PBStools: User's Guide

Rowan Haigh and Jon T. Schnute

April 30, 2013

Contents

List of Tables

List of Figures

Page left blank for printing purposes.

What is PBStools?

PBStools provides an R interface for algorithms commonly used in fisheries. The scope of this package is by no means comprehensive, having grown from the need to satisfy tasks specific to British Columbia (BC) ocean fisheries. Many of the functions provide a quick way to visualize data, and in some cases perform preliminary analyses. Though oriented to users at the Pacific Biological Station (PBS), these functions may provide a broad utility to users at other locales. The User Guide is organised into sections that loosely classify the functions by theme – (1) Utility, (2) Biology, (3) Fishery, (4) Survey, (5) Spatial, (6) Temporal, and (7) Catch Reconstruction. Within each section, the functions are described alphabetically.

PBStools depends heavily on two other R package: **PBSmapping** (Schnute *et al.*, 2004a; Boers *et al.*, 2004) and **PBSmodelling** (Schnute *et al.*, 2006). We use the latter to implement several Graphical User Interfaces (GUIs) that facilitate a few select functions. Most functions, however, are designed for use on the command line or in sourced code. Many of the functions invisibly return auxiliary output in a list object called `PBStool`, which is located in the temporary working environment `.PBStoolEnv`. Accessor functions called `ttget`, `ttcall`, and `ttprint` enable the user to get, call, and print objects from the `.PBStoolEnv` environment, while `ttput` puts (writes) obects to `.PBStoolEnv`.

Amongst the various functions in the package, we include those that implement some of the models published by the authors. For example, `calcSG` estimates parameters for the Schnute growth model (Schnute, 1981), which offers a versatile alternative to the von Bertalanffy growth model (`calcVB`). The functions presented in the Survey section make use of the binomial-gamma model (Schnute and Haigh, 2003) for characterizing survey strata populations when designing or simulating groundfish trawl surveys. The function `runCCA` implements the composition analysis of Schnute and Haigh (2007) to estimate total mortality Z of rockfish (*Sebastes* spp.) using three combined effects: survival at age (S_a), fishery selectivity at age (β_a), and relative recruitment anomalies (R_a). Finally, the function `buildCatch` provides an efficient method for reconstructing rockfish catch from 1918 to the present along the BC coast (Haigh and Yamanaka, 2011).

Also available in the package directory `./library/PBStools/sql` we provide a number of useful SQL queries for DFO[1] commercial fisheries databases – PacHarvest (trawl, 1996–2007), PacHarvHL (hook and line, 1994–2006), GFCatch (historical landings, 1954–1995), GFBioSQL (biological samples, 1946–2012), and GFFOS (integrated fisheries, 2002–present). To launch SQL queries, **PBStools** relies on the R package **RODBC**. If you have access to the DFO network and have privileges to query these databases, the function `getData` can send the queries to the remote servers and return a data frame called `PBSdat` in the global environment. In the document, we highlight queries for DFO personnel using text with a background shaded `moccasin`[2]. (Examples are shaded `aliceblue` and console output is shaded `honeydew`.) Note that many of these queries might act as useful templates for users outside DFO for similar purposes. Querying databases directly via SQL

[1] Department of Fisheries and Oceans, a.k.a. Fisheries and Oceans Canada

[2] RGBs: moccasin=(255,228,181), aliceblue=(240,248,255), honeydew=(240,255,240)

commands from R usually proves much more efficient than launching Microsoft Access Queries from a front-end database shell.

Originally, **PBStools** evolved over time (2007–2012) within the R package **PBSfishery**[3], along with a convenient Graphical User Interface (GUI) tool for interacting with **PBSmapping** and useful datasets (regional boundaries, key codes, example data). In April 2012, we decided to split **PBSfishery** into three separate libraries – **PBStools**, **PBSmapx**, and **PBSdata** – for distribution on CRAN.

What is PBS?

The Pacific Biological Station is the oldest fisheries research centre on Canada's Pacific coast and forms part of a network of nine major scientific facilities operated by Fisheries and Oceans Canada. Located in Nanaimo, British Columbia, the Station is home to scientists, technicians, support staff and ships' crews whose common interests are the coastal waters of British Columbia, the Northeast Pacific Ocean, the Western Arctic and navigable waters east to the Manitoba, Saskatchewan border.

PBS was established in 1908 and is the principal centre for fisheries research on Canada's west coast. There are some 22 structures on the site including a four-story office/wet lab building, specialty storage structures for hazardous chemicals and salt water pumping facilities. PBS maintains a number of workshops for research support. There is a wharf used for loading, unloading, and berthage of research vessels, as well as a small boat dock for inshore research boats. PBS also maintains a library and meeting facilities. Aquatic facilities, primarily used by Aquaculture Science, include ambient temperature and heated salt water and fresh water.

Research at PBS responds to stock assessment, aquaculture, marine environment, habitat, ocean science, and fish productivity priorities. Some fisheries management activities are also conducted here.

Figure 1. Pacific Biological Station (PBS), Nanaimo BC

For more information, see:
http://www.pac.dfo-mpo.gc.ca/science/facilities-installations/pbs-sbp/index-eng.htm.

[3]found on Google's Project Hosting site: http://code.google.com/p/pbs-fishery/

1 Utility functions

The utility functions described in this section cover a broad range of activities. The number of functions has grown over time and inevitably, some of the functions may replicate the behaviour of functions in other packages. Additionally, the utility of these functions can either decline or cease based on changes in dependent packages. Feedback is appreciated from any users of **PBStools**.

1.1 biteData

Subset a matrix or data frame using a vector object with the same name as one of the fields in the data matrix/frame and return the subset data matrix/frame. If there are no fields with the name of the vector object, the data matrix/frame is returned unaltered.

Due to the current nature of the algorithm, the user cannot supply an ad hoc vector directly as the argument. This means the vector object must exist before calling `biteData`.

Not allowed: `biteData(swiss,Education=1:5)`
Allowed: `Education=1:5; biteData(swiss,Education)`

Example (biteData):

```
pbsfun=function(){
  cat("Records in original swiss object by Education\n")
  print(sapply(split(swiss$Education,swiss$Education),length))
  Education=1:5; test=biteData(swiss,Education)
  cat("Records in swiss object subset where Education=1:5\n")
  print(sapply(split(test$Education,test$Education),length))
  invisible() }
pbsfun()
```

Output (biteData):

```
Records in original swiss object by Education
 1  2  3  5  6  7  8  9 10 11 12 13 15 19 20 28 29 32 53
 1  3  4  2  4  7  4  3  2  1  5  3  1  1  1  1  2  1  1
Records in swiss object subset where Education=1:5
1 2 3 5
1 3 4 2
```

1.2 chewData

Remove records from a data frame or matrix that contribute little information to unique categories of the factor specified. Records are removed if a specified factor's categories don't number more than the specified minimum. If there are no fields with the name of the specified factor, the data matrix/frame is returned unaltered.

Example (chewData):

```
pbsfun=function(){
  cat("Unique records in 'iris$Petal.Width'\n")
  print(sapply(split(iris$Petal.Width,iris$Petal.Width),length))
  test=chewData(iris,"Petal.Width",5)
  cat("'Petal.Width' categories fewer than 5 removed\n")
  print(sapply(split(test$Petal.Width,test$Petal.Width),length))
  invisible() }
pbsfun()
```

Output (chewData):

```
Unique records in 'iris$Petal.Width'
0.1 0.2 0.3 0.4 0.5 0.6   1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9   2 2.1 2.2
  5  29   7   7   1   1   7   3   5  13   8  12   4   2  12   5   6   6   3
2.3 2.4 2.5
  8   3   3
'Petal.Width' categories fewer than 5 removed
0.1 0.2 0.3 0.4   1 1.2 1.3 1.4 1.5 1.8 1.9   2 2.1 2.3
  5  29   7   7   7   5  13   8  12  12   5   6   6   8
```

1.3 collectFigs

Collect figures (currently only encapsulated postscript `.eps` supported) into one document using a latex compiler (`latex.exe`, `dvips.exe`, `ps2pdf.exe`). The code constructs a `.tex` file that uses all available `.eps` files in the specified directory. The final result is a bookmarked PDF file called `<fout>.pdf`, where `fout` is user-supplied prefix for the output file.

1.4 confODBC

Use a command line utility found in Windows OS called `odbcconf.exe` that configures an ODBC Data Source Name from the system's command line. The function runs the command line utility with the user-supplied information, and configures a User DSN on the his/her computer.

In Windows XP, the new User DSN can be seen by navigating to:
<Control Panel><Administrative Tools><Data Sources (ODBC)> (Figure 2).

Example (confODBC):

```
pbsfun=function(os=.Platform$OS.type,dfo=TRUE) {
  if (os=="windows" && dfo)
    confODBC(dsn="Popsicle", server="GFDB", db="PacHarvest",
      driver="SQL Server", descr="Icy sweet nonsense", trusted=TRUE)
  else showMessage("Only functional for DFO personnel using Microsoft Windows OS")
  invisible() }
pbsfun()
```

1.5 convFY

Convert a vector of dates into a vector of fishing or fiscal years based on a specified month to start the fishing/fiscal year. The function returns a numeric vector of

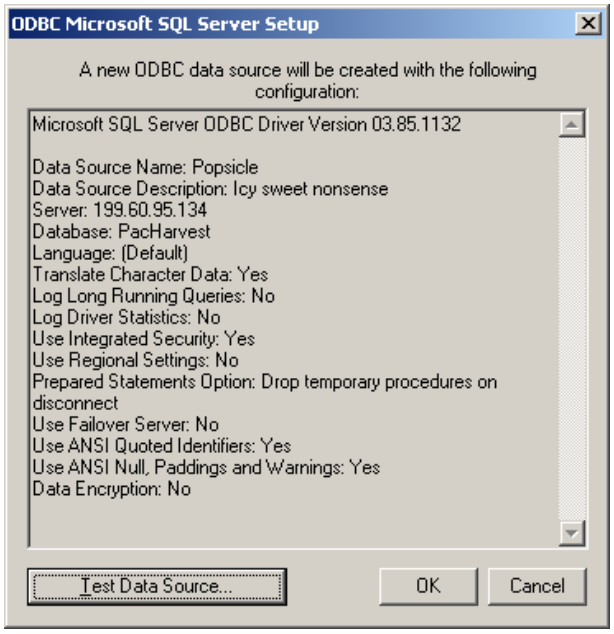

Figure 2. (*PBStools-confODBC*) ODBC data source configuration for DSN "Popsicle".

fishing/fiscal years of equal length to the input vector of dates. The output vector has names representing year-month `YYYY-MM`. If the starting month is set to 1, the output dates also represent calendar years.

Example (convFY):

```
pbsfun=function() {
  cat("Fishing years starting in April\n")
  print(convFY(paste("2009-",pad0(1:12,2),"-15",sep="")))
  invisible() }
pbsfun()
```

Output (convFY):

```
Fishing years starting in April
2009-01 2009-02 2009-03 2009-04 2009-05 2009-06 2009-07 2009-08 2009-09
   2008    2008    2008    2009    2009    2009    2009    2009    2009
2009-10 2009-11 2009-12
   2009    2009    2009
```

1.6 convYM

Convert a 2-element vector of date limits into a vector of year-months where every month within the limits is represented. The function returns an expanded character vector of ordered year-months `YYYY-MM` between and including the specified date limits. This character vector can be used for labelling and other factor-like routines.

Example (convYM):

```
pbsfun=function() {
  cat("Year-month labels between the specified date limits\n")
  print(convYM(c("2009-07-01","2011-11-11")))
  invisible() }
pbsfun()
```

Output (convYM):

```
Year-month labels between the specified date limits
 [1] "2009-07" "2009-08" "2009-09" "2009-10" "2009-11" "2009-12" "2010-01"
 [8] "2010-02" "2010-03" "2010-04" "2010-05" "2010-06" "2010-07" "2010-08"
[15] "2010-09" "2010-10" "2010-11" "2010-12" "2011-01" "2011-02" "2011-03"
[22] "2011-04" "2011-05" "2011-06" "2011-07" "2011-08" "2011-09" "2011-10"
[29] "2011-11"
```

1.7 convYP

Convert a vector of dates into one of year-periods through binning the dates into intervals specified in days. The routine actually fudges the integer day interval to a real number that will create even breaks. The function returns a vector (with the same length as the input date vector) that specifies dates as real numbers where the integer part is the year and the fraction part is the proportion of the year corresponding to the bin's right-most value. Each element of the vector also has a name that specifies the year and bin number.

Example (convYP):

```
pbsfun=function() {
  x=paste("2009-",c("03-31","06-30","09-30","12-30"),sep="")
  cat("Year periods based on numeric value of input dates\n")
  print(convYP(x,30))
  invisible() };  pbsfun()
```

Output (convYP):

```
Year periods based on numeric value of input dates
2009-03 2009-06 2009-09 2009-12
2009.25 2009.50 2009.75 2010.00
```

1.8 createDSN

Create a suite of User DSNs for PBS groundfish databases, currently hosted on the server SVBCPBSGFIIS. This function uses `confODBC` and will overwrite existing DSNs if they exist. Currently, six DSNs for DFO databases will be created:

`GFBioSQL` `GFCatch` `GFCruise` `PacHarvest` `PacHarvHL` `PacHarvSable`

In XP, the new User DSNs can be seen by navigating to:
<Control Panel><Administrative Tools><Data Sources (ODBC)> (Figure 3).

Example (createDSN):

```
pbsfun=function(os=.Platform$OS.type,dfo=FALSE) {
  if (os=="windows" && dfo) createDSN()
  else showMessage("Only functional for DFO personnel using Microsoft Windows OS")
  invisible() }
pbsfun()
```

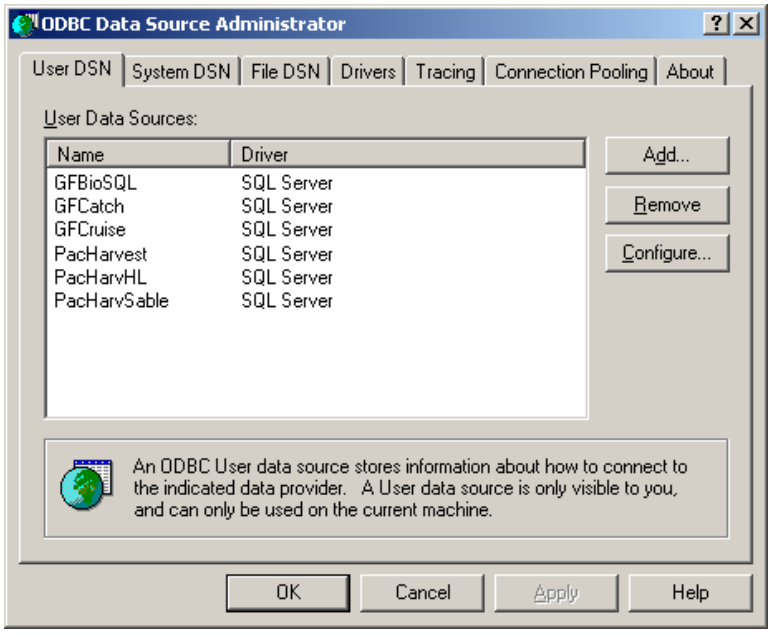

Figure 3. (*PBStools-createDSN*) ODBC data source configuration for DFO groundfish database DSNs.

1.9 crossTab

Run a cross tabulation on a data frame, summarizing z-values by specified 'factors'. This function requires the R package **reshape**, and uses the functions `melt.data.frame` and `cast` therein. The output reports a cross tabulation of `y` with values `z` summarized by `func`. The function `crossTab` is useful for summarizing data query results (e.g., total catch by year and PMFC major area) when exploring / debugging functions like `buildCatch`.

1.10 fitLogit

Fit binomial data using a `logit` link function in a `glm`. The function returns a GLM fit of the binomial data using the specified independent observations. See the 'Value' section of the `glm` help file in the **stats** package.

1.11 flagIt

Take a coordinate `(a,b)` and label it using a diagonal line of radius `r` and angle `A`. A diagonal dotted line in light grey radiates out from a central coordinate and the coordinate values are used to label the coordinate itself. The function adjusts for the aspect ratio created by different x- and y-limits, and for different x- and y-dimensions of the plot. This

ensures that an angle looks correct in the plotting space. The code invisibly returns a list of the vectors (a,x) and (b,y), the angle in radians, the original x-value calculated in a square Cartesian system, and the final x- and y-coordinates on the periphery of a circle.

