm(formula_l = l ~ Season, formula_s = s ~ 1, data = wind_RP$CP,
  left = "LastPresent", right = "FirstAbsent",
  allCombos = TRUE, sizeCol = "Size")
class(mod3)
names(mod3)
class(mod3[[1]])
class(mod3[[1]][[1]])
```

`cpmCPCellPlot` *Plot cell-specific decay curve for carcass persistence*

Description

Produce the figure panel for a specific cell (factor level combination) including the specific fitted decay curves.

Usage

```
cpmCPCellPlot(model, specificCell, col, axis_y = TRUE, axis_x = TRUE)
```

Arguments

`model`	model of class cpm
`specificCell`	name of the specific cell to plot
`col`	color to use
`axis_y`	logical of whether or not to plot the y axis
`axis_x`	logical of whether or not to plot the x axis

`cpmFail` *Check if a CP model is well-fit*

Description

Run a check the arg is a well-fit cpm object

Usage

```
cpmFail(cpmod)
```

Arguments

`cpmod`	A `cpm` object to test

Value

logical value indicating a failed fit (TRUE) or successful (FALSE)

`cpmSetFail` *Check if cpm models fail*

Description

Run a check on each model within a `cpmSet` object to determine if it failed or not

Usage

```
cpmSetFail(cpmSetToCheck)
```

Arguments

`cpmSetToCheck`	A `cpmSet` object to test

Value

A vector of logical values indicating if each of the models failed

---

`cpmSetFailRemove` *Remove failed cpm models from a `cpmSet` object*

Description

Remove all failed models within a `cpmSet` object

Usage

```
cpmSetFailRemove(cpmSetToTidy)
```

Arguments

`cpmSetToTidy`	A `cpmSet` object to tidy

Value

A `cpmSet` object with failed models removed

`cpmSetSizeFail` *Check if all of the cpm models fail*

Description

Run a check on each model within a `cpmSetSize` object to determine if they all failed or not

Usage

```
cpmSetSizeFail(cpmSetSizeToCheck)
```

Arguments

`cpmSetSizeToCheck`	A `cpmSetSize` object to test

Value

A list of vectors of logical values indicating if each of the models failed

---

`cpmSetSizeFailRemove` *Remove failed cpm models from a* `cpmSetSize` *object*

Description

Remove failed models from a `cpmSetSize` object

Usage

```
cpmSetSizeFailRemove(cpmSetSizeToTidy)
```

Arguments

`cpmSetSizeToTidy`	A list of `cpmSetSize` objects to tidy

Value

A list of `cpmSet` objects with failed models removed

`cpmSetSpecCPCellPlot` *Plot cell-specific decay curve for carcass persistence*

Description

Produce the figure panel for a specific cell (factor level combination) including the specific fitted decay curves.

Usage

```
cpmSetSpecCPCellPlot(modelSet, specificModel, specificCell, cols, axes)
```

Arguments

`modelSet`	modelSet of class cpmSet
`specificModel`	name of the specific submodel to plot
`specificCell`	name of the specific cell to plot
`cols`	named vector of the colors to use for the distributions
`axes`	named vector of logical values indicating whether or not to plot the x axis and the y axis

Value

a specific cell plot panel

`dateCols` *Select the date columns from a data table*

Description

Simple function to facilitate selection of date columns from a data table

Usage

```
dateCols(data)
```

Arguments

`data`	data table potentially containing columns that could be coerced (via `checkDate()`) into a properly formatted date

Value

column names of columns that can be coerced to dates

`dateToDay` *Calculate day of study from calendar date*

Description

Convert calendar date to integer day from a reference date (`ref`).

Usage

```
dateToDay(date, ref = NULL)
```

Arguments

`date`	A date or vector of dates to convert to days.
`ref`	Reference date.

Value

Numeric value(s) of days from `ref`.

Examples

```
x <- c("2018-01-01", "2018-02-01")
dateToDay(x, x[1])
```

`defineUnitCol` *Auto-parsing to find the name of the unit column (*`unitCol`*) If a unit column is not explicitly defined by user in the arg list to* `estM` *or* `estg`*, then* `defineUnitCol` *parses the CO, DWP, and SS files to extract the unit column if possible. Criteria that a column must meet to be a unit column are that it is found in both* `data_CO` *and* `data_DWP`*, all units in* `data_CO` *must also be included among units in* `data_DWP`*, all units in both* `data_CO` *and* `data_DWP` *must be included among the column names in* `data_SS`*. If* `data_DWP = NULL`*, then the unit column must be included in* `data_CO` *and all its units must be included among the column names of* `data_SS`*.*

Description

Auto-parsing to find the name of the unit column (`unitCol`)

If a unit column is not explicitly defined by user in the arg list to `estM` or `estg`, then `defineUnitCol` parses the CO, DWP, and SS files to extract the unit column if possible.

Criteria that a column must meet to be a unit column are that it is found in both `data_CO` and `data_DWP`, all units in `data_CO` must also be included among units in `data_DWP`, all units in both `data_CO` and `data_DWP` must be included among the column names in `data_SS`. If `data_DWP = NULL`, then the unit column must be included in `data_CO` and all its units must be included among the column names of `data_SS`.

Usage

```
defineUnitCol(data_CO, data_SS = NULL, data_DWP = NULL)
```

Arguments

`data_CO`	carcass observation data (data frame)
`data_SS`	search schedule data (data frame)
`data_DWP`	density-weighted proportion data (data frame)

Value

name of unit column (`unitCol`), if a unique unit column can be identified. If no unit column is present or there is more than one unit column, `defineUnitCol` stops with an error.

`desc` *Descriptive statistics for a fitted CP model*

Description

Given a `cpm` object, calculate convenient descriptive statistics, including the median CP, specified `rI` statistics, and `pda` and `pdb` statistics for the fitted model (EoA parameterization), and location and scale parameters for the fitted model (`survival` package parameterization) along with estimated CIs.

Usage

```
desc(model_CP, Ir = c(1, 3, 7, 14, 28), CL = 0.9, nsim = 10000)
```

Arguments

`model_CP`	A fitted CP model (`cpm` object)
`Ir`	The intervals for which to calculate the r statistics
`CL`	The confidence level for the CIs.
`nsim`	Number of simulation draws for estimating CIs

Details

The CIs for the r statistics (and the medianCP for the Weibull) ara based on simulation of the `pda` and `pdb` parameters, calculation of the statistics, and taking the empirical distribution of the simulated values. Other CIs are based on the assumed bivariate normal distributions of the appropriately transformed `l` and `s` parameters in the fitted model using `beta_hat` and `varbeta`.

NOTE: `rI` is the probability that a carcass that arrives at a uniform random time in an interval of `I` days will persist until the first search after arrival.

Value

Matrix of point and interval estimates for the median CP and the r statistics for the specified intervals. The matrix is assigned to class `descCP` that is simply a matrix with dimensions `ncell` x `(1 + 3*(5 + length(Ir)))`, column names that give the number of observations in each cell, statistic name and upper and lower bounds (in triplets), and row names giving the names of the cells. `CL`, `nsim`, and the name of the fitted model (`model_CP`) are included as object attributes.

See Also

`cpm`, `rcp`, `ppersist`

`dlModTabSE` *Create the download version of the Searcher Efficiency model table*

Description

Format a user-friendly version of the parameter table from a Searcher Efficiency model, based on confidence level of interest

Usage

```
dlModTabSE(modTab, CL = 0.9)
```

Arguments

`modTab`	model table
`CL`	Confidence level

Value

download version of the SE model table

`DWPCols` *Select the DWP-ok columns from a data table*

Description

Simple function to facilitate selection of columns that could be DWP values from a data table

Usage

```
DWPCols(data)
```

Arguments

`data`	data table

Value

column names of columns that can be DWP values

---

dwpm *Fit density-weighted proportion (DWP) models.*

Description

Carcass density is modeled as a function of distance from turbine. Format and usage parallel that of common `R` functions `lm`, `glm`, and `gam` and the GenEst functions `pkm` and `cpm`.

Usage

```
dwpm(data_DWP, type = "data", unitCol = NULL, dwpCols = NULL)
```

Arguments

`data_DWP`	data frame with structure depending on model type. In general, `data_DWP` would be a data frame if a model is to be fit or if point estimates only are provided as pre-simulated DWP data, and, if pre-simulated data with variation are provided, then a 2-d array (if one carcass class) or a list of 2-d arrays (if more than one carcass class). See "Details" for details.
`type`	model type may be `rings`, `glm`, `TWL`, or `data`. Currently, only the `data` type is supported.
`unitCol`	name of the column with the units, which must be non-numeric
`dwpCols`	name(s) of the columns with the DWP data

Details

The fraction of carcasses falling in the area searched at a turbine may be a function of carcass class (e.g., large or small) and/or direction from the turbine. Data may be provided for fitting a distance model(s) or, alternatively, simulated turbine-wise DWP data from custom-fitted models may be provided. If pre-fit, pre-simulated data are used, then `glm` returns a `dwpm` object with `type = data`.

To fit a model, `data_DWP` should be a data frame with a row for each carcass and columns giving (at a minimum) unique carcass IDs, turbine ID, distance from turbine, and fraction of area searched at the given distance at the given turbine. Optional columns may include carcass class, covariates that may influence detection probability (e.g., visibility class), and direction. If covariates are to be included in the model, then the fraction of area column would give the fraction of the area in the given covariate level at that distance. Alternatively, prefab data may be provided in a dataframe, with structure depending on data type. The simplest case would be that point estimates only are provided. In that case, if there are no distinctions among carcass classes (e.g., size), then `data_DWP` should be a dataframe with one column giving the unit (e.g., turbine) and one column with the DWP at each unit; if distinctions are made among carcass classes, then `data_DWP` would be a data frame with a unit column and a DWP column for each carcass class. If the DWP estimates incorporate uncertainties, then `data_DWP` should be an array with `n_unit * nsim` rows and with colunms for units and DWPs for each carcass class.

Value

an object of an object of class `dwpm`, which is a list with model `type` (currently only `type = data` is supported) and `model`, which gives the simulated DWP values as an array (if there's only a single carcass class) or a list of arrays (if there are more than one carcass classes).

`estg` *Estimate all carcass-level detection rates and arrival intervals*

Description

Estimate g values and arrival intervals for a set of carcasses from fitted pk and cp models and search data

Usage

```
estg(data_CO, COdate, data_SS, SSdate = NULL, model_SE, model_CP,
  model_DWP = NULL, sizeCol = NULL, unitCol = NULL, IDcol = NULL,
  nsim = 1000, max_intervals = 8)
```

Arguments

`data_CO`	Carcass Observation data
`COdate`	Column name for the date found data
`data_SS`	Search Schedule data
`SSdate`	Column name for the date searched data. Optional. If not provided, `estg` will try to find the SSdate among the columns in data_SS. See `prepSS`.
`model_SE`	Searcher Efficiency model (or list of models if there are multiple carcass classes)
`model_CP`	Carcass Persistence model (or list of models if there are multiple carcass classes)
`model_DWP`	Density weighted proportion model (or list of models if there are multiple carcass classes)
`sizeCol`	Name of column in `data_CO` where the carcass classes are recorded. Optional. If not provided, no distinctions are made among sizes. `sizeCol` not only identifies what the name of the size segregating class
`unitCol`	Column name for the unit indicator
`IDcol`	Column name for unique carcass IDs (required)
`nsim`	the number of simulation draws
`max_intervals`	maximum number of arrival interval intervals to consider for each carcass. Optional. Limiting the number of search intervals can greatly increase the speed of calculations with only a slight reduction in accuracy in most cases.

Value

list of [1] g estimates (`ghat`) and [2] arrival interval estimates (`Aj`) for each of the carcasses. The row names of the `Aj` matrix are the units at which carcasses were found. Row names of `ghat` are the carcass IDs (in `data_CO`).

Examples

