Package ‘catchMSY’

November 27, 2022

Type Package

Title A Package that Implements and Documents the catch-MSY Stock Assessment Method

Version 0.0.7

Date 2020-09-13

Author Malcolm Haddon

Maintainer Malcolm Haddon <Malcolm.Haddon@gmail.com>

Description catchMSY is a package dedicated solely to the implementation of the catch-MSY stock assessment method proposed by Martell and Froese, 2013; see the vignette). It includes an Rcpp routine to speed the analysis by a significant degree. An earlier version is included in the datalowSA R package. In this version I have added the option of using an approximate Fox model of productivity rather than the more usual Schaefer model. This permits the beginning of an appreciation of the extra uncertainty that can arise from using an alternative model for the stock reduction process that underlies the functioning of the catch-MSY procedure.

License GPL-3

Depends R (>= 3.0)

Encoding UTF-8

LazyData TRUE

Imports Rcpp (>= 1.0.0), makehtml

LinkingTo Rcpp

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Reference Haddon, M., Burch, P., Dowling, N., and R. Little (2018) Reducing the Number of Undefined Species in Future Status of Australian Fish Stocks Reports: Phase Two - training in the assessment of data-poor stocks. CSIRO. FRDC Final Report 2017/102. Hobart. 125 p.

NeedsCompilation yes

Archs i386, x64

R topics documented:

`catchMSY` *catchMSY a set of functions to assist with catch-MSY assessments*

Description

The catchMSY package provides three categories of functions analytical functions that enable the production of data-poor model-assisted stock assessments, utility functions that assist with data manipulation and extracting informaiton from output objects, and plotting functions that facilitate the illustration of the results of the assessments. In addition there are example data sets with which to illustrate the methods.

Analytical functions

central generates three estimates of central tendency

doproject generates constant catch projections after running cMSY

run_cMSY The main function for conducting a modified Catch-MSY analysis.

sraMSY Is called by run_cMSY and it calls oneSRA for as many iterations or replicates as entered. It produces large arrays of the biomass trajectories from each SRA along with whether or not each trajectory meets the acceptance criteria or not. Not exported but can be read using r4tier5:::sraMSY

oneSRA Is called by sraMSY. It takes in the vector of initial biomass depletions plus the randomly generated set of population model parameters and runs the SRA for each of the combinations of parameters and initial depletion levels. Not exported but can be read using catchMSY:::oneSRA

pulloutStats summaries the results from the Catch-MSY analysis by generating the mean, minimum, maximum, and quantiles of the resulting r, K, and MSY values.

Utility functions

gettier5data gets the columns of data required for Tier5, the input data.frame must contain at least year and catch, but can also contain species

gettraject extracts the plausible biomass trajectories from the output of cMSY

halftable halves the height of a tall narrow data.frame

makedeplet converts the biomass trajetories into a depletion matrix

pulloutStats summaries results from the Catch-MSY analysis

catchMSY A brief description of all functions in datalowSA

summarycMSY makes tables of msy, r, K, meanr, meanK, and all picks

tier4to5 generates a Tier5 formatted dataset from a tier4 dataset

whichsps generates a listing of which species are in the tier4 data

Plotting functions

plotMSY6 generates 6 graphs illustrating the array of rK parameter combinations and whether they were successful or not. That plot is coloured by how many trajectories across the initial depletion range were successful.

plottrajectory plots out the predicted biomass trajectories from those parameter combinations that have been accepted. It can either put all trajectories on one plot or generate a separate plot for each rK parameter set. Each individual biomass trajectory represents a set of population model parameters and a single initial depletion. It is possible to only print a specified number of parameter sets rather than all of them.

Data sets

fishdat A dataset containing the fish data.frame, the glb list, and the props data.frame set up ready for use with datalowSA. In particular it can be used with fitASPM, fitSPM, run_cMSY, and DBSRA. see ?fishdat

invert A dataset containing the fish data.frame as a 31 x 7 matrix, the glb and props data.frames are set to NULL. The fish data.frame has both the standardized cpue as well as the unstandardized geom, that is the geometric mean cpue. This is particularly set up to be used with the SPM functions but also the Catch-MSY routines. see ?invert

sps A dataset containing 9 columns of typical scalefish fisheries data

Vignettes

To learn more about catchMSY, start with the vignette: `browseVignettes(package = "catchMSY")`

`central` *central generates three estimates of central tendency*

Description

central generates three estimates of central tendency and the quantiles about the distribution of the inut vector of values. The three measures are the arithmetic mean, the naive geometric mean, and the bias corrected geometric mean.

Usage

```
central(x, P = 0.9)
```

Arguments

`x`	the vector of values whose central tendency is to be characertized
`P`	the quantiles to be determined

Value

a 4 x 2 matrix containing the central tendency measures and the quantiles

Examples

```
x <- rnorm(1000,mean=5,sd=1)
central(x,P=0.9)
x <- rlnorm(1000,meanlog=2,sdlog=0.2)
central(x,P=0.95)
exp(2)
```

checkdata — *checkdata determines which methods match the input data*

Description

checkdata ensures the input fishery data contains the data necessary for catchMSY and surplus production modelling. It reports which analyses are possible.

Usage

```
checkdata(indata, glob = NA)
```

Arguments

`indata`	this can be either the complete list of data objects as obtained from readdata or just the matrix of fisheries data.
`glob`	in case one inputs a data.frame of fishery data this allows one to enter the globals list containing as a minimum the spsname/title and resilience, so that a test can be made for the spm analysis

Value

a 2 x 2 matrix with vaiable and true or false for presence

Examples

```
## Not run:
data(invert)
fish <- invert$fish
checkdata(invert)
checkdata(fish)

## End(Not run)
```

cMSYphaseplot — *cMSYphaseplot plots the phase plot and catch and harvest rate plots*

Description

cMSYphaseplot extracts the necessary data to enable the production of a phase plot of estimated average biomass against estimated average harvets rate. It plots the Bmsy = 0.5B0 for the Schaefer model as well as 20 of data is identified by a larger green point and the last year by a larger red point. It also plots the expected harvest rate that should lead to MSY, called Ftarg and that, which if continued for long enough, would drive the biomass below the limit reference point, Flim. Points above the Flim line (or Ftarg depending on which management objectives are used) would be classed as over-fishing leading to a status of 'depleting' and points to the left of 0.2B0 would be over-fished or 'depleted'. In addition, the function plots the catch history and the implied harvest rates just below the phase plot to aid in its interpretation.