Example (flagIt):

```
pbsfun = function() {
  plot(0,0,type="n",xlim=c(0,20),ylim=c(0.2,0.8))
  points(10,0.5,pch=20,col="blue")
  for (i in seq(10,360,10))
    flagIt(a=10, b=0.5, r=0.25, A=i,col="blue",cex=0.7)
}; pbsfun()
```

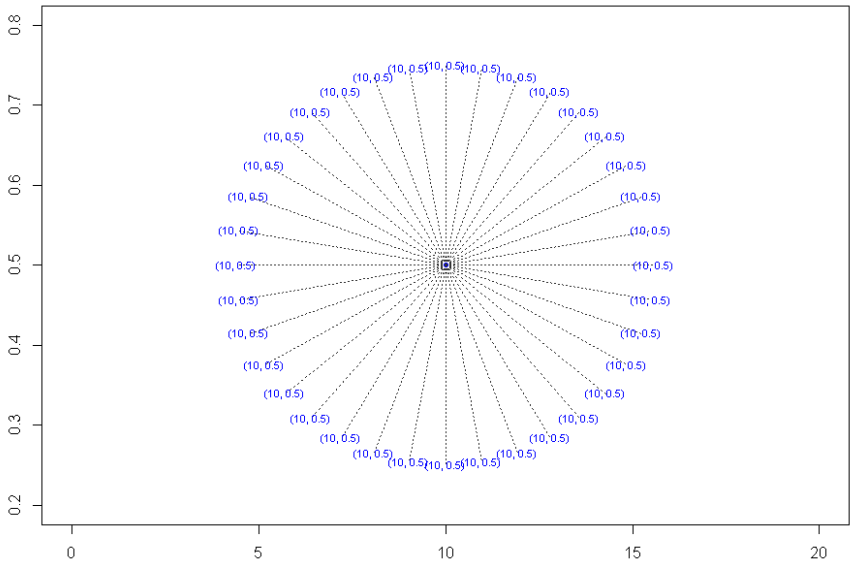

Figure 4. (*PBStools-flagIt*) The point (10, 0.5) flagged using vectors at 10° angles.

1.12 getData

Attempt to get data from a variety of types (e.g., data objects/files, queries/tables) and remote sources (`ORA`, `SQL`, `MDB`, `DBF`, `XLS`, `FILE`). The data table retrieved is placed in a global data frame object called `PBSdat`. If the type is `MDB`, the query/table name and the database name are attached as attributes. If the type is `SQL` or `ORA`, the DB name, query name, and SQL code are attached as attributes. This function, which remains one of the key utilities in **PBStools** for retrieving data, is essentially a glossy wrapper for the function `sqlQuery` in the package **RODBC**.

There are a few quirks that a user should keep in mind. This function was designed primarily to access remote SQL Server and Oracle databases using files containing structured query language (SQL). Unfortunately, SQL syntax varies between the two database systems but the differences are fewer than the similarities. Invariably, the Oracle system offers the most challenges to one's sanity.

A major disadvantage is the inabilty in Oracle to create temporary tables that exist only when the query is run. In SQL Server, temporary tables adopt the hatch/pound prefix

'#' and are incredibly useful for running pieces of code that generate results for use by subsequent pieces of code. In Oracle, result subsets must take the form of embedded `SELECT` statements, which can get get very messy very quickly.

Another feature of Oracle is the need to prefix table names with the name of the database. The function `getData` does this automatically by replacing the variable `@table` in the SQL code file with the value from the argument `dbName`. The user could also spell out the table name when building a query file (e.g., `GFFOS.GEAR` instead of `@table.GEAR`). If the user chooses to send an SQL expression directly (not using an SQL code file) to an Oracle database (i.e., `type="ORAX"`), then the user must include the table name prefix explicitly.

Although, `getData` constructs an ODBC connection string, Oracle still needs information in a file call `tnsnames.ora` from a local installation of an Oracle client program (see http://www.orafaq.com/wiki/Tnsnames.ora). As we are biologists rather than computer programmers, it remains unclear to us whether the Oracle software installation is strictly necessary. It could be that RODBC simply needs the information in a file called `tnsnames.ora` that exists on the user's path. Feedback would be welcome on this issue.

1.12.1 Argument details

`fqtName`	string specifying name of file, query, or table; can also be an explicit SQL statement when `TYPE=SQLX` or `TYPE=ORAX`.
`dbName`	string specifying the name of a remote database; types supported: `XLS` (Excel), `MDB` (ACCESS), `SQL` (Sequel Server), `ORA` (Oracle).
`strSpp`	string specifying species code for the SQL variable `@sppcode`; IMPORTANT: to retrieve a remote table or query rather than send SQL code, set `strSpp=NULL`.
`server`	string specifying the name of a remote server (e.g., SVBCPBSGFIIS for an SQL server, GFSH or ORAPROD for an Oracle server.
`type`	type of file: `FILE` = local data object, a saved binary file `*.rda`, a dumped ASCII file `.r`, a comma-delimited file `.csv` or `.txt`; specify the file name without the extension. `XLS` = Microsoft Excel spreadsheet (specific worksheet is specified by `fqtName`). `DBF` = Dbase IV table. As each `.dbf` file acts like a separate table, use `fqtName` to specify the `.dbf` file without its extension. `MDB` = Microsoft ACCESS database query or table. `SQL` = SQL Server SQL query (code file) or table. `ORA` = Oracle SQL query (code file) or table. `SQLX` = SQL Server SQL query code (direct expression). `ORAX` = Oracle SQL query code (direct expression).
`path`	string specifying path to local file, MDB database, or SQL query code file.
`trusted`	logical: if `TRUE`, allow SQL Server to use a trusted DFO login ID.
`uid, pwd`	user ID and password for authentication (if required).

`noFactors`	logical: if `TRUE`, convert all factor fields to character fields.
`noLogicals`	logical: if `TRUE`, convert all logical fields to characters "T" or "F".
`rownum`	numeric indicating how many rows of a table to return. The default 0 means all rows are returned (entire table); this argument only affects downloads of remote database tables.
`mindep`	numeric specifying minimum depth for the SQL variable `@mindep`.
`maxdep`	numeric specifying maximum depth for the SQL variable `@maxdep`.
`surveyid`	numeric specifying survey ID in GFBio for the SQL variable `@surveyid`.
`survserid`	numeric specifying survey series ID in GFBio for the SQL variable `@survserid`.
`fisheryid`	numeric specifying fishery ID number for the SQL variable `@fisheryid`.
`logtype`	string specifying log type code for the SQL variable `@logtypeval`.
`doors`	numeric specifying door spread width for the SQL variable `@doorsval`.
`speed`	numeric specifying vessel trawling speed for the SQL variable `@speedval`.
`mnwt`	numeric specifying mean weight (g) of a species for the SQL variable `@mnwt`.
`tarSpp`	string specifying species code(s) for the SQL variable `@tarcode`.
`major`	numeric specifying PMFC major area codes(s) for the SQL variable `@major`.
`top`	numeric specifying top N records for the SQL variable `@top`.
`dummy`	numeric or character to use *ad hoc* wherever the SQL variable `@dummy` appears.
`...`	additional arguments for `RODBC`'s function `sqlQuery` (specifically if the user wishes to specify `rows_at_time=1`).

1.13 getFile

Attempt to get data from the specified file name (without specifying the extension). If a data object with this name exists in the specified target environment (local or global), the function will not attempt to reload it unless the users specifies `reload=TRUE`. Failing to find an R data object, the function tries the binary libraries for package data objects, unless the user specifies `ignore.data=TRUE`. Thereafter, the code searches for remote data files in the following order: (i) local binary files with the extension `.rda`, (ii) ASCII data files created using `dump` and with the extension `.r`, (iii) comma-data delimited files `.csv`, and (iv) ordinary text files `.txt`.

Depending on the choice of scope (`L` = local, `G` = global), the function places the newly acquired data into an R-object with the same name as that used for the search into either the parent frame or the global environment, respectively.

1.13.1 Argument details

`...`	a sequence of names or literal character strings.
`list`	a character vector of potential objects.
`path`	string specifying path to data files `*.rda` and `*.r`.
`scope`	string specifying either local `"L"` or global `"G"` placement (destination) of data file.
`use.pkg`	logical: if `TRUE` ignore binaries called by the `data` command.
`reload`	logical: if `TRUE`, force a reloading of the data object into R's memory.
`try.all.frames`	logical: if `TRUE`, look for named object(s) in all frames, starting from the parent frame and working backwards.
`use.all.packages`	logical: if `TRUE`, look for data binaries in all the packages installed in the user's R library folder.

1.14 getName

Get string names from user supplied input. If the name supplied exists as an object in the parent frame, the object will be assessed for its potential as a source of names using the following criteria:

If `fnam` exists as a list, the function returns the names of the list.
If `fnam` exists as a string vector, the function returns the strings in the vector.
If `fnam` does not exist as an object, it simply returns itself as a string.

Example (getName):

```
pbsfun=function() {
  cat("Data object 'swiss' doesn't appear in the parent frame\n")
  print(getName(swiss))
  swiss=swiss
  cat("And now it does, so it acts like a source of names\n")
  print(getName(swiss))
  invisible() }
pbsfun()
```

Output (getName):

```
Data object 'swiss' doesn't appear in the parent frame
[1] "swiss"
attr(,"type")
[1] "literal"
attr(,"len")
[1] 1
And now it does, so it acts like a source of names
[1] "Fertility"        "Agriculture"      "Examination"
[4] "Education"        "Catholic"         "Infant.Mortality"
attr(,"type")
[1] "list"
attr(,"len")
[1] 6
```

1.15 isThere

Check to see whether objects physically exist in the specified environment. This differs from the function `exists` in that the latter sees objects across environments. This function looks in the specified environment and the object must be physically present to elicit a `TRUE` response.

Example (isThere):

```
pbsfun=function() {
  cat("Data object 'swiss' appears to exist in pos=0\n")
  cat(paste("   exists(\"swiss\",envir=sys.frame(0))",
    exists("swiss",envir=sys.frame(0))),"\n")
  cat("But it isn't really there...\n")
  cat(paste("   isThere(\"swiss\",envir=sys.frame(0))",
    isThere("swiss",envir=sys.frame(0))),"\n")
  invisible() }
pbsfun()
```

Output (isThere):

```
Data object 'swiss' appears to exist in pos=0
   exists("swiss",envir=sys.frame(0)) TRUE
But it isn't really there...
   isThere("swiss",envir=sys.frame(0)) FALSE
```

1.16 lenv, penv, genv

These shortcut functions get the local (`lenv`), parent (`penv`), and global (`genv`) environments, respectively.

1.17 listTables

List the tables in a specified SQL, Oracle or Microsoft ACCESS database. User can choose table type and/or search table names through pattern matching. The results of the pkgRODBC function `sqlTables` are dumped to a data frame object called `PBSdat`. If no pattern is specified, all table names in the target database are returned. If a pattern is provided, only table names containing the pattern are returned. Pattern matching is case-sensitive. Additionally, the user can choose table types, depending on the server type.

Example (listTables):

```
pbsfun=function(os=.Platform$OS.type,dfo=FALSE) {
  if (os=="windows") {
    if(dfo) {
      cat("Tables in 'PacHarvest' matching pattern 'Species'\n")
      listTables("PacHarvest",pattern="Species") }
    else {
      cat("Tables in 'Examples.mdb'\n")
      listTables("Examples", type="MDB",ttype=c("TABLE","VIEW"),
        path=.getSpath()) } }
  else showMessage("If logged onto DFO network, set argument 'dfo=T'")
  invisible() }
pbsfun()
```

Output (listTables):

```
Tables in 'Examples.mdb'
"Ex01_Sample_Info"   "Ex02_Species_Abundance"   "Ex03_Portfolio"
```

1.18 makeLTH

Make a longtable header for printing an `xtable` in Sweave, source:
http://tex.stackexchange.com/questions/41067/caption-for-longtable-in-sweave?rq=1

This code was provided by a clever Swedish fellow (pseudonym = `chepec`) to elegantly break a longtable across pages by specifying *Continued on next page* as a footer where the table breaks, and changes the caption on the following page to "Table xx – continued from previous page".

1.19 prime

Report the prime numbers given an integer vector. If none are found, the function returns `NULL`. The function will reject non-vector objects, non-numeric vectors, and numeric vectors that do not comprise integers.

1.20 revStr

Reverse a string character by character. Code taken from examples in the **base** package `strsplit`.

Example (revStr):

```
pbsfun=function() {
  cat(revStr("Nardwuar the Human Serviette"),"\n")
  invisible() }
pbsfun()
```

Output (revStr):

```
etteivreS namuH eht rauwdraN
```

1.21 runModules

Display an interactive GUI to run the various modules in **PBStools** (Figure 5).

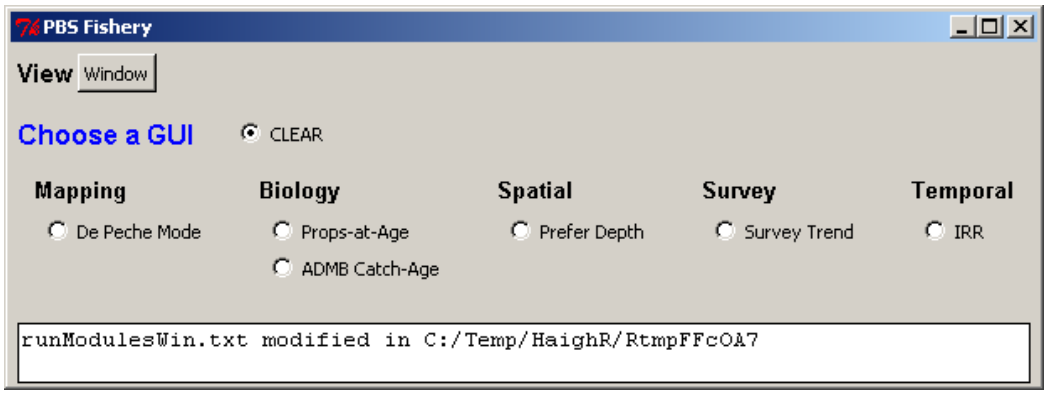

Figure 5. (*GUI-runModules*) GUI menu control to run available **PBStools** GUIs.

1.22 scaleVec

Scale a vector to span a target minimum and maximum (see **PBSmodelling** functions `scalePar` and `restorePar`.)

This function combines the utility of **PBSmodelling**'s `scalePar` and `restorePar` so that a user can rescale a vector to lie between any minimum and maximum value (including `c(0,1)`). The target minimum must be less than the target maximum, otherwise the function stops execution. The output is a numeric vector `V` re-scaled to range from `Tmin` and `Tmax`. Figure 6 shows the transformation of a random uniform vector between 100 and 1000 (x-axis) to lie between -3 and 5 (y-axis).

Example (scaleVec):

```
pbsfun=function() {
  V = runif(50,100,1000)
  T = scaleVec(V, Tmin=-3, Tmax=5)
  lmf = lm(T~V)
  par0 = par(no.readonly=TRUE); on.exit(par(par0))
  expandGraph(cex=1.2,mgp=c(1.8,0.5,0))
  plot(V,T,type="n",cex.lab=1.5)
  abline(lmf,col="dodgerblue",lwd=2)
  points(V,T,pch=21,cex=1.2,col="navyblue",bg="yellow")
  expr=paste(c("addLabel(0.1,0.85,expression(T == ",round(lmf$coeff[1],5),
    " + ",round(lmf$coeff[2],5),"* V~~~~R^2 == ",
    round(summary(lmf)$adj.r.squared,2),"),adj=0)"),collapse="")
  eval(parse(text=expr))
  invisible(expr) }
pbsfun()
```

1.23 showError, showMessage

Display a message on the current device. If `err==TRUE`, code execution stops. A wrapper function called `showError` is included for backwards compatibility.

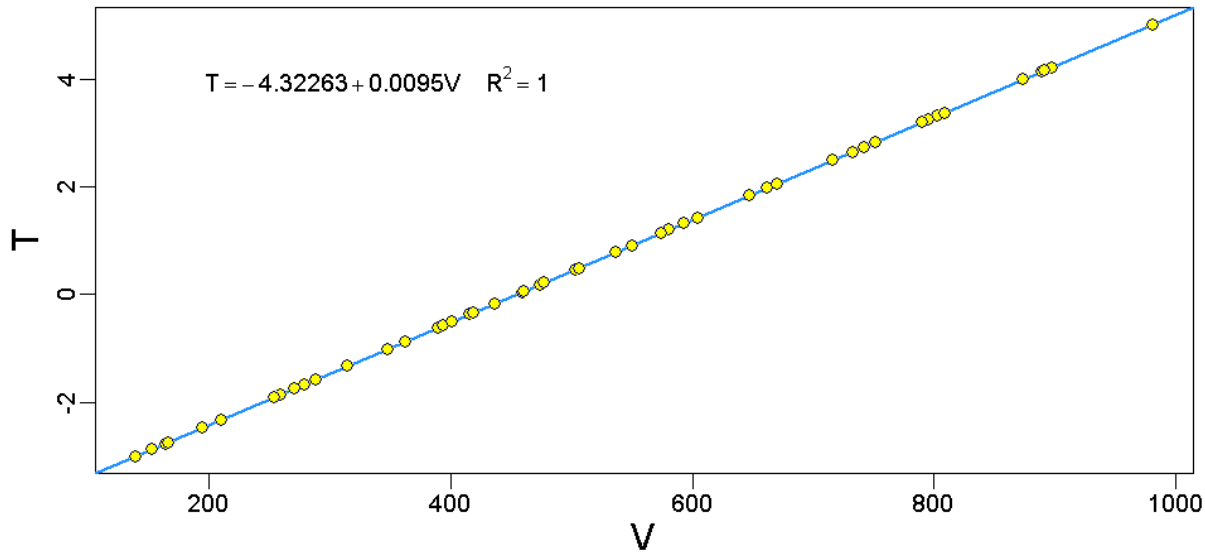

Figure 6. (*PBStools-scaleVec*) A sample of 50 points from the random uniform distribution betweeen 100 and 1000 rescaled to lie between -3 and 5.