```
data(mock)
model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
             data = mock$SE)
model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
              data = mock$CP, dist = "weibull",
              left = "LastPresentDecimalDays",
              right = "FirstAbsentDecimalDays"
            )
ghat <- estg(data_CO = mock$CO, COdate = "DateFound",  data_SS = mock$SS,
     model_SE = model_SE, model_CP = model_CP, unitCol = "Unit", nsim = 100)
```

`estgGeneric` *Estimate generic g*

Description

Generic g estimation by simulation from given SE model and CP models under a specific search schedule.

The g estimated by `estgGeneric` is a generic aggregate detection probability and represents the probability of detecting a carcass that arrives at a (uniform) random time during the period monitored, for each of the possible cell combinations, given the SE and CP models. This is somethat different from the GenEst estimation of g when the purpose is to estimate total mortality (M), in which case the detection probability varies with carcass arrival interval and is difficult to summarize statistically. The `estgGeneric` estimate is a useful "big picture" summary of detection probability, but would be difficult to work with for estimating M with precision.

Usage

```
estgGeneric(days, model_SE, model_CP, nsim = 1000)
```

Arguments

`days`	Search schedule data as a vector of days searched
`model_SE`	Searcher Efficiency model (`pkm` object)
`model_CP`	Carcass Persistence model (`cpm` object)
`nsim`	the number of simulation draws

Value

`gGeneric` object that is a list of [1] a list of g estimates, with one element in the list corresponding to each of the cells from the cross-model combination and [2] a table of predictors and cell names associated with the gs

Examples

```
data(mock)
model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
              data = mock$SE)
model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
              data = mock$CP, left = "LastPresentDecimalDays",
              right = "FirstAbsentDecimalDays")
avgSS <- averageSS(mock$SS)
ghatsGeneric <- estgGeneric(days = avgSS, model_SE = model_SE,
 model_CP = model_CP)
```

`estgGenericSize` *Estimate generic detection probability for multiple carcass classes*

Description

Generic g estimation for a combination of SE model and CP model under a given search schedule

The g estimated by `estgGenericSize` is a generic aggregate detection probability and represents the probability of detecting a carcass that arrives at a (uniform) random time during the period monitored, for each of the possible cell combinations, given the SE and CP models. This is somethat different from the GenEst estimation of g when the purpose is to estimate total mortality (M), in which case the detection probability varies with carcass arrival interval and is difficult to summarize statistically. The `estgGeneric` estimate is a useful "big picture" summary of detection probability, but would be difficult to work with for estimating M with precision.

Usage

```
estgGenericSize(days, modelSetSize_SE, modelSetSize_CP,
  modelSizeSelections_SE, modelSizeSelections_CP, nsim = 1000)
```

Arguments

`days`	Search schedule data as a vector of days searched
`modelSetSize_SE`	Searcher Efficiency model set for multiple sizes
`modelSetSize_CP`	Carcass Persistence model set for multiple sizes
`modelSizeSelections_SE`	vector of SE models to use, one for each size. Size names are required, and names must match those of modelSetSize_SE. E.g., `c(lrg = "p ~ Visibility; k ~ 1",sml = "p ~ 1; k ~ 1")`. Model formulas are read as text and must have exact matches among models listed in modelSetSize_SE. For example, if one of the `modelSizeSelections_SE` elements is `lrg = "p ~ Visibility; k ~ 1"`, then `"p ~ Visibility; k ~ 1"` must be in `names(modelSizeSelections_SE)[["lrg"]]`.
`modelSizeSelections_CP`	vector of CP models to use, one for each size
`nsim`	the number of simulation draws

Value

list of g estimates, with one element in the list corresponding to each of the cells from the cross-model combination

Examples

```
data(mock)
pkmModsSize <- pkm(formula_p = p ~ HabitatType,
                 formula_k = k ~ HabitatType, data = mock$SE,
                 obsCol = c("Search1", "Search2", "Search3", "Search4"),
                 sizeCol = "Size", allCombos = TRUE)
cpmModsSize <- cpm(formula_l = l ~ Visibility,
                 formula_s = s ~ Visibility, data = mock$CP,
                 left = "LastPresentDecimalDays",
                 right = "FirstAbsentDecimalDays",
                 dist = c("exponential", "lognormal"),
                 sizeCol = "Size", allCombos = TRUE)

pkMods <- c("S" = "p ~ 1; k ~ 1", "L" = "p ~ 1; k ~ 1",
           "M" = "p ~ 1; k ~ 1", "XL" = "p ~ 1; k ~ 1"
          )
cpMods <- c("S" = "dist: exponential; l ~ 1; NULL",
            "L" = "dist: exponential; l ~ 1; NULL",
            "M" = "dist: exponential; l ~ 1; NULL",
            "XL" = "dist: exponential; l ~ 1; NULL"
          )
avgSS <- averageSS(mock$SS)
gsGeneric <- estgGenericSize(nsim = 1000, days = avgSS,
               modelSetSize_SE = pkmModsSize,
               modelSetSize_CP = cpmModsSize,
               modelSizeSelections_SE = pkMods,
               modelSizeSelections_CP = cpMods
             )
```

`estM` *Estimate mortality*

Description

Given given fitted Searcher Efficiency and Carcass Persistence models; Search Schedule, Density Weighted Proportion, and Carcass Observation data; and information about the fraction of the the facility that was surveyed.

Usage

```
estM(data_CO, data_SS, data_DWP = NULL, frac = 1,
  COdate = "DateFound", model_SE, model_CP, model_DWP = NULL,
  unitCol = NULL, SSdate = NULL, sizeCol = NULL, IDcol = NULL,
  DWPCol = NULL, nsim = 1000, max_intervals = 8)
```

Arguments

`data_CO`	Carcass Observation data
`data_SS`	Search Schedule data
`data_DWP`	Survey unit (rows) by carcass class (columns) density weighted proportion table
`frac`	fraction carcasses on ground that was surveyed but not accounted for in DWP
`COdate`	Column name for the date found data
`model_SE`	Searcher Efficiency model (or list of models if there are multiple carcass classes)
`model_CP`	Carcass Persistence model (or list of models if there are multiple carcass classes)
`model_DWP`	fitted dwp model (optional)
`unitCol`	Column name for the unit indicator (optional)
`SSdate`	Column name for the date searched data
`sizeCol`	Name of colum in `data_CO` where the carcass classes are recorded. Optional. If none provided, it is assumed there is no distinctions among carcass classes.
`IDcol`	column with unique carcass (CO) identifier
`DWPCol`	Column name for the DWP values in the DWP table when no carcass class is used and there is more than one column in `data_DWP` that could be interpreted as DWP.
`nsim`	the number of simulation draws
`max_intervals`	maximum number of arrival intervals to consider for each carcass

Value

list of Mhat, Aj, ghat, DWP (by carcass), and Xtot = total number of carcasses observe

Examples

```
 ## Not run:
 data(mock)
 model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
              data = mock$SE
             )
 model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
               data = mock$CP, dist = "weibull",
               left = "LastPresentDecimalDays",
               right = "FirstAbsentDecimalDays"
             )
 eM <- estM(nsim = 1000, data_CO = mock$CO, data_SS = mock$SS,
         data_DWP = mock$DWP, frac = 1, model_SE = model_SE,
         model_CP = model_CP, COdate = "DateFound",
         DWPCol = "S", sizeCol = NULL
       )

## End(Not run)
```

`expandModelSetCP` *Expand a CP model set for plotting*

Description

Expand the exponential models across the other distributions

Usage

```
expandModelSetCP(modelSet)
```

Arguments

`modelSet`	cp model set of class cpmSet

Value

updated model set

`GenEst` *Generalized estimation of mortality*

Description

This package is designed to analyze searcher efficiency, carcass persistence, search schedule, and carcass observation data for the estimation of bird and bat mortality at wind and solar power facilities.

Information

```
browseVignettes("GenEst")
packageDescription("GenEst")
disclaimerUSGS()
disclaimerWEST()
```

Data sets

`mock`
`wind_cleared`
`wind_RP`
`wind_RPbat`
`solar_powerTower`
`solar_PV`
`solar_trough`

Main command-line functions

`pkm`, `cpm`, `dwpm` estimate searcher efficiency (`pk`), carcass persistence (`cp`), and (`dwp`) parameters

`estM` estimate mortality given `pkm`, `cpm` and data

`calcSplits` split mortality estimates by subcategories

`plot` S3 function for `pkm`, `pkmSet`, `cpm`, `cpmSet`, `estM`, `splitFull`, `splitSummary`, `gGeneric`, and `gGenericSize` objects

`transposeSplits` transpose 2-d splits

`summary` S3 function for `estM`, `splitFull`, `gGeneric`, `gGenericSize` objects

`aicc` S3 function for extracting models' AICc values from `pkm`, `pkmSet`, `pkmSize`, `pkmSetSize`, `cpm`, `cpmSet`, `cpmSize`, and `cpmSetSize` objects

`desc` Calculate descriptive statistics for a fitted CP model

`estgGeneric` estimate detection probability (g) for given searcher efficiency and carcass persistence model

`runGenEst()` start the GUI

Potentially useful calculation functions

`rpk`, `qpk`, `rcp`, `rdwp`
`estg`, `calcg`
`ppersist`, `SEsi`
`alogit`, `logit`
`pkLogLik`, `cpLogLik`
`calcRate`, `calcTsplit`, `ltranspose`
`refMod`
`countCarcs`
`simpleMplot`

Potentially useful editing functions

`estgGenericSize`
`prepSS`
`averageSS`
`tidyModelSetCP`
`tidyModelSetSE`
`checkDate`
`dateCols`
`dateToDay`
`defineUnitCol`
`dlModTabSE`
`prettyModTabCP`
`prettyModTabSE`
`prettySplitTab`

Other functions

`trimSetSize`
`combinePreds`
`combinePredsAcrossModels`
`pkmSetSizeFailRemove`
`pkmSetFailRemove`
`cpmSetSizeFailRemove`
`cpmSetSizeFail`
`cpmSetFailRemove`
`CO_DWP`
`CPcols`
`cpmCPCellPlot`
`cpmFail`
`cpmSetFail`
`cpmSetSpecCPCellPlot`
`DWPCols`
`expandModelSetCP`
`obsCols_SE`
`pkmFail`
`pkmSetAllFail`
`pkmSetFail`
`pkmSetSizeFail`
`plotCPCells`
`plotCPFigure`
`plotCPHeader`
`predsCols`
`SEsi_left`
`SEsi_right`
`SEsi0`
`sizeCols`
`matchCells`
`checkComponents`
`checkSpecificModelCP`
`checkSpecificModelSE`
`combinePredsAcrossModels`
`CPdistOptions`
`obsCols_fta`
`obsCols_ltp`
`pkmParamPlot`
`pkmSECellPlot`
`pkmSet`
`pkmSetSpecParamPlot`
`pkmSetSpecSECellPlot`
`plotSEBoxPlots`
`plotSEBoxTemplate`
`plotSECells`
`plotSEFigure`
`plotSEHeader`

```
prepPredictors
readCSV
removeCols
```

Internal functions (not exported)

```
_GenEst_calcRateC
_GenEst_calcTsplitC
calcRateC
calcTsplitC
aboutContent
aboutPanel
analysisPanel
b
big
cButtonStyle
center
classText
clearNotifications
CPMainPanel
CPPanel
downloadCPFig
estText
CPSidebar
initialReactiveValues
createvtext
dataDownloadWidget
dataInputPanel
dataInputSidebar
dataInputWidget
dataTabPanel
disclaimersContent
disclaimersPanel
downloadCPFig
downloadCPMod
downloadCPMres
downloadSEmod
downloadgFig
downloadMFig
downloadSEFig
downloadsPanel
downloadTable
eventReaction
GeneralInputSidebar
GeneralInputsPanel
GenEstAcknowledgements
GenEstAuthors
GenEstGUIauthors
```