Usage

```
cMSYphaseplot(answer, fish)
```

Arguments

`answer`	the output from the run_cMSY function
`fish`	the fishery data put into run_CMSY

Value

a list of meanB, meanH, msy, Bmsy, Hmsy, and Hlim, returned invisibly

Examples

```
## Not run:
  data(invert)
  fish <- invert$fish
  glb <- invert$glb
  answer <- run_cMSY(fish,glb,n=1000,sigpR=0.025)
  plotprep(width=7,height=6)
  cMSYphaseplot(answer,fish)

## End(Not run)
```

`datatemplate` *datatemplate produces a standard format file as a template*

Description

datatemplate produces a standard format file as a data-file template. Once the data-file is created, go in and edit it appropriately to suit exactly the data you have for your own species. Note that in the example commas are used to separate individual data fields, this is required if you are going to use the function 'readdata' to get your data into catchMSY, which is the recommended method. But of course you are free to use catchMSY however you wish! Each section in the data file, except the title, is identified using CAPITAL letters, as in RESILIENCE, NYRS, and YEARS. These, again are needed by 'readdata' to identify the various sections and proceed to reading that data appropriately.

Usage

```
datatemplate(filename = "tcinvert.csv", title = "Trawl_Caught_Invertebrate")
```

Arguments

`filename`	- the name for the generated data-file, a character string, default=tcinvert.csv
`title`	the first line of the data file; identify its contents

Value

a standard format data-file ready to be read by readdata and it outputs the full address of the file-name.

Examples

```
## Not run:
  datatemplate(filename="fishery1.csv")
  ans <- readdata("fishery1.csv")
  str(ans)

## End(Not run)              # filename="C:/Users/had06a/Dropbox/Public/fishery1.csv"
```

`detSRA` *detSRA a deterministic SRA from random starting points*

Description

detSRA implements the deterministic dynamics of each of the Stock reduction analyses used by the cmsy approach.

Usage

```
detSRA(N, pbound, catch)
```

Arguments

`N`	the number of replicates; 10000 should be a minimum, 20000 better
`pbound`	a list containing the r and K values for the surplus production model that forms the basis of the dynamics
`catch`	a vector of catches for use with the cmsy SRA

Value

a list of the initial bounds on r and K, the successful trajectories, and the successful pairs of r and K parameters

Examples

```
print ("wait on suitable inbuilt data")
```

`docatchMSY` *docatchMSY will run a complete analysis and generate plots and tables*

Description

docatchMSY will run a complete analysis and generate plots as .png files, and tables as .csv files, into 'rundir' with the option of displaying them all in a local website (for which it uses the R package makehtml found in www.github.com/makehtml). The algorithm used to tun the catchMSY procedure was derived for the R-code made available with the original Martell and Froese, (2013) publication. To open the local website go to 'rundir' and double click on 'analysis'.html

Usage

```
docatchMSY(
  rundir,
  datafile,
  reps = 20000,
  schaef = 1,
  intensityplot = FALSE,
  verbose = TRUE,
  runnotes = "",
  openresults = TRUE,
  cleanslate = FALSE
)
```

Arguments

`rundir`	the directory into which all result files are placed
`datafile`	either the name of the datafile as character, containing spsname, resilience, number of years of data and the fish data.frame, OR a list containing at list the fish data.frame and the glb globals
`reps`	the number of replicate trials to attempt, default=20000
`schaef`	should the schaefer model = 1, or the approximate fox model =0, be used to describe the underlying dynamics
`intensityplot`	Should the time-consuming density plot of the acceptable trajectories be plotted, default=FALSE
`verbose`	should progress messages be set to the console, default=TRUE
`runnotes`	extra information printed to the home page of the local website, default=""
`openresults`	should the local website be opened.
`cleanslate`	should the directory be emptied of all files first? All html, csv, png, RData, and css files in the directory will be deleted. default=FALSE. This is obviously a very powerful and potentially dangerous argument, hence it needs to be set =TRUE explicitly.

Value

invisibly returns a list of the rundir, the answer output object from run_cMSYm the summcMSY object, the start and end times, and the glb and fish objects

References

Martell, S. and R. Froese (2013) A simple method for estimating MSY from catch and resilience. _Fish and Fisheries_ __14__: 504-514

Examples

```
print("later!")
```

`doconstC` *doconstC calculates and plots projections under a constant catch*

Description

doconstC merely combines the functions gettraject, doproject, makedeplet, and plotconstC to simplify the process of conducting projections using constant catches. Compare this with separately using the functions gettraject, doproject, makedeplet, and plotconstC in sequence, which is all that dococnstC does.

Usage

```
doconstC(
  inR1,
  projn = 5,
  constCatch = 100,
  lastyear = 2017,
  limit = 0.2,
  target = 0.48,
  console = TRUE,
  intensity = NA
)
```

Arguments

`inR1`	the R1 object that is within the list generated by run_cMSY.
`projn`	the number of years of projection with a default of 5
`constCatch`	the constant catch to be applied to each successful trajectory
`lastyear`	the final year of the known catches and biomass trajectories
`limit`	the depletion level acting as the limit referencepoint
`target`	the depletion level used as a biomass target for the species.
`console`	logical, should results be printed to the console. Default =TRUE
`intensity`	the value that defines the density of trajectories required to give rise to full colour; default = NA which implies grey

Value

a matrix of the all years with the proportion < 20 proportion > 48 the proportion of trajectories that were increasing relative to the lastyear of data.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
reps <- 10000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=1e-6)
out <- doconstC(answer$R1,projn=5,constCatch=150,lastyear=2017,
                limit=0.2,target=0.4)
```

```
str(out)

## End(Not run)
```

doproject *doproject after running the cMSY analysis the plausible trajectories*

Description

after running the run_cMSY analysis this function projects each of the accepted biomass trajectories

Usage

