Package 'FSA'

November 22, 2018

Version 0.8.22

Date 2018-11-22

Title Simple Fisheries Stock Assessment Methods

Description A variety of simple fish stock assessment methods.
Detailed vignettes are available on the fishR website <http://derekogle.com/fishR/>.

URL https://github.com/droglenc/FSA

BugReports https://github.com/droglenc/FSA/issues

License GPL (>= 2)

LazyData true

Depends R (>= 3.2.0)

Imports graphics, grDevices, stats, tools, utils, car, dplyr,
dunn.test, epitools, gplots, lmtest, plotrix, plyr, sciplot,
withr

Suggests asbio, DescTools, fishmethods, FSAdata, knitr, marked, nlme,
nlstools, pkgdown, psych, Rcapture, testthat, rmarkdown

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Author Derek Ogle [aut, cre] (<https://orcid.org/0000-0002-0370-9299>),
Powell Wheeler [aut] (Added method='Burnham' to removal()),
Alexis Dinno [aut] (Provided base functionality of dunnTest())

Maintainer Derek Ogle <derek@derekogle.com>

Repository CRAN

Date/Publication 2018-11-22 16:30:06 UTC

R topics documented:

`addZeroCatch` *Adds zeros for catches of species not collected in some sampling events.*

Description

Adds zeros for catches of species that were not captured in a sampling event but were captured in at least one other sampling event (i.e., adds zeros to the data.frame for capture events where a species was not observed).

Usage

```
addZeroCatch(df, eventvar, specvar, zerovar, na.rm = TRUE)
```

Arguments

`df`	A data.frame that contains the capture summary data as described in the details.
`eventvar`	A string for the variable that identifies unique capture events.
`specvar`	A string or vector of strings for the variable(s) that identify the "species" (if multiple variables, could be species, sex, and life stage, for example) captured. See examples.
`zerovar`	A string or vector of strings for the variable(s) that should be set equal to zero. See details and examples.
`na.rm`	A logical that indicates if rows where `specvar` that are `NA` should be removed after adding the zeros. See details.

Details

The data.frame in `df` must contain a column that identifies a unique capture event (given in `eventvar`), a column with the name for the species captured (given in `specvar`), and a column that contains the number of that species captured (potentially given to `zerovar`; see details). All sampling event and species combinations where catch information does not exist is identified and a new data.frame that contains a zero for the catch for all of these combinations is created. This new data.frame is appended to the original data.frame to construct a data.frame that contains complete catch information – i.e., including zeros for species in events where that species was not captured.

The data.frame may contain other information related to the catch, such as number of recaptured fish, number of fish released, etc. These additional variables can be included in `zerovar` so that zeros will be added to these variables as well (e.g., if the catch of the species is zero, then the number of recaptures must also be zero). All variables not given in `eventvar`, `specvar`, or `zerovar` will be assumed to be related to `eventvar` and `specvar` (e.g., date, gear type, and habitat) and, thus, will be repeated with these variables.

In situations where no fish were captured in some events, the `df` may contain rows that have a value for `eventvar` but not for `specvar`. These rows are important because zeros need to be added for each observed species for these events. However, in these situations, a `<NA>` species will appear in the resulting data.frame. It is unlikely that these "missing" species are needed so they will be removed if `na.rm=TRUE` (DEFAULT) is used.

One should test the results of this function by creating a frequency table of the `eventvar` or `specvar`. In either case, the table should contain the same value in each cell of the table. See the examples.

Value

A data.frame with the same structure as `df` but with rows of zero observation data appended.

IFAR Chapter

2-Basic Data Manipulations

Note

An error will be returned if either `specvar` or `eventvar` are factors with any `NA` levels. This usually arises if the data.frame was subsetted/filtered prior to using `addZeroCatch`. See `filterD` or `droplevels` for descriptions of how to drop unused levels.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

`complete` in **tidyr** package.

Examples

```
## Example Data #1 (some nets missing some fish, ancillary net data)
df1 <- data.frame(net=c(1,1,1,2,2,3),
                  eff=c(1,1,1,1,1,1),
                  species=c("BKT","LKT","RBT","BKT","LKT","RBT"),
                  catch=c(3,4,5,5,4,3))
df1
# not all 1s
xtabs(~net+species,data=df1)

df1mod1 <- addZeroCatch(df1,"net","species",zerovar="catch")
df1mod1
# check, should all be 3
xtabs(~net,data=df1mod1)
# check, should all be 1
xtabs(~net+species,data=df1mod1)
# correct mean/sd of catches
Summarize(catch~species,data=df1mod1)
# incorrect mean/sd of catches (no zeros)
Summarize(catch~species,data=df1)
```

```
# Same as example 1 but with no ancillary data specific to the net number
df2 <- df1[,-2]
df2
df1mod2 <- addZeroCatch(df2,"net","species",zerovar="catch")
df1mod2
# check, should all be 1
xtabs(~net+species,data=df1mod2)

## Example Data #3 (All nets have same species ... no zeros needed)
df3 <- data.frame(net=c(1,1,1,2,2,2,3,3,3),
                  eff=c(1,1,1,1,1,1,1,1,1),
                  species=c("BKT","LKT","RBT","BKT","LKT",
                            "RBT","BKT","LKT","RBT"),
                  catch=c(3,4,5,5,4,3,3,2,7))
df3
# should all be 1 for this example
xtabs(~net+species,data=df3)

# should receive a warning and table should still all be 1
df3mod1 <- addZeroCatch(df3,"net","species",zerovar="catch")
xtabs(~net+species,data=df3mod1)

## Example Data #4 (another variable that needs zeros)
df4 <- df1
df4$recaps <- c(0,0,0,1,2,1)
df4
# not all 1s
xtabs(~net+species,data=df4)

df4mod1 <- addZeroCatch(df4,"net","species",zerovar=c("catch","recaps"))
# note zeros in both variables
df4mod1
# check, should all be 1
xtabs(~net+species,data=df4mod1)
# observe difference from next
Summarize(catch~species,data=df4)
Summarize(catch~species,data=df4mod1)
# observe difference from next
Summarize(recaps~species,data=df4)
Summarize(recaps~species,data=df4mod1)

## Example Data #5 (two "specvar"s)
df5 <- df1
df5$sex <- c("m","m","f","m","f","f")
df5
# not all 1s
xtabs(~sex+species+net,data=df5)

df5mod1 <- addZeroCatch(df5,"net",c("species","sex"),zerovar="catch")
df5mod1
# all 1s
xtabs(~sex+species+net,data=df5mod1)
str(df5mod1)
```

```
## Example Data #6 (three "specvar"s)
df6 <- df5
df6$size <- c("lrg","lrg","lrg","sm","lrg","sm")
df6

df6mod1 <- addZeroCatch(df6,"net",c("species","sex","size"),zerovar="catch")
df6mod1
```

---

ageBias *Compute and view possible differences between paired sets of ages.*

Description

Constructs age-agreement tables, statistical tests to detect differences, and plots to visualize potential differences in paired age estimates. Ages may be from, for example, two readers of the same structure, one reader at two times, two structures (e.g., scales, spines, otoliths), or one structure and known ages.

Usage

```
ageBias(formula, data, ref.lab = tmp$Enames, nref.lab = tmp$Rname,
  method = stats::p.adjust.methods, sig.level = 0.05, min.n.CI = 3)

## S3 method for class 'ageBias'
summary(object, what = c("table", "symmetry", "Bowker",
  "EvansHoenig", "McNemar", "bias", "diff.bias", "n"),
  flip.table = FALSE, zero.print = "-", digits = 3,
  cont.corr = c("none", "Yates", "Edwards"), ...)

## S3 method for class 'ageBias'
plot(x, xvals = c("reference", "mean"),
  xlab = ifelse(xvals == "reference", x$ref.lab, "Mean Age"),
  ylab = paste(x$nref.lab, "-", x$ref.lab), xlim = NULL, ylim = NULL,
  yaxt = graphics::par("yaxt"), xaxt = graphics::par("xaxt"),
  col.agree = "gray60", lwd.agree = lwd, lty.agree = 2, lwd = 1,
  sfrac = 0, show.pts = NULL, pch.pts = 20, cex.pts = ifelse(xHist
  | yHist, 1.5, 1), col.pts = "black", transparency = 1/10,
  show.CI = FALSE, col.CI = "black", col.CIsig = "red",
  lwd.CI = lwd, sfrac.CI = sfrac, show.range = NULL,
  col.range = ifelse(show.CI, "gray70", "black"), lwd.range = lwd,
  sfrac.range = sfrac, pch.mean = 19, pch.mean.sig = ifelse(show.CI |
  show.range, 21, 19), cex.mean = lwd, yHist = TRUE, xHist = NULL,
  hist.panel.size = 1/7, col.hist = "gray90", allowAdd = FALSE, ...)
```

Arguments

formula	A formula of the form nrefvar~refvar, where nrefvar and refvar generically represent variables that contain the paired "nonreference" and "reference" age estimates, respectively. See details.
data	A data.frame that minimally contains the paired age estimates given in formula.
ref.lab	A string label for the reference age estimates.
nref.lab	A string label for the nonreference age estimates.
method	A string for which method to use to adjust p-values for multiple comparisons. See ?p.adjust.methods.
sig.level	A numeric value to determine whether a p-value indicates a significant result. The confidence level used in plot is 100*(1-sig.level).
min.n.CI	A numeric value (default is 3) that is the smallest sample size for which a confidence interval will be computed.
what	A string that indicates what type of summary to print or plot to construct. See details.
flip.table	A logical that indicates whether the age-agreement table should be 'flipped' (i.e., rows are reversed so that younger ages are at the bottom of the table). This makes the table more directly comparable to the age bias plot.
zero.print	A string for what should be printed in place of the zeros on an age-agreement table. The default is to print a single dash.
digits	A numeric for the minimum number of digits to print when showing what="bias" or what="diff.bias" in summary.
cont.corr	A string that indicates the continuity correction method to be used with (only) McNemar test. If "none" (default) then no continuity correction is used, if "Yates" then 0.5 is used, and if "Edwards" then 1 is used.
...	Additional arguments for methods.
x, object	An object of class ageBias, usually a result from ageBias.
xvals	A string for whether the x-axis values are reference ages or mean of the reference and nonreference ages.
xlab, ylab	A string label for the x-axis (reference) or y-axis (non-reference) age estimates, respectively.
xlim, ylim	A numeric vector of length 2 that contains the limits of the x-axis (reference ages) or y-axis (non-reference ages), respectively.
xaxt, yaxt	A string which specifies the x- and y-axis types. Specifying "n" suppresses plotting of the axis. See ?par.
col.agree	A string or numeric for the color of the 1:1 or zero (if difference=TRUE) reference line.
lwd.agree	A numeric for the line width of the 1:1 or zero (if difference=TRUE) reference line.
lty.agree	A numeric for the line type of the 1:1 or zero (if difference=TRUE) reference line.
lwd	A numeric that controls the separate 'lwd' argument (e.g., lwd.CI and lwd.range).

sfrac	A numeric that controls the separate 'sfrac' arguments (e.g., sfrac.CI and sfrac.range). See sfrac in plotCI of **plotrix**.
show.pts	A logical for whether or not the raw data points are plotted.
pch.pts	A numeric for the plotting character of the raw data points.
cex.pts	A character expansion value for the size of the symbols for pch.pts.
col.pts	A string or numeric for the color of the raw data points. The default is to use black with the transparency found in transparency.
transparency	A numeric (between 0 and 1) for the level of transparency of the raw data points. This number of points plotted on top of each other will represent the color in col.pts.
show.CI	A logical for whether confidence intervals should be plotted or not.
col.CI	A string or numeric for the color of confidence interval bars that are considered non-significant.
col.CIsig	A string or numeric for the color of confidence interval bars that are considered significant.
lwd.CI	A numeric for the line width of the confidence interval bars.
sfrac.CI	A numeric for the size of the ends of the confidence interval bars. See sfrac in plotCI of **plotrix**.
show.range	A logical for whether or not vertical bars that represent the range of the data points are plotted.
col.range	A string or numeric for the color of the range intervals.
lwd.range	A numeric for the line width of the range intervals.
sfrac.range	A numeric for the size of the ends of the range intervals. See sfrac in plotCI of **plotrix**.
pch.mean	A numeric for the plotting character used for the mean values when the means are considered insignificant.
pch.mean.sig	A numeric for the plotting character for the mean values when the means are considered significant.
cex.mean	A character expansion value for the size of the mean symbol in pch.mean and pch.mean.sig.
yHist	A logical for whether a histogram of the y-axis variable should be added to the right margin of the age bias plot. See details.
xHist	A logical for whether a histogram of the x-axis variable should be added to the top margin of the age bias plot. See details.
hist.panel.size	A numeric between 0 and 1 that indicates the proportional size of histograms (relative to the entire plotting pane) in the plot margins (only used if xHist=TRUE or yHist=TRUE).
col.hist	A string for the color of the bars in the marginal histograms (only used if xHist=TRUE or yHist=TRUE).
allowAdd	A logical that will allow the user to add items to the main (i.e., not the marginal histograms) plot panel (if TRUE). Defaults to FALSE.

Details

Generally, one of the two age estimates will be identified as the "reference" set. In some cases this may be the true ages, the ages from the more experienced reader, the ages from the first reading, or the ages from the structure generally thought to provide the most accurate results. In other cases, such as when comparing two novice readers, the choice may be arbitrary. The reference ages will form the columns of the age-agreement table and will be the "constant" age used in the t-tests and age bias plots (i.e., the x-axis). See further details below.

The age-agreement table is constructed with `what="table"` in `summary`. The agreement table can be "flipped" (i.e., the rows in descending rather than ascending order) with `flip.table=TRUE`. By default, the tables are shown with zeros replaced by dashes. This behavior can be changed with `zero.print`.

Three statistical tests of symmetry for the age-agreement table can be computed with `what=` in `summary`. The "unpooled" or Bowker test as described in Hoenig *et al.* (1995) is constructed with `what="Bowker"`, the "semi-pooled" or Evans-Hoenig test as described in Evans and Hoenig (1998) is constructed with `what="EvansHoenig"`, and the "pooled" or McNemar test as described in Evans and Hoenig (1998) is constructed with `what="McNemar"`. All three tests are computed with `what="symmetry"`.

The age bias plot introduced by Campana *et al.* (1995) can be constructed with `plotAB`. Muir *et al.* (2008) modified the original age bias plot by plotting the difference between the two ages on the y-axis (still against a reference age on the x-axis). McBride (2015) introduced the Bland-Altman plot for comparing fish ages where the difference in ages is plotted on the y-axis versus the mean of the ages on the x-axis. Modifications of these plots are created with `plot` (rather than `plotAB`) using `xvals=` to control what is plotted on the x-axis (i.e., `xvals="reference"` uses the reference ages, whereas `xvals="mean"` uses the mean of the two ages for the x-axis). In both plots, a horizontal agreement line at a difference of zero is shown for comparative purposes. In addition, a histogram of the differences is shown in the right margin (can be excluded with `yHist=FALSE`.) A histogram of the reference ages is shown by default in the top margin for the modified age bias plot, but not for the modified Bland-Altman-like plot (the presence of this histogram is controlled with `xHist=`).

By default, the modified age bias plot shows the mean and range for the nonreference ages at each of the reference ages. Means shown with an open dot are mean age differences that are significantly different from zero. The ranges of differences in ages at each reference age can be removed with `show.range=FALSE`. A confidence interval for difference in ages can be added with `show.CI=FALSE`. Confidence intervals are only shown if the sample size is greater than the value in `min.n.CI=` (also from the original call to `ageBias`). Confidence intervals plotted in red with an open dot (by default; these can be changed with `col.CIsig` and `pch.mean.sig`, respectively) do not contain the reference age (see discussion of t-tests below). Individual points are plotted if `show.pts=TRUE`, where there color is controlled by `col.pts=` and `transparency=`. See examples for use of these arguments.

The main (i.e., not the marginal histograms) can be "added to" after the plot is constructed if `allowAdd=TRUE` is used. For example, the Bland-Altman-like plot can be augmented with a horizontal line at the mean difference in ages, a line for the regression between the differences and the means, or a generalized additive model that describes the relationship between the differences and the means. See the examples for use of `allowAdd=TRUE`. It is recommended to save the plotting parameters (e.g., `op <- par(no.readonly=TRUE)`) before using `plot` with `allowAdd=TRUE` so that the original parameters can be reset (e.g., `par(op)`); see the examples.

Individual t-tests to determine if the mean age of the nonreference set at a particular age of the

reference set is equal to the reference age (e.g., is the mean age of the nonreference set at age-3 of the reference set statistically different from 3?) are shown with `what="bias"` in `summary`. The results provide a column that indicates whether the difference is significant or not as determined by adjusted p-values from the t-tests and using the significance level provided in `sig.level` (defaults to 0.05). Similar results for the difference in ages (e.g., is the mean reference age minus nonreference age at a nonreference age of 3 different from 0?) are constructed with `what="diff.bias"` in `summary`.

The sample size present in the age-agreement table is found with `what="n"`.

Value

`ageBias` returns a list with the following items:

- data A data.frame with the original paired age estimates and the difference between those estimates.
- agree The age-agreement table.
- bias A data.frame that contains the bias statistics.
- bias.diff A data.frame that contains the bias statistics for the differences.
- ref.lab A string that contains an optional label for the age estimates in the columns (reference) of the age-agreement table.
- nref.lab A string that contains an optional label for the age estimates in the rows (non-reference) of the age-agreement table.

`summary` returns the result if `what=` contains one item, otherwise it returns nothing. Nothing is returned by `plot` or `plotAB`, but see details for a description of the plot that is produced.

Testing

Tested all symmetry test results against results in Evans and Hoenig (2008), the McNemar and Evans-Hoenig results against results from `compare2` in **fishmethods**, and all results using the `AlewifeLH` data set from **FSAdata** against results from the online resource at http://www.nefsc.noaa.gov/fbp/age-prec/.

IFAR Chapter

4-Age Comparisons. **Note that `plot` has changed since IFAR was published. Some of the original functionality is in `plotAB`.**

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Campana, S.E., M.C. Annand, and J.I. McMillan. 1995. Graphical and statistical methods for determining the consistency of age determinations. Transactions of the American Fisheries Society 124:131-138. [Was (is?) available from http://www.bio.gc.ca/otoliths/documents/Campana%20et%20al%201995%20TAFS.

Evans, G.T. and J.M. Hoenig. 1998. Testing and viewing symmetry in contingency tables, with application to readers of fish ages. Biometrics 54:620-629.

Hoenig, J.M., M.J. Morgan, and C.A. Brown. 1995. Analysing differences between two age determination methods by tests of symmetry. Canadian Journal of Fisheries and Aquatic Sciences 52:364-368.

McBride, R.S. 2015. Diagnosis of paired age agreement: A simulation approach of accuracy and precision effects. ICES Journal of Marine Science 72:2149-2167.

Muir, A.M., M.P. Ebener, J.X. He, and J.E. Johnson. 2008. A comparison of the scale and otolith methods of age estimation for lake whitefish in Lake Huron. North American Journal of Fisheries Management 28:625-635. [Was (is?) available from http://www.tandfonline.com/doi/abs/10.1577/M06-160.1]

See Also

See agePrecision for measures of precision between pairs of age estimates. See compare2 in **fishmethods** for similar functionality. See plotAB for a more traditional age-bias plot.

Examples

```
ab1 <- ageBias(scaleC~otolithC,data=WhitefishLC,
               ref.lab="Otolith Age",nref.lab="Scale Age")
summary(ab1)
summary(ab1,what="symmetry")
summary(ab1,what="Bowker")
summary(ab1,what="EvansHoenig")
summary(ab1,what="McNemar")
summary(ab1,what="McNemar",cont.corr="Yates")
summary(ab1,what="bias")
summary(ab1,what="diff.bias")
summary(ab1,what="n")
summary(ab1,what=c("n","symmetry","table"))
# flip table (easy to compare to age bias plot) and show zeroes (not dashes)
summary(ab1,what="table",flip.table=TRUE,zero.print="0")


############################################################
## Differences Plot (inspired by Muir et al. (2008))
# Default (ranges, open circles for sig diffs, marginal hists)
plot(ab1)
# Show CIs for means (with and without ranges)
plot(ab1,show.CI=TRUE)
plot(ab1,show.CI=TRUE,show.range=FALSE)
# show points (with and without CIs)
plot(ab1,show.CI=TRUE,show.range=FALSE,show.pts=TRUE)
plot(ab1,show.range=FALSE,show.pts=TRUE)
# Use same symbols for all means (with ranges)
plot(ab1,pch.mean.sig=19)
# Use same symbols/colors for all means/CIs (without ranges)
plot(ab1,show.range=FALSE,show.CI=TRUE,pch.mean.sig=19,col.CIsig="black")
# Remove histograms
plot(ab1,xHist=FALSE)
```

```
plot(ab1,yHist=FALSE)
plot(ab1,xHist=FALSE,yHist=FALSE)
## Suppress confidence intervals for n < a certain value
##   must set this in the original ageBias() call
ab2 <- ageBias(scaleC~otolithC,data=WhitefishLC,min.n.CI=8,
               ref.lab="Otolith Age",nref.lab="Scale Age")
plot(ab2,show.CI=TRUE,show.range=FALSE)


############################################################
## Differences Plot ( inspired by Bland-Altman plots in McBride (2015))
plot(ab1,xvals="mean")
## Modify axis limits
plot(ab1,xvals="mean",xlim=c(1,17))
## Add and remove histograms
plot(ab1,xvals="mean",xHist=TRUE)
plot(ab1,xvals="mean",xHist=TRUE,yHist=FALSE)
plot(ab1,xvals="mean",yHist=FALSE)

############################################################
## Adding post hoc analyses to the main plot
# get original graphing parameters to be reset at the end
op <- par(no.readonly=TRUE)

# get raw data
tmp <- ab1$d
head(tmp)

# Add mean difference (w/ approx. 95% CI)
bias <- mean(tmp$diff)+c(-1.96,0,1.96)*se(tmp$diff)
plot(ab1,xvals="mean",xlim=c(1,17),allowAdd=TRUE)
abline(h=bias,lty=2,col="red")
par(op)

# Same as above, but without marginal histogram, and with
#   95% agreement lines as well (1.96SDs)
#   (this is nearly a replicate of a Bland-Altman plot)
bias <- mean(tmp$diff)+c(-1.96,0,1.96)*se(tmp$diff)
agline <- mean(tmp$diff)+c(-1.96,1.96)*sd(tmp$diff)
plot(ab1,xvals="mean",yHist=FALSE,allowAdd=TRUE)
abline(h=bias,lty=2,col="red")
abline(h=agline,lty=3,lwd=2,col="blue")
par(op)

# Add linear regression line of differences on means (w/ approx. 95% CI)
lm1 <- lm(diff~mean,data=tmp)
xval <- seq(0,19,0.1)
pred1 <- predict(lm1,data.frame(mean=xval),interval="confidence")
bias1 <- data.frame(xval,pred1)
head(bias1)
plot(ab1,xvals="mean",xlim=c(1,17),allowAdd=TRUE)
lines(lwr~xval,data=bias1,lty=2,col="red")
lines(upr~xval,data=bias1,lty=2,col="red")
```

```
lines(fit~xval,data=bias1,lty=2,col="red")
par(op)

# Add loess of differences on means (w/ approx. 95% CI as a polygon)
lo2 <- loess(diff~mean,data=tmp)
xval <- seq(min(tmp$mean),max(tmp$mean),0.1)
pred2 <- predict(lo2,data.frame(mean=xval),se=TRUE)
bias2 <- data.frame(xval,pred2)
bias2$lwr <- bias2$fit-1.96*bias2$se.fit
bias2$upr <- bias2$fit+1.96*bias2$se.fit
head(bias2)
plot(ab1,xvals="mean",xlim=c(1,17),allowAdd=TRUE)
with(bias2,polygon(c(xval,rev(xval)),c(lwr,rev(upr)),
                   col=col2rgbt("red",1/10),border=NA))
lines(fit~xval,data=bias2,lty=2,col="red")
par(op)

# Same as above, but polygon and line behind the points
plot(ab1,xvals="mean",xlim=c(1,17),col.pts="white",allowAdd=TRUE)
with(bias2,polygon(c(xval,rev(xval)),c(lwr,rev(upr)),
                   col=col2rgbt("red",1/10),border=NA))
lines(fit~xval,data=bias2,lty=2,col="red")
points(diff~mean,data=tmp,pch=19,col=col2rgbt("black",1/8))
par(op)

# Can also be made with the reference ages on the x-axis
lo3 <- loess(diff~otolithC,data=tmp)
xval <- seq(min(tmp$otolithC),max(tmp$otolithC),0.1)
pred3 <- predict(lo3,data.frame(otolithC=xval),se=TRUE)
bias3 <- data.frame(xval,pred3)
bias3$lwr <- bias3$fit-1.96*bias3$se.fit
bias3$upr <- bias3$fit+1.96*bias3$se.fit
plot(ab1,show.range=FALSE,show.pts=TRUE,col.pts="white",allowAdd=TRUE)
with(bias3,polygon(c(xval,rev(xval)),c(lwr,rev(upr)),
                   col=col2rgbt("red",1/10),border=NA))
lines(fit~xval,data=bias3,lty=2,col="red")
points(diff~otolithC,data=tmp,pch=19,col=col2rgbt("black",1/8))
par(op)
```

`agePrecision` *Compute measures of precision among sets of ages.*

Description

Computes overall measures of precision for multiple age estimates made on the same individuals. Ages may be from two or more readers of the same structure, one reader at two or more times, or two or more structures (e.g., scales, spines, otoliths). Measures of precision include ACV (Average Coefficient of Variation), APE (Average Percent Error), AAD (Average Absolute Deviation), and ASD (Average Standard Devation), and various percentage difference values.

Usage

```
agePrecision(formula, data)

## S3 method for class 'agePrec'
summary(object, what = c("precision", "difference",
  "absolute difference", "details"), percent = TRUE, trunc.diff = NULL,
  digits = 4, show.prec2 = FALSE, ...)
```

Arguments

formula	A formula of the form ~var1+var2+var3+... or, alternatively, var1~var2+var3+..., where the varX generically represent the variables that contain the age estimates. The alternative formula allows for similar code as used in ageBias and can have only one variable on the left-hand side.
data	A data.frame that minimally contains the variables in formula.
object	An object of class agePrec, usually from agePrecision.
what	A string (or vector of strings) that indicates what type of summary to print. See details.
percent	A logical that indicates whether the difference table (see details) should be represented as percentages (TRUE; default) or frequency (FALSE) of fish.
trunc.diff	A single integer that identifies the age for which all values that age and greater are combined into one category. See the examples.
digits	A single numeric that indicates the minimum number of digits to print when using summary.
show.prec2	A logical that indicates whether the precision metrics that use the median (i.e., ACV2 and APE2) should be shown when only two age estimates were made (in this instance they will be exactly equal to ACV and APE). Default is to not show these values in this situation.
...	Additional arguments for methods.

Details

If what="precision" in summary then a summary table that contains the following items will be printed:

- n Number of fish in data.
- validn Number of fish in data that have non-NA data for all R age estimates.
- R Number of age estimates given in formula.
- PercAgree The percentage of fish for which all age estimates perfectly agree.
- ASD The average (across all fish) standard deviation of ages within a fish.
- ACV The average (across all fish) coefficient of variation of ages within a fish using the **mean** as the divisor. See the IFAR chapter for calculation details.
- ACV2 The average (across all fish) coefficient of variation of ages within a fish using the **median** as the divisor. This will only be shown if R>2 or show.prec2=TRUE.

- AAD The average (across all fish) absolute deviation of ages within a fish.
- APE The average (across all fish) percent error of ages within a fish using the **mean** as the divisor. See the IFAR chapter for calculation details.
- APE2 The average (across all fish) percent error of ages within a fish using the **median** as the divisor. This will only be shown if R>2 or `show.prec2=TRUE`.
- AD The average (across all fish) index of precision (D).

Note that ACV2 and APE2 will not be printed when `what="precision"` if only two sets of ages are given (because mean=median such that ACV=ACV2 and APE=APE2). If `what="difference"` is used in `summary`, then a table that describes either the percentage (if `percent=TRUE`, DEFAULT) or frequency of fish by the difference in paired age estimates. This table has one row for each possible pair of age estimates.

If `what="absolute difference"` is used in `summary`, then a table that describes either the percentage (if `percent=TRUE`, DEFAULT) or frequency of fish by the absolute value of the difference in paired age estimates. This table has one row for each possible pair of age estimates. The "1" column, for example, represents age estimates that disagree by one year (in either direction).

If `what="detail"` is used in `summary`, then a data.frame of the original `data` along with the intermediate calculations of the mean age, median age, modal age (will be `NA` if a mode does not exist (e.g., all different ages) or multiple modes exist), standard deviation of age (SD), coefficient of variation using the mean as the divisor (CV), coefficient of variation using the median as the divisor (CV2), absolute deviation using the mean as the divisor (AD), absolute deviation using the median as the divisor (AD2), average percent error (PE), and index of precision (D) for each individual will be printed.

All percentage calculations above use the `validn` value in the denominator.

Value

The main function returns a list with the following items:

- detail A data.frame with all data given in `data` and intermediate calculations for each fish. See details
- rawdiff A frequency table of fish by differences for each pair of ages.
- absdiff A frequency table of fish by absolute differences for each pair of ages.
- AAD The average absolute deviation.
- APE The average percent error (using the **mean** age as the divisor).
- APE2 The average percent error (using the **median** age as the divisor).
- ASD The average standard deviation.
- ACV The average coefficient of variation (using the **mean** age as the divisor).
- ACV2 The average coefficient of variation (using the **median** age as the divisor).
- AD The average index of precision.
- R The number of readings for each individual fish.
- n Number of fish in `data`.
- validn Number of fish in `data` that have non-`NA` data for all R age estimates.

The `summary` returns the result if `what=` contains only one item, otherwise it returns nothing. See details for what is printed.

Testing

Tested all precision results against published results in Herbst and Marsden (2011) for the WhitefishLC data and the results for the AlewifeLH data set from **FSAdata** against results from the online resource at http://www.nefsc.noaa.gov/fbp/age-prec/.

IFAR Chapter

4-Age Comparisons.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Beamish, R.J. and D.A. Fournier. 1981. A method for comparing the precision of a set of age determinations. Canadian Journal of Fisheries and Aquatic Sciences 38:982-983. [Was (is?) available from http://www.pac.dfo-mpo.gc.ca/science/people-gens/beamish/PDF_files/compareagecjfas1981.pdf.]

Campana, S.E. 1982. Accuracy, precision and quality control in age determination, including a review of the use and abuse of age validation methods. Journal of Fish Biology 59:197-242. [Was (is?) available from http://www.denix.osd.mil/nr/crid/Coral_Reef_Iniative_Database/References_for_Reef_Assessment_files/Can

Campana, S.E., M.C. Annand, and J.I. McMillan. 1995. Graphical and statistical methods for determining the consistency of age determinations. Transactions of the American Fisheries Society 124:131-138. [Was (is?) available from http://www.bio.gc.ca/otoliths/documents/Campana%20et%20al%201995%20TAFS.

Chang, W.Y.B. 1982. A statistical method for evaluating the reproducibility of age determination. Canadian Journal of Fisheries and Aquatic Sciences 39:1208-1210. [Was (is?) available from http://www.nrcresearchpress.com/doi/abs/10.1139/f82-158.]

McBride, R.S. 2015. Diagnosis of paired age agreement: A simulation approach of accuracy and precision effects. ICES Journal of Marine Science, 72:2149-2167.

See Also

See ageBias for computation of the full age agreement table, along with tests and plots of age bias.

Examples

```
## Example with just two age estimates
ap1 <- agePrecision(~otolithC+scaleC,data=WhitefishLC)
summary(ap1)
summary(ap1,what="precision")
summary(ap1,what="difference")
summary(ap1,what="difference",percent=FALSE)
summary(ap1,what="absolute")
summary(ap1,what="absolute",percent=FALSE)
summary(ap1,what="absolute",trunc.diff=4)
summary(ap1,what=c("precision","difference"))
summary(ap1,what="detail")
```

```
barplot(ap1$rawdiff,ylab="Frequency",xlab="Otolith - Scale Age")
plot(AD~mean,data=ap1$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Mean Age",ylab="Absolute Deviation Age")
plot(SD~mean,data=ap1$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Mean Age",ylab="Standard deviation Age")
plot(SD~AD,data=ap1$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Absolute Deviation Age",ylab="Standard deviation Age")
plot(CV~PE,data=ap1$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Percent Error Age",ylab="Coefficient of Variation Age")

## Example with three age estimates
ap2 <- agePrecision(~otolithC+finrayC+scaleC,data=WhitefishLC)
summary(ap2,digits=3)
summary(ap2,what="precision")
summary(ap2,what="difference")
summary(ap2,what="absolute",percent=FALSE,trunc.diff=4)
summary(ap2,what="detail",digits=3)

plot(AD~mean,data=ap2$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Mean Age",ylab="Absolute Deviation Age")
plot(SD~mean,data=ap2$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Mean Age",ylab="Standard Deviation Age")
plot(SD~AD,data=ap2$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Absolute Deviation Age",ylab="Standard Deviation Age")
plot(CV~PE,data=ap2$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Percent Error Age",ylab="Coefficient of Variation Age")
plot(median~mean,data=ap2$detail,pch=19,col=col2rgbt("black",1/5),
     xlab="Mean Age",ylab="Median Age")
```

`alkAgeDist` *Proportions-at-age from an age-length key*

Description

Computes the proportions-at-age (with standard errors) in a larger sample based on an age-length-key created from a subsample of ages through a two-stage random sampling design. Follows the methods in Quinn and Deriso (1999).

Usage

```
alkAgeDist(key, lenA.n, len.n)
```

Arguments

`key`	A numeric matrix that contains the age-length key. See details.
`lenA.n`	A numeric vector of sample sizes for each length interval in the *aged sample*.
`len.n`	A numeric vector of sample sizes for each length interval in the *complete sample* (i.e., all fish regardless of whether they were aged or not).

Details

The age-length key in `key` must have length intervals as rows and ages as columns. The row names of `key` (i.e., `rownames(key)`) must contain the minimum values of each length interval (e.g., if an interval is 100-109 then the corresponding row name must be 100). The column names of `key` (i.e., `colnames(key)`) must contain the age values (e.g., the columns can NOT be named with "age.1", for example).

The length intervals in the rows of `key` must contain all of the length intervals present in the larger sample. Thus, the length of `len.n` must, at least, equal the number of rows in `key`. If this constraint is not met, then the function will stop with an error message.

The values in `lenA.n` are equal to what the row sums of `key` would have been before `key` was converted to a row proportions table. Thus, the length of `lenA.n` must also be equal to the number of rows in `key`. If this constraint is not met, then the function will stop with an error message.

Value

A data.frame with as many rows as ages (columns) present in `key` and the following three variables:

- age The ages.
- prop The proportion of fish at each age.
- se The SE for the proportion of fish at each age.

Testing

The results from this function perfectly match the results in Table 8.4 (left) of Quinn and Deriso (1999) using `SnapperHG2` from **FSAdata**. The results also perfectly match the results from using `alkprop` in **fishmethods**.

IFAR Chapter

5-Age-Length Key.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Lai, H.-L. 1987. Optimum allocation for estimating age composition using age-length key. Fishery Bulletin, 85:179-185.

Lai, H.-L. 1993. Optimum sampling design for using the age-length key to estimate age composition of a fish population. Fishery Bulletin, 92:382-388.

Quinn, T. J. and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York, New York. 542 pages.

See Also

See alkIndivAge and related functions for a completely different methodology. See alkprop from **fishmethods** for the exact same methodology but with a different format for the inputs.

Examples

```
## Example -- Even breaks for length categories
WR1 <- WR79
# add length intervals (width=5)
WR1$LCat <- lencat(WR1$len,w=5)
# get number of fish in each length interval in the entire sample
len.n <- xtabs(~LCat,data=WR1)
# isolate aged sample and get number in each length interval
WR1.age <- subset(WR1, !is.na(age))
lenA.n <- xtabs(~LCat,data=WR1.age)
# create age-length key
raw <- xtabs(~LCat+age,data=WR1.age)
( WR1.key <- prop.table(raw, margin=1) )

# use age-length key to estimate age distribution of all fish
alkAgeDist(WR1.key,lenA.n,len.n)
```

`alkIndivAge` *Use an age-length key to assign age to individuals in the unaged sample.*

Description

Use either the semi- or completely-random methods from Isermann and Knight (2005) to assign ages to individual fish in the unaged sample according to the information in an age-length key supplied by the user.

Usage

```
alkIndivAge(key, formula, data, type = c("SR", "CR"), breaks = NULL,
  seed = NULL)
```

Arguments

`key`	A numeric matrix that contains the age-length key. The format of this matrix is important. See details.
`formula`	A formula of the form `age~length` where `age` generically represents the variable that will contain the estimated ages once the key is applied (i.e., should currently contain no values) and `length` generically represents the variable that contains the known length measurements. If only `~length` is used, then a new variable called "age" will be created in the resulting data frame.

data	A data.frame that minimally contains the length measurements and possibly contains a variable that will receive the age assignments as given in formula.
type	A string that indicates whether to use the semi-random (type="SR", default) or completely-random (type="CR") methods for assigning ages to individual fish. See the IFAR chapter for more details.
breaks	A numeric vector of lower values that define the length intervals. See details.
seed	A single numeric that is given to set.seed to set the random seed. This allows repeatability of results.

Details

The age-length key in key must have length intervals as rows and ages as columns. The row names of key (i.e., rownames(key)) must contain the minimum values of each length interval (e.g., if an interval is 100-109, then the corresponding row name must be 100). The column names of key (i.e., colnames(key)) must contain the age values (e.g., the columns can NOT be named with "age.1", for example).

The length intervals in the rows of key must contain all of the length intervals present in the unaged sample to which the age-length key is to be applied (i.e., sent in the length portion of the formula). If this constraint is not met, then the function will stop with an error message.

If breaks=NULL, then the length intervals for the unaged sample will be determined with a starting interval at the minimum value of the row names in key and a width of the length intervals as determined by the minimum difference in adjacent row names of key. If length intervals of differing widths were used when constructing key, then those breaks should be supplied to breaks=. Use of breaks= may be useful when "uneven" length interval widths were used because the lengths in the unaged sample are not fully represented in the aged sample. See the examples.

Assigned ages will be stored in the column identified on the left-hand-side of formula (if the formula has both a left- and right-hand-side). If this variable is missing in formula, then the new column will be labeled with age.

Value

The original data.frame in data with assigned ages added to the column supplied in formula or in an additional column labeled as age. See details.

Testing

The type="SR" method worked perfectly on a small example. The type="SR" method provides results that reasonably approximate the results from alkAgeDist and alkMeanVar, which suggests that the age assessments are reasonable.

IFAR Chapter

5-Age-Length Key.

Author(s)

Derek H. Ogle, <derek@derekogle.com>. This is largely an R version of the SAS code provided by Isermann and Knight (2005).

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Isermann, D.A. and C.T. Knight. 2005. A computer program for age-length keys incorporating age assignment to individual fish. North American Journal of Fisheries Management, 25:1153-1160. [Was (is?) from http://www.tandfonline.com/doi/abs/10.1577/M04-130.1.]

See Also

See alkAgeDist and alkMeanVar for alternative methods to derived age distributions and mean (and SD) values for each age. See alkPlot for methods to visualize age-length keys.

Examples

```
## First Example -- Even breaks for length categories
WR1 <- WR79
# add length categories (width=5)
WR1$LCat <- lencat(WR1$len,w=5)
# isolate aged and unaged samples
WR1.age <- subset(WR1, !is.na(age))
WR1.len <- subset(WR1, is.na(age))
# note no ages in unaged sample
head(WR1.len)
# create age-length key
raw <- xtabs(~LCat+age,data=WR1.age)
( WR1.key <- prop.table(raw, margin=1) )
# apply the age-length key
WR1.len <- alkIndivAge(WR1.key,age~len,data=WR1.len)
# now there are ages
head(WR1.len)
# combine orig age & new ages
WR1.comb <- rbind(WR1.age, WR1.len)
# mean length-at-age
Summarize(len~age,data=WR1.comb,digits=2)
# age frequency distribution
( af <- xtabs(~age,data=WR1.comb) )
# proportional age distribution
( ap <- prop.table(af) )

## Second Example -- length sample does not have an age variable
WR2 <- WR79
# isolate age and unaged samples
WR2.age <- subset(WR2, !is.na(age))
WR2.len <- subset(WR2, is.na(age))
# remove age variable (for demo only)
WR2.len <- WR2.len[,-3]
# add length categories to aged sample
WR2.age$LCat <- lencat(WR2.age$len,w=5)
# create age-length key
raw <- xtabs(~LCat+age,data=WR2.age)
( WR2.key <- prop.table(raw, margin=1) )
# apply the age-length key
```

```
WR2.len <- alkIndivAge(WR2.key,~len,data=WR2.len)
# add length cat to length sample
WR2.len$LCat <- lencat(WR2.len$len,w=5)
head(WR2.len)
# combine orig age & new ages
WR2.comb <- rbind(WR2.age, WR2.len)
Summarize(len~age,data=WR2.comb,digits=2)

## Third Example -- Uneven breaks for length categories
WR3 <- WR79
# set up uneven breaks
brks <- c(seq(35,100,5),110,130)
WR3$LCat <- lencat(WR3$len,breaks=brks)
WR3.age <- subset(WR3, !is.na(age))
WR3.len <- subset(WR3, is.na(age))
head(WR3.len)
raw <- xtabs(~LCat+age,data=WR3.age)
( WR3.key <- prop.table(raw, margin=1) )
WR3.len <- alkIndivAge(WR3.key,age~len,data=WR3.len,breaks=brks)
head(WR3.len)
WR3.comb <- rbind(WR3.age, WR3.len)
Summarize(len~age,data=WR3.comb,digits=2)
```

`alkMeanVar` *Mean Values-at-age from an age-length key*

Description

Computes the mean value-at-age in a larger sample based on an age-length-key created from a subsample of ages through a two-stage random sampling design. The mean values could be mean length-, weight-, or fecundity-at-age, for example. The methods of Bettoli and Miranda (2001) or Quinn and Deriso (1999) are used. A standard deviation is computed for the Bettoli and Miranda (2001) method and standard error for the Quinn and Deriso (1999) method. See the testing section notes.

Usage

```
alkMeanVar(key, formula, data, len.n, method = c("BettoliMiranda",
  "QuinnDeriso"))
```

Arguments

`key`	A numeric matrix that contains the age-length key. See details.
`formula`	A formula of the form `var~lencat+age` where `var` generically represents the variable to be summarized (e.g., length, weight, fecundity), `lencat` generically represents the variable that contains the length intervals, and `age` generically represents the variable that contains the assigned ages.

`data`	A data.frame that minimally contains the length intervals, assessed ages, and the variable to be summarized (i.e., this should be the aged sample) as given in `formula`.
`len.n`	A vector of sample sizes for each length interval in the *complete sample* (i.e., all fish regardless of whether they were aged or not).
`method`	A string that indicates which method of calculation should be used. See details.

Details

The age-length key `key` must have length intervals as rows and ages as columns. The row names of `key` (i.e., `rownames(key)`) must contain the minimum values of each length interval (e.g., if an interval is 100-109, then the corresponding row name must be 100). The column names of `key` (i.e., `colnames(key)`) must contain the age values (e.g., the columns can NOT be named with "age.1", for example).

The length intervals in the rows of `key` must contain all of the length intervals present in the larger sample. Thus, the length of `len.n` must, at least, equal the number of rows in `key`. If this constraint is not met, then the function will stop with an error message.

Note that the function will stop with an error if the formula in `formula` does not meet the specific criteria outlined in the parameter list above.

Value

A data.frame with as many rows as ages (columns) present in `key` and the following three variables:

- age The ages.
- mean The mean value at each age.
- sd,se The SD if `method="BettoliMiranda"` or SE of the mean if `method="QuinnDeriso"` for the value at each age.

Testing

The results of these functions have not yet been rigorously tested. The Bettoli and Miranda (2001) results appear, at least, approximately correct when compared to the results from `alkIndivAge`. The Quinn and Deriso (1999) results appear at least approximately correct for the mean values, but do not appear to be correct for the SE values. Thus, a note is returned with the Quinn and Deriso (1999) results that the SE should not be trusted.

IFAR Chapter

5-Age-Length Key.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Bettoli, P. W. and Miranda, L. E. 2001. A cautionary note about estimating mean length at age with subsampled data. North American Journal of Fisheries Management, 21:425-428.

Quinn, T. J. and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York, New York. 542 pages

See Also

See alkIndivAge and related functions for a completely different methodology. See alkAgeDist for a related method of determining the proportion of fish at each age. See the **ALKr** package.

Examples

```
## Example -- Even breaks for length categories
WR1 <- WR79
# add length intervals (width=5)
WR1$LCat <- lencat(WR1$len,w=5)
# get number of fish in each length interval in the entire sample
len.n <- xtabs(~LCat,data=WR1)
# isolate aged sample
WR1.age <- subset(WR1, !is.na(age))
# create age-length key
raw <- xtabs(~LCat+age,data=WR1.age)
( WR1.key <- prop.table(raw, margin=1) )

## use age-length key to estimate mean length-at-age of all fish
# Bettoli-Miranda method
alkMeanVar(WR1.key,len~LCat+age,WR1.age,len.n)

# Quinn-Deriso method
alkMeanVar(WR1.key,len~LCat+age,WR1.age,len.n,method="QuinnDeriso")
```

`alkPlot` *Plots to visualize age-length keys.*

Description

Various plots to visualize the proportion of fish of certain ages within length intervals in an age-length key.