```
GenEstInlineCSS
GenEstLicense
GenEstLogos
GenEstShinyJS
GenEstUI
GenEstServer
gettingStartedContent
gettingStartedPanel
gMainPanel
gPanel
gSidebar
helpPanel
initialOutput
kFixedWidget
kFixedWidgetHeader
kFixedWidgetRow
li
loadedDataPanel
MMainPanel
modelInputWidget
modelOutputPanel
modelOutputWidget
modelRunWidget
modelSelectionWidget
modelSelectionWidgetHeader
modelSelectionWidgetRow
modelSetCells
modelSetModelCells
modelSetModelPredictors
modelSetPredictors
modNamePaste
modNameSplit
MPanel
msgFracNote
msgList
msgModDone
msgModFail
msgModPartialFail
msgModRun
msgModSENobs
msgModWarning
msgSampleSize
msgSplitFail
msgSSavgFail
msgSSinputFail
MSidebar
navbar
ol
```

```
pickSizeclass
plotNA
prepSizeclassText
preTextMaker
reaction
reactionMessageDone
reactionMessageRun
reNULL
reVal
SEboxes
SEcols
selectData
selectedDataPanel
SEMainPanel
SEPanel
SEpanel
SESidebar
setkNeed
setNotSuspending
small
splitButtonWidget
style
trimSetSize
u
ul
update_input
update_output
update_rv
updateColNames_size
updateSizeclasses
updatesizeCol
widgetMaker
```

`logit` *Compute the logit or anti-logit*

Description

Compute the logit or anti-logit

Usage

```
logit(x)

alogit(x)
```

Arguments

`x`	A number. For `logit`, a probability (between 0 and 1, inclusive). For `alogit`, any real number.

Value

`logit`: The logit of `x`.

`alogit`: The anti-logit of `x`.

Examples

```
logit(0.5)

alogit(0)
```

`ltranspose` *Transpose a list of arrays*

Description

A list of `n` arrays, each with dimension `m` x `k` is redimensioned to a list of `m` arrays, each with dimension `m` x `k`. NOTE: Attributes are not preserved.

Usage

```
ltranspose(M)
```

Arguments

`M`	a list of `n` `m` x `k` arrays

Value

a list of `m` `n` x `k` arrays

`mock` *A mock example data set*

Description

A template dataset used for testing purposes. Dataset containing SE, CP, SS, DWP, and CO data. Data are mostly random without patterns.

Usage

```
mock
```

Format

A list with 5 items:

SE Searcher efficiency trial data

CP Carcass persistence trial data

SS Search schedule data

DWP Density weighted proportion of area searched data

CO Carcass observations

Source

```
mock
```

`model_utility_functions` *model utility functions (not exported)*

Description

model utility functions that are not exported

Usage

```
matchCells(specific, modelSet)

modelSetModelPredictors(modelSet)

modelSetPredictors(modelSet)

modelSetModelCells(modelSet)

modelSetCells(modelSet)
```

Arguments

`specific`	specific model compared against the full set
`modelSet`	full model set to compare to the specific

`obsCols_fta` *Select the columns from a data table that could be CP First Time Absent observations*

Description

Simple function to facilitate selection of columns that could be First Time Absent observations for a CP model

Usage

```
obsCols_fta(data)
```

Arguments

`data`	data table

Value

column names of columns that can be observations

`obsCols_ltp` *Select the columns from a data table that could be CP Last Time Present observations*

Description

Simple function to facilitate selection of columns that could be Last Time Present observations for a CP model

Usage

```
obsCols_ltp(data)
```

Arguments

`data`	data table

Value

column names of columns that can be observations

`obsCols_SE` *Select the columns from a data table that could be SE observations*

Description

Simple function to facilitate selection of columns that could be observations for an SE model

Usage

```
obsCols_SE(data)
```

Arguments

`data`	data table

Value

column names of columns that can be observations

`pkLogLik` *Calculate the negative log-likelihood of a searcher efficiency model*

Description

The function used to calculate the negative-loglikelihood of a given searcher efficiency model (`pkm`) with a given data set

Usage

```
pkLogLik(misses, foundOn, beta, nbeta_p, cellByCarc, maxmisses, cellMM,
  kFixed = NULL)
```

Arguments

`misses`	Number of searches when carcass was present but not found.
`foundOn`	Search on which carcass was found.
`beta`	Parameters to be optimized.
`nbeta_p`	Number of parameters associated with p.
`cellByCarc`	Which cell each observation belongs to.
`maxmisses`	Maximum possible number of misses for a carcass.
`cellMM`	Combined pk model matrix.
`kFixed`	Value of k if fixed.

Value

Negative log likelihood of the observations, given the parameters.

---

pkm *Fit pk searcher efficiency models.*

Description

Searcher efficiency is modeled as a function of the number of times a carcass has been missed in previous searches and any number of covariates. Format and usage parallel that of common `R` functions `lm`, `glm`, and `gam`. However, the input data (`data`) is structured differently to accommodate the multiple-search searcher efficiency trials (see Details), and model formulas may be entered for both `p` (akin to an intercept) and `k` (akin to a slope).

Usage

```
pkm(formula_p, formula_k = NULL, data, obsCol = NULL, kFixed = NULL,
  allCombos = FALSE, sizeCol = NULL, CL = 0.9, kInit = 0.7,
  quiet = FALSE, ...)

pkm0(formula_p, formula_k = NULL, data, obsCol = NULL, kFixed = NULL,
  kInit = 0.7, CL = 0.9, quiet = FALSE)

pkmSet(formula_p, formula_k = NULL, data, obsCol = NULL,
  kFixed = NULL, kInit = 0.7, CL = 0.9, quiet = FALSE)

pkmSize(formula_p, formula_k = NULL, data, kFixed = NULL,
  obsCol = NULL, sizeCol = NULL, allCombos = FALSE, kInit = 0.7,
  CL = 0.9, quiet = FALSE)
```

Arguments

`formula_p`	Formula for p; an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. Details of model specification are given under "Details".
`formula_k`	Formula for k; an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. Details of model specification are given under "Details".
`data`	Data frame with results from searcher efficiency trials and any covariates included in `formula_p` or `formula_k` (required).
`obsCol`	Vector of names of columns in `data` where results for each search occasion are stored (optional). If `obsCol` is not provided, `pkm` uses as `obsCol` all columns with names that begin with an `"s"` or `"S"` and end with a number, e.g., "s1", "s2", "s3", etc. This option is included as a convenience for the user, but care must be taken that other data are not stored in columns with names matching that pattern. Alternatively, `obsCol` may be entered as a vector of names, like `c("s1","s2","s3")`, `paste0("s",1:3)`, or `c("initialSearch","anotherSearch","lastSearch")`. The columns must be in chronological order, that is, it is assumed that the first column is for the first search after carcass arrival, the second column is for the second search, etc.

`kFixed`	Parameter for user-specified `k` value (optional). If a value is provided, `formula_k` is ignored and the model is fit under the assumption that the `k` parameter is fixed and known to be `kFixed` $\in [0,1]$. If a `sizeCol` is provided, `kFixed` may either be `NULL`, a single number in [0, 1], or a vector with `kFixed` values for two or more of the carcass size classes. For example, if there are three sizes (`S`, `M`, and `L`), `kFixed` could be `c(S = 0.3,M = 0.8,L = 1.0)` to assign fixed `k` values to each size. To fit `k` for size `S` and to assign values of 0.8 and 1.0 to sizes `M` and `L`, resp., use `kFixed = c(S = 0.3,M = 0.8,L = 1.0)`. If there are more than one size classes and `kFixed` is a scalar, then all size classes are assigned the same `kFixed` value (unless `kFixed` is named, e.g., `kFixed = c(S = 0.5)`, in which case only the named size is assigned the `kFixed`).
`allCombos`	logical. If `allCombos = FALSE`, then the single model expressed by `formula_p` and `formula_k` is fit using a call to `pkm0`. If `allCombos = TRUE`, a full set of `pkm` submodels derived from combinations of the given covariates for `p` and `k` is fit. For example, submodels of `formula_p = p ~ A * B` would be `p ~ A * B`, `p ~ A + B`, `p ~ A`, `p ~ B`, and `p ~ 1`. Models for each pairing of a `p` submodel with a `k` submodel are fit via `pkmSet`, which fits each model combination using successive calls to `pkm0`, which fits a single model.
`sizeCol`	character string. The name of the column in `data` that gives the carcass class of the carcasses in the field trials. If `sizeCol = NULL`, then models are not segregated by size. If a `sizeCol` is provided, then separate models are fit for the `data` subsetted by `sizeCol`.
`CL`	numeric value in (0, 1). confidence level
`kInit`	numeric value in (0, 1). Initial value used for numerical optimization of `k`. Default is `kInit = 0.7`. It is rarely (if ever) necessary to use an alternative initial value.
`quiet`	Logical indicator of whether or not to print messsages
`...`	additional arguments passed to subfunctions

Details

The probability of finding a carcass that is present at the time of search is `p` on the first search after carcass arrival and is assumed to decrease by a factor of `k` each time the carcass is missed in searches. Both `p` and `k` may depend on covariates such as ground cover, season, species, etc., and a separate model format (`formula_p` and `formula_k`) may be entered for each. The models are entered as they would be in the familiar `lm` or `glm` functions in R. For example, `p` might vary with `A` and `B`, while `k` varies only with `A`. A user might then enter `p ~ A + B` for `formula_p` and `k ~ A` for `formula_k`. Other R conventions for defining formulas may also be used, with `A:B` for the interaction between covariates A and B and `A * B` as short-hand for `A + B + A:B`.

Search trial `data` must be entered in a data frame with data in each row giving the fate of a single carcass in the field trials. There must be a column for each search occassion, with 0, 1, or NA depending on whether the carcass was missed, found, or not available (typically because it was found and removed on a previous search, had been earlier removed by scavengers, or was not searched for) on the given search occasion. Additional columns with values for categorical covariates (e.g., visibility = E, M, or D) may also be included.

When all trial carcasses are either found on the first search or are missed on the first search after carcass placement, pkm effects a necessary adjustment to the for accuracy; otherwise, the model would not be able to determine the uncertainty and would substantially over-estimate the variance of the parameter estimates, giving $\hat{p}$ essentially equal to 0 or 1 with approximately equal probability. The adjustment is to fit the model on an adjusted data set with duplicated copies of the original data (`2n` observations) but with one carcass having the opposite fate of the others. For example, in field trials with very high searcher efficiency and `n = 10` carcasses, all of which are found in the first search after carcass placement, the original data set would have a carcass observation column consisting of 1s (`rep(1,10)`). The adjusted data set would have an observation column consisting of `2n -1` 1s and one 0. In this case, the point estimate of `p` is `1/(2n)` with distribution that closely resembling the Bayesian posterior distributions of `p` with a uniform or a Jeffreys prior. The adjustment is applied on a cellwise basis in full cell models (e.g., 1, A, B, A * B). In the additive model with two predictors (A + B), the adjustment is made only when a full level of covariate A or B is all 0s or 1s.

Value

an object of an object of class `pkm`, `pkmSet`, `pkmSize`, or `pkmSetSize`.

`pkm0()` returns a `pkm` object, which is a description of a single, fitted pk model. Due to the large number and complexity of components of a`pkm` model, only a subset of them is printed automatically; the rest can be viewed/accessed via the `$` operator if desired. These are described in detail in the '`pkm` Components' section.

`pkmSet()` returns a list of `pkm` objects, one for each of the submodels, as described with parameter `allCombos = TRUE`.

`pkmSize()` returns a list of `pkmSet` objects (one for each 'size') if `allCombos = T`, or a list of `pkm` objects (one for each 'size') if `allCombos = T`

`pkm` returns a `pkm`, `pkmSet`, `pkmSize`, or `pkmSetSize` object:

- `pkm` object if `allCombos = FALSE,sizeCol = NULL`
- `pkmSet` object if `allCombos = TRUE,sizeCol = NULL`
- `pkmSize` object if `allCombos = FALSE,sizeCol != NULL`
- `pkmSetSize` object if `allCombos = TRUE,sizeCol != NULL`

`pkm` Components

The following components of a `pkm` object are displayed automatically:

`call` the function call to fit the model

`formula_p` the model formula for the `p` parameter

`formula_k` the model formula for the `k` parameter

`predictors` list of covariates of `p` and/or `k`

`AICc` the AIC value as corrected for small sample size

`convergence` convergence status of the numerical optimization to find the maximum likelihood estimates of `p` and `k`. A value of `0` indicates that the model was fit successfully. For help in deciphering other values, see `optim`.

`cell_pk` summary statistics for estimated cellwise estimates of `p` and `k`, including the number of carcasses in each cell, medians and upper & lower bounds on CIs for each parameter, indexed by cell (or combination of covariate levels).

The following components are not printed automatically but can be accessed via the `$` operator:

`data` the data used to fit the model

`data0` `$data` with NA rows removed

`betahat_p, betahat_k` parameter estimates for the terms in the regression model for for `p` and `k` (logit scale). If `k` is fixed or not provided, then `betahat_k` is not calculated.

`varbeta` the variance-covariance matrix of the estimators for `c(betahat_p,betahat_k)`.

`cellMM_p, cellMM_k` cellwise model (design) matrices for covariate structures of `p_formula` and `k_formula`

`levels_p, levels_k` all levels of each covariate of `p` and `k`

`nbeta_p, nbeta_k` number of parameters to fit the `p` and `k` models

`cells` cell structure of the pk-model, i.e., combinations of all levels for each covariate of `p` and `k`. For example, if `covar1` has levels `"a"`, `"b"`, and `"c"`, and `covar2` has levels `"X"` and `"Y"`, then the cells would consist of `a.X`, `a.Y`, `b.X`, `b.Y`, `c.X`, and `c.Y`.

`ncell` total number of cells

`predictors_k, predictors_p` covariates of `p` and `k`

`observations` observations used to fit the model

`kFixed` the input `kFixed`

`AIC` the AIC value for the fitted model

`carcCells` the cell to which each carcass belongs

`CL` the input `CL`

`loglik` the log-liklihood for the maximum likelihood estimate

`pOnly` a logical value telling whether `k` is included in the model. `pOnly = TRUE` if and only if `length(obsCol) == 1)` and `kFixed = NULL`.

`data_adj` `data0` as adjusted for the 2n fix to accommodate scenarios in which all trial carcasses are either found or all are not found on the first search occasion (uncommon)

`fixBadCells` vector giving the names of cells adjusted for the 2n fix

Advanced

`pkmSize` may also be used to fit a single model for each carcass class if `allCombos = FALSE`. To do so, `formula_p` and `formula_k` must be a named list of formulas with names matching the sizes listed in `unique(data[,sizeCol])`. The return value is then a list of `pkm` objects, one for each size.

See Also

`rpk`, `qpk`, `aicc`, `plot.pkm`

Examples