```
doproject(intraj, constC, projn = 5, sigpR = 0.025, schaefer = 1)
```

Arguments

intraj	the successful biomass trajectories obtained from gettraject
constC	the constant catch that is to be applied as a projection
projn	the number of years to be projected.
sigpR	the process error, the same as used in run_cMSY
schaefer	use the schaefer (when = 1) or approximate Fox (when = 0)

Value

the same biomass trajecotry matrix as input excpet the empty years will have been filled with projections of the surplus production dynamics made under the constant catch level.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
reps <- 5000  # one would run at least 20000, preferably more
schaefer <- 1
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04,schaefer=schaefer)
traject <- gettraject(answer$R1,projn=5)
newtraj <- doproject(traject,constC=100,schaefer=schaefer)
head(newtraj)
trajdepl <- makedeplet(newtraj)
plotconstC(trajdepl,endyear=2017,constC=100,target=0.40)

## End(Not run)
```

`fillell2` *fillell2 runs the criteria of success on each SRA*

Description

fillell2 runs the acceptance criteria over the biomass trajectories obtained from the Stock Reduction Analysis for each combination of r, K, and starting depletion conducted in makebiomC. The criteria agsint which each trajectory are tested include that the final depletion be less than the maximum of the expected range of final depletions and greater than the minimum of final depletion levels. In addition I have included that the initial biomass level be greater than the final biomass level. This differs from the extra criteria used by Martell and Froese (2013), who selected the smallest K still able to generate the mean predicted MSY.

Usage

```
fillell2(biot, intheta, mult, ct, Hmax = 1, Fyear)
```

Arguments

`biot`	the matrix of biomass trajectories obtained from makebiomC, containing the SRA trajectories for the aprticular combination of r and K across the range of starting biomass trajectories.
`intheta`	the array of parameters associated with the biomass trajectories. Includes r, K, the range of depletions and the process error added to productivity - sigR.
`mult`	a multiplier for K to allow for stock biomasses to rise above K
`ct`	the vector of catches per year
`Hmax`	upper limit of harvest rate included in the constraints; defaults to 1.0, which implies no upper limit.
`Fyear`	is the index to the year in which a rnage of harvest rates is to be used to constrina the acceptable trajectories.

Value

a vector of 1 and 0 relating to an acceptable trajectory (one that met the criteria) and unacceptable (one that failed the criteria)

Examples

```
## Not run:
print("This is the function that imposes constraints on the biomass")
print("trajectories. Add new ones to the end of the long if statement")
print("They need to define what is required to keep a trajectory")

## End(Not run)   #  biot=biot; intheta=itheta[tick,]; mult=mult; ct=ct; Hmax=maxH;Fyear=NA
```

`getConst` *getConst extracts 'nb' numbers from a line of text*

Description

getConst parses a line of text and extracts 'nb' pieces of text as numbers

Usage

```
getConst(inline, nb, index = 2)
```

Arguments

`inline`	text line to be parsed, usually obtained using readLines
`nb`	the number of numbers to extract
`index`	which non-empty object to begin extracting from?

Value

a vector of length 'nb'

Examples

```
txtline <- "MaxDL , 32,32,32"
getConst(txtline,nb=3,index=2)
```

`getLNCI` *getLNCI gets the log-normal confidence intervals*

Description

getLNCI takes the mean and the standard deviation and produces the asymmetric log-normal confidence intervals around the mean values

Usage

```
getLNCI(av, se, P = 95)
```

Arguments

`av`	the mean value or a vector of mean values
`se`	the standard deviation
`P`	the percent used for the CI, defaults to 95.

Value

a vector of three for a single input or a matrix of 3 columns for input vectors

Examples

```
## Not run:
  av <- c(4.0,2.15)
  se <- 0.33
  getLNCI(av,se,P=95)
  se <- c(0.33,0.4)
  getLNCI(av,se)

## End(Not run)
```

`getmax` *getmax generates the upper bound for a plot*

Description

getmax generates an upper bound for a plot where it is unknown whether the maximum is greater than zero of not. If > 0 then multiplying by the default mult of 1.05 works well but if the outcome if < 0 then the multiplier needs to be adjusted appropriately so the maximum is slightly higher than the maximum of the data

Usage

```
getmax(x, mult = 1.05)
```

Arguments

`x`	the vector of data to be tested for its maximum
`mult`	the multiplier for both ends, defaults to 1.05 (=0.95 if < 0)

Value

a suitable upper bound for a plot if required

Examples

```
## Not run:
vect <- rnorm(10,mean=0,sd=2)
sort(vect,decreasing=TRUE)
getmax(vect,mult=1.0)
vect <- rnorm(10,mean = -5,sd = 1.5)
sort(vect,decreasing=TRUE)
getmax(vect,mult=1.0)

## End(Not run)
```

getmin *getmin generates the lower bound for a plot*

Description

getmin generates a lower bound for a plot where it is unknown whether the minumum is less than zero of not. If less than 0 then multiplying by the default mult of 1.05 works well but if the outcome if > 0 then the multiplier needs to be adjusted appropriately so the minimum is slightly lower than the minimum of the data

Usage

```
getmin(x, mult = 1.05)
```

Arguments

`x`	the vector of data to be tested for its minimum
`mult`	the multiplier for both ends, defaults to 1.05 (=0.95 if >0)

Value

a suitable lower bound for a plot if required

Examples

```
vect <- rnorm(10,mean=0,sd=2)
sort(vect)
getmin(vect,mult=1.0)
```

getprop *getprop extract proportion of values between lim1 and lim2*

Description

getprop extracts the proportion of values in invect either below lim1, between lim1 and lim2, or above lim2. If lim2 has the value 0.0 and lim1 > 0 then the proportion below lim2 is extracted. If lim1 = 0.0 and lim2 > 0, then the proportion > lim2 is extracted. If both lim1 and lim2 > 0 then the proportion between the two is extracted.

Usage

```
getprop(invect, lim1 = 0, lim2 = 0)
```

Arguments

`invect`	the collection of values to be subdivided
`lim1`	the lower limit of values
`lim2`	the upper limit of values

Value

a scalar containing the proportion of records

Examples

```
## Not run:
  x <- 1:100
  getprop(x,20)
  getprop(x,20,50)
  getprop(x,lim2=80)

## End(Not run)
```

`getsingle` *getsingle extracts a single number from an input line of characters*

Description

getsingle splits up a text line and translates the first non- empty character string into a number.