Usage

```
alkPlot(key, type = c("barplot", "area", "lines", "splines", "bubble"),
  xlab = "Length", ylab = ifelse(type != "bubble", "Proportion",
  "Age"), xlim = NULL, ylim = NULL, showLegend = FALSE,
  lbl.cex = 1.25, leg.cex = 1, lwd = 2, span = 0.25,
```

```
pal = paletteChoices(), grid = TRUE, col = "gray80", buf = 0.45,
add = FALSE, ...)
```

Arguments

key	A numeric matrix that contains the age-length key.
type	A string that indicates the type of plot to construct. See details.
xlab, ylab	A string that contains the label for the x- or y-axis.
xlim, ylim	A numeric of length 2 that provide the limits for the x-axis or y-axis.
showLegend	A logical that indicates whether a legend should be displayed (not implemented for type="bubble"). See examples.
lbl.cex	A numeric character expansion value for labels inside the bars when type="barplot" or on the lines when type="lines" or type="splines". Only used if showLegend=FALSE.
leg.cex	A numeric character expansion value for labels on the legend when showLegend=TRUE.
lwd	A numeric that indicates the line width when type="lines" or type="splines".
span	A numeric that indicates the span value to use in loess when type="splines".
pal	A string that indicates the palette to generate colors for the bars, areas, lines, or spline lines. The name of a palette must be one of "rich", "cm", "default", "grey", "gray", "heat", "jet", "rainbow", "topo", or "terrain". See chooseColors.
grid	A logical that indicates whether a grid should be placed under the bubbles when type="bubble" or a character or appropriate vector that identifies a color for the grid. See examples.
col	A string that indicates the color of the bubbles when type="bubble".
buf	A single numeric that indicates the relative width of the bubbles when type="bubble". A value of 0.5 means that two full-width bubbles would touch each other either in the x- or y-direction (i.e., this would represent half of the minimum of the physical distance between values one-unit apart on the x- and y-axes). Set this to a value less than 0.5 so that the bubbles will not touch (the default is 0.45).
add	A logical that indicates whether the data should be added to an already existing plot. May be useful for visually comparing age-length keys. Only implemented when type="bubble".
...	Additional arguments to pass to plot or barplot.

Details

A variety of plots can be used to visualize the proportion of fish of certain ages within length intervals of an age-length key. The types of plots are described below and illustrated in the examples.

- A "stacked" bar chart where vertical bars over length intervals sum to 1 but are segmented by the proportion of each age in that length interval is constructed with type="barplot". The ages will be labeled in the bar segments unless showLegend=TRUE is used.
- A "stacked" area chart similar to the bar chart described above is constructed with type="area".
- A plot with (differently colored) lines that connect the proportions of ages within each length interval is constructed with type="lines".

- A plot with (differently colored) lines, as estimated by loess splines, that connect the proportions of ages within each length interval is constructed with `type="splines"`.
- A "bubble" plot where circles whose size is proportional to the proportion of fish of each age in each length interval is constructed with `type="bubble"`. The color of the bubbles can be controlled with `col=` and an underlying grid for ease of seeing the age and length interval for each bubble can be controlled with `grid=`. Bubbles from a second age-length key can be overlaid on an already constructed bubble plot by using `add=TRUE` in a second call to `alkPlot`.

Note that all plots are "vertically conditional" – i.e., each represents the proportional ages WITHIN each length interval.

Value

None, but a plot is constructed.

IFAR Chapter

5-Age-Length Key.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See `alkIndivAge` for using an age-length key to assign ages to individual fish.

Examples

```
## Make an example age-length key
WR.age <- subset(WR79, !is.na(age))
WR.age$LCat <- lencat(WR.age$len,w=5)
raw <- xtabs(~LCat+age,data=WR.age)
( WR.key <- prop.table(raw, margin=1) )

## Various visualizations of the age-length key
alkPlot(WR.key,"barplot")
alkPlot(WR.key,"barplot",pal="gray")
alkPlot(WR.key,"barplot",showLegend=TRUE)
alkPlot(WR.key,"area")
alkPlot(WR.key,"area",showLegend=TRUE)
alkPlot(WR.key,"area",pal="gray")
alkPlot(WR.key,"lines")
alkPlot(WR.key,"lines",pal="gray")
alkPlot(WR.key,"lines",showLegend=TRUE)
alkPlot(WR.key,"splines")
alkPlot(WR.key,"splines",span=0.2)
```

```
alkPlot(WR.key,"splines",pal="gray",showLegend=TRUE)
alkPlot(WR.key,"bubble")
alkPlot(WR.key,"bubble",grid=FALSE)
alkPlot(WR.key,"bubble",grid="blue")
alkPlot(WR.key,"bubble",grid=rgb(0,0,0,0.2),col=rgb(0,0,0,0.5))
```

---

binCI *Confidence intervals for binomial probability of success.*

Description

Uses one of three methods to compute a confidence interval for the probability of success (p) in a binomial distribution.

Usage

```
binCI(x, n, conf.level = 0.95, type = c("wilson", "exact",
  "asymptotic"), verbose = FALSE)
```

Arguments

x	A single or vector of numbers that contains the number of observed successes.
n	A single or vector of numbers that contains the sample size.
conf.level	A single number that indicates the level of confidence (default is 0.95).
type	A string that identifies the type of method to use for the calculations. See details.
verbose	A logical that indicates whether x, n, and x/n should be included in the returned matrix (=TRUE) or not (=FALSE; DEFAULT).

Details

This function will compute confidence interval for three possible methods chosen with the type argument.

type="wilson"	Wilson's (Journal of the American Statistical Association, 1927) confidence interval for a proportion.
type="exact"	Computes the Clopper/Pearson exact CI for a binomial success probability.
type="asymptotic"	This uses the normal distribution approximation.

Note that Agresti and Coull (2000) suggest that the Wilson interval is the preferred method and is, thus, the default type.

Value

A #x2 matrix that contains the lower and upper confidence interval bounds as columns and, if verbose=TRUE x, n, and x/n .

Note

This is primarily a wrapper function for `binom.exact`, `binom.wilson`, and `binom.approx` (documented in `binom.conf.int`) from the **epitools** package.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Agresti, A. and B.A. Coull. 1998. Approximate is better than "exact" for interval estimation of binomial proportions. American Statistician, 52:119-126.

See Also

See `binom.test`; `binconf` in **Hmisc**; `binom.exact`, `binom.wilson`, and `binom.approx` documented in `binom.conf.int` in **epitools**, and functions in **binom**.

Examples

```
## All types at once
binCI(7,20)

## Individual types
binCI(7,20,type="wilson")
binCI(7,20,type="exact")
binCI(7,20,type="asymptotic")
binCI(7,20,type="asymptotic",verbose=TRUE)

## Multiple types
binCI(7,20,type=c("exact","asymptotic"))
binCI(7,20,type=c("exact","asymptotic"),verbose=TRUE)

## Use with multiple inputs
binCI(c(7,10),c(20,30),type="wilson")
binCI(c(7,10),c(20,30),type="wilson",verbose=TRUE)
```

BluegillJL *Capture histories (2 samples) of Bluegill from Jewett Lake, MI.*

Description

Each line consists of the capture history over two samples of Bluegill (*Lepomis macrochirus*) in Jewett Lake (MI). This file contains the capture histories for only Bluegill larger than 6-in.

Format

A data frame with 277 observations on the following 2 variables.

first a numeric vector of indicator variables for the first sample (1=captured)

second a numeric vector of indicator variables for the second sample (1=captured)

Topic(s)

- Population Size
- Abundance
- Mark-Recapture
- Capture-Recapture
- Petersen
- Capture History

Source

From example 8.1 in Schneider, J.C. 1998. Lake fish population estimates by mark-and-recapture methods. Chapter 8 in Schneider, J.C. (ed.) 2000. Manual of fisheries survey methods II: with periodic updates. Michigan Department of Natural Resources, Fisheries Special Report 25, Ann Arbor. [Was (is?) from http://www.michigandnr.com/publications/pdfs/IFR/manual/SMII%20Chapter08.pdf.]

See Also

Used in `mrClosed` examples.

Examples

```
str(BluegillJL)
head(BluegillJL)
```

`bootstrap` *Case bootstrapping and associated S3 methods.*

Description

The `bootCase` function was added to **FSA** to maintain backward compatability (because `bootCase` was removed from **car**), mostly for users of the Introductory Fisheries Analyses with R book. `bootCase` is largerly a wrapper to `Boot` from **car** with `method="case"`. It is suggested that `Boot` from **car** be used instead. S3 methods are also provided to construct non-parametric bootstrap confidence intervals, predictions with non-parametric confidence intervals, hypothesis tests, and plots of the parameter estimates for `bootCase` objects.

Usage

```
bootCase(object, f. = stats::coef, B = R, R = 999)

## S3 method for class 'bootCase'
confint(object, parm = NULL, level = conf.level,
  conf.level = 0.95, plot = FALSE, err.col = "black", err.lwd = 2,
  rows = NULL, cols = NULL, ...)

## S3 method for class 'bootCase'
predict(object, FUN, conf.level = 0.95,
  digits = NULL, ...)

## S3 method for class 'bootCase'
htest(object, parm = NULL, bo = 0,
  alt = c("two.sided", "less", "greater"), plot = FALSE, ...)

## S3 method for class 'bootCase'
hist(x, same.ylim = TRUE, ymax = NULL,
  rows = round(sqrt(ncol(x))), cols = ceiling(sqrt(ncol(x))), ...)

## S3 method for class 'bootCase'
plot(x, ...)
```

Arguments

object, x	A regression object of type lm, glm or class nls for bootCase or a bootCase object for the S3 methods.
f.	A function that will be applied to the updated regression object to compute the statistics of interest. The default is coef, to return to regression coefficient estimates.
B, R	Number of bootstrap samples.
parm	A number or string that indicates which column of object contains the parameter estimates to use for the confidence interval or hypothesis test.
level	Same as conf.level.
conf.level	A level of confidence as a proportion.
plot	A logical that indicates whether a plot should be constructed. If confint then a histogram of the parm parameters from the bootstrap samples with error bars that illustrate the bootstrapped confidence intervals will be constructed. If codehtest then a histogram of the parm parameters with a vertical line illustrating the bo value will be constructed.
err.col	A single numeric or character that identifies the color for the error bars on the plot.
err.lwd	A single numeric that identifies the line width for the error bars on the plot.
rows	A single numeric that contains the number of rows to use on the graphic.
cols	A single numeric that contains the number of columns to use on the graphic.

...	Additional items to send to functions. See details.
FUN	The function to be applied for the prediction. See the examples.
digits	A single numeric that indicates the number of digits for the result.
bo	The null hypothesized parameter value.
alt	A string that indicates the "direction" of the alternative hypothesis. See details.
same.ylim	A logical that indicates whether the same limits for the y-axis should be used on each histogram. Defaults to TRUE. Ignored if ylmts is non-null.
ymax	A single value that sets the maximum y-axis limit for each histogram or a vector of length equal to the number of groups that sets the maximum y-axis limit for each histogram separately.
col	A named color for the histogram bars.

Details

confint finds the two quantiles that have the (1-conf.level)/2 proportion of bootstrapped parameter estimates below and above. This is an approximate 100conf.level% confidence interval.

predict applies a user-supplied function to each row of object and then finds the median and the two quantiles that have the proportion (1-conf.level)/2 of the bootstrapped predictions below and above. The median is returned as the predicted value and the quantiles are returned as an approximate 100conf.level% confidence interval for that prediction. Values for the independent variable in FUN must be a named argument sent in the ... argument (see examples). Note that if other arguments are needed in FUN besides values for the independent variable, then these are included in the ... argument AFTER the values for the independent variable.

In htest the "direction" of the alternative hypothesis is identified by a string in the alt= argument. The strings may be "less" for a "less than" alternative, "greater" for a "greater than" alternative, or "two.sided" for a "not equals" alternative (the DEFAULT). In the one-tailed alternatives the p-value is the proportion of bootstrapped parameter estimates in object$coefboot that are extreme of the null hypothesized parameter value in bo. In the two-tailed alternative the p-value is twice the smallest of the proportion of bootstrapped parameter estimates above or below the null hypothesized parameter value in bo.

Value

If object is a matrix, then confint returns a matrix with as many rows as columns (i.e., parameter estimates) in object and two columns of the quantiles that correspond to the approximate confidence interval. If object is a vector, then confint returns a vector with the two quantiles that correspond to the approximate confidence interval.

htest returns a two-column matrix with the first column containing the hypothesized value sent to this function and the second column containing the corresponding p-value.

hist constructs histograms of the bootstrapped parameter estimates.

plot constructs scatterplots of all pairs of bootstrapped parameter estimates.

predict returns a matrix with one row and three columns, with the first column holding the predicted value (i.e., the median prediction) and the last two columns holding the approximate confidence interval.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

S. Weisberg (2005). *Applied Linear Regression*, third edition. New York: Wiley, Chapters 4 and 11.

See Also

Boot in **car**.

Examples

```
fnx <- function(days,B1,B2,B3) {
  if (length(B1) > 1) {
    B2 <- B1[2]
    B3 <- B1[3]
    B1 <- B1[1]
  }
  B1/(1+exp(B2+B3*days))
}
nl1 <- nls(cells~fnx(days,B1,B2,B3),data=Ecoli,
           start=list(B1=6,B2=7.2,B3=-1.45))
nl1.bootc <- bootCase(nl1,coef,B=99)  # B=99 too few to be useful
confint(nl1.bootc,"B1")
confint(nl1.bootc,c(2,3))
confint(nl1.bootc,conf.level=0.90)
confint(nl1.bootc,plot=TRUE)
predict(nl1.bootc,fnx,days=1:3)
predict(nl1.bootc,fnx,days=3)
htest(nl1.bootc,1,bo=6,alt="less")
hist(nl1.bootc)
plot(nl1.bootc)
cor(nl1.bootc)
```

BrookTroutTH *Catch-at-age for Tobin Harbor, Isle Royale Brook Trout.*

Description

Catch-at-age in fyke nets from 1996-1998 for "Coaster" Brook Trout (*Salvelinus fontinalis*) in Tobin Harbor, Isle Royale, Lake Superior.

Format

A data frame with 7 observations on the following 2 variables.

age A numeric vector of assigned ages

catch A numeric vector of number of Brook Trout caught

Topic(s)

- Mortality
- Catch Curve
- Chapman-Robson

Source

Quinlan, H.R. 1999. Biological Characteristics of Coaster Brook Trout at Isle Royale National Park, Michigan, 1996-98. U.S. Fish and Wildlife Service Ashland Fishery Resources Office report. November 1999. [Was (is?) from http://www.fws.gov/midwest/ashland/brook/biochar/biolchar.html.]

See Also

Used in `catchCurve` and `chapmanRobson` examples.

Examples

```
str(BrookTroutTH)
head(BrookTroutTH)
plot(log(catch)~age,data=BrookTroutTH)
```

---

capHistConvert *Convert between capture history data.frame formats.*

Description

Use to convert between simple versions of several capture history data.frame formats – "individual", "frequency", "event", "MARK", and "RMark". The primary use is to convert to the "individual" format for use in `capHistSum`.

Usage

```
capHistConvert(df, cols2use = NULL, cols2ignore = NULL,
  in.type = c("frequency", "event", "individual", "MARK", "marked",
  "RMark"), out.type = c("individual", "event", "frequency", "MARK",
  "marked", "RMark"), id = NULL, event.ord = NULL, freq = NULL,
  var.lbls = NULL, var.lbls.pre = "event",
  include.id = ifelse(is.null(id), FALSE, TRUE))
```

Arguments

df	A data.frame that contains the capture histories and, perhaps, a unique fish identifier or frequency variable. See details.
cols2use	A string or numeric vector that indicates columns in df to use. Negative numeric values will not use those columns. Cannot use both cols2use and col2ignore.
cols2ignore	A string or numeric vector that indicates columns in df to ignore. Typical columns to ignore are those that are not either in id= or freq= or part of the capture history data. Cannot use both cols2use and col2ignore.
in.type	A single string that indicates the type of capture history format to convert **FROM**.
out.type	A single string that indicates the type of capture history format to convert **TO**.
id	A string or numeric that indicates the column in df that contains the unique identifier for an individual fish. This argument is only used if in.type="event", in.type="individual", or, possibly, in.type="RMark".
event.ord	A string that contains a vector of ordered levels to be used when in.type="event". The default is to order alphabetically which may not be desirable if, for example, the events are labeled as 'first', 'second', 'third', and 'fourth'. In this case, use event.ord=c("first","second","third","fourth").
freq	A string or numeric that indicates the column in df that contains the frequency of individual fish corresponding to a capture history. This argument is only used if in.type="MARK", in.type="frequency", or, possibly, in.type="RMark".
var.lbls	A string vector of labels for the columns that contain the returned individual or frequency capture histories. If var.lbls=NULL or the length is different then the number of events then default labels using var.lbls.pre will be used. This argument is only used if out.type="frequency" or out.type="individual".
var.lbls.pre	A single string used as a prefix for the labels of the columns that contain the returned individual or frequency capture histories. This prefix will be appended with a number corresponding to the sample event. This argument is only used if out.type="frequency" or out.type="individual" and will be ignored if a proper vector is given in var.lbls.
include.id	A logical that indicates whether a unique fish identifier variable/column should be included in the output data.frame. This argument is only used if out.type="individual" or out.type="RMark".

Details

capHistSum requires capture histories to be recorded in the "individual" format. In this format, the data frame contains (at least) as many columns as sample events and as many rows as individually tagged fish. Optionally, the data.frame may also contain a column with unique fish identifiers (e.g., tag numbers). Each cell in the capture history portion of the data.frame contains a '0' if the fish of that row was NOT seen in the event of that column and a '1' if the fish of that row WAS seen in the event of that column. For example, suppose that five fish were marked on four sampling events; fish '17' was captured on the first two events; fish '18' was captured on the first and third events; fish '19' was captured on only the third event; fish '20' was captured on only the fourth event; and fish '21' was captured on the first and second events. The "individual" capture history date.frame for these data looks like:

fish	event1	event2	event3	event4
17	1	1	0	0
18	1	0	1	0
19	0	0	1	0
20	0	0	0	1
21	1	1	0	0

The "frequency" format data.frame (this format is used in **Rcapture**) has unique capture histories in separate columns, as in the "individual" format, but also includes a column with the frequency of individuals that had the capture history of that row. It will not contain a fish identifier variable. The same data from above looks like:

event1	event2	event3	event4	freq
1	1	0	0	2
1	0	1	0	1
0	0	1	0	1
0	0	0	1	1

The "event" format data.frame has a column with the unique fish identifier and a column with the event in which the fish of that row was observed. The same data from above looks like:

fish	event
17	1
18	1
21	1
17	2
21	2
18	3
19	3
20	4

MARK (http://www.phidot.org/software/mark/index.html) is the "gold-standard" software for analyzing complex capture history information. In the "MARK" format the 0s and 1s of the capture histories are combined together as a string without any spaces. Thus, the "MARK" format has the capture history strings in one column with an additional column that contains the frequency of individuals that exhibited the capture history of that row. The final column ends with a semi-colon. The same data from above looks like:

ch	freq
0001	1;
0010	1;
1010	1;
1100	2;

The RMark and **marked** are packages used to replace some of the functionality of MARK or to interact with MARK. The "RMark" or "marked" format requires the capture histories as one string (must be a character string and called 'ch'), as in the "MARK" format, but without the semicolon. The data.frame may be augmented with an identifier for individual fish OR with a frequency variable. If augmented with a unique fish identification variable then the same data from above looks like:

fish	ch
17	1100
18	1010
19	0010
20	0001
21	1100

However, if augmented with a frequency variable then the same data from above looks like:

ch	freq
0001	1
0010	1
1010	1
1100	2

Each of the formats can be used to convert from (i.e., in `in.type=`) or to convert to (i.e., in `out.type=`) with the exception that only the individual fish identifier version can be converted to when `out.type="RMark"`.

Value

A data frame of the proper type given in `out.type` is returned. See details.

Warning

`capHistConvert` may give unwanted results if the data are `in.type="event"` but there are unused levels for the variable, as would result if the data.frame had been subsetted on the event variable. The unwanted results can be corrected by using `droplevels` before `capHistConvert`. See the last example for an example.

IFAR Chapter

9-Abundance from Capture-Recapture Data.

Note

The formats as used here are simple in the sense that one is only allowed to have the individual fish identifier or the frequency variable in addition to the capture history information. More complex analyses may use a number of covariates. For these more complex analyses, one should work directly with the **Rcapture**, RMark, or **marked** packages.

This function also assumes that all unmarked captured fish are marked and returned to the population (i.e., no losses at the time of marking are allowed).

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See capHistSum to summarize "individual" capture histories into a format usable in mrClosed and mrOpen. Also see **Rcapture**, RMark, or **marked** packages for handling more complex analyses.

Examples

```
## A small example of 'event' format
( ex1 <- data.frame(fish=c(17,18,21,17,21,18,19,20),yr=c(1987,1987,1987,1988,1988,1989,1989,1990)) )
# convert to 'individual' format
( ex1.E2I <- capHistConvert(ex1,id="fish",in.type="event") )
# convert to 'frequency' format
( ex1.E2F <- capHistConvert(ex1,id="fish",in.type="event",out.type="frequency") )
# convert to 'MARK' format
( ex1.E2M <- capHistConvert(ex1,id="fish",in.type="event",out.type="MARK") )
# convert to 'RMark' format
( ex1.E2R <- capHistConvert(ex1,id="fish",in.type="event",out.type="RMark") )

## convert converted 'individual' format ...
# to 'frequency' format (must ignore "id")
( ex1.I2F <- capHistConvert(ex1.E2I,id="fish",in.type="individual",out.type="frequency") )
# to 'MARK' format
( ex1.I2M <- capHistConvert(ex1.E2I,id="fish",in.type="individual",out.type="MARK") )
# to 'RMark' format
( ex1.I2R <- capHistConvert(ex1.E2I,id="fish",in.type="individual",out.type="RMark") )
# to 'event' format
( ex1.I2E <- capHistConvert(ex1.E2I,id="fish",in.type="individual",out.type="event") )

#' ## convert converted 'frequency' format ...
# to 'individual' format
( ex1.F2I <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency") )
( ex1.F2Ia <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency",include.id=TRUE) )
# to 'Mark' format
( ex1.F2M <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency",
                            out.type="MARK") )
# to 'RMark' format
( ex1.F2R <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency",
                            out.type="RMark") )
( ex1.F2Ra <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency",
                             out.type="RMark",include.id=TRUE) )
# to 'event' format
```

```
( ex1.F2E <- capHistConvert(ex1.E2F,freq="freq",in.type="frequency",
                            out.type="event") )

## convert converted 'MARK' format ...
# to 'individual' format
( ex1.M2I <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK") )
( ex1.M2Ia <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK",include.id=TRUE) )
# to 'frequency' format
( ex1.M2F <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK",out.type="frequency") )
# to 'RMark' format
( ex1.M2R <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK",out.type="RMark") )
( ex1.M2Ra <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK",out.type="RMark",include.id=TRUE) )
# to 'event' format
( ex1.M2E <- capHistConvert(ex1.E2M,freq="freq",in.type="MARK",out.type="event") )

## convert converted 'RMark' format ...
# to 'individual' format
( ex1.R2I <- capHistConvert(ex1.E2R,id="fish",in.type="RMark") )
# to 'frequency' format
( ex1.R2F <- capHistConvert(ex1.E2R,id="fish",in.type="RMark",out.type="frequency") )
# to 'MARK' format
( ex1.R2M <- capHistConvert(ex1.E2R,id="fish",in.type="RMark",out.type="MARK") )
# to 'event' format
( ex1.R2E <- capHistConvert(ex1.E2R,id="fish",in.type="RMark",out.type="event") )

## Remove semi-colon from MARK format to make a RMark 'frequency' format
ex1.E2R1 <- ex1.E2M
ex1.E2R1$freq <- as.numeric(sub(";","",ex1.E2R1$freq))
ex1.E2R1
# convert this to 'individual' format
( ex1.R2I1 <- capHistConvert(ex1.E2R1,freq="freq",in.type="RMark") )
( ex1.R2I1a <- capHistConvert(ex1.E2R1,freq="freq",in.type="RMark",include.id=TRUE) )
# convert this to 'frequency' format
( ex1.R2F1 <- capHistConvert(ex1.E2R1,freq="freq",in.type="RMark",out.type="frequency") )
# convert this to 'MARK' format
( ex1.R2M1 <- capHistConvert(ex1.E2R1,freq="freq",in.type="RMark",out.type="MARK") )
# convert this to 'event' format
( ex1.R2E1 <- capHistConvert(ex1.E2R1,freq="freq",in.type="RMark",out.type="event") )


#############################################################################
## A small example using character ids
( ex2 <- data.frame(fish=c("id17","id18","id21","id17","id21","id18","id19","id20"),
                    yr=c(1987,1987,1987,1988,1988,1989,1989,1990)) )
# convert to 'individual' format
( ex2.E2I <- capHistConvert(ex2,id="fish",in.type="event") )
# convert to 'frequency' format
( ex2.E2F <- capHistConvert(ex2,id="fish",in.type="event",out.type="frequency") )
# convert to 'MARK' format
( ex2.E2M <- capHistConvert(ex2,id="fish",in.type="event",out.type="MARK") )
# convert to 'RMark' format
( ex2.E2R <- capHistConvert(ex2,id="fish",in.type="event",out.type="RMark") )
```

```
## convert converted 'individual' format ...
# to 'frequency' format
( ex2.I2F <- capHistConvert(ex2.E2I,id="fish",in.type="individual",out.type="frequency") )
# to 'MARK' format
( ex2.I2M <- capHistConvert(ex2.E2I,id="fish",in.type="individual",out.type="MARK") )
# to 'RMark' format
( ex2.I2R <- capHistConvert(ex2.E2I,id="fish",in.type="individual",out.type="RMark") )
# to 'event' format
( ex2.I2E <- capHistConvert(ex2.E2I,id="fish",in.type="individual",out.type="event") )

## demo use of var.lbls
( ex2.E2Ia <- capHistConvert(ex2,id="fish",in.type="event",var.lbls.pre="Sample") )
( ex2.E2Ib <- capHistConvert(ex2,id="fish",in.type="event",
              var.lbls=c("first","second","third","fourth")) )

## demo use of event.ord
( ex2.I2Ea <- capHistConvert(ex2.E2Ib,id="fish",in.type="individual",out.type="event") )
( ex2.E2Ibad <- capHistConvert(ex2.I2Ea,id="fish",in.type="event") )
( ex2.E2Igood <- capHistConvert(ex2.I2Ea,id="fish",in.type="event",
                 event.ord=c("first","second","third","fourth")) )

## ONLY RUN IN INTERACTIVE MODE
if (interactive()) {

#########################################################################
## A larger example of 'frequency' format (data from Rcapture package)
data(bunting,package="Rcapture")
head(bunting)
# convert to 'individual' format
bun.F2I <- capHistConvert(bunting,in.type="frequency",freq="freq")
head(bun.F2I)
# convert to 'MARK' format
bun.F2M <- capHistConvert(bunting,id="id",in.type="frequency",freq="freq",out.type="MARK")
head(bun.F2M)
# convert converted 'individual' back to 'MARK' format
bun.I2M <- capHistConvert(bun.F2I,id="id",in.type="individual",out.type="MARK")
head(bun.I2M)
# convert converted 'individual' back to 'frequency' format
bun.I2F <- capHistConvert(bun.F2I,id="id",in.type="individual",
           out.type="frequency",var.lbls.pre="Sample")
head(bun.I2F)


#########################################################################
## A larger example of 'marked' or 'RMark' format, but with a covariate
##   and when the covariate is removed there is no frequency or individual
##   fish identifier.
data(dipper,package="marked")
head(dipper)
# isolate males and females
dipperF <- subset(dipper,sex=="Female")
dipperM <- subset(dipper,sex=="Male")
# convert females to 'individual' format
```

```
dipF.R2I <- capHistConvert(dipperF,cols2ignore="sex",in.type="RMark")
head(dipF.R2I)
# convert males to 'individual' format
dipM.R2I <- capHistConvert(dipperM,cols2ignore="sex",in.type="RMark")
head(dipM.R2I)
# add sex variable to each data.frame and then combine
dipF.R2I$sex <- "Female"
dipM.R2I$sex <- "Male"
dip.R2I <- rbind(dipF.R2I,dipM.R2I)
head(dip.R2I)
tail(dip.R2I)

} # end interactive


## An example of problem with unused levels
## Create a set of test data with several groups
( df <- data.frame(fish=c("id17","id18","id21","id17","id21","id18","id19","id20","id17"),
                   group=c("B1","B1","B1","B2","B2","B3","B4","C1","C1")) )
#  Let's assume the user wants to subset the data from the "B" group
( df1 <- subset(df,group %in% c("B1","B2","B3","B4")) )
#  Looks like capHistConvert() is still using the unused factor
#  level from group C
capHistConvert(df1,id="fish",in.type="event")
# use droplevels() to remove the unused groups and no problem
df1 <- droplevels(df1)
capHistConvert(df1,id="fish",in.type="event")
```

---

capHistSum *Summarize capture histories in individual fish format.*

Description

Use to summarize a capture history data file that is in the "individual" fish format (see capHistConvert for a discussion of data file format types). Summarized capture history results may be used in the Lincoln-Petersen, Schnabel, Schumacher-Eschmeyer, or Jolly-Seber methods for estimating population abundance (see mrClosed and mrOpen).

Usage

```
capHistSum(df, cols2use = NULL, cols2ignore = NULL)

is.CapHist(x)

## S3 method for class 'CapHist'
plot(x, what = c("u", "f"), pch = 19,
  cex.pch = 0.7, lwd = 1, ...)
```

Arguments

df	A data.frame that contains the capture histories (and, perhaps, other information) in "individual" fish format. See details.
cols2use	A string or numeric vector that indicates columns in df that contain the capture histories. Negative numeric values will not use those columns. Cannot use both cols2use and col2ignore. See details.
cols2ignore	A string or numeric vector that indicates columns in df that do not contain the capture histories and should be ignored. Cannot use both cols2use and col2ignore.
x	An object from capHistSum.
what	A string that indicates what type of diagnostic plot to construct with plot. See details.
pch	A numeric that indicates the plotting character for the diagnostic plot.
cex.pch	A numeric that indicates the character expansion value for the plotting characters in the diagnostic plot. The default is to be "slightly smaller" (i.e., cex.pch=0.7).
lwd	A numeric that indicates the line width in the diagnostic plot.
...	Optional arguments to send to plot.

Details

This function requires the capture history data file to be in the "individual" fish format. See capHistConvert for a description of this (and other) formats and for methods to convert from other formats to the "individual" fish format. In addition, this function requires only the capture history portion of the data file. Thus, if df contains columns with non-capture history information (e.g., fish ID, length, location, etc.) then use cols2use= to identify which columns contain only the capture history information. Columns to use can be identified by listing the column numbers (e.g., columns 2 through 7 could be included with cols2use=2:7). In many instances it may be easier to identify columns to *exclude* which can be done by preceding the column number by a negative sign (e.g., columns 1 through 3 are excluded with cols2use=-(1:3)).

The object returned from this function can be used directly in mrClosed and mrOpen. See examples of this functionality on the help pages for those functions.

The plot function can be used to construct the two diagnostic plots described by Baillargeon and Rivest (2007). The what="f" plot will plot the log of the number of fish seen i times divided by choose(t,i) against i. The what="u" plot will plot the log of the number of fish seen for the first time on event i against i. Baillargeon and Rivest (2007) provide a table that can be used to diagnosed types of heterogeneities in capture probabilities from these plots.

Value

If the capture history data file represents only two samples, then a list with the following two components is returned.

- caphist A vector summarizing the frequency of fish with each capture history.
- sum A data.frame that contains the number of marked fish from the first sample (M), the number of captured fish in the second sample (n), and the number of recaptured (i.e. previously marked) fish in the second sample (m).

If the capture history data file represents more than two samples, then a list with the following five components is returned

- `caphist` A vector summarizing the frequency of fish with each capture history.
- `sum` A data frame that contains the the number of captured fish in the ith sample (`n`), the number of recaptured (i.e. previously marked) fish in the ith sample (`m`), the number of marked fish returned to the population following the ith sample (`R`; this will equal `n` as the function currently does not handle mortalities); the number of marked fish in the population prior to the ith sample (`M`); the number of fish first seen in the ith sample (`u`); the number of fish last seen in the ith sample (`v`); and the number of fish seen i times (`f`).
- `methodB.top` A matrix that contains the top of the Method B table used for the Jolly-Seber method (i.e., a contingency table of capture sample (columns) and last seen sample (rows)).
- `methodB.bot` A data.frame that contains the bottom of the Method B table used for the Jolly-Seber method (i.e., the number of marked fish in the sample (`m`), the number of unmarked fish in the sample (`u`), the total number of fish in the sample (`n`), and the number of marked fish returned to the population following the sample (`R`).
- `m.array` A matrix that contains the the so-called "m-array". The first column contains the number of fish captured on the ith event. The columns labeled with "cX" prefix show the number of fish originally captured in the ith row that were captured in the Xth event. The last column shows the number of fish originally captured in the ith row that were never recaptured.

IFAR Chapter

9-Abundance from Capture-Recapture Data.

Note

This function assumes that all unmarked captured fish are marked and returned to the population (i.e., no losses at the time of marking are allowed).

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Baillargeon, S. and Rivest, L.-P. (2007). Rcapture: Loglinear models for capture-recapture in R. Journal of Statistical Software, 19(5):1-31.

See Also

See `descriptive` in **Rcapture** for `m.array` and some of the same values in `sum`. See `capHistConvert` for a descriptions of capture history data file formats and how to convert between them. See `mrClosed` and `mrOpen` for how to estimate abundance from the summarized capture history information.

Examples

```
# data.frame with IDs in the first column
head(PikeNYPartial1)

# Three ways to ignore first column of ID numbers
( ch1 <- capHistSum(PikeNYPartial1,cols2use=-1) )
( ch1 <- capHistSum(PikeNYPartial1,cols2ignore=1) )
( ch1 <- capHistSum(PikeNYPartial1,cols2ignore="id") )

# diagnostic plots
plot(ch1)
plot(ch1,what="f")
plot(ch1,what="u")

# An examle with only two sample events (for demonstration only)
( ch2 <- capHistSum(PikeNYPartial1,cols2use=-c(1,4:5)) )
( ch2 <- capHistSum(PikeNYPartial1,cols2use=2:3) )
( ch2 <- capHistSum(PikeNYPartial1,cols2ignore=c(1,4:5)) )
```

---

catchCurve *Mortality estimates from the descending limb of a catch curve.*

Description

Fits a linear model to the user-defined descending limb of a catch curve. Method functions extract estimates of the instantaneous (Z) and total annual (A) mortality rates with associated standard errors and confidence intervals. A plot method highlights the descending limb, shows the linear model on the descending limb, and, optionally, prints the estimated Z and A.

Usage

```
catchCurve(x, ...)

## Default S3 method:
catchCurve(x, catch, ages2use = age,
  weighted = FALSE, negWeightReplace = 0, ...)

## S3 method for class 'formula'
catchCurve(x, data, ages2use = age, weighted = FALSE,
  negWeightReplace = 0, ...)

## S3 method for class 'catchCurve'
summary(object, parm = c("both", "all", "Z", "A",
  "lm"), ...)

## S3 method for class 'catchCurve'
coef(object, parm = c("all", "both", "Z", "A",
```

```
  "lm"), ...)

## S3 method for class 'catchCurve'
anova(object, ...)

## S3 method for class 'catchCurve'
confint(object, parm = c("all", "both", "Z", "A",
  "lm"), level = conf.level, conf.level = 0.95, ...)

## S3 method for class 'catchCurve'
rSquared(object, digits = getOption("digits"),
  percent = FALSE, ...)

## S3 method for class 'catchCurve'
plot(x, pos.est = "topright", cex.est = 0.95,
  ylab = "log(Catch)", xlab = "Age", col.pt = "gray30",
  col.mdl = "black", lwd = 2, lty = 1, ...)
```

Arguments

x	A numerical vector of assigned ages in the catch curve or a formula of the form catch~age when used in catchCurve. An object saved from catchCurve (i.e., of class catchCurve) when used in the methods.
...	Additional arguments for methods.
catch	A numerical vector of catches or CPUEs for the ages in the catch curve. Not used if x is a formula.
ages2use	A numerical vector of ages that define the descending limb of the catch curve.
weighted	A logical that indicates whether a weighted regression should be used. See details.
negWeightReplace	A single non-negative numeric that will replace negative weights (defaults to 0). Only used when weighted=TRUE. See details.
data	A data.frame from which the variables in the x formula can be found. Not used if x is not a formula.
object	An object saved from the catchCurve call (i.e., of class catchCurve).
parm	A numeric or string (of parameter names) vector that specifies which parameters are to be given confidence intervals. If parm="lm" then confidence intervals for the underlying linear model are returned.
level	Same as conf.level. Used for compatibility with the generic confint function.
conf.level	A number representing the level of confidence to use for constructing confidence intervals.
digits	The number of digits to round the rSquared result to.
percent	A logical that indicates if the rSquared result should be returned as a percentage (=TRUE) or as a proportion (=FALSE; default).

pos.est	A string to identify where to place the estimated mortality rates on the plot. Can be set to one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center" for positioning the estimated mortality rates on the plot. Typically "bottomleft" (DEFAULT) and "topright" will be "out-of-the-way" placements. Set pos.est to NULL to remove the estimated mortality rates from the plot.
cex.est	A single numeric character expansion value for the estimated mortality rates on the plot.
ylab	A label for the y-axis ("log(Catch)" is the default).
xlab	A label for the x-axis ("Age" is the default).
col.pt	A string that indicates the color of the plotted points.
col.mdl	A string that indicates the color of the fitted line.
lwd	A numeric that indicates the line width of the fitted line.
lty	A numeric that indicates the type of line used for the fitted line.

Details

The default is to use all ages in the age vector. This is appropriate only when the age and catch vectors contain only the ages and catches on the descending limb of the catch curve. Use ages2use to isolate only the catch and ages on the descending limb.

If weighted=TRUE then a weighted regression is used where the weights are the log(number) at each age predicted from the unweighted regression of log(number) on age (as proposed by Maceina and Bettoli (1998)). If a negative weight is computed it will be changed to the value in negWeightReplace and a warning will be issued.

Value

A list that contains the following items:

- age The original vector of assigned ages.
- catch The original vector of observed catches or CPUEs.
- age.e A vector of assigned ages for which the catch curve was fit.
- log.catch.e A vector of log catches or CPUEs for which the catch curve was fit.
- W A vector of weights used in the catch curve fit. Will be NULL unless weighted=TRUE.
- lm An lm object from the fit to the ages and log catches or CPUEs on the descending limb (i.e., in age.e and log.catch.e).

Testing

Tested the results of catch curve, both unweighted and weighted, against the results in Miranda and Bettoli (2007). Results for Z and the SE of Z matched perfectly. Tested the unweighted results against the results from agesurv in **fishmethods** using the rockbass data.frame in **fishmethods**. Results for Z and the SE of Z matched perfectly.

IFAR Chapter

11-Mortality.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Maceina, M.J., and P.W. Bettoli. 1998. Variation in Largemouth Bass recruitment in four mainstream impoundments on the Tennessee River. North American Journal of Fisheries Management 18:998-1003.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

See Also

See agesurv in **fishmethods** for similar functionality. See chapmanRobson and agesurvcl in **fishmethods** for alternative methods to estimate mortality rates. See metaM for empirical methods to estimate natural mortality.

Examples

```
plot(catch~age,data=BrookTroutTH,pch=19)

## demonstration of formula notation
cc1 <- catchCurve(catch~age,data=BrookTroutTH,ages2use=2:6)
summary(cc1)
cbind(Est=coef(cc1),confint(cc1))
rSquared(cc1)
plot(cc1)
summary(cc1,parm="Z")
cbind(Est=coef(cc1,parm="Z"),confint(cc1,parm="Z"))

## demonstration of excluding ages2use
cc2 <- catchCurve(catch~age,data=BrookTroutTH,ages2use=-c(0,1))
summary(cc2)
plot(cc2)

## demonstration of using weights
cc3 <- catchCurve(catch~age,data=BrookTroutTH,ages2use=2:6,weighted=TRUE)
summary(cc3)
plot(cc3)

## demonstration of returning the linear model results
summary(cc3,parm="lm")
cbind(Est=coef(cc3,parm="lm"),confint(cc3,parm="lm"))
```

```
## demonstration of ability to work with missing age classes
df <- data.frame(age=c(  2, 3, 4, 5, 7, 9,12),
                 ct= c(100,92,83,71,56,35, 1))
cc4 <- catchCurve(ct~age,data=df,ages2use=4:12)
summary(cc4)
plot(cc4)

## demonstration of ability to work with missing age classes
## evein if catches are recorded as NAs
df <- data.frame(age=c(  2, 3, 4, 5, 6, 7, 8, 9,10,11,12),
                 ct= c(100,92,83,71,NA,56,NA,35,NA,NA, 1))
cc5 <- catchCurve(ct~age,data=df,ages2use=4:12)
summary(cc5)
plot(cc5)
```

---

chapmanRobson *Computes Chapman-Robson estimates of S and Z.*

Description

Computes the Chapman-Robson estimates of annual survival rate (S) and instantaneous mortality rate (Z) from catch-at-age data on the descending limb of a catch-curve. Method functions extract estimates with associated standard errors and confidence intervals. A plot method highlights the descending-limb, shows the linear model on the descending limb, and, optionally, prints the estimated Z and A.

Usage

```
chapmanRobson(x, ...)

## Default S3 method:
chapmanRobson(x, catch, ages2use = age,
  zmethod = c("Smithetal", "Hoenigetal", "original"), ...)

## S3 method for class 'formula'
chapmanRobson(x, data, ages2use = age,
  zmethod = c("Smithetal", "Hoenigetal", "original"), ...)

## S3 method for class 'chapmanRobson'
summary(object, parm = c("all", "both", "Z",
  "S"), verbose = FALSE, ...)

## S3 method for class 'chapmanRobson'
coef(object, parm = c("all", "both", "Z", "S"),
  ...)

## S3 method for class 'chapmanRobson'
```

```
confint(object, parm = c("all", "both", "S",
  "Z"), level = conf.level, conf.level = 0.95, ...)

## S3 method for class 'chapmanRobson'
plot(x, pos.est = "topright", cex.est = 0.95,
  ylab = "Catch", xlab = "Age", col.pt = "gray30",
  axis.age = c("both", "age", "recoded age"), ...)
```

Arguments

x	A numerical vector of the assigned ages in the catch curve or a formula of the form catch~age when used in chapmanRobson. An object saved from chapmanRobson (i.e., of class chapmanRobson) when used in the methods.
...	Additional arguments for methods.
catch	A numerical vector of the catches or CPUEs for the ages in the catch curve. Not used if x is a formula.
ages2use	A numerical vector of the ages that define the descending limb of the catch curve.
zmethod	A string that indicates the method to use for estimating Z. See details.
data	A data frame from which the variables in the x formula can be found. Not used if x is not a formula.
object	An object saved from the chapmanRobson call (i.e., of class chapmanRobson).
parm	A numeric or string (of parameter names) vector that specifies which parameters are to be given confidence intervals If missing, all parameters are considered.
verbose	A logical that indicates whether the method should return just the estimate (FALSE; default) or a more verbose statement.
level	Same as conf.level. Used for compatibility with the generic confint function.
conf.level	A number representing the level of confidence to use for constructing confidence intervals.
pos.est	A string to identify where to place the estimated mortality rates on the plot. Can be set to one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center" for positioning the estimated mortality rates on the plot. Typically "bottomleft" (DEFAULT) and "topright" will be "out-of-the-way" placements. Set pos.est to NULL to remove the estimated mortality rates from the plot.
cex.est	A single numeric character expansion value for the estimated mortality rates on the plot.
ylab	A label for the y-axis ("Catch" is the default).
xlab	A label for the x-axis ("Age" is the default).
col.pt	A string that indicates the color of the plotted points.
axis.age	A string that indicates the type of x-axis to display. The age will display only the original ages, recoded age will display only the recoded ages, and both (DEFAULT) displays the original ages on the main axis and the recoded ages on the secondary axis.

Details

The default is to use all ages in the age vector. This is only appropriate if the age and catch vectors contain only the ages and catches on the descending limb of the catch curve. Use `ages2use` to isolate only the catch and ages on the descending limb.

The Chapman-Robson method provides an estimate of the annual survival rate, with the annual mortality rate (A) determined by 1-S. The instantaneous mortality rate is often computed as -log(S). However, Hoenig et al. (1983) showed that this produced a biased (over)estimate of Z and provided a correction. The correction is applied by setting `zmethod="Hoenigetal"`. Smith et al. (2012) showed that the Hoenig et al. method should be corrected for a variance inflation factor. This correction is applied by setting `zmethod="Smithetal"` (which is the default behavior). Choose `zmethod="original"` to use the original estimates for Z and it's SE as provided by Chapman and Robson.

Value

A list with the following items:

- age the original vector of assigned ages.
- catch the original vector of observed catches or CPUEs.
- age.e a vector of assigned ages used to estimate mortalities.
- catch.e a vector of catches or CPUEs used to estimate mortalities.
- age.r a vector of recoded ages used to estimate mortalities. See references.
- n a numeric holding the intermediate calculation of n. See references.
- T a numeric holding the intermediate calculation of T. See references.
- est A 2x2 matrix that contains the estimates and standard errors for S and Z.

Testing

Tested the results of chapmanRobson against the results in Miranda and Bettoli (2007). The point estimates of S matched perfectly but the SE of S did not because Miranda and Bettoli used a rounded estimate of S in the calculation of the SE of S but chapmanRobson does not.

Tested the results against the results from `agesurv` in **fishmethods** using the `rockbass` data.frame in **fishmethods**. Results for Z and the SE of Z matched perfectly for non-bias-corrected results. The estimate of Z, but not the SE of Z, matched for the bias-corrected (following Smith et al. (2012)) results. **FSA** uses equation 2 from Smith et al. (2012) whereas **fishmethods** appears to use equation 5 from the same source to estimate the SE of Z.

IFAR Chapter

11-Mortality.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Chapman, D.G. and D.S. Robson. 1960. The analysis of a catch curve. Biometrics. 16:354-368.

Hoenig, J.M. and W.D. Lawing, and N.A. Hoenig. 1983. Using mean age, mean length and median length data to estimate the total mortality rate. International Council for the Exploration of the Sea, CM 1983/D:23, Copenhagen.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Robson, D.S. and D.G. Chapman. 1961. Catch curves and mortality rates. Transactions of the American Fisheries Society. 90:181-189.

Smith, M.W., A.Y. Then, C. Wor, G. Ralph, K.H. Pollock, and J.M. Hoenig. 2012. Recommendations for catch-curve analysis. North American Journal of Fisheries Management. 32:956-967.

See Also

See `agesurv` in **fishmethods** for similar functionality. See `catchCurve` and `agesurvcl` in **fishmethods** for alternative methods. See `metaM` for empirical methods to estimate natural mortality.