```
head(data(wind_RP))
mod1 <- pkm(formula_p = p ~ Season, formula_k = k ~ 1, data = wind_RP$SE)
class(mod1)
mod2 <- pkm(formula_p = p ~ Season, formula_k = k ~ 1, data = wind_RP$SE,
```

```
    allCombos = TRUE)
  class(mod2)
  names(mod2)
  class(mod2[[1]])
  mod3 <- pkm(formula_p = p ~ Season, formula_k = k ~ 1, data = wind_RP$SE,
    allCombos = TRUE, sizeCol = "Size")
  class(mod3)
  names(mod3)
  class(mod3[[1]])
  class(mod3[[1]][[1]])
```

`pkmFail` *Check if a pk model is well-fit*

Description

Run a check the arg is a well-fit pkm object

Usage

```
pkmFail(pkmod)
```

Arguments

`pkmod`	A `pkm` object to test

Value

logical value indicating a failed fit (TRUE) or successful (FALSE)

`pkmParamPlot` *Plot parameter box plots for each cell for either p or k*

Description

Boxplot for pk model cells (soon to be deprecated)

Usage

```
pkmParamPlot(model, pk = "p", col)
```

Arguments

`model`	model of class pkm
`pk`	character of "p" or "k" to delineate between parameter graphed
`col`	color to use

Value

a parameter plot panel

`pkmSECellPlot` *Plot cell-specific decay curve for searcher efficiency*

Description

Plot cell-specific decay curve for searcher efficiency

Usage

```
pkmSECellPlot(model, specificCell, col, axis_y = TRUE, axis_x = TRUE)
```

Arguments

`model`	model of class pkm
`specificCell`	name of the specific cell to plot (soon to be deprecated)
`col`	color to use
`axis_y`	logical of whether or not to plot the y axis
`axis_x`	logical of whether or not to plot the x axis

Value

a cell plot panel

`pkmSetAllFail` *Check if all of the pkm models fail within a given set*

Description

Run a check on each model within a `pkmSet` object to determine if they all failed or not

Usage

```
pkmSetAllFail(pkmSetToCheck)
```

Arguments

`pkmSetToCheck`	A `pkmSet` object to test

Value

A logical value indicating if all models failed in the set

`pkmSetFail` *Check if pkm models fail*

Description

Run a check on each model within a `pkmSet` object to determine if it failed or not

Usage

```
pkmSetFail(pkmSetToCheck)
```

Arguments

`pkmSetToCheck`	A `pkmSet` object to test

Value

A vector of logical values indicating if each of the models failed

---

`pkmSetFailRemove` *Remove failed pkm models from a* `pkmSet` *object*

Description

Remove all failed models within a `pkmSet` object

Usage

```
pkmSetFailRemove(pkmSetToTidy)
```

Arguments

`pkmSetToTidy`	A `pkmSet` object to tidy

Value

A `pkmSet` object with failed models removed

`pkmSetSizeFail` *Check if all of the pkm models fail*

Description

Run a check on each model within a `pkmSetSize` object to determine if they all failed or not

Usage

```
pkmSetSizeFail(pkmSetSizeToCheck)
```

Arguments

`pkmSetSizeToCheck`	A `pkmSetSize` object to test

Value

A list of logical vectors indicating which models failed

`pkmSetSizeFailRemove` *Remove failed pkm models from a* `pkmSetSize` *object*

Description

Remove failed models from a `pkmSetSize` object

Usage

```
pkmSetSizeFailRemove(pkmSetSizeToTidy)
```

Arguments

`pkmSetSizeToTidy`	A list of `pkmSetSize` objects to tidy

Value

A list of `pkmSet` objects with failed models removed

`pkmSetSpecParamPlot` *p or k box plots for an SE model set*

Description

Plot parameter box plots for each cell within a model for either p or k with comparison to the cellwise model (soor to be deprecated)

Usage

```
pkmSetSpecParamPlot(modelSet, specificModel, pk = "p", cols)
```

Arguments

`modelSet`	modelSet of class pkmSet
`specificModel`	name of the specific submodel to plot
`pk`	character of "p" or "k" to delineate between parameter graphed
`cols`	named vector of colors to use for the specific and reference models

Value

a specific parameter plot panel

`pkmSetSpecSECellPlot` *Plot cell-specific decay curve for searcher efficiency for a specific model with comparison to the cellwise model*

Description

Plot cell-specific decay curve for searcher efficiency for a specific model with comparison to the cellwise model

Usage

```
pkmSetSpecSECellPlot(modelSet, specificModel, specificCell, cols, axes)
```

Arguments

`modelSet`	modelSet of class pkmSet (soon to be deprecated)
`specificModel`	name of the specific submodel to plot
`specificCell`	name of the specific cell to plot
`cols`	named vector of colors to use for the specific and reference models
`axes`	named vector of logical values indicating whether or not to plot the x axis and the y axis

Value

a specific cell plot panel

`pllogis` *The CDF of the loglogistic distribution*

Description

The CDF of the loglogistic distribution

Usage

```
pllogis(q, pda, pdb)
```

Arguments

`q`	a numeric vector of quantiles
`pda`	the α parameter
`pdb`	the β parameter

Details

There are several common parameterizations of the loglogistic distribution. The one used here gives the following:

CDF `Pr(X <= x) = 1/(1 + (x/`β`)^-`α`)`

PDF `Pr(X = x) = (`α/β`) * (x/`β`)^(`α` -1)/(1 + (x/`β`)^`α`)^2`

Value

`Pr(X <= q | pda,pdb)`

`plot.cpm` *Plot results of a single CP model*

Description

Plot a single `cpm` model

Usage

```
## S3 method for class 'cpm'
plot(x, col = "black", ...)
```

Arguments

`x`	model of class cpm
`col`	color to use
`...`	to be passed down

Examples

```
  data(wind_RP)
  mod <- cpm(formula_l = l ~ Season, formula_s = s ~ Season,
           data = wind_RP$CP, left = "LastPresent", right = "FirstAbsent")
 plot(mod)
```

`plot.cpmSet` *Plot results of a set of CP models*

Description

Produce a set of figures for a set of CP models, as fit by `cpmSet`

Usage

```
## S3 method for class 'cpmSet'
plot(x, specificModel = NULL, cols = NULL, ...)
```

Arguments

`x`	pk model set of class pkmSet
`specificModel`	the name(s) or index number(s) of specific model(s) to restrict the plot
`cols`	named vector of the colors to use for the distributions
`...`	to be passed down

Examples

```
  data(wind_RP)
  mod <- cpmSet(formula_l = l ~ Season, formula_s = s ~ Season,
           data = wind_RP$CP, left = "LastPresent", right = "FirstAbsent")
 plot(mod)
```

`plot.estM` *Plot total mortality estimation*

Description

`plot` defined for class `estM` objects

Usage

```
## S3 method for class 'estM'
plot(x, ..., CL = 0.9)
```

Arguments

`x`	`estM` object
`...`	arguments to pass down
`CL`	confidence level

Examples

```
 ## Not run:
 data(mock)
 model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
              data = mock$SE)
 model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
               data = mock$CP, dist = "weibull",
               left = "LastPresentDecimalDays",
               right = "FirstAbsentDecimalDays")
 eM <- estM(nsim = 1000, data_CO = mock$CO, data_SS = mock$SS,
         data_DWP = mock$DWP, frac = 1, model_SE = model_SE,
         model_CP = model_CP, COdate = "DateFound",
         DWPCol = "S", sizeCol = NULL)
 plot(eM)

## End(Not run)
```

`plot.gGeneric` *Plot results of a single generic ghat estimation*

Description

Plot method for a single generic `ghat` estimation

Usage

```
## S3 method for class 'gGeneric'
plot(x, CL = 0.9, ...)
```

Arguments

`x`	`estgGeneric` output
`CL`	confidence level to use
`...`	to be passed down

Value

generic detection probability plot

Examples

```
data(mock)
model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
              data = mock$SE)
model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
              data = mock$CP, left = "LastPresentDecimalDays",
              right = "FirstAbsentDecimalDays")
avgSS <- averageSS(mock$SS)
ghatsGeneric <- estgGeneric(nsim = 1000, avgSS, model_SE, model_CP)
plot(ghatsGeneric)
```

`plot.gGenericSize` *Plot results of a set of size-based generic ghat estimations*

Description

Plot method for a size-based generic `ghat` estimation

Usage

```
## S3 method for class 'gGenericSize'
plot(x, CL = 0.9, ...)
```

Arguments

`x`	`estgGenericSize` output
`CL`	confidence level to use
`...`	to be passed down

Value

size-based detection probability plot

Examples