1.24 spooler

Spool a named list or list of lists into a new field of an existing data frame where the names of the lists match existing fields in the target data frame. The elements of the list are values to match in the existing fields. The values placed in `newfld` are either matched values (if a simple list is supplied) or list names concatenated with matched values (if a list of lists is supplied).

This function is most useful when multiple fields exist that describe a similar concept. For example area might be any combination of the fields `major`, `minor`, `locality`, `srfa`, `srfs`, and `popa`. Obviously some of these overlap and the user has to be mindful of which combinations to collapse into one field. If multiple matches are available, the code gives preference to the first search field in the list. For example, `area` in lines 4 and 5 of the output below can either be `major-7+8` or `srfs-MR`, but preference is given to the former because the call supplied a list with `major` before `srfs`.

Example (spooler):

```
pbsfun=function(){
  data(pop.age)
  temp=pop.age[sample(1:nrow(pop.age),10),]
  spooler(list(major=list(3:4,7:8),srfs=list("GS","MI","MR")),"area",temp)
  print(temp)
  invisible() }
pbsfun()
```

Output (spooler):

```
       EID         X       Y    SID ttype stype       date year sex mat
 1  316045        NA      NA 184123     4     2 2001-06-28 2001   2   6
 2  354633 -129.3400 51.3192 227794     4     6 2003-05-24 2003   2  NA
 3   19177        NA      NA  50125     1     0 1990-06-25 1990   2   2
 4  106475        NA      NA  49478     1     2 1979-05-28 1979   1   7
 5    5662        NA      NA  49550     1     2 1980-06-25 1980   2   7
 6  157884        NA      NA  50102     1     2 1990-03-01 1990   2   4
 7  648040 -129.3825 51.2942 329663     2     5 1982-11-15 1982   1   1
 8  215189 -128.7558 51.3392 154697     2     0 1984-08-15 1984   2   2
 9  174570 -130.7317 51.7942 115700     4     2 1999-08-22 1999   2  NA
10  195574 -130.7400 51.7717 137896     4     2 2000-06-01 2000   1  NA
    age ameth len  wt gear major    catch srfa srfs      area
 1   50     3 460  NA    1     6 47174.09  5CD   MR   srfs-MR
 2    8     3 335  NA    1     6  6173.39  5AB   GS   srfs-GS
 3   12     3 410  NA    1     6  4436.00  5AB   GS   srfs-GS
 4   26     3 440  NA    1     7 13658.00  5CD   MR major-7+8
 5   31     3 470  NA    1     7 25141.00  5CD   MR major-7+8
 6   14     3 420  NA    1     6  8661.00  5CD   MR   srfs-MR
 7    6     3 280 305    1     6   284.00  5AB   GS   srfs-GS
 8    7     3 330  NA    1     5   319.00  5AB   GS   srfs-GS
 9   25     3 440  NA    1     6  4422.59  5CD   MR   srfs-MR
```

1.25 stdConc

Standardise a chemical concentration from various aavailable unit combinations to a common concentration and adjust for moisture. Note that the input object `dat` mixes data types (`numeric` and `character`). These types are maintained in a single-row data frame but a vector will change all numerics to character. The function `stdConc` automatically forces all inputs back to their intended mode. The output is one-row data frame with fields `c(stdAmt, unit)`, which represents the standardised concentration as a numeric value and a string unit fraction.

The available units for combinations in the numerator and denominator are:

`mcg`	microgram	`mg`	milligram	`g`	gram
`kg`	kilogram	`L`	litre	`pct`	percent (%)
`pdw`	% dry weight	`ppm`	parts per million	`ppt`	parts per thousand

1.26 toUpper

Capitalise the first letter of each word in a sentence or phrase. The function uses `strsplit` and `toupper` along with a whole mess of `sapply` calls. A vector of sentences or phrases can be passed to `x`.

1.27 ttget, ttcall, ttprint, ttput, tlisp

These functions are wrappers to the **PBSmodelling** accessor functions (`tget`, `tcall`, `tprint`, `tput`, and `lisp`) that get/print/list objects from or put objects into a temporary work environment, in this case `.PBStoolEnv`. Working objects include `PBStool`, which acts as a storage object for many of the functions. These accessor functions were

developed as a response to the CRAN repository policy statement: "Packages should not modify the global environment (user's workspace)."

1.28 wrapText

Wrap a long text string to a desired width, mark it with prefixes, and indent the lines following the first. Using the package **base** function `strwrap`, this function splits a long text string into a target width, indents lines following the first, and adds a prefix string `prefix[1]` to the first line and `prefix[2]` to all indented lines.

Example (wrapText):

```
pbsfun=function(){
  txt=wrapText(paste("USA state names:  ",
    paste(state.name,collapse=", "),sep=""),width=72,exdent=5)
  showMessage(txt,as.is=TRUE,adj=0,col="blue",cex=.9,x=.05)
  cat(txt,"\n"); invisible() }
pbsfun()
```

Output (wrapText):

```
> USA state names: Alabama, Alaska, Arizona, Arkansas, California,
+      Colorado, Connecticut, Delaware, Florida, Georgia, Hawaii,
+      Idaho, Illinois, Indiana, Iowa, Kansas, Kentucky, Louisiana,
+      Maine, Maryland, Massachusetts, Michigan, Minnesota,
+      Mississippi, Missouri, Montana, Nebraska, Nevada, New Hampshire,
+      New Jersey, New Mexico, New York, North Carolina, North Dakota,
+      Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South
+      Carolina, South Dakota, Tennessee, Texas, Utah, Vermont,
+      Virginia, Washington, West Virginia, Wisconsin, Wyoming
```

1.29 zapDupes

Zap (remove) records from a data frame using an codeindex to determine whether duplicated indexed records exist. The function orders the data frame based on the `index`, then removes records based on duplicated indices. There is no control over which of the duplicated records will be zapped. Pre-existing attributes other than `names` and `row.names` are transferred to the reduced object.

Example (zapDupes):

```
pbsfun=function(){
  data(ltmose07)
  cat(paste("'ltmose07' has",nrow(ltmose07),"rows\n"))
  zap=zapDupes(ltmose07,"PID")
  print(zap,quote=FALSE)
  cat(paste("After zapping using index 'PID', there are",nrow(zap),"rows\n"))
  invisible() }
pbsfun()
```

Output (zapDupes):

```
'ltmose07' has 55 rows
  PID POS         X        Y index
1   1   1 -125.1118 49.09033     1
2   2   1 -125.9806 49.47160     2
3   3   1 -125.9806 49.47160     3
4   4   1 -126.4951 49.86156     4
5   5   1 -127.7800 49.94667     5
6   6   1 -128.2633 50.28333     6
7   7   1 -127.9032 50.36946     7
After zapping using index 'PID', there are 7 rows
```

2 Biological functions

DFO:
A suitable dataset for the biological functions can be obtained by running the SQL query `gfb_bio.sql`:

```
getData("gfb_bio.sql", "GFBioSQL", strSpp="396", path=.getSpath())
```

Substitute your species code for `strSpp`. The package contains an example dataset for Pacific ocean perch (POP, code = "396") *Sebastes alutus* in PMFC areas 5A, 5B, 5C, and 5D. To make it available in the global environment, type `data(pop.age)`.

2.1 calcLenWt

Calculate the length-weight relationship for a fish species. The functions spits results out to a comma-delimited text file (`.csv`). If the user specifies `plotit=TRUE` then image files are also generated. The code is currently set to spew `.png` files but with some code tweaking, other file types could be generated.

The parameterisation of the length-weight model is:

$$W_{s,i} = a_s(L_{s,i})^{b_s} \tag{1}$$

where

$W_{s,i}$ = the observation of weight (kg) of individual i of sex s,
$L_{s,i}$ = the observation of length (cm) of individual i of sex s,
a_s = the growth rate scalar for sex s , and
b_s = the growth rate exponent for sex s.

Example (calcLenWt):

```
pbsfun=function(){
  data(pop.age)
  calcLenWt(dat=pop.age,strSpp="396",ttype=list(c(2:3)),plotit=TRUE)
  invisible()}
pbsfun()
```

2.2 calcSG

Calculate the length or weight *vs.* age relationship by fitting the data using a versatile Schnute growth model (Schnute, 1981). Each column of the plot shows the fits for Males, Females, and M+F (Figure 8). If the data are plotted to the screen device, each row shows the fits by trip type, and each page shows the fits by area. If plots are sent to image files (`.png` or `.wmf`), each trip type and area combination goes to separate image files (handy for stacking in a `.doc` file).

The versatile Schnute fit requires initial parameter estimates and bounds. These are taken from the data object `parVec`. If the species code you are using is not contained in this file, the function uses the initial estimates for Pacific Ocean Perch (*Sebastes alutus*).

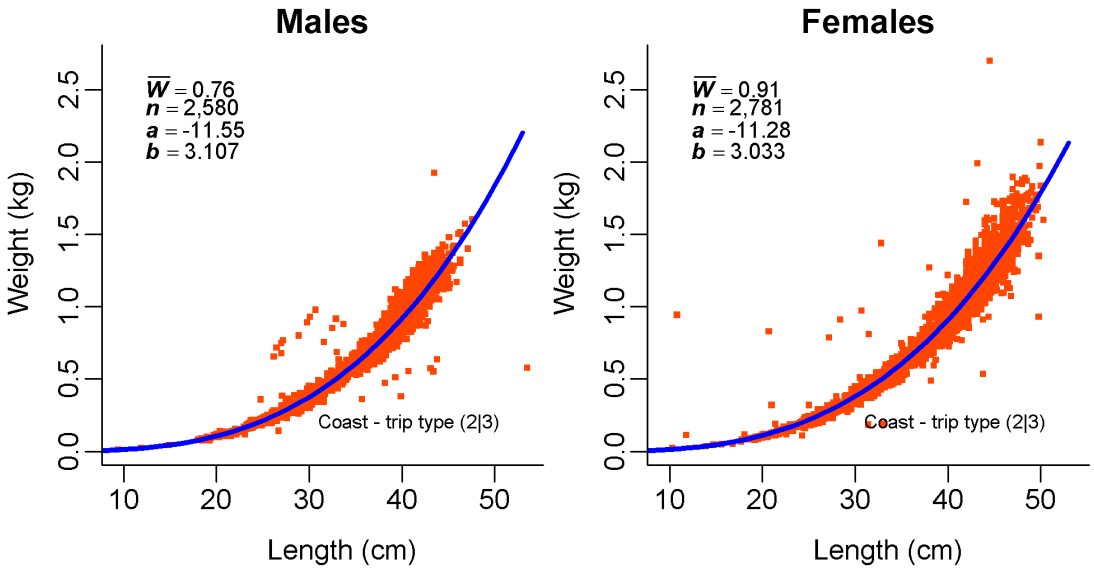

Figure 7. (*PBStools-calcLenWt*) Regression analyses showing the fitted model and survey length-weight pairs used to estimate a_s and b_s.

Example (calcSG):

```
pbsfun=function(){
  data(pop.age)
  calcSG(areas=list(srfa=list(c("5AB","5CD"))),ttype=list(2:3),tau=c(10,50),pix=TRUE)
  invisible() }
pbsfun()
```

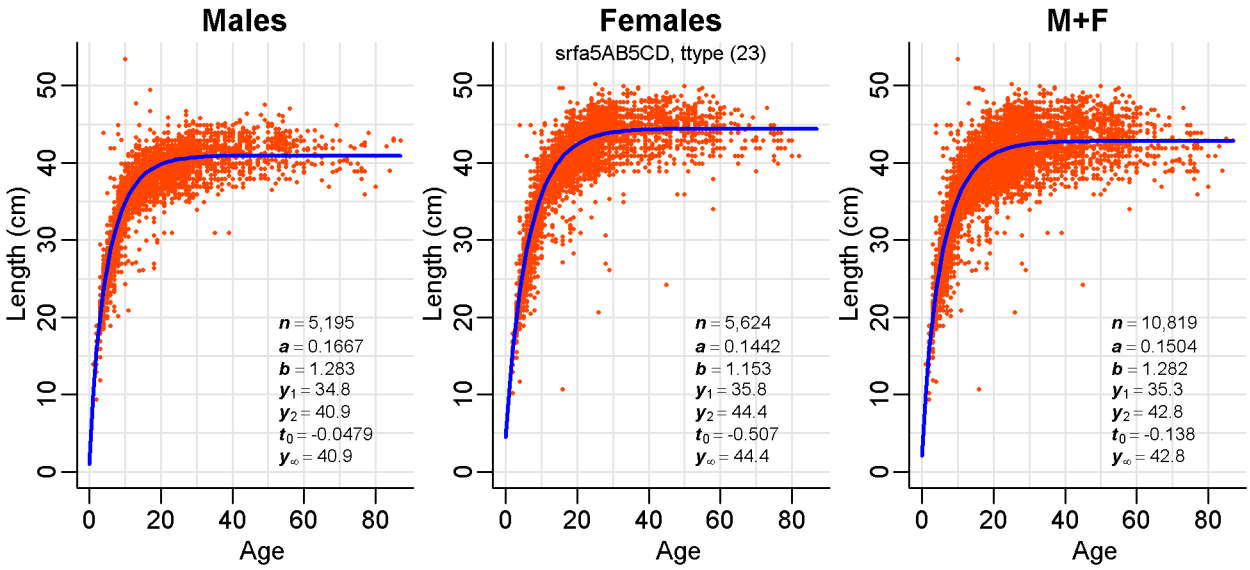

Figure 8. (*PBStools-calcSG*) Schnute versatile growth model fits to POP length and age survey data in slope rockfish assessment areas (`srfa`) 5AB and 5CD combined.

2.3 calcVB

Calculate the length (or weight) vs. age relationship by fitting these data to a von Bertalanffy growth model (Figure 9). This function provides a quick screen for combinations of management area and trip type. Figures generally comprise three columns where each column shows the model fits for Males, Females, and M+F. If the data are plotted to the screen device, each row shows the fits by trip type, and each page shows the fits by area. If plots are sent to image files `.png` or `.wmf`, each trip type and area combination goes to separate image files (handy for stacking in a `.doc` file). Note that areas can be combined if they are specified as lists within lists.

The von Bertalanffy fit requires initial parameter estimates and bounds. These are taken from the data object `parVec`. If a parameter vector is not available for a particular species, the user must either supply his own or use one available in the list (e.g., `parVec[["vonB"]][["length"]][["396"]]`).

Example (calcVB):

```
pbsfun=function(){
  data(pop.age)
  calcVB(areas=list(srfa=list(c("5AB","5CD"))),ttype=list(2:3),pix=TRUE)
  invisible()}
pbsfun()
```

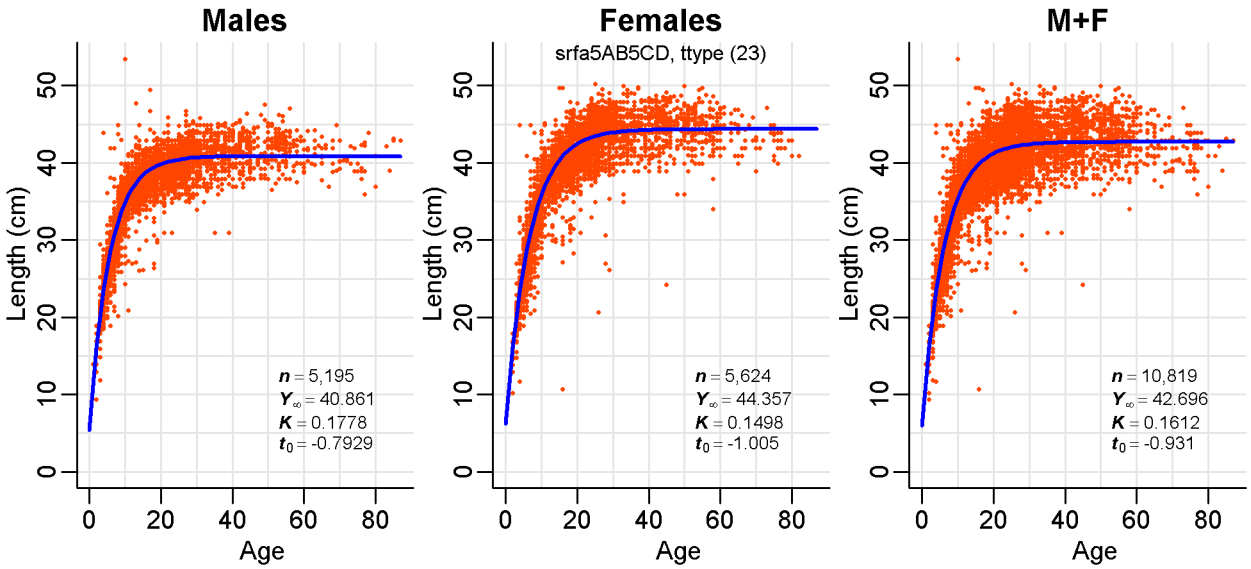

Figure 9. (*PBStools-calcVB*) von Bertalanffy growth model fits to POP length and age data in slope rockfish assessment areas (`srfa`) 5AB and 5CD combined.