Usage

```
getsingle(inline, sep = ",")
```

Arguments

`inline`	the line of text, usually taken after using readLines
`sep`	the separator used to divide the numbers from descriptive text. defaults to a comma.

Value

a single number

Examples

```
## Not run:
x <- "12.3 , this is a number"
y <- "21.3 # 22.3 # here are two numbers"
getsingle(x)
getsingle(y,sep="#")

## End(Not run)
```

getsingleNum *getsingleNum find a line of text and extracts a single number*

Description

getsingleNum uses grep to find an input line. If the variable being searched for fails then NULL is returned

Usage

```
getsingleNum(varname, intxt)
```

Arguments

`varname`	the name of the variable to get from intxt
`intxt`	text to be parsed, usually obtained using readLines

Value

a single number or, if no value is in the data file a NULL

Examples

```
## Not run:
 txtlines <- c("replicates, 100","Some_other_text, 52")
 getsingleNum("replicates",txtlines)
 getsingleNum("eeplicates",txtlines)
 getsingleNum("other",txtlines)

## End(Not run)
```

getStr *getStr obtains a string from an input text line*

Description

getStr obtains a string from an input text line in which any parts are separated by ','. Then, after ignoring the first component, assumed to be a label, it returns the first nb parts.

Usage

```
getStr(inline, nb)
```

Arguments

`inline`	input text line with components separated by ','
`nb`	number of parts to return

Value

a vector of character string(s)

Examples

```
txt <- "runlabel, development_run, label for this particular run"
getStr(txt,1)
```

`gettraject` *gettraject extracts the plausible biomass trajectories from cMSY*

Description

gettraject extracts the final plausible biomass trajectories from the R1 object that is part of the output from a run_cMSY analysis. The R1 object contains the table of biomass trajectories, the identifer of the individual trajectories within each rK pair that succeeded, and the rK pairs that were trialed. The output is a matrix of only the successful biomass trajectories with the associated rK and starting depletion appending to each trajectory. If no projections are wanted then projn should be set to 0

Usage

```
gettraject(inR1, projn = 0)
```

Arguments

inR1	the R1 object that is within the list generated by run_cMSY.
projn	the number of extra projection years to allow for in the biomass trajectories placed into the output matrix ready to be filled by the doproject function. Defaults to 0 which leaves out room set up for projections.

Value

a matrix of the accepted biomass trajectories extended by NAs of the length of projn, plus the rK pair and initial depletion that gave rise to the successful biomass trajectory.

Examples

```
data(invert)
fish <- invert$fish
glb <- invert$glb
reps <- 5000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04)
traject <- gettraject(answer$R1,projn=5)
newtraj <- doproject(traject,constC=150)
head(newtraj)
trajdepl <- makedeplet(newtraj)
plotconstC(trajdepl,endyear=2017,constC=150,target=0.40)
```

halftable *halftable halves the height of a tall narrow data.frame*

Description

halftable would be used when printing a table using kable from knitr where one of the columns was Year. The objective would be to split the table in half taking the bottom half and attaching it on the right hand side of the top half. The year column would act as the index.

Usage

```
halftable(inmat, yearcol = "Year", subdiv = 3)
```

Arguments

`inmat`	the data.frame to be subdivided
`yearcol`	the column name of the year field
`subdiv`	the number of times the data.frame should be subdivided; the default is 3 but the numbers can only be 2 or 3.

Value

a data.frame half the height and double the width of the original

Examples

```
## Not run:
x <- as.data.frame(matrix(runif(80),nrow=20,ncol=4))
x[,1] <- 1986:2005
x[,4] <- paste0("text",1:20)
halftable(x,yearcol="V1",subdiv=2)
halftable(x[,c(1,2,4)],yearcol="V1")
x1 <- rbind(x,x[1,])
x1[21,"V1"] <- 2006
halftable(x1,yearcol="V1",subdiv=3)

## End(Not run)
```

invert *invert data derived from a trawl caught invertebrate fishery.*

Description

A dataset containing the fish data.frame as a 31 x 7 matrix, the glb and props data.frames are set to NULL. The fish data.frame has both the standardized cpue as well as the unstandardized geom, that is the geometric mean cpue. This is particularly set up to be used with the SPM functions but also the Catch-MSY routines.

Usage

```
invert
```

Format

A list of three objects only two of which contains data

fish a data.frame containing year, catch, cpue, SE of the cpue, geom, which is the unstandardized geometric mean CPUE, vessel, which is the number of active vessels reporting catches, and records, which is the number of cpue records reported each year

glb contains the resilience and spsname

props set to NULL

Examples

```
 ## Not run:
 data(invert)
 str(invert)
 print(invert$fish)

## End(Not run)
```

`iscol` *incol is a utility to determine is a column is present in a matrix*

Description

incol is a utility to determine whether a names columns is present in a given matrix or data.frame.

Usage

```
iscol(incol = "year", inmat)
```

Arguments

`incol`	the name of the column; defaults to "year" as an example
`inmat`	the matrix or data.frame within which to search for incol

Value

TRUE or FALSE

Examples

```
## Not run:
test <- matrix(c(1,2,3,4),nrow=2,ncol=2,dimnames=list(1:2,c("year","Catch")))
print(test)
iscol("year",test)
iscol("Catch",test)
iscol("catch",test)
iscol("ages",test)

## End(Not run)
```

makebiomC — *makebiomC runs the surplus production model inside catch-MSY.*

Description

runs the surplus production model inside catch-MSY. This is written in C++ because it contains two for-loops the whole of which is run for however many iterations are used. Without using C++ the process can take 15 minutes rather than 20 seconds. This function is only called inside sraMSY, which is another function that did not strictly need to be exported to the public gaze.

Usage

```
makebiomC(intheta, bd, ct, schaefer)
```

Arguments

`intheta`	the vector of parameters for the spm and the biomass
`bd`	the initial biomass depletion vector
`ct`	the vector of catches leading to the stock reduction
`schaefer`	if 1 or TRUE use schaefer model, if 0 use approx Fox

makecMSYdat — *makecMSYdat generates a cMSY dataset out of other data files*

Description

makecMSYdat expects to find in the input data.frame or matrix, columns containing 'Year', and 'Total' (meaning total removals = catch + discards), and 'Resilience' being any of 'verylow', 'low', 'medium', or 'high'.