```
  data(mock)
  pkmModsSize <- pkm(formula_p = p ~ HabitatType,
                  formula_k = k ~ HabitatType, data = mock$SE,
                  obsCol = c("Search1", "Search2", "Search3", "Search4"),
                  sizeCol = "Size", allCombos = TRUE)
  cpmModsSize <- cpm(formula_l = l ~ Visibility,
                  formula_s = s ~ Visibility, data = mock$CP,
                  left = "LastPresentDecimalDays",
                  right = "FirstAbsentDecimalDays",
                  dist = c("exponential", "lognormal"),
                  sizeCol = "Size", allCombos = TRUE)
  pkMods <- c("S" = "p ~ 1; k ~ 1", "L" = "p ~ 1; k ~ 1",
             "M" = "p ~ 1; k ~ 1", "XL" = "p ~ 1; k ~ 1"
            )
  cpMods <- c("S" = "dist: exponential; l ~ 1; NULL",
              "L" = "dist: exponential; l ~ 1; NULL",
              "M" = "dist: exponential; l ~ 1; NULL",
              "XL" = "dist: exponential; l ~ 1; NULL"
            )
  avgSS <- averageSS(mock$SS)
  gsGeneric <- estgGenericSize(nsim = 1000, days = avgSS,
                modelSetSize_SE = pkmModsSize,
                modelSetSize_CP = cpmModsSize,
                modelSizeSelections_SE = pkMods,
                modelSizeSelections_CP = cpMods
              )
 plot(gsGeneric)
```

`plot.pkm` *Plot results of a single pk model*

Description

Plot a single `pkm` model

Usage

```
## S3 method for class 'pkm'
plot(x, col = NULL, CL = NULL, ...)
```

Arguments

`x`	model of class pkm
`col`	color to use
`CL`	confidence level to show in boxplots and confidence bounds
`...`	arguments to be passed to sub functions

Value

a plot

Examples

```
data(wind_RP)
mod <- pkm(formula_p = p ~ Season, formula_k = k ~ 1, data = wind_RP$SE)
plot(mod)
```

`plot.pkmSet` *Plot results of a set of SE models*

Description

Produce a set of figures for a set of SE models, as fit by `pkmSet`

Usage

```
## S3 method for class 'pkmSet'
plot(x, specificModel = NULL, cols = NULL,
  CL = NULL, ...)
```

Arguments

`x`	pk model set of class pkmSet
`specificModel`	the name(s) or index number(s) of specific model(s) to restrict the plot
`cols`	named vector of colors to use for the specific and reference models
`CL`	confidence level
`...`	to be sent to subfunctions

Value

a set of plots

Examples

```
data(wind_RP)
mod <- pkmSet(formula_p = p ~ Season, formula_k = k ~ Season,
         data = wind_RP$SE
       )
plot(mod)
```

`plot.splitFull` *Plot summary statistics for splits of mortality estimates*

Description

The S3 `plot` method for `splitFull` objects constructs boxplots of the mortality estimates for all combinations of splitting covariates summarized in the `splits` variable. This is a simple wrapper function for creating a `splitSummary` object by calling `summary.splitFull` and plotting the result via `plot.splitSummary`.

Usage

```
## S3 method for class 'splitFull'
plot(x, rate = FALSE, CL = 0.9,
  commonScale = FALSE, ...)
```

Arguments

`x`	A `splitSummary` object (result of `calcSplits`) that includes summary statistics for simulated mortality estimates for all combinations of levels of 1 or 2 splitting covariates.
`rate`	`logical` scalar indicating whether the figures should be plotted as number of fatalities per split category (`rate = TRUE`) or fatality rates per unit time (`rate = TRUE`). If the splits do not include either a `split_SS` or `split_time` variable, the `rate` arg is ignored.
`CL`	desired confidence level to show in box plots
`commonScale`	Boolean: Should panels share a common y-axis scale? Relevant only when there are two splitting variables.
`...`	to be passed down

`plot.splitSummary` *Plot summary statistics for splits of mortality estimates*

Description

The S3 `plot` method for `splitSummary` objects constructs boxplots of the mortality estimates for all combinations of splitting covariates summarized in the `splits` variable.

For 1-covariate splits, box plots showing median, IQR, and confidence intervals (for the `CL` attribute for the splits object). For 2-covariate splits, the box plots are in an array with levels of the temporal split (`split_SS` or `split_time`) arranged horizontally (if present) and the levels of the `split_CO` variable arranged vertically. If no temporal splits are present, then the box plots along the levels of the first `split_CO` variable are arranged horizontally and the levels of the second variable are are arranged vertically.

Usage

```
## S3 method for class 'splitSummary'
plot(x, rate = FALSE, commonScale = FALSE, ...)
```

Arguments

`x`	A `splitSummary` object (result of `calcSplits`) that includes summary statistics for simulated mortality estimates for all combinations of levels of 1 or 2 splitting covariates.
`rate`	`logical` scalar indicating whether the figures should be plotted as number of fatalities per split category (`rate = TRUE`) or fatality rates per unit time (`rate = TRUE`). If the splits do not include either a `split_SS` or `split_time` variable, the `rate` arg is ignored.
`commonScale`	boolean to indicate whether to plot separate splits panels with a common scale on their y-axes (or have y-axes scaled to fit each graph separately)
`...`	additional arguments to be passed down

`plotCPCells` *Plot the cellwise results of a single model in a set of CP models*

Description

Produce a set of cellwise figures for a specific CP model, as fit by `cpmSet`

Usage

```
plotCPCells(modelSet, specificModel, cols)
```

Arguments

`modelSet`	cp model set of class cpmSet
`specificModel`	the name of the specific model for the plot
`cols`	named vector of the colors to use for the distributions

Value

a plot

`plotCPFigure` *Plot results of a single CP model in a set*

Description

Produce a figures for a specific CP model, as fit by `cpmSet`

Usage

```
plotCPFigure(modelSet, specificModel, cols = CPcols())
```

Arguments

`modelSet`	cp model set of class `cpmSet`
`specificModel`	the name of the specific model for the plot
`cols`	named vector of the colors to use for the distributions

Value

a plot

`plotCPHeader` *The CP plot header*

Description

Produce the header for a CP plot

Usage

```
plotCPHeader(modelSet, specificModel, cols = CPcols())
```

Arguments

`modelSet`	cp model set of class cpmSet
`specificModel`	the name of the specific model for the plot
`cols`	named vector of the colors to use for the distributions

Value

a plot

`plotSEBoxPlots` *p and k box plots for an SE model set*

Description

Plot parameter box plots for each cell within a model for both p and k with comparison to the cellwise model (soon to be deprecated)

Usage

```
plotSEBoxPlots(modelSet, specificModel, cols)
```

Arguments

`modelSet`	modelSet of class pkmSet
`specificModel`	name of the specific submodel to plot
`cols`	named vector of colors to use for the specific and reference models

Value

a set of parameter plot panels

`plotSEBoxTemplate` *template box plot*

Description

Plot template box plot (soon to be deprecated)

Usage

```
plotSEBoxTemplate(modelSet, specificModel, cols)
```

Arguments

`modelSet`	modelSet of class pkmSet
`specificModel`	name of the specific submodel to plot
`cols`	named vector of colors to use for the specific and reference models

Value

a template box plot

`plotSECells` *Plot the cellwise results of a single model in a set of SE models*

Description

Produce a set of cellwise figures for a specific SE model, as fit by `pkmSet` (soon to be deprecated)

Usage

```
plotSECells(modelSet, specificModel, cols)
```

Arguments

`modelSet`	pk model set of class pkmSet
`specificModel`	the name of the specific model for the plot
`cols`	named vector of colors to use for the specific and reference models

Value

a plot

`plotSEFigure` *Plot results of a single SE model in a set*

Description

Produce a figures for a specific SE model, as fit by `pkmSet` (soon to be deprecated)

Usage

```
plotSEFigure(modelSet, specificModel, app, cols)
```

Arguments

`modelSet`	pk model set of class `pkmSet`
`specificModel`	the name of the specific model for the plot
`app`	logical indicating if the plot is for the app
`cols`	named vector of colors to use for the specific and reference models

Value

a plot

`plotSEHeader` *The SE plot header*

Description

Produce the header for an SE plot (soon to be deprecated)

Usage

```
plotSEHeader(modelSet, specificModel, app = FALSE, cols = SEcols())
```

Arguments

`modelSet`	pk model set of class pkmSet
`specificModel`	the name of the specific model for the plot
`app`	logical indicating if the plot is for the app
`cols`	named vector of colors to use for the specific and reference models

Value

a plot

`ppersist` *Calculate the probability of persistence to detection*

Description

Given a set of CP parameters (of `"ppersist"` type), calculate the probability of persistence to detection for a carcass.

Usage

```
ppersist(pda, pdb, dist, t_arrive0, t_arrive1, t_search)
```

Arguments

`pda`	parameter a.
`pdb`	parameter b.
`dist`	Distribution used.
`t_arrive0`	Beginning of arrival window.
`t_arrive1`	End of arrival window.
`t_search`	Search time.

Value

Probability of persistence of detection to at t_search, given arrival between t_arrive0 and t_arrive1

`predsCols` *Select the predictor-ok columns from a data table*

Description

Simple function to facilitate selection of columns that could be predictors for SE or CP models from a data table

Usage

```
predsCols(data)
```

Arguments

`data`	data table

Value

column names of columns that can be predictors

`prepPredictors` *Prepare predictors based on inputs*

Description

Prepare predictor inputs from the app for use in the model function

Usage

```
prepPredictors(preds = NULL)
```

Arguments

`preds`	predictors, as input to the app

Value

prepared predictors (or 1 if no predictors)

`prepSS` *Create search schedule data into an prepSS object for convenient splits analyses*

Description

Since data_SS columns largely have a specific, required format, the `prepSS` function can often automatically decipher the data, but the user may specify explicit instructions for parsing the data for safety if desired. If the data are formatted properly, the automatic parsing is reliable in most cases. There are two exceptions. (1) If there is more than one column with possible dates (formatted as formal dates (as class `Date`, `POSIXlt` or `POSIXct`) or character strings or factors that can be unambiguously interpreted as dates (with assumed format "2018-05-15" or "2018/5/15"). In that case, the user must specify the desired dates as `dateColumn`. (2) If there is a covariate column consisting entirely of 0s and 1s. In that case, the user must specify the column(s) in `covars`.

Usage

```
prepSS(data_SS, SSdate = NULL, preds = NULL)
```

Arguments

`data_SS`	data frame or matrix with search schedule parameters, including columns for search dates, covariates (describing characteristics of the search intervals), and each unit (with 1s and 0s to indicate whether the given unit was searched (= 1) or not (= 0) on the given date)
`SSdate`	name of the column with the search dates in it (optional). If no `SSdate` is given, `prepSS` will try to find the date column based on data formats. If there is exactly one column that can be interpreted as dates, that column will be taken as the dates searched. If more than one date column is found, `prepSS` exits with an error message.
`preds`	vector of character strings giving the names of columns to be interpreted as potential covariates (optional). Typically, it is not necessary for a user to provide a value for `preds`. It is used only to identify specific columns of 1s and 0s as covariates rather than as search schedules.

Value

`prepSS` object that can be conveniently used in the splitting functions.

Examples

```
data(mock)
prepSS(mock$SS)
```

`prettyModTabCP` *Create the pretty version of the Carcass Persistence model table*

Description

Format a reader-friendly version of the parameter table from a carcass persistence model showing CIs for medianCP and for rI's for intervals of Ir

Usage

```
prettyModTabCP(modTab)
```

Arguments

`modTab`	`descCP` object or NULL

Value

pretty version of the CP model table in a data frame with point and interval estimates for medianCP and rI statistics. Output table is ready for rendering in shiny and posting in the GUI

`prettyModTabSE` *Create the pretty versions of model and summary tables*

Description

Format reader-friendly versions of results summary tables for searcher efficiency (GUI display and download), carcass persistence, and splits.

Usage

```
prettyModTabSE(modTab, CL = 0.9)
```

Arguments

`modTab`	model table
`CL`	Confidence level

Value

pretty version of the SE model table

`prettySplitTab` *Create the pretty version of the split summary table*

Description

Format a reader-friendly version of the split summary table a mortality estimation

Usage

```
prettySplitTab(splitSummary)
```

Arguments

`splitSummary`	a split summary

Value

split pretty table

`print.corpus_frame` *Generic S3 function for printing corpus_frame*

Description

Generic S3 function for printing corpus_frame

Usage

```
## S3 method for class 'corpus_frame'
print(x, ...)
```

Arguments

`x`	data frame to print
`...`	other arguments

Value

prints data frame

`print.cpm` *Print a `cpm` model object*

Description

Print a `cpm` model object

Usage