Examples

```
plot(catch~age,data=BrookTroutTH,pch=19)

## demonstration of formula notation
cr1 <- chapmanRobson(catch~age,data=BrookTroutTH,ages2use=2:6)
summary(cr1)
summary(cr1,verbose=TRUE)
cbind(Est=coef(cr1),confint(cr1))
plot(cr1)
plot(cr1,axis.age="age")
plot(cr1,axis.age="recoded age")
summary(cr1,parm="Z")
cbind(Est=coef(cr1,parm="Z"),confint(cr1,parm="Z"))

## demonstration of excluding ages2use
cr2 <- chapmanRobson(catch~age,data=BrookTroutTH,ages2use=-c(0,1))
summary(cr2)
plot(cr2)

## demonstration of ability to work with missing age classes
age <- c(  2, 3, 4, 5, 7, 9,12)
ct  <- c(100,92,83,71,56,35, 1)
cr3 <- chapmanRobson(age,ct,4:12)
summary(cr3)
plot(cr3)
```

ChinookArg *Lengths and weights for Chinook Salmon from three locations in Argentina.*

Description

Lengths and weights for Chinook Salmon from three locations in Argentina.

Format

A data frame with 112 observations on the following 3 variables:

tl Total length (cm)

w Weight (kg)

loc Capture location (`Argentina`, `Petrohue`, `Puyehue`)

Topic(s)

- Weight-Length

Source

From Figure 4 in Soto, D., I. Arismendi, C. Di Prinzio, and F. Jara. 2007. Establishment of Chinook salmon (*Oncorhynchus tshawytscha*) in Pacific basins of southern South America and its potential ecosystem implications. Revista Chilena d Historia Natural, 80:81-98. [Was (is?) from http://www.scielo.cl/pdf/rchnat/v80n1/art07.pdf.]

See Also

Used in `lwCompPreds` examples.

Examples

```
str(ChinookArg)
head(ChinookArg)
op <- par(mfrow=c(2,2),pch=19,mar=c(3,3,0.5,0.5),mgp=c(1.9,0.5,0),tcl=-0.2)
plot(w~tl,data=ChinookArg,subset=loc=="Argentina")
plot(w~tl,data=ChinookArg,subset=loc=="Petrohue")
plot(w~tl,data=ChinookArg,subset=loc=="Puyehue")
par(op)
```

chooseColors *Create a list of colors from among a variety of color palettes.*

Description

Create a list of colors from among a variety of color palettes.

Usage

```
chooseColors(pal = paletteChoices(), num, rev = FALSE, ...)

paletteChoices()
```

Arguments

pal	A character that is the name of a palette. Must be one of "rich", "cm", "default", "grey", "gray", "heat", "jet", "rainbow", "topo", or "terrain", which are given in paletteChoices.
num	The number of colors to be returned.
rev	A logical that indicates if the default order of colors should be reversed (=TRUE) or not (=FALSE).
...	Other arguments to the various palette functions.

Value

A vector of colors of length num.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See rich.colors in **gplots**, cm.colors, heat.colors, topo.colors, terrain.colors, rainbow, colorRampPalette, and colors.

Examples

```
n <- 20
# Color Wheels
pie(rep(1,n), col=chooseColors("rich",n))
pie(rep(1,n), col=chooseColors("rainbow",n))
pie(rep(1,n), col=chooseColors("topo",n))
pie(rep(1,n), col=chooseColors("gray",n))
pie(rep(1,n), col=chooseColors("jet",n))
# colors reversed order
pie(rep(1,n), col=chooseColors("jet",n,rev=TRUE))
```

CodNorwegian *Stock and recruitment data for Norwegian cod, 1937-1960.*

Description

Norwegian cod (*Gadus morhua*) stock and recruitment by year, 1937-1960.

Format

A data frame of 24 observations on the following 3 variables:

year Year of data

recruits Recruits – year-class strength index

stock Spawning stock index

Topic(s)

- Stock-Recruit
- Recruitment

Source

From Garrod, D.J. 1967. Population dynamics of the Arcto-Norwegian cod. Journal of the Fisheries Research Board of Canada, 24:145-190.

See Also

Used in srStarts, srFuns, and nlsTracePlot examples.

Examples

```
str(CodNorwegian)
head(CodNorwegian)
op <- par(mfrow=c(1,2),pch=19,mar=c(3,3,0.5,0.5),mgp=c(1.9,0.5,0),tcl=-0.2)
plot(recruits~year,data=CodNorwegian,type="l")
plot(recruits~stock,data=CodNorwegian)
par(op)
```

col2rgbt *Converts an R color to RGB (red/green/blue) including a transparency (alpha channel).*

Description

Converts an R color to RGB (red/green/blue) including a transparency (alpha channel). Similar to col2rgb except that a transparency (alpha channel) can be included.

Usage

```
col2rgbt(col, transp = 1)
```

Arguments

col	A vector of any of the three kinds of R color specifications (i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
transp	A numeric vector that indicates the transparency level for the color. The transparency values must be greater than 0. Transparency values greater than 1 are interpreted as the number of points plotted on top of each other before the transparency is lost and is, thus, transformed to the inverse of the transparency value provided.

Value

A vector of hexadecimal strings of the form "#rrggbbaa" as would be returned by rgb.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See col2rgb for similar functionality.

Examples

```
col2rgbt("black")
col2rgbt("black",1/4)
clrs <- c("black","blue","red","green")
col2rgbt(clrs)
col2rgbt(clrs,1/4)
trans <- (1:4)/5
col2rgbt(clrs,trans)
```

`compIntercepts` *Tests for significant differences among all pairs of intercepts in a dummy variable regression.*

Description

Tests for significant differences among all pairs of intercepts in a dummy variable regression where the dummy variables all stem from one factor.

Usage

```
compIntercepts(mdl, common.cov = mean(x), conf.level = 0.95,
  digits = getOption("digits"), ...)

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

Arguments

`mdl`	A `lm` object.
`common.cov`	A value to be used as the common value of the covariate in the adjustment process. See details.
`conf.level`	A single number that represents the level of confidence to use for constructing confidence intervals.
`digits`	A numeric that controls the number of digits to print.
`...`	Other arguments to be passed to the `TukeyHSD` or `print` functions.
`x`	A `compIntercepts` object (i.e., returned from `compIntercepts`).

Details

In a dummy variable regression without the interaction(s) between the covariate (x) and the dummy variable(s) (i.e., parallel lines) the coefficient for the dummy variables tests for a difference in intercepts between the level of the dummy variable and the reference level. Thus, all dummy variables from a particular linear model fit only compare intercepts with the reference level. Other intercept comparisons may be found by changing the reference level, which requires refitting the model.

Alternatively, Tukey's HSD method of multiple comparisons may be used, but this requires adjusting the original observations as if the original observations were all collected at the exact same value of the covariate (x). Because of this required adjustment, the `TukeyHSD` function is inappropriate for testing for difference in intercepts in a dummy variable regression.

This function provides a statistical comparison of all pairs of intercepts by first adjusting the observed data to a common value of the covariate (`common.cov`), computing a one-way ANOVA to determine if the mean adjusted values differ by level of the group factor in the original dummy variable regression, and then submitting the one-way ANOVA results to the `TukeyHSD` function to determine for which levels the mean adjusted values differ. The levels for which the mean adjusted values differ are also the levels for which the intercepts differ.

The default is to compute the adjusted values at the mean value of the covariate (i.e., `common.cov=mean(x)`. However, if interest is in the intercepts (i.e., at X=0) then `common.cov=0` should be used instead.

Value

A list with the following four components:

`comparison`	The comparison results as returned from `TukeyHSD`.
`common.cov`	The value of the common covariate sent in `common.cov`.
`adjvals`	A vector of values of the response variable adjusted to the `common.cov` value of the covariate. This vector can b
`means`	A vector of mean adjusted values at the value of the common covariate.
`digits`	The value sent in `digits`.
`rnm`	The name of the response (LHS) variable.
`cnm`	The name of the covariate variable.

The `print` function prints the comparison and adjusted means in a nice format.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

`TukeyHSD` and `compSlopes` from **FSA**.

Examples

```
## Reduce Mirex data to years where slopes don't differ to illustrate this
##   function ... see compSlopes() for analysis of full data set.
Mirex <- Mirex[Mirex$year!="1996" & Mirex$year!="1999",]
Mirex$year <- factor(Mirex$year)
## Fit DVR, see that slopes don't differ,
##   compare intercepts, visualize results
lm1 <- lm(mirex~weight*year,data=Mirex)
anova(lm1)
compIntercepts(lm1)
## Fit model without interaction to avoid warning, but
##   note that the compIntercepts() results are the same
lm2 <- lm(mirex~weight+year,data=Mirex)
compIntercepts(lm2)
fitPlot(lm1,legend="topleft")
```

`compSlopes` *Tests for significant differences among all pairs of slopes in a dummy variable regression (DVR).*

Description

Tests for significant differences among all pairs of slopes in a dummy variable regression where the dummy variables all stem from one factor.

Usage

```
compSlopes(mdl, method = stats::p.adjust.methods, conf.level = 0.95,
  order.slopes = TRUE, digits = getOption("digits"))

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

Arguments

`mdl`	A `lm` object.
`method`	A string indicating the method of p-value adjustment to use. See details and `p.adjust.methods` (documented in `p.adjust`.
`conf.level`	A single number that represents the level of confidence to use for constructing confidence intervals.
`order.slopes`	A logical indicating whether the slopes should be ordered from smallest to largest upon output.
`digits`	A numeric that controls the number of digits to print.
`x`	A `compSlopes` object (i.e., returned from `compSlopes`).
`...`	Other arguments sent to `print`.

Details

In a dummy variable regression the coefficient for the interaction between the covariate (x) and a dummy variable tests for a difference in slopes between the level of the dummy variable and the reference level. Thus, all dummy variables from a particular linear model fit only compare slopes with the reference level. Other slope comparisons may be found by changing the reference level, which requires refitting the model. This function automates this sequential process and produces a data.frame that shows the estimated difference, an unadjusted confidence interval for the difference, and the unadjusted and adjusted (for multiple comparisons) p-values for testing that the difference in slopes is equal to zero for each pair of levels. The adjusted p-values may be computed with any of the methods coded in `p.adjust` (see `p.adjust.methods` there).

Value

A list with three components. The first component contains the p-value adjustment method in `method`. The second component, called `comparisons`, is a data.frame that contains the following:

comparison	Description of how the difference in levels was computed.
diff	The estimated difference in slope values.
lwr	Lower confidence bound for difference in slope values.
upr	Upper confidence bound for difference in slope values.
p.unadj	Unadjusted p-value for testing the difference in slopes is zero.
p.adj	Adjusted p-value for testing the difference in slopes is zero.

The third component, called slopes, is a data.frame that contains the following:

level	A level name.
slope	The estimated slope value for the given level.
XX LCI	Lower confidence bound for difference in slope values.
XX UCI	Upper confidence bound for difference in slope values.
p.unadj	Unadjusted p-value for testing the slope is zero.
p.adj	Adjusted p-value for testing the slope is zero.

The print function prints the results nicely.

Note

This function only works for linear models with one factor variable.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

compIntercepts in **FSA**.

Examples

```
Mirex$year <- factor(Mirex$year)
# fit a dummy variable regression, see that slopes differ
lm1 <- lm(mirex~weight*year,data=Mirex)
anova(lm1)
# compare all pairs of slopes using default Holm control
compSlopes(lm1)
# compare all pairs of slopes using the false discovery rate control
compSlopes(lm1,method="fdr")
# visualize the results
fitPlot(lm1)
```

`confint.nlsBoot` *Associated S3 methods for nlsBoot from nlstools.*

Description

Provides S3 methods to construct non-parametric bootstrap confidence intervals and hypothesis tests for parameter values and predicted values of the response variable for a `nlsBoot` object from the **nlstools** package.

Usage

```
## S3 method for class 'nlsBoot'
confint(object, parm = NULL, level = conf.level,
  conf.level = 0.95, plot = FALSE, err.col = "black", err.lwd = 2,
  rows = NULL, cols = NULL, ...)

## S3 method for class 'nlsBoot'
predict(object, FUN, conf.level = 0.95,
  digits = NULL, ...)

htest(object, ...)

## S3 method for class 'nlsBoot'
htest(object, parm = NULL, bo = 0,
  alt = c("two.sided", "less", "greater"), plot = FALSE, ...)
```

Arguments

`object`	An object saved from `nlsBoot()`.
`parm`	An integer that indicates which parameter to compute the confidence interval or hypothesis test for. The confidence interval Will be computed for all parameters if `NULL`.
`level`	Same as `conf.level`. Used for compatibility with the main `confint`.
`conf.level`	A level of confidence as a proportion.
`plot`	A logical that indicates whether a plot should be constructed. If `confint`, then a histogram of the `parm` parameters from the bootstrap samples with error bars that illustrate the bootstrapped confidence intervals will be constructed. If codehtest, then a histogram of the `parm` parameters with a vertical lines illustrating the bovalue will be constructed.
`err.col`	A single numeric or character that identifies the color for the error bars on the plot.
`err.lwd`	A single numeric that identifies the line width for the error bars on the plot.
`rows`	A numeric that contains the number of rows to use on the graphic.
`cols`	A numeric that contains the number of columns to use on the graphic.

...	Additional arguments to functions.
FUN	The function to be applied for the prediction. See the examples.
digits	A single numeric that indicates the number of digits for the result.
bo	The null hypothesized parameter value.
alt	A string that identifies the "direction" of the alternative hypothesis. See details.

Details

confint finds the two quantiles that have the proportion (1-conf.level)/2 of the bootstrapped parameter estimates below and above. This is an approximate 100conf.level% confidence interval.

In htest the "direction" of the alternative hypothesis is identified by a string in the alt= argument. The strings may be "less" for a "less than" alternative, "greater" for a "greater than" alternative, or "two.sided" for a "not equals" alternative (the DEFAULT). In the one-tailed alternatives the p-value is the proportion of bootstrapped parameter estimates in object$coefboot that are extreme of the null hypothesized parameter value in bo. In the two-tailed alternative the p-value is twice the smallest of the proportion of bootstrapped parameter estimates above or below the null hypothesized parameter value in bo.

In predict, a user-supplied function is applied to each row of the coefBoot object in a nlsBoot object and then finds the median and the two quantiles that have the proportion (1-conf.level)/2 of the bootstrapped predictions below and above. The median is returned as the predicted value and the quantiles are returned as an approximate 100conf.level% confidence interval for that prediction.

Value

confint returns a matrix with as many rows as columns (i.e., parameter estimates) in the object$coefboot data frame and two columns of the quantiles that correspond to the approximate confidence interval.

htest returns a matrix with two columns. The first column contains the hypothesized value sent to this function and the second column is the corresponding p-value.

predict returns a matrix with one row and three columns, with the first column holding the predicted value (i.e., the median prediction) and the last two columns holding the approximate confidence interval.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

Boot and related methods in **car** and summary.nlsBoot in **nlstools**.

Examples

```
fnx <- function(days,B1,B2,B3) {
  if (length(B1) > 1) {
    B2 <- B1[2]
    B3 <- B1[3]
    B1 <- B1[1]
  }
```

```
  B1/(1+exp(B2+B3*days))
}
nl1 <- nls(cells~fnx(days,B1,B2,B3),data=Ecoli,
           start=list(B1=6,B2=7.2,B3=-1.45))
if (require(nlstools)) {
  nl1.bootn <-  nlstools::nlsBoot(nl1,niter=99) # too few to be useful
  confint(nl1.bootn,"B1")
  confint(nl1.bootn,c(2,3))
  confint(nl1.bootn,conf.level=0.90)
  confint(nl1.bootn,plot=TRUE)
  predict(nl1.bootn,fnx,days=3)
  predict(nl1.bootn,fnx,days=1:3)
  htest(nl1.bootn,1,bo=6,alt="less")
}
```

`CutthroatAL` *Capture histories (9 samples) of Cutthroat Trout from Auke Lake.*

Description

Individual capture histories of Cutthroat Trout (*Oncorhynchus clarki*) in Auke Lake, Alaska, from samples taken in 1998-2006.

Format

A data frame with 1684 observations on the following 10 variables.

id Unique identification numbers for each fish

y1998 Indicator variable for whether the fish was captured in 1998 (1=captured)

y1999 Indicator variable for whether the fish was captured in 1999 (1=captured)

y2000 Indicator variable for whether the fish was captured in 2000 (1=captured)

y2001 Indicator variable for whether the fish was captured in 2001 (1=captured)

y2002 Indicator variable for whether the fish was captured in 2002 (1=captured)

y2003 Indicator variable for whether the fish was captured in 2003 (1=captured)

y2004 Indicator variable for whether the fish was captured in 2004 (1=captured)

y2005 Indicator variable for whether the fish was captured in 2005 (1=captured)

y2006 Indicator variable for whether the fish was captured in 2006 (1=captured)

Topic(s)

- Population Size
- Abundance
- Mark-Recapture
- Capture-Recapture
- Jolly-Seber
- Capture History

Note

Entered into "RMark" format (see CutthroatALf in **FSAdata**) and then converted to individual format with capHistConvert

Source

From Appendix A.3 of Harding, R.D., C.L. Hoover, and R.P. Marshall. 2010. Abundance of Cutthroat Trout in Auke Lake, Southeast Alaska, in 2005 and 2006. Alaska Department of Fish and Game Fisheries Data Series No. 10-82. [Was (is?) from http://www.sf.adfg.state.ak.us/FedAidPDFs/FDS10-82.pdf.]

See Also

Used in mrOpen examples.

Examples

```
str(CutthroatAL)
head(CutthroatAL)
```

---

depletion *Computes the Leslie or DeLury population estimate from catch and effort data.*

Description

Computes the Leslie or DeLury estimates of population size and catchability coefficient from paired catch and effort data. The Ricker modification may also be used.

Usage

```
depletion(catch, effort, method = c("Leslie", "DeLury", "Delury"),
  Ricker.mod = FALSE)

## S3 method for class 'depletion'
summary(object, parm = c("all", "both", "No", "q",
  "lm"), verbose = FALSE, ...)

## S3 method for class 'depletion'
coef(object, parm = c("all", "both", "No", "q",
  "lm"), ...)

## S3 method for class 'depletion'
confint(object, parm = c("all", "both", "No", "q",
  "lm"), level = conf.level, conf.level = 0.95, ...)
```

```
## S3 method for class 'depletion'
anova(object, ...)

## S3 method for class 'depletion'
rSquared(object, digits = getOption("digits"),
  percent = FALSE, ...)

## S3 method for class 'depletion'
plot(x, xlab = NULL, ylab = NULL, pch = 19,
  col.pt = "black", col.mdl = "gray70", lwd = 1, lty = 1,
  pos.est = "topright", cex.est = 0.95, ...)
```

Arguments

catch	A numeric vector of catches of fish at each time.
effort	A numeric vector of efforts expended at each time.
method	A single string that indicates which depletion method to use
Ricker.mod	A single logical that indicates whether to use the modification proposed by Ricker (=TRUE) or not (=FALSE, default).
object	An object saved from the removal call (i.e., of class depletion).
parm	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
verbose	A logical that indicates whether a reminder of the method used should be printed with the summary results.
...	Additional arguments for methods.
level	Same as conf.level but used for compatibility with generic confint function.
conf.level	A single number that represents the level of confidence to use for constructing confidence intervals.
digits	The number of digits to round the rSquared result to.
percent	A logical that indicates if the rSquared result should be returned as a percentage (=TRUE) or as a proportion (=FALSE; default).
x	An object saved from the depletion call (i.e., of class depletion).
xlab	A label for the x-axis.
ylab	A label for the y-axis.
pch	A numeric that indicates the type of plotting character.
col.pt	A string that indicates the color of the plotted points.
col.mdl	A string that indicates the color of the fitted line.
lwd	A numeric that indicates the line width of the fitted line.
lty	A numeric that indicates the type of line used for the fitted line.

`pos.est` A single string to identify where to place the estimated population estimate and catchability on the plot. Can be set to one of `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` or `"center"` for positioning the estimated mortality rates on the plot. Typically `"bottomleft"` (DEFAULT) and `"topright"` will be "out-of-the-way" placements. Set `pos.est` to `NULL` to remove the estimated population size and catchability coefficient from the plot.

`cex.est` A single numeric that identifies the character expansion value for the estimated population estimate and catchability placed on the plot.

Details

For the Leslie method, a linear regression model of catch-per-unit-effort on cumulative catch prior to the sample is fit. The catchability coefficient (q) is estimated from the negative of the slope and the initial population size (No) is estimated by dividing the intercept by the catchability coefficient. If `Ricker.mod=TRUE` then the cumulative catch is modified to be the cumulative catch prior to the sample plus half of the catch of the current sample.

For the DeLury method, a linear regression model of log (catch-per-unit-effort) on cumulative effort is fit. The catchability coefficient (q) is estimated from the negative of the slope and the initial population size (No) is estimated by dividing the intercept as an exponent of e by the catchability coefficient. If `Ricker.mod=TRUE` then the cumulative effort is modified to be the cumulative effort prior to the sample plus half of the effort of the current sample.

Standard errors for the catchability and population size estimates are computed from formulas on page 298 (for Leslie) and 303 (for DeLury) from Seber (2002). Confidence intervals are computed using standard large-sample normal distribution theory with the regression error df.

Value

A list with the following items:

- method A string that indicates whether the `"Leslie"` or `"DeLury"` model was used.
- catch The original vector of catches.
- effort The original vector of efforts.
- cpe A computed vector of catch-per-unit-effort for each time.
- KorE A computed vector of cumulative catch (K; Leslie method) or effort (E; DeLury method).
- lm The `lm` object from the fit of CPE on K (Leslie method) or log(CPE) on E (DeLury method).
- est A 2x2 matrix that contains the estimates and standard errors for No and q.

testing

The Leslie method without the Ricker modification and the DeLury method with the Ricker modification matches the results from `deplet` in **fishmethods** for the `darter` (from **fishmethods**), `LobsterPEI` and `BlueCrab` from **FSAdata**, and `SMBassLS` for N0 to whole numbers, the SE for No to one decimal, q to seven decimals, and the SE of q to at least five decimals.

The Leslie method matches the results of Seber (2002) for N0, q, and the CI for Q but not the CI for N (which was so far off that it might be that Seber's result is incorrect) for the lobster data and

the q and CI for q but the NO or its CI (likely due to lots of rounding in Seber 2002) for the Blue Crab data.

The Leslie and DeLury methods match the results of Ricker (1975) for No and Q but not for the CI of No (Ricker used a very different method to compute CIs).

IFAR Chapter

10-Abundance from Depletion Data.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Seber, G.A.F. 2002. The Estimation of Animal Abundance. Edward Arnold, Second edition (reprinted).

See Also

See `removal` for related functionality and `deplet` in **fishmethods** for similar functionality.

Examples

```
## Leslie model examples
# no Ricker modification
l1 <- depletion(SMBassLS$catch,SMBassLS$effort,method="Leslie")
summary(l1)
summary(l1,verbose=TRUE)
summary(l1,parm="No")
rSquared(l1)
rSquared(l1,digits=1,percent=TRUE)
cbind(Est=coef(l1),confint(l1))
cbind(Est=coef(l1,parm="No"),confint(l1,parm="No"))
cbind(Est=coef(l1,parm="q"),confint(l1,parm="q"))
summary(l1,parm="lm")
plot(l1)

# with Ricker modification
l2 <- depletion(SMBassLS$catch,SMBassLS$effort,method="Leslie",Ricker.mod=TRUE)
summary(l2)
cbind(Est=coef(l2),confint(l1))
plot(l2)

## DeLury model examples
# no Ricker modification
d1 <- depletion(SMBassLS$catch,SMBassLS$effort,method="DeLury")
```

```
summary(d1)
summary(d1,parm="q")
summary(d1,verbose=TRUE)
rSquared(d1)
cbind(Est=coef(d1),confint(d1))
summary(d1,parm="lm")
plot(d1)

# with Ricker modification
d2 <- depletion(SMBassLS$catch,SMBassLS$effort,method="DeLury",Ricker.mod=TRUE)
summary(d2)
cbind(Est=coef(d2),confint(d2))
cbind(Est=coef(d2,parm="q"),confint(d2,parm="q"))
plot(d2)
```

`diags` *Extract diagonals from a matrix.*

Description

Extract diagonals from a matrix.

Usage

```
diags(x, which = 0, incl.labels = c("none", "row", "column"),
  val.name = "value", label.name = "label")
```

Arguments

`x`	A matrix with more than one row AND more than one column.
`which`	A single numeric that indicates which diagonal to extract. A value of zero extracts the main diagonal, whereas negative values extract diagonals from the upper triangle and positive values extract diagonals from the lower triangle. Diagonals further from the main diagonal have `which` values further from zero. If `is.null(which)`, then a matrix of diagonal indices for `which` is shown.
`incl.labels`	A single string that indicates whether `"row"`, `"column"`, or no (`"none"`) labels from `x` should be returned with the values on the diagonal. Will return numeric values if the labels are all diagonal, otherwise character labels are returned.
`val.name`	A single string to name the variable that contains the values from the diagonal in the returned data.frame.
`label.name`	A single string to name the variable that contains the labels in the returned data.frame (see `incl.labels`)

Value

A data.frame with one variable that contains the values from the chosen diagonal of `x` and, optionally, a second variable that contains the chosen labels for those values.

Author(s)

Derek H. Ogle, <derek@derekogle.com>, but relied heavily on http://stackoverflow.com/a/27935808/1123933.

Examples

```
## Square numeric matrix
mat1 <- matrix(1:16,nrow=4)
colnames(mat1) <- LETTERS[1:ncol(mat1)]
rownames(mat1) <- 1:nrow(mat1)
mat1
diags(mat1,which=NULL)
diags(mat1)
diags(mat1,which=-1)
diags(mat1,which=2)
diags(mat1,incl.labels="row")
diags(mat1,which=2,incl.labels="row")
diags(mat1,which=2,incl.labels="col")
( tmp <- diags(mat1,which=2,incl.labels="row",val.name="Freq",label.name="age") )
str(tmp)

## Rectangular numeric matrix
mat2 <- matrix(1:20,nrow=4)
colnames(mat2) <- LETTERS[1:ncol(mat2)]
rownames(mat2) <- 1:nrow(mat2)
mat2
diags(mat2,which=NULL)
diags(mat2,which=-1,incl.labels="row")
diags(mat2,which=2,incl.labels="row")
diags(mat2,which=-4,incl.labels="col")

## Rectangular character matrix
mat3 <- matrix(LETTERS[1:24],nrow=3)
colnames(mat3) <- letters[1:ncol(mat3)]
rownames(mat3) <- 1:nrow(mat3)
mat3
diags(mat3,which=NULL)
diags(mat3,which=-1,incl.labels="row")
diags(mat3,which=2,incl.labels="row")
diags(mat3,which=-4,incl.labels="col")
```

---

dunnTest *Dunn's Kruskal-Wallis Multiple Comparisons.*

Description

Performs Dunn's (1964) test of multiple comparisons following a significant Kruskal-Wallis test, possibly with a correction to control the experimentwise error rate. This is largely a wrapper for the dunn.test function in **dunn.test**. Please see and cite that package.

Usage

```
dunnTest(x, ...)

## Default S3 method:
dunnTest(x, g,
  method = dunn.test::p.adjustment.methods[c(4, 2:3, 5:8, 1)],
  two.sided = TRUE, altp = two.sided, ...)

## S3 method for class 'formula'
dunnTest(x, data = NULL,
  method = dunn.test::p.adjustment.methods[c(4, 2:3, 5:8, 1)],
  two.sided = TRUE, altp = two.sided, ...)

## S3 method for class 'dunnTest'
print(x, dunn.test.results = FALSE, ...)
```

Arguments

x	A numeric vector of data values or a formula of the form x~g.
...	Not yet used.
g	A factor vector or a (non-numeric) vector that can be coerced to a factor vector.
method	A single string that identifies the method used to control the experimentwise error rate. See the list of methods in p.adjustment.methods (documented with dunn.test) in **dunn.test**.
two.sided	A single logical that indicates whether a two-sided p-value should be returned (TRUE; default) or not. See details.
altp	Same as two.sided. Allows similar code with the dunn.test function in **dunn.test**. two.sided is maintained because it pre-dates altp.
data	A data.frame that minimally contains x and g.
dunn.test.results	A single logical that indicates whether the results that would have been printed by dunn.test function in **dunn.test** are shown.

Details

This function performs "Dunn's" test of multiple comparisons following a Kruskal-Wallis test. Unadjusted one- or two-sided p-values for each pairwise comparison among groups are computed following Dunn's description as implemented in the dunn.test function from **dunn.test**. These p-values may be adjusted using methods in the p.adjustment.methods function in **dunn.test**.

This function is largely a wrapper for the dunn.test function in **dunn.test**. Changes here are the possible use of formula notation, results not printed by the main function (but are printed in a more useful format (in my opinion) by the print function), the p-values are adjusted by default with the "holm" method, and two-sided p-values are returned by default. See dunn.test function in **dunn.test** for more details underlying these computations.

Value

A list with three items – method is the long name of the method used to control the experimentwise error rate, dtres is the strings that would have been printed by the dunn.test function in **dunn.test**, and res is a data.frame with the following variables:

- Comparison: Labels for each pairwise comparison.
- Z: Values for the Z test statistic for each comparison.
- P.unadj: Unadjusted p-values for each comparison.
- P.adj: Adjusted p-values for each comparison.

Note

The data.frame will be reduced to only those rows that are complete cases for x and g. In other words, rows with missing data for either x or g are removed from the analysis and a warning will be issued.

There are a number of functions in other packages that do similar analyses.

The results from DunnTest match the results (in a different format) from the dunn.test function from **dunn.test**.

The pairw.kw function from the **asbio** package performs the Dunn test with the Bonferroni correction. The pairw.kw also provides a confidence interval for the difference in mean ranks between pairs of groups. The p-value results from DunnTest match the results from pairw.kw.

The posthoc.kruskal.nemenyi.test function from the **PMCMR** package uses the "Nemenyi" (1963) method of multiple comparisons.

The kruskalmc function from the **pgirmess** package uses the method described by Siegel and Castellan (1988).

It is not clear which method the kruskal function from the **agricolae** package uses. It does not seem to output p-values but it does allow for a wide variety of methods to control the experimentwise error rate.

Author(s)

Derek H. Ogle, <derek@derekogle.com>, but this is largely a wrapper (see details) for dunn.test in **dunn.test** written by Alexis Dinno.

References

Dunn, O.J. 1964. Multiple comparisons using rank sums. Technometrics 6:241-252.

See Also

See kruskal.test, dunn.test in **dunn.test**, posthoc.kruskal.nemenyi.test in **PMCMR**, kruskalmc in **pgirmess**, and kruskal in **agricolae**.

Examples

```
## pH in four ponds data from Zar (2010)
ponds <- data.frame(pond=as.factor(rep(1:4,each=8)),
                    pH=c(7.68,7.69,7.70,7.70,7.72,7.73,7.73,7.76,
                         7.71,7.73,7.74,7.74,7.78,7.78,7.80,7.81,
                         7.74,7.75,7.77,7.78,7.80,7.81,7.84,NA,
                         7.71,7.71,7.74,7.79,7.81,7.85,7.87,7.91))
ponds2 <- ponds[complete.cases(ponds),]

# non-formula usage (default "holm" method)
dunnTest(ponds2$pH,ponds2$pond)

# formula usage (default "holm" method)
dunnTest(pH~pond,data=ponds2)

# other methods
dunnTest(pH~pond,data=ponds2,method="bonferroni")
dunnTest(pH~pond,data=ponds2,method="bh")
dunnTest(pH~pond,data=ponds2,method="none")

# one-sided
dunnTest(pH~pond,data=ponds2,two.sided=FALSE)

# warning message if incomplete cases were removed
dunnTest(pH~pond,data=ponds)

# print dunn.test results
tmp <- dunnTest(pH~pond,data=ponds2)
print(tmp,dunn.test.results=TRUE)
```

Ecoli *Population growth of Escherichia coli.*

Description

The number of *Escherichia coli* cells versus time.

Format

A data frame with 8 observations on the following 2 variables:

days Elapsed duration of the experiment

cells Number of cells in the population

Topic(s)

- Nonlinear Model
- Other

Source

McKendrick, A.G. and M. Kesava Pai. 1911. The Rate of Multiplication of Micro-Organisms: a Mathematical Study. Proceedings of the Royal Society of Edinburgh. 31: 649-655.

See Also

Used in `bootCase` and `fitPlot` examples.

`expandCounts` *Repeat individual fish data (including lengths) from tallied counts.*

Description

Repeat individual fish data, including lengths, from tallied counts and, optionally, add a random digit to length measurements to simulate actual length of fish in the bin. This is useful as a precursor to summaries that require information, e.g., lengths, of individual fish (e.g., length frequency histograms, means lengths).

Usage

```
expandCounts(data, cform, lform = NULL, removeCount = TRUE,
  lprec = 0.1, new.name = "newlen", cwid = 0, verbose = TRUE, ...)
```

Arguments

`data`	A data.frame that contains variables in `cform` and `lform`.
`cform`	A formula of the form `~countvar` where `countvar` generically represents the variable in `data` that contains the counts of individuals. See details.
`lform`	An optional formula of the form `~lowerbin+upperbin` where `lowerbin` and `upperbin` generically represent the variables in `data` that identify the lower- and upper-values of the length bins. See details.
`removeCount`	A single logical that indicates if the variable that contains the counts of individuals (as given in `cform`) should be removed form the returned data.frame. The default is `TRUE` such that the variable will be removed as the returned data.frame contains individuals and the counts of individuals in tallied bins is not relevant to an individual.
`lprec`	A single numeric that controls the precision to which the random lengths are recorded. See details.
`new.name`	A single string that contains a name for the new length variable if random lengths are to be created.
`cwid`	A single positive numeric that will be added to the lower length bin value in instances where the count exceeds one but only a lower (and not an upper) length were recorded. See details.
`verbose`	A logical indicating whether progress message should be printed or not.
`...`	Not yet implemented.

Details

Fisheries data may be recorded as tallied counts in the field. For example, field biologists may have simply recorded that there were 10 fish in one group, 15 in another, etc. More specifically, the biologist may have recorded that there were 10 male Bluegill from the first sampling event between 100 and 124 mm, 15 male Bluegill from the first sampling event between 125 and 149 mm, and so on. At times, it may be necessary to expand these counts such that the repeated information appears in individual rows in a new data.frame. In this specific example, the tallied counts would be repeated such that the male, Bluegill, first sampling event, 100-124 mm information would be repeated 10 times; the male, Bluegill, first sampling event, 125-149 mm information would be repeated 15 times, and so on. This function facilitates this type of expansion.

Length data has often been collected in a "binned-and-tallied" format (e.g., 10 fish in the 100-124 mm group, 15 in the 125-149 mm group, etc.). This type of data collection does not facilitate easy or precise calculations of summary statistics of length (i.e., mean and standard deviations of length). Expanding the data as described above does not solve this problem because the length data are still essentially categorical (i.e., which group the fish belongs to rather than what it's actual length is). To facilitate computation of summary statistics, the data can be expanded as described above and then a length can be randomly selected from within the recorded length bin to serve as a "measured" length for that fish. This function performs this type of expansion by randomly selecting the length from a uniform distribution within the length bin (e.g., each value between 100 and 124 mm has the same probability of being selected).

This function makes some assumptions for some coding situations. First, it assumes that all `lowerbin` values are actually lower than all `upperbin` values. The function will throw an error if this is not true. Second, it assumes that if a `lowerbin` but no `upperbin` value is given then the `lowerbin` value is the exact measurement for those fish. Third, it assumes that if an `upperbin` but no `lowerbin` value is given that this is a data entry error and that the `upperbin` value should be the `lowerbin` value. Fourth, it assumes that it is a data entry error if `varcount` is zero or `NA` and `lowerbin` or `upperbin` contains values (i.e., why would there be lengths if no fish were captured?).

Value

A data.frame of the same structure as `data` except that the variable in `cform` may be deleted and the variable in `new.name` may be added. The returned data.frame will have more rows than `data` because of the potential addition of new individuals expanded from the counts in `cform`.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See `expandLenFreq` for expanding length frequencies where individual fish measurements were made on individual fish in a subsample and the remaining fish were simply counted.

Examples

```
# all need expansion
( d1 <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                   lwr.bin=c(15,15.5,16,16,17,17),
```

```
                  upr.bin=c(15.5,16,16.5,16.5,17.5,17.5),
                  freq=c(6,4,2,3,1,1)) )
expandCounts(d1,~freq)
expandCounts(d1,~freq,~lwr.bin+upr.bin)

# some need expansion
( d2 <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                  lwr.bin=c(15,15.5,16,16,17.1,17.3),
                  upr.bin=c(15.5,16,16.5,16.5,17.1,17.3),
                  freq=c(6,4,2,3,1,1)) )
expandCounts(d2,~freq)
expandCounts(d2,~freq,~lwr.bin+upr.bin)

# none need expansion
( d3 <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                  lwr.bin=c(15,15.5,16,16,17.1,17.3),
                  upr.bin=c(15,15.5,16,16,17.1,17.3),
                  freq=c(6,4,2,3,1,1)) )
expandCounts(d3,~freq)
expandCounts(d3,~freq,~lwr.bin+upr.bin)

# some need expansion, but different bin widths
( d4 <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                  lwr.bin=c(15,  15,  16,  16,  17.1,17.3),
                  upr.bin=c(15.5,15.9,16.5,16.9,17.1,17.3),
                  freq=c(6,4,2,3,1,1)) )
expandCounts(d4,~freq)
expandCounts(d4,~freq,~lwr.bin+upr.bin)

# some need expansion but include zeros and NAs for counts
( d2a <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max","Max","Max","Max"),
                   lwr.bin=c(15,  15.5,16  ,16  ,17.1,17.3,NA,NA,NA),
                   upr.bin=c(15.5,16  ,16.5,16.5,17.1,17.3,NA,NA,NA),
                   freq=c(6,4,2,3,1,1,NA,0,NA)) )
expandCounts(d2a,~freq,~lwr.bin+upr.bin)

# some need expansion but include NAs for upper values
( d2b <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                   lwr.bin=c(15,  15.5,16  ,16  ,17.1,17.3),
                   upr.bin=c(NA  ,NA  ,16.5,16.5,17.1,17.3),
                   freq=c(6,4,2,3,1,1)) )
expandCounts(d2b,~freq,~lwr.bin+upr.bin)

# some need expansion but include NAs for upper values
( d2c <- data.frame(name=c("Johnson","Johnson","Jones","Frank","Frank","Max"),
                   lwr.bin=c(NA,NA,  16  ,16  ,17.1,17.3),
                   upr.bin=c(15,15.5,16.5,16.5,17.1,17.3),
                   freq=c(6,4,2,3,1,1)) )
expandCounts(d2c,~freq,~lwr.bin+upr.bin)

## Not run:
##!!##!!## Change path to where example file is and then run to demo
```

```
## Read in datafile (note periods in names)
df <- read.csv("c:/aaawork/consulting/R_WiDNR/Statewide/Surveysummaries2010.csv")
str(df)
## narrow variables for simplicity
df1 <- df[,c("County","Waterbody.Name","Survey.Year","Gear","Species",
             "Number.of.Fish","Length.or.Lower.Length.IN","Length.Upper.IN",
             "Weight.Pounds","Gender")]
## Sum the count to see how many fish there should be after expansion
sum(df1$Number.of.Fish)

## Simple expansion
df2 <- expandCounts(df1,~Number.of.Fish)

## Same expansion but include random component to lengths (thus new variable)
##   also note default lprec=0.1
df3 <- expandCounts(df1,~Number.of.Fish,~Length.or.Lower.Length.IN+Length.Upper.IN)


## End(Not run)
```

`expandLenFreq` *Expands a length frequency based on a subsample.*

Description

Creates a vector of lengths for the individuals not measured based on the lengths measured in a subsample of individuals.

Usage

```
expandLenFreq(x, w, additional, startcat = NULL, total = additional +
  length(x), decimals = decs$wdec, show.summary = TRUE, ...)
```

Arguments

`x`	A numeric vector of length measurements.
`w`	A number that indicates the width of length classes to create.
`additional`	The number of individuals that were not measured in the sample (for which measurements should be determined).
`startcat`	A number that indicates the beginning of the first length-class.
`total`	The total number of individuals in the sample (including those that were measured in the subsample).
`decimals`	A number that indicates the number of decimals used in the output vector of estimated lengths.
`show.summary`	A logical that indicates whether a summary of the process should be shown at the end.
`...`	Optional arguments to be passed to `lencat`.

Details

Creates a vector of lengths for the individuals not measured based on the lengths measured in a subsample of individuals. Length categories are created first that begin with the value in `startcat` (or the minimum observed value by default) and continue by values of `w` until a category value greater than the largest observed length in `x`. Categories of different widths are not allowed.

The resulting "expanded" lengths are created by allocating individuals to each length class based on the proportion of measured individuals in the subsample in that length class. Individuals within a length class are then assigned a specific length within that length class based on a uniform distribution. Because the expanded number of individuals in a length class is rounded down based on the measured number per length class, not all individuals will initially be assigned a length value. The remaining individuals are assigned to a length class randomly according to weights based on the proportion of individuals in the measured length classes. Finally, these individuals are randomly assigned a specific length within the respective length class from a uniform distribution, same as above.

The resulting length assignments are rounded to the number of decimals shown in `decimal`. If `decimals` is not set by the user then it will default to the same number of decimals shown in the `w` value. Care is taken to make sure that the rounded result will not pass out of the given length category (i.e., will not be allowed to round up to the next length category). Generally speaking, one will want to use more decimals then is shown in `w`. For example, one may want to create length categories with a width of 1 inch (i.e., `w=1`) but have the results recorded as if measured to within 0.1 inch (i.e., `decimals=1`).

Value

Returns a vector that consists of measurements for the non-measured individuals in the entire sample.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See `expandCounts` for expanding more than just lengths or expanding lengths when there is a known number in each length bin. See `lencat` for creating length bins.

Examples

```
## Set the random seed for reproducibility
set.seed(15343437)

## First example
# random lengths measured to nearest 0.1 unit -- values in a vector
len1 <- round(runif(50,0.1,9.9),1)
# assignment of integer lengths to 110 non-measured individuals
new.len1a <- expandLenFreq(len1,w=1,total=160)
new.len1a
# assignment of lengths to 0.1 to 110 non-measured individuals
new.len1b <- expandLenFreq(len1,w=1,total=160,decimals=1)
```

```
new.len1b

## Second example -- if values are in a data.frame
# random lengths measured to nearest 0.1 unit
len2 <- data.frame(len=round(runif(50,10,117),1))
# assignment of lengths to 0.1 for 140 non-measured indivs
new.len2a <- expandLenFreq(len2$len,w=10,total=190,decimals=1)
new.len2a

## Third example
# hypothetically measured lengths
len <- c(6.7,6.9,7.3,7.4,7.5,8.2,8.7,8.9)
# find lengths for unmeasured fish assuming a total of 30
newlen1 <- expandLenFreq(len,w=0.5,total=30,decimals=1)
newlen1
# set a starting length category
newlen2 <- expandLenFreq(len,w=0.5,startcat=6.2,total=30,decimals=1)
newlen2
```

`extraTests` *Likelihood ratio and extra sum-of-squares tests.*

Description

Likelihood ratio and extra sum-of-squares tests with multiple `lm` or `nls` models nested within one common model. This function is most useful when the nested functions are all at the same level; otherwise use `anova()` or `lrtest()` which are more flexible.

Usage

```
lrt(sim, ..., com, sim.names = sim.name, sim.name = NULL,
  com.name = NULL)

extraSS(sim, ..., com, sim.names = sim.name, sim.name = NULL,
  com.name = NULL)

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

Arguments

`sim`	The results of one `lm` or `nls` model, for example, that is a nested subset of the model in `com=`.
`...`	More model results that are nested subsets of the model in `com=`.
`com`	The results of one `lm` or `nls` model, for example, that the models in `sim=` and `...` are a subset of.

`sim.name, sim.names`
: A string vector of "names" for simple model in `sim=` and `....` `sim.names` is preferred but `sim.name` is allowed to allow for a common typing mistake.

`com.name`
: A single "name" string for the complex model in `com=`.

`x`
: An object from `lrt()` or `extraSS()`.

Details

`anova` and `lrtest` (from **lmtest**) provide simple methods for conducting extra sum-of-squares or likelihood ratio tests when one model is nested within another model or when there are several layers of simple models all sequentially nested within each other. However, to compare several models that are nested at the same level with one common more complex model, then `anova()` and `lrtest()` must be repeated for each comparison. This repetition can be eliminated with `lapply()` but then the output is voluminous. This function is designed to remove the repetitiveness and to provide output that is compact and easy to read.

Value

The main function returns a matrix with as many rows as model comparisons and columns of the following types:

- `DfO` The error degrees-of-freedom from the subset (more simple) model.
- `RSSO`, `logLikO` The residual sum-of-squares (from `extraSS`) or log-likelihood (from `lrt`) from the subset (more simple) model.
- `DfA` The error degrees-of-freedom from the `com=` model.
- `RSSA`, `logLikA` The residual sum-of-squares (from `extraSS`) or log-likelihood (from `lrt`) from the `com=` model.
- `Df` The difference in error degrees-of-freedom between the two models.
- `SS`, `logLik` The difference in residual sum-of-squares (from `extraSS`) or log-likelihood (from `lrt`) between the two models.
- `F`, `Chisq` The corresponding F- (from `extraSS`) or Chi-square (from `lrt`) test statistic.
- `Pr(>F)`, `Pr(>Chisq)` The corresponding p-value.

Note

This function is experimental at this point. It seems to work fine for `lm` and `nls` models. An error will be thrown by `extraSS` for other model classes, but `lrt` will not (but it has not been thoroughly tests for other models).

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Example data
df <- data.frame(x=c(1,2,3,4,5,6,7,8,9,10),
                 y=c(4,6,5,7,9,8,7,12,16,22),
                 z=as.factor(rep(c("A","B"),each=5)),
                 w=as.factor(rep(c("A","B"),times=5)))
df$x2 <- df$x^2

## Linear (lm()) models
#  ... regression
fit.0 <- lm(y~1,data=df)
fit.1 <- lm(y~x,data=df)
fit.2 <- lm(y~x2+x,data=df)
extraSS(fit.0,fit.1,com=fit.2)
lrt(fit.0,fit.1,com=fit.2)

# ... show labels for models
extraSS(fit.0,fit.1,com=fit.2,
    sim.names=c("Null Model","Linear"),com.name="Quadratic")
lrt(fit.0,fit.1,com=fit.2,
    sim.names=c("Null Model","Linear"),com.name="Quadratic")

#  ... dummy variable regression
fit.2b <- lm(y~x*z,data=df)
extraSS(fit.0,fit.1,com=fit.2b)
lrt(fit.0,fit.1,com=fit.2b)

#  ... ANOVAs
fit.1 <- lm(y~w,data=df)
fit.2 <- lm(y~w*z,data=df)
extraSS(fit.0,fit.1,com=fit.2)
lrt(fit.0,fit.1,com=fit.2)


## Non-linear (nls()) models
fit.0 = nls(y~c,data=df,start=list(c=10))
fit.1 = nls(y~a*x+c,data=df,start=list(a=1,c=1))
fit.2 = nls(y~b*x2+a*x+c,data=df,start=list(a=-1,b=0.3,c=10))
extraSS(fit.0,fit.1,com=fit.2)
lrt(fit.0,fit.1,com=fit.2)

## General least-squares (gls()) models
## Not run:
  require(nlme)
  fit.0 <- gls(y~1,data=df,method="ML")
  fit.1 <- gls(y~x,data=df,method="ML")
  fit.2 <- gls(y~x2+x,data=df,method="ML")
  lrt(fit.0,fit.1, com=fit.2)
  ## will return an error ... does not work with gls() models
  # extraSS(fit.0,fit.1, com=fit.2)

## End(Not run)
```

fact2num *Converts "numeric" factor levels to numeric values.*

Description

Converts “numeric” factor levels to numeric values.