Usage

```
makecMSYdat(
  indat,
  spsname = "",
  yearcol = "Year",
  catchcol = "Catch",
  resil = "low"
)
```

Arguments

`indat`	the tier 4 datafile for one species
`spsname`	the name of the species to be analysed
`yearcol`	the column name of the year data
`catchcol`	the column namae of the total catch data
`resil`	the value of resilience for the species as a character of either "verylow", "low", "medium", or "high". Note the lower case, defaults to "low"

Value

a list containing a matrix of year and catch, and a list of the resilience and spsname

Examples

```
## Not run:
print("read the csv file containing the general data into 'dat'")
print("then run  makecMSYdat(dat,'spsname')")

## End(Not run)
```

`makedeplet` *makedeplet converts the biomass trajectories into a depletion matrix*

Description

makedeplet converts the biomass trajectories into a depletion matrix by dividing through each trajectory by its respective K value. Usually this would be done after the matrix had been projected forward.

Usage

```
makedeplet(intraj)
```

Arguments

`intraj`	the matrix derived from gettraject containing the successful biomass trajectories.

Value

a matrix of only the biomass trajectories once they have each been divided by their respective K values

Examples

```
## Not run:
traject <- rbind(c(rnorm(10,mean=200,sd=10),0.5,300,0.65),
                 c(rnorm(10,mean=200,sd=10),0.5,300,0.65))
colnames(traject) <- c(1:10,"r","K","initD")
makedeplet(traject)

## End(Not run)
```

plotcMSY6 *plotcMSY6 plots out a summary of the Catch-MSY results in 6 graphs*

Description

plotcMSY6 generates 6 graphs illustrating the array of rK parameter combinations and whether they were successful or not. That plot is coloured by how many trajectories across the initial depletion range were successful.

Usage

```
plotcMSY6(cMSY, catch, label = NA)
```

Arguments

cMSY	the list from running summcMSY on the output from run_cMSY
catch	the catch in each year
label	simply a text label for the y-axes; default = NA

Value

nothing, but it does generate a plot to the screen

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
reps <- 5000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04)
summcMSY <- summarycMSY(answer,fish,final=FALSE)
plotcMSY6(summcMSY,fish[,"catch"],label=glb$spsname)
summcMSY <- summarycMSY(answer,fish,final=TRUE)
str(summcMSY,max.level=1)
plotcMSY6(summcMSY,fish[,"catch"],label=glb$spsname)

## End(Not run)
```

plotconstC *plotconstC summarizes constant catch projections in catch-MSY*

Description

plotconstC plots and summarizes the outcome of constant catch projections made on each of the successful trajectories from the catch-MSY analysis. The catch-MSY analysis provides an uncertain estimate of MSY and of final depletion. But by conducting constant catch projectios the implications in terms of what proportion of trajectories increase and what proportion decrease can aid decisions and could form he basis for the development of formal harvest control rules. It will be noticed that a large proportion of trajectories can be below 20 notice also that at the same time a high proportion can be above 48 assuming that this Commonwealth default target is used. It is the case that a target of 0.4 or 40 species, as a means of preventing the management of such species from prevent the primary economic drivers of a fishery from being caught.

Usage

```
plotconstC(
  deplete,
  endyear,
  constC = 0,
  limit = 0.2,
  target = 0.48,
  console = TRUE,
  intensity = NA,
  contours = TRUE,
  bounds = c(0.1, 0.9)
)
```

Arguments

`deplete`	the output of 'gettraject', a matrix of biomass trajectories
`endyear`	the final year of the known catches and biomass trajectories
`constC`	the constant catch applied to generate the proections
`limit`	biomass depletion level used as a limit for the species
`target`	the depletion level used as a target for the species concerned.
`console`	a boolean determinng whether the projection years results are printed to the console
`intensity`	defaults to NA but if contains a value this is the density of trajectories that lead to full colour
`contours`	default to TRUE, should the 10 be plotted? Alternatively change bounds for differnet values
`bounds`	default = c(0.1,0.9) so that the 0.1 and 0.9 quantiles contours are plotted. If you want different values put two values here.

Value

a matrix of all years with the proportion < 20 (the input target), the mean and median depletion, and the proportion of trajectories that were increasing relative to the endyear of data.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
reps <- 10000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=1e-8)
traject <- gettraject(answer$R1,projn=5)
newtraj <- doproject(traject,constC=300)
trajdepl <- makedeplet(newtraj)
plotconstC(trajdepl,endyear=2017,constC=300,limit=0.2,target=0.40)

## End(Not run)
```

plotfish *plotfish plots the catch and optionally the cpue from fish*

Description

plotfish uses the matrix of fishery data used in the datalowSA standard data format. It requires the matrix or data.frame to contain the columns 'year', 'catch', and optionally 'cpue'.

Usage

```
plotfish(fish, glb, ce = TRUE, title = TRUE, fnt = 7)
```

Arguments

`fish`	the matrix or data.frame containing year, catch, and cpue.
`glb`	the list of biologicals potentially containing the spsname
`ce`	a logical parameter determining whether to plot the cpue or not. the default = TRUE
`title`	determines whether or not the spsname is printed at the top of the plot. Default = TRUE but for a more formal publication it might need to be set to FALSE, which also reallocates the room given to the title to the plot.
`fnt`	the font used in the plot and axes.

Value

prints the location of the png file produced to the console

Examples

```
## Not run:
  data(invert)
  plotfish(invert$fish,invert$glb,ce=TRUE)

## End(Not run)
```

plotprep *plotprep sets up a window and the par values for a single plot*

Description

plotprep sets up a window and the par values for a single plot. It checks to see if a graphics device is open and opens a new one if not. This is simply a utility function to save typing the standard syntax. Some of the defaults can be changed. Typing the name without () will provide a template for modification. If 'windows' is called repeatedly this will generate a new active graphics device each time leaving the older ones inactive but present. For quick exploratory plots this behaviour is not wanted, hence the check if an active device exists already or not.