2.4 compCsum

Compare cumulative relative frequency curves for specified factors such as age. Typically, factors are fields in the data object and comparisons are performed for each specified area and trip type. If the data are plotted to the screen device, each panel

contains curves by combinations of trip type (rows) and factor (columns), and each page shows these combinations by area. If plots are sent to image files `.png` or `.wmf`, either each row is sent to separate files when `singles=TRUE` (handy for stacking in a `.doc` file) or each page is sent to a separate file when `pages=TRUE`.

In special cases `yfac="area"` or `yfac="trip"`, each panel depicts curves by year, and the factors (curves) are created as new fields from inputs `areas` or `ttype`. (The option `yfac="trip"` is not implemented at present.)

Example (compCsum):

```
pbsfun=function(){
  data(pop.age)
  compCsum(pop.age, areas=list(srfa=list(c("5AB","5CD"))),
    ttype=list(c(2,3)), years=1995:2008)
  invisible()}
pbsfun()
```

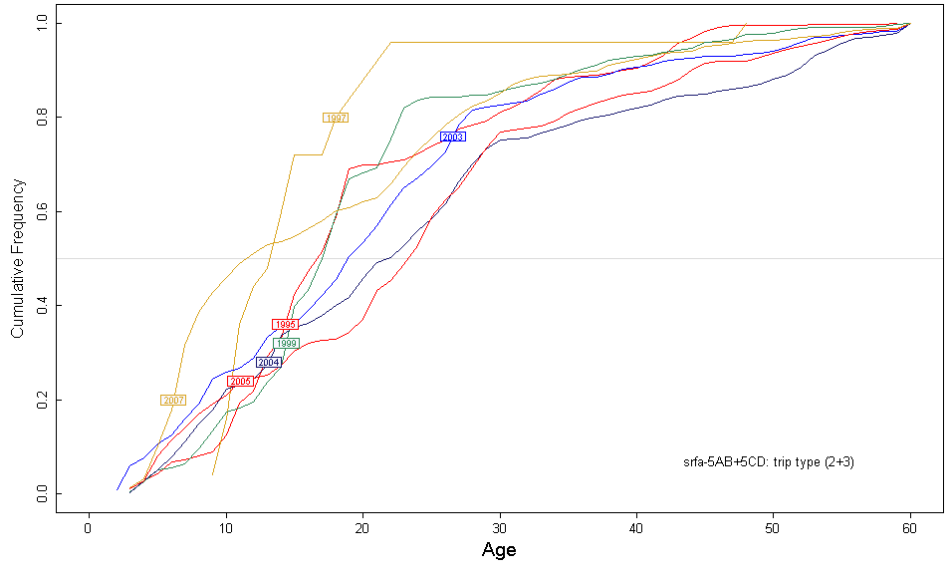

Figure 10. (*PBStools-compCsum*) Annual cumulative frequency curves for POP age proportions in slope rockfish assessment areas 5AB + 5CD using research and charter trip data.

2.5 estOgive

Estimate the cumulative distribution of the proportion of mature fish smoothed by binning ages. The cumulative distribution of mature fish is plotted and the point at 50% maturity is interpolated. The x-axis (age or length) can be binned to smooth out rough or sparse data.

Example (estOgive):

```
pbsfun=function(){
  data(pop.age)
  estOgive(obin=2, xlim=c(0,30), sex=1:2, method=c("empir","dblnorm"), cex=0.9)
  invisible()}
pbsfun()
```

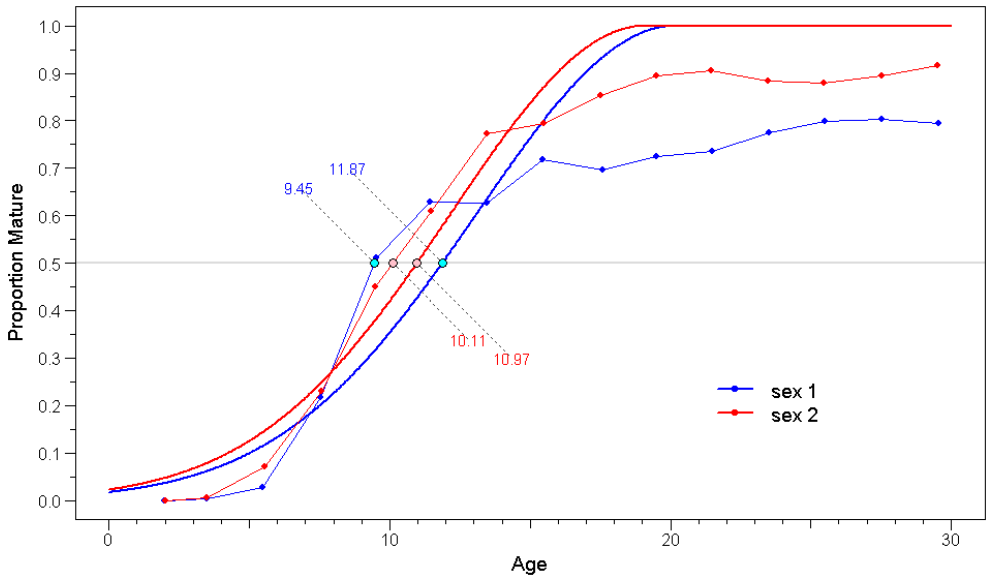

Figure 11. (*PBStools-estOgive*) Maturity ogives for POP estimated from plotting the cumulative sum of the proportion mature:total fish using binned ages. Diagonal dashed lines indicate ages at 50% maturity. Superimposed are model fits (smooth curves) to a double normal distribution.

2.6 genPa

Generate proportions-at-age using the catch curve composition of Schnute and Haigh (2007) (see Table 2, p.219 therein). Their model uses a design vector ϕ and a parameter vector θ; the latter contains the quantities to be estimated in a negative log likelihood using the complete parameter vector (θ, σ) for a logistic-normal model or (θ, n) for a Dirichlet model. Here we use θ primarily as a parameter vector, but include the design elements $b_1, ...b_m$ for expediency.

Additionally, for the selectivity component, we use the parameterisation of Edwards *et al.* (2012) where μ represents the age of full selectivity and $\sigma = \sqrt{\nu}$ represents the variance of the left limb of a double normal (see equation F.7 in the Pacific Ocean Perch assessment). This parameretisation replaces (T2.2) in Schnute and Haigh (2007), which uses α and β_k.

The output is a proportions-at-age vector of length `np`.
If `sim=TRUE`, the components S_a, β_a, R_a, and p_a are writtten to the global list object `PBStool`.

Example (genPa):

```
pbsfun=function(){
  genPa(); unpackList(PBStool); unpackList(theta)
  par0 = par(no.readonly=TRUE); on.exit(par(par0))
  expandGraph(mfrow=c(2,2),mar=c(1,4,1,0.5),oma=c(3,0,0,0),mgp=c(2,.5,0))
  for (i in c("Sa","Ba","Ra","pa")) {
    expr = paste(
      c("plot(1:np,",i,",type=\"l\",lwd=2,col=\"blue\",xlab=\"\",ylab=\"\");",
      "mtext(expression(italic(",substring(i,1,1),
      "[a])),side=2,line=2,cex=1.5,las=1);",
      "points(1:np,",i,",pch=21,bg=\"aliceblue\",cex=1.2)"),collapse="")
    eval(parse(text=expr))
    if (i=="Ba") lines(rep(mu,2),c(0.99,par()$usr[3]),lty=2,col="grey30")
    if (i=="Ra") for (j in 1:length(rho))
      lines(rep(bh[j],2),c(rho[j]+0.98,par()$usr[3]),lty=2,col="grey30")
    if (i=="pa") lines(1:np,scaleVec(Sa,pa[np],max(pa)),lty=2,col="grey30") }
  mtext("     Age",outer=TRUE,side=1,line=1.5,cex=1.5)
  invisible() }
pbsfun()
```

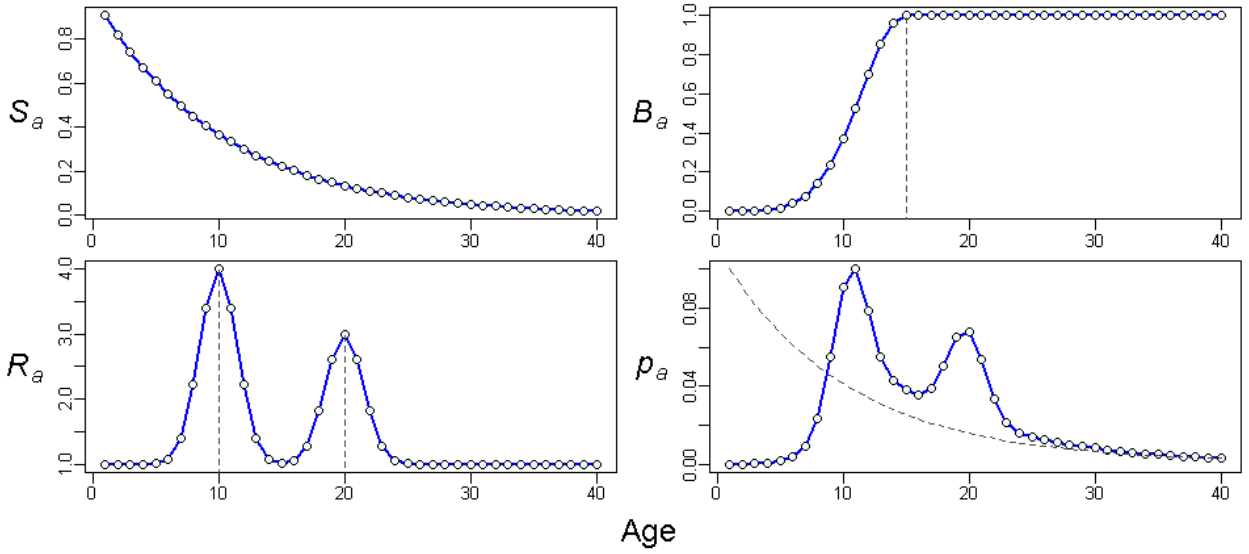

Figure 12. (*PBStools-genPa*) Plots of survival S_a, selectivity β_a, recruitment R_a, and predicted proportions p_a as functions of age a, based on the catch curve model of Schnute and Haigh (2007).

2.7 histMetric

Plot a matrix of histograms or cumulative frequencies for a specified biology metric by year and trip type. The function displays the relative frequency of a specified metric like length by year (row) and trip type (column). Cells where the number of specimens `<= minN` are left blank. If only one trip type is specified or available, the cells display the annual histograms by row with an automatically calculated number of columns.

Example (histMetric):

```
pbsfun=function(){
  data(pop.age)
  histMetric(year=2003:2007, xlim=c(0,60), fill=TRUE, bg="moccasin", xint=2)
  invisible()}
pbsfun()
```

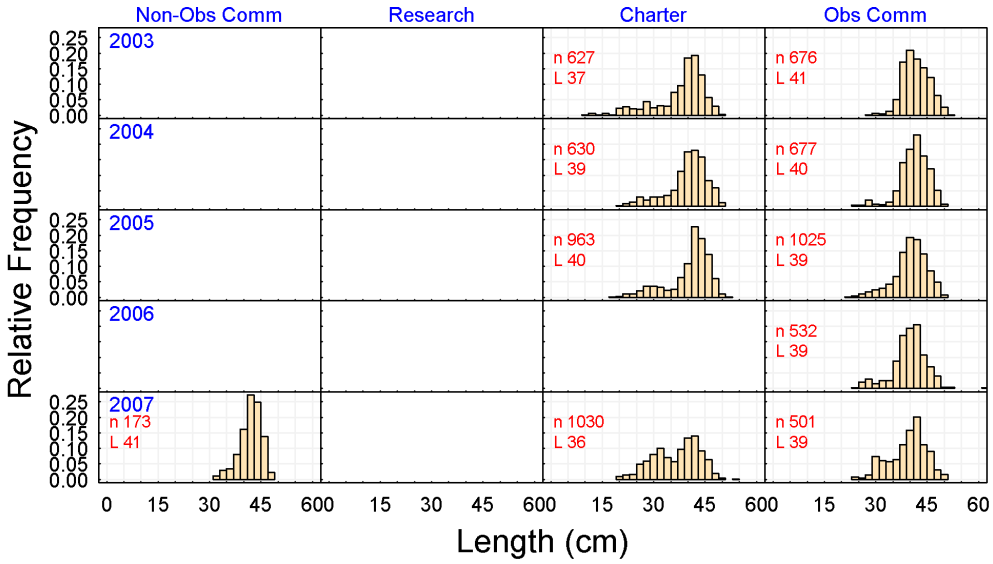

Figure 13. (*PBStools-histMetric*) Histograms of POP length samples from 2003 to 2007 for the four trip types. Values of n (number of specimens) and L (mean length in cm) appear in red.

2.8 histTail

Plot a histogram for a specified biology metric with an option to zoom in on the tail detail. The function displays the relative frequency of a specified metric like age. Its functionality is minimal, providing a quick histogram for a specified field in a data frame. The tail histogram appears as an inset on the right showing a zoom-in of the frequency of the upper tail data. The user can specify whether to plot the relative probability distribution (`prob=TRUE`) or a frequency distribution (`prob=FALSE`). The default provides the former for the main histogram and the latter for the tail zoom-in.

Example (histTail):

```
pbsfun=function(){
  data(pop.age)
  histTail(xfld="age", tailmin=60, prob=FALSE)
  invisible()}
pbsfun()
```

2.9 mapMaturity

Map each maturity code to show its relative occurrence by month. For each maturity code the function plots a row of monthly boxes colour-coded by the relative occurrence of the code in that month compared to the year.

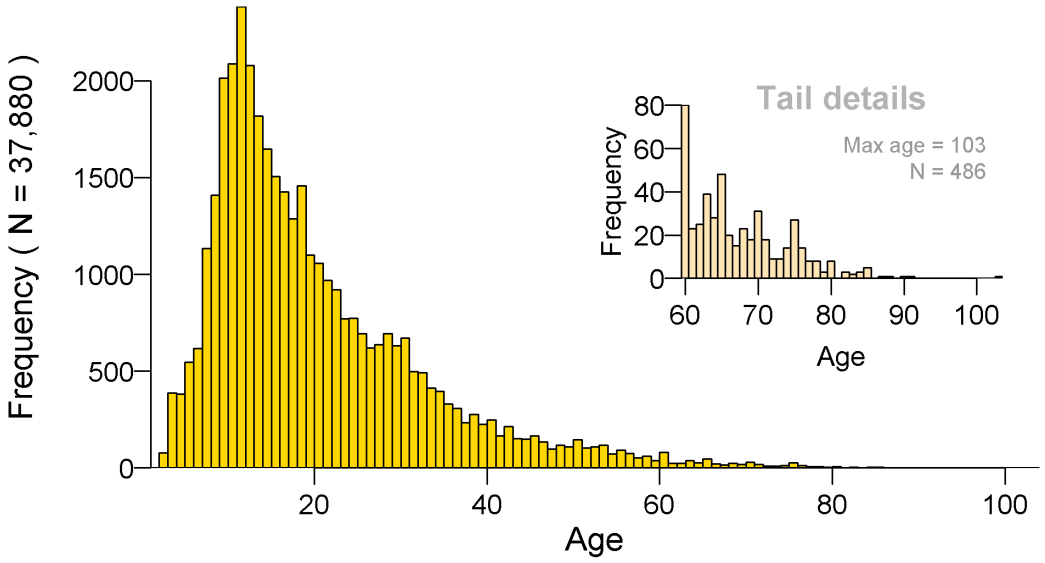

Figure 14. (*PBStools-histTail*) Histogram of POP ages with an inset that zooms in on the tail region of the distribution.