Usage

```
fact2num(object)
```

Arguments

object	A vector with “numeric” factor levels to be converted to numeric values.

Value

A numeric vector.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
junk <- factor(c(1,7,2,4,3,10))
str(junk)
junk2 <- fact2num(junk)
str(junk2)

## ONLY RUN IN INTERACTIVE MODE
if (interactive()) {

bad <- factor(c("A","B","C"))
# This will result in an error -- levels are not 'numeric'
bad2 <- fact2num(bad)

}  ## END IF INTERACTIVE MODE
```

`fishR` *Opens web pages associated with the fishR website.*

Description

Opens web pages associated with the fishR website in a browser. The user can open the main page or choose a specific page to open.

Usage

```
fishR(where = c("home", "IFAR", "general", "books", "AIFFD", "posts",
  "news"))
```

Arguments

`where`	A string that indicates a particular page on the fishR website to open.

Value

None, but a webpage will be opened in the default browser.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Not run:
## Opens an external webpage ... only run interactively
fishR()            # home page
fishR("IFAR")      # Introduction to Fisheries Analysis with R page
fishR("general")   # examples page
fishR("books")     # examples page
fishR("AIFFD")     # Analysis & Interpretation of Freshw. Fisher. Data page
fishR("posts")     # blog posts (some examples) page

## End(Not run)
```

`fitPlot` *Fitted model plot for an lm, glm, or nls object.*

Description

A generic function for constructing a fitted model plot for an `lm`, `glm`, or `nls` object. Supported objects are linear models from simple linear regression (SLR), indicator variable regression (IVR), one-way ANOVA, or two-way ANOVA models; general linear models that are logistic regressions with a binary response; and non-linear regression with a single numerical response variable, at least one continuous explanatory variable and up to two group-factor explanatory variables.

Usage

```
fitPlot(object, ...)

## S3 method for class 'lm'
fitPlot(object, ...)

## S3 method for class 'SLR'
fitPlot(object, plot.pts = TRUE, pch = 16,
  col.pt = "black", col.mdl = "red", lwd = 3, lty = 1,
  interval = c("none", "confidence", "prediction", "both"),
  conf.level = 0.95, lty.ci = 2, lty.pi = 3,
  xlab = object$Enames[1], ylab = object$Rname, main = "", ...)

## S3 method for class 'IVR'
fitPlot(object, ...)

## S3 method for class 'POLY'
fitPlot(object, ...)

## S3 method for class 'ONEWAY'
fitPlot(object, xlab = object$Enames[1],
  ylab = object$Rname, main = "", type = "b", pch = 16, lty = 1,
  col = "black", interval = TRUE, conf.level = 0.95,
  ci.fun = iCIfp(conf.level), col.ci = col, lty.ci = 1, ...)

## S3 method for class 'TWOWAY'
fitPlot(object, which, change.order = FALSE,
  xlab = object$Enames[ord[1]], ylab = object$Rname, main = "",
  type = "b", pch = c(16, 21, 15, 22, 17, 24, c(3:14)), lty = c(1:6,
  1:6, 1:6), col = "default", interval = TRUE, conf.level = 0.95,
  ci.fun = iCIfp(conf.level), lty.ci = 1, legend = "topright",
  cex.leg = 1, box.lty.leg = 0, ...)

## S3 method for class 'nls'
fitPlot(object, d, pch = c(19, 1), col.pt = c("black",
```

```
  "red"), col.mdl = col.pt, lwd = 2, lty = 1, plot.pts = TRUE,
  jittered = FALSE, ylim = NULL, legend = FALSE,
  legend.lbls = c("Group 1", "Group 2"), ylab = names(mdl$model)[1],
  xlab = names(mdl$model)[xpos], main = "", ...)

## S3 method for class 'glm'
fitPlot(object, ...)

## S3 method for class 'logreg'
fitPlot(object, xlab = names(object$model)[2],
  ylab = names(object$model)[1], main = "", plot.pts = TRUE,
  col.pt = "black", transparency = NULL, plot.p = TRUE,
  breaks = 25, p.col = "blue", p.pch = 3, p.cex = 1,
  yaxis1.ticks = seq(0, 1, 0.1), yaxis1.lbls = c(0, 0.5, 1),
  yaxis2.show = TRUE, col.mdl = "red", lwd = 2, lty = 1,
  mdl.vals = 50, xlim = range(x), ...)
```

Arguments

object	An lm or nls object (i.e., returned from fitting a model with either lm or nls).
...	Other arguments to be passed to the plot functions.
plot.pts	A logical that indicates (TRUE (default)) whether the points are plotted along with the fitted lines. Set to FALSE to plot just the fitted lines.
pch	A numeric or vector of numerics that indicates what plotting character codes should be used. In SLR this is the single value to be used for all points. In IVR a vector is used to identify the characters for the levels of the second factor.
col.pt	A string used to indicate the color of the plotted points. Used only for SLR and logistic regression objects.
col.mdl	A string used to indicate the color of the fitted line. Used only for SLR and logistic regression objects.
lwd	A numeric used to indicate the line width of the fitted line.
lty	A numeric or vector of numerics used to indicate the type of line used for the fitted line. In SLR this is a single value to be used for the fitted line. In IVR a vector is used to identify the line types for the levels of the second factor. See par.
interval	In SLR or IVR, a string that indicates whether to plot confidence (="confidence") or prediction (="prediction") intervals. For a SLR object both can be plotted by using ="both". In one-way or two-way ANOVA, a logical that indicates whether the confidence intervals should be plotted or not.
conf.level	A decimal numeric that indicates the level of confidence to use for confidence and prediction intervals.
lty.ci	a numeric used to indicate the type of line used for the confidence band lines for SLR objects or interval lines for one-way and two-way ANOVA. For IVR, the confidence band types are controlled by lty.
lty.pi	a numeric used to indicate the type of line used for the prediction band lines for SLR objects. For IVR, the prediction band types are controlled by lty. See par.

xlab	a string for labeling the x-axis.
ylab	a string for labeling the y-axis.
main	a string for the main label to the plot. Defaults to the model call.
type	The type of graphic to construct in a one-way and two-way ANOVA. If "b" then points are plotted and lines are used to connect points (DEFAULT). If "p" then only points are used and if "l" then only lines are drawn.
col	A vector of color names or numbers or the name of a palette (see details) that indicates what color of points and lines to use for the levels of the first factor in an IVR or the second factor in a two-way ANOVA.
ci.fun	A function used to put error bars on the one-way or two-way ANOVA graphs. The default is to use the internal iCIfp function which will place t-distribution based confidence intervals on the graph. The user can provide alternative functions that may plot other types of 'error bars'. See examples in lineplot.CI function of **sciplot** package.
col.ci	A vector of color names or numbers or the name of a palette (see details) that indicates what colors to use for the confidence interval bars in one-way and two-way ANOVAs.
which	A character string listing the factor in the two-way ANOVA for which the means should be calculated and plotted. This argument is used to indicate for which factor a main effects plot should be constructed. If left missing then an interaction plot is constructed.
change.order	A logical that is used to change the order of the factors in the lm object. This is used to change which factor is plotted on the x-axis and which is used to connect the means when constructing an interaction plot (ignored if which is used).
legend	Controls use and placement of the legend. See details.
cex.leg	A single numeric values used to represent the character expansion value for the legend. Ignored if legend=FALSE.
box.lty.leg	A single numeric values used to indicate the type of line to use for the box around the legend. The default is to not plot a box.
d	A data frame that contains the variables used in construction of the nls object.
jittered	A logical that indicates whether the points should be jittered horizontally.
ylim	A vector of length two to control the y-axis in the nonlinear regression plot.
legend.lbls	A vector of strings that will be the labels for the legend in an nls fitPlot graphic.
transparency	A numeric that indicates how many points would be plotted on top of each other in a logistic regression before the 'point' would have the full pt.col color. The reciprocal of this value is the alpha transparency value.
plot.p	A logical that indicates if the proportion for categorized values of X are plotted (TRUE; default).
breaks	A number that indicates how many intervals over which to compute proportions or a numeric vector that contains the endpoints of the intervals over which to compute proportions if plot.p=TRUE.
p.col	A color to plot the proportions.

p.pch	A plotting character for plotting the proportions.
p.cex	A character expansion factor for plotting the proportions.
yaxis1.ticks	A numeric vector that indicates where tick marks should be placed on the left y-axis (for the proportion of 'successes') for the logistic regression plot.
yaxis1.lbls	A numeric vector that indicates labels for the tick marks on the left y-axis (for the proportion of 'successes') for the logistic regression plot.
yaxis2.show	A logical that indicates whether the right y-axis should be created (=TRUE; default) or not for the logistic regression plot.
mdl.vals	A numeric that represents the number of values to use for plotting the logistic regression. A larger number means a smoother line.
xlim	A vector of length two to control the x-axis in the logistic regression plot. If this is changed from the default then the domain over which the logistic regression model is plotted will change.

Details

This function does not work with a multiple linear regression, indicator variable regressions with more than two factors, ANOVAs other than one-way and two-way, or models with a categorical response variable. In addition, if the linear model contains a factor then the model must be fit with the quantitative explanatory variable first, followed by the factor(s). This function only works for non-linear models with two or fewer groups.

This function is basically a wrapper to a variety of other functions. For one-way or two-way ANOVAs the primary functions called are `interaction.plot` and `lineplot.CI`. For simple linear regression the function performs similarly to `abline` except that the line is constrained to the domain. For indicator variable regression the function behaves as if several `abline` functions had been called.

A legend can be added to the plot in three different ways. First, if `legend = TRUE` then the R console is suspended until the user places the legend on the graphic by clicking on the graphic at the point where the upper-left corner of the legend should appear. Second, the `legend=` argument can be set to one of `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` and `"center"`. In this case, the legend will be placed inside the plot frame at the given location. Finally, the `legend=` argument can be set to a vector of length two which identifies the plot coordinates for the upper-left corner of where the legend should be placed. A legend will not be drawn if `legend = FALSE` or `legend = NULL`. A legend also will not be drawn if there are not multiple groups in the model.

Value

None. However, a fitted-line plot is produced.

Note

This function is meant to allow newbie students the ability to visualize the most common linear models found in an introductory or intermediate level undergraduate statistics course without getting "bogged-down" in the gritty details of a wide variety of functions. This generic function and it's S3 functions allow the student to visualize the means plot of a one-way anova, the main effects and interaction plots of a two-way ANOVA, the fit of a simple linear regression, the fits of many

lines in an indicator variable regression, and the fit of a non-linear model with a simple and mostly common set of arguments – generally, all that is required is a fitted linear model of the type mentioned here as the first argument. This function thus allows newbie students to interact with and visualize moderately complex linear models in a fairly easy and efficient manner. THIS IS NOT A RESEARCH GRADE FUNCTION and the user should learn how to use the functions that this function is based on, build plots from "scratch", or use more sophisticated plotting packages (e.g., **ggplot2** or **lattice**).

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See abline, regLine in **car**, plotmeans in **gplots**, error.bars in **psych**, interaction.plot, and lineplot.CI in **sciplot** for similar functionality. See residPlot for related functionality.

Examples

```
# create year as a factor variable
Mirex$fyear <- factor(Mirex$year)
# reduce number of years for visual simplicity
Mirex2 <- filterD(Mirex,fyear %in% c(1977,1992))

## Indicator variable regression with two factors
lm1 <- lm(mirex~weight*fyear*species,data=Mirex2)
fitPlot(lm1)
fitPlot(lm1,ylim=c(0,0.8),legend="topleft")

## Indicator variable regression with two factors (but different orders)
lm1r <- lm(mirex~fyear*weight*species,data=Mirex2)
fitPlot(lm1r)
lm1r2 <- lm(mirex~fyear*species*weight,data=Mirex2)
fitPlot(lm1r2)
lm1r3 <- lm(mirex~species*fyear*weight,data=Mirex2)
fitPlot(lm1r3)

## Indicator variable regression with one factor (also showing confidence bands)
lm2 <- lm(mirex~weight*fyear,data=Mirex2)
fitPlot(lm2,legend="topleft")
fitPlot(lm2,legend="topleft",interval="confidence")
fitPlot(lm2,legend="topleft",col="rich",pch=18,lty=1)

## Indicator variable regression with one factor (as first variable)
lm2r <- lm(mirex~fyear*weight,data=Mirex2)
fitPlot(lm2r,legend="topleft",interval="both")

## Indicator variable regression with one factor (assuming parallel lines)
lm3 <- lm(mirex~weight+fyear,data=Mirex2)
fitPlot(lm3,legend="topleft")
fitPlot(lm3,legend="topleft",col="default")
```

```
## Simple linear regression (showing color change and confidence and prediction bands)
lm4 <- lm(mirex~weight,data=Mirex)
fitPlot(lm4,pch=8,col.pt="red")
fitPlot(lm4,col.mdl="blue")
fitPlot(lm4,interval="both")

## One-way ANOVA
lm5 <- lm(mirex~fyear,data=Mirex)
fitPlot(lm5)
fitPlot(lm5,col="red")
fitPlot(lm5,col.ci="red")

## Two-way ANOVA
lm6 <- lm(mirex~fyear*species,data=Mirex)
# interaction plots and a color change
fitPlot(lm6,legend="bottomleft")
fitPlot(lm6,change.order=TRUE)
fitPlot(lm6,col="jet")
# main effects plots
fitPlot(lm6,which="species")
fitPlot(lm6,which="fyear")

## Polynomial regression
lm7 <- lm(mirex~weight+I(weight^2),data=Mirex)
fitPlot(lm7,interval="both")

## Non-linear model example
lr.sv <- list(B1=6,B2=7.2,B3=-1.5)
nl1 <- nls(cells~B1/(1+exp(B2+B3*days)),start=lr.sv,data=Ecoli)
fitPlot(nl1,Ecoli,cex.main=0.7,lwd=2)
fitPlot(nl1,Ecoli,xlab="Day",ylab="Cellsx10^6/ml",plot.pts=FALSE)

## Logistic regression example
## NASA space shuttle o-ring failures -- from graphics package
fail <- factor(c(2,2,2,2,1,1,1,1,1,1,2,1,2,1,1,1,1,2,1,1,1,1,1),
levels = 1:2, labels = c("no","yes"))
temperature <- c(53,57,58,63,66,67,67,67,68,69,70,70,70,70,72,73,75,75,76,76,78,79,81)
d <- data.frame(fail,temperature)
glm1 <- glm(fail~temperature,data=d,family="binomial")
fitPlot(glm1)
fitPlot(glm1,breaks=seq(52,82,2))
fitPlot(glm1,yaxis1.ticks=c(0,1),yaxis1.lbls=c(0,1))
# changing the size of the y-axis labels
par(cex.axis=1.5,cex.lab=1.5)
fitPlot(glm1)
```

FSA *Fisheries stock assessment methods and data.*

Description

Functions to support basic fisheries stock assessment methods.

Details

Functions from this package can be used to perform a variety of basic fisheries stock assessment methods. Detailed descriptions for most functions are available in the Introductory Fisheries Analysis with R book (Ogle 2016). Vignettes for the boxed examples in the "Analysis and Interpretation of Freshwater Fisheries Data" book can be viewed with `fishR("AIFFD")`.

Questions, comments, or suggestions should be given on the GitHub FSA Issues page.

Packages with related functionality by the same author are

- The FSAdata package contains additional data sets.
- The FSAsim package simulation routines for various fisheries methods.
- The FSAWs package contains functions for developing and validating standard weight equations.

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

`fsaNews` *Read news and changes for the 'FSA' package.*

Description

Opens up the News.md GitHub file for the 'FSA' package in an external browser.

Usage

```
fsaNews()

FSANews()
```

Value

None.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Not run:
## Opens an external webpage ... only run interactively
FSANews()

## End(Not run)
```

FSAUtils *Capitalizes the first letter of first or all words in a string.*

Description

Capitalizes the first letter of first or all words in a string.

Usage

```
capFirst(x, which = c("all", "first"))
```

Arguments

x	A single string.
which	A single string that indicates whether all (the default) or only the first words should be capitalized.

Value

A single string with the first letter of the first or all words capitalized.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Capitalize first letter of all words (the default)
capFirst("Derek Ogle")
capFirst("derek ogle")
capFirst("derek")

## Capitalize first letter of only the first words
capFirst("Derek Ogle",which="first")
capFirst("derek ogle",which="first")
capFirst("derek",which="first")
## apply to all elements in a vector
vec <- c("Derek Ogle","derek ogle","Derek ogle","derek Ogle","DEREK OGLE")
capFirst(vec)
capFirst(vec,which="first")
```

```
## check class types
class(vec)
vec1 <- capFirst(vec)
class(vec1)
fvec <- factor(vec)
fvec1 <- capFirst(fvec)
class(fvec1)
```

---

geomean *Calculates the geometric mean or geometric standard deviation.*

Description

Calculates the geometric mean or standard deviation of a vector of numeric values.

Usage

```
geomean(x, na.rm = FALSE, zneg.rm = FALSE)

geosd(x, na.rm = FALSE, zneg.rm = FALSE)
```

Arguments

x	Vector of numeric values.
na.rm	Logical indicating whether to remove missing values or not.
zneg.rm	Logical indicating whether to ignore or remove zero or negative values found in x.

Details

The geometric mean is computed by log transforming the raw data in x, computing the arithmetic mean of the transformed data, and back-transforming this mean to the geometric mean by exponentiating.

The geometric standard deviation is computed by log transforming the raw data in x, computing the arithmetic standard deviation of the transformed data, and back-transforming this standard deviation to the geometric standard deviation by exponentiating.

Value

A numeric value that is the geometric mean or geometric standard deviation of the numeric values in x.

Note

This function is largely an implementation of the code suggested by Russell Senior on R-help in November, 1999.

See Also

See geometric.mean in **psych** and Gmean for geometric mean calculators. See Gsd (documented with Gmean) from **DescTools** for geometric standard deviation calculators.

Examples

```
## generate random lognormal data
d <- rlnorm(500,meanlog=0,sdlog=1)
# d has a mean on log scale of 0; thus, gm should be exp(0)~=1
# d has a sd on log scale of 1; thus, gsd should be exp(1)~=2.7
geomean(d)
geosd(d)

## Not run:
## Demonstrate handling of zeros and negative values
x <- seq(-1,5)
# this will given an error
geomean(x)
# this will only give a warning, but might not be what you want
geomean(x,zneg.rm=TRUE)

## End(Not run)
```

---

growthModels — *Creates a function for a specific parameterization of the von Bertalanffy, Gompertz, Richards, and logistic growth functions.*

Description

Creates a function for a specific parameterizations of the von Bertalanffy, Gompertz, Richards, and logistic growth functions. Use growthFunShow() to see the equations for each growth function.

Usage

```
vbFuns(param = c("Typical", "typical", "Traditional", "traditional",
  "BevertonHolt", "Original", "original", "vonBertalanffy", "GQ",
  "GallucciQuinn", "Mooij", "Weisberg", "Ogle", "Schnute", "Francis",
  "Laslett", "Polacheck", "Somers", "Somers2", "Pauly", "Fabens",
  "Fabens2", "Wang", "Wang2", "Wang3", "Francis2", "Francis3"),
  simple = FALSE, msg = FALSE)

GompertzFuns(param = c("Ricker1", "Ricker2", "Ricker3", "QuinnDeriso1",
  "QuinnDeriso2", "QuinnDeriso3", "QD1", "QD2", "QD3", "Original",
  "original", "Troynikov1", "Troynikov2"), simple = FALSE, msg = FALSE)

RichardsFuns(param = 1, simple = FALSE, msg = FALSE)
```

```
logisticFuns(param = c("CJ1", "CJ2", "Karkach", "Haddon",
  "CampanaJones1", "CampanaJones2"), simple = FALSE, msg = FALSE)

growthFunShow(type = c("vonBertalanffy", "Gompertz", "Richards",
  "Logistic", "Schnute"), param = NULL, case = param, plot = FALSE,
  ...)
```

Arguments

param	A string (for von Bertalanffy, Gompertz, and logistic) or numeric (for Richards) that indicates the specific parameterization of the growth function. See details.
simple	A logical that indicates whether the function will accept all parameter values in the first parameter argument (=FALSE; DEFAULT) or whether all individual parameters must be specified in separate arguments (=TRUE).
msg	A logical that indicates whether a message about the growth function and parameter definitions should be output (=TRUE) or not (=FALSE; DEFAULT).
type	A string (in growthFunShow) that indicates the type of growth function to show.
case	A numeric that indicates the specific case of the Schnute function to use. See details.
plot	A logical that indicates whether the growth function expression should be shown as an equation in a simple plot.
...	Not implemented.

Value

The functions ending in xxxFuns return a function that can be used to predict fish size given a vector of ages and values for the growth function parameters and, in some parameterizations, values for constants. The result should be saved to an object that is then the function name. When the resulting function is used, the parameters are ordered as shown when the definitions of the parameters are printed after the function is called (if msg=TRUE). If simple=FALSE (DEFAULT), then the values for all parameters may be included as a vector in the first parameter argument (but in the same order). Similarly, the values for all constants may be included as a vector in the first constant argument (i.e., t1). If simple=TRUE, then all parameters and constants must be declared individually. The resulting function is somewhat easier to read when simple=TRUE, but is less general for some applications.

An expression of the equation for each growth function may be created with growthFunShow. In this function type= is used to select the major function type (e.g., von Bertalanffy, Gompertz, Richards, Logistic, Schnute) and param= is used to select a specific parameterization of that growth function. If plot=TRUE, then a simple graphic will be created with the equation using plotmath for a pretty format.

IFAR Chapter

12-Individual Growth.

Note

Take note of the following for parameterizations (i.e., param) of each growth function:

- von Bertalanffy
  - The 'Original' and 'vonBertalanffy' are synonymous as are 'Typical', 'Traditional', and 'BevertonHolt'. Further note that the 'Ogle' parameterization has the 'Original'/'vonBertalanffy' and 'Typical'/'Traditional'/'BevertonHolt' parameterizations as special cases.
- Gompertz
  - The 'Ricker2' and 'QuinnDeriso1' are synonymous, as are 'Ricker3' and 'QuinnDeriso2'.
  - The parameterizations and parameters for the Gompertz function are varied and confusing in the literature. I have attempted to use a uniform set of parameters in these functions, but that makes a direct comparison to the literature difficult. Common sources for Gompertz models are listed in the references below. I make some comments here to aid comparisons to the literature.
  - Within FSA, L0 is the mean length at age 0, Linf is the mean asymptotic length, ti is the age at the inflection point, gi is the instantaneous growth rate at the inflection point, t* is a dimensionless parameter related to time/age, and a is a dimensionless parameter related to growth.
  - In the Quinn and Deriso (1999) functions (the 'QuinnDerisoX' functions), the a parameter here is equal to lambda/K there and the gi parameter here is equal to the K parameter there. Also note that their Y is L here.
  - In the Ricker (1979)[p. 705] functions (the 'RickerX' functions), the a parameter here is equal to k there and the gi parameter here is equal to the g parameter there. Also note that their w is L here. In the Ricker (1979) functions as presented in Campana and Jones (1992), the a parameter here is equal to k parameter there and the gi parameter here is equal to the G parameter there. Also note that their X is L here.
  - The function in Ricker (1975)[p. 232] is the same as 'Ricker2' where the a parameter here is qual to G there and the gi parameter here is equal to the g parameter there. Also note that their w is L here.
  - The function in Quist et al. (2012)[p. 714] is the same as 'Ricker1' where the gi parameter here is equal to the G parameter there and the ti parameter here is equal to the t0 parameter there.
  - The function in Katsanevakis and Maravelias (2008) is the same as 'Ricker1' where the gi parameter here is equal to k2 parameter there and the ti parameter here is equal to the t2 parameter there.
- Richards
  - Within FSA, Linf is the mean asymptotic length, ti is the age at the inflection point, k is related to growth (slope at the inflection point), b is related to the vertical position of the inflection point, and L0 is the mean length at age-0.
  - The parameterizations (1-6) correspond to functions/equations 1, 4, 5, 6, 7, and 8, respectively, in Tjorve and Tjorve (2010). Note that their A, S, k, d, and B are Linf, a, k, b, and L0, respectively, here (in FSA).
- logistic
  - Within FSA, L0 is the mean length at age 0, Linf is the mean asymptotic length, ti is the age at the inflection point, and gninf is the instantaneous growth rate at negative infinity.

Author(s)

Derek H. Ogle, <derek@derekogle.com>, thanks to Gabor Grothendieck for a hint about using `get()`.

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Campana, S.E. and C.M. Jones. 1992. Analysis of otolith microstructure data. Pages 73-100 In D.K. Stevenson and S.E. Campana, editors. Otolith microstructure examination and analysis. Canadian Special Publication of Fisheries and Aquatic Sciences 117. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/141734.pdf.]

Fabens, A. 1965. Properties and fitting of the von Bertalanffy growth curve. Growth 29:265-289.

Francis, R.I.C.C. 1988. Are growth parameters estimated from tagging and age-length data comparable? Canadian Journal of Fisheries and Aquatic Sciences, 45:936-942.

Gallucci, V.F. and T.J. Quinn II. 1979. Reparameterizing, fitting, and testing a simple growth model. Transactions of the American Fisheries Society, 108:14-25.

Garcia-Berthou, E., G. Carmona-Catot, R. Merciai, and D.H. Ogle. A technical note on seasonal growth models. Reviews in Fish Biology and Fisheries 22:635-640. [Was (is?) from https://www.researchgate.net/publication

Gompertz, B. 1825. On the nature of the function expressive of the law of human mortality, and on a new method of determining the value of life contingencies. Philosophical Transactions of the Royal Society of London. 115:513-583.

Haddon, M., C. Mundy, and D. Tarbath. 2008. Using an inverse-logistic model to describe growth increments of Blacklip Abalone (*Haliotis rubra*) in Tasmania. Fishery Bulletin 106:58-71. [Was (is?) from http://aquaticcommons.org/8857/1/haddon_Fish_Bull_2008.pdf.]

Karkach, A. S. 2006. Trajectories and models of individual growth. Demographic Research 15:347-400. [Was (is?) from http://www.demographic-research.org/volumes/vol15/12/15-12.pdf.]

Katsanevakis, S. and C.D. Maravelias. 2008. Modeling fish growth: multi-model inference as a better alternative to a priori using von Bertalanffy equation. Fish and Fisheries 9:178-187.

Mooij, W.M., J.M. Van Rooij, and S. Wijnhoven. 1999. Analysis and comparison of fish growth from small samples of length-at-age data: Detection of sexual dimorphism in Eurasian Perch as an example. Transactions of the American Fisheries Society 128:483-490.

Polacheck, T., J.P. Eveson, and G.M. Laslett. 2004. Increase in growth rates of southern Bluefin Tuna (*Thunnus maccoyii*) over four decades: 1960 to 2000. Canadian Journal of Fisheries and Aquatic Sciences, 61:307-322.

Quinn, T. J. and R. B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press, New York, New York. 542 pages.

Quist, M.C., M.A. Pegg, and D.R. DeVries. 2012. Age and Growth. Chapter 15 in A.V. Zale, D.L Parrish, and T.M. Sutton, Editors Fisheries Techniques, Third Edition. American Fisheries Society, Bethesda, MD.

Richards, F. J. 1959. A flexible growth function for empirical use. Journal of Experimental Biology 10:290-300.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Ricker, W.E. 1979. Growth rates and models. Pages 677-743 In W.S. Hoar, D.J. Randall, and J.R. Brett, editors. Fish Physiology, Vol. 8: Bioenergetics and Growth. Academic Press, NY, NY. [Was (is?) from https://books.google.com/books?id=CB1qu2VbKwQC&pg=PA705&lpg=PA705&dq=Gompertz+fish&source=bl

Schnute, J. 1981. A versatile growth model with statistically stable parameters. Canadian Journal of Fisheries and Aquatic Sciences, 38:1128-1140.

Somers, I. F. 1988. On a seasonally oscillating growth function. Fishbyte 6(1):8-11. [Was (is?) from http://www.worldfishcenter.org/Naga/na_2914.pdf.]

Tjorve, E. and K. M. C. Tjorve. 2010. A unified approach to the Richards-model family for use in growth analyses: Why we need only two model forms. Journal of Theoretical Biology 267:417-425. [Was (is?) from https://www.researchgate.net/profile/Even_Tjorve/publication/46218377_A_unified_approach_to_the_Richa model_family_for_use_in_growth_analyses_why_we_need_only_two_model_forms/links/54ba83b80cf29e0cb04bd24e.pdf

Troynikov, V. S., R. W. Day, and A. M. Leorke. Estimation of seasonal growth parameters using a stochastic Gompertz model for tagging data. Journal of Shellfish Research 17:833-838. [Was (is?) from https://www.researchgate.net/profile/Robert_Day2/publication/249340562_Estimation_of_seasonal_growth_parameter

Vaughan, D. S. and T. E. Helser. 1990. Status of the Red Drum stock of the Atlantic coast: Stock assessment report for 1989. NOAA Technical Memorandum NMFS-SEFC-263, 117 p. [Was (is?) from http://docs.lib.noaa.gov/noaa_documents/NMFS/SEFSC/TM_NMFS_SEFSC/NMFS_SEFSC_TM_263.pdf.]

Wang, Y.-G. 1998. An improved Fabens method for estimation of growth parameters in the von Bertalanffy model with individual asymptotes. Canadian Journal of Fisheries and Aquatic Sciences 55:397-400.

Weisberg, S., G.R. Spangler, and L. S. Richmond. 2010. Mixed effects models for fish growth. Canadian Journal of Fisheries And Aquatic Sciences 67:269-277.

Winsor, C.P. 1932. The Gompertz curve as a growth curve. Proceedings of the National Academy of Sciences. 18:1-8. [Was (is?) from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076153/pdf/pnas01729-0009.pdf.]

See Also

See Schnute for an implementation of the Schnute (1981) model.

Examples

```
############################################################
## Simple Examples -- Von B
( vb1 <- vbFuns() )
ages <- 0:20
plot(vb1(ages,Linf=20,K=0.3,t0=-0.2)~ages,type="b",pch=19)
( vb2 <- vbFuns("Francis") )
plot(vb2(ages,L1=10,L2=19,L3=20,t1=2,t3=18)~ages,type="b",pch=19)
( vb2c <- vbFuns("Francis",simple=TRUE) )   # compare to vb2

## Simple Examples -- Gompertz
( gomp1 <- GompertzFuns() )
plot(gomp1(ages,Linf=800,gi=0.5,ti=5)~ages,type="b",pch=19)
( gomp2 <- GompertzFuns("Ricker2") )
plot(gomp2(ages,L0=2,a=6,gi=0.5)~ages,type="b",pch=19)
( gomp2c <- GompertzFuns("Ricker2",simple=TRUE) )   # compare to gomp2
( gompT <- GompertzFuns("Troynikov1"))

## Simple Examples -- Richards
( rich1 <- RichardsFuns() )
```

```
plot(rich1(ages,Linf=800,k=0.5,a=1,b=6)~ages,type="b",pch=19)
( rich2 <- RichardsFuns(2) )
plot(rich2(ages,Linf=800,k=0.5,ti=3,b=6)~ages,type="b",pch=19)
( rich3 <- RichardsFuns(3) )
plot(rich3(ages,Linf=800,k=0.5,ti=3,b=0.15)~ages,type="b",pch=19)
( rich4 <- RichardsFuns(4) )
plot(rich4(ages,Linf=800,k=0.5,ti=3,b=0.95)~ages,type="b",pch=19)
lines(rich4(ages,Linf=800,k=0.5,ti=3,b=1.5)~ages,type="b",pch=19,col="blue")
( rich5 <- RichardsFuns(5) )
plot(rich5(ages,Linf=800,k=0.5,L0=50,b=1.5)~ages,type="b",pch=19)
( rich6 <- RichardsFuns(6) )
plot(rich6(ages,Linf=800,k=0.5,ti=3,Lninf=50,b=1.5)~ages,type="b",pch=19)
( rich2c <- RichardsFuns(2,simple=TRUE) ) # compare to rich2

## Simple Examples -- Logistic
( log1 <- logisticFuns() )
plot(log1(ages,Linf=800,gninf=0.5,ti=5)~ages,type="b",pch=19)
( log2 <- logisticFuns("CJ2") )
plot(log2(ages,Linf=800,gninf=0.5,a=10)~ages,type="b",pch=19)
( log2c <- logisticFuns("CJ2",simple=TRUE) ) # compare to log2
( log3 <- logisticFuns("Karkach") )
plot(log3(ages,L0=10,Linf=800,gninf=0.5)~ages,type="b",pch=19)
( log4 <- logisticFuns("Haddon") )


############################################################
## Examples of fitting
##   After the last example a plot is constructed with three
##   or four lines on top of each other illustrating that the
##   parameterizations all produce the same fitted values.
##   However, observe the correlations in the summary() results.

## Von B
plot(tl~age,data=SpotVA1,pch=19)

# Fitting the typical parameterization of the von B function
fit1 <- nls(tl~vb1(age,Linf,K,t0),data=SpotVA1,
            start=vbStarts(tl~age,data=SpotVA1))
summary(fit1,correlation=TRUE)
curve(vb1(x,Linf=coef(fit1)),from=0,to=5,col="red",lwd=10,add=TRUE)

# Fitting the Francis parameterization of the von B function
fit2 <- nls(tl~vb2c(age,L1,L2,L3,t1=0,t3=5),data=SpotVA1,
            start=vbStarts(tl~age,data=SpotVA1,type="Francis",ages2use=c(0,5)))
summary(fit2,correlation=TRUE)
curve(vb2c(x,L1=coef(fit2)[1],L2=coef(fit2)[2],L3=coef(fit2)[3],t1=0,t3=5),
      from=0,to=5,col="blue",lwd=5,add=TRUE)

# Fitting the Schnute parameterization of the von B function
vb3 <- vbFuns("Schnute")
fit3 <- nls(tl~vb3(age,L1,L3,K,t1=0,t3=4),data=SpotVA1,
            start=vbStarts(tl~age,data=SpotVA1,type="Schnute",ages2use=c(0,4)))
summary(fit3,correlation=TRUE)
```

```
curve(vb3(x,L1=coef(fit3),t1=c(0,4)),from=0,to=5,col="green",lwd=2,add=TRUE)

## Gompertz
# Make some fake data using the original parameterization
gompO <- GompertzFuns("original")
# setup ages, sample sizes (general reduction in numbers with
# increasing age), and additive SD to model
t <- 1:15
n <- c(10,40,35,25,12,10,10,8,6,5,3,3,3,2,2)
sd <- 15
# expand ages
ages <- rep(t,n)
# get lengths from gompertz and a random error for individuals
lens <- gompO(ages,Linf=450,a=1,gi=0.3)+rnorm(length(ages),0,sd)
# put together as a data.frame
df <- data.frame(age=ages,len=round(lens,0))

plot(len~age,data=df,pch=19,col=rgb(0,0,0,1/5))

# Fit first Ricker parameterization
fit1 <- nls(len~gomp1(age,Linf,gi,ti),data=df,start=list(Linf=500,gi=0.3,ti=3))
summary(fit1,correlation=TRUE)
curve(gomp1(x,Linf=coef(fit1)),from=0,to=15,col="red",lwd=10,add=TRUE)

# Fit third Ricker parameterization
fit2 <- nls(len~gomp2(age,L0,a,gi),data=df,start=list(L0=30,a=3,gi=0.3))
summary(fit2,correlation=TRUE)
curve(gomp2(x,L0=coef(fit2)),from=0,to=15,col="blue",lwd=5,add=TRUE)

# Fit third Quinn and Deriso parameterization (using simple=TRUE model)
gomp3 <- GompertzFuns("QD3",simple=TRUE)
fit3 <- nls(len~gomp3(age,Linf,gi,t0),data=df,start=list(Linf=500,gi=0.3,t0=0))
summary(fit3,correlation=TRUE)
curve(gomp3(x,Linf=coef(fit3)[1],gi=coef(fit3)[2],t0=coef(fit3)[3]),
      from=0,to=15,col="green",lwd=2,add=TRUE)

## Richards

## Not run:
# Fit first Richards parameterization ... DOES NOT CONVERGE
fit1 <- nls(len~rich1(age,Linf,k,a,b),data=df,
            start=list(Linf=450,k=0.3,a=0.2,b=3))
summary(fit1,correlation=TRUE)
curve(rich1(x,Linf=coef(fit1)),from=0,to=15,col="red",lwd=10,add=TRUE)

# Fit second Richards parameterization ... DOES NOT CONVERGE
fit2 <- nls(len~rich2(age,Linf,k,ti,b),data=df,
            start=list(Linf=450,k=0.25,ti=3,b=3))
summary(fit2,correlation=TRUE)
curve(rich2(x,Linf=coef(fit2)),from=0,to=15,col="blue",lwd=7,add=TRUE)

## End(Not run)
```

```
plot(len~age,data=df,pch=19,col=rgb(0,0,0,1/5))

# Fit third Richards parameterization
fit3 <- nls(len~rich3(age,Linf,k,ti,b),data=df,
            start=list(Linf=450,k=0.25,ti=3,b=-0.1))
summary(fit3,correlation=TRUE)
curve(rich3(x,Linf=coef(fit3)),from=0,to=15,col="green",lwd=4,add=TRUE)

# Fit fourth Richards parameterization
fit4 <- nls(len~rich4(age,Linf,k,ti,b),data=df,
            start=list(Linf=450,k=0.25,ti=3,b=0.7))
summary(fit4,correlation=TRUE)
curve(rich4(x,Linf=coef(fit4)),from=0,to=15,col="black",lwd=1,add=TRUE)

## Logistic
plot(len~age,data=df,pch=19,col=rgb(0,0,0,1/5))

# Fit first Campana-Jones parameterization
fit1 <- nls(len~log1(age,Linf,gninf,ti),data=df,
            start=list(Linf=450,gninf=0.45,ti=4))
summary(fit1,correlation=TRUE)
curve(log1(x,Linf=coef(fit1)),from=0,to=15,col="red",lwd=10,add=TRUE)

# Fit second Campana-Jones parameterization
fit2 <- nls(len~log2(age,Linf,gninf,a),data=df,
            start=list(Linf=450,gninf=0.45,a=7))
summary(fit2,correlation=TRUE)
curve(log2(x,Linf=coef(fit2)),from=0,to=15,col="blue",lwd=5,add=TRUE)

# Fit Karkach parameterization (using simple=TRUE model)
log3 <- logisticFuns("Karkach",simple=TRUE)
fit3 <- nls(len~log3(age,Linf,L0,gninf),data=df,
            start=list(Linf=450,L0=30,gninf=0.45))
summary(fit3,correlation=TRUE)
curve(log3(x,Linf=coef(fit3)[1],L0=coef(fit3)[2],gninf=coef(fit3)[3]),
      from=0,to=15,col="green",lwd=2,add=TRUE)


################################################################################
## Create expressions of the models
################################################################################
# Typical von Bertalanffy ... Show as a stand-alone plot
growthFunShow("vonBertalanffy","Typical",plot=TRUE)
# Get and save the expression
( tmp <- growthFunShow("vonBertalanffy","Typical") )
# Use expression as title on a plot
lens <- vb1(ages,Linf=20,K=0.3,t0=-0.2)
plot(lens~ages,type="b",pch=19,main=tmp)
# Put expression in the main plot
text(10,5,tmp)
# Put multiple expressions on a plot
op <- par(mar=c(0.1,0.1,0.1,0.1))
plot(0,type="n",xlab="",ylab="",xlim=c(0,1),ylim=c(0,3),xaxt="n",yaxt="n")
```

```
text(0,2.5,"Original:",pos=4)
text(0.5,2.5,growthFunShow("vonBertalanffy","Original"))
text(0,1.5,"Typical:",pos=4)
text(0.5,1.5,growthFunShow("vonBertalanffy","Typical"))
text(0,0.5,"Francis:",pos=4)
text(0.5,0.5,growthFunShow("vonBertalanffy","Francis"))
par(op)
```

`headtail` *Shows rows from the head and tail of a data frame or matrix.*

Description

Shows rows from the head and tail of a data frame or matrix.

Usage

```
headtail(x, n = 3L, which = NULL, addrownums = TRUE, ...)
```

Arguments

`x`	A data frame or matrix.
`n`	A single numeric that indicates the number of rows to display from each of the head and tail of structure.
`which`	A numeric or string vector that contains the column numbers or names to display. Defaults to showing all columns.
`addrownums`	If there are no row names for the MATRIX, then create them from the row numbers.
`...`	Arguments to be passed to or from other methods.

Value

A matrix or data.frame with 2*n rows.

Note

If `n` is larger than the number of rows in `x` then all of `x` is displayed.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
headtail(iris)
headtail(iris,10)
headtail(iris,which=c("Sepal.Length","Sepal.Width","Species"))
headtail(iris,which=grep("Sepal",names(iris)))
headtail(iris,n=200)

## Make a matrix for demonstration purposes only
miris <- as.matrix(iris[,1:4])
headtail(miris)
headtail(miris,10)
headtail(miris,addrownums=FALSE)
headtail(miris,10,which=2:4)

## Make a tbl_df type from dplyr ... note how headtail()
## is not limited by the tbl_df restriction on number of
## rows to show (but head() is).
if (require(dplyr)) {
  iris2 <- tbl_df(iris)
  class(iris2)
  headtail(iris2,n=15)
  head(iris2,n=30)
}
```

---

hist.formula *Creates separate histograms by levels.*

Description

Creates separate histograms of a quantitative variable by levels of a factor variable.

Usage

```
## S3 method for class 'formula'
hist(formula, data = NULL, main = "",
  right = FALSE, pre.main = "", xlab = NULL, ylab = "Frequency",
  same.breaks = TRUE, breaks = "Sturges", w = NULL,
  same.ylim = TRUE, ymax = NULL, col = "gray90",
  nrow = round(sqrt(num)), ncol = ceiling(sqrt(num)), byrow = TRUE,
  iaxs = TRUE, ...)
```

Arguments

formula	A formula. See details.
data	An optional data frame that contains the variables in the model.
main	A character string used as the main title for when a SINGLE histogram is produced.

right	A logical that indicates if the histogram bins are right-closed (left open) intervals (=TRUE) or not (=FALSE; default).
pre.main	A character string to be used as a prefix for the main title when multiple histograms are produced. See details.
xlab	A character label for the x-axis. Defaults to name of quantitative variable in formula.
ylab	A character label for the y-axis. Defaults to "Frequency".
same.breaks	A logical that indicates whether the same break values (i.e., bins) should be used on each histogram. Ignored if breaks or w is provided by the user. Defaults to TRUE.
breaks	A single numeric that indicates the number of bins or breaks or a vector that contains the lower values of the breaks. Ignored if w is not NULL. See hist for more details.
w	A single numeric that indicates the width of the bins to use. The bins will start at "rounded" values depending on the value of w. See lencat for more details.
same.ylim	A logical that indicates whether the same limits for the y-axis should be used on each histogram. Defaults to TRUE.
ymax	A single value that sets the maximum y-axis limit for each histogram or a vector of length equal to the number of groups that sets the maximum y-axis limit for each histogram separately. If NULL (default), then a value will be found.
col	A string that indicates the color for the bars on the histogram. Defaults to a light shade of gray (i.e., "gray90").
nrow	A single numeric that contains the number of rows to use on the graphic.
ncol	A single numeric that contains the number of columns to use on the graphic.
byrow	A single logical that indicates if the histograms should fill rows first (=TRUE or columns first (=FALSE).
iaxs	A single logical that indicates whether both axes should be plotted using xaxs="i" and yaxs="i" (the default) or xaxs="r" and yaxs="r" (what R typically does).
...	Other arguments to pass through to the default hist().

Details

The formula must be of the form ~quantitative, quantitative~1, quantitative~factor, or quantitative~factor*factor2 where quantitative is the quantitative variable to construct the histograms for and factor or factor2 are factor variables that contain the levels for which separate histograms should be constructed.

If the formula is of the form ~quantitative or quantitative~1 then only a single histogram of the quantitative variable will be produced. This allows hist.formula() to be used similarly to hist() but with a data= argument.

The function produces a single (but see below) graphic that consists of a grid on which the separate histograms are printed. The rows and columns of this grid are determined to construct a plot that is as square as possible. However, the rows and columns can be set by the user with the nrow= and ncol= arguments. If the product of the number of rows and number of columns set by the user is less than the total number of histograms to be constructed then multiple pages of histograms will

be produced (each requiring the user to click on the graph to go to the next graph). The x-axis of each separate histogram will be labeled identically. The default x-axis label is the name of the quantitative variable. This can be changed by the user with the `xlab=` argument.

The default for `right=` is not the same as that used in `hist()` from **graphics**. Thus, right-open (left-closed) bins are the default.

The `iaxs=` argument defaults to `TRUE` so that `xaxs="i"` and `yaxs="i"` are used for both axes, which eliminates the "floating" x-axis that R typically plots for histograms.

Value

A graphic is produced and nothing is returned unless `formula` results in only one histogram. In that case, an object of class `"histogram"` is returned, which is described in `hist`.

IFAR Chapter

3-Plotting Fundamentals.

Note

Students often need to look at the distribution of a quantitative variable separated for different levels of a categorical variable. One method for examining these distributions is with `boxplot(quantitative~factor)`. Other methods use functions in **Lattice** and **ggplots2** but these packages have some learning 'overhead' for newbie students. The formula notation, however, is a common way in R to tell R to separate a quantitative variable by the levels of a factor. Thus, this function adds code for formulas to the generic `hist` function. This allows newbie students to use a common notation (i.e., formula) to easily create multiple histograms of a quantitative variable separated by the levels of a factor.

Author(s)

Derek H. Ogle, <derek@derekogle.com>, but this implementation is largely a modification of the code provided by Marc Schwartz on the R-help mailing list on 1Jun07.

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See base `hist` for related functionality and `multhist` in **plotrix** for similar functionality.

Examples

```
## Using the defaults
hist(Sepal.Length~Species,data=iris)

## Add x-labels and use a pre-fix on the main labels
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",
     pre.main="Species==")

## Use different breaks and different y-axis limits for each graph
```

```
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",
     same.breaks=FALSE,same.ylim=FALSE)

## Use same but user-controlled breaks for each graph
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",
     breaks=seq(4,8,1))

## Use same but user-controlled maximum value for y-axis
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",ymax=30)

## Control the organization of the 'grid' of histograms
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",nrow=1,ncol=3)

## Use right=FALSE & freq=FALSE to demon sending an argument used by base hist()
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",right=FALSE,
     freq=FALSE,ymax=2)

## Add a junk variable to the iris data set to show two factors on RHS
iris$junk <- factor(sample(c("A","B"),nrow(iris),replace=TRUE))
hist(Sepal.Length~Species*junk,data=iris,xlab="Sepal Length (cm)")

## Single histogram without grouping using formula notation
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)")

## Single histogram with "axis correction" turned off (compare to previous)
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)",iaxs=FALSE)

## Single histogram with "axis correction", testing xlim and ylim
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)",
     xlim=c(3.8,8.2),ylim=c(0,35))
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)",
     xlim=c(3.8,8.2),ymax=35)

## Using the bin width argument
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)",w=1)
hist(~Sepal.Length,data=iris,xlab="Sepal Length (cm)",w=0.25)
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",w=1)
hist(Sepal.Length~Species,data=iris,xlab="Sepal Length (cm)",w=0.25)

## Using a vector (and not a data.frame)
vec <- 1:100
hist(~vec)
```

`histFromSum` *Create a histogram from a frequency table.*

Description

Creates a histogram from values in a frequency table. Primarily used with already summarized length frequency data.

Usage

```
histFromSum(x, ...)