```
## S3 method for class 'cpm'
print(x, ...)
```

Arguments

`x`	a `cpm` model object
`...`	to be passed down

`print.pkm` *Print a `pkm` model object*

Description

Print a `pkm` model object

Usage

```
## S3 method for class 'pkm'
print(x, ...)
```

Arguments

`x`	a `pkm` model object
`...`	to be passed down

---

qpk *Quantiles of marginal distributions of $\hat{p}$ and $\hat{k}$*

Description

Calculate quantiles of marginal distributions of $\hat{p}$ and $\hat{k}$ for a `pkm` model object

Usage

```
qpk(p, model)
```

Arguments

`p`	vector of probabilities
`model`	A `pkm` object (which is returned from `pkm()`)

Value

either a list of `ncell` × `length(p)` matrices of quantiles for `$p` and `$k` for cells defined by the `model` object (if `model$pOnly == FALSE`) or a `ncell` × `length(p)` matrix of quantiles for `p`

See Also

`rpk`, `pkm`

Examples

```
 # 90% confidence intervals for \code{p} and \code{k}
  mod <- pkm(formula_p = p ~ Visibility * Season, formula_k = k ~ Season,
   data = wind_cleared$SE)
  qpk(p = c(0.05, 0.95), model = mod)
```

---

rcp *Simulate parameters from a fitted cp model*

Description

Simulate parameters from a `cpm` model object, and format them as either type `"survreg"` or `"ppersist"`

Usage

```
rcp(n, model, type = "survreg")
```

Arguments

`n`	the number of simulation draws
`model`	A `cpm` object (which is returned from `cpm`)
`type`	The type of parameters requested. `"survreg"` or `"ppersist"`

Value

list of two matrices of `n` simulated `l` and `s` (if `type = "survreg"`) or `a` and `b` (if `type = "ppersist"`)for cells defined by the `model` object.

Examples

```
data(wind_RP)
mod <- cpm(formula_l = l ~ 1, data = wind_RP$CP, left = "LastPresent",
         right = "FirstAbsent"
       )
rcp(n = 10, model = mod, type = "survreg")
rcp(n = 10, model = mod, type = "ppersist")
```

`rdwp` *Simulate parameters from a fitted dwp model*

Description

Simulate parameters from a `dwpm` model object

Usage

```
rdwp(n, model)
```

Arguments

`n`	the number of simulation draws
`model`	A `dwpm` object (which is returned from `dwpm()`)

Details

If the model type = `data`, then the number of simulated columns must be either `>=n` (in which case the first n colunms are taken as the simulated DWP) or 1 (in which case, DWP is assumed constant).

Value

array of `n` simulated `dwp` values for each unit. Dimensions = c(n, number of units).

`readCSV` *Read in csv files in either format*

Description

Handle reading in of a csv that is either comma-decimal or semicolon-comma separation style

Usage

```
readCSV(path)
```

Arguments

`path`	file path

Value

read in data table

`refMod` *Return the model with the greatest log-likelihood*

Description

Compares all fitted models in a list and returns the model with the greatest log-likelihood

Usage

```
refMod(modelSet)
```

Arguments

`modelSet`	a list of fitted models with a `loglik` element. Models may be `pkm`, `cpm`, `survreg` objects or any objects with a `loglik` component.

Value

The model object with the greatest log-likelihood among the models in `modelSet`

`removeCols` *Remove selected columns from column names*

Description

Simple function to facilitate removal of columns selected

Usage

```
removeCols(colNames, selCols)
```

Arguments

`colNames`	column names from which some could be removed
`selCols`	selected columns to be removed

Value

column names without selected columns

`rpk` *Simulate parameters from a fitted pk model*

Description

Simulate parameters from a `pkm` model object

Usage

```
rpk(n, model)
```

Arguments

`n`	the number of simulation draws
`model`	A `pkm` object (which is returned from `pkm()`)

Value

list of pairs of matrices of `n` simulated `p` and `k` for cells defined by the `model` object.

See Also

`rpk`, `pkm`

Examples

```
data(wind_RP)
mod <- pkm(formula_p = p ~ 1, formula_k = k ~ Season, data = wind_RP$SE)
rpk(n = 10, model = mod)
```

`runGenEst` *Launch the GenEst Application*

Description

Launches a local version of the GenEst application by running `runApp` pointed to the app subdirectory in the local `GenEst` package folder.

Usage

```
runGenEst()
```

`SEcols` *Produce a named vectory with standard SE plot colors*

Description

Produce a named vectory with standard SE plot colors. soon to be deprecated.

Usage

```
SEcols()
```

`SEpanel` *Produce a single panel in an SE summary/diagnostic plot*

Description

Each call to `SEpanel` produces a single panel showing searcher efficiency as a function of number of searches. Includes raw data (`found` and `available`) and model fits for a specific model (`y_spc`) and for the reference model (`y_ref`) for the `pkmSet` object from the reference model was extracted. For interal use only, for producing figs for `plot.pkmSet`.

Usage

```
SEpanel(found, available, y_spc, y_ref, xends, cols_SE)
```

Arguments

`found`	vector of number carcasses found on the ith attempt
`available`	vector of number carcasses found on the ith attempt
`y_spc`	vector of model fits for the specific model
`y_ref`	vector of model fits for the reference model
`xends`	x-axis buffer (numeric scalar) on sides of figs
`cols_SE`	named vector of colors (character)

Value

NULL inserts a panel with no labels into a preformatted figure

---

`SEsi` *Calculate decayed searcher efficiency*

Description

Calculate searcher efficiency after some searches under pk values

Usage

```
SEsi(days, pk)
```

Arguments

`days`	search days
`pk`	`p` and `k` values

Value

searcher efficiency that matches the output of ppersist

`SEsi0` *Calculate decayed searcher efficiency for a single pk*

Description

Calculate searcher efficiency after some searches for a single pk combination

Usage

```
SEsi0(days, pk)
```

Arguments

`days`	search days
`pk`	pk combination

Value

searcher efficiency that matches the output of ppersist

`SEsi_left` *Calculate conditional probability of observation at a search*

Description

Calculate the conditional probability of observing a carcass at search oi as a function arrival interval (assuming carcass is not removed by scavengers before the time of the final search)

Usage

```
SEsi_left(oi, pk, rng = NULL)
```

Arguments

`oi`	number of searches after arrival
`pk`	numeric array of searcher efficiency p and k parameters (p = pk[, 1] and k = pk[, 2])
`rng`	optional parameter giving the range of intervals to consider

Value

numeric array of probability of observing a carcass at oi for given that it arrived in intervals 1:oi if rng = NULL (or in intervals `rng`), assuming the carcass had not been previously discovered or removed by scavengers

---

`SEsi_right` *Calculate conditional probability of observation after a series of searches*

Description

Calculate the conditional probability of observing a carcass after i = 1:nsi searches (assuming carcass is not previous discovered by searchers or removed by scavengers)

Usage

```
SEsi_right(nsi, pk)
```

Arguments

`nsi`	number of searches after arrival
`pk`	numeric array of searcher efficiency p and k parameters (p = pk[, 1] and k = pk[, 2])

Value

numeric nsi x dim(pk)[1] array of probabilities of observing a carcass after 1:nsi searches (assuming that the carcass had not been previously discovered or removed by scavengers

---

`simpleMplot` *Plot a total mortality estimation for a simple situation*

Description

Function underneath `plot.estM`, which defines the plot method for a mortality object, composed of a hisogram with the empirical PDF and summary statistics

Usage

```
simpleMplot(M, ..., Xmin = 0, CL = 0.9)
```

Arguments

`M`	Mortality object
`...`	arguments to pass down
`Xmin`	minimum number of observable carcasses
`CL`	confidence level

`sizeCols` *Select the potential carcass class columns from a data table*

Description

Simple function to facilitate selection of columns that could be carcass class values from a data table.

Usage

```
sizeCols(data)
```

Arguments

`data`	data table

Value

column names of columns that can be carcass class values

`solar_powerTower` *Power Tower Example Dataset*

Description

An example data set for estimating fatalities from a concentrating solar-thermal (power tower) generation facility.

The simulated site consists of a single tower generating approximately 130 MW. The tower is surrounded by a 250 meter radius circular inner field of heliostats, searched on a weekly schedule. From the inner circle, 18 concentric rings of heliostats 50 meters deep extend to the boundaries of the simulated site. Rings are subdivided into 8 arcs each, with arcs 1-8 immediately adjacent to the central circle. Arcs are search using distance sampling techniques on a weekly schedule, with 29 arcs searched per weekday.

There are two sources of mortality simulated: flux and non-flux (collision or unknown cause).Flux carcasses are generated (weibull) about the tower, with 5% to be found in the outer field. Non-flux mortality is assumed uniform across the site.

The dataset consists of five parts: Data on carcass observations (CO) from inner and outer heliostat searches, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS), and density weighted proportion (DWP) of area searched at each turbine (which is an area adjustment factor to account for incomplete search coverage).

Usage

```
solar_powerTower
```

Format

`solar_powerTower` is a list with 5 elements:

`SE` Searcher efficiency trial data

`CP` Carcass persistence trial data

`SS` Search schedule parameters

`DWP` Density weighted proportion of area searched

`CO` Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`Field` indicates carcass placed in inner or outer heliostat field, with levels `"inner"` or `outer`.

`"Search1",...,"Search5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`cpID` unique ID for each carcass

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date an arc at the site was searched, a column of `SearchDates`, and a column for each arc, and one column at the end for the inner heliostat field, labeled `center`. In addition, there is a column to indicate the `Season`. A column with search dates and columns for each distinct area (arcs and center) searched are required. Other columns are optional.

- `SearchDate` columns of dates on which an arc was searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.
- `Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each arc and columns for each carcass size class (labels must match those of the class factors in the carcass observation file). Values represent the density-weighted proportion of the searched area for each size (or the fraction of carcasses that fall in the searched area). In this example, within the inner field (center) observers are unobstructed in ability to discover carcasses, for a DWP of 1. In the outer heliostat field observers walk along transects separated by 50 meters, but the entire area is surveyed, so DWP = 1.

- `Unit` unique ID for each arc, plus one labeled `center` for the inner heliostat field. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.
- `bat` DWP associated with size class Bat
- `sml` DWP associated with size class Small
- `med` DWP associated with size class Medium
- `lrg` DWP associated with size class Large

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

- `carcID` unique identifier for each carcass.
- `Unit` identifier for which unit the given carcass was found at: `"arc19"`, `"arc65"`, etc, for arcs in the outer heliostat field, or `"center"`, indicating the inner heliostat field.
- `Species` species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`
- `Size` size: `"bat"`, `"lrg"`, `"med"`, `"sml"`
- `Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`
- `Flux` An optional field indicating whether there Was evidence the animal was killed by flux. `"TRUE"`, or `"False"`.
- `Field` Optional indicator of whether the animal found in the `"inner"` or `"outer"` heliostat field?
- `Ring` Optional note animals found in the outer heliostat field indicating which concentric ring the carcass was found in.

`Distance` Optional note animals found in the outer heliostat field representing the perpendicular distance from the searcher the carcass was discovered at.

`DateFound` dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`

`X` Distance in meters from the Western edge of the facility.

`Y` Distance in meters from the Southern edge of the facility.

Source

`solar_powerTower`

`solar_PV` *Photovoltaic Example Dataset*

Description

An example data set for estimating fatalities from a large photovoltaic solar generation facility.

The simulated site is organized into 300 arrays of panels. As observers walk north-south along paths between arrays, they look east or west down rows between solar panels 150 meters long, with 38 searchable rows per array. Observers consistently look for animals down one cardinal direction, making this a one-sided distance sample. Searches are scheduled on a seven day rotation, with 60 arrays searched per weekday. A sitewide clearout search is implemented before the first scheduled winter search.

The dataset consists of five parts: Data on carcass observations (CO) from array searches, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS), and density weighted proportion (DWP) of area searched at each array (which is an area adjustment factor to account for incomplete search coverage).

Usage

`solar_PV`

Format

`solar_PV` is a list with 5 elements:

`SE` Searcher efficiency trial data

`CP` Carcass persistence trial data

`SS` Search schedule parameters

`DWP` Density weighted proportion of area searched

`CO` Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`"Search1",...,"Search5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

`Distance` the distance a carcass was placed from the observer's transect. Used in determining probability to detect with distance sampling.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`Index` unique ID for each carcass

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date an array at the site was searched, a column of `SearchDates`, and a column for each array. In addition, there is an optional column to indicate the `Season`. The columns for distinct area (array) and the date column are required, and the names of the columns for search areas must match the names of areas used in the DWP and CO files.

`SearchDate` columns of dates when arrays were searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each array and columns for each carcass size class (labels must match those of the class factors in the carcass observation file). Values represent the density-weighted proportion of the searched area for each size (or the fraction of carcasses that fall in the searched area). In this example, observers walk along transects separated by 150 meters, and search coverage is assumed to be 100 DWP = 1 for each unit. This requires that carcasses be placed at random locations in the field, even at distances from the transects that would make it unlikely to observe small carcasses.

`Unit` unique ID for each array. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.

`bat` DWP associated with size class Bat

`sml` DWP associated with size class Small

`med` DWP associated with size class Medium

`lrg` DWP associated with size class Large

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

`Index` unique identifier for each carcass.

`Unit` identifier for which unit the given carcass was found at: `"arc19"`, `"arc65"`, etc, for arcs in the outer heliostat field, or `"center"`, indicating the inner heliostat field.

`Species` species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`

`Size` size: `"bat"`, `"lrg"`, `"med"`, `"sml"`

`Row` Optional indicator of which row within an array a carcass was found at.

`Distance` The perpendicular distance from the searcher's transect at which the carcass was discovered at.

`DateFound` dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`

`X` UTM Easting of carcass.

`Y` UTM Northing of carcass.

Source

`solar_PV`

`solar_trough` *Trough-based solar thermal power simulated example*

Description

An example data set for estimating fatalities from a trough-based solar thermal electric power generation facility. The simulated site is inspected daily along ten 2000 meter long transects, which run north-south. Observers look up to 150 meters away down the rows created by troughs (east-west). One sided distance sampling will be used, with observers looking consistently in one cardinal direction as they travel through the facility. A sitewide clearout search is implemented before the first scheduled winter search.

The dataset consists of five parts: Data on carcass observations (CO) from daily searches, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS), and density weighted proportion (DWP) of area searched for the rows within each transect (which is an area adjustment factor to account for incomplete search coverage).

Usage