Example (mapMaturity):

```
pbsfun=function(){
  data(pop.age)
  mapMaturity(brks=c(0,.01,.05,.25,.5,1), sex=2, hpage=3.5)
  invisible()}
pbsfun()
```

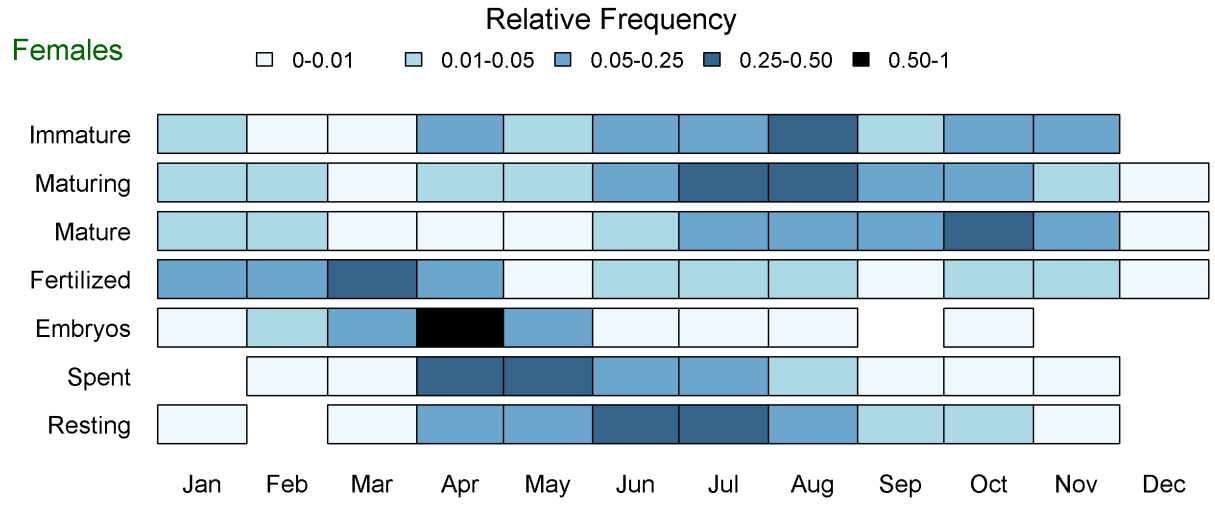

Figure 15. (*PBStools-mapMaturity*) Relative monthly frequency of observed maturity codes for female POP.

2.10 plotProp

Display proportions-at-age (or some other metric such as length or weight) vs. year using the **PBSmodelling** function `plotBubbles` controlled by an interactive GUI invoked

by `plotProp()` (Figure 16). Data inputs appear on the left. The file name can be selected from the user's global environment `ls(1)` or from a list of binary files (`rda`) in the user's current working directory. If no field is specified for stratification the value is set to `year`, effectively meaning that the annual data are not stratified. If the check box is activated, the stratifying field can also serve to apply relative weightings to the observed proportions. Note that for most stratifying variables, weighting makes no sense. A field that would be meaningful is the `catch` of the species.

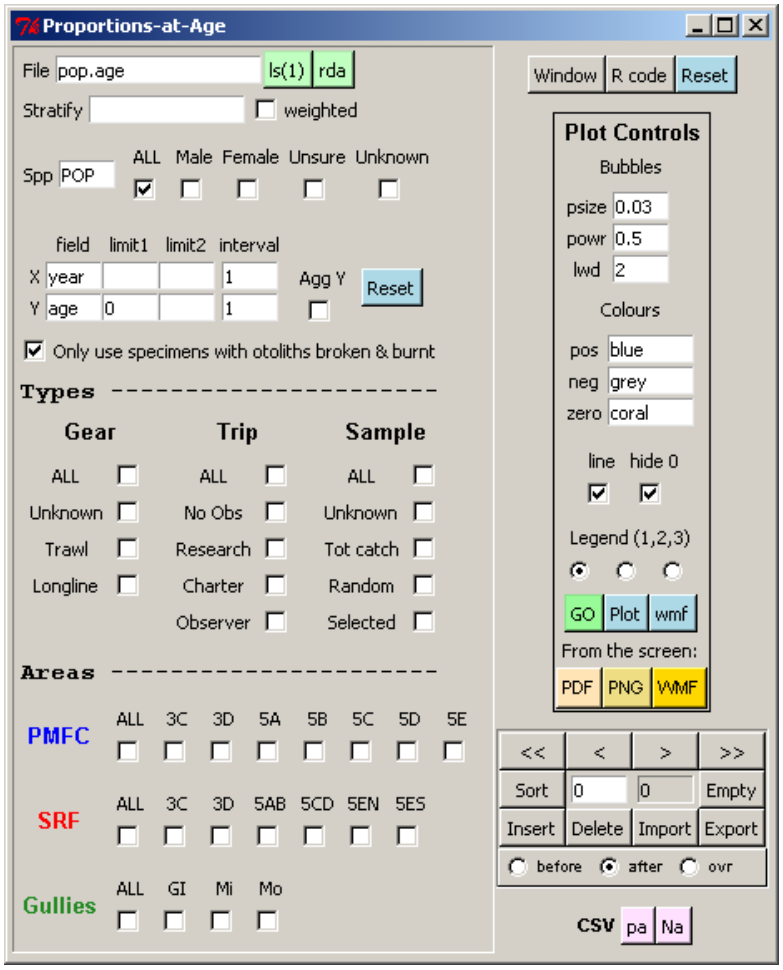

Figure 16. (*GUI-plotProp*) Proportions-at-age GUI to control the plotting of annual proportions as bubbles.

After choosing a species code, the user can select any combination of the four values of sex that might be available: male (`sex=1`), female (`sex=2`), observed but not sure (`sex=3`), and not observed (`sex=0`). The limits table of the two variables (usually year on the x-axis and age on the y-axis) can be left blank initially as the plotting routine will determine these automatically. Thereafter, the user will likely manipulate these to find a suitable plot. The Agg Y check box, if checked, will aggregate y-values at both ends of the specified y-limits. If the user wishes to aggregate only at the upper end, choose 0 or 1 for the lower limit. Most of the time, users will want to use only ages that have been determined from broken and burnt otoliths (`ameth=3`).

The GUI provides a menu of choices for certain fields in the data object: gear type (`gear`), trip type (`ttype`), sample type (`stype`), Pacific Marine Fisheries Commission areas (`major`), slope rockfish assessment areas (srfa), and slope rockfish gullies (`srfs`).

The right-hand side of the GUI provides some tweaks for output appearance and file format. The proportion bubbles can be manipulated through psize (maximum circle radius), powr (power transformation of proportions), and lwd (line thickness of bubbles). The bubble colours can be chosen separately for positive, negative, and zero values (even though proportions cannot be negative). Horizontal guide lines and zero-value proportions can be turned on or off.

The user has a choice of legend that accompanies the plot. The first choice is the most comprehensive in that it details how many records occur in various categories for seven of the important fields (along the bottom of the plot) as well the number of specimens used in each year (along the top). The second and third choices of legend simply display the number of specimens and samples per, respectively, along the bottom.

The action buttons GO and Plot both plot the results from the GUI choices (Figure 17); however, the former recalculates (qualifies, stratifies, weights, etc.) the input data object whereas the latter simply plots the massaged data object. The differences are supplied for users who may wish to change graphical parameters (e.g., bubble size) after the data have been qualified. This distinction must be noted as a user may change field selections like gear type or area and press the Plot button expecting the data object to reflect these changes. It is usually safest to use the GO button most of the time.

The blue wmf button redraws the plot on a metafile device directly instead of the graphical window. For dumping images from the screen to various file formats, push one of the yellow buttons marked PDF, PNG, or WMF. Below the history widget, there are two pink buttons pa and Na for dumping the proportions-at-age (or length or weight) and numbers-at-age, respectively, to `.csv` text files.

2.11 predictRER

A discriminant function for the accurate classification of Rougheye Rockfish (RER, *Sebastes aleutianus*) and Blackspotted Rockfish (BSR, *Sebastes melanostictus*). Exploring 35 morphometric and 9 meristic characters, Orr and Hawkins (2008) provide a discriminant function (using only 6 morphometrics L and 2 meristics N) that claims to correctly classify the two species 97.8% of the time.

The discriminant function:

$$\begin{aligned} D = {} & 101.557\lambda_1 + 52.453\lambda_2 + 51.920\lambda_3 - 38.604\lambda_4 - 22.601\lambda_5 - 10.203\lambda_6 \\ & + 0.294N_1 + 0.564N_2 - 10.445 \end{aligned} \tag{2}$$

where, $\lambda_n = 100\dfrac{L_n}{S}$, i.e., percent Standard Length

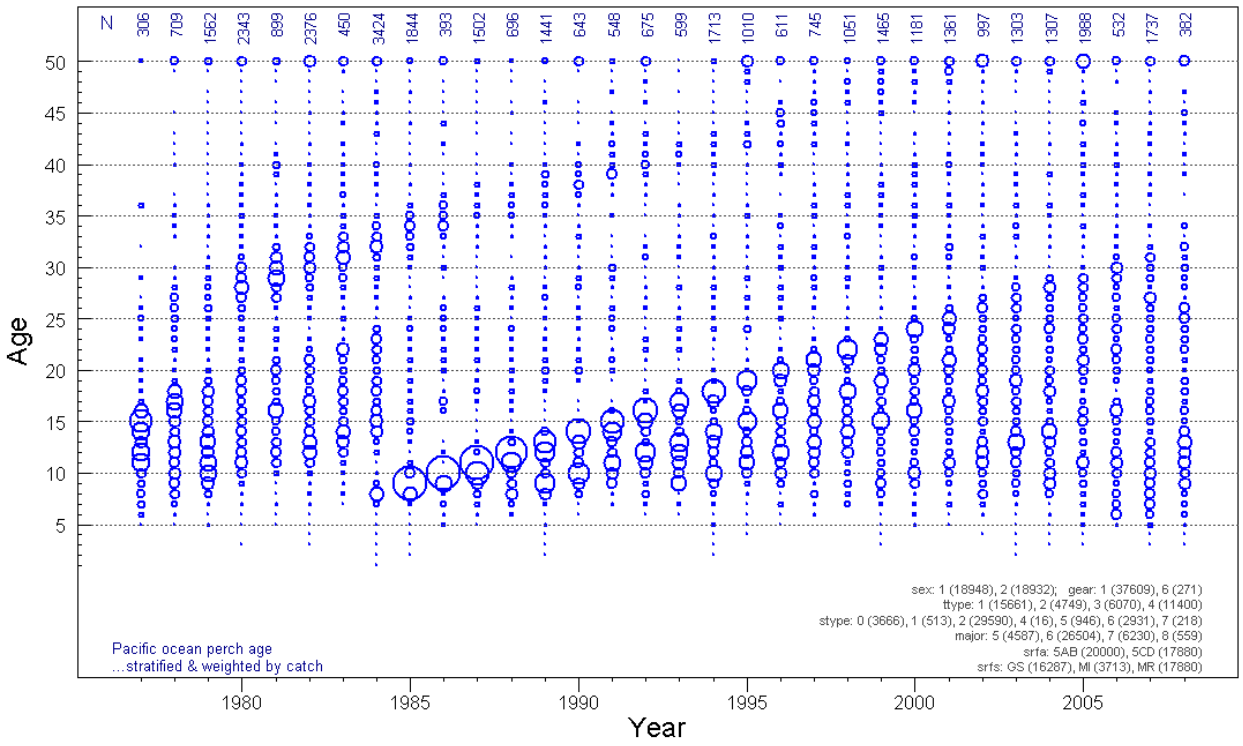

Figure 17. (*PBStools-plotProp*) Proportions-at-age for POP rendered as proportional bubbles.

S = standard fish length measured from tip of snout,
L_1 = length of dorsal-fin spine 1,
L_2 = snout length,
L_3 = length of gill rakers,
L_4 = length of pelvic-fin rays,
L_5 = length of soft-dorsal-fin base,
L_6 = preanal length,
N_1 = number of gill rakers,
N_2 = number of dorsal-fin rays.

Output is a numeric scalar D representing the calculated discriminant value.
When $D < 0$, the prediction is that the rockfish is *S. aleutianus*.
When $D > 0$, the prediction is that the rockfish is *S. melanostictus*.

Example (predictRER):

```
pbsfun = function(){
  test = simRER(1000)
  Dval = apply(test,1,function(x){predictRER(x[1],x[2:7],x[8:9])})
  cat(paste("simRER(1000)\nRER correctly predicted ",
    round((length(Dval[Dval<0])/length(Dval)*100),1),"% of the time\n\n",sep=""))
  test = simBSR(1000)
  Dval = apply(test,1,function(x){predictRER(x[1],x[2:7],x[8:9])})
  cat(paste("simBSR(1000)\nBSR correctly predicted ",
    round((length(Dval[Dval>0])/length(Dval)*100),1),"% of the time\n",sep=""))
  invisible() }
pbsfun()
```

Output (predictRER):

```
simRER(1000)
RER correctly predicted 98% of the time

simBSR(1000)
BSR correctly predicted 94.1% of the time
```

2.12 processBio

Process further the global data object `PBSdat` that results from the SQL query `gfb_bio.sql`. Currently, this function can add fields `srfa` (slope rockfish assessment areas), `srfs` (slope rockfish assessment subareas or gulllies), and `popa` (Pacific ocean perch areas identified by Schnute *et al.* (2001). These three areas do not exist as fields in GFBioSQL.

Example (processBio):

```
getData("gfb_bio.sql", "GFBioSQL", strSpp="396", path=.getSpath())
pop.bio = processBio(PBSdat)
```

2.13 reportCatchAge

This function is very specific in that it takes as its input the report file generated by Jon Schnute's ADMB template file (`.tpl`) for a Pacific ocean perch catch-at-age model. The Report Section of the template file (displayed below) saves a list object as a text file (`.rep`) using PBS formatting. Once the function reads this list object into R, a GUI (Figure 18) provides the user with various plotting options to view the results of the analysis.

```
REPORT_SECTION
Calc_SDpars();
report << "# POP Catch-at-Age Results" << endl;
report << "$nyr\n"     << "$$vector mode=\"numeric\" \n"    << NYear   << endl;
report << "$nag1\n"    << "$$vector mode=\"numeric\" \n"    << NAge1   << endl;
report << "$nage\n"    << "$$vector mode=\"numeric\" \n"    << NAge    << endl;
report << "$k\n"       << "$$vector mode=\"numeric\" \n"    << k       << endl;
report << "$pmin\n"    << "$$vector mode=\"numeric\" \n"    << PMin    << endl;
report << "$objfn\n"   << "$$vector mode=\"numeric\" \n"    << lf      << endl;
report << "$pnlty\n"   << "$$vector mode=\"numeric\" \n"    << penalty << endl;
report << "$fmor\n"    << "$$vector mode=\"numeric\" \n"    << F       << endl;
report << "$mmor\n"    << "$$vector mode=\"numeric\" \n"    << M       << endl;
report << "$rmean\n"   << "$$vector mode=\"numeric\" \n"    << R       << endl;
report << "$gma\n"     << "$$vector mode=\"numeric\" \n"    << gamma   << endl;
report << "$al\n"      << "$$vector mode=\"numeric\" \n"    << alpha   << endl;
report << "$bta\n"     << "$$vector mode=\"numeric\" \n"    << beta_a  << endl;
report << "$qq\n"      << "$$vector mode=\"numeric\" \n"    << q       << endl;
report << "$BPars\n"   << "$$vector mode=\"numeric\" \n"    << BPars   << endl;
report << "$rho\n"     << "$$vector mode=\"numeric\" \n"    << rho     << endl;
report << "$kap2\n"    << "$$vector mode=\"numeric\" \n"    << kap_2   << endl;
report << "$sig12\n"   << "$$vector mode=\"numeric\" \n"    << sig1_2  << endl;
report << "$tau12\n"   << "$$vector mode=\"numeric\" \n"    << tau1_2  << endl;
```

```
report << "$tau22\n"   << "$$vector mode=\"numeric\" \n"    << tau2_2  << endl;
report << "$Bt\n"      << "$$vector mode=\"numeric\" \n"    << B_t     << endl;
report << "$SSB\n"     << "$$vector mode=\"numeric\" \n"    << SSB     << endl;
report << "$RecProd\n" << "$$vector mode=\"numeric\" \n"    << RecProd << endl;
report << "$mcvec0\n"  << "$$vector mode=\"numeric\" \n"    << mcvec   << endl;
report << "$cdata\n"   << "$$data ncol=6 modes=\"integer numeric numeric numeric numeric numeric\"
         colnames=\"yr catch cf B1 B2 B3\" byrow=TRUE\n" << data_cpue  << endl;
report << "$uat\n"     << "$$matrix mode=\"numeric\" ncol=" << NYear << "\n" << u_at    << endl;
report << "$Nat\n"     << "$$matrix mode=\"numeric\" ncol=" << NYear << "\n" << Nat     << endl;
report << "$wgtat\n"   << "$$matrix mode=\"numeric\" ncol=" << NYear << "\n" << w_at    << endl;
report << "$pat\n"     << "$$matrix mode=\"numeric\" ncol=" << NYear << "\n" << p_at    << endl;
report << "$wgta\n"    << "$$vector mode=\"numeric\" \n"    << wgta    << endl;
report << "$mata\n"    << "$$vector mode=\"numeric\" \n"    << mata    << endl;
```