## Default S3 method:
histFromSum(x, y, ...)

## S3 method for class 'table'
histFromSum(x, ...)

## S3 method for class 'formula'
histFromSum(x, data = NULL, ...)
```

Arguments

x	A numeric vector of bin/category values, a formula of the form `freq~cat` where `freq` contains the count/frequency values and `cat` contains the bin/category values, an object of class `table` from `table()` or `xtabs()`.
...	Additional arguments for `hist`.
y	A numeric vector of count/frequency values.
data	A data.frame that contains the `freq` and `cat` variables if a formula is given in `x`.

Details

Creates a histogram fro values in a frequency table. The frequency table may be constructed from `xtabs`, `table`, or be in the form of a matrix or a data.frame (as if read in from an external data file).

Value

None, but a graphic is created.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See `hist` and `hist.formula` for related functionality.

Examples

```
## Make some dummy data with a length category variable
set.seed(634434789)
df <- data.frame(tl=round(rnorm(100,100,20)))
df$lcat10 <- lencat(df$tl,w=10)

## Summarize as tables
( tbl1 <- xtabs(~lcat10,data=df) )
( tbl2 <- table(df$lcat10) )
```

```
## Turn the tables into a data.frame for testing (convert
## the categories variables to numeric with fact2num())
df2 <- data.frame(tbl1)
df2$lcat10 <- fact2num(df2$lcat10)

## Turn the table into a matrix for testing
( mat1 <- cbind(lcat10=as.numeric(rownames(tbl1)),freq=tbl1) )

## Histogram of the raw data ... set breaks and x-axis label
brks <- seq(20,160,10)
xlbl <- "Total Length (mm)"
hist(~tl,data=df,breaks=brks,xlab=xlbl)

## Use this function with various inputs ... changed colors
## on each plot so that it was obvious that a new plot was made.
# table from xtabs()
histFromSum(tbl1,breaks=brks,xlab=xlbl,col="gray75")
# table from table()
histFromSum(tbl2,breaks=brks,xlab=xlbl,col="gray70")
# vectors from data.frame
histFromSum(df2$lcat10,df2$Freq,breaks=brks,xlab=xlbl,col="gray65")
# vectors from matrix
histFromSum(mat1[,"lcat10"],mat1[,"freq"],breaks=brks,xlab=xlbl,col="gray60")
# formula from a data.frame
histFromSum(Freq~lcat10,data=df2,breaks=brks,xlab=xlbl,col="gray55")
```

---

hoCoef *Performs a hypothesis test that a linear model parameter is equal to a specific value.*

Description

Performs a hypothesis test that a linear model parameter is equal to a specific value. Useful for testing that a parameter is equal to a value other than 0.

Usage

```
hoCoef(object, term = 2, bo = 0, alt = c("two.sided", "less",
  "greater"))
```

Arguments

`object`	A `lm` object.
`term`	A single numeric that indicates which term in the model to use in the hypothesis test.
`bo`	The null hypothesized parameter value.
`alt`	A string that identifies the "direction" of the alternative hypothesis. The strings may be `"less"` for a "less than" alternative, `"greater"` for a "greater than" alternative, or `"two.sided"` (DEFAULT) for a "not equals" alternative.

Details

The "direction" of the alternative hypothesis is identified by a string in the `alt` argument.

If the `lm` object is from a simple linear regression with an intercept then `term=1` will use the intercept and `term=2` will use the slope in the hypothesis test.

Value

A matrix that contains the term number, hypothesized value, parameter estimate, standard error of the parameter estimate, t test statistic, degrees-of-freedom, and corresponding p-value.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

`htest.nlsBoot`.

Examples

```
# Simple linear regression test HA:slope!=0.1
lm1 <- lm(mirex~weight, data=Mirex)
hoCoef(lm1,2,0.1)
```

---

hyperCI — *Confidence interval for population size (N) in hypergeometric distribution.*

Description

Computes a confidence interval for population size (N) in hypergeometric distribution.

Usage

```
hyperCI(M, n, m, conf.level = 0.95)
```

Arguments

`M`	Number of successes in the population.
`n`	Number of observations in the sample.
`m`	Number of observed successes in the sample.
`conf.level`	Level of confidence to use for constructing confidence intervals (default is `0.95`).

Details

This is an inefficient brute-force algorithm. The algorithm computes the `conf.level` range of possible values for `m`, as if it was unknown, for a large range of values of N. It then finds all possible values of N for which `m` was in the `conf.level` range. The smallest and largest values of N for which `m` was in the `conf.level` range are the CI endpoints.

Value

A 1x2 matrix that contains the lower and upper confidence interval bounds.

Note

This algorithm is experimental at this point.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
hyperCI(50,25,10)
```

---

jolly *Jolly-Seber analysis from multiple mark-recapture events from an open population.*

Description

This function takes the two parts of a Method B table and uses the Jolly-Seber method to estimate the population size at each possible sample period and the apparent survival rate and number of additional individuals added to the population between possible sample periods. This method assumes that the population is open.

Usage

```
jolly(...)

mrOpen(mb.top, mb.bot = NULL, type = c("Jolly", "Manly"),
  conf.level = 0.95, phi.full = TRUE)

## S3 method for class 'mrOpen'
summary(object, parm = c("N", "phi", "B", "M"),
  verbose = FALSE, ...)

## S3 method for class 'mrOpen'
confint(object, parm = c("N", "phi", "B"),
  level = NULL, conf.level = NULL, verbose = FALSE, ...)
```

Arguments

...	Additional arguments for methods.
mb.top	A matrix that contains the "top" of the Method B table (i.e., a contingency table of capture sample (columns) and last seen sample (rows)) or an object of class CapHist from capHistSum. See details.
mb.bot	A data frame that contains the "bottom" of the Method B table (i.e., the number of marked fish in the sample (m), the number of unmarked fish in the sample (u), the total number of fish in the sample (n), and the number of marked fish returned to the population following the sample (R)).
type	A string that indicates whether the large sample (normal theory) method of Jolly (type="Jolly") or the "arbitrary" method of Manly (type="Manly") should be used to construct confidence intervals.
conf.level	A single numeric that indicates the level of confidence to use for constructing confidence intervals (default is 0.95). See details.
phi.full	A logical that indicates whether the standard error for phi should include only sampling variability (phi.full=FALSE) or sampling and individual variability (phi.full=TRUE,default).
object	An object from mrOpen (i.e., of class mrOpen).
parm	A string that identifies the model parameters for which to return summaries or confidence intervals. By default, all parameters are returned.
verbose	A logical that indicates if the observables and other notes should be printed in summary and if the type of confidence interval used should be printed in confint. See details.
level	Same as conf.level but used for compatibility with generic confint function.

Details

jolly is just a convenience wrapper that produces the exact same results as mrOpen.

If mb.top contains an object from the capHistSum function then mb.bot can be left missing. In this case, the function will extract the needed data from the methodB.top and methodB.bot portions of the CapHist class object.

If mb.top is a matrix then it must be square, must have non-negative and no NA values in the upper triangle, and all NA values on the lower triangle and diagonal. If mb.bot is a matrix then it must have four rows named m, u, n, and R (see capHistSum for definitions), all values must be non-NA, and the first value of m must be 0. The last value of R can either be 0 or some positive number (it is ultimately ignored in all calculations).

All parameter estimates are performed using equations 4.6-4.9 from Pollock et al (1990) and from page 204 in Seber 2002. If type="Jolly" then all standard errors (square root of the variances) are from equations 4.11, 4.12, and 4.14 in Pollock et al. (1990) (these are different than those in Seber (2002) ... see Pollock et al.'s note on page 21). If type="Jolly" and phi.full=TRUE then the full variance for the phi parameter is given as in eqn 4.18 in Pollock et al. (1990), otherwise eqn 4.13 from Pollock et al. (1990) is used. When type="Jolly" the confidence interval are produced using normal theory (i.e., estimate +/- z*SE). If type="Manly" then the confidence intervals for N and phi (none will be produced for B) are constructed using the methods of Manly (1984) and as described in 2.24-2.33 of Krebs (1989). No standard errors are returned when type="Manly".

The summary function returns estimates of M, N, phi, B, and their associated standard errors and, if verbose=TRUE the intermediate calculations of "observables" from the data – n, m, R, r, and z.

The level of confidence is not set in the confint function, in contrast to most confint functions. Rather the confidence level is set in the main mrOpen function.

Value

A list with the following items:

- df A data frame that contains observable summaries from the data and estimates of the number of extant marked fish (M), population size for each possible sample period (N), apparent survival rate between each possible pair of sample periods (phi), and the number of additional individuals added to the population between each possible pair of sample periods (B). In addition to the estimates, values of the standard errors and the lower and upper confidence interval bounds for each parameter are provided (however, see the details above).
- type The provided type of confidence intervals that was used.
- phi.full The provided logical that indicates the type of standard error for phi that was used.
- conf.level The provided level of confidence that was used.

Testing

The formulas have been triple-checked against formulas in Pollock et al. (1990), Manly (1984), and Seber (2002).

The results for the CutthroatAL data file (as analyzed in the example) was compared to results from the JOLLY program available at http://www.mbr-pwrc.usgs.gov/software/jolly.html. The r and z values matched, all M and N estimates match at one decimal place, all phi are within 0.001, and all B are within 0.7. The SE match for M except for two estimates that are within 0.1, match for N except for one estimate that is within 0.1, are within 0.001 for phi, and are within 1.3 for B (except for for the first estimate which is dramatically off).

The results of mrOpen related to Table 4.4 of Pollock et al. (1990) match (to one decimal place) except for three estimates that are within 0.1% for N, match (to two decimal places) for phi except for where Pollock set phi>1 to phi=1, match for B except for Pollock set B<0 to B=0. The SE match (to two decimal places) for N except for N15 (which is within 0.5, <5%), match (to three decimal places) for phi except for phi15 (which is within 0.001, <0.5%), match (to two decimal places) for B except for B17 and B20 which are within 0.2 (<0.2%)

All point estimates of M, N, phi, and B and the SE of phi match the results in Table 2.3 of Krebs (1989) (within minimal rounding error for a very small number of results). The SE of N results are not close to those of Krebs (1989) (who does not provide a formula for SE so the discrepancy cannot be explored). The SE of B results match those of Krebs (1989) for 5 of the 8 values and are within 5% for 2 of the other 3 values (the last estimate is off by 27%).

For comparing to Jolly's data as presented in Tables 5.1 and 5.2 of Seber (2002), M was within 4 (less than 1.5%), N was within 3% (except N2 which was within 9%), phi was within 0.01 (less than 1.5

IFAR Chapter

9-Abundance from Capture-Recapture Data and 11-Mortality.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Jolly, G.M. 1965. Explicit estimates from capture-recapture data with both death and immigration – stochastic model. Biometrika, 52:225-247.

Krebs, C.J. 1989. Ecological Methodology. Harper & Row Publishers, New York.

Leslie, P.H. and D. Chitty. 1951. The estimation of population parameters from data obtained by means of the capture-recapture method. I. The maximum likelihood equations for estimating the death-rate. Biometrika, 38:269-292.

Manly, B.F.J. 1984. Obtaining confidence limits on parameters of the Jolly-Seber model for capture-recapture data. Biometrics, 40:749-758.

Pollock, K.H., J.D. Nichols, C. Brownie, and J.E. Hines. 1991. Statistical inference for capture-recapture experiments. Wildlife Monographs, 107:1-97.

Seber, G.A.F. 1965. A note on the multiple recapture census. Biometrika 52:249-259.

Seber, G.A.F. 2002. The Estimation of Animal Abundance. Edward Arnold, second edition (reprinted).

See Also

capHistSum, mrClosed

Examples

```
## First example -- capture histories summarized with capHistSum()
ch1 <- capHistSum(CutthroatAL,cols2use=-1)  # ignore first column of fish ID
ex1 <- mrOpen(ch1)
summary(ex1)
summary(ex1,verbose=TRUE)
summary(ex1,parm="N")
summary(ex1,parm=c("N","phi"))
confint(ex1)
confint(ex1,parm="N")
confint(ex1,parm=c("N","phi"))
confint(ex1,verbose=TRUE)

## Second example - Jolly's data -- summarized data entered "by hand"
s1 <- rep(NA,13)
s2 <- c(10,rep(NA,12))
s3 <- c(3,34,rep(NA,11))
s4 <- c(5,18,33,rep(NA,10))
s5 <- c(2,8,13,30,rep(NA,9))
s6 <- c(2,4,8,20,43,rep(NA,8))
s7 <- c(1,6,5,10,34,56,rep(NA,7))
s8 <- c(0,4,0,3,14,19,46,rep(NA,6))
s9 <- c(0,2,4,2,11,12,28,51,rep(NA,5))
s10 <- c(0,0,1,2,3,5,17,22,34,rep(NA,4))
```

```
s11 <- c(1,2,3,1,0,4,8,12,16,30,rep(NA,3))
s12 <- c(0,1,3,1,1,2,7,4,11,16,26,NA,NA)
s13 <- c(0,1,0,2,3,3,2,10,9,12,18,35,NA)
jolly.top <- cbind(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13)

n <- c(54,146,169,209,220,209,250,176,172,127,123,120,142)
R <- c(54,143,164,202,214,207,243,175,169,126,120,120,0)
m <- c(0,10,37,56,53,77,112,86,110,84,77,72,95)
u <- n-m
jolly.bot <- rbind(m,u,n,R)

ex2 <- mrOpen(jolly.top,jolly.bot)
summary(ex2,verbose=TRUE)
confint(ex2,verbose=TRUE)

ex3 <- mrOpen(jolly.top,jolly.bot,type="Manly")
summary(ex3,verbose=TRUE)
confint(ex3,verbose=TRUE)

## demonstrate use of jolly()
ex3a <- jolly(jolly.top,jolly.bot)
```

---

kCounts *Specific utilities for use in a knitr document.*

Description

Specific utilities for pretty printing various items in a knitr document.

Usage

```
kCounts(value, capitalize = FALSE)

kPvalue(value, digits = 4, include.p = TRUE, latex = TRUE)

purl2(file, out.dir = NULL, newname = NULL, topnotes = NULL,
  moreItems = NULL, blanks = c("extra", "all", "none"),
  delHeader = NULL, timestamp = TRUE, ...)

reproInfo(out = c("r", "markdown", "latex"), rqrdPkgs = NULL,
  elapsed = NULL, width = 0.95 * getOption("width"), addTOC = TRUE,
  newpage = FALSE, links = NULL, closeGraphics = TRUE, ind = 1)
```

Arguments

value	A single numeric count or p-value.
capitalize	A logical that indicates if the returned words should be capitalized or not (the default).

digits	Number of decimal places to round the values to.
include.p	A logical that indicates whether the result should be a character string with "p=" appended to the numerical result.
latex	A logical that indicates whether the resultant p-value string should be contained within dollar signs to form a latex formula.
file	A string that contains the root name of the .RNW file. This will also be the name of the resultant purled file with .R appended.
out.dir	A string that indicates the directory structure in which the purled file should be located. This should not have a forward slash at the end.
newname	A string for the output filename (without the extension) from purl2.
topnotes	A character vector of lines to be added to the top of the output file. Each value in the vector will be placed on a single line at the top of the output file.
moreItems	A string that contains additional words that when found in the purled file will result in the entire line with those words to be deleted.
blanks	A string that indicates if blank lines should be removed. If blanks="all" then all blank lines will be removed. If blanks="extra" then only "extra" blanks lines will be removed (i.e., one blank line will be left where there was originally more than one blank line).
delHeader	A single character that denotes the top and bottom of a block of lines that should be deleted from the script created by purl2.
timestamp	A logical that indicates whether a timestamp comment should be appended to the bottom of the script created by purl2.
...	Additional arguments for the original purl.
out	A string that indicates the type of output from reproInfo – Markdown, LaTeX, or simple R code.
rqrdPkgs	A string vector that contains packages that are required for the vignette and for which all dependencies should be found.
elapsed	A numeric, usually from proc.time, that is the time required to run the vignette. If NULL then this output will not be used. See the note below.
width	A numeric that indicates the width to use for wrapping the reproducibility information when out="r".
addTOC	A logical that indicates whether or not a table of contents entry for the reproducibility section should be added to the LaTeX output. Used only if Rout="latex"
newpage	A logical that indicates whether or not the reproducibility information should begin on a new page. Used only if Rout="latex"
links	A named character vector that will add a links bullet to the reproducibility information. The names will be shown and the values are the links. Used only if Rout="markdown.
closeGraphics	A logical that indicates whether the graphics device should be closed or not.
ind	An integer that indicates the CRAN mirror to use. Defaults to 1.

Details

- kCounts is used to convert numeric numbers to 'word' numbers in a sentence.
- kPvalue is used to print 'pretty' p-values.
- purl2 is used to create a modified (see below) Stangled or purled script.
- reproInfo is used to print 'reproducibility information' for the document.

Value

- kCounts returns a numeric value if the count is less than zero or greater than ten and returns a character string of the number 'name'. See the examples.
- kPvalue returns a character string of the supplied p-value rounded to the requested number of digits or a character string that indicates what the p-value is less than the value with a '5' in the digits+1 place. See the examples.
- purl2 is a modification of purl from **knitr** that creates a file with the same name as file but with lines removed that contain certain words (those found in ItemsToRemove and moreItems).
- reproInfo returns Markdown, LaTeX, or R code that prints "reproducibility information" at the bottom of the knitted document.

Note

In reproInfo, elapsed can be used to print the time it took to process the document by sending the elapsed time for processing to this argument. The simplest way to get an approximate elapsed time is to put `st <- proc.time()` very early (first line?) in your knitr code, put `et <- proc.time()-st` very late in your knitr code (i.e., just prior to reproInfo), and then used `elapsed=et["user.self"]+et["sys.self"]` in reproInfo.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See formatC for functionality similar to kPvalue. See purl and knit in **knitr** for functionality similar to purl2.

Examples

```
kCounts(7)
kCounts(17)
kCounts(0)
kCounts(-6)
kCounts(3,capitalize=TRUE)

kPvalue(0.123456789)
kPvalue(0.000123456)
kPvalue(0.000012345)
kPvalue(0.000012345,include.p=FALSE)
kPvalue(0.000012345,include.p=FALSE,latex=FALSE)
```

ksTest *Kolmogorov-Smirnov Tests.*

Description

Performs a one- or two-sample Kolmogorov-Smirnov test. Includes the option to perform the two-sample test using the formula notation.

Usage

```
ksTest(x, ...)

## Default S3 method:
ksTest(x, y, ..., alternative = c("two.sided", "less",
  "greater"), exact = NULL)

## S3 method for class 'formula'
ksTest(x, data = NULL, ...,
  alternative = c("two.sided", "less", "greater"), exact = NULL)
```

Arguments

x	A numeric vector of data values or a formula (see details).
...	Parameters of the distribution specified (as a character string) by y.
y	A numeric vector of data values, a character string naming a cumulative distribution function, or an actual cumulative distribution function. See ks.test.
alternative	A string that indicates the alternative hypothesis. See ks.test.
exact	NULL or a logical that indicates whether an exact p-value should be computed. See ks.test. Not available if ties are present, nor for the one-sided two-sample case.
data	A data frame that contains the variables in the formula for x.

Details

This is exactly ks.test except that a formula may be used for the two-sample situation. The default version is simply a pass through to ks.test. See ks.test for more details.

Value

See ks.test.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

ks.test.

Examples

```
## see ks.test for other examples
x <- rnorm(50)
y <- runif(30)
df <- data.frame(dat=c(x,y),grp=rep(c("X","Y"),c(50,30)))

## one-sample (from ks.test) still works
ksTest(x+2, "pgamma", 3, 2)
ks.test(x+2, "pgamma", 3, 2)

## first two-sample example in ?ks.test
ksTest(x,y)
ks.test(x,y)

## same as above but using data.frame and formula
ksTest(dat~grp,data=df)
```

---

lagratio *Ratio of lagged observations.*

Description

Computes the ratio of lagged observations in a vector.

Usage

```
lagratio(x, lag = 1L, recursion = 1L, differences = recursion,
  direction = c("backward", "forward"), ...)
```

Arguments

x	A numeric vector or matrix.
lag	An integer representing the lag 'distance'.
recursion	An integer that indicates the level of recursion for the calculations. A 1 will simply compute the ratios. A 2, for example, will compute the ratios, save the result, and then compute the ratios of the results using the same lag. See examples.
differences	Same as recursion. Used for symmetry with diff.
direction	A string that indicates the direction of calculation. A "backward" indicates that 'latter' values are divided by 'former' values. A "forward" indicates that 'former' values are divided by 'latter' values. See examples.
...	Additional arguments to diff().

Details

This function behaves similarly to `diff()` except that it returns a vector or matrix of ratios rather than differences.

Value

A vector or matrix of lagged ratios.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

`diff`

Examples

```
## Backward lagged ratios
# no recursion
lagratio(1:10,1)
lagratio(1:10,2)
# with recursion
lagratio(1:10,1,2)
lagratio(1:10,2,2)

## Forward lagged ratios
# no recursion
lagratio(10:1,1,direction="forward")
lagratio(10:1,2,direction="forward")
# with recursion
lagratio(10:1,1,2,direction="forward")
lagratio(10:1,2,2,direction="forward")
```

---

lencat *Constructs length class/category variable.*

Description

Constructs a vector that contains the length class or category to which an individual belongs. Optionally, that vector can be appended to the original data frame.

Usage

```
lencat(x, ...)

## Default S3 method:
lencat(x, w = 1, startcat = NULL, breaks = NULL,
  right = FALSE, use.names = FALSE, as.fact = use.names,
  droplevels = drop.levels, drop.levels = FALSE, ...)

## S3 method for class 'formula'
lencat(x, data, w = 1, startcat = NULL,
  breaks = NULL, right = FALSE, use.names = FALSE,
  as.fact = use.names, droplevels = drop.levels, drop.levels = FALSE,
  vname = NULL, ...)
```

Arguments

x	A numeric vector that contains the length measurements or a formula of the form ~x where "x" generically represents a variable in data that contains length measurements. This formula can only contain one variable.
...	Not implemented.
w	A single numeric that indicates the width of length categories to create. Ignored if breaks is not NULL.
startcat	A single numeric that indicates the beginning of the first length category. Only used with w. See details for how this is handled when NULL.
breaks	A numeric vector of lower values for the break points of the length categories.
right	A logical that indicates if the intervals should be closed on the right (and open on the left) or vice versa.
use.names	A logical that indicates whether the names for the values in breaks should be used for the levels in the new variable. Will throw a warning and then use default levels if TRUE but names(breaks) is NULL.
as.fact	A logical that indicates that the new variable should be returned as a factor (=TRUE) or not (=FALSE; default).
droplevels, drop.levels	A logical that indicates that the new variable should retain all levels indicated in breaks (=FALSE; default) or not. Ignored if as.fact=FALSE.
data	A data.frame that minimally contains the length measurements given in the variable in the formula.
vname	A string that contains the name for the new length class variable.

Details

If breaks is non-NULL, then w and startcat will be ignored. The vector of values in breaks should begin with a value smaller than the minimum observed value and end with a value larger than the maximum observed value. If the lowest break value is larger than the minimum observed value, then an error will occur. If the largest break value is smaller than the maximum observed

value, then an additional break value larger than the maximum observed value will be added to breaks (and a warning will be sent). The values in breaks do not have to be equally spaced.

If breaks=NULL (the default), then the value in w is used to create equally spaced categories. If startcat=NULL (the default), then the length categories will begin with the first value less than the minimum observed value "rounded" by w. For example, if the minimum observed value is 67, then the first length category will be 65 if w=5, 60 if w=10, 50 if w=25, and 50 if w=50. The length categories will continue from this starting value by values of w until a value greater than the largest observed value in x. The length categories are left-inclusive and right-exclusive by default (i.e., right=FALSE).

The start of the length categories may also be set with startcat. The number in the startcat argument should be less than the smallest value in x. Additionally, the number of decimals in startcat should not be more than the number of decimals in w (e.g., startcat=0.4 and w=1 will result in an error).

One may want to convert apparent numeric values to factor values if some of the length categories are missing (e.g., if factor values are used, for example, then tables of the length category values will have values for all length categories; i.e., it will have zeros for the length categories that are missing). The numeric values can be converted to factors by including as.fact. See the "real data" example.

The observed values in x should be rounded to the appropriate number of decimals to avoid misplacement of individuals into incorrect length categories due to issues with machine-precision (see discussion in all.equal.)

Value

If the formula version of the function is used, then a data.frame is returned with the a new variable, named as in vname (defaults to LCat), appended to the original data.frame. If the default version of the function is used, then a single vector is returned. The returned values will be numeric unless breaks is named and use.names=TRUE or if as.fact=TRUE.

IFAR Chapter

2-Data Manipulation.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Examples

```
# Create random lengths measured to nearest 0.1 unit
df1 <- data.frame(len=round(runif(50,0.1,9.9),1))

# Create length categories by 0.1 unit
df1$LCat1 <- lencat(df1$len,w=0.1)
xtabs(~LCat1,data=df1)
```

```
# length categories by 0.2 units
df1$LCat2 <- lencat(df1$len,w=0.2)
xtabs(~LCat2,data=df1)

# length categories by 0.2 units starting at 0.1
df1$LCat3 <- lencat(df1$len,w=0.2,startcat=0.1)
xtabs(~LCat3,data=df1)

# length categories as set by breaks
df1$LCat4 <- lencat(df1$len,breaks=c(0,2,4,7,10))
xtabs(~LCat4,data=df1)

## A Second example
# random lengths measured to nearest unit
df2 <- data.frame(len=round(runif(50,10,117),0))

# length categories by 5 units
df2$LCat1 <- lencat(df2$len,w=5)
xtabs(~LCat1,data=df2)

# length categories by 5 units starting at 7
df2$LCat2 <- lencat(df2$len,w=5,startcat=7)
xtabs(~LCat2,data=df2)

# length categories by 10 units
df2$LCat3 <- lencat(df2$len,w=10)
xtabs(~LCat3,data=df2)

# length categories by 10 units starting at 5
df2$LCat4 <- lencat(df2$len,w=10,startcat=5)
xtabs(~LCat4,data=df2)

# length categories as set by breaks
df2$LCat5 <- lencat(df2$len,breaks=c(5,50,75,150))
xtabs(~LCat5,data=df2)

## A Third example
# random lengths measured to nearest 0.1 unit
df3 <- data.frame(len=round(runif(50,10,117),1))

# length categories by 5 units
df3$LCat1 <- lencat(df3$len,w=5)
xtabs(~LCat1,data=df3)

## A Fourth example
# random lengths measured to nearest 0.01 unit
df4 <- data.frame(len=round(runif(50,0.1,9.9),2))

# length categories by 0.1 unit
df4$LCat1 <- lencat(df4$len,w=0.1)
xtabs(~LCat1,data=df4)
```

```
# length categories by 0.1 unit, but without missing categories
df4$LCat2 <- lencat(df4$len,w=0.1,as.fact=TRUE)
xtabs(~LCat2,data=df4)

# length categories by 2 unit
df4$LCat3 <- lencat(df4$len,w=2)
xtabs(~LCat3,data=df4)

## A Fifth example -- with real data
# remove variables with "anu" and "radcap" just for simplicity
smb1 <- smb2 <- SMBassWB[,-c(8:20)]

# 10 mm length classes - in default LCat variable
smb1$LCat10 <- lencat(smb1$lencap,w=10)
head(smb1)
xtabs(~LCat10,data=smb1)

# Same as previous but returned as a factor so that levels with no fish are still seen
smb1$LCat10A <- lencat(smb1$lencap,w=10,as.fact=TRUE)
head(smb1)
xtabs(~LCat10A,data=smb1)

# Same as previous but returned as a factor with unused levels dropped
smb1$LCat10B <- lencat(smb1$lencap,w=10,as.fact=TRUE,droplevels=TRUE)
head(smb1)
xtabs(~LCat10B,data=smb1)

# 25 mm length classes - in custom variable name
smb1$LCat25 <- lencat(smb1$lencap,w=25)
head(smb1)
xtabs(~LCat25,data=smb1)

# using values from psdVal for Smallmouth Bass
smb1$PSDCat1 <- lencat(smb1$lencap,breaks=psdVal("Smallmouth Bass"))
head(smb1)
xtabs(~PSDCat1,data=smb1)

# add category names
smb1$PSDCat2 <- lencat(smb1$lencap,breaks=psdVal("Smallmouth Bass"),use.names=TRUE)
head(smb1)
xtabs(~PSDCat2,data=smb1)

# same as above but drop the unused levels
smb1$PSDCat2A <- lencat(smb1$lencap,breaks=psdVal("Smallmouth Bass"),
                        use.names=TRUE,droplevels=TRUE)
head(smb1)
xtabs(~PSDCat2A,data=smb1)
str(smb1)

# same as above but not returned as a factor (returned as a character)
smb1$PSDcat2B <- lencat(smb1$lencap,breaks=psdVal("Smallmouth Bass"),
                        use.names=TRUE,as.fact=FALSE)
str(smb1)
```

```
## A Sixth example -- similar to the fifth example but using the formula notation
# 10 mm length classes - in default LCat variable
smb2 <- lencat(~lencap,data=smb2,w=10)
head(smb2)

# 25 mm length classes - in custom variable name
smb2 <- lencat(~lencap,data=smb2,w=25,vname="LenCat25")
head(smb2)

# using values from psdVal for Smallmouth Bass
smb2 <- lencat(~lencap,data=smb2,breaks=psdVal("Smallmouth Bass"),vname="LenPsd")
head(smb2)

# add category names
smb2 <- lencat(~lencap,data=smb2,breaks=psdVal("Smallmouth Bass"),vname="LenPsd2",
               use.names=TRUE,droplevels=TRUE)
head(smb2)
str(smb2)
```

`logbtcf` *Constructs the correction-factor used when back-transforming log-transformed values.*

Description

Constructs the correction-factor used when back-transforming log-transformed values according to Sprugel (1983). Sprugel's main formula – exp((syx^2)/2) – is used when syx is estimated for natural log transformed data. A correction for any base is obtained by multiplying the syx term by log_e(base) to give exp(((log_e(base)*syx)^2)/2). This more general formula is implemented here (if, of course, the base is exp(1) then the general formula reduces to the original specific formula).

Usage

```
logbtcf(obj, base = exp(1))
```

Arguments

`obj`	An object from `lm`.
`base`	A single numeric that indicates the base of the logarithm used.

Value

A numeric value that is the correction factor according to Sprugel (1983).

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Sprugel, D.G. 1983. Correcting for bias in log-transformed allometric equations. Ecology 64:209-210.

Examples

```
# toy data
df <- data.frame(y=rlnorm(10),x=rlnorm(10))
df$logey <- log(df$y)
df$log10y <- log10(df$y)
df$logex <- log(df$x)
df$log10x <- log10(df$x)

# model and predictions on loge scale
lme <- lm(logey~logex,data=df)
( ploge <- predict(lme,data.frame(logex=log(10))) )
( pe <- exp(ploge) )
( cfe <- logbtcf(lme) )
( cpe <- cfe*pe )

# model and predictions on log10 scale
lm10 <- lm(log10y~log10x,data=df)
plog10 <- predict(lm10,data.frame(log10x=log10(10)))
p10 <- 10^(plog10)
( cf10 <- logbtcf(lm10,10) )
( cp10 <- cf10*p10 )

# cfe and cf10, cpe and cp10 should be equal
all.equal(cfe,cf10)
all.equal(cpe,cp10)
```

`lwCompPreds` *Constructs plots of predicted weights at given lengths among different groups.*

Description

Constructs plots of predicted weights at given lengths among different groups. These plots allow the user to explore differences in predicted weights at a variety of lengths when the weight-length relationship is not the same across a variety of groups.

Usage

```
lwCompPreds(object, lens = NULL, qlens = c(0.05, 0.25, 0.5, 0.75,
  0.95), qlens.dec = 1, base = exp(1), interval = c("confidence",
  "prediction", "both"), center.value = 0, lwd = 1,
  connect.preds = TRUE, show.preds = FALSE, col.connect = "gray50",
  ylim = NULL, main.pre = "Length==", cex.main = 0.8,
```

```
  xlab = "Groups", ylab = "Predicted Weight", yaxs = "r",
  rows = round(sqrt(num)), cols = ceiling(sqrt(num)))
```

Arguments

object	An lm object (i.e., returned from fitting a model with lm). This model should have log(weight) as the response and log(length) as the explanatory covariate and an explanatory factor variable that describes the different groups.
lens	A numeric vector that indicates the lengths at which the weights should be predicted.
qlens	A numeric vector that indicates the quantiles of lengths at which weights should be predicted. This is ignored if lens is non-null.
qlens.dec	A single numeric that identifies the decimal place that the lengths derived from qlens should be rounded to (Default is 1).
base	A single positive numeric value that indicates the base of the logarithm used in the lm object in object. The default is exp(1), or the value e.
interval	A single string that indicates whether to plot confidence (="confidence"), prediction (="prediction"), or both (="both") intervals.
center.value	A single numeric value that indicates the log length used if the log length data was centered when constructing object.
lwd	A single numeric that indicates the line width to be used for the confidence and prediction interval lines (if not interval="both") and the prediction connections line. If interval="both" then the width of the prediction interval will be one less than this value so that the CI and PI appear different.
connect.preds	A logical that indicates whether the predicted values should be connected with a line across groups or not.
show.preds	A logical that indicates whether the predicted values should be plotted with a point for each group or not.
col.connect	A color to use for the line that connects the predicted values (if connect.preds=TRUE).
ylim	A numeric vector of length two that indicates the limits of the y-axis to be used for each plot. If null then limits will be chosen for each graph individually.
main.pre	A character string to be used as a prefix for the main title. See details.
cex.main	A numeric value for the character expansion of the main title. See details.
xlab	A single string for labeling the x-axis.
ylab	A single string for labeling the y-axis.
yaxs	A single string that indicates how the y-axis is formed. See par for more details.
rows	A single numeric that contains the number of rows to use on the graphic.
cols	A single numeric that contains the number of columns to use on the graphic.
...	Other arguments to pass through to the plot function.

Value

None. However, a plot is produced.

IFAR Chapter

7-Weight-Length.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Examples

```
# add log length and weight data to ChinookArg data
ChinookArg$logtl <- log(ChinookArg$tl)
ChinookArg$logwt <- log(ChinookArg$w)
# fit model to assess equality of slopes
lm1 <- lm(logwt~logtl*loc,data=ChinookArg)
anova(lm1)

# set graphing parameters so that the plots will look decent
op <- par(mar=c(3.5,3.5,1,1),mgp=c(1.8,0.4,0),tcl=-0.2)
# show predicted weights (w/ CI) at the default quantile lengths for each year
lwCompPreds(lm1,xlab="Location")
# show predicted weights (w/ CI) at the quartile lengths for each year
lwCompPreds(lm1,xlab="Location",qlens=c(0.25,0.5,0.75))
# show predicted weights (w/ CI) at certain lengths for each year
lwCompPreds(lm1,xlab="Location",lens=c(60,90,120,150))
# show predicted weights (w/ just PI) at certain lengths for each year
lwCompPreds(lm1,xlab="Location",lens=c(60,90,120,150),interval="prediction")
# show predicted weights (w/ CI and PI) at certain lengths for each year
lwCompPreds(lm1,xlab="Location",lens=c(60,90,120,150),interval="both")
# show predicted weights (w/ CI and points at the prediction) at certain lengths for each year
lwCompPreds(lm1,xlab="Location",lens=c(60,90,120,150),show.preds=TRUE)
# show predicted weights (w/ CI but don't connect means) at certain lengths for each year
lwCompPreds(lm1,xlab="Location",lens=c(60,90,120,150),connect.preds=FALSE,show.preds=TRUE)

# fit model with centered data
mn.logtl <- mean(ChinookArg$logtl,na.rm=TRUE)
ChinookArg$clogtl <- ChinookArg$logtl-mn.logtl
lm2 <- lm(logwt~clogtl*loc,data=ChinookArg)
lwCompPreds(lm2,xlab="Location",center.value=mn.logtl)
lwCompPreds(lm2,xlab="Location",lens=c(60,90,120,150),center.value=mn.logtl)

# fit model with a different base (plot should be the same as the first example)
ChinookArg$logtl <- log10(ChinookArg$tl)
ChinookArg$logwt <- log10(ChinookArg$w)
lm1 <- lm(logwt~logtl*loc,data=ChinookArg)
lwCompPreds(lm1,base=10,xlab="Location")

if (interactive()) {
  # should give error, does not work for only a simple linear regression
```

```
  lm2 <- lm(logwt~logtl,data=ChinookArg)
  lwCompPreds(lm2)
  # or a one-way ANOVA
  lm3 <- lm(logwt~loc,data=ChinookArg)
  lwCompPreds(lm3)
}

## return graphing parameters to original state
par(op)
```

`mapvalues` *Replace specified values in a vector or factor with new values.*

Description

Same function as in **plyr**. Imported by **FSA** and exported to minimize conflicts with other commonly used packages. See `mapvalues` for documentation.

`Mirex` *Mirex concentration, weight, capture year, and species of Lake Ontario salmon.*

Description

Mirex concentration, weight, capture year, and species of Lake Ontario Coho and Chinook salmon.

Format

A data frame with 122 observations on the following 4 variables.

year a numeric vector of capture years

weight a numeric vector of salmon weights (kg)

mirex a numeric vector of mirex concentration in the salmon tissue (mg/kg)

species a factor with levels `chinook` and `coho`

Details

The `year` variable should be converted to a factor as shown in the example.

Topic(s)

- Linear models
- Other

Source

From (actual data) Makarewicz, J.C., E.Damaske, T.W. Lewis, and M. Merner. 2003. Trend analysis reveals a recent reduction in mirex concentrations in coho (*Oncorhynchus kisutch*) and chinook (*O. tshawytscha*) salmon from Lake Ontario. Environmental Science and Technology, 37:1521-1527.

See Also

Used in `fitPlot`, `residPlot`, `compSlopes`, `compIntercepts`, `hoCoef`, and `rSquared` examples.

Examples

```
Mirex$year <- factor(Mirex$year)
lm1 <- lm(mirex~weight*year*species,data=Mirex)
anova(lm1)
```

Mmethods *Estimate natural mortality from a variety of empirical methods.*

Description

Several methods can be used to estimated natural mortality (M) from other types of data, including parameters from the von Bertalanffy growth equation, maximum age, and temperature. These relationships have been developed from meta-analyses of a large number of populations. Several of these methods are implemented in this function.

Usage

```
Mmethods(what = c("all", "tmax", "K", "Hoenig", "Pauly"))

metaM(method = Mmethods(), justM = TRUE, tmax = NULL, K = NULL,
  Linf = NULL, t0 = NULL, b = NULL, L = NULL, Temp = NULL,
  t50 = NULL, Winf = NULL)

## S3 method for class 'metaM'
print(x, digits = 4, ...)
```

Arguments

`what`	A string that indicates what grouping of methods to return. Defaults to returning all methods.
`method`	A string that indicates which method or equation to use. See details.
`justM`	A logical that indicates whether just the estimate of M (`TRUE`; Default) or a more descriptive list should be returned.
`tmax`	The maximum age for the population of fish.
`K`	The Brody growth coefficient from the fit of the von Bertalanffy growth function.

Linf	The asymptotic mean length (cm) from the fit of the von Bertalanffy growth function.
t0	The x-intercept from the fit of the von Bertalanffy growth function.
b	The exponent from the weight-length relationship (slope from the logW-logL relationship).
L	The body length of the fish (cm).
Temp	The temperature experienced by the fish (C).
t50	The age (time) when half the fish in the population are mature.
Winf	The asymptotic mean weight (g) from the fit of the von Bertalanffy growth function.
x	A metaM object returned from metaM when justM=FALSE.
digits	A numeric that controls the number of digits printed for the estimate of M.
...	Additional arguments for methods. Not implemented.

Details

One of several methods is chosen with method. The available methods can be seen with Mmethods() and are listed below with a brief description of where the equation came from. The sources (listed below) should be consulted for more specific information.

- method="HoenigNLS": The "modified Hoenig equation derived with a non-linear model" as described in Then et al. (2015) on the third line of Table 3. This method was the preferred method suggested by Then et al. (2015). Requires only tmax.
- method="PaulyLNoT": The "modified Pauly length equation" as described on the sixth line of Table 3 in Then et al. (2015). Then et al. (2015) suggested that this is the preferred model if maximum age (tmax) information was not available. Requires K and Linf.
- method="PaulyL": The "Pauly (1980) equation using fish lengths" from his equation 11. This is the most commonly used method in the literature. Note that Pauly used common logarithms as used here but the model is often presented in other sources with natural logarithms. Requires K, Linf, and T.
- method="PaulyW": The "Pauly (1980) equation for weights" from his equation 10. Requires K, Winf, and T.
- method="HoeingO", method="HoeingOF", method="HoeingOM", method="HoeingOC": The original "Hoenig (1983) composite", "fish", "mollusc", and "cetacean" (fit with OLS) equations from the second column on page 899 of Hoenig (1983). Requires only tmax.
- method="HoeingO2", method="HoeingO2F", method="HoeingO2M", method="HoeingO2C": The original "Hoenig (1983) composite", "fish", "mollusc", and "cetacean" (fit with Geometric Mean Regression) equations from the second column on page 537 of Kenchington (2014). Requires only tmax.
- method="HoenigLM": The "modified Hoenig equation derived with a linear model" as described in Then et al. (2015) on the second line of Table 3. Requires only tmax.
- method="HewittHoenig": The "Hewitt and Hoenig (2005) equation" from their equation 8. Requires only tmax.

- method="tmax1": The "one-parameter tmax equation" from the first line of Table 3 in Then et al. (2015). Requires only tmax.
- method="K1": The "one-parameter K equation" from the fourth line of Table 3 in Then et al. (2015). Requires only K.
- method="K2": The "two-parameter K equation" from the fifth line of Table 3 in Then et al. (2015). Requires only K.
- method="JensenK1": The "Jensen (1996) one-parameter K equation". Requires only K.
- method="JensenK2": The "Jensen (2001) two-parameter K equation" from their equation 8. Requires only K.
- method="Gislason": The "Gislason et al. (2010) equation" from their equation 2. Requires K, Linf, and L.
- method="AlversonCarney": The "Alverson and Carney (1975) equation" as given in equation 10 of Zhang and Megrey (2006). Requires tmax and K.
- method="Charnov": The "Charnov et al. (2013) equation" as given in the second column of page 545 of Kenchington (2014). Requires K, Linf, and L.
- method="ZhangMegreyD", method="ZhangMegreyP": The "Zhang and Megrey (2006) equation" as given in their equation 8 but modified for demersal or pelagic fish. Thus, the user must choose the fish type with group. Requires tmax, K, t0, t50, and b.
- method="RikhterEfanov1": The "Rikhter and Efanov (1976) equation (#2)" as given in the second column of page 541 of Kenchington (2014) and in Table 6.4 of Miranda and Bettoli (2007). Requires only t50.
- method="RikhterEfanov2": The "Rikhter and Efanov (1976) equation (#1)" as given in the first column of page 541 of Kenchington (2014). Requires t50, K, t0, and b.

Value

Mmethods returns a character vector with a list of methods. If only one method is chosen then metaM returns a single numeric if justM=TRUE or, otherwise, a metaM object that is a list with the following items:

- method: The name for the method within the function (as given in method).
- name: A more descriptive name for the method.
- givens: A vector of values required by the method to estimate M.
- M: The estimated natural mortality rate.

If multiple methods are chosen then a data.frame is returned with the method name abbreviation in the method variable and the associated estimated M in the M variable.

Testing

Kenchington (2014) provided life history parameters for several stocks and used many models to estimate M. I checked the calculations for the PaulyL, PaulyW, HoenigO for Hgroup="all" and Hgroup="fish", HoenigO2 for Hgroup="all" and Hgroup="fish", "JensenK1", "Gislason", "AlversonCarney", "Charnov", "ZhangMegrey", "RikhterEfanov1", and "RikhterEfanov2" methods for three stocks. All results perfectly matched Kenchington's results for Chesapeake Bay

Anchovy and Rio Formosa Seahorse. For the Norwegian Fjord Lanternfish, all results perfectly matched Kenchington's results except for when `Hgroup="fish"` for both `HoenigO` and `HoenigO2`.

Results for the Rio Formosa Seahorse data were also tested against results from `M.empirical` from **fishmethods** for the `PaulyL`, `PaulyW`, `HoenigO` for `Hgroup="all"` and `Hgroup="fish"`, `"Gislason"`, and `"AlversonCarney"` methods (the only methods in common between the two packages). All results matched perfectly.