```
solar_trough
```

Format

`solar_trough` is a list with 5 elements:

`SE` Searcher efficiency trial data

`CP` Carcass persistence trial data

`SS` Search schedule parameters

`DWP` Density weighted proportion of area searched

`CO` Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`"Search1",...,"Search5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

`Distance` the distance a carcass was placed from the observer's transect.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`Index` unique ID for each carcass

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"`

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date a transect at the site was searched, a column of `SearchDates`, and a column for each transect. In addition, there is an optional column to indicate the `Season`. The columns for distinct area (array) and the date column are required, and the names of the columns for search areas must match the names of areas used in the DWP and CO files.

`SearchDate` columns of dates when a transect was searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.

`Season` `"winter"`, `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each transect and columns for each carcass size class (labels must match those of the class factors in the carcass observation file). Values represent the density-weighted proportion of the searched area for each size (or the fractionof carcasses that fall in the searched area). Since the whole site was searched, DWP is uniformly set equal to 1.

`Unit` unique ID for each transect. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.

`bat` DWP associated with size class Bat

`sml` DWP associated with size class Small

`med` DWP associated with size class Medium

`lrg` DWP associated with size class Large

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

`Index` unique identifier for each carcass.

`Unit` identifier for which transect the given carcass was found at. Values must match with DWP Transect values Search Schedule column names.

`Species` species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`

`Size` size: `"bat"`, `"lrg"`, `"med"`, `"sml"`

`Row` Optional indicator of which row within an array a carcass was found at.

`Distance` The perpendicular distance from the searcher's transect at which the carcass was discovered at.

`DateFound` dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`)

`X` UTM Easting of carcass.

`Y` UTM Northing of carcass.

Source

`solar_trough`

`summary.estM` *Summarize total mortality estimation*

Description

`summary` defined for class `estM` objects

Usage

```
## S3 method for class 'estM'
summary(object, ..., CL = 0.9)
```

Arguments

`object`	`estM` object
`...`	arguments to pass down
`CL`	confidence level

`summary.gGeneric` *Summarize the gGeneric list to a simple table*

Description

methods for `summary` applied to a `gGeneric` list

Usage

```
## S3 method for class 'gGeneric'
summary(object, ..., CL = 0.9)
```

Arguments

`object`	gGeneric output list (each element is a named vector of gGeneric values for a cell in the model combinations)
`...`	arguments to be passed down
`CL`	confidence level

Value

a summary table of g values (medians and confidence bounds) for each cell combination within the gGeneric list

Examples

```
data(mock)
model_SE <- pkm(formula_p = p ~ HabitatType, formula_k = k ~ 1,
              data = mock$SE)
model_CP <- cpm(formula_l = l ~ Visibility, formula_s = s ~ Visibility,
              data = mock$CP, left = "LastPresentDecimalDays",
              right = "FirstAbsentDecimalDays")
avgSS <- averageSS(mock$SS)
ghatsGeneric <- estgGeneric(nsim = 1000, avgSS, model_SE, model_CP)
summary(ghatsGeneric)
```

`summary.gGenericSize` *Summarize the gGenericSize list to a list of simple tables*

Description

methods for `summary` applied to a `gGenericSize` list

Usage

```
## S3 method for class 'gGenericSize'
summary(object, ..., CL = 0.9)
```

Arguments

`object`	gGenericSize output list (each element is a size-named list of named vectors of gGeneric values for a cell in the model combinations)
`...`	arguments to be passed down
`CL`	confidence level

Value

a list of summary tables of g values (medians and confidence bounds) for each cell combination within the gGeneric list

Examples

```
  data(mock)
  pkmModsSize <- pkm(formula_p = p ~ HabitatType,
                   formula_k = k ~ HabitatType, data = mock$SE,
                   obsCol = c("Search1", "Search2", "Search3", "Search4"),
                   sizeCol = "Size", allCombos = TRUE)
  cpmModsSize <- cpm(formula_l = l ~ Visibility,
                   formula_s = s ~ Visibility, data = mock$CP,
                   left = "LastPresentDecimalDays",
                   right = "FirstAbsentDecimalDays",
                   dist = c("exponential", "lognormal"),
                   sizeCol = "Size", allCombos = TRUE)
  pkMods <- c("S" = "p ~ 1; k ~ 1", "L" = "p ~ 1; k ~ 1",
             "M" = "p ~ 1; k ~ 1", "XL" = "p ~ 1; k ~ 1"
            )
  cpMods <- c("S" = "dist: exponential; l ~ 1; NULL",
              "L" = "dist: exponential; l ~ 1; NULL",
              "M" = "dist: exponential; l ~ 1; NULL",
              "XL" = "dist: exponential; l ~ 1; NULL"
            )
  avgSS <- averageSS(mock$SS)
  gsGeneric <- estgGenericSize(nsim = 1000, days = avgSS,
                 modelSetSize_SE = pkmModsSize,
                 modelSetSize_CP = cpmModsSize,
                 modelSizeSelections_SE = pkMods,
                 modelSizeSelections_CP = cpMods
               )
 summary(gsGeneric)
```

`summary.splitFull` *Summarize results of mortality estimate splits*

Description

Mortality estimates can be calculated for the various levels of splitting covariates such as season, species, or visibility class using `calcSplits`, which gives full arrays of simulated M estimates (i.e., for each level of each splitting covariate, each discovered carcass, and each simulation draw). summary(splits, CL = 0.90, ...) gives summary statistics of the estimates.

Usage

```
## S3 method for class 'splitFull'
summary(object, CL = 0.9, ...)
```

Arguments

`object`	A `splitFull` object (`calcSplits`) that gives simulated mortality estimates for all combinations of levels of 1 or 2 splitting covariates.
`CL`	desired confidence level for summary CIs (numeric scalar in (0, 1))
`...`	to be passed down

Value

an object of class `splitSummary`, which gives 5-number summaries for all combinations of levels among the splitting covariates in the `splits`. The 5-number summaries include the mean and alpha/2, 0.25, 0.5, 0.75, and 1 - alpha/2 quantiles of mortality estimates, where alpha = 1 - CL. A graphical representation of the results can be produced using `plot(splits,CL,...)`. For splits along CO covariates, the levels are organized alphabetically (but with numeric suffixes appearing in numeric order, e.g., "t1", "t2", "t10" rather than "t1", "t10", "t2").

`tidyModelSetCP` *Tidy a CP model set*

Description

Remove bad fit models

Usage

```
tidyModelSetCP(modelSet)
```

Arguments

`modelSet`	cp model set of class cpmSet

Value

a trimmed model set

`tidyModelSetSE` *Tidy an SE model set*

Description

Remove bad fit models

Usage

```
tidyModelSetSE(modelSet)
```

Arguments

`modelSet`	pk model set of class pkmSet

Value

a trimmed model set

`transposeSplits` *Transpose a* `splitFull` *array (preserving attributes)*

Description

Transpose a `splitFull` array (preserving attributes)

Usage

```
transposeSplits(splits)
```

Arguments

`splits`	a `splitFull` object, which is a list of `n` `m` x `k` arrays with attributes describing characteristics of the splits

Value

a list of `m` `n` x `k` arrays as a `splitFull` object

`trimSetSize` *Trim a Model-Set-Size Complex to a Single Model Per Size*

Description

Select a single model from each carcass class (based on the model names).

Usage

```
trimSetSize(modSetSize, mods)
```

Arguments

`modSetSize`	modSetSize complex (cpm or pkm)
`mods`	named (according to carcass classes) vector of model names to use

Value

modSetSize reduced to a single model per carcass class

`update_input` *Update the inputs when an event occurs*

Description

When an event occurs in the GenEst GUI, the input values may need to be updated. This function contains all of the possible updates based on the event options (or lacks any updates if the event doesn't require any).

Usage

```
update_input(eventName, rv, input, session)
```

Arguments

`eventName`	Character name of the event. One of "clear_all", "file_SE", "file_SE_clear", "file_CP", "file_CP_clear", "file_SS", "file_SS_clear", "file_DWP", "file_DWP_clear", "file_CO", "file_CO_clear", "class", "obsSE", "predsSE", "run_SE", "run_SE_clear", "outSEclass", "outSEp", "outSEk", "ltp", "fta", "predsCP", "run_CP", "run_CP_clear", "outCPclass", "outCPdist", "outCPl", "outCPs", "run_M", "run_M_clear", "split_M", "split_M_clear", "transpose_split", "run_g", "run_g_clear", or "outgclass".
`rv`	Reactive values list for the GenEst GUI.
`input`	`input` list for the GenEst GUI.
`session`	Environment for the GenEst GUI.

`update_output` *Update the outputs when an event occurs*

Description

When an event occurs in the GenEst GUI, the output values may need to be updated. This function contains all of the possible updates based on the event options (or lacks any updates if the event doesn't require any).

Usage

```
update_output(eventName, rv, output, input)
```

Arguments

`eventName`	Character name of the event. One of "clear_all", "file_SE", "file_SE_clear", "file_CP", "file_CP_clear", "file_SS", "file_SS_clear", "file_DWP", "file_DWP_clear", "file_CO", "file_CO_clear", "class", "obsSE", "predsSE", "run_SE", "run_SE_clear", "outSEclass", "outSEp", "outSEk", "ltp", "fta", "predsCP", "run_CP", "run_CP_clear", "outCPclass", "outCPdist", "outCPl", "outCPs", "run_M", "run_M_clear", "split_M", "split_M_clear", "transpose_split", "run_g", "run_g_clear", or "outgclass".
`rv`	Reactive values list for the GenEst GUI.
`output`	`output` list for the GenEst GUI.
`input`	`input` lisst for the GenEst GUI

Value

Updated `output` list.

`update_rv` *Update the reactive value list when an event occurs*

Description

When an event occurs in the GenEst GUI, the reactive values need to be updated. This function contains all of the possible updates based on the event options.

Usage