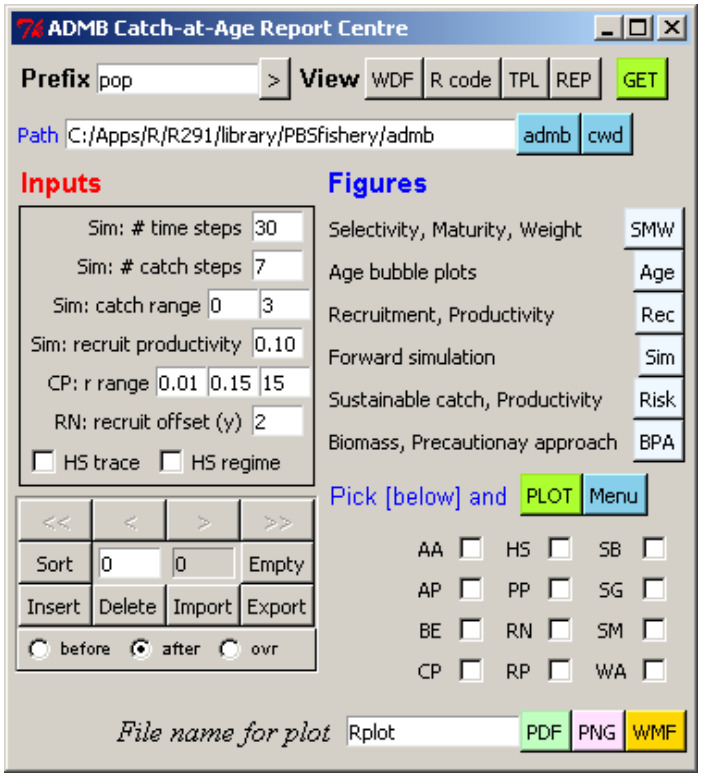

Figure 18. (*GUI-reportCatchAge*) ADMB Catch-at-Age Report Centre GUI to control the plotting of POP model results contained in the report file generated by code in the template file.

ADMB projects are controlled by a prefix (e.g., `pop`) that ADMB uses to accept input (`pop.tpl`, `pop.dat`, `pop.pin`) and spawn copious output (`pop.exe`, `pop.cpp`, `pop.o`, `pop.std`, `pop.rep`, `...`). The R package **PBSadmb** (available on CRAN) provides an easy to use GUI system for running ADMB directly from R.

Aside from reading in the report file (e.g., `pop.rep`), the GUI offers a few user-controlled tweaks (see `reportCatchAge` help file for details) that affect the simulation and recruitment plots. The light blue buttons on the right recreate six pairs of plots similar to those that appear in the 2001 POP assessment (Schnute *et al.*, 2001). Alternatively, the user can pick and choose combinations of plots from the menu check boxes (lower right),

then press the green PLOT button. The plot codes appear in the help file; however, we reproduce them below for convenience. Figure 19 results from choosing the BE check box and plotting.

Table 1. Menu of figure codes to select and PLOT.

Code	Plot type from report file
AA	Ages actual
AP	Ages predicted
BE	Biomass estimates
CP	Catch (sustainable) vs. productivity
HS	Harvest strategy (precautionary approach)
PP	Probability of achieving productivity
RN	Recruitment numbers
RP	Recruitment productivity
SB	Simulation: biomass at fixed catches
SG	Simulation: growth rate vs. catch
SM	Selectivity and maturity vs. age
WA	Weight vs. age

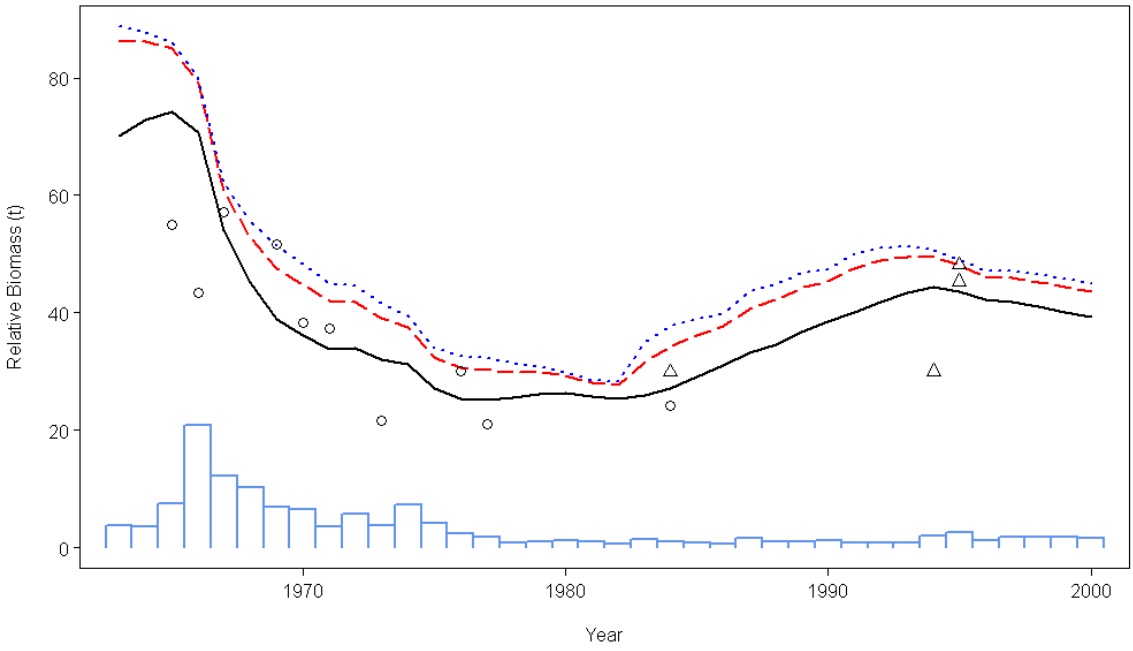

Figure 19. (*PBStools-reportCatchAge*) Annual estimates of POP biomass: available to the fishery (solid back line), mature (dashed red line), and total (dotted blue line) for the Goose Island Gully population in Queen Charlotte Sound. Circles indicate research survey estimates, scaled to available biomass by q_1. Plus symbols indicate charter survey estimates, scaled with q_2. The blue bar plot represents annual catches.

2.14 requestAges

This function queries the DFO database GFBioSQL for available otoliths to age. It chooses random otoliths based on trips' commercial catch weights of the specified species (e.g., "396" for POP). The function invisibly returns the otolith data with supplementary quarterly commercial catch. Also, it writes this otolith data to binary `.rda` and ascii `.csv` files. The random samples formatted to simulate an otolith tray (for ageing personnel) is written to a file called `oto123-C(YYYY)-areas(0+0+0)-sex(1+2)-N999.csv` (where `123` = species code, `C` = Commercial or `S` = Survey, `YYYY` = year, `0` = PMFC area code, and `999` = number of otoliths).

2.15 simBSR, simRER

Simulate biological data that best characterizes Blackspotted Rockfish (BSR, *Sebastes melanostictus*) and Rougheye Rockfish (RER, *S. aleutianus*), using parameter estimates from Table 2 in Orr and Hawkins (2008). This table provides a range of BSR standard lengths (mm) and distributions for morphometrics and meristics. The functions `simBSR` and `simRER` sample from a random uniform distribution for S and from random normal distributions for model inputs of λ and N – tables below give (μ, σ).

Balckspotted Rockfish:

S	=	standard fish length measured from tip of snout	95.5–539
λ_1	=	length of dorsal-fin spine 1	(7.8, 0.7)
λ_2	=	snout length	(8.0, 0.6)
λ_3	=	length of gill rakers	(5.6, 0.6)
λ_4	=	length of pelvic-fin rays	(21.4, 1.2)
λ_5	=	length of soft-dorsal-fin base	(21.4, 1.5)
λ_6	=	preanal length	(70.2, 2.6)
N_1	=	number of gill rakers	(33.0, 1.2)
N_2	=	number of dorsal-fin rays	(13.7, 0.5)

Rougheye Rockfish:

S	=	standard fish length measured from tip of snout	63.4–555.2
λ_1	=	length of dorsal-fin spine 1	(5.8, 0.6)
λ_2	=	snout length	(7.5, 0.7)
λ_3	=	length of gill rakers	(4.9, 0.6)
λ_4	=	length of pelvic-fin rays	(22.1, 1.1)
λ_5	=	length of soft-dorsal-fin base	(22.8, 1.2)
λ_6	=	preanal length	(71.8, 2.3)
N_1	=	number of gill rakers	(31.2, 1.0)
N_2	=	number of dorsal-fin rays	(13.7, 0.5)

where, $\lambda_n = 100\frac{L_n}{S}$, i.e., percent Standard Length

The function returns a numeric matrix with dimensions `c(Nfish,9)` where columns are labelled `c('S', 'L1', 'L2', 'L3', 'L4', 'L5', 'L6', 'N1', 'N2')`. The values are described above, but generally, `S` = standard length of the fish (mm), `L` = six diagnostic

length measurements (mm), and `N` = numbers of gill rakers and dorsal fin rays.

2.16 sumBioTabs

This function provides a simple utility to summarise the number of samples and specimens by factor pairs in a biological data object. Aside from the factor strings chosen, the object must contain the field `SID` (sample ID) if a sample summary is indicated. The latter also require the package **reshape**. The summary is sent to a `CSV` file.

2.17 weightBio

This function executes a method for representing commercial age structure for a given species through weighting age frequencies n_a or proportions n'_a by catch. For simplicity, we assume that age frequencies n_a are weighted, unless otherwise specified. The weighting occurs at two levels: quarterly q and annually y. Notation is summarized in Table 2.

Table 2. Notation for weighted commercial age equations for a given species.

Symbol	Description
	Indices
a	age class (1 to 60, where 60 is an accumulator age-class)
t	trips as sample units (may contain more than one sample)
q	quarters (1 to 4, defined mechanically to occur every 91.5 days) within years
y	years (1977 to present)
	Data
n_{atqy}	frequency at age a for trip t in quarter q of year y
S_{tqy}	trip catch of a given species for trip t in quarter q of year y
S'_{tqy}	S_{tqy} as a proportion of total catch $S_{qy} = \sum_t S_{tqy}$
m_{aqy}	weighted age frequencies at age a in quarter q of year y
C_{qy}	commercial catch in quarter q of year y
C'_{qy}	C_{qy} as a proportion of total catch $C_y = \sum_q C_{qy}$
w_{aq}	weighted age frequencies at age a in year y
p_{aq}	weighted proportions at age a in year y

For each quarter q we weight trip frequencies n_{at} by trip catch of a given species. (We use trip as the sample unit, though sometimes one trip can contain multiple samples. This means we sometimes merge samples into the sample unit.) Within any quarter q and year y there is a set of trip catches S_{tay} that can be transformed into a set of catch proportions:

$$S'_{tay} = \frac{S_{tay}}{\sum_t S_{tay}} \ . \tag{3}$$

The age frequencies are weighted using S'_{tay} to derive weighted age frequencies by quarter:

$$m_{aqy} = \sum_t n_{atqy} S'_{tay} \ . \tag{4}$$

If this transformation is applied to frequencies (as opposed to proportions), it reduces them from the original, and so we rescale m_{aqy}:

$$m_{aqy} = m_{aqy} \frac{\sum_a n_{aqy}}{\sum_a m_{aqy}} \ . \tag{5}$$

to retain the original number of observations. This step is not strictly necessary because at the end of the two-step weighting, we standardize the weighted frequencies to represent proportions-at-age.

At the second level of stratification by year y, we calculate the annual proportion of quarterly commercial

$$C'_{qy} = \frac{C_{qy}}{\sum_q C_{qy}} \tag{6}$$

to weight m_{aqy} and derive weighted age frequencies by year:

$$w_{ay} = \sum_q m_{aqy} C'_{qy} \ . \tag{7}$$

Again, if this transformation is applied to frequencies (as opposed to proportions), it reduces them from the original, and so we rescale w_{ay}:

$$w_{ay} = w_{ay} \frac{\sum_a m_{ay}}{\sum_a w_{ay}} \ . \tag{8}$$

to retain the original number of observations.

Finally, we standardize the weighted frequencies to represent proportions-at-age:

$$p_{ay} = \frac{w_{ay}}{\sum_a w_{ay}} \ . \tag{9}$$

If initially we had used proportions n'_{atqy} instead of frequencies n_{atqy} , the final standardization would not be necessary; however, its application does not affect the outcome.

It sometimes matters if we choose to weight frequencies rather than proportions: the numeric outcome can be very different, especially if the input samples comprise few observations. Theoretically, weighting frequencies emphasizes our belief in individual observations at specific ages while weighting proportions emphasizes our belief in sampled age distributions. Neither method yields inherently better results; however, if your original sampling methodology favours sampling few fish from many tows rather than sampling many fish from few tows, then weighting frequencies probably makes more sense than weighting proportions.

3 Fishery functions

Most of the fishery functions use commercial catch and effort data. Some functions tap into the DFO databases (SQL server and Oracle) directly, and therefore cannot be used by those not logged onto the DFO network. Other functions can use local data files. An example dataset, `testdatC` included in the **PBSdata** package, provides a random subset of commercial observer trawl data in DFO's PacHarvest database.

3.1 calcRatio

Calculate the annual ratios of one catch to another (or one field to another) for various PMFC major areas and coastwide. A suitable dataset can be obtained by DFO personnel by running the SQL query `pht_catORF.sql`, where ORF refers to rockfish other than POP, for a specified species code. The query returns a data frame with fields of `catch` and `discard` for the target, as well as `POP` and `ORF` catch. Total rockfish catch (`TRF`) can be calculated by summing `POP` and `ORF`. The mean ratios are invisibly returned in a matrix where rows are fishing years and columns are PMFC area codes. Figure 13 provides visual results of yellowmouth rockfish `catch` to `ORF` (all rockfish catch other than POP). Discard rates could be represented by setting the ratio to `discard` over `catch`.

Example (calcRatio):

```
pbsfun=function(dfo=FALSE){
  if (dfo) {
    getData("pht_tcatORF.sql",strSpp="440",path=.getSpath())
    ymr.pop.orf = PBSdat
    pmean=calcRatio(dat=ymr.pop.orf, nfld="landed", dfld="ORF",
      major=3:9, plot=TRUE, ylim=c(0,.35))
    print(pmean) }
    else cat("If you are logged onto the DFO network,\nrun 'pbsfun(dfo=TRUE)'\n.")
  invisible() }
pbsfun(TRUE)
```

3.2 dumpMod, dumpRat

Dump catch from modern databases that has been collected and organised during a catch reconstruction (see `buildCatch`) into CSV tables. The function `dumpCat` flattens the input array `dat` from four or five dimensions to two dimensions and sends the result to a comma-delimited text file (`.csv`).

Dump catch ratios calculated during a catch reconstruction into CSV tables. The reconstruction procedure stores ratio arrays in a list object `PBStool123`, where `123` is the species Hart code. The function `dumpRat` extracts the arrays and sends them to a comma-delimited text file (`.csv`).