IFAR Chapter

11-Mortality.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Alverson, D.L. and M.J. Carney. 1975. A graphic review of the growth and decay of population cohorts. Journal du Conseil International pour l'Exploration de la Mer. 36:133-143.

Charnov, E.L., H. Gislason, and J.G. Pope. 2013. Evolutionary assembly rules for fish life histories. Fish and Fisheries. 14:213-224.

Gislason, H., N. Daan, J.C. Rice, and J.G. Pope. 2010. Size, growth, temperature and the natural mortality of marine fish. Fish and Fisheries 11:149-158.

Hewitt, D.A. and J.M. Hoenig. 2005. Comparison of two approaches for estimating natural mortality based on longevity. Fishery Bulletin. 103:433-437. [Was (is?) from http://fishbull.noaa.gov/1032/hewitt.pdf.]

Hoenig, J.M. 1983. Empirical use of longevity data to estimate mortality rates. Fishery Bulletin. 82:898-903. [Was (is?) from http://www.afsc.noaa.gov/REFM/age/Docs/Hoenig_EmpiricalUseOfLongevityData.pdf.]

Jensen, A.L. 1996. Beverton and Holt life history invariants result from optimal trade-off of reproduction and survival. Canadian Journal of Fisheries and Aquatic Sciences. 53:820-822. [Was (is?) from .]

Jensen, A.L. 2001. Comparison of theoretical derivations, simple linear regressions, multiple linear regression and principal components for analysis of fish mortality, growth and environmental temperature data. Environometrics. 12:591-598. [Was (is?) from http://deepblue.lib.umich.edu/bitstream/handle/2027.42/35236/

Kenchington, T.J. 2014. Natural mortality estimators for information-limited fisheries. Fish and Fisheries. 14:533-562.

Pauly, D. 1980. On the interrelationships between natural mortality, growth parameters, and mean environmental temperature in 175 fish stocks. Journal du Conseil International pour l'Exploration de la Mer. 39:175-192. [Was (is?) from http://innri.unuftp.is/pauly/On%20the%20interrelationships%20betwe.pdf.]

Rikhter, V.A., and V.N. Efanov. 1976. On one of the approaches for estimating natural mortality in fish populations (in Russian). ICNAF Research Document 76/IV/8, 12pp.

Then, A.Y., J.M. Hoenig, N.G. Hall, and D.A. Hewitt. 2015. Evaluating the predictive performance of empirical estimators of natural mortality rate using information on over 200 fish species. ICES Journal of Marine Science. 72:82-92.

Zhang, C-I and B.A. Megrey. 2006. A revised Alverson and Carney model for estimating the instantaneous rate of natural mortality. Transactions of the American Fisheries Society. 135-620-633. [Was (is?) from http://www.pmel.noaa.gov/foci/publications/2006/zhan0531.pdf.]

See Also

See `M.empirical` in **fishmethods** for similar functionality.

Examples

```
## List names for available methods
Mmethods()
Mmethods("tmax")

## Simple Examples
metaM("tmax",tmax=20)
metaM("tmax",tmax=20,justM=FALSE)
metaM("HoenigNLS",tmax=20)
metaM("HoenigNLS",tmax=20,justM=FALSE)

## Example Patagonian Sprat ... from Table 2 in Cerna et al. (2014)
## http://www.scielo.cl/pdf/lajar/v42n3/art15.pdf
Temp <- 11
Linf <- 17.71
K <- 0.78
t0 <- -0.46
tmax <- t0+3/K
t50 <- t0-(1/K)*log(1-13.5/Linf)
metaM("RikhterEfanov1",t50=t50)
metaM("PaulyL",K=K,Linf=Linf,Temp=Temp)
metaM("PaulyL",K=K,Linf=Linf,Temp=Temp,justM=FALSE)
metaM("HoenigNLS",tmax=tmax)
metaM("HoenigO",tmax=tmax)
metaM("HewittHoenig",tmax=tmax)
metaM("AlversonCarney",K=K,tmax=tmax)

## Example of multiple calculations
metaM(c("RikhterEfanov1","PaulyL","HoenigO","HewittHoenig","AlversonCarney"),
     K=K,Linf=Linf,Temp=Temp,tmax=tmax,t50=t50)

## Example of multiple methods using Mmethods
# select some methods
metaM(Mmethods()[-c(15,20,22:24,26)],K=K,Linf=Linf,Temp=Temp,tmax=tmax,t50=t50)
# select just the Hoenig methods
metaM(Mmethods("Hoenig"),K=K,Linf=Linf,Temp=Temp,tmax=tmax,t50=t50)
```

`mrClosed` *Estimate initial population size for single or multiple census mark-recapture data.*

Description

Estimates of the initial population size, along with associated confidence intervals, are constructed from single or multiple census mark-recapture data using a variety of methods. For single census data, the initial population size (N) is estimated from the number of marked animals from a first sample (M), number of captured animals in a second sample (n), and the number of recaptured marked animals in the second sample (m) using either the 'naive' Petersen method or Chapman, Ricker, or Bailey modifications of the Petersen method. Single census data can also be separated by group (e.g., size class) to estimate the initial population size by class and for the overall population size. For multiple census data, the initial population size is estimated from the number of captured animals (n), number of recaptured marked animals (m), the number of marked animals that are marked and returned to the population (R), or the number of extant marked animals prior to the sample (M) on each of several samples using either the Schnabel (1938) or Schumacher-Eschmeyer (1943) method.

Usage

```
mrClosed(M = NULL, n = NULL, m = NULL, R = NULL,
  method = c("Petersen", "Chapman", "Ricker", "Bailey", "Schnabel",
  "SchumacherEschmeyer"), labels = NULL, chapman.mod = TRUE)

## S3 method for class 'mrClosed1'
summary(object, digits = 0, incl.SE = FALSE,
  incl.all = TRUE, verbose = FALSE, ...)

## S3 method for class 'mrClosed1'
confint(object, parm = NULL, level = conf.level,
  conf.level = 0.95, digits = 0, type = c("suggested", "binomial",
  "hypergeometric", "normal", "Poisson"), bin.type = c("wilson", "exact",
  "asymptotic"), poi.type = c("exact", "daly", "byar", "asymptotic"),
  incl.all = TRUE, verbose = FALSE, ...)

## S3 method for class 'mrClosed2'
summary(object, digits = 0, verbose = FALSE, ...)

## S3 method for class 'mrClosed2'
confint(object, parm = NULL, level = conf.level,
  conf.level = 0.95, digits = 0, type = c("suggested", "normal",
  "Poisson"), poi.type = c("exact", "daly", "byar", "asymptotic"),
  verbose = FALSE, ...)

## S3 method for class 'mrClosed2'
plot(x, pch = 19, col.pt = "black",
  xlab = "Marked in Population", ylab = "Prop. Recaptures in Sample",
  loess = FALSE, lty.loess = 2, lwd.loess = 1,
  col.loess = "gray20", trans.loess = 10, span = 0.9, ...)
```

Arguments

M	A numeric representing the number of marked fish from the first sample (single-census), an object from capHistSum() (single- or multiple-census), or numeric vector of marked fish prior to ith samples (multiple-census).
n	A numeric representing the number of captured fish in the second sample (single-census) or numeric vector of captured fish in ith sample (multiple-census).
m	A numeric representing the number of recaptured (marked) fish in the second sample (single-census) or numeric vector of recaptured (marked) fish in ith sample (multiple-census).
R	A numeric vector representing the number of marked fish returned to the population (multiple-census). Note that several references use the number of "new" marks returned to the population rather than the "total" number of marks returned to the population that is used here.
method	A single string that identifies the type of calculation method to use in the main function.
labels	A character or character vector used to label the rows of the resulting output matrix when using a single census method separated by groups. Must be the same length as M, n, and m. Defaults to upper-case letters if no values are given.
chapman.mod	A logical that represents whether the Chapman modification should be used (=TRUE, default) or not (=FALSE) when performing the Schnabel multiple census method.
object, x	An mrClosed1 or mrClosed2 object.
digits	The number of decimal digits to round the population estimates to. If incl.SE=TRUE then SE will be rounded to one more decimal place then given in digits.
incl.SE	A logical that indicates whether the results should include the calculated SE value. See details.
incl.all	A logical that indicates whether an overall population estimate should be computed when using a single census method that has been separated into subgroups. See details.
verbose	A logical that indicates whether a reminder of the inputted values and what type of method was used should be printed with the summary and confidence interval results.
...	Additional arguments for methods.
parm	Not used here (included in confint generic).
level	Same as conf.level but used for compatibility with confint generic.
conf.level	A numeric representing the level of confidence to use for confidence intervals.
type	A single string that identifies the distribution to use when constructing confidence intervals in confint. See details.
bin.type	A string that identifies the method used to construct binomial confidence intervals (default is "wilson"). This is only used if type="binomial" in confint. See details of binCI.

poi.type	A string that identifies the method used to construct Poisson confidence intervals (default is "exact"). This is only used if type="Poisson" in confint. See details of poiCI.
pch	A numeric used to indicate the type of plotting character.
col.pt	a string used to indicate the color of the plotted points.
xlab	A label for the x-axis.
ylab	A label for the y-axis.
loess	A logical that indicates if a loess smoother line (and approximate 95% confidence band) is fit to and shown on plot.
lty.loess	A single numeric used to indicate the type of line used for the loess line.
lwd.loess	A single numeric used to indicate the line width of the loess line.
col.loess	A single string used to indicate the color of the loess line.
trans.loess	A single numeric that indicates how transparent the loess band should be (larger numbers are more transparent).
span	A single numeric that controls the degree of smoothing. Values closer to 1 are more smooth.

Details

For single census data, the following methods can be used:

- method="Petersen". The 'naive' Petersen as computed using equation 2.1 from Krebs (1989).
- method="Chapman". The Chapman (1951) modification of the Petersen method as computed using equation 2.2 from Krebs (1989).
- method="Ricker". The Ricker (1975) modification of the Petersen as computed using equation 3.7 from Ricker (1975). This is basically the same method="Chapman" except that Ricker (1975) did NOT subtract a 1 from the answer in the final step. Thus, the estimate from method="Chapman" will always be one less than the estimate from method="Ricker".
- method="Bailey". The Bailey (1951, 1952) modification of the Petersen as computed using equation 2.3 from Krebs (1989).

If M contains an object from capHistSum and one of Petersen, Chapman, Ricker, or Bailey methods has been selected with method= then n= and m= can be left missing or will be ignored and the needed data will be extracted from the sum portion of the CapHist class object. If the data were not summarized with capHistSum then all of M=, n=, and m= must be supplied by the user.

The population estimate (as computed with the formulas noted in the table above) is extracted with summary. In addition, the standard error of the population estimate (SE) can be extracted by including incl.SE=TRUE. The SE is from equation 3.6 (p. 78) in Ricker (1975) for the Petersen method, from p. 60 (near bottom) of Seber (2002) for the Chapman method, from p. 61 (middle) of Seber (2002) (and as noted on p. 79 of Ricker (1975)) for the Bailey method, and from equation 3.8 (p. 78) in Ricker (1975) for the Ricker method.

Confidence intervals for the initial population size from the single census methods can be constructed using four different distributions as chosen with type= in confint. If type="suggested" then the type of confidence interval suggested by the rules on p. 18 in Krebs (1989) are used. The general methods for constructing confidence intervals for N are described below

- type="hypergeometric". Uses hyperCI. This is experimental at this point.
- type="binomial". Use binCI to construct a confidence interval for m/n (Petersen method) or (m+1)/(n+1) (Chapman, Bailey, Ricker methods), divides M or (M+1) by the CI endpoints, and subtract 1 (for the Chapman method).
- type="Poisson". Use poiCI to construct a confidence interval for m (Petersen method) or (m+1) (Chapman, Bailey, Ricker methods), substitute the CI endpoints into the appropriate equation for estimating N, and subtract 1 (for the Chapman method).
- type="normal". Used equation 2.4 (p.20) from Krebs (2002) for the Petersen method. For the other methods, used N+/- Z(0.975)*SE, where the SE was computed as noted above.

If incl.all=TRUE in summary and population estimates have been constructed for multiple sub-groups then an overall population estimate is included by summing the population estimates for the multiple sub-groups. If incl.SE=TRUE, then an overall SE is computed by taking the square root of the summed VARIANCES for the multiple sub-groups.

For multiple census data, the following methods can be declared for use with the method= argument:

- method="Schnabel". The Schnabel (1938) method as computed with equation 3.15 from Ricker (1975).
- method="SchumacherEschmeyer". The Schumacher and Eschmeyer (1943) method as computed with equation 3.12 from Ricker (1975) eqn 3.12.

If M contains an object from capHistSum and the Schnabel or Schumacher-Eschmeyer methods has been chosen then n, m and R can be left missing or will be ignored. In this case, the needed data is extracted from the sum portion of the CapHist class object. Otherwise, the user must supply vectors of results in n, m, and R or M.

The population estimate for each method is extracted with summary. Standard errors for the population estimate can NOT be computed for the Schnabel or Schumacher-Eschmeyer methods (a warning will be produced if incl.SE=TRUE is used).

Confidence intervals for the initial population size using multiple census methods can be constructed using the normal or Poisson distributions for the Schnabel method or the normal distribution for the Schumacher-Eschmeyer method as chosen with type=. If type="suggested" then the type of confidence interval suggested by the rule on p. 32 of Krebs (1989) is used (for the Schnabel method). If type="Poisson" for the Schnabel method then a confidence interval for the sum of m is computed with poiCI and the end points are substituted into the Schnabel equation to produce a CI for the population size. If type="normal" for the Schnabel method then the standard error for the *inverse* of the population estimate is computed as the square root of equation 2.11 from Krebs (1989) or equation 3.16 from Ricker (1975). The standard error for the Schumacher-Eschmeyer method is for the *inverse* of the population estimate and is computed with equation 2.14 from Krebs (1989) [Note that the divisor in Krebs (1989) is different than the divisor in equation 3.12 in Ricker (1975), but is consistent with equation 4.17 in Seber (2002).] The confidence interval for the *inverse* population estimate is constructed from the inverse population estimate plus/minus a t critical value times the standard error for the inverse population estimate. The t critical value uses the number of samples minus 1 for the Schnabel method and the number of samples minus 2 when for the Schumacher-Eschmeyer method according to p. 32 of Krebs (1989) (note that this is different than what Ricker (1975) does). Finally, the confidence interval for the population estimate is obtained by inverting the confidence interval for the inverse population estimate. Note that confidence intervals for the population size when type="normal" may contain negative values (for the upper value)

when the population estimate is relatively large and the number of samples is small (say, three) because the intervals are originally constructed on the inverted population estimate and they use the t-distribution.

The `plot` can be used to identify assumption violations in the Schnabel and Schumacher-Eschmeyer methods (an error will be returned if used with any of the other methods). If the assumptions ARE met then the plot of the proportion of marked fish in a sample versus the cumulative number of marked fish should look linear. A loess line (with approximate 95% confidence bands) can be added to aid interpretation with `loess=TRUE`. Note, however, that adding the loess line may return a number of warning or produce a non-informative if the number of samples is small (<8).

Value

A list with the following items

- M The number of marked fish from the first sample that was provided.
- n The number of captured fish in the second sample that was provided.
- m The number of recaptured (marked) fish in the second sample that was provided.
- M1 The adjusted (depending on `type`) number of marked fish from the first sample.
- n1 The adjusted (depending on `type`) number of captured fish in the second sample.
- m1 The adjusted (depending on `type`) number of recaptured (marked) fish in the second sample.
- cf A correction factor for the population estimate that depends on `type`.
- method The type of method used (provided by the user).
- methodLbl A label for the type of method used.
- N The estimated initial population size.
- labels Labels for the rows of summary matrix.

Testing

The results from the single census methods have had the following checks. The population estimates for all methods match reputable sources. The SE for the Chapman and Bailey methods match the results from `mrN.single` in **fishmethods**, The CI for the Petersen, Chapman, and Bailey methods partially match (are within 1

The results for the multiple census methods have had the following checks. The population estimates for both methods match reputable sources. The intermediate calculations for both methods match those in Krebs (1989). The confidence interval for the Schnabel method using the Poisson distribution does NOT match Krebs (1989). This appears to be a difference in the use `poiCI` here versus distributional tables in Krebs (i.e., the difference appears to be completely in the critical values from the Poisson distribution). The confidence interval for the Schnabel method using the normal or the Poisson distribution do NOT match Ricker (1975), but there is not enough information in Ricker to determine why (it is likely due to numerical differences on the inverse scale). The confidence interval for the Schumacher-Eschmeyer method do match Krebs (1989) but not Ricker (1975). The Ricker result may be due to different df as noted above.

IFAR Chapter

9-Abundance from Capture-Recapture Data.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Krebs, C.J. 1989. Ecological Methodology. Addison-Welsey Educational Publishing.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Seber, G.A.F. 2002. The Estimation of Animal Abundance and Related Parameters. Edward Arnold, second edition.

Schnabel, Z.E. 1938. The estimation of the total fish population of a lake. American Mathematician Monthly, 45:348-352.

Schumacher, F.X. and R.W. Eschmeyer. 1943. The estimation of fish populations in lakes and ponds. Journal of the Tennessee Academy of Sciences, 18:228-249.

See Also

See `capHistSum` for generating input data from capture histories. See `poiCI`, `binCI`, and `hyperCI` for specifics on functions used in confidence interval construction. See `mrOpen` for handling mark-recapture data in an open population. See `SunfishIN` in **FSAdata** for an example to test matching of results with Ricker (1975)' See `mrN.single` and `schnabel` in **fishmethods** for similar functionality.

Examples

```
### Single census with no sub-groups
## Petersen estimate -- the default
mr1 <- mrClosed(346,184,49)
summary(mr1)
summary(mr1,verbose=TRUE)
summary(mr1,incl.SE=TRUE)
summary(mr1,incl.SE=TRUE,digits=1)
confint(mr1)
confint(mr1,verbose=TRUE)
confint(mr1,type="hypergeometric")

## Chapman modification of the Petersen estimate
mr2 <- mrClosed(346,184,49,method="Chapman")
summary(mr2,incl.SE=TRUE)
summary(mr2,incl.SE=TRUE,verbose=TRUE)

### Single census, using capHistSum() results
## data in capture history format
str(BluegillJL)
ch1 <- capHistSum(BluegillJL)
mr3 <- mrClosed(ch1)
summary(mr3,verbose=TRUE)
confint(mr3,verbose=TRUE)
```

```
### Single census with sub-groups
marked <- c(93,35,72,16,46,20)
captured <- c(103,30,73,17,39,18)
recaps <- c(20,23,52,15,35,16)
lbls <- c("YOY","Juvenile","Stock","Quality","Preferred","Memorable")
mr4 <- mrClosed(marked,captured,recaps,method="Ricker",labels=lbls)
summary(mr4)
summary(mr4,incl.SE=TRUE)
summary(mr4,incl.SE=TRUE,verbose=TRUE)
summary(mr4,incl.SE=TRUE,incl.all=FALSE,verbose=TRUE)
confint(mr4)
confint(mr4,verbose=TRUE)
confint(mr4,incl.all=FALSE,verbose=TRUE)

### Multiple Census
## Data in summarized form ... Schnabel method
mr5 <- with(PikeNY,mrClosed(n=n,m=m,R=R,method="Schnabel"))
plot(mr5)
plot(mr5,loess=TRUE)
summary(mr5)
summary(mr5,verbose=TRUE)
confint(mr5)
confint(mr5,verbose=TRUE)

## Schumacher-Eschmeyer method
mr6 <- with(PikeNY,mrClosed(n=n,m=m,R=R,method="Schumacher"))
summary(mr6)
confint(mr6)

### Capture history data summarized by capHistSum()
# ignore first column of ID numbers
ch2 <- capHistSum(PikeNYPartial1,cols2ignore="id")

## Schnabel method
mr7 <- mrClosed(ch2,method="Schnabel")
plot(mr7)
summary(mr7)
confint(mr7)
```

`nlsTracePlot` *Adds model fits from nls iterations to active plot.*

Description

Adds model fits from iterations of the `nls` algorithm as returned when `trace=TRUE`. Useful for diagnosing model fitting problems or issues associated with starting values.

Usage

```
nlsTracePlot(object, fun, from = NULL, to = NULL, n = 199, lwd = 2,
  pal = paletteChoices(), rev.col = FALSE, legend = "topright",
  cex.leg = 0.9, box.lty.leg = 0, add = TRUE)
```

Arguments

object	An object saved from nls or from capture.output using try with nls. See details.
fun	A function that represents the model being fit in nls. This must take the x-axis variable as the first argument and model parameters as a vector in the second argument. See details.
from, to	The range over which the function will be plotted. Defaults to range of the x-axis of the active plot.
n	The number of value at which to evaluate the function for plotting (i.e., the number of values from from to to). Larger values make smoother lines.
lwd	A numeric used to indicate the line width of the fitted line.
pal	A character that is the name of a palette. Must be one of "rich", "cm", "default", "grey", "gray", "heat", "jet", "rainbow", "topo", or "terrain", which are given in paletteChoices.
rev.col	A logical that indicates that the order of colors for plotting the lines should be reversed.
legend	Controls use and placement of the legend. See details.
cex.leg	A single numeric value that represents the character expansion value for the legend. Ignored if legend=FALSE.
box.lty.leg	A single numeric values that indicates the type of line to use for the box around the legend. The default is to not plot a box.
add	A logical indicating whether the lines should be added to the existing plot (defaults to =TRUE).

Details

Nonlinear models fit with the nls function start with starting values for model parameters and iteratively search for other model parameters that continuously reduce the residual sum-of-squares (RSS) until some pre-determined criterion suggest that the RSS cannot be (substantially) further reduced. With good starting values and well-behaved data, the minimum RSS may be found in a few (<10) iterations. However, poor starting values or poorly behaved data may lead to a prolonged and possibly failed search. An understanding of the iterations in a prolonged or failed search may help identify the failure and lead to choices that may result in a successful search. The trace=TRUE argument of nls allows one to see the values at each iterative step. The function documented here plots the "trace" results at each iteration on a previously existing plot of the data. This creates a visual of the iterative process.

The object argument may be an object saved from a successful run of nls. See the examples with SpotVA1 and CodNorwegion.

However, if nls fails to converge to a solution then no useful object is returned. In this case, trace=TRUE must be added to the failed nls call. The call is then wrapped in try to work-around the failed convergence error. This is also wrapped in capture.output to capture the "trace" results. This is then saved to an object that which can then be the object of the function documented here. This process is illustrated with the example using BSkateGB.

The function in fun is used to make predictions given the model parameter values at each step of the iteration. This function must accept the explanatory/independent variable as its first argument and values for all model parameters in a vector as its second argument. These types of functions are returned by vbFuns, GompertzFuns, logisticFuns, and RichardsFuns for common growth models and srFuns for common stock-recruitment models. See the examples.

Value

A matrix with the residual sum-of-squares in the first column and parameter estimates in the remaining columns for each iteration (rows) of nls as provided when trace=TRUE.

Note

The position of the "legend" can be controlled in three ways. First, if legend=TRUE, then the R console is suspended until the user places the legend on the plot by clicking on the point where the upper-left corner of the legend should appear. Second, legend= can be set to one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". In this case, the legend will be placed inside the plot frame at the given location. Finally, legend= can be set to a vector of length two which identifies the plot coordinates for the upper-left corner of where the legend should be placed. A legend will not be drawn if legend=FALSE or legend=NULL.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Examples following a successful fit
vb1 <- vbFuns()
fit1 <- nls(tl~vb1(age,Linf,K,t0),data=SpotVA1,start=list(Linf=12,K=0.3,t0=0))
plot(tl~age,data=SpotVA1,pch=21,bg="gray40")
nlsTracePlot(fit1,vb1,legend="bottomright")

r1 <- srFuns("Ricker")
fitSR1 <- nls(log(recruits)~log(r1(stock,a,b)),data=CodNorwegian,start=list(a=3,b=0.03))
plot(recruits~stock,data=CodNorwegian,pch=21,bg="gray40",xlim=c(0,200))
nlsTracePlot(fitSR1,r1)

# no plot, but returns trace results as a matrix
( tmp <- nlsTracePlot(fitSR1,r1,add=FALSE) )

## Not run:
if (require(FSAdata)) {
  data(BSkateGB,package="FSAdata")
  wtr <- filterD(BSkateGB,season=="winter")
  bh1 <- srFuns()
```

```
  trc <- capture.output(try(
  fitSR1 <- nls(recruits~bh1(spawners,a,b),wtr,
                start=srStarts(recruits~spawners,data=wtr),trace=TRUE)
  ))
  plot(recruits~spawners,data=wtr,pch=21,bg="gray40")
  nlsTracePlot(trc,bh1)
  # zoom in on y-axis
  plot(recruits~spawners,data=wtr,pch=21,bg="gray40",ylim=c(0.02,0.05))
  nlsTracePlot(trc,bh1,legend="top")
  # return just the trace results
  ( tmp <- nlsTracePlot(trc,bh1,add=FALSE) )
}

## End(Not run)
```

---

oddeven *Determine if a number is odd or even.*

Description

Determine if a number is odd or even.

Usage

```
is.odd(x)

is.even(x)
```

Arguments

x	A numeric vector.

Value

A logical vector of the same length as x.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## Individual values
is.odd(1)
is.odd(2)
is.even(3)
is.even(4)

## Vector of values
```

```
d <- 1:8
data.frame(d,odd=is.odd(d),even=is.even(d))
```

---

perc *Computes the percentage of values in a vector less than or greater than (and equal to) some value.*

Description

Computes the percentage of values in a vector less than or greater than (and equal to) a user-supplied value.

Usage

```
perc(x, val, dir = c("geq", "gt", "leq", "lt"), na.rm = TRUE,
  digits = getOption("digits"))
```

Arguments

x	A numeric vector.
val	A single numeric value.
dir	A string that indicates whether the percentage is for values in x that are "greater than and equal" "geq", "greater than" "gt", "less than and equal" "leq", "less than" "lt" the value in val.
na.rm	A logical that indicates whether NA values should be removed (DEFAULT) from x or not.
digits	A single numeric that indicates the number of decimals the percentage should be rounded to.

Details

This function is most useful when used with an apply-type of function.

Value

A single numeric that is the percentage of values in x that meet the criterion in dir relative to val.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

Examples

```
## vector of values
( tmp <- c(1:8,NA,NA) )

## percentages excluding NA values
perc(tmp,5)
perc(tmp,5,"gt")
perc(tmp,5,"leq")
perc(tmp,5,"lt")

## percentages including NA values
perc(tmp,5,na.rm=FALSE)
perc(tmp,5,"gt",na.rm=FALSE)
perc(tmp,5,"leq",na.rm=FALSE)
perc(tmp,5,"lt",na.rm=FALSE)
```

`PikeNY` *Summarized multiple mark-recapture data for all Northern Pike from Buckhorn Marsh, NY.*

Description

Summary results of capture histories (number captured, number of recaptured fish, and number of unmarked fish that were marked) for all Buckhorn Marsh Northern Pike (*Esox lucius*).

Format

A data frame with 21 observations on the following 4 variables:

date Capture date

n Total fish captured in each sample

m Marked fish captured in each sample

R Marked fish returned to the population

Topic(s)

- Population Size
- Abundance
- Mark-Recapture
- Capture-Recapture
- Schnabel
- Schumacher-Eschmeyer

Source

New York Power Authority. 2004. Use of Buckhorn Marsh and Grand Island tributaries by Northern Pike for spawning and as a nursery. Technical report, New York Power Authority, January 2004. Niagara Power Project (FERC No. 2216).

See Also

Used in `mrClosed` examples. Also see `PikeNYPartial1`.

Examples

```
str(PikeNY)
head(PikeNY)
```

PikeNYPartial1 — *Capture histories (4 samples), in capture history format, of a subset of Northern Pike from Buckhorn Marsh, NY.*

Description

Each line consists of the capture history over four samples of Northern Pike (*Esox lucius*) in Buckhorn Marsh. This file contains the capture histories for only those pike captured from April 1-4.

Format

A data frame with 57 observations on the following 4 variables.

id A unique identification numbers

first Indicator variable for the first sample (1=captured)

second Indicator variable for the second sample (1=captured)

third Indicator variable for the third sample (1=captured)

fourth Indicator variable for the fourth sample (1=captured)

Topic(s)

- Population Size
- Abundance
- Mark-Recapture
- Capture-Recapture
- Schnabel
- Schumacher-Eschmeyer
- Capture History

Source

Summary values taken from Table C-1 of New York Power Authority. 2004. Use of Buckhorn Marsh and Grand Island tributaries by Northern Pike for spawning and as a nursery. Technical report, New York Power Authority, January 2004. Niagara Power Project (FERC No. 2216).

See Also

Used in `capHistSum` and `mrClosed` examples. Also see `PikeNY`.

Examples

```
str(PikeNYPartial1)
head(PikeNYPartial1)
```

`plotAB` *Construct traditional (Campana-like) age-bias plots.*

Description

Constructs a traditional (e.g., like that described in Campana *et al.* (1995)) age-bias plot to visualize potential differences in paired age estimates. Ages may be from, for example, two readers of the same structure, one reader at two times, two structures (e.g., scales, spines, otoliths), or one structure and known ages.

Usage

```
plotAB(x, what = c("bias", "Campana", "numbers"), xlab = x$ref.lab,
  ylab = x$nref.lab, xlim = NULL, ylim = NULL,
  yaxt = graphics::par("yaxt"), xaxt = graphics::par("xaxt"),
  col.agree = "gray60", lwd.agree = lwd, lty.agree = 2, lwd = 1,
  sfrac = 0, pch.mean = 19, pch.mean.sig = 21, cex.mean = lwd,
  col.CI = "black", col.CIsig = "red", lwd.CI = lwd,
  sfrac.CI = sfrac, show.n = FALSE, nYpos = 1.03, cex.n = 0.75,
  cex.numbers = 0.75, col.numbers = "black", ...)
```

Arguments

`x`	An object of class `ageBias`, usually a result from `ageBias`.
`what`	A string that indicates what type of plot to construct. See details.
`xlab, ylab`	A string label for the x-axis (reference) or y-axis (non-reference) age estimates, respectively.
`xlim, ylim`	A numeric vector of length 2 that contains the limits of the x-axis (reference ages) or y-axis (non-reference ages), respectively.
`xaxt, yaxt`	A string which specifies the x- and y-axis types. Specifying "n" suppresses plotting of the axis. See `?par`.

col.agree	A string or numeric for the color of the 1:1 or zero (if difference=TRUE) reference line.
lwd.agree	A numeric for the line width of the 1:1 or zero (if difference=TRUE) reference line.
lty.agree	A numeric for the line type of the 1:1 or zero (if difference=TRUE) reference line.
lwd	A numeric that controls the separate 'lwd' argument (e.g., lwd.CI and lwd.range).
sfrac	A numeric that controls the separate 'sfrac' arguments (e.g., sfrac.CI and sfrac.range). See sfrac in plotCI of **plotrix**.
pch.mean	A numeric for the plotting character used for the mean values when the means are considered insignificant.
pch.mean.sig	A numeric for the plotting character for the mean values when the means are considered significant.
cex.mean	A character expansion value for the size of the mean symbol in pch.mean and pch.mean.sig.
col.CI	A string or numeric for the color of confidence interval bars that are considered non-significant.
col.CIsig	A string or numeric for the color of confidence interval bars that are considered significant.
lwd.CI	A numeric for the line width of the confidence interval bars.
sfrac.CI	A numeric for the size of the ends of the confidence interval bars. See sfrac in plotCI of **plotrix**.
show.n	A logical for whether the sample sizes for each level of the x-axis variable is shown (=TRUE, default) or not (=FALSE).
nYpos	A numeric for the relative Y position of the sample size values when show.n=TRUE. For example, if nYpos=1.03 then the sample size values will be centered at 3 percent above the top end of the y-axis.
cex.n	A character expansion value for the size of the sample size values.
cex.numbers	A character expansion value for the size of the numbers plotted when what="numbers" is used.
col.numbers	A string for the color of the numbers plotted when what="numbers" is used.
...	Additional arguments for methods.

Details

Two types of plots for visualizing differences between sets of two age estimates may be created. The reference ages are plotted on the x-axis and the nonreference ages are on the y-axis. The 1:1 (45 degree) agreement line is shown for comparative purposes. The default plot (using what="bias") was inspired by the age bias plot introduced by Campana *et al.* (1995). The default settings for this age bias plot show the mean and confidence interval for the nonreference ages at each of the reference ages. The level of confidence is controlled by sig.level= given in the original ageBias call (i.e., confidence level is 100*(1-sig.level)). Confidence intervals are only shown if the sample size is greater than the value in min.n.CI= (also from the original call to ageBias). Confidence

intervals plotted in red with an open dot (by default; these can be changed with `col.CIsig` and `pch.mean.sig`, respectively) do not contain the reference age (see discussion of t-tests in `ageBias`). Sample sizes at each reference age are shown if `show.n=TRUE`. The position of the sample sizes is controlled with `nYpos=`, whereas their size is controlled with `cex.n`. Arguments may be used to nearly replicate the age bias plot as introduced by Campana *et al.* (1995) as shown in the examples.

The frequency of observations at each unique (x,y) coordinate are shown by using `what="numbers"` in `plotAB`. The size of the numbers is controlled with `cex.numbers`.

Value

Nothing, but see details for a description of the plot that is produced.

IFAR Chapter

4-Age Comparisons. **This is most of the original functionality that was in `plot` in the book. See examples.**

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Campana, S.E., M.C. Annand, and J.I. McMillan. 1995. Graphical and statistical methods for determining the consistency of age determinations. Transactions of the American Fisheries Society 124:131-138. [Was (is?) available from http://www.bio.gc.ca/otoliths/documents/Campana%20et%20al%201995%20TAFS.

See Also

See `ageBias` and its plot method for what I consider a better age-bias plot; `agePrecision` for measures of precision between pairs of age estimates; and `compare2` in **fishmethods** for similar functionality.

Examples

```
# Must create ageBias object first
ab1 <- ageBias(scaleC~otolithC,data=WhitefishLC,
               ref.lab="Otolith Age",nref.lab="Scale Age")

# Default plot
plotAB(ab1)

# Very close to Campana et al. (2001)
plotAB(ab1,pch.mean.sig=19,col.CIsig="black",sfrac=0.01,
       ylim=c(-1,23),xlim=c(-1,23))
# Show sample sizes (different position and size than default)
plotAB(ab1,show.n=TRUE,nYpos=0.02,cex.n=0.5)

# Traditional numbers plot
plotAB(ab1,what="numbers")
```

plotBinResp *Plots a binary response variable versus a quantitative explanatory variable.*

Description

A function to plot a binary response variable versus a quantitative explanatory variable.

Usage

```
plotBinResp(x, ...)

## Default S3 method:
plotBinResp(x, y, xlab = paste(deparse(substitute(x))),
  ylab = paste(deparse(substitute(y))), plot.pts = TRUE,
  col.pt = "black", transparency = NULL, plot.p = TRUE,
  breaks = 25, p.col = "blue", p.pch = 3, p.cex = 1.25,
  yaxis1.ticks = seq(0, 1, 0.1), yaxis1.lbls = c(0, 0.5, 1),
  yaxis2.show = TRUE, ...)

## S3 method for class 'formula'
plotBinResp(x, data = NULL, xlab = names(mf)[2],
  ylab = names(mf)[1], ...)
```

Arguments

x	A quantitative explanatory variable or a formula of the form factor~quant.
...	Other arguments to be passed to the plot functions.
y	A binary response variable.
xlab	A string for labeling the x-axis.
ylab	A string for labeling the y-axis.
plot.pts	A logical that indicates (TRUE (default)) whether the points should be plotted (TRUE; default) or not (FALSE).
col.pt	A string used to indicate the color of the plotted points. Will be transparent unless transparency=1.
transparency	A numeric that indicates how many points would be plotted on top of each other before the 'point' would have the full col.pt color. The reciprocal of this value is the alpha transparency value.
plot.p	A logical that indicates if the proportion for categorized values of X are plotted (TRUE; default).
breaks	A number that indicates how many intervals over which to compute proportions or a numeric vector that contains the endpoints of the intervals over which to compute proportions if plot.p=TRUE.
p.col	A color to plot the proportions.

p.pch	A plotting character for plotting the proportions.
p.cex	A character expansion factor for plotting the proportions.
yaxis1.ticks	A numeric vector that indicates where tick marks should be placed on the left y-axis (for the proportion of 'successes').
yaxis1.lbls	A numeric vector that indicates labels for the tick marks on the left y-axis (for the proportion of 'successes').
yaxis2.show	A logical that indicates whether the right y-axis should be created (=TRUE; default) or not.
data	The data frame from which the formula should be evaluated.

Details

This function produces a plot that can be used to visualize the density of points for a binary response variable as a function of a quantitative explanatory variable. In addition, the proportion of "1"s for the response variable at various "levels" of the explanatory variable are shown.

Value

None. However, a plot is produced.

Note

This function is meant to allow newbie students the ability to visualize the data corresponding to a binary logistic regression without getting "bogged-down" in the gritty details of how to produce this plot.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

fitPlot and cdplot.

Examples

```
## NASA space shuttle o-ring failures -- from graphics package
fail <- factor(c(2,2,2,2,1,1,1,1,1,1,2,1,2,1,1,1,1,2,1,1,1,1,1),
levels = 1:2, labels = c("no","yes"))
temperature <- c(53,57,58,63,66,67,67,67,68,69,70,70,70,70,72,73,75,75,76,76,78,79,81)
d <- data.frame(temperature,fail,fail2=factor(fail,levels=c("yes","no")))

## Default plot (using formula notation)
plotBinResp(fail~temperature,data=d)
plotBinResp(fail2~temperature,data=d)

## Controlling where proportions are computed with a sequence in breaks
plotBinResp(fail~temperature,data=d,breaks=seq(50,85,5))
```

```
## Controlling where proportions are computed with an integer in breaks
plotBinResp(fail~temperature,data=d,breaks=10)

## Controlling where proportions are computed at each value of x
plotBinResp(fail~temperature,data=d,breaks=NULL)

## Don't plot points, just plot proportions
plotBinResp(fail~temperature,data=d,plot.pts=FALSE)

## Don't plot proportions, just plot points
plotBinResp(fail~temperature,data=d,plot.p=FALSE)

## Change points colors, and eliminate transparency
plotBinResp(fail~temperature,data=d,col.pt="red",transparency=1)

## Remove the right y-axis
plotBinResp(fail~temperature,data=d,yaxis2.show=FALSE)

## Change left y-axis ticks
plotBinResp(fail~temperature,data=d,yaxis1.ticks=c(0,1),yaxis1.lbls=c(0,1))
```

`poiCI` *Confidence interval for Poisson counts.*

Description

Computes a confidence interval for the Poisson counts.

Usage

```
poiCI(x, conf.level = 0.95, type = c("exact", "daly", "byar",
  "asymptotic"), verbose = FALSE)
```

Arguments

`x`	A single number or vector that represents the number of observed successes.
`conf.level`	A number that indicates the level of confidence to use for constructing confidence intervals (default is `0.95`).
`type`	A string that identifies the type of method to use for the calculations. See details.
`verbose`	A logical that indicates whether `x` should be included in the returned matrix (=`TRUE`) or not (=`FALSE`; DEFAULT).

Details

Computes a CI for the Poisson counts using the `exact`, gamma distribution (`daly`‘), Byar’s (`byar`), or normal approximation (`asymptotic`) methods. This is largely a wrapper to `pois.exact`, `pois.daly`, `pois.byar`, and `pois.approx` functions documented in `pois.conf.int`in **epitools**.

Value

A #x2 matrix that contains the lower and upper confidence interval bounds as columns and, if `verbose=TRUE` `x`.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See `pois.exact`, `pois.daly`, `pois.byar`, and `pois.approx` (documented in `pois.conf.int`) in **epitools** for more description and references.

Examples

```
## Demonstrates using all types at once
poiCI(12)

## Selecting types
poiCI(12,type="daly")
poiCI(12,type="byar")
poiCI(12,type="asymptotic")
poiCI(12,type="asymptotic",verbose=TRUE)
poiCI(12,type=c("exact","daly"))
poiCI(12,type=c("exact","daly"),verbose=TRUE)

## Demonstrates use with multiple inputs
poiCI(c(7,10),type="exact")
poiCI(c(7,10),type="exact",verbose=TRUE)
```

---

psdAdd *Creates a vector of Gabelhouse lengths for each species in an entire data frame.*

Description

Creates a vector of the Gabelhouse lengths specific to a species for all individuals in an entire data frame.

Usage

```
psdAdd(len, ...)

## Default S3 method:
psdAdd(len, spec, units = c("mm", "cm", "in"),
  use.names = TRUE, addSpec = NULL, addLens = NULL, verbose = TRUE,
  ...)
```

```
## S3 method for class 'formula'
psdAdd(len, data = NULL, units = c("mm", "cm", "in"),
  use.names = TRUE, addSpec = NULL, addLens = NULL, verbose = TRUE,
  ...)
```

Arguments

len	A numeric vector that contains lengths measurements or a formula of the form len~spec where "len" generically represents the length variable and "spec" generically represents the species variable. Note that this formula can only contain two variables and must have the length variable on the left-hand-side and the species variable on the right-hand-side.
...	Not used.
spec	A character or factor vector that contains the species names. Ignored if len is a formula.
units	A string that indicates the type of units used for the lengths. Choices are mm for millimeters (DEFAULT), cm for centimeters, and in for inches.
use.names	A logical that indicates whether the vector returned is numeric (=FALSE) or string (=TRUE; default) representations of the Gabelhouse lengths. See details.
addSpec	A character vector of species names for which addLens will be provided.
addLens	A numeric vector of lengths that should be used in addition to the Gabelhouse lengths for the species in addSpec. See examples.
verbose	A logical that indicates whether detailed messages about species without Gabelhouse lengths or with no recorded values should be printed or not.
data	A data.frame that minimally contains the length measurements and species names if len is a formula.

Details

This computes a vector that contains the Gabelhouse lengths specific to each species for all individuals in an entire data frame. The vector can be appended to an existing data.frame to create a variable that contains the Gabelhouse lengths for each individual. The Gabelhouse length value will be NA for each individual for which a Gabelhouse length definitions do not exist in PSDlit. Species names in the data.frame must be the same as those used in PSDlit. See the examples for one method for changing species names to something that this function will recognize.

Individuals shorter than "stock" length will be listed as substock if use.names=TRUE or 0 if use.names=FALSE.

Additional lengths to be used for a species may be included by giving a vector of species names in addSpec and a corresponding vector of additional lengths in addLens. Note, however, that use.names will be reset to FALSE if addSpec and addLens are specified, as there is no way to order the names for all species when additional lengths are used.

Value

A numeric or factor vector that contains the Gabelhouse length categories.

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Guy, C.S., R.M. Neumann, and D.W. Willis. 2006. New terminology for proportional stock density (PSD) and relative stock density (RSD): proportional size structure (PSS). Fisheries 31:86-87. [Was (is?) from http://pubstorage.sdstate.edu/wfs/415-F.pdf.]

Guy, C.S., R.M. Neumann, D.W. Willis, and R.O. Anderson. 2006. Proportional size distribution (PSD): A further refinement of population size structure index terminology. Fisheries 32:348. [Was (is?) from http://pubstorage.sdstate.edu/wfs/450-F.pdf.]

Willis, D.W., B.R. Murphy, and C.S. Guy. 1993. Stock density indices: development, use, and limitations. Reviews in Fisheries Science 1:203-222. [Was (is?) from http://web1.cnre.vt.edu/murphybr/web/Readings/Willis%20

See Also

psdVal, psdCalc, psdPlot, PSDlit, and wrAdd for related functions. See mapvalues for help in changing species names to match those in PSDlit.

Examples

```
## Create random data for three species
# only for repeatability
set.seed(345234534)
dbg <- data.frame(species=factor(rep(c("Bluegill"),30)),tl=round(rnorm(30,130,50),0))
dbg$wt <- round(4.23e-06*dbg$tl^3.316+rnorm(30,0,10),1)
dlb <- data.frame(species=factor(rep(c("Largemouth Bass"),30)),tl=round(rnorm(30,350,60),0))
dlb$wt <- round(2.96e-06*dlb$tl^3.273+rnorm(30,0,60),1)
dbt <- data.frame(species=factor(rep(c("Bluefin Tuna"),30)),tl=round(rnorm(30,1900,300),0))
dbt$wt <- round(4.5e-05*dbt$tl^2.8+rnorm(30,0,6000),1)
df <- rbind(dbg,dlb,dbt)
str(df)

## Examples (non-dplyr)
# Add variable using category names -- formula notation
df$PSD <- psdAdd(tl~species,data=df)
head(df)
# Add variable using category names -- non-formula notation
df$PSD1 <- psdAdd(df$tl,df$species)
head(df)
# Add variable using length values as names
df$PSD2 <- psdAdd(tl~species,data=df,use.names=FALSE)
head(df)
# Add additional length and name for Bluegill
df$PSD3 <- psdAdd(tl~species,data=df,addSpec="Bluegill",addLens=175)
```

```
head(df)
# Add additional lengths and names for Bluegill and Largemouth Bass from a data.frame
addls <- data.frame(species=c("Bluegill","Largemouth Bass","Largemouth Bass"),
                    lens=c(175,254,356))
df$psd4 <- psdAdd(tl~species,data=df,addSpec=addls$species,addLens=addls$lens)
head(df)

## All of the above but using dplyr
if (require(dplyr)) {
  df <- df %>%
    mutate(PSD1A=psdAdd(tl,species)) %>%
    mutate(PSD2A=psdAdd(tl,species,use.names=FALSE)) %>%
    mutate(psd3a=psdAdd(tl,species,addSpec="Bluegill",addLens=175)) %>%
    mutate(psd4a=psdAdd(tl,species,addSpec=addls$species,addLens=addls$lens))
}
df
```

---

psdCalc *Convenience function for calculating PSD-X and PSD X-Y values.*

Description

Convenience function for calculating (traditional) PSD-X and (incremental) PSD X-Y values for all Gabelhouse lengths and increments thereof.