```
update_rv(eventName, rv, input)
```

Arguments

`eventName`	Character name of the event. One of "clear_all", "file_SE", "file_SE_clear", "file_CP", "file_CP_clear", "file_SS", "file_SS_clear", "file_DWP", "file_DWP_clear", "file_CO", "file_CO_clear", "class", "obsSE", "predsSE", "run_SE", "run_SE_clear", "outSEclass", "outSEp", "outSEk", "ltp", "fta", "predsCP", "run_CP", "run_CP_clear", "outCPclass", "outCPdist", "outCPl", "outCPs", "run_M", "run_M_clear", "split_M", "split_M_clear", "transpose_split", "run_g", "run_g_clear", "outgclass", "load_RP", "load_RPbat", "load_cleared", "load_PV", "load_trough", "load_powerTower", or "load_mock"
`rv`	Reactive values list for the GenEst GUI, created by `initialReactiveValues`, which calls `reactiveValues`
`input`	`input` list for the GenEst GUI.

Value

Updated `rv` list.

`wind_cleared` *Wind cleared plot (60m) Search Example*

Description

A complete example data set for estimating fatalities from 60 m cleared plots at 23 out of 100 searches at a wind power facility. Data on carcass observations (CO) from a search of all terrain out to 60m from each of 100 turbines at a theoretical site, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS) parameters (for example, which turbines were searched on which days), and density weighted proportion (DWP) of area searched at each turbine (which is an area adjustment factor to account for incomplete search coverage).

Usage

```
wind_cleared
```

Format

`wind_cleared` is a list with 5 elements:

`SE`	Searcher efficiency trial data
`CP`	Carcass persistence trial data
`SS`	Search schedule parameters
`DWP`	Density weighted proportion of area searched
`CO`	Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`pkID` unique ID for each carcass

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`Season` `"spring"`, `"summer"`, or `"fall"`

`Visibility` indicator for visibility class of the ground, with `"RP"` for carcasses placed on a road or turbine pad, `"M"` for moderate visibility (e.g., plowed field; short, sparse vegetation), or `"D"` for difficult visibility

`"s1",...,"s5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "SN" could be used to indicate that the turbine was not searched because of snow, or "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`cpID` unique ID for each carcass

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`Season` `"spring"`, `"summer"`, or `"fall"`

`Visibility` indicator for visibility class of the ground, with `"RP"` for carcasses placed on a road or turbine pad, `"M"` for moderate visibility (e.g., plowed field; short, sparse vegetation), or `"D"` for difficult visibility.

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date a turbine at the site was searched, a column of `SearchDates`, and a column for each turbine. In addition, there is a column to indicate the `Season`. A column with search dates and columns for each turbine searched are required. Other columns are optional.

`SearchDate` columns of dates on which at least one turbine was searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.

`Season` `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

`t1`, etc. unique ID for all turbines that were searched on at least one search date. Values are either 1 or 0, indicating whether the given turbine (column) was searched or not on the given date (row).

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each turbine and columns for each carcass size class. Values represent the density-weighted proportion of the searched area for each size (or the fraction of carcasses that fall in the searched area).

`Turbine` unique ID for each turbine. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.

`Size` `bat`, `sml`, `med`, `lrg`

`Season` `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

`carcID` unique identifier for each carcass: `"x30"`, `"x46"`, etc.

`Turbine` identifier for which turbine the given carcass was found at: `"t19"`, `"t65"`, `"t49"`, etc.

`TurbineType` the type of turbine: `"X"`, `"Y"` or `"Z"`.

`DateFound` dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`)

`Visibility` visibility class: `"RP"`, `"M"`, or `"D"`, as described in `$CP` and `$SE`

`Species` species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`

`SpeciesGroup` species group: `"bat0"`, `"bat1"`, `"brd1"`, `"brd2"`, `"brd3"`

`Size` size: `"bat"`, `"lrg"`, `"med"`, `"sml"`

`Distance` distance from the turbine

Source

`wind_cleared`

`wind_RP` *Wind Road and Pad (120m) Example*

Description

This example dataset is based on 120 m radius road and pad searches of all 100 turbines at a theoretical site. The simulated site consists of 100 turbines, searched on roads and pads only, out to 120 meters. Search schedule differs by turbine and season, with more frequent searches in the fall, and a subset of twenty turbines searched at every scheduled search.

Data on carcass observations (CO) from searches, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS) parameters (for example, which turbines were searched on which days), and density weighted proportion (DWP) of area searched at each turbine (which is an area adjustment factor to account for incomplete search coverage).

Usage

`wind_RP`

Format

`wind_RP` is a list with 5 elements:

`SE`	Searcher efficiency trial data
`CP`	Carcass persistence trial data
`SS`	Search schedule parameters
`DWP`	Density weighted proportion of area searched
`CO`	Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`pkID` unique ID for each carcass

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird

`Season` `"spring"`, `"summer"`, or `"fall"`

`"s1",...,"s5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "SN" could be used to indicate that the turbine was not searched because of snow, or "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`cpID` unique ID for each carcass

`Size` `"bat"`; or `"lrg"`, `"med"`, or `"sml"` bird.

`Season` `"spring"`, `"summer"`, or `"fall"`

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date a turbine at the site was searched, a column of `SearchDates`, and a column for each turbine. In addition, there is a column to indicate the `Season`. A column with search dates and columns for each turbine searched are required. Other columns are optional.

`SearchDate` columns of dates on which at least one turbine was searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.

`Season` `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

`t1`, etc. unique ID for all turbines that were searched on at least one search date. Values are either 1 or 0, indicating whether the given turbine (column) was searched or not on the given date (row).

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each turbine and columns for each carcass size class. Values represent the density-weighted proportion of the searched area for each size (or the fraction of carcasses that fall in the searched area).

`Turbine` unique ID for each turbine. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.

`bat` DWP associated with size class Bat.

`sml` DWP associated with size class Small.

`med` DWP associated with size class Medium.

`lrg` DWP associated with size class Large.

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

`carcID` unique identifier for each carcass: `"x30"`, `"x46"`, etc.

`Turbine` identifier for which turbine the given carcass was found at: `"t19"`, `"t65"`, `"t49"`, etc.

`TurbineType` the type of turbine: `"X"`, `"Y"` or `"Z"`.

`DateFound` dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`

`Species` species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`

`SpeciesGroup` species group: `"bat0"`, `"bat1"`, `"brd1"`, `"brd2"`, `"brd3"`

`Size` size: `"bat"`, `"lrg"`, `"med"`, `"sml"`

`Distance` distance from the turbine

Source

`wind_RP`

`wind_RPbat` *Wind Bat-Only Road and Pad (120m) Example*

Description

This example dataset considers only bats found on 120 m radius road and pad searches of all 100 turbines at a theoretical site. The simulated site consists of 100 turbines, searched on roads and pads only, out to 120 meters. Search schedule differs by turbine and season, with more frequent searches in the fall, and a subset of twenty turbines searched at every scheduled search.

Data on carcass observations (CO) from searches, field trials for estimating carcass persistence (CP) and searcher efficiency (SE), search schedule (SS) parameters (for example, which turbines were searched on which days), and density weighted proportion (DWP) of area searched at each turbine (which is an area adjustment factor to account for incomplete search coverage).

Usage

```
wind_RPbat
```

Format

`wind_RPbat` is a list with 5 elements:

`SE` Searcher efficiency trial data

`CP` Carcass persistence trial data

`SS` Search schedule parameters

`DWP` Density weighted proportion of area searched

`CO` Carcass observations

Searcher Efficiency (`SE`)

`$SE` is a data frame with each row representing the fate of a single carcass in the searcher efficiency trials. There are columns for:

`pkID` unique ID for each carcass

`Season` `"spring"`, `"summer"`, or `"fall"`

`"s1",...,"s5"` fate of carcass on the 1st, 2nd, 3rd, 4th, and 5th search after placement. A value of 1 implies that a carcass was discovered by searchers, 0 implies the carcass was present but not discovered, and any other value is interpreted as "no search" or "carcass not present" and ignored in the model. In this data set, `NA` indicates that a carcass had been previously discovered and removed from the field. A user may use a variety of values to differentiate different reasons no search was conducted or the carcass was not present. For example, "SN" could be used to indicate that the turbine was not searched because of snow, or "NS" to indicate the search was not scheduled in that location at that time, or "SC" to indicate the carcass had been removed by scavengers prior to the search.

Carcass Persistence (`CP`)

`$CP` is a data frame with each row representing the fate of a single carcass in the carcass persistence trials. There are columns for:

`cpID` unique ID for each carcass

`Season` `"spring"`, `"summer"`, or `"fall"`

`LastPresent`, `FirstAbsent` endpoints of the interval bracketing the time the carcass was scavenged or otherwise removed from the field. For example, `LastPresent = 2.04`, `FirstAbsent = 3.21` indicates that the carcass was last observed 2.04 days after being placed in the field and was noted missing 3.21 days after being placed. If the precise time of carcass removal is known (e.g., recorded by camera), then `LastPresent` and `FirstAbsent` should be set equal to each other. If a carcass persists beyond the last day of the field trial, `LastPresent` is the last time it was observed and `FirstAbsent` is entered as `Inf` or `NA`.

Search Schedule (`SS`)

`$SS` is a data frame with a row for each date a turbine at the site was searched, a column of `SearchDates`, and a column for each turbine. In addition, there is a column to indicate the `Season`. A column with search dates and columns for each turbine searched are required. Other columns are optional.

`SearchDate` columns of dates on which at least one turbine was searched. Format in this data is `"%Y-%m-%d CDT"`, but time zone (`CDT`) is optional. A time stamp may be included if desired (e.g., `2018-03-20 02:15:41`). Alternatively, `\` can be used in place of `-`.

`Season` `"spring"`, `"summer"`, or `"fall"` to indicate which season the search was conducted in. `Season` is optional but may be used as a temporal covariate for fatality estimates.

`t1`, **etc.** unique ID for all turbines that were searched on at least one search date. Values are either 1 or 0, indicating whether the given turbine (column) was searched or not on the given date (row).

Density Weighted Proportion (`DWP`)

`$DWP` is a data frame with a row for each turbine and columns for each carcass size class. Values represent the density-weighted proportion of the searched area for each size (or the fraction of carcasses that fall in the searched area).

`Turbine`	unique ID for each turbine. IDs match those used in the `$CO` data frame and the column names in the `$SS` data.
`bat`	Contains the DWP for each turbine, with respect to size class (in this case, bats only.

Carcass Observations (`CO`)

`$CO` is a data frame with a row for carcass observed in the carcass searches and a number of columns giving information about the given carcass (date found, size, species, etc.)

`carcID`	unique identifier for each carcass: `"x30"`, `"x46"`, etc.
`Turbine`	identifier for which turbine the given carcass was found at: `"t19"`, `"t65"`, `"t49"`, etc.
`TurbineType`	the type of turbine: `"X"`, `"Y"` or `"Z"`.
`DateFound`	dates entered in the same format as in `$SS$SearchDate`. Every date entered here is (and must be) included in the search schedule (`$SS$SearchDate`
`Species`	species of the carcass: `"BA"`, `"BB"`, `"BC"`, `"BD"`, `"BE"`, `"LA"`, `"LB"`, `"LD"`, `"LE"`, `"MA"`, `"MB"`, `"SA"`, `"SB"`, `"SC"`, `"SD"`, `"SE"`, `"SF"`, `"SG"`
`SpeciesGroup`	species group: `"bat0"`, `"bat1"`, `"brd1"`, `"brd2"`, `"brd3"`
`Distance`	Distance from the turbine.

Source

`wind_RPbat`

Index