3.3 formatCatch

Format a table of numeric values so that each cell displays N significant figures in character format. The algorithm relies on string manipulation to achieve a result that cannot be obtained by the base package's `format` function alone. The formatted table can

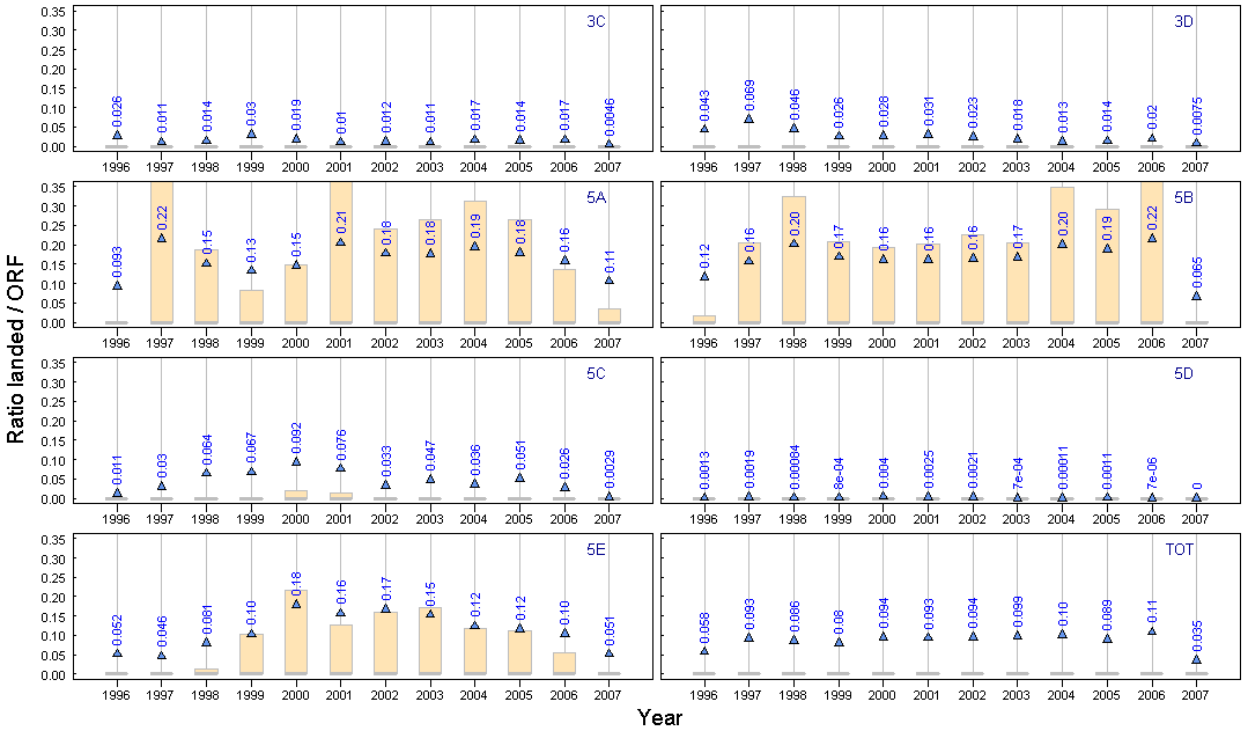

Figure 20. (*PBStools-calcRatio*) Distribution of proportions of yellowmouth rockfish landings to landings of all rockfish other than Pacific ocean perch in the trawl fishery by fishing year (Apr to Mar). Vertical grey bars indicate the extent of the data. Horizontal grey bars within the boxplots indicate median ratios (all zero here). Blue triangles indicate mean ratios (also presented as numbers). Panels summarize ratios by PMFC areas; the final panel summarizes ratios on a coastwide basis.

then be passed to **xtable**'s function `xtable`, which produces output that can be passed to `print.xtable` for use in LaTeX.

3.4 getCatch

Extract catch records for a species from various databases and combine them into one catch file. This function provides a quick method of extracting contiguous datasets and combining them into one data object. The primary use for this dataset is to create the catch input for the function `weightBio`. Currently, `getCatch` only functions for the trawl fishery as the hook and line databases are messy and not contiguous. For a comprehensive catch reconstruction, see `buildCatch`.

The function returns a data object of survey catch records `Scat123.wB` and commercial catch records `Ccat123.wB` resulting from the merge of various database query results. The results are also written to system binary files `Scat123.wB.rda`, `Ccat123.wB.rda`, where `123` specifies the species code and `wB` specifies creation for use in the function `weightBio`.

3.5 glimmer

This function performs a standardised GLM analysis and plots the results. It calculates parameter coefficients using the **stats** package functions `lm` and `aov`. (The `aov` results are not currently used for anything.) See Section 7 of Schnute *et al.* (2004b) for a detailed discussion on 'Standardized CPUE Analyses' and the additive lognormal model:

$$\log y_{hijklmn} = \mu_h + \alpha_i + \beta_j + \gamma_k + \delta_l + \phi_m + \sigma\epsilon_{hijklmn} \tag{10}$$

The example below uses the example dataset `testdatC` (from **PBSdata**), which provides a random subset of commercial trawl data. The GLM model (Figure 21) suggests an increasing trend in CPUE for these particular data, though index values appear to be imprecise. Factor effects are fairly typical for a shelf/slope rockfish species, especially the dome-shaped curve for depth. We don't anticipate a large vessel effect in the subset because the vessel codes assigned to each record have not only been masked using simple integers, but have also been randomly re-assigned (without replacement) to the tow set. In real life, certain vessels are considerably better at capturing fish than others and can produce a strong factor effect. The latitudinal effect highlights the predominance of POP in the three gullies (Goose Island, Mitchell's, Moresby) of Queen Charlotte Sound.

Example (glimmer):

```
pbsfun=function(){
  data(testdatC);  x=testdatC
  glmdat=data.frame(
    year=as.numeric(substring(x$date,1,4)),
    month=as.numeric(substring(x$date,6,7)),
    depth=x$fdep,latitude=x$Y,vessel=x$cfv,
    effort=x$eff,catch=x$POP)
  attr(glmdat,"spp")="396"
  glimmer(glmdat,Vpro=.025,dlim=c(100,500,50))
  invisible() }
pbsfun()
```

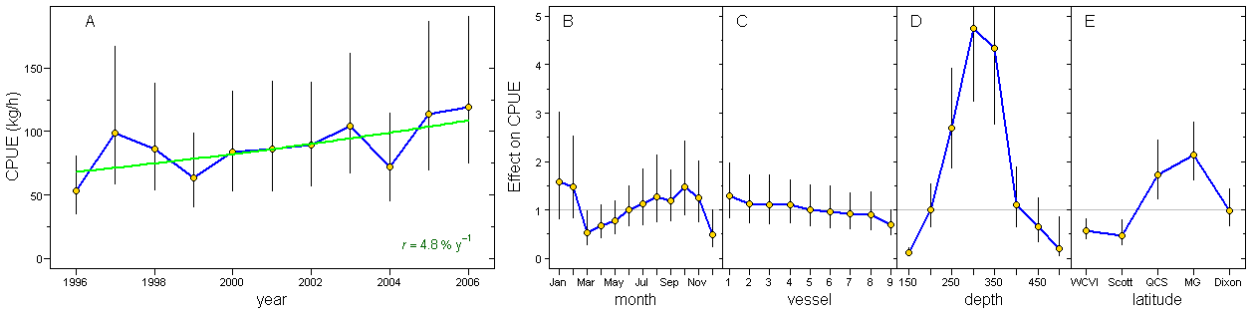

Figure 21. (*PBStools-glimmer*) Commercial POP CPUE U (kg/h) estimated from an additive lognormal model as $\log_2 U$: (A) year as $2^{\mu+\alpha_i}$, (B) month as 2^{β_j}, (C) vessel as 2^{γ_k}, (D) depth zone (50m intervals) as 2^{δ_l}, and (E) latitudinal groups as 2^{ϕ_m}. Vertical error bars indicate 95% confidence limits. Green line in (A) shows the back-transformed linear fit through α_i; implies an annual gain of 4.8%/y.

3.6 makeCATtables

This function creates tables of commercial catch by management fishing year. Various fisheries can be specified. The code scrolls through the specified fisheries, issuing the appropriate SQL query for each catch summary. The results are sent to a text file called `cattab-XXX.csv`, where `XXX` = the specified `strSpp` (species code). Table 3 lists the available SQL queries for this function.

Table 3. Codes for the available fisheries.

code	description	SQL query
1	GF Catch trawl tows (sales receipts)	gfc_catch_fyear.sql
2	PacHarvest trawl tows (observer or fisher logs)	pht_catch_fyear.sql
3	PacHarvHL Zn hook & line validation records	phhl_vcatch_fyear.sql
4	PacHarvHL Zn hook & line fisherlog records	phhl_fcatch_fyear.sql
5	PacHarvHL Schedule II hook & line validation records	phhl_vcatch_fyear.sql
6	PacHarvHL Schedule II hook & line fisherlog records	phhl_fcatch_fyear.sql
7	PacHarvHL Halibut longline validation records	phhl_hcatch_fyear.sql

Example (makeCATtables):

```
makeCATtables(strSpp="396",comm=2)
```

Sends the output displayed in Table 4 to `cattab-396.csv`:

Table 4. POP catch (t) from running `makeCATtables`.

fyear	3C	3D	4B	5A	5B	5C	5D	5E	UNK
1996	118	497	0	407	4180	561	32	694	0
97	54	47	NA	36	521	62	0	372	NA
1997	404	214	0	876	3323	372	181	647	NA
1998	299	164	0	916	3098	477	226	766	NA
1999	309	246	NA	981	3275	610	136	666	NA
2000	293	233	NA	578	3082	369	160	1257	NA
2001	293	207	0	660	3084	329	173	1133	NA
2002	300	264	0	794	3136	320	171	1125	6
2003	304	218	0	762	3551	245	108	920	NA
2004	317	220	0	752	3722	130	72	904	NA
2005	285	219	0	856	2894	139	185	761	NA
2006	275	207	0	503	3101	81	44	853	NA
UNK	NA	NA	NA	NA	NA	NA	NA	NA	508

3.7 plotCatch

This function displays the annual catch history contained in the specified file as stacked bars (Figure 22). Each bar shows the cumulative catch from specified fields.

Decadal mean catch is displayed in horizontal bands. If the user wishes to change the bar colours by passing in a vector `col` to the function, s/he must also pass in a similar vector `fill` for the legend.

Example (plotCatch):

```
plotCatch(dat=ymr.rem)
```

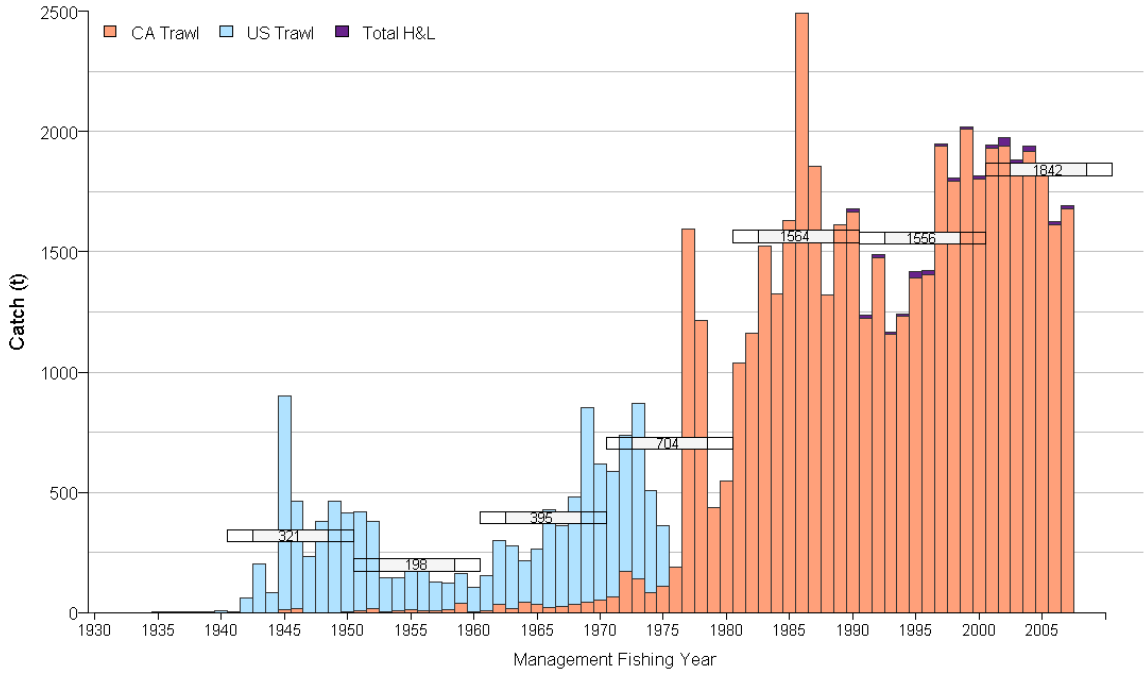

Figure 22. (*PBStools-plotCatch*) Catch history of Yellowmouth Rockfish (*Sebastes reedi*) by US and Canadian fleets along the BC coast. Mean annual catches by decade are displayed in horizontal boxes.

3.8 plotConcur

Create a barplot of the top 20 concurrent species from tows capturing the target species `strSpp` between a minimum and maximum depth. This function uses the SQL code `pht_concurrent.sql` to query the observer trawl database PacHarvest. A variant of this (`phhl_concurent.sql`) taps into the hook and line database PacHarvHL. This function can only be used by personnel logged onto the DFO network.

The default settings for this function create a one-panel plot (Figure 23). If a multiple figure plot is desired, say `par(mfrow=c(2,2))`, then set the argument `reset.mf=FALSE` in calls to `plotConcur` after setting the `mfrow` option in `par`.

Example (plotConcur):

```
plotConcur(strSpp="410", mindep=150, maxdep=435) # DFO only
```

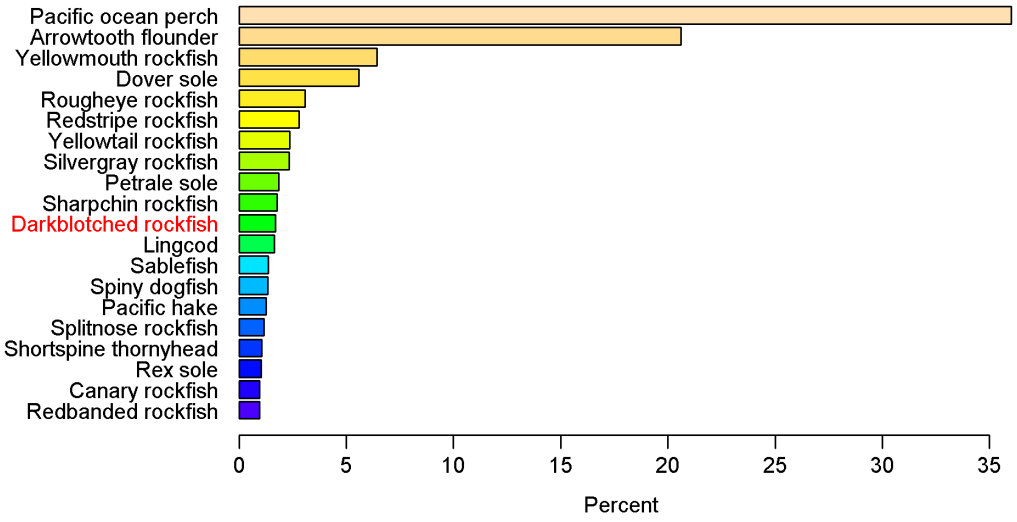

Figure 23. (*PBStools-plotConcur*) Concurrence of species in trawl tows (1996-2007) capturing Darkblotched Rockfish (*Sebastes crameri*) at depths 150–435 m. Abundance is expressed as a percent of total catch weight. The target species is highlighted in red.

3.9 plotFOScatch

This function queries the DFO FOS[4] database for catches of the specified species, and plots monthly catches as bars stacked by PMFC area. The SQL query `fos_catch.sql` (DFO personnel only) obtains catch from all fisheries, and the user chooses which fishery to display. Barplot categories are combinations of year and month, and catch is colour-coded by PMFC area (blues for 3C-3D, yellow-orange-red for 5A-5B-5C, and greens for 5D-5E) to provide a visual feel for the part of the coast (WCVI[5], QSC[6], or WCHG[7]) that provides the predominant catch (Figure 24). The total catch is reported at the end of each fishing year.

In addition to the plot, catch summaries by fishery, year, and PMFC area are dumped to a file called `catFOSfyr.csv`. Also, internally-derived objects of interest (such as catch-by-month and catch-by-year matrices) are saved in the list object `PBStool` located in the environment `.PBStoolEnv`.

Example (plotFOScatch):

```
plotFOScatch(strSpp="401")  # DFO only
```

3.10 runCCA

Perform a catch-curve analysis using a model formulated by Schnute and Haigh (2007). The code allows a user to perform both frequentist (NLM) and Bayesian (BRugs) analyses.