Usage

```
psdCalc(formula, data, species, units = c("mm", "cm", "in"),
  method = c("multinomial", "binomial"), conf.level = 0.95,
  addLens = NULL, addNames = NULL, justAdds = FALSE,
  what = c("all", "traditional", "incremental", "none"),
  drop0Est = TRUE, showIntermediate = FALSE, digits = 0)
```

Arguments

formula	A formula of the form ~length where "length" generically represents a variable in data that contains the observed lengths. Note that this formula may only contain one variable and it must be numeric.
data	A data.frame that minimally contains the observed lengths given in the variable in formula.
species	A string that contains the species name for which Gabelhouse lengths exist. See psdVal for details.
units	A string that indicates the type of units used for the lengths. Choices are mm for millimeters (DEFAULT), cm for centimeters, and in for inches.
method	A character that identifies the confidence interval method to use. See details in psdCI.

conf.level	A number that indicates the level of confidence to use for constructing confidence intervals (default is 0.95).
addLens	A numeric vector that contains minimum lengths for additional categories. See psdVal for details.
addNames	A string vector that contains names for the additional lengths added with addLens. See psdVal for details.
justAdds	A logical that indicates whether just the values related to the length sin addLens should be returned.
what	A string that indicates the type of PSD values that will be printed. See details.
drop0Est	A logical that indicates whether the PSD values that are zero should be dropped from the output.
showIntermediate	A logical that indicates whether the number of fish in the category and the number of stock fish (i.e., "intermediate" values) should be included in the returned matrix. Default is to not include these values.
digits	A numeric that indicates the number of decimals to round the result to. Default is zero digits following the recommendation of Neumann and Allen (2007).

Details

Computes the (traditional) PSD-X and (incremental) PSD X-Y values, with associated confidence intervals, for each Gabelhouse length. All PSD-X and PSD X-Y values are printed if what="all" (DEFAULT), only PSD-X values are printed if what="traditional", only PSD X-Y values are printed if what="incremental", and nothing is printed (but the matrix is still returned) if what="none".

Confidence intervals can be computed with either the multinomial (Default) or binomial distribution as set in method. See details in psdCI for more information.

Value

A matrix with columns that contain the computed PSD-X or PSD X-Y values and associated confidence intervals. If showIntermediate=TRUE then the number of fish in the category and the number of stock fish will also be shown.

Testing

Point estimate calculations match those constructed "by hand."

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Guy, C.S., R.M. Neumann, and D.W. Willis. 2006. New terminology for proportional stock density (PSD) and relative stock density (RSD): proportional size structure (PSS). Fisheries 31:86-87. [Was (is?) from http://pubstorage.sdstate.edu/wfs/415-F.pdf.]

Guy, C.S., R.M. Neumann, D.W. Willis, and R.O. Anderson. 2006. Proportional size distribution (PSD): A further refinement of population size structure index terminology. Fisheries 32:348. [Was (is?) from http://pubstorage.sdstate.edu/wfs/450-F.pdf.]

Neumann, R. M. and Allen, M. S. 2007. Size structure. In Guy, C. S. and Brown, M. L., editors, Analysis and Interpretation of Freshwater Fisheries Data, Chapter 9, pages 375-421. American Fisheries Society, Bethesda, MD.

Willis, D.W., B.R. Murphy, and C.S. Guy. 1993. Stock density indices: development, use, and limitations. Reviews in Fisheries Science 1:203-222. [Was (is?) from http://web1.cnre.vt.edu/murphybr/web/Readings/Willis%20

See Also

See psdVal, psdPlot, psdAdd, PSDlit, tictactoe, lencat, and rcumsum for related functionality.

Examples

```
## Random length data
# suppose this is yellow perch to the nearest mm
yepdf <- data.frame(yepmm=round(c(rnorm(100,mean=125,sd=15),rnorm(50,mean=200,sd=25),
                    rnorm(20,mean=300,sd=40)),0),
                    species=rep("Yellow Perch",170))
psdCalc(~yepmm,data=yepdf,species="Yellow perch",digits=1)
psdCalc(~yepmm,data=yepdf,species="Yellow perch",digits=1,drop0Est=TRUE)

## add a length
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=150)

## add lengths with names
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=150,addNames="minLen")
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c("minLen"=150))
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c(150,275),addNames=c("minSlot","maxSlot"))
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c("minLen"=150,"maxslot"=275))

## add lengths with names, return just those values that use those lengths
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c("minLen"=150),justAdds=TRUE)
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c("minLen"=150),justAdds=TRUE,
        what="traditional")
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c(150,275),
        addNames=c("minSlot","maxSlot"),justAdds=TRUE)
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=c(150,275),
        addNames=c("minSlot","maxSlot"),justAdds=TRUE,what="traditional")

## different output types
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=150,what="traditional")
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=150,what="incremental")
psdCalc(~yepmm,data=yepdf,species="Yellow perch",addLens=150,what="none")
```

```
## Show intermediate values
psdCalc(~yepmm,data=yepdf,species="Yellow perch",showInterm=TRUE)
psdCalc(~yepmm,data=yepdf,species="Yellow perch",what="traditional",showInterm=TRUE)
psdCalc(~yepmm,data=yepdf,species="Yellow perch",what="incremental",showInterm=TRUE)

## Control the digits
psdCalc(~yepmm,data=yepdf,species="Yellow perch",digits=1)
```

---

psdCI *Compute confidence intervals for PSD-X and PSD X-Y values.*

Description

Compute confidence intervals for (traditional) PSD-X and (incremental) PSD X-Y values as requested by the user.

Usage

```
psdCI(indvec, ptbl, n, method = c("binomial", "multinomial"),
  bin.type = c("wilson", "exact", "asymptotic"), conf.level = 0.95,
  label = NULL, digits = 1)
```

Arguments

indvec	A numeric vector of 0s and 1s that identify the linear combination of proportions from ptbl that the user is interested in. See details.
ptbl	A numeric vector or array that contains the proportion or percentage of all individuals in each length category. See details.
n	A single numeric of the number of fish used to construct ptbl.
method	A string that identifies the confidence interval method to use. See details.
bin.type	A string that identifies the type of method to use for calculation of the confidence intervals when Rmethod="binomial". See details of binCI.
conf.level	A number that indicates the level of confidence to use for constructing confidence intervals (default is 0.95).
label	A single string that can be used to label the row of the output matrix.
digits	A numeric that indicates the number of decimals to round the result to.

Details

Computes confidence intervals for (traditional) PSD-X and (incremental) PSD X-Y values. Two methods can be used as chosen with method=. If method="binomial" then the binomial distribution (via binCI()) is used. If method="multinomial" then the multinomial method described by Brenden et al. (2008) is used. This function is defined to compute one confidence interval so

method="binomial" is the default. See examples and psdCalc for computing several simultaneous confidence intervals.

A table of proportions within each length category is given in ptbl. If ptbl has any values greater than 1 then it is assumed that a table of percentages was supplied and the entire table will be divided by 100 to continue. The proportions must sum to 1 (with some allowance for rounding).

A vector of length equal to the length of ptbl is given in indvec which contains zeros and ones to identify the linear combination of values in ptbl to use to construct the confidence intervals. For example, if ptbl has four proportions then indvec=c(1,0,0,0) would be used to construct a confidence interval for the population proportion in the first category. Alternatively, indvec=c(0,0,1,1) would be used to construct a confidence interval for the population proportion in the last two categories. This vector must not contain all zeros or all ones.

Value

A matrix with columns that contain the computed PSD-X or PSD X-Y value and the associated confidence interval. The confidence interval values were set to zero or 100 if the computed value was negative or greater than 100, respectively.

Testing

The multinomial results match the results given in Brendent et al. (2008).

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Brenden, T.O., T. Wagner, and B.R. Murphy. 2008. Novel tools for analyzing proportional size distribution index data. North American Journal of Fisheries Management 28:1233-1242. [Was (is?) from http://qfc.fw.msu.edu/Publications/Publication%20List/2008/Novel%20Tools%20for%20Analyzing%20Proportional%

See Also

See psdVal, psdPlot, psdAdd, PSDlit, tictactoe, lencat, and rcumsum for related functionality.

Examples

```
## similar to Brenden et al. (2008)
n <- 997
ipsd <- c(130,491,253,123)/n

## single binomial
psdCI(c(0,0,1,1),ipsd,n=n)
psdCI(c(1,0,0,0),ipsd,n=n,label="PSD S-Q")
```

```
## single multinomial
psdCI(c(0,0,1,1),ipsd,n=n,method="multinomial")
psdCI(c(1,0,0,0),ipsd,n=n,method="multinomial",label="PSD S-Q")

## multiple multinomials (but see psdCalc())
lbls <- c("PSD S-Q","PSD Q-P","PSD P-M","PSD M-T","PSD","PSD-P")
imat <- matrix(c(1,0,0,0,
                 0,1,0,0,
                 0,0,1,0,
                 0,0,0,1,
                 0,1,1,1,
                 0,0,1,1),nrow=6,byrow=TRUE)
rownames(imat) <- lbls
imat

mcis <- t(apply(imat,MARGIN=1,FUN=psdCI,ptbl=ipsd,n=n,method="multinomial"))
colnames(mcis) <- c("Estimate","95% LCI","95% UCI")
mcis

## Multiple "Bonferroni-corrected" (for six comparisons) binomial method
bcis <- t(apply(imat,MARGIN=1,FUN=psdCI,ptbl=ipsd,n=n,conf.level=1-0.05/6))
colnames(bcis) <- c("Estimate","95% LCI","95% UCI")
bcis
```

`PSDlit` *Gabelhouse five-cell length categories for various species.*

Description

Cutoffs for the Gabelhouse five-cell length categories for a variety of species.

Format

A data frame of 58 observations on the following 11 variables:

species Species name.

stock.in Stock length in inches.

quality.in Quality length in inches.

preferred.in Preferred length in inches.

memorable.in Memorable length in inches.

trophy.in Trophy length in inches.

stock.cm Stock length in cm.

quality.cm Quality length in cm.

preferred.cm Preferred length in cm.

memorable.cm Memorable length in cm.

trophy.cm Trophy length in cm.

Topic(s)

- Size structure
- Proportional size structure
- Relative stock density
- Proportional stock density

IFAR Chapter

6-Size Structure.

Source

Original summary table from Dr. Michael Hansen, University of Wisconsin-Stevens Point. Additional species have been added by the package author from the literature.

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See psdVal, psdCalc, psdPlot, psdAdd, and tictactoe for related functionality.

Examples

```
str(PSDlit)
head(PSDlit)
```

---

psdPlot *Length-frequency histogram with Gabelhouse lengths highlighted.*

Description

Constructs a length-frequency histogram with Gabelhouse lengths highlighted.

Usage

```
psdPlot(formula, data, species = "List", units = c("mm", "cm", "in"),
  startcat = 0, w = 1, justPSDQ = FALSE, main = "",
  xlab = "Length", ylab = "Number", xlim = NULL, ylim = c(0,
  max(h$counts)), substock.col = "white", stock.col = "gray90",
  psd.col = "black", psd.lty = 2, psd.lwd = 1, show.abbrevs = TRUE,
  psd.add = TRUE, psd.pos = "topleft", psd.cex = 0.75, ...)
```

Arguments

formula	A formula of the form ~length where "length" generically represents a variable in data that contains length measurements. Note that this formula can only contain one variable.
data	A data.frame that minimally contains the length measurements given in the variable in the formula.
species	A string that contains the species name for which Gabelhouse length categories exist. See psdVal for details.
units	A string that indicates the type of units used for the length measurements. Choices are mm for millimeters (DEFAULT), cm for centimeters, and in for inches.
startcat	A number that indicates the beginning of the first length-class.
w	A number that indicates the width of length classes to create.
justPSDQ	A logical that indicates whether just stock and quality (for PSD-Q calculations) categories should be used. If FALSE (default) then the five Gabelhouse categories will be used.
main	A string that serves as the main label for the histogram.
xlab	A string that serves as the label for the x-axis.
ylab	A string that serves as the label for the y-axis.
xlim	A numeric vector of length two that indicates the minimum and maximum values (i.e., fish lengths) for the x-axis.
ylim	A numeric vector of length two that indicates the minimum and maximum values for the y-axis.
substock.col	A string that indicates the color to use for the bars representing under-stock size fish.
stock.col	A string that indicates the color to use for the bars representing stock size fish.
psd.col	A string that indicates the color to use for the vertical lines at the Gabelhouse length category values.
psd.lty	A numeric that indicates the line type to use for the vertical lines at the Gabelhouse length category values.
psd.lwd	A numeric that indicates the line width to use for the vertical lines at the Gabelhouse length category values.
show.abbrevs	A logical that indicates if the abbreviations for the Gabelhouse length categories should be added to the top of the plot.
psd.add	A logical that indicates if the calculated PSD values should be added to the plot (default is TRUE).
psd.pos	A string that indicates the position for where the PSD values will be shown. See details in legend.
psd.cex	A numeric value that indicates the character expansion for the PSD values text.
...	Arguments to be passed to the low-level plotting functions.

Details

Constructs a length-frequency histogram with the stock-sized fish highlighted, the Gabelhouse lengths marked by vertical lines, and the (traditional) PSD-X values superimposed.

The length of fish plotted on the x-axis can be controlled with `xlim`, however, the minimum value in `xlim` must be less than the stock length for that species.

Value

None. However, a graphic is produced.

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Guy, C.S., R.M. Neumann, and D.W. Willis. 2006. New terminology for proportional stock density (PSD) and relative stock density (RSD): proportional size structure (PSS). Fisheries 31:86-87. [Was (is?) from http://pubstorage.sdstate.edu/wfs/415-F.pdf.]

Guy, C.S., R.M. Neumann, D.W. Willis, and R.O. Anderson. 2006. Proportional size distribution (PSD): A further refinement of population size structure index terminology. Fisheries 32:348. [Was (is?) from http://pubstorage.sdstate.edu/wfs/450-F.pdf.]

Willis, D.W., B.R. Murphy, and C.S. Guy. 1993. Stock density indices: development, use, and limitations. Reviews in Fisheries Science 1:203-222. [Was (is?) from http://web1.cnre.vt.edu/murphybr/web/Readings/Willis%20

See Also

See `psdVal`, `psdCalc`, `psdAdd`, `PSDlit`, `lencat`, `tictactoe`, `lencat`, and `rcumsum` for related functionality.

Examples

```
## Random length data
# suppose this is yellow perch to the nearest mm
mm <- c(rnorm(100,mean=125,sd=15),rnorm(50,mean=200,sd=25),
        rnorm(20,mean=300,sd=40))
# same data to the nearest 0.1 cm
cm <- mm/10
# same data to the nearest 0.1 in
inch <- mm/25.4
# put together as data.frame
df <- data.frame(mm,cm,inch)

## Example graphics
```

```
op <- par(mar=c(3,3,2,1),mgp=c(1.7,0.5,0))
# mm data using 10-mm increments
psdPlot(~mm,data=df,species="Yellow perch",w=10)
# cm data using 1-cm increments
psdPlot(~cm,data=df,species="Yellow perch",units="cm",w=1)
# inch data using 1-in increments
psdPlot(~inch,data=df,species="Yellow perch",units="in",w=1)
# same as first with some color changes
psdPlot(~mm,data=df,species="Yellow perch",w=10,substock.col="gray90",
        stock.col="gray30")
# ... but without the PSD values
psdPlot(~mm,data=df,species="Yellow perch",w=10,psd.add=FALSE)
# ... demonstrate use of xlim
psdPlot(~mm,data=df,species="Yellow perch",w=10,xlim=c(100,300))

## different subsets of fish
# ... without any sub-stock fish
brks <- psdVal("Yellow Perch")
tmp <- Subset(df,mm>brks["stock"])
psdPlot(~mm,data=tmp,species="Yellow perch",w=10)
# ... without any sub-stock or stock fish
tmp <- Subset(df,mm>brks["quality"])
psdPlot(~mm,data=tmp,species="Yellow perch",w=10)
# ... with only sub-stock, stock, and quality fish ... only PSD-Q
tmp <- Subset(df,mm<brks["preferred"])
psdPlot(~mm,data=tmp,species="Yellow perch",w=10)
# ... with only sub-stock fish (don't run ... setup to give an error)
tmp <- Subset(df,mm<brks["stock"])
## Not run:  psdPlot(~mm,data=tmp,species="Yellow perch",w=10)
par(op)
```

---

psdVal *Finds Gabelhouse lengths (for PSD calculations) for a species.*

Description

Returns a vector with the five Gabelhouse lengths for a chosen species.

Usage

```
psdVal(species = "List", units = c("mm", "cm", "in"),
  incl.zero = TRUE, addLens = NULL, addNames = NULL)
```

Arguments

species	A string that contains the species name for which to find Gabelhouse lengths. See details.
units	A string that indicates the units for the returned lengths. Choices are mm for millimeters (DEFAULT), cm for centimeters, and in for inches.

incl.zero	A logical that indicates if a zero is included in the first position of the returned vector (DEFAULT) or not. This position will be named "substock". See details.
addLens	A numeric vector that contains minimum length definitions for additional categories. See details.
addNames	A string vector that contains names for the additional length categories added with addLens. See details.

Details

Finds the Gabelhouse lengths from data(PSDlit) for the species given in species. The species name must be spelled exactly (within capitalization differences) as it appears in data(PSDlit). Type psdVal() to see the list of species and how they are spelled.

A zero is included in the first position of the returned vector if incl.zero=TRUE. This is useful when computing PSD values with a data.frame that contains fish smaller than the stock length.

Additional lengths may be added to the returned vector with addLens. Names for these lengths can be included in addNames. If addNames is non-NULL, then it must be of the same length as addLens. If addLens is non-NULL but addNames is NULL, then the default names will be the same as the lengths in addLens. The addLens argument is useful for calculating PSD values that are different from the Gabelhouse lengths.

Value

A vector of minimum values for length categories for the chosen species.

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Guy, C.S., R.M. Neumann, and D.W. Willis. 2006. New terminology for proportional stock density (PSD) and relative stock density (RSD): proportional size structure (PSS). Fisheries 31:86-87. [Was (is?) from http://pubstorage.sdstate.edu/wfs/415-F.pdf.]

Guy, C.S., R.M. Neumann, D.W. Willis, and R.O. Anderson. 2006. Proportional size distribution (PSD): A further refinement of population size structure index terminology. Fisheries 32:348. [Was (is?) from http://pubstorage.sdstate.edu/wfs/450-F.pdf.]

Willis, D.W., B.R. Murphy, and C.S. Guy. 1993. Stock density indices: development, use, and limitations. Reviews in Fisheries Science 1:203-222. [Was (is?) from http://web1.cnre.vt.edu/murphybr/web/Readings/Willis%20

See Also

See psdCalc, psdPlot, psdAdd, PSDlit, tictactoe, lencat, and rcumsum for related functionality.

Examples

```
# List all of the species
psdVal()
# Demonstrate typical usages
psdVal("Yellow perch")
psdVal("Walleye",units="cm")
psdVal("Bluegill",units="in")
psdVal("Bluegill",units="in",incl.zero=FALSE)
psdVal("Bluegill")
# Demonstrate that it will work with mis-capitalization
psdVal("bluegill")
psdVal("Yellow Perch")
# Demonstrate adding in user-defined categories
psdVal("Bluegill",units="in",addLens=7)
psdVal("Bluegill",units="in",addLens=7,addNames="MinLen")
psdVal("Bluegill",units="in",addLens=c(7,9),addNames=c("MinSlot","MaxSlot"))
psdVal("Bluegill",units="in",addLens=c("MinLen"=7))
psdVal("Bluegill",units="in",addLens=c("MinSlot"=7,"MaxSlot"=9))
```

---

rcumsum *Computes the prior to or reverse cumulative sum of a vector.*

Description

Computes the prior-to (i.e., the cumulative sum prior to but not including the current value) or the reverse (i.e., the number that large or larger) cumulative sum of a vector. Also works for 1-dimensional tables, matrices, and data.frames, though it is best used with vectors.

Usage

```
rcumsum(x)

pcumsum(x)
```

Arguments

x	a numeric object.

Value

A numeric vector that contains the prior-to or reverse cumulative sums.

Note

An NA in the vector causes all returned values at and after the first NA for pcumsum and at and before the last NA for rcumsum to be NA. See the examples.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

cumsum.

Examples

```
## Simple example
cbind(vals=1:10,
      cum=cumsum(1:10),
      pcum=pcumsum(1:10),
      rcum=rcumsum(1:10))

## Example with NA
vals <- c(1,2,NA,3)
cbind(vals,
      cum=cumsum(vals),
      pcum=pcumsum(vals),
      rcum=rcumsum(vals))

## Example with NA
vals <- c(1,2,NA,3,NA,4)
cbind(vals,
      cum=cumsum(vals),
      pcum=pcumsum(vals),
      rcum=rcumsum(vals))

## Example with a matrix
mat <- matrix(c(1,2,3,4,5),nrow=1)
cumsum(mat)
pcumsum(mat)
rcumsum(mat)

## Example with a table (must be 1-d)
df <- sample(1:10,100,replace=TRUE)
tbl <- table(df)
cumsum(tbl)
pcumsum(tbl)
rcumsum(tbl)

## Example with a data.frame (must be 1-d)
df <- sample(1:10,100,replace=TRUE)
tbl <- as.data.frame(table(df))[,-1]
cumsum(tbl)
pcumsum(tbl)
rcumsum(tbl)
```

`removal` *Population estimates for k-, 3-, or 2-pass removal data.*

Description

Computes estimates, with confidence intervals, of the population size and probability of capture from the number of fish removed in k-, 3-, or 2-passes in a closed population.

Usage

```
removal(catch, method = c("CarleStrub", "Zippin", "Seber3", "Seber2",
  "RobsonRegier2", "Moran", "Schnute", "Burnham"), alpha = 1, beta = 1,
  CS.se = c("Zippin", "alternative"), conf.level = 0.95,
  just.ests = FALSE, Tmult = 3, CIMicroFish = FALSE)

## S3 method for class 'removal'
summary(object, parm = c("No", "p", "p1"),
  digits = getOption("digits"), verbose = FALSE, ...)

## S3 method for class 'removal'
confint(object, parm = c("No", "p"),
  level = conf.level, conf.level = NULL,
  digits = getOption("digits"), verbose = FALSE, ...)
```

Arguments

`catch`	A numerical vector of catch at each pass.
`method`	A single string that identifies the removal method to use. See details.
`alpha`	A single numeric value for the alpha parameter in the CarleStrub method (default is 1).
`beta`	A single numeric value for the beta parameter in the CarleStrub method (default is 1).
`CS.se`	A single string that identifies whether the SE in the CarleStrub method should be computed according to Seber or Zippin.
`conf.level`	A single number representing the level of confidence to use for constructing confidence intervals. This is sent in the main `removal` function rather than `confint`.
`just.ests`	A logical that indicates whether just the estimates (=`TRUE`) or the return list (=`FALSE`; default; see below) is returned.
`Tmult`	A single numeric that will be multiplied by the total catch in all samples to set the upper value for the range of population sizes when minimizing the log-likelihood and creating confidence intervals for the Moran and Schnute methods. Large values are much slower to compute, but values that are too low may result in missing the best estimate. A warning is issued if too low of a value is suspected.

CIMicroFish	A logical that indicates whether the t value used to calculate confidence intervals when method="Burnham" should be rounded to two or three decimals and whether the confidence intervals for No should be rounded to whole numbers as done in MicroFish 3.0. The default (=FALSE) is to NOT round the t values or No confidence interval. This option is provided only so that results will exactly match MicroFish results (see testing).
object	An object saved from removal().
parm	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
digits	A single numeric that controls the number of decimals in the output from summary and confint.
verbose	A logical that indicates whether descriptive labels should be printed from summary and if certain warnings are shown with confint.
...	Additional arguments for methods.
level	Not used, but here for compatibility with generic confint function.

Details

The main function computes the estimates and associated standard errors, if possible, for the initial population size, No, and probability of capture, p, for eight methods chosen with method=. The possible methods are:

- method="CarleStrub": The general weighted k-pass estimator proposed by Carle and Strub (1978). This function iteratively solves for No in equation 7 of Carle and Strub (1978).
- method="Zippin": The general k-pass estimator generally attributed to Zippin. This function iteratively solves for No in bias corrected version of equation 3 (page 622) of Carle and Strub (1978). These results are not yet trustworthy (see Testing section below).
- method="Seber3": The special case for k=3 estimator shown in equation 7.24 of Seber(2002).
- method="Seber2": The special case for k=2 estimator shown on page 312 of Seber(2002).
- method="RobsonRegier2": The special case for k=2 estimator shown by Robson and Regier (1968).
- method="Moran": The likelihood method of Moran (1951) as implemented by Schnute (1983).
- method="Schnute": The likelihood method of Schnute (1983) for the model that has a different probability of capture for the first sample but a constant probability of capture for all ensuing samples.
- method="Burnham": The general k-pass estimator likelihood method created by Ken Burnham and presented by Van Deventer and Platts (1983). This method is used in the Microfish software (Van Deventer 1989).

Confidence intervals for the first five methods are computed using standard large-sample normal distribution theory. Note that the confidence intervals for the 2- and 3-pass special cases are only approximately correct if the estimated population size is greater than 200. If the estimated population size is between 50 and 200 then a 95% CI behaves more like a 90% CI.

Confidence intervals for the next two methods use likelihood ratio theory as described in Schnute (1983) and are only produced for the No parameter. Standard errors are not produced with the Moran or Schnute methods.

Confidence intervals for the last method are computed as per Ken Burnham's instructions for the Burnham Method (Jack Van Deventer, personal communication). Specifically, they are calculated with the t-statistic and No-1 degrees of freedom. Please note that the MicroFish software rounds the t-statistic before it calculates the confidence intervals about No and p. If you need the confidence interals produced by FSA::removal to duplicate MicroFish, please use CIMicroFish=TRUE.

Value

A vector that contains the estimates and standard errors for No and p if `just.ests=TRUE` or (default) a list with at least the following items:

- catch The original vector of observed catches.
- method The method used (provided by the user).
- lbl A descriptive label for the method used.
- est A matrix that contains the estimates and standard errors for No and p.

In addition, if the Moran or Schnute methods are used the list will also contain

- min.nlogLH The minimum value of the negative log-likelihood function.
- Tmult The Tmult value sent by the user.

testing

The Carle-Strub method matches the examples in Carle and Strub (1978) for No, p, and the variance of No. The Carle-Strub estimates of No and p match the examples in Cowx (1983) but the SE of No does not. The Carle-Strub estimates of No match the results (for estimates that they did not reject) from Jones and Stockwell (1995) to within 1 individual in most instances and within 1% for all other instances (e.g., off by 3 individuals when the estimate was 930 individuals).

The Seber3 results for No match the results in Cowx (1983).

The Seber2 results for No, p, and the SE of No match the results in example 7.4 of Seber (2002) and in Cowx (1983).

The RobsonRegier2 results for No and the SE of NO match the results in Cowx (1983)

The Zippin method results do not match the examples in Seber (2002) or Cowx (1983) because `removal` uses the bias-corrected version from Carle and Strub (1978) and does not use the tables in Zippin (1958). The Zippin method is not yet trustworthy.

The Moran and Schnute methods match the examples in Schnute (1983) perfectly for all point estimates and within 0.1 units for all confidence intervals.

The Burnham method was tested against the free (gratis) Demo Version of MicroFish 3.0. Powell Wheeler used R to simulate 100, three-pass removal samples with capture probabilities between 0 and 1 and population sizes <= 1000. The Burnham method implemented here exactly matched MicroFish in all 100 trials for No and p. In addition, the CIs for No exactly matched all 100 trials when CIMicroFish=TRUE. Powell was not able to check the CIs for p because the MicroFish 'Quick Population Estimate' does not report them.

IFAR Chapter

10-Abundance from Depletion Data.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

A. Powell Wheeler, <powell.wheeler@gmail.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Carle, F.L. and M.R. Strub. 1978. A new method for estimating population size from removal data. Biometrics, 34:621-630.

Cowx, I.G. 1983. Review of the methods for estimating fish population size from survey removal data. Fisheries Management, 14:67-82.

Moran, P.A.P. 1951. A mathematical theory of animal trapping. Biometrika 38:307-311.

Robson, D.S., and H.A. Regier. 1968. Estimation of population number and mortality rates. pp. 124-158 in Ricker, W.E. (editor) Methods for Assessment of Fish Production in Fresh Waters. IBP Handbook NO. 3 Blackwell Scientific Publications, Oxford.

Schnute, J. 1983. A new approach to estimating populations by the removal method. Canadian Journal of Fisheries and Aquatic Sciences, 40:2153-2169.

Seber, G.A.F. 2002. The Estimation of Animal Abundance. Edward Arnold, second edition (Reprint).

van Dishoeck, P. 2009. Effects of catchability variation on performance of depletion estimators: Application to an adaptive management experiment. Masters Thesis, Simon Fraser University. [Was (is?) from http://rem-main.rem.sfu.ca/theses/vanDishoeckPier_2009_MRM483.pdf.]

Van Deventer, J.S. 1989. Microcomputer Software System for Generating Population Statistics from Electrofishing Data–User's Guide for MicroFish 3.0. USDA Forest Service, General Technical Report INT-254. 29 p. [Was (is?) from https://relicensing.pcwa.net/documents/Library/PCWA-L

Van Deventer, J.S., and W.S. Platts. 1983. Sampling and estimating fish populations from streams. Transactions of the 48th North American Wildlife and Natural Resource Conference. pp. 349-354.

See Also

See `depletion` for related functionality.

Examples

```
## First example -- 3 passes
ct3 <- c(77,50,37)

# Carle Strub (default) method
p1 <- removal(ct3)
summary(p1)
summary(p1,verbose=TRUE)
summary(p1,parm="No")
summary(p1,parm="p")
```

```
confint(p1)
confint(p1,parm="No")
confint(p1,parm="p")

# Moran method
p2 <- removal(ct3,method="Moran")
summary(p2,verbose=TRUE)
confint(p2,verbose=TRUE)
#'
# Schnute method
p3 <- removal(ct3,method="Schnute")
summary(p3,verbose=TRUE)
confint(p3,verbose=TRUE)

# Burnham method
p4 <- removal(ct3,method="Burnham")
summary(p4)
summary(p4,verbose=TRUE)
summary(p4,parm="No")
summary(p4,parm="p")
confint(p4)
confint(p4,parm="No")
confint(p4,parm="p")
## Second example -- 2 passes
ct2 <- c(77,37)

# Seber method
p4 <- removal(ct2,method="Seber2")
summary(p4,verbose=TRUE)
confint(p4)


### Test if catchability differs between first sample and the other samples
# chi-square test statistic from  negative log-likelihoods
#   from Moran and Schnute fits (from above)
chi2.val <- 2*(p2$min.nlogLH-p3$min.nlogLH)
# p-value ... no significant difference
pchisq(chi2.val,df=1,lower.tail=FALSE)

# Another LRT example ... sample 1 from Schnute (1983)
ct4 <- c(45,11,18,8)
p2a <- removal(ct4,method="Moran")
p3a <- removal(ct4,method="Schnute")
chi2.val <- 2*(p2a$min.nlogLH-p3a$min.nlogLH)  # 4.74 in Schnute(1983)
pchisq(chi2.val,df=1,lower.tail=FALSE)       # significant difference (catchability differs)
summary(p3a)


### Using lapply() to use removal() on many different groups
###   with the removals in a single variable ("long format")
## create a dummy data frame
lake <- factor(rep(c("Ash Tree","Bark","Clay"),each=5))
year <- factor(rep(c("2010","2011","2010","2011","2010","2011"),times=c(2,3,3,2,2,3)))
```

```
pass <- factor(c(1,2,1,2,3,1,2,3,1,2,1,2,1,2,3))
catch <- c(57,34,65,34,12,54,26,9,54,27,67,34,68,35,12)
d <- data.frame(lake,year,pass,catch)

## create a variable that indicates each different group
d$group <- with(d,interaction(lake,year))
d
## split the catch by the different groups (creates a list of catch vectors)
ds <- split(d$catch,d$group)
## apply removal() to each catch vector (i.e., different group)
res <- lapply(ds,removal,just.ests=TRUE)
res <- data.frame(t(data.frame(res,check.names=FALSE)))
## get rownames from above and split into separate columns
nms <- t(data.frame(strsplit(rownames(res),"\\.")))
attr(nms,"dimnames") <- NULL
fnl <- data.frame(nms,res)
## put names together with values
rownames(fnl) <- NULL
colnames(fnl)[1:2] <- c("Lake","Year")
fnl


### Using apply() to use removal() on many different groups
###   with the removals in several variables ("wide format")
## create a dummy data frame (just reshaped from above as
## an example; -5 to ignore the group variable from above)
d1 <- reshape(d[,-5],timevar="pass",idvar=c("lake","year"),direction="wide")
## apply restore() to each row of only the catch data
res1 <- apply(d1[,3:5],MARGIN=1,FUN=removal,method="CarleStrub",just.ests=TRUE)
res1 <- data.frame(t(data.frame(res1,check.names=FALSE)))
## add the grouping information to the results
fnl1 <- data.frame(d1[,1:2],res1)
## put names together with values
rownames(fnl1) <- NULL
fnl1
```

`residPlot` *Construct a residual plot from lm or nls objects.*

Description

Constructs a residual plot for `lm` or `nls` objects. Different symbols for different groups can be added to the plot if an indicator variable regression is used.

Usage

```
residPlot(object, ...)

## S3 method for class 'lm'
```

```
residPlot(object, ...)

## S3 method for class 'SLR'
residPlot(object, xlab = "Fitted Values",
  ylab = "Residuals", main = "", pch = 16, col = "black",
  lty.ref = 3, lwd.ref = 1, col.ref = "black",
  resid.type = c("raw", "standardized", "studentized"),
  outlier.test = TRUE, alpha = 0.05, loess = FALSE, lty.loess = 2,
  lwd.loess = 1, col.loess = "black", trans.loess = 8,
  inclHist = TRUE, ...)

## S3 method for class 'POLY'
residPlot(object, ...)

## S3 method for class 'IVR'
residPlot(object, legend = "topright", cex.leg = 1,
  box.lty.leg = 0, ...)

## S3 method for class 'ONEWAY'
residPlot(object, xlab = "Fitted Values",
  ylab = "Residuals", main = "", pch = 16, col = "black",
  lty.ref = 3, lwd.ref = 1, col.ref = "black",
  resid.type = c("raw", "standardized", "studentized"), bp = TRUE,
  outlier.test = TRUE, alpha = 0.05, loess = FALSE, lty.loess = 2,
  lwd.loess = 1, col.loess = "black", trans.loess = 8,
  inclHist = TRUE, ...)

## S3 method for class 'TWOWAY'
residPlot(object, xlab = "Fitted Values",
  ylab = "Residuals", main = "", pch = 16, col = "black",
  lty.ref = 3, lwd.ref = 1, col.ref = "black",
  resid.type = c("raw", "standardized", "studentized"), bp = TRUE,
  outlier.test = TRUE, alpha = 0.05, loess = FALSE, lty.loess = 2,
  lwd.loess = 1, col.loess = "black", trans.loess = 8,
  inclHist = TRUE, ...)

## S3 method for class 'nls'
residPlot(object, xlab = "Fitted Values",
  ylab = "Residuals", main = "", pch = 16, col = "black",
  lty.ref = 3, lwd.ref = 1, col.ref = "black",
  resid.type = c("raw", "standardized", "studentized"), loess = FALSE,
  lty.loess = 2, lwd.loess = 1, col.loess = "black",
  trans.loess = 8, inclHist = TRUE, ...)

## S3 method for class 'nlme'
residPlot(object, xlab = "Fitted Values",
  ylab = "Residuals", main = "", pch = 16, col = "black",
  lty.ref = 3, lwd.ref = 1, col.ref = "black",
```

```
  resid.type = c("raw", "standardized", "studentized"), loess = FALSE,
  lty.loess = 2, lwd.loess = 1, col.loess = "black",
  trans.loess = 8, inclHist = TRUE, ...)
```

Arguments

object	An lm or nls object (i.e., returned from fitting a model with either lm or nls).
...	Other arguments to the generic plot function.
xlab	A string for labeling the x-axis.
ylab	A string for labeling the y-axis.
main	A string for the main label to the plot. See details.
pch	A numeric that indicates the plotting character to be used or a vector of numerics that indicates what plotting character codes to use for the levels of the second factor. See par.
col	A vector of color names that indicates what color of points and lines to use for the levels of the first factor. See par.
lty.ref	A numeric that indicates the line type to use for the reference line at residual=0. See par.
lwd.ref	A numeric that indicates the line width to use for the reference line at residual=0. See par.
col.ref	A numeric or character that indicates the line color to use for the reference line at residual=0. See par.
resid.type	The type of residual to use. 'Raw' residuals are used by default. See details.
outlier.test	A logical that indicates if an outlierTest will TRUE (default) be performed and if the individual with the largest studentized residual is deemed to be a significant outlier it will be noted on the residual plot by its observation number.
alpha	A numeric that indicates the alpha level to use for the outlier test (only used if outlier.test=TRUE).
loess	A logical that indicates if a loess smoother line and approximate confidence interval band is fit to and shown on the residual plot (TRUE).
lty.loess	A numeric that indicates the line type to use for loess fit line. See par.
lwd.loess	A numeric that indicates the line width to use for loess fit line. See par.
col.loess	A numeric or character that indicates the line color to use for loess fit line. See par.
trans.loess	A single numeric that indicates how transparent the loess band should be (larger numbers are more transparent).
inclHist	A logical that indicates if a second pane that includes the histogram of residuals should be constructed.
legend	If TRUE, draw a legend and the user must click in the upper-left corner of where the legend should be placed; if FALSE do not draw a legend. If a vector of length 2 then draw the upper left corner of the legend at the coordinates given in the vector of length 2.

cex.leg	A single numeric values used to represent the character expansion value for the legend. Ignored if legend=FALSE.
box.lty.leg	A single numeric values used to indicate the type of line to use for the box around the legend. The default is to not plot a box.
bp	A logical that indicates if the plot for the one-way and two-way ANOVA will be a boxplot (TRUE; default) or not.

Details

Three types of residuals are allowed for most model types. Raw residuals are simply the difference between the observed response variable and the predicted/fitted value. Standardized residuals are internally studentized residuals returned by rstandard for linear models and are the raw residual divided by the standard deviation of the residuals for nonlinear models (as is done by nlsResiduals from **nlstools**). Studentized residuals are the externally studentized residuals returned by rstudent for linear models and are not available for nonlinear models.

Externally Studentized residuals are not supported for nls or nlme objects.

If outlier.test=TRUE then significant outliers are detected with outlierTest from the **car** package. See the help for this function for more details.

The user can include the model call as a title to the residual plot by using main="MODEL". This only works for models created with lm().

If the user chooses to add a legend without identifying coordinates for the upper-left corner of the legend (i.e., legend=TRUE) then the R console is suspended until the user places the legend by clicking on the produced graphic at the point where the upper-left corner of the legend should appear. A legend will only be placed if the mdl is an indicator variable regression, even if legend=TRUE.

Value

None. However, a residual plot is produced.

Note

This function is meant to allow newbie students the ability to easily construct residual plots for one-way ANOVA, two-way ANOVA, simple linear regression, and indicator variable regressions. The plots can be constructed by submitting a saved linear model to this function which allows students to interact with and visualize moderately complex linear models in a fairly easy and efficient manner.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See residualPlots in **car** and nlsResiduals in **nlstools**) for similar functionality and fitPlot and outlierTest in **car** for related methods.

Examples

```
# create year factor variable & reduce number of years for visual simplicity
Mirex$fyear <- factor(Mirex$year)
Mirex2 <- filterD(Mirex,fyear %in% c(1977,1992))

## Indicator variable regression with two factors
lm1 <- lm(mirex~weight*fyear*species,data=Mirex2)
# defaults
residPlot(lm1)
# remove the histogram
residPlot(lm1,inclHist=FALSE)
# add the loess line
residPlot(lm1,loess=TRUE,inclHist=FALSE)
# modify colors used
residPlot(lm1,col="rainbow",inclHist=FALSE)
# use only one point type -- notice that all points are of same type
residPlot(lm1,pch=16,inclHist=FALSE)
# use only one point and one color (might as well not use legend also)
residPlot(lm1,pch=16,col="black",legend=FALSE,inclHist=FALSE)
# can accomplish same thing just by removing the legend
residPlot(lm1,legend=FALSE,inclHist=FALSE)
# modify the reference line
residPlot(lm1,col.ref="blue",lwd.ref=5,inclHist=FALSE)
# include model in the title
residPlot(lm1,main="MODEL")
# use Studentized residuals
residPlot(lm1,resid.type="studentized",inclHist=FALSE)
# use Standardized residuals
residPlot(lm1,resid.type="standardized",inclHist=FALSE)

## Indicator variable regression with same two factors but in different order
##   (notice use of colors and symbols)
lm1a <- lm(mirex~weight*species*fyear,data=Mirex2)
residPlot(lm1a)

## Indicator variable regression with only one factor
lm2 <- lm(mirex~weight*fyear,data=Mirex)
residPlot(lm2)
residPlot(lm2,inclHist=FALSE)
residPlot(lm2,inclHist=FALSE,pch=19)
residPlot(lm2,inclHist=FALSE,pch=19,col="black")
residPlot(lm2,inclHist=FALSE,legend=FALSE)
residPlot(lm2,inclHist=FALSE,pch=2,col="red",legend=FALSE)

## Indicator variable regression (assuming same slope)
lm3 <- lm(mirex~weight+fyear,data=Mirex)
residPlot(lm3)

## Simple linear regression
lm4 <- lm(mirex~weight,data=Mirex)
residPlot(lm4)
```

```
## One-way ANOVA
lm5 <- lm(mirex~fyear,data=Mirex)
# default (uses boxplots)
residPlot(lm5)
# use points rather than boxplot
residPlot(lm5,bp=FALSE)

## Two-Way ANOVA
lm6 <- lm(mirex~species*fyear,data=Mirex)
# default (uses boxplots)
residPlot(lm6)
# No boxplots
residPlot(lm6,bp=FALSE)

## Examples showing outlier detection
x <- c(runif(100))
y <- c(7,runif(99))
lma <- lm(y~x)
residPlot(lma)
# with studentized residuals
residPlot(lma,resid.type="studentized")
# multiple outliers
y <- c(7,runif(98),-5)
lmb <- lm(y~x)
residPlot(lmb)
# check that NAs are handled properly ... label should be 100
y <- c(NA,NA,runif(97),7)
lmc <- lm(y~x)
residPlot(lmc)

## Nonlinear regression
# from first example in nls()
DNase1 <- subset(DNase,Run==1)
fm1DNase1 <- nls(density~SSlogis(log(conc),Asym,xmid,scal),DNase1)
residPlot(fm1DNase1)
residPlot(fm1DNase1,resid.type="standardized")
```

`rSquared` *Extract the coefficient of determination from a linear model object.*

Description

Extracts the coefficient of determination (i.e., "r-squared") from a linear model (i.e., `lm`) object.

Usage

```
rSquared(object, ...)

## Default S3 method:
```

```
rSquared(object, ...)

## S3 method for class 'lm'
rSquared(object, digits = getOption("digits"),
  percent = FALSE, ...)
```

Arguments

object	An object saved from lm.
...	Additional arguments for methods.
digits	A single number that is the number of digits to round the returned result to.
percent	A logical that indicates if the result should be returned as a percentage (=TRUE) or as a proportion (=FALSE; default).

Details

This is a convenience function to extract the r.squared part from summary(lm).

Value

A numeric, as either a proportion or percentage, that is the coefficient of determination for a linear model.

Examples

```
lm1 <- lm(mirex~weight, data=Mirex)
rSquared(lm1)
rSquared(lm1,digits=3)
rSquared(lm1,digits=1,percent=TRUE)

## rSquared only works with lm objects
## Not run:
nls1 <- nls(mirex~a*weight^b,data=Mirex,start=list(a=1,b=1))
rSquared(nls1)

## End(Not run)
```

Schnute *The four-parameter growth function from Schnute (1981).*

Description

The four-parameter growth function from Schnute (1981).

Usage

```
Schnute(t, case = 1, t1 = NULL, t3 = NULL, L1 = NULL, L3 = NULL,
  a = NULL, b = NULL)
```

Arguments

t	A numeric vector of ages over which to model growth.
case	A string that indicates the case of the Schnute growth function to use.
t1	The (young) age that corresponds to L1. Set to minimum value in t by default.
t3	The (old) age that corresponds to L3. Set to maximum value in t by default.
L1	The mean size/length at t1.
L3	The mean size/length at t3.
a	A dimensionless parameter that is related to the time/age at the inflection point.
b	A dimensionless parameter that is related to size/length at the inflection point.

Value

Schnute returns a predicted size given the case of the function and the provided parameter values.

SchnuteModels returns a graphic that uses plotmath to show the growth function equation in a pretty format.

IFAR Chapter

None specifically, but 12-Individual Growth is related.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Schnute, J. 1981. A versatile growth model with statistical stable parameters. Canadian Journal of Fisheries and Aquatic Sciences 38:1128-1140.

See Also

See vbFuns, GompertzFuns, RichardsFuns, and logisticFuns for similar functionality for other models.

Examples

```
## See the formulae
growthFunShow("Schnute",1,plot=TRUE)
growthFunShow("Schnute",2,plot=TRUE)
growthFunShow("Schnute",3,plot=TRUE)
growthFunShow("Schnute",4,plot=TRUE)

## Simple examples
ages <- 1:15
s1 <- Schnute(ages,case=1,t1=1,t3=15,L1=30,L3=400,a=0.3,b=1)
s2 <- Schnute(ages,case=2,t1=1,t3=15,L1=30,L3=400,a=0.3,b=1)
s3 <- Schnute(ages,case=3,t1=1,t3=15,L1=30,L3=400,a=0.3,b=1)
s4 <- Schnute(ages,case=4,t1=1,t3=15,L1=30,L3=400,a=0.3,b=1)
```

```
plot(s1~ages,type="l",lwd=2)
lines(s2~ages,lwd=2,col="red")
lines(s3~ages,lwd=2,col="blue")
lines(s4~ages,lwd=2,col="green")
```

---

se *Computes standard error of the mean.*

Description

Computes the standard error of the mean (i.e., standard deviation divided by the square root of the sample size).

Usage

```
se(x, na.rm = TRUE)
```

Arguments

x	A numeric vector.
na.rm	A logical that indicates whether missing values should be removed before computing the standard error.

Value

A single numeric that is the standard error of the mean of x.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See se in **sciplot** for similar functionality.

Examples

```
x <- 1:20
sd(x)/sqrt(length(x))
se(x)

# all return NA if missing values are not removed
x2 <- c(x,NA)
sd(x2)/sqrt(length(x2))

# Better if missing values are removed
se(x2,na.rm=FALSE)
```

```
sd(x2,na.rm=TRUE)/sqrt(length(x2[complete.cases(x2)]))
se(x2)
```

SMBassLS *Catch-effort data for Little Silver Lake (Ont) Smallmouth Bass.*

Description

Catch-effort data for Smallmouth Bass (*Micropterus dolomieu*) in Little Silver Lake, Ont.