Usage

```
plotprep(
  width = 7,
  height = 4,
  usefont = 7,
  cex = 0.9,
  newdev = TRUE,
  filename = "",
  resol = 300,
  verbose = TRUE
)
```

Arguments

`width`	defaults to 6 inches = 15.24cm - width of plot
`height`	defaults to 3 inches = 7.62cm - height of plot
`usefont`	default is 7 (bold Times) 1 sans serif, 2 sans serif bold
`cex`	default is 0.85, the font size font used for text in the plots
`newdev`	reuse a previously defined graphics device or make new one; defaults to TRUE
`filename`	defaults to "" = do not save to a filename. If a filename is input the last three characters will be checked and if they are not png then .png will be added.
`resol`	resolution of the png file, if defined, default=300
`verbose`	set this to FALSE to turn off the reminder to include a graphics.off() command after the plot. Default=TRUE

Value

Checks for and sets up a graphics device and sets the default plotting par values. This changes the current plotting options!

Examples

```
## Not run:
 x <- rnorm(1000,mean=0,sd=1.0)
 plotprep()
 hist(x,breaks=30,main="",col=2)

## End(Not run)
```

`plottrajectory` *plottrajectory plots predicted trajectories*

Description

plottrajectory plots out the predicted trajectories from those parameter combinations that have been accepted. In addition, and more importantly, it identifies those trajectories that succeeded and puts them into a smaller matrix than the complete set of trialed rK combinations and each of the successful trajectories. This can project a limited number of trajectories, determined by oneplot and plotall. If oneplot is true it does not matter what is in plotall, all trajectories are given in a single plot along with the median values in red. Similarly for the harvest rate

Usage

```
plottrajectory(
  inR1,
  years,
  catch,
  inparbound,
  scalar = 1000,
  Bmax = 2,
  oneplot = TRUE,
  plotout = TRUE,
  plotall = 7,
  schaefer = 1
)
```

Arguments

`inR1`	the output from run_cMST
`years`	the vector of years
`catch`	the vector of catches per year
`inparbound`	the set of parameter vectors that were successful
`scalar`	literally scales the catch to the same units as biomass; defaults to 1000 so as to convert Kg to tonnes.
`Bmax`	Deprecated. No longer used. Remove from any code will be depleted from later iterations of datalowSA
`oneplot`	Plot all trajectories on top of each other rather than individually. defaults to TRUE.
`plotout`	produce a plot or not? Defaults to TRUE
`plotall`	when oneplot=FALSE how many plots to generate; defaults to 7, which plots the successful trajectories from the first seven r-K pairs that had successful outcomes. Useful numbers are 7 and 15 as the total catch history is also illustrated along with the mean MSY to aid in understandig the trajectories. To see all trajectories set plotall=TRUE.
`schaefer`	use the schaefer (when = 1) or approximate Fox (when = 0)

Value

a list of the yes/no vector and of the accepted rK pairs. This is returned invisibly.

Examples

```
## Not run:
 data(invert)
 fish <- invert$fish
 glb <- invert$glb
 schaefer=1
 reps <- 5000  # one would run at least 20000, preferably more
 answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04,schaefer=schaefer)
 summcMSY <- summarycMSY(answer,fish,final=TRUE)
 # plotprep(width=8,height=5,newdev=FALSE)
 out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                       oneplot=FALSE,scalar=1.0,plotout=TRUE,
                        plotall=7,schaefer=schaefer)

## End(Not run)  #
```

`pulloutStats` *pulloutStats summaries results from the catch-MSY analysis*

Description

pulloutStats summaries the results from the catch-MSY analysis by generating the mean, minimum, maximum, and quantiles of the resulting r, K, MSY, and last year depletion values. It is used within run_cMSY, but can be used outside by selecting the R1 or Rfirst from the answer obtained from the run_cMSY function.

Usage

```
pulloutStats(inR1, probabs = c(0.025, 0.05, 0.5, 0.95, 0.975), schaefer = 1)
```

Arguments

`inR1`	the input parameter vectors with their respective ok values
`probabs`	the percentiles used in pulling out the quantiles of the r, K, and MSY values; default c(0.025, 0.05, 0.1,0.5,0.9, 0.95, 0.975)
`schaefer`	use the schaefer (when = 1) or approximate Fox (when = 0)

Value

a list containing a matrix of summary statistics regarding the r, K, MSY, and last year depletion values, he biomass trajectories, and the depletion trajectories

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
nyr <- length(fish$year)
answer <- run_cMSY(fish,glb,n=5000,sigpR=0.04)
results <- pulloutStats(answer$R1)
str(results)

## End(Not run)
```

`readdata` *readdata reads in a standard format data-file*

Description

readdata reads in a standard format data-file, as might be generated by using the function dataTemplate, which produces an example data-file which can be used to demonstrate the analyses possible in catchMSY, or can be used as a template to edit and input your own dataset.

Usage

```
readdata(filename, verbose = TRUE)
```

Arguments

`filename`	the filename (including the full path if required) containing the data in the standard format
`verbose`	Default=TRUE, prints out details as data is read in.

Value

a list of two objects, fish and glb. fishincludes at least the year, catch, and cpue, but can also contain other columns, such as SE of cpue, vessels, effort, etc., glb contains the resilience and the spsname

Examples

```
## Not run:
dataTemplate(filename="test.csv", title="A test of the functions")
ans <- readdata("test.csv")
str(ans)

## End(Not run)           # filename="sardine.csv";
```

`removeEmpty` *removeEmpty removes empty strings from a vector of strings*

Description

removeEmpty removes empty strings from a vector of strings. Such spaces often created by spurious commas at the end of lines. It also removes strings made up only of spaces and removes spaces from inside of inidivdual chunks of text.

Usage

```
removeEmpty(invect)
```

Arguments

`invect`	a vector of input strings, possibly containing empty strings

Value

a possibly NULL vector of strings

Examples

```
x <- c("1","","2","","   ","3"," ","4","","a string","end")
x
length(x)
length(removeEmpty(x))
removeEmpty(x)
```

---

runcatchMSY *runcatchMSY - sets up and runs the catch-MSY simulations*

Description

runcatchMSY - sets up and runs the catch-MSY simulations The input data is merely a matrix with, as a minimum, a column of years, and a column of catches, the input object 'glob' needs to contain the resilience. Strictly it only needs the resilience. runcatchMSY calls sraMSY, which, in turn calls oneSRA. Eventually it calls plottrajectory to plot out results. This code derives from example code produced by Martell and Froese (2013) A simple method for estimating MSY from catch and resilience Fish and Fisheries 14:504-514.

Usage

```
runcatchMSY(
  indat,
  glob,
  n = 10000,
  incB = 0.025,
  sigpR = 0.025,
  multK = 1,
  finaldepl = NA,
  start_k = NA,
  start_r = NA,
  initdepl = NA,
  maximumH = 1,
  Hyear = NA,
  schaefer = 1
)
```

Arguments

`indat`	- a data.frame, with at least a 'catch' column containing catch at time t, a 'year' column for year. The 'fish' object from the standard data file or readdata.
`glob`	the globals list 'glb', from readdata or from one of the included data sets. It contains at least a 'resilience' object for resilience, which is either 'verylow', 'low', 'medium", or 'high', and finally a 'spsname' object, which, not surprisingly, is the name of the species concerned.
`n`	- defaults to 10000; the number of random selections of r and K. Defines the number of replicate searches within the parameter space.

`incB`	the increments between the bounds of the initial biomass depletion levels; defaults to 0.025, but have used 0.05 previously
`sigpR`	the measure of process error added to the dynamics. If set to a very small value, 1e-06, the model will act as deterministic.
`multK`	a multiplier for K to allow for stock biomass to rise above K, defaults to 1.0, ie. K is the upper limit.
`finaldepl`	this allows the option of externally setting the final depletion where there have been major reductions in catch that have not been due to a reduction in the stock; defaults to NA, which sets the finaldepl to the pre-defined priors
`start_k`	allows an option to alter the starting K values; for example in Orange Roughy, gigantic initial catches possibly make up a significant proportion of the initial biomass so multiplying by 60 or 100 will lead to ridiculous initial K values. A vector of two numbers is required. The default is NA, which means it will be c(maxcatch,60*maxcatch)
`start_r`	allows for altering the default starting r values; for example in a species though to be of resilience verylow there may be uncertainty over how low and one might want a range from say 0.01 - 0.3 instead of 0.015 - 0.125. The default is NA, which implies the schedule of values in to code will be used. A vector of two numbers is required.
`initdepl`	this allows the option of externally setting the initial depletion. This may be useful where there is evidence that the stock really has been unfished and is expected to be much closer to 100 than the default setting of c(0.5, 0.975); defaults to NA, which sets the initdepl to the pre-defined priors
`maximumH`	upper limit of harvest rate included in the constraints; defaults to 1.0, which implies no upper limit.
`Hyear`	is the index to the year in which a range of harvest rates is to be used to constrain the acceptable trajectories.
`schaefer`	use the schaefer (when = 1) or approximate Fox (when = 0)

Value

plots up nine graphs summarizing catches, r, K, and MSY. returns a list containing the vector of relative counts of successful trajectories for different combinations of r and K.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
schaefer=1
reps <- 5000  # one would run at least 20000, preferably more
answer <- runcatchMSY(fish,glb,n=reps,sigpR=0.04)
summcMSY <- summarycMSY(answer,fish,final=TRUE)
str(summcMSY,max.level=1)
out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=FALSE,scalar=1.0,plotout=TRUE,
                      plotall=7,schaefer=schaefer)
plotcMSY6(summcMSY,fish[,"catch"],label=glb$spsname)
ans <- pulloutStats(answer$R1,probabs=c(0.025,0.05,0.5,0.95,0.975))
out <- plotconstC(ans$deplet,endyear=2017,constC=0,console=FALSE,intensity=NA)
outC <- doconstC(answer$R1,constCatch=50,lastyear=2017,console=FALSE,intensity=NA)
```

```
## End(Not run)
```

---

run_cMSY *run_cMSY - sets up and runs the catch-MSY simulations*

Description

run_cMSY - sets up and runs the catch-MSY simulations The input data is merely a matrix with, as a minimum, a column of years, and a column of catches, the input object 'glob' needs to contain the resilience. Strictly it only needs the resilience. run_cMSY calls sraMSY, which, in turn calls oneSRA. Eventially it calls plottrajectory to plot out results. This code derives from example code produced by Martell and Froese (2013) A simple method for estimating MSY from catch and resilience Fish and Fisheries 14:504-514.

Usage

```
run_cMSY(
  indat,
  glob,
  n = 10000,
  incB = 0.025,
  sigpR = 0.025,
  multK = 1,
  finaldepl = NA,
  start_k = NA,
  start_r = NA,
  initdepl = NA,
  maximumH = 1,
  Hyear = NA,
  schaefer = 1
)
```

Arguments

`indat`	- a data.frame, with at least a 'catch' column containing catch at time t, a 'year' column for year. The 'fish' object from the standard data file or readdata.
`glob`	the globals list 'glb', from readdata or from one of the included data sets. It contains at least a 'resilience' object for resilience, which is either 'verylow', 'low', 'medium", or 'high', and finally a 'spsname' object, which, not surprisingly, is the name of the species concerned.
`n`	- defaults to 10000; the number of random selections of r and K. Defines the number of replicate searches within the parameter space.
`incB`	the increments between the bounds of the initial biomass depletion levels; defaults to 0.025, but have used 0.05 previously
`sigpR`	the measure of process error added to the dynamics. If set to a very small value, 1e-06, the model will act as deterministic.
`multK`	a multiplier for K to allow for stock biomasses to rise above K, defaults to 1.0, ie. K is the upper limit.