Format

A data frame with 10 observations on the following 3 variables:

day Day of the catch

catch Number of smallmouth bass caught

effort Number of traps set per day

Topic(s)

- Population size
- Abundance
- Depletion methods
- Leslie method
- DeLury method
- Catchability

Source

From Omand, D.N. 1951. A study of populations of fish based on catch-effort statistics. Journal of Wildlife Management, 15:88-98.

See Also

Used in depletion examples.

Examples

```
str(SMBassLS)
head(SMBassLS)
```

SMBassWB *Growth increment data for West Bearskin Lake, MN, Smallmouth Bass.*

Description

Growth data from Smallmouth Bass (*Micropterus dolomieu*) captured in West Bearskin Lake, MN. Five samples were collected over three years (1988-1990) with two gears (fall – trapnets, spring – electrofishing).

Format

A data frame of 445 observations on the following 20 variables:

species Species of the fish (`SMB` for each fish in this file)

lake Lake fish was captured in (`WB` for each fish in this file)

gear Gear used to capture the fish (`T`=Trapnet and `E`=Electrofishing)

yearcap Year fish was captured (`1988`, `1989`, or `1990`)

fish A unique identifier for each fish

agecap Assigned age-at-capture for the fish (from scales)

lencap Total length-at-capture for the fish (mm)

anu1 Magnified scale radius (mm) to the 1st annulus

anu2 Magnified scale radius (mm) to the 2nd annulus

anu3 Magnified scale radius (mm) to the 3rd annulus

anu4 Magnified scale radius (mm) to the 4th annulus

anu5 Magnified scale radius (mm) to the 5th annulus

anu6 Magnified scale radius (mm) to the 6th annulus

anu7 Magnified scale radius (mm) to the 7th annulus

anu8 Magnified scale radius (mm) to the 8th annulus

anu9 Magnified scale radius (mm) to the 9th annulus

anu10 Magnified scale radius (mm) to the 10th annulus

anu11 Magnified scale radius (mm) to the 11th annulus

anu12 Magnified scale radius (mm) to the 12th annulus

radcap Total scale radius at time of capture

Topic(s)

- Growth increment analysis
- Weisberg linear growth model
- Back-Calculation

Note

Data are in one-fish-per-line format.

Source

Data from the linear growth modeling software distributed in support of Weisberg, S. 1993. Using hard-part increment data to estimate age and environmental effects. Canadian Journal of Fisheries and Aquatic Sciences 50:1229-1237.

See Also

Used in `capHistSum` and `mrClosed` examples. Also see `wblake` from **alr4** for the same dataset with only the `agecap`, `lencap`, and `radcap` variables.

Examples

```
str(SMBassWB)
head(SMBassWB)
```

SpotVA1 *Age and length of spot.*

Description

Ages (from otoliths) and lengths of Virginia Spot (*Leiostomus xanthurus*).

Format

A data frame of 403 observations on the following 2 variables:

tl Measured total lengths (in inches)

age Ages assigned from examination of otoliths

Details

Final length measurements were simulated by adding a uniform error to the value at the beginning of the length category.

Topic(s)

- Growth
- von Bertalanffy

Source

Extracted from Table 1 in Chapter 8 (Spot) of the VMRC Final Report on Finfish Ageing, 2002 by the Center for Quantitative Fisheries Ecology at Old Dominion University.

See Also

Used in vbFuns, vbStarts, and nlsTracePlot examples. Also see SpotVA2 in **FSAdata** for related data.

Examples

```
str(SpotVA1)
head(SpotVA1)
plot(tl~age,data=SpotVA1)
```

---

srStarts *Finds reasonable starting values for parameters in specific parameterizations of common stock-recruitment models.*

Description

Finds reasonable starting values for parameters in specific parameterizations of the "Beverton-Holt", "Ricker", "Shepherd", or "Saila-Lorda" stock-recruitment models. Use `srFunShow()` to see the equations of each model.

Usage

```
srStarts(formula, data = NULL, type = c("BevertonHolt", "Ricker",
  "Shepherd", "SailaLorda", "independence"), param = 1, fixed = NULL,
  plot = FALSE, col.mdl = "gray70", lwd.mdl = 3, lty.mdl = 1,
  cex.main = 0.9, col.main = "red", dynamicPlot = FALSE, ...)
```

Arguments

`formula`	A formula of the form `Recruits~Stock`.
`data`	A data frame in which `Recruits` and `Stock` are found.
`type`	A string that indicates the type of the stock-recruitment model. Must be one of `"BevertonHolt"`, `"Ricker"`, `"Shepherd"`, or `"SailaLorda"`.
`param`	A numeric that indicates the parameterization of the stock-recruitment model type. This is ignored if `type="Shepherd"` or `type="SailaLorda"`
`fixed`	A named list that contains user-defined rather than automatically generated (i.e., fixed) starting values for one or more parameters. See details.
`plot`	A logical that indicates whether or not a plot of the data with the model fit at the starting values superimposed is created.
`col.mdl`	A color for the model when `plot=TRUE`.
`lwd.mdl`	A line width for the model when `plot=TRUE`.
`lty.mdl`	A line type for the model when `plot=TRUE`.
`cex.main`	A character expansion value for the main title when `plot=TRUE`.

col.main	A color for the main title when plot=TRUE.
dynamicPlot	DEPRECATED.
...	Further arguments passed to the methods.

Details

This function attempts to find reasonable starting values for a variety of parameterizations of the "Beverton-Holt", "Ricker", "Shepherd", or "Saila-Lorda" stock-recruitment models. There is no guarantee that these starting values are the 'best' starting values. One should use them with caution and should perform sensitivity analyses to determine the impact of different starting values on the final model results.

Starting values for the first parameterization of the Beverton-Holt model were derived by linearizing the function (inverting both sides and simplifying), fitting a linear model to the observed data, and extracting parameter values from the corresponding linear model parameters. Starting values for the other parameterizations of the Beverton-Holt model were derived from known relationships between the parameters of each parameterization and the first parameterization. If the computed starting value for the Rp parameter was larger than the largest observed recruitment value, then the starting value for Rp was set to the largest observed recruitment value.

Starting values for the Shepherd function were the same as those for the first parameterization of the Beverton-Holt function with the addition that c=1.

Starting values for the Ricker parameterizations followed the same general procedure as described for the Beverton-Holt parameterizations. If the computed starting value for atilde was less than zero then the starting value was set to 0.00001.

Starting values for the Saila-Lorda function were the same as those for the first parameterization of the Ricker function with the addition that c=1.

Value

A list that contains reasonable starting values. Note that the parameters will be listed in the same order and with the same names as listed in srFuns.

IFAR Chapter

13-Recruitment.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Beverton, R.J.H. and S.J. Holt. 1957. On the dynamics of exploited fish populations, Fisheries Investigations (Series 2), volume 19. United Kingdom Ministry of Agriculture and Fisheries, 533 pp.

Iles, T.C. 1994. A review of stock-recruitment relationships with reference to flatfish populations. Netherlands Journal of Sea Research 32:399-420.

Quinn II, T.J. and R.B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press.

Ricker, W.E. 1954. Stock and recruitment. Journal of the Fisheries Research Board of Canada 11:559-623.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Shepherd, J. 1982. A versatile new stock-recruitment relationship for fisheries and construction of sustainable yield curves. Journal du Conseil International pour l'Exploration de la Mar 40:67-75.

See Also

See `srFunShow` and `srFuns` for related functionality. See `nlsTracePlot` for help troubleshooting nonlinear models that don't converge.

Examples

```
## Simple Examples
srStarts(recruits~stock,data=CodNorwegian)
srStarts(recruits~stock,data=CodNorwegian,param=2)
srStarts(recruits~stock,data=CodNorwegian,param=3)
srStarts(recruits~stock,data=CodNorwegian,param=4)
srStarts(recruits~stock,data=CodNorwegian,type="Ricker")
srStarts(recruits~stock,data=CodNorwegian,type="Ricker",param=2)
srStarts(recruits~stock,data=CodNorwegian,type="Ricker",param=3)
srStarts(recruits~stock,data=CodNorwegian,type="Shepherd")
srStarts(recruits~stock,data=CodNorwegian,type="SailaLorda")
srStarts(recruits~stock,data=CodNorwegian,type="independence")

## Simple Examples with a Plot
srStarts(recruits~stock,data=CodNorwegian,type="Ricker",plot=TRUE)
srStarts(recruits~stock,data=CodNorwegian,type="BevertonHolt",plot=TRUE)
srStarts(recruits~stock,data=CodNorwegian,type="Shepherd",plot=TRUE)
srStarts(recruits~stock,data=CodNorwegian,type="SailaLorda",plot=TRUE)
srStarts(recruits~stock,data=CodNorwegian,type="independence",plot=TRUE)

## See examples in srFuns() for use of srStarts() when fitting stock-recruit models
```

`stockRecruitment` *Creates a function for a specific parameterization of a common stock-recruitment function .*

Description

Creates a function for a specific parameterization of a "Beverton-Holt", "Ricker", "Shepherd", or "Saila-Lorda" stock-recruitment function. Use `srFunShow()` to see the equations of each function.

Usage

```
srFuns(type = c("BevertonHolt", "Ricker", "Shepherd", "SailaLorda",
  "independence"), param = 1, simple = FALSE, msg = FALSE)

srFunShow(type = c("BevertonHolt", "Ricker", "Shepherd", "SailaLorda"),
  param = 1, plot = FALSE, ...)
```

Arguments

type	A string that indicates the type of stock-recruitment function.
param	A single numeric that indicates the parameterization of the stock-recruitment function.
simple	A logical that indicates whether the user should be allowed to send all parameter values in the first parameter argument (=FALSE; default) or whether all individual parameters must be specified (=TRUE).
msg	A logical that indicates whether a message about the function and parameter definitions should be output (=TRUE) or not (=FALSE; default).
plot	A logical that indicates whether the growth function expression should be shown as an equation in a simple plot.
...	Not implemented.

Value

srFuns returns a function that can be used to predict recruitment given a vector of stock sizes and values for the function parameters. The result should be saved to an object that can then be used as a function name. When the resulting function is used, the parameters are ordered as shown when the definitions of the parameters are printed after the function is called (assuming that msg=TRUE). The values for both/all parameters can be included as a vector of length two/three in the first parameter argument. If simple=FALSE then the values for all parameters can be included as a vector in the first parameter argument. If simple=TRUE then all parameters must be declared individually in each function. The resulting function is somewhat easier to read when simple=TRUE.

srFunShow returns an expression that can be use with plotmath to show the function equation in a pretty format. See examples.

IFAR Chapter

13-Recruitment.

Author(s)

Derek H. Ogle, <derek@derekogle.com>, thanks to Gabor Grothendieck for a hint about using get().

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

Beverton, R.J.H. and S.J. Holt. 1957. On the dynamics of exploited fish populations, Fisheries Investigations (Series 2), volume 19. United Kingdom Ministry of Agriculture and Fisheries, 533 pp.

Iles, T.C. 1994. A review of stock-recruitment relationships with reference to flatfish populations. Netherlands Journal of Sea Research 32:399-420.

Quinn II, T.J. and R.B. Deriso. 1999. Quantitative Fish Dynamics. Oxford University Press.

Ricker, W.E. 1954. Stock and recruitment. Journal of the Fisheries Research Board of Canada 11:559-623.

Ricker, W.E. 1975. Computation and interpretation of biological statistics of fish populations. Technical Report Bulletin 191, Bulletin of the Fisheries Research Board of Canada. [Was (is?) from http://www.dfo-mpo.gc.ca/Library/1485.pdf.]

Shepherd, J. 1982. A versatile new stock-recruitment relationship for fisheries and construction of sustainable yield curves. Journal du Conseil International pour l'Exploration de la Mar 40:67-75.

See Also

See srStarts for related functionality.

Examples

```
## See the formulae
## Simple Examples
# show what a message looks like with the function definition
srFuns("Ricker",msg=TRUE)

# create some dummy stock data
stock <- seq(0.01,1000,length.out=199)

# Beverton-Holt #1 parameterization
( bh1 <- srFuns() )
plot(bh1(stock,a=0.5,b=0.01)~stock,type="l",lwd=2,ylab="Recruits",xlab="Spawners",ylim=c(0,50))

# Ricker #1 parameterization
( r1 <- srFuns("Ricker") )
lines(r1(stock,a=0.5,b=0.005)~stock,lwd=2,col="red")

# Shephered parameterization
( s1 <- srFuns("Shepherd") )
lines(s1(stock,a=0.5,b=0.005,c=2.5)~stock,lwd=2,col="blue")

# Saila-Lorda parameterization
( sl1 <- srFuns("SailaLorda") )
lines(sl1(stock,a=0.5,b=0.005,c=1.05)~stock,lwd=2,col="salmon")

## Examples of fitting stock-recruitment functions
# Fitting the Beverton-Holt #1 parameterization with multiplicative errors
bh1s <- srStarts(recruits~stock,data=CodNorwegian)
```

```
fit1 <- nls(log(recruits)~log(bh1(stock,a,b)),data=CodNorwegian,start=bh1s)
summary(fit1,correlation=TRUE)
plot(recruits~stock,data=CodNorwegian,pch=19,xlim=c(0,200))
curve(bh1(x,a=coef(fit1)[1],b=coef(fit1)[2]),from=0,to=200,col="red",lwd=3,add=TRUE)

# Fitting the Ricker #3 parameterization with multiplicative errors
r3 <- srFuns("Ricker",param=3)
r3s <- srStarts(recruits~stock,data=CodNorwegian,type="Ricker",param=3)
fit2 <- nls(log(recruits)~log(r3(stock,a,Rp)),data=CodNorwegian,start=r3s)
summary(fit2,correlation=TRUE)
curve(r3(x,a=coef(fit2)[1],Rp=coef(fit2)[2]),from=0,to=200,col="blue",lwd=3,add=TRUE)

################################################################################
## Create expressions of the functions
################################################################################
# Simple example
srFunShow()
srFunShow(plot=TRUE)
srFunShow("BevertonHolt",1,plot=TRUE)

# Get and save the expression
( tmp <- srFunShow("BevertonHolt",1) )
# Use expression as title on a plot
plot(bh1(stock,a=0.5,b=0.01)~stock,type="l",ylim=c(0,50),
     ylab="Recruits",xlab="Spawners",main=tmp)
# Put expression in the main plot
text(800,10,tmp)
# Put multiple expressions on a plot
op <- par(mar=c(0.1,0.1,0.1,0.1))
plot(0,type="n",xlab="",ylab="",xlim=c(0,1),ylim=c(0,3),xaxt="n",yaxt="n")
text(0,2.5,"Beverton-Holt #1:",pos=4)
text(0.5,2.5,srFunShow("BevertonHolt",1))
text(0,1.5,"Ricker #2:",pos=4)
text(0.5,1.5,srFunShow("Ricker",2))
text(0,0.5,"Shepherd:",pos=4)
text(0.5,0.5,srFunShow("Shepherd"))
par(op)
```

Subset *Subsets/filters a data frame and drops the unused levels.*

Description

Subsets/filters a data frame and drops the unused levels.

Usage

```
Subset(x, subset, select, drop = FALSE, resetRownames = TRUE, ...)

filterD(x, ..., except = NULL)
```

Arguments

`x`	A data frame.
`subset`	A logical expression that indicates elements or rows to keep: missing values are taken as false.
`select`	An expression, that indicates columns to select from a data frame.
`drop`	passed on to `[` indexing operator.
`resetRownames`	A logical that indicates if the rownames should be reset after the subsetting (`TRUE`; default). Resetting rownames will simply number the rows from 1 to the number of rows in the result.
`...`	further arguments to be passed to or from other methods.
`except`	Indices of columns from which NOT to drop levels.

Details

Newbie students using R expect that when a factor variable is subsetted with `subset` or filtered with `filter` that any original levels that are no longer used after the subsetting or filtering will be ignored. This, however, is not the case and often results in tables with empty cells and figures with empty bars. One remedy is to use `drop.levels` from **gdata** immediately following the `subset` or `filter` call. This generally becomes a repetitive sequence for most newbie students; thus, `Subset` and `filterD` incorporate these two functions into one function.

`Subset` is a wrapper to `subset` with a catch for non-data.frames and a specific call to `drop.levels` just before the data.frame is returned. I also added an argument to allow resetting the row names. `filterD` is a wrapper for `filter` from **dplyr** followed by `drop.levels` just before the data.frame is returned. Otherwise, there is no new code here.

These functions are used only for data frames.

Value

A data frame with the subsetted rows and selected variables.

IFAR Chapter

Basic Data Manipulations.

Author(s)

Derek H. Ogle, `<derek@derekogle.com>`

See Also

See `subset` and `filter` from **dplyr** for similar functionality. See `drop.levels` in **gdata** and `droplevels` for related functionality.

Examples

```
## The problem -- note use of unused level in the final table.
levels(iris$Species)
iris.set1 <- subset(iris,Species=="setosa" | Species=="versicolor")
levels(iris.set1$Species)
xtabs(~Species,data=iris)

## A simpler fix using Subset
iris.set2 <- Subset(iris,Species=="setosa" | Species=="versicolor")
levels(iris.set2$Species)
xtabs(~Species,data=iris.set2)

## A simpler fix using filterD
iris.set3 <- filterD(iris,Species=="setosa" | Species=="versicolor")
levels(iris.set3$Species)
xtabs(~Species,data=iris.set3)
```

`Summarize` *Summary statistics for a numeric variable.*

Description

Summary statistics for a single numeric variable, possibly separated by the levels of a factor variable or variables. This function is very similar to `summary` for a numeric variable.

Usage

```
Summarize(object, ...)

## Default S3 method:
Summarize(object, digits = getOption("digits"),
  na.rm = TRUE, exclude = NULL, nvalid = c("different", "always",
  "never"), percZero = c("different", "always", "never"), ...)

## S3 method for class 'formula'
Summarize(object, data = NULL,
  digits = getOption("digits"), na.rm = TRUE, exclude = NULL,
  nvalid = c("different", "always", "never"), percZero = c("different",
  "always", "never"), ...)
```

Arguments

`object`	A vector of numeric data.
`...`	Not implemented.
`digits`	A single numeric that indicates the number of decimals to round the numeric summaries.

na.rm	A logical that indicates whether numeric missing values (NA) should be removed (=TRUE, default) or not.
exclude	A string that contains the level that should be excluded from a factor variable.
nvalid	A string that indicates how the "validn" result will be handled. If "always" then "validn" will always be shown and if "never" then "validn" will never be shown. However, if "different" (DEFAULT), then "validn" will only be shown if it differs from "n" (or if at least one group differs from "n" when summarized by multiple groups).
percZero	A string that indicates how the "percZero" result will be handled. If "always" then "percZero" will always be shown and if "never" then "percZero" will never be shown. However, if "different" (DEFAULT), then "percZero" will only be shown if it is greater than zero (or if at least one group is greater than zero when summarized by multiple groups).
data	A data.frame that contains the variables in formula.

Details

This function is primarily used with formulas of the following types (where quant and factor generically represent quantitative/numeric and factor variables, respectively):

Formula	Description of Summary
~quant	Numerical summaries (see below) of quant.
quant~factor	Summaries of quant separated by levels in factor.
quant~factor1*factor2	Summaries of quant separated by the combined levels in factor1 and factor2.

Numerical summaries include all results from summary (min, Q1, mean, median, Q3, and max) and the sample size, valid sample size (sample size minus number of NAs), and standard deviation (i.e., sd). NA values are removed from the calculations with na.rm=TRUE (the DEFAULT). The number of digits in the returned results are controlled with digits=.

Value

A named vector or data frame (when a quantitative variable is separated by one or two factor variables) of summary statistics for numeric data.

Note

Students often need to examine basic statistics of a quantitative variable separated for different levels of a categorical variable. These results may be obtained with tapply, by, or aggregate (or with functions in other packages), but the use of these functions is not obvious to newbie students or return results in a format that is not obvious to newbie students. Thus, the formula method to Summarize allows newbie students to use a common notation (i.e., formula) to easily compute summary statistics for a quantitative variable separated by the levels of a factor.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See summary for related one dimensional functionality. See tapply, summaryBy in **doBy**, describe in **psych**, describe in **prettyR**, and basicStats in **fBasics** for similar "by" functionality.

Examples

```
## Create a data.frame of "data"
n <- 102
d <- data.frame(y=c(0,0,NA,NA,NA,runif(n-5)),
                w=sample(7:9,n,replace=TRUE),
                v=sample(0:2,n,replace=TRUE),
                g1=factor(sample(c("A","B","C",NA),n,replace=TRUE)),
                g2=factor(sample(c("male","female","UNKNOWN"),n,replace=TRUE)),
                g3=sample(c("a","b","c","d"),n,replace=TRUE),
                stringsAsFactors=FALSE)

# typical output of summary() for a numeric variable
summary(d$y)

# this function
Summarize(d$y,digits=3)
Summarize(~y,data=d,digits=3)
Summarize(y~1,data=d,digits=3)

# note that nvalid is not shown if there are no NAs and
#   percZero is not shown if there are no zeros
Summarize(~w,data=d,digits=3)
Summarize(~v,data=d,digits=3)

# note that the nvalid and percZero results can be forced to be shown
Summarize(~w,data=d,digits=3,nvalid="always",percZero="always")

## Numeric vector by levels of a factor variable
Summarize(y~g1,data=d,digits=3)
Summarize(y~g2,data=d,digits=3)
Summarize(y~g2,data=d,digits=3,exclude="UNKNOWN")

## Numeric vector by levels of two factor variables
Summarize(y~g1+g2,data=d,digits=3)
Summarize(y~g1+g2,data=d,digits=3,exclude="UNKNOWN")

## What happens if RHS of formula is not a factor
Summarize(y~w,data=d,digits=3)

## Summarizing multiple variables in a data.frame (must reduce to numerics)
lapply(as.list(d[,1:3]),Summarize,digits=4)
```

---

sumTable *Creates a one- or two-way table of summary statistics.*

Description

Creates a one- or two-way table of summary statistics for a quantitative variable.

Usage

```
sumTable(formula, ...)

## S3 method for class 'formula'
sumTable(formula, data = NULL, FUN = mean,
  digits = getOption("digits"), ...)
```

Arguments

formula	A formula with a quantitative variable on the left-hand-side and one or two factor variables on the right-hand-side. See details.
...	Other arguments to pass through to FUN.
data	An optional data frame that contains the variables in formula.
FUN	A scalar function that identifies the summary statistics. Applied to the quantitative variable for all data subsets identified by the combination of the factor(s). Defaults to mean.
digits	A single numeric that indicates the number of digits to be used for the result.

Details

The formula must be of the form quantitative~factor or quantitative~factor*factor2 where quantitative is the quantitative variable to construct the summaries for and factor and factor2 are factor variables that contain the levels for which separate summaries should be constructed. If the variables on the right-hand-side are not factors, then they will be coerced to be factors and a warning will be issued.

This function is largely a wrapper to tapply(), but only works for one quantitative variable on the left-hand-side and one or two factor variables on the right-hand-side. Consider using tapply for situations with more factors on the right-hand-side.

Value

A one-way array of values if only one factor variable is supplied on the right-hand-side of formula. A two-way matrix of values if two factor variables are supplied on the right-hand-side of formula. These are the same classes of objects returned by tapply.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See tapply for a more general implementation. See Summarize for a similar computation when only one factor variable is given.

Examples

```
## The same examples as in the old aggregate.table in gdata package
## but data in data.frame to illustrate formula notation
d <- data.frame(g1=sample(letters[1:5], 1000, replace=TRUE),
                g2=sample(LETTERS[1:3], 1000, replace=TRUE),
                dat=rnorm(1000))

sumTable(dat~g1*g2,data=d,FUN=length)       # get sample size
sumTable(dat~g1*g2,data=d,FUN=validn)       # get sample size (better way)
sumTable(dat~g1*g2,data=d,FUN=mean)         # get mean
sumTable(dat~g1*g2,data=d,FUN=sd)           # get sd
sumTable(dat~g1*g2,data=d,FUN=sd,digits=1)  # show digits= argument

## Also demonstrate use in the 1-way example -- but see Summarize() in FSA package
sumTable(dat~g1,data=d,FUN=validn)
sumTable(dat~g1,data=d,FUN=mean)

## Example with a missing value (compare to above)
d$dat[1] <- NA
sumTable(dat~g1,data=d,FUN=validn)  # note use of validn
sumTable(dat~g1,data=d,FUN=mean,na.rm=TRUE)
```

`tictactoe` *Construct a base tic-tac-toe plot for presenting predator-prey PSD values.*

Description

Construct a base tic-tac-toe plot for presenting predator-prey PSD values. Predator-prey PSD values are added with `plotCI` from **plotrix**.

Usage

```
tictactoe(predobj = c(30, 70), preyobj = c(30, 70),
  predlab = "Predator PSD", preylab = "Prey PSD", obj.col = "black",
  obj.trans = 0.2, bnd.col = "black", bnd.lwd = 1, bnd.lty = 2)
```

Arguments

`predobj`	A vector of length 2 that contains the target objective range for the predator.
`preyobj`	A vector of length 2 that contains the target objective range for the prey.
`predlab`	A string representing a label for the x-axis.
`preylab`	A string representing a label for the y-axis.
`obj.col`	A string designating a color to which the target objective regions should be shaded.

obj.trans	A numeric (decimal) that indicates the level of transparency for marking the target objective regions.
bnd.col	A string that indicates a color for the boundaries of the target objective regions.
bnd.lwd	A numeric that indicates the line width for the boundaries of the target objective regions.
bnd.lty	A numeric that indicates the line type for the boundaries of the target objective regions.

Details

This function simply creates a base tic-tac-toe plot. Observed values, with confidence intervals, are added to this plot with plotCI from **plotrix**; see examples.

Value

None. However, a graphic is produced.

IFAR Chapter

6-Size Structure.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See psdVal and psdCalc for related functionality.

Examples

```
## Create hypothetical data for plotting one point .. similar to what might come from psdCalc()
prey <- c(45.4,30.2,56.8)
pred <- c(24.5,10.2,36.7)
names(prey) <- names(pred) <- c("Estimate","95% LCI","95% UCI")
prey
pred

tictactoe()
if (require(plotrix)) {
  plotCI(prey[1],pred[1],li=prey[2],ui=prey[3],err="x",pch=16,add=TRUE)
  plotCI(prey[1],pred[1],li=pred[2],ui=pred[3],err="y",pch=16,add=TRUE)
}

## Create hypothetical data for plotting three points .. similar to what might come from psdCalc()
prey <- rbind(c(45.4,30.2,56.8),
```

```
              c(68.2,56.7,79.4),
              c(17.1, 9.5,26.3))
pred <- rbind(c(24.5,10.2,36.7),
              c(14.2, 7.1,21.3),
              c(16.3, 8.2,24.4))
colnames(prey) <- colnames(pred) <- c("Estimate","95% LCI","95% UCI")
prey
pred

tictactoe()
if (require(plotrix)) {
  plotCI(prey[,1],pred[,1],li=prey[,2],ui=prey[,3],err="x",pch=16,add=TRUE)
  plotCI(prey[,1],pred[,1],li=pred[,2],ui=pred[,3],err="y",pch=16,add=TRUE)
}
lines(prey[,1],pred[,1])
text(prey[,1],pred[,1],labels=c(2010,2011,2012),adj=c(-0.5,-0.5))
```

`validn` *Finds the number of valid (non-NA) values in a vector.*

Description

Finds the number of valid (non-NA) values in a vector.

Usage

```
validn(object)
```

Arguments

`object`	A vector.

Value

A single numeric value that is the number of non-NA values in a vector.

IFAR Chapter

2-Basic Data Manipulations.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

See Also

See `valid.n` in **plotrix** and `nobs` in **gdata** for similar functionality. See `is.na` for finding the missing values.

Examples

```
junk1 <- c(1,7,2,4,3,10,NA)
junk2 <- c("Derek","Hugh","Ogle","Santa","Claus","Nick",NA,NA)
junk3 <- factor(junk2)
junk4 <- c(TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,NA,NA)
junk5 <- data.frame(junk1)
junk6 <- data.frame(junk3)

validn(junk1)
validn(junk2)
validn(junk3)
validn(junk4)
validn(junk5)
validn(junk6)
```

---

vbStarts *Find reasonable starting values for a von Bertalanffy growth function.*

Description

Finds reasonable starting values for the parameters in a specific parameterization of the von Bertalanffy growth function.

Usage

```
vbStarts(formula, data = NULL, param = c("Typical", "typical",
  "Traditional", "traditional", "BevertonHolt", "Original", "original",
  "vonBertalanffy", "GQ", "GallucciQuinn", "Mooij", "Weisberg", "Ogle",
  "Schnute", "Francis", "Somers", "Somers2", "Pauly"), type = param,
  fixed = NULL, meth0 = c("yngAge", "poly"), methLinf = c("Walford",
  "oldAge", "longFish"), num4Linf = 1, ages2use = NULL,
  methEV = c("means", "poly"), valOgle = NULL, plot = FALSE,
  col.mdl = "gray70", lwd.mdl = 3, lty.mdl = 1, cex.main = 0.9,
  col.main = "red", dynamicPlot = FALSE, ...)
```

Arguments

formula	A formula of the form len~age.
data	A data frame that contains the variables in formula.
type, param	A string that indicates the parameterization of the von Bertalanffy model.
fixed	A named list that contains user-defined rather than automatically generated (i.e., fixed) starting values for one or more parameters. See details.
meth0	A string that indicates how the t0 and L0 parameters should be derived. See details.
methLinf	A string that indicates how Linf should be derived. See details.

num4Linf	A single numeric that indicates how many of the longest fish (if methLinf="longFish") or how any of the oldest ages (if methLinf="oldAge") should be averaged to estimate a starting value for Linf.
ages2use	A numerical vector of the two ages to be used in the Schnute or Francis parameterizations. See details.
methEV	A string that indicates how the lengths of the two ages in the Schnute parameterization or the three ages in the Francis parameterization should be derived. See details.
valOgle	A single named numeric that is the set Lr or tr value for use in type="Ogle". See details.
plot	A logical that indicates whether a plot of the data with the superimposed model fit at the starting values should be created.
col.mdl	A color for the model when plot=TRUE.
lwd.mdl	A line width for the model when plot=TRUE.
lty.mdl	A line type for the model when plot=TRUE.
cex.main	A character expansion value for the main title when plot=TRUE.
col.main	A color for the main title when plot=TRUE.
dynamicPlot	DEPRECATED.
...	Further arguments passed to the methods.

Details

This function attempts to find reasonable starting values for a variety of parameterizations of the von Bertalanffy growth function. There is no guarantee that these starting values are the 'best' starting values. One should use them with caution and should perform sensitivity analyses to determine the impact of different starting values on the final model results.

If methLinf="Walford", then the Linf and K parameters are estimated via the concept of the Ford-Walford plot. If methLinf="oldAge" then Linf is estimated as the mean length of the num4Linf longest observed lengths.

The product of the starting values for Linf and K is used as a starting value for omega in the GallucciQuinn and Mooij parameterizations. The result of log(2) divided by the starting value for K is used as the starting value for t50 in the Weisberg parameterization.

If meth0="yngAge", then a starting value for t0 or L0 is found by algebraically solving the typical or original parameterization, respectively, for t0 or L0 using the mean length of the first age with more than one data point as a "known" quantity. If meth0="poly" then a second-degree polynomial model is fit to the mean length-at-age data. The t0 starting value is set equal to the root of the polynomial that is closest to zero. The L0 starting value is set equal to the mean length at age-0 predicted from the polynomial function.

Starting values for the L1 and L3 parameters in the Schnute parameterization and the L1, L2, and L3 parameters in the Francis parameterization may be found in two ways. If methEV="poly", then the starting values are the predicted length-at-age from a second-degree polynomial fit to the mean lengths-at-age data. If methEV="means" then the observed sample means at the corresponding ages are used. In the case where one of the supplied ages is fractional, then the value returned will be linearly interpolated between the mean lengths of the two closest ages. The ages to be used for L1

and L3 in the Schnute and Francis parameterizations are supplied as a numeric vector of length 2 in `ages2use=`. If `ages2use=NULL` then the minimum and maximum observed ages will be used. In the Francis method, L2 will correspond to the age half-way between the two ages in `ages2use=`. A warning will be given if L2<L1 for the Schnute method or if L2<L1 or L3<L2 for the Francis method.

Starting values for the Somers and Pauly parameterizations are the same as the traditional parameterization for Linf, K, and t0. However, for the Pauly parameterization the starting value for Kpr is the starting value for K divided by 1 minus the starting value of NGT. The starting values of C, ts, WP, and NGT are set at constants that are unlikely to work for all species. Thus, the user should use the `fixed` argument to fix starting values for these parameters that are more likely to result in a reliable fit.

Value

A list that contains reasonable starting values. Note that the parameters will be listed in the same order and with the same names as listed in `vbFuns`.

IFAR Chapter

12-Individual Growth.

Note

The 'original' and 'vonBertalanffy' and the 'typical' and 'BevertonHolt' parameterizations are synonymous.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See references in `vbFuns`.

See Also

See `growthFunShow` to display the equations for the parameterizations used in **FSA** and `vbFuns` for functions that represent the von Bertalanffy parameterizations. See `nlsTracePlot` for help troubleshooting nonlinear models that don't converge.

Examples

```
## Simple examples of each parameterization
vbStarts(tl~age,data=SpotVA1)
vbStarts(tl~age,data=SpotVA1,type="Original")
vbStarts(tl~age,data=SpotVA1,type="GQ")
vbStarts(tl~age,data=SpotVA1,type="Mooij")
vbStarts(tl~age,data=SpotVA1,type="Weisberg")
vbStarts(tl~age,data=SpotVA1,type="Francis",ages2use=c(0,5))
```

```
vbStarts(tl~age,data=SpotVA1,type="Schnute",ages2use=c(0,5))
vbStarts(tl~age,data=SpotVA1,type="Somers")
vbStarts(tl~age,data=SpotVA1,type="Somers2")
vbStarts(tl~age,data=SpotVA1,type="Pauly")
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(tr=0))
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(Lr=8))

## Using a different method to find Linf
vbStarts(tl~age,data=SpotVA1,methLinf="oldAge")
vbStarts(tl~age,data=SpotVA1,methLinf="oldAge",num4Linf=2)
vbStarts(tl~age,data=SpotVA1,methLinf="longFish")
vbStarts(tl~age,data=SpotVA1,methLinf="longFish",num4Linf=10)
vbStarts(tl~age,data=SpotVA1,type="Original",methLinf="oldAge")
vbStarts(tl~age,data=SpotVA1,type="Original",methLinf="oldAge",num4Linf=2)
vbStarts(tl~age,data=SpotVA1,type="Original",methLinf="longFish")
vbStarts(tl~age,data=SpotVA1,type="Original",methLinf="longFish",num4Linf=10)
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(tr=0),methLinf="oldAge",num4Linf=2)
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(Lr=8),methLinf="longFish",num4Linf=10)

## Using a different method to find t0 and L0
vbStarts(tl~age,data=SpotVA1,meth0="yngAge")
vbStarts(tl~age,data=SpotVA1,type="original",meth0="yngAge")

## Using a different method to find the L1, L2, and L3
vbStarts(tl~age,data=SpotVA1,type="Francis",ages2use=c(0,5),methEV="means")
vbStarts(tl~age,data=SpotVA1,type="Schnute",ages2use=c(0,5),methEV="means")

## Examples with a Plot
vbStarts(tl~age,data=SpotVA1,plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="original",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="GQ",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Mooij",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Weisberg",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Francis",ages2use=c(0,5),plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Schnute",ages2use=c(0,5),plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Somers",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Somers2",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Pauly",plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(tr=0),plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(Lr=8),plot=TRUE)

## Examples where some parameters are fixed by the user
vbStarts(tl~age,data=SpotVA1,fixed=list(Linf=15))
vbStarts(tl~age,data=SpotVA1,fixed=list(Linf=15,K=0.3))
vbStarts(tl~age,data=SpotVA1,fixed=list(Linf=15,K=0.3,t0=-1))
vbStarts(tl~age,data=SpotVA1,fixed=list(Linf=15,K=0.3,t0=-1),plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Pauly",fixed=list(t0=-1.5),plot=TRUE)
vbStarts(tl~age,data=SpotVA1,type="Ogle",valOgle=c(tr=2),fixed=list(Lr=10),plot=TRUE)

## See examples in vbFuns() for use of vbStarts() when fitting Von B models
```

WhitefishLC *Assigned ages from two readers on three structures for Lake Whitefish from Lake Champlain.*

Description

Assigned ages from two readers on three structures for Lake Whitefish (*Coregonus clupeaformis*) from Lake Champlain in 2009.

Format

A data frame with 151 observations on the following 11 variables:

fishID A unique fish identification number

tl Total length (in mm)

scale1 Assessed age from scales by first reader

scale2 Assessed age from scales by second reader

scaleC Consensus age from scales by both reader

finray1 Assessed age from fin rays by first reader

finray2 Assessed age from fin rays by second reader

finrayC Consensus age from fin rays by both reader

otolith1 Assessed age from otoliths by first reader

otolith2 Assessed age from otoliths by second reader

otolithC Consensus age from otoliths by both reader

Topic(s)

- Age
- Ageing Error
- Precision
- Bias
- Age Comparisons

Source

Data from Herbst, S.J. and J.E. Marsden. 2011. Comparison of precision and bias of scale, fin ray, and otolith age estimates for lake whitefish (*Coregonus clupeaformis*) in Lake Champlain. Journal of Great Lakes Research. 37:386-389. Contributed by Seth Herbst. **Do not use for other than educational purposes without permission from the author.** [Was (is?) from http://www.uvm.edu/rsenr/emarsden/documents/Herbst%20and%20Marsden%20whitefish%20age%20structure%20compar

See Also

Used in `ageBias` and `agePrecision` examples.

Examples

```
str(WhitefishLC)
head(WhitefishLC)
```

WR79 *Ages and lengths for a hypothetical sample from Westerheim and Ricker (1979).*

Description

Ages and lengths for a hypothetical sample in Westerheim and Ricker (1979).

Format

A data frame of 2369 observations on the following 3 variables:

ID Unique fish identifiers

len Length of an individual fish

age Age of an individual fish

Details

Age-length data in 5-cm increments taken exactly from Table 2A of the source which was a sample from a hypothetical population in which year-class strength varied in the ratio 2:1 and the rate of increase in length decreased with age. Actual lengths in each 5-cm interval were simulated with a uniform distribution. The aged fish in this file were randomly selected and an assessed age was assigned according to the information in Table 2A.

Topic(s)

- Age-Length Key

Source

Simulated from Table 2A in Westerheim, S.J. and W.E. Ricker. 1979. Bias in using age-length key to estimate age-frequency distributions. Journal of the Fisheries Research Board of Canada. 35:184-189.

Examples

```
str(WR79)
head(WR79)

## Extract the aged sample
WR79.aged <- subset(WR79,!is.na(age))
str(WR79.aged)
```

```
## Extract the length sample
WR79.length <- subset(WR79,is.na(age))
str(WR79.length)
```

---

wrAdd *Computes a vector of relative weights specific to a species in an entire data frame.*

Description

This computes a vector that contains the relative weight specific to each species for all individuals in an entire data frame.

Usage

```
wrAdd(wt, ...)

## Default S3 method:
wrAdd(wt, len, spec, units = c("metric", "English"),
  ...)

## S3 method for class 'formula'
wrAdd(wt, data, units = c("metric", "English"), ...)
```

Arguments

wt	A numeric vector that contains weight measurements or a formula of the form wt~len+spec where "wt" generically represents the weight variable, "len" generically represents the length variable, and "spec" generically represents the species variable. Note that this formula can only contain three variables and they must be in the order of weight first, length second, species third.
...	Not used.
len	A numeric vector that contains length measurements. Not used if wt is a formula.
spec	A character or factor vector that contains the species names. Not used if wt is a formula.
units	A string that indicates whether the weight and length data in formula are in ("metric" (DEFAULT; mm and g) or "English" (in and lbs) units.
data	A data.frame that minimally contains variables of the the observed lengths, observed weights, and the species names given in the formula=.

Details

This computes a vector that contains the relative weight specific to each species for all individuals in an entire data frame. The vector can be appended to an existing data.frame to create a variable that contains the relative weights for each individual. The relative weight value will be NA for each individual for which a standard weight equation does not exist in WSlit, a standard weight equation for the units given in units= does not exist in WSlit, a standard weight equation for the 75th percentile does not exist in WSlit, or if the individual is shorter or longer than the lengths for which the standard weight equation should be applied. Either the linear or quadratic equation has been listed as preferred for each species, so only that equation will be used. The use of the 75th percentile is by far the most common and, because this function is designed for use on entire data frames, it will be the only percentile allowed. Thus, to use equations for other percentiles, one will have to use "manual" methods. See WSlit and wsVal for more details about types of equations, percentiles, finding which species have published standard weight equations, etc. See the examples for one method for changing species names to something that this function will recognize.

Value

Returns A numeric vector that contains the computed relative weights, in the same order as in data=.

IFAR Chapter

8-Condition.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See wsVal, WSlit, and psdAdd for related functionality. See mapvalues for help in changing species names to match those in WSlit.

Examples

```
## Create random data for three species
# just to control the randomization
set.seed(345234534)
dbt <- data.frame(species=factor(rep(c("Bluefin Tuna"),30)),tl=round(rnorm(30,1900,300),0))
dbt$wt <- round(4.5e-05*dbt$tl^2.8+rnorm(30,0,6000),1)
dbg <- data.frame(species=factor(rep(c("Bluegill"),30)),tl=round(rnorm(30,130,50),0))
dbg$wt <- round(4.23e-06*dbg$tl^3.316+rnorm(30,0,10),1)
dlb <- data.frame(species=factor(rep(c("Largemouth Bass"),30)),tl=round(rnorm(30,350,60),0))
dlb$wt <- round(2.96e-06*dlb$tl^3.273+rnorm(30,0,60),1)
df <- rbind(dbt,dbg,dlb)
str(df)
```

```
df$Wr1 <- wrAdd(wt~tl+species,data=df)
## same but with non-formula interface
df$Wr2 <- wrAdd(df$wt,df$tl,df$species)

## Same as above but using dplyr
if (require(dplyr)) {
  df <- mutate(df,Wr3a=wrAdd(wt,tl,species))
}

df

## Example with only one species in the data.frame
bg <- Subset(df,species=="Bluegill")
bg$Wr4 <- wrAdd(wt~tl+species,data=bg)
```

`WSlit` *All known standard weight equations.*

Description

Parameters for all known standard weight equations.

Format

A data frame with observations on the following 13 variables:

species Species name.

units Units of measurements. `Metric` uses lengths in mm and weight in grams. `English` uses lengths in inches and weight in pounds.

type Type of equation (`linear` or `quadratic`).

ref Reference quartile (`75`, `50`, or `25`).

measure The type of length measurement used – total length (`TL`) or fork length (`FL`).

method The type of method used to derive the equation (`RLP`,`EmP`, or `Other`).

min.len Minimum total length (mm or in, depending on `units`) for which the equation should be applied.

max.len Maximum total length (mm or in, depending on `units`) for which the equation should be applied.

int The intercept for the model.

slope The slope for the linear models or the linear coefficient for the quadratic equation.

quad The quadratic coefficient in the quadratic equations.

source Source of the equation. These match the sources given in Neumann et al. 2012.

comment Comments about use of equation.

Details

The minimum TL for the English units were derived by rounding the converted minimum TL for the metric units to what seemed like common units (inches, half inches, or quarter inches).

Topic(s)

- Relative weight
- Standard weight
- Condition

IFAR Chapter

8-Condition.

Source

Most of these equations can be found in Neumann, R.M., C.S. Guy, and D.W. Willis. 2012. Length, Weight, and Associated Indices. Chapter 14 in Zale, A.V., D.L. Parrish, and T.M. Sutton, editors. Fisheries Techniques. American Fisheries Society, Bethesda, MD.

Some species were not in Neumann et al (2012) and are noted as such in the `comments` variable.

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See `wsVal` and `wrAdd` for related functionality.

Examples

```
str(WSlit)
head(WSlit)
```

---

wsVal *Finds standard weight equation coefficients for a particular species.*

Description

Returns a vector that contains all known or a subset of information about the standard weight equation for a given species, type of measurement units, and reference percentile.

Usage

```
wsVal(species = "List", units = c("metric", "English"), ref = 75,
  simplify = FALSE)
```

Arguments

species	A string that contains the species name for which to find coefficients. See details.
units	A string that indicates whether the coefficients for the standard weight equation to be returned are in ("metric" (DEFAULT; mm and g) or "English" (in and lbs) units.
ref	A numeric that indicates which percentile the equation should be returned for. Note that the vast majority of equations only exist for the 75th percentile (DEFAULT).
simplify	A logical that indicates whether the 'units', 'ref', 'measure', 'method', 'comments', and 'source' fields should be included (=FALSE) or not (=TRUE; DEFAULT). See details.

Details

This function extract all known information from WSlit about the following standard weight equation,

$$log_{10}(Ws) = log_{10}(a) + blog_{10}(L) + blog_{10}(L)^2$$

See WSlit for more information about the meaning of each value returned.

Note from above that the coefficients are returned for the TRANSFORMED model. Thus, to obtain the standard weight (Ws), the returned coefficients are used to compute the common log of Ws which must then bed raised to the power of 10 to compute the Ws.

Value

A one row data frame from WSlit that contains all known information about the standard weight equation for a given species, type of measurement units, and reference percentile if simplify=FALSE. If simplify=TRUE then only the species; minimum and maximum length for which the standard equation should be applied; and intercept, slope, and quadratic coefficients for the standard weight equation. Note that the maximum length and the quadratic coefficient will not be returned if they do not exist in WSlit.

If no arguments are given to this function, a species name is mis-spelled, or if a standard weight equation does not exist (in WSlit) for a particular species, then a warning will be issued and a list of species names will be printed.

IFAR Chapter

8-Condition.

Author(s)

Derek H. Ogle, <derek@derekogle.com>

References

Ogle, D.H. 2016. Introductory Fisheries Analyses with R. Chapman & Hall/CRC, Boca Raton, FL.

See Also

See wrAdd and WSlit for related functionality.

Examples

```
wsVal()
wsVal("Bluegill")
wsVal("Bluegill",units="metric")
wsVal("Bluegill",units="English")
wsVal("Bluegill",units="English",simplify=TRUE)
wsVal("Ruffe",units="metric",simplify=TRUE)
wsVal("Ruffe",units="metric",ref=50,simplify=TRUE)
```

Index