`finaldepl`	this allows the option of externally setting the final depletion where there have been major reductions in catch that have not been due to a reduction in the stock; defaults to NA, which sets the finaldepl to the pre-defined priors
`start_k`	allows an option to alter the starting K values; for example in Orange Roughy, gigantic initial catches possibly make up a significant proportion of the initial biomass so multiplying by 60 or 100 will lead to ridiculous initial K values. A vector of two numbers is required. The default is NA, which means it will be c(maxcatch,60*maxcatch)
`start_r`	allows for altering the default starting r values; for example in a species though to be of resilience verylow there may be uncertainty over how low and one might want a range from say 0.01 - 0.3 instead of 0.015 - 0.125. The default is NA, which implies the schedule of values in to code will be used. A vector of two numbers is required.
`initdepl`	this allows the option of externally setting the initial depletion. This may be useful where there is evidence that the stock really has been unfished and is expected to be much closer to 100 than the default setting of c(0.5, 0.975); defaults to NA, which sets the initdepl to the pre-defined priors
`maximumH`	upper limit of harvest rate included in the constraints; defaults to 1.0, which implies no upper limit.
`Hyear`	is the index to the year in which a range of harvest rates is to be used to constrain the acceptable trajectories.
`schaefer`	use the schaefer (when = 1) or approximate Fox (when = 0)

Value

plots up nine graphs summarizing catches, r, K, and MSY. returns a list containing the vector of relative counts of successful trajectories for different combinations of r and K.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb
schaefer=1
reps <- 5000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04)
summcMSY <- summarycMSY(answer,fish,final=TRUE)
str(summcMSY,max.level=1)
out <- plottrajectory(answer$R1,fish$year,fish$catch,answer$parbound,
                      oneplot=FALSE,scalar=1.0,plotout=TRUE,
                      plotall=7,schaefer=schaefer)
plotcMSY6(summcMSY,fish[,"catch"],label=glb$spsname)
ans <- pulloutStats(answer$R1,probabs=c(0.025,0.05,0.5,0.95,0.975))
out <- plotconstC(ans$deplet,endyear=2017,constC=0,console=FALSE,intensity=NA)
outC <- doconstC(answer$R1,constCatch=50,lastyear=2017,console=FALSE,intensity=NA)

## End(Not run)
```

sraMSY — *sraMSY sets up input parameters and data for makebiomC within run_cmSY*

Description

sraMSY sets up the run parameters in terms of input parameter bounds, the number of replicates, and other parameters. Not exported but can be viewed using catchMSY:::sraMSY. The code for makebiomC cannot be seen without looking at the source package because it is written in C++, but its help can be seen using catchMSY:::makebiomC

Usage

```
sraMSY(theta, N, startbd, nyr, ct, yr, mult, maxH, Fyear, schaefer)
```

Arguments

`theta`	a list containing the initial parameter values including the initial r range (2 numbers), the initial k range (2 numbers), the final depletion range (2 numbers), and the recruitment variability
`N`	the number of replicate searches across the parameter space
`startbd`	the set of initial starting depletion levels
`nyr`	the number of years of catch data
`ct`	the vector of catches per year
`yr`	the vector of years
`mult`	a multiplier for K to allow for stock biomasses to rise above K; default value = 1.0, so K is the default upper limit
`maxH`	upper limit of harvest rate included in the constraints; defaults to 1.0, which implies no upper limit.
`Fyear`	is the index to the year in which a rnage of harvest rates is to be used to constrina the acceptable trajectories.
`schaefer`	use the schaefer (= 1) or approximate Fox (when = 0)

Value

a list containing itheta (the vectors of parameters), elltot (a matrix of the yes/no vectors for each initial biomass depletion level), and biomass (the biomass trajectories for each of the N parameter vectors) # theta=parbound;N=n;startbd=startbd;nyr=nyr;ct=ct;yr=yr;mult=multK; maxH=1.0;Fyear=Hyear

summarycMSY *summarycMSY makes tables of msy,r,K,meanr,meanK,and all picks*

Description

summarycMSY generates a list of countcolour, meanmsy, meanr, meanK, r, K, msy, pickC (a list of pickblk, pickblu, pickyel,pickmax, rnot, knot), and years

Usage

```
summarycMSY(ans, fish, final = TRUE)
```

Arguments

`ans`	the object output from run_cMSY
`fish`	the input data often named fish
`final`	whether or not to consider the first phase or the final phase, the default is TRUE, which considers the final phase

Value

a list of 12 objects used to summarize and plot the output. Returned invisibly; need to first allocate to an object.

Examples

```
## Not run:
data(invert)
fish <- invert$fish
glb <- invert$glb

nyr <- length(fish$year)
reps <- 5000  # one would run at least 20000, preferably more
answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04)
summcMSY <- summarycMSY(answer,fish,final=TRUE)
str(summcMSY,max.level=1)

## End(Not run)  # ans=answer; fish=fish; final=TRUE
```

trendMSY *trendMSY calculates the mean MSY per K class*

Description

trendMSY subdivides the range of the successful K values into a set of classes of width ’inc’ and for each class calculates the mean (geometric mean) of the MSY for the subset of r and K values. This enables the central value of MSY to be plotted on the scatter of successful points.

Usage

```
trendMSY(inr, inK, inc = 100, schaefer = 1)
```

Arguments

`inr`	the set of r values to be tested, derives from summarycMSY
`inK`	the set of K values to be tested, derives from summarycMSY
`inc`	the class width of the K classes, default=100
`schaefer`	use Schaefer=1 or Fox model = 0, default=1

Value

a matrix of r, K and MSY values

Examples

```
## Not run:
 data(invert)
 fish <- invert$fish
 glb <- invert$glb
 reps <- 5000  # one would run at least 20000, preferably more
 schaef <- 1
 answer <- run_cMSY(fish,glb,n=reps,sigpR=0.04,schaefer=schaef)
 summcMSY <- summarycMSY(answer,fish,final=TRUE)
 trendMSY(summcMSY$r,summcMSY$K,inc=200,schaefer=schaef)

## End(Not run)
```

`which.closest` *which.closest find the number closest to a given value*

Description

which.closest finds either the number in a vector which is closest to the input value or its index value

Usage

```
which.closest(x, invect, index = T)
```

Arguments

`x`	the value to lookup
`invect`	the vector in which to lookup the value x
`index`	should the closest value be returned or its index; default=TRUE

Value

by default it returns the index in the vector of the value closest to the input value

Examples

```
vals <- rnorm(100,mean=5,sd=2)
pick <- which.closest(5.0,vals,index=TRUE)
pick
vals[pick]
vals[c(pick-5):(pick+5)]
which.closest(5.0,vals,index=FALSE)
```

Index