[4]Fishery Operations System
[5]west coast of Vancouver Island
[6]Queen Charlotte Sound or central BC coast
[7]west coast Haida Gwaii (offshore west, Hecate Strait to the east, & Dixon Entrance to the north)

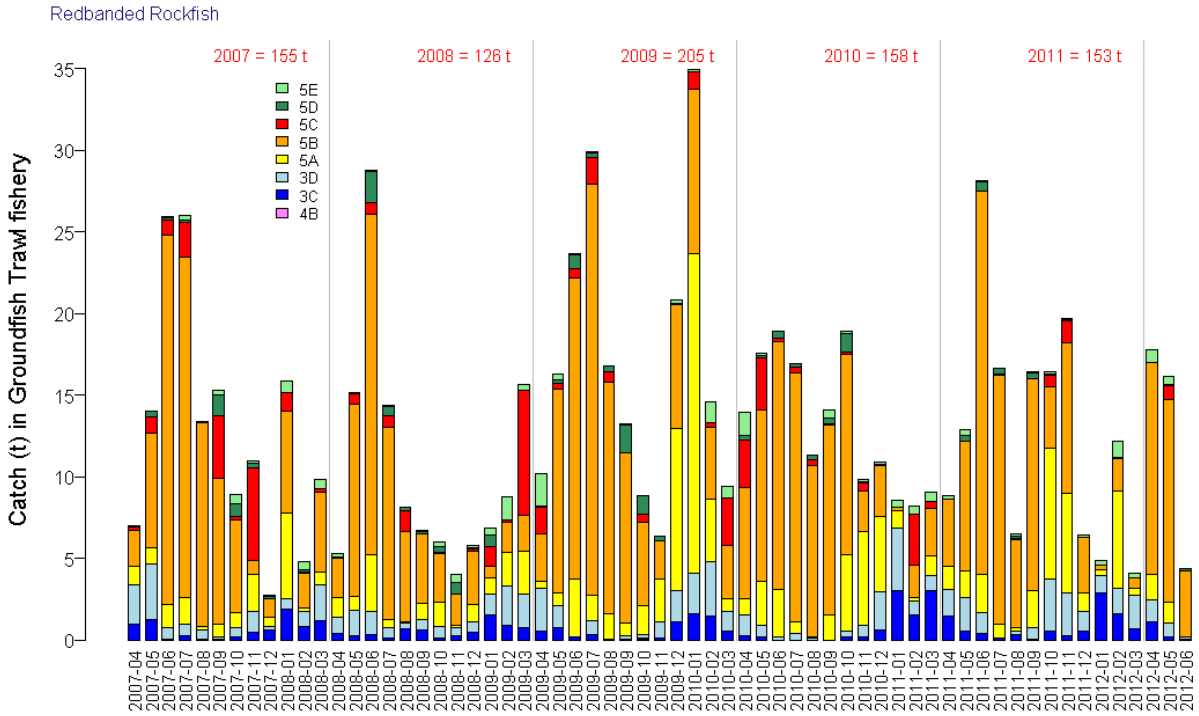

Figure 24. (*PBStools-plotFOScatch*) Monthly catch of Redbanded Rockfish (*Sebastes babcocki*) from BC's seven offshore PMFC areas. Fishing year is assumed to run from April to March.

The function opens up a notebook GUI with two tabs named NLM and BRugs. The user needs to find a decent modal fit using `NLM` before attempting to run the Bayesian analysis, which employs Markov chain Monte Carlo (MCMC) techniques using the Gibbs sample algorithm.

Currently, the NLM fit can use one of three sampling distributions: multinomial, Dirichlet, and logistic-normal. The BRugs model is only coded for four cases using the Dirichlet distribution:

`M`	case 1: survival only, which depends only on mortality,
`MS`	case 2: survival and selectivity only,
`MA`	case 3: survival and recruitment anomalies only,
`MSA`	case 4: survival, selectivity, and recruitment anomalies.

The Bayesian model is coded in the OpenBUGS language and can be found in the examples subdirectory `library/PBStools/examples`.

3.11 sumCatTabs

The catch reconstruction algorithm of Haigh and Yamanaka (2011) builds input catch data files for a specified rockfish species from numerous databases (SQL Server and Oracle). This function takes those input files and summarizes catch by year and PMFC area.

3.12 trackBycat

Track fish group catches by year and PMFC area that occur between depths corresponding to a target species' depth-of-occurrence. The function currently uses three SQL query files called `gfb_bycatch.sql`, `pht_bycatch.sql`, and `fos_bycatch.sql`, which tap into the DFO databases GFBioSQL, PacHarvest, and GFFOS, respectively.

The function displays two barplots of annual catch, absolute and relative, where bars are sectioned by fish groups.

If `rda=NULL`, then three SQL queries extract annual fish group catches from the databases above. These catches are transferred to a 3-dimensional array `bycatch`:

`year`	years spanning all datasets;
`fish`	fish groups: POP, rockfish, turbot, flatfish, hake, sharks, other;
`db`	DFO databases: `gfb` = GFBioSQL, `pht` = PacHarvest, `fos` = GFFOS.

The catch array is saved to an `.rda` file with a name that indicates major areas and trawl type (bottom or midwater).

If the argument `rda` is given an `.rda` name, the `bycatch` array is loaded, by-passing the SQL calls.

The `bycatch` array is returned invisibly. Additionally, a list object called `PBStool`, located in the user's global environment, is populated with:

`module`	the name of the **PBStools** module to which this function belongs;
`call`	the call to the function `trackBycat`;
`plotname`	name of the plot, should the user which to use it, e.g. `.plotDev(act="png")`;
`bartab`	matrix of catch (tonnes) by year and fish group, summed over the databases;
`amat`	matrix of absolute catch (kt) used in the upper barplot;
`rmat`	matrix of relative catch (0:1) used in the lower barplot;
`clrs`	vector of colour names to distinguish the fish groups within any one bar.

4 Survey functions

The survey functions arose from a paper we wrote for designing trawl surveys (Schnute and Haigh, 2003). Essentially, every survey stratum hosts a unique population for the target species of interest. Each population can be summarized by three parameters that determine a binomial-gamma distribution: p = proportion of tows that do not catch the target species, μ = mean density of the target species in tows that catch the target species, and ρ = the coefficient of variation of these positive-catch densities.

4.1 bootBG

Bootstrap stratified biomass estimates using random draws from the binomial-gamma distribution using strata population parameters (Schnute and Haigh, 2003) supplied. This function depends on the R-package **boot**. These parameters are used to estimate the survey population using random draws from the binomial-gamma distribution.

No value is explicitly returned by the function. A list object `PBStool` (located in the environment `.PBStoolEnv`) provides results of the analysis:

`dat`	data frame: a subset from the user input that contains population parameters for each unique stratum.
`bgtab`	matrix: parameter values and moment estimates for each stratum.
`dStab`	matrix: sampled density estimates for each stratum.
`BStab`	matrix: sampled biomass estimates for each stratum.
`bgsamp`	list: final simulated set of densities sampled from the binomial-gamma distribution and listed by stratum.
`Bboot`	list: results of boot::boot() listed for each simulation.
`Bqnt`	array: bootstrap quantiles for each simulation and type of confidence interval calculation.
`Best`	vector: biomass estimate for each simulation.
`Bobs`	scalar: observed biomass estimate (moment calculation).
`group`	vector: stratum grouping showing allocation of K tows.

The results contained in `PBStool` can be displayed using the function `showAlpha` (Figure 25), which plots bootstrapped biomass values and their quantile confidence levels α (see Fig. 3 in Schnute and Haigh (2003)).

Example (bootBG):

```
pbsfun=function(SID=1,S=500){
  bootBG(dat=pop.pmr.qcss,SID=SID,S=S)
  showAlpha()
  invisible() }
pbsfun()
```

4.2 calcMoments

Calculate survey moments, including relative biomass, for a species from raw survey data. The function uses `getData` to calculate survey specs using the SQL queries:

```
gfb_survey_stratum.sql
gfb_survey_catch.sql
```

These queries are modified SQL procedures originally designed by Norm Olsen (PBS, Nanaimo) to build the GFBioSQL tables:

```
BOOT_HEADER
BOOT_DETAIL
```

The function invisibly returns a list of survey moments (`B` = biomass estimate, `N` = number of tows, `V` = variance of biomass estimate, `sig` = standard deviation of biomass estimate, `CV` = coefficient of variation) by strata. The following example gets moment estimates for Sablefish (*Anoplopoma fimbria*) from the QCS Synoptic survey in 2003 and 2007.

Example (calcMoments):

```
pbsfun=function(strSpp="455", survID=c(1,121),dfo=TRUE){
  if (dfo) {
    momList=calcMoments(strSpp,survID)
    print(momList) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
```

Output (calcMoments):

```
$`1`
                B   N            V         sig         CV
18       1.058758  29 1.253268e-06 0.001119494 0.10573660
19     240.446606  56 1.186942e-02 0.108946846 0.04531020
20     343.011654  29 4.480791e-02 0.211678786 0.06171183
21      97.324284   6 5.088662e-03 0.071334858 0.07329605
22       2.187556   5 5.840734e-06 0.002416761 0.11047767
23      26.737186  39 1.892251e-04 0.013755912 0.05144861
24     105.330602  51 1.364367e-03 0.036937341 0.03506801
25     354.018155  19 3.207522e-02 0.179095553 0.05058937
Total 1170.114799 234 9.540189e-02 0.308871962 0.02639672

$`121`
               B   N            V         sig         CV
18      1.429550  33 2.599952e-06 0.001612437 0.11279330
19    168.390766  62 8.226319e-03 0.090699057 0.05386225
20    142.210387  24 4.710287e-03 0.068631531 0.04826056
21    126.053435   7 6.162359e-02 0.248240990 0.19693314
22      5.755936  19 8.220823e-05 0.009066875 0.15752218
23     15.778769  57 1.383198e-04 0.011760944 0.07453651
24    147.342124  48 1.772628e-03 0.042102586 0.02857471
25    265.025548   7 1.680502e-02 0.129634159 0.04891383
Total 871.986514 257 9.336097e-02 0.305550267 0.03504071
```

4.3 calcPMR

This function calculates the binomial-gamma parameters (p, μ, ρ) of a sample population, as described by Schnute and Haigh (2003):

`p` = proportion of zero values,
`mu` = the mean of the non-zero values,
`rho` = the CV of the non-zero values.
The output is a vector `c(n, p, mu, rho)`.

Example (calcPMR):

```
pbsfun=function(P=list(n=1000,p=0.25,mu=1,rho=0.5)) {
  unpackList(P)
  x = sampBG(n,p,mu,rho) # create a sample population
  y = calcPMR(x)
  cat("True parameters for sampling the binomial-gamma distribution:\n")
  print(unlist(P))
  cat("Estimated parameters of the sampled binomial-gamma distribution:\n")
  print(y)
  invisible() }
pbsfun()
```

Output (calcPMR):

```
True parameters for sampling the binomial-gamma distribution:
      n       p      mu     rho
1000.00    0.25    1.00    0.50
Estimated parameters of the sampled binomial-gamma distribution:
           n            p           mu          rho
1000.0000000    0.2340000    1.0244274    0.4912166
```

4.4 getBootRuns

This function calls the SQL query `gfb_boot.sql` to download the bootstrap results for the selected species from GFBioSQL tables `BOOT_HEADER` and `BOOT_DETAIL`. The bootstrap tables are maintained by Norm Olsen (PBS, Nanaimo).

The function returns a data frame containing the bootstrapped biomass estimates and confidence limits for `strSpp` from all surveys in which the species was observed (see output below for Roughtail Skate *Bathyraja trachura*). The data frame is ordered by `date` and `bootID`.

Example (getBootRuns):

```
pbsfun=function(strSpp="057",dfo=TRUE) {
  if (dfo) {
    bootTab=getBootRuns(strSpp)
    print(bootTab) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
```

Output (getBootRuns):

```
  survID bootID       date year defDoor defSpeed
1      1     17 2003-07-03 2003    72.4      5.8
2      1     35 2003-07-03 2003    72.4      5.8
3      3     19 2005-07-05 2005    75.2      5.8
4      3     37 2005-07-05 2005    75.2      5.8
6    123     10 2006-08-28 2006    64.3      5.5
5    122     12 2007-09-11 2007    68.5      5.4
7    129     34 2008-08-25 2008    80.7      5.1

                                        runDesc    biomass   bootMean
1                                2003 QCS Synoptic 15351.1431 15235.4372
2 April 2009 Update for 2003 QCS Synoptic Survey 15395.4115 15158.6269
3                                2005 QCS Synoptic   607.6942   604.6557
4 April 2009 Update for 2005 QCS Synoptic Survey   607.6942   585.2095
6            2006 WQCI with 2007 stratification.  1187.0096  1187.4331
5                               2007 WCQCI Synoptic  7818.5815  7850.4116
7                               2008 WCQCI Synoptic 26656.8828 26645.2189

  bootMedian  bootLoCI  bootHiCI    bootRE catchWt numSets numPosSets
1 14366.3844  4515.370 44178.042 0.5411575   27.20     235          4
2 13944.6661  5033.791 46233.239 0.5540315   27.20     234          4
3   607.6942     0.000  1215.388 0.9110855    1.20     224          1
4   607.6942     0.000  1823.083 0.9677396    1.20     224          1
6  1187.0096     0.000  3409.161 0.6316274    4.91      96          3
5  7540.2430  2155.460 22393.798 0.5037474   25.58     112          4
7 25591.3002 10155.392 60289.184 0.4287963   26.56     117          8
```

4.5 getPMR

Get `p`, `mu`, and `rho` values for populations by strata from survey data housed in GFBioSQL using the query `gfb_pmr.sql` stored in `library/PBStools/sql`.
The function executes the SQL query for each survey ID `i`:

```
getData("gfb_pmr.sql","GFBioSQL",strSpp,path=.getSpath(),surveyid=i)
```

and collects the results in a data frame `pmrTab`. Each row of this data frame describes the population parameters (`p`, `mu`, `rho`) for each unique survey ID and survey stratum `h`. The output table also has named attributes: `fqt` (query name), `strSpp` (string species code), `h` (strata), `surveys.all` (data frame from a call to `getBootRuns`), and `surveys.selected` (records from `surveys.all` with the specified `survID`).

Example (getPMR):

```
pbsfun=function(dfo=TRUE){
  if (dfo) {
    pmrTab=getPMR(strSpp="396", survID=c(1,121))
    print(pmrTab) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
```

Output (getPMR):

```
pmr for species 396
-------------------
Survey ID: 1, 121
Done and done in 2.4 sec
   SID  h          p           mu       rho    A  n k
1    1 18 0.86206897 0.0086473494 1.0956975 5093 29 1
2    1 22 0.80000000 0.0020885547 0.4472136 2024  5 1
3    1 24 0.03921569 1.4766607698 1.5224993 3976 51 1
4    1 23 0.46153846 0.1064740216 1.6957557 4332 39 1
5    1 21 0.00000000 1.7507531455 0.5205263  496  6 1
6    1 25 0.00000000 0.3422540526 1.5298557 1220 19 1
7    1 20 0.00000000 3.1035614080 0.8172012 2931 29 1
8    1 19 0.21428571 1.5815733782 2.2459383 5582 56 1
9  121 18 0.96969697 0.0013227513 0.1740777 5093 33 1
10 121 22 0.94736842 0.0002308403 0.2294157 2024 19 1
11 121 24 0.00000000 0.9134459070 1.9488203 3976 48 1
12 121 23 0.45614035 0.2017947511 2.7973655 4332 57 1
13 121 21 0.14285714 0.9402312891 1.1255596  496  7 1
14 121 25 0.28571429 0.2134502924 0.5085555 1220  7 1
15 121 20 0.00000000 1.7714366587 1.1180694 2931 24 1
16 121 19 0.50000000 0.7139279217 1.9723779 5582 62 1
```

4.6 makePMRtables

Create `.csv` files containing (p, μ, ρ) (Schnute and Haigh, 2003) values for strata in the specified surveys. The code scrolls through the survey IDs, issuing sequential SQL queries to the DFO database GFBioSQL, and collects the debris in `.csv` files, one for each name in the list surveys.

Example (makePMRtables):

```
pbsfun=function(strSpp="451",dfo=FALSE) {
  if (dfo) makePMRtables(strSpp)
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
pbsfun()
```

Output:

See files `'pmrqcss.csv'`, `'pmrwcvi.csv'`, and `'pmrwqci.csv'` in your current working directory. These file contain binomial-gamma parameters for Shortspine Thornyhead (*Sebastolobus alascanus*) in each of the survey's strata.

4.7 makeSSID

Make a data object of survey series information called `ssid` for use in the package **PBSdata**. The function imports the SQL table `SURVEY` from the DFO database GFBioSQL using:

```
getData("SURVEY","GFBioSQL")
```

The survey information is then summarised by its series number and saved to a binary

`.rda` file.

4.8 sampBG

Take random samples from the binomial-gamma distribution. This function generates random samples from both the binomial and the gamma distributions, and returns the product of the two.

Example (sampBG):

```
pbsfun=function(P=list(n=1000,p=0.25,mu=1,rho=0.5)) {
  unpackList(P)
  x = sampBG(n,p,mu,rho) # create a sample population
  y = calcPMR(x)
  cat("True parameters for sampling the binomial-gamma distribution:\n")
  print(unlist(P))
  cat("Estimated parameters of the sampled binomial-gamma distribution:\n")
  print(y)
  invisible() }
pbsfun()
```

Output (sampBG):

```
True parameters for sampling the binomial-gamma distribution:
      n       p      mu     rho
1000.00    0.25    1.00    0.50
Estimated parameters of the sampled binomial-gamma distribution:
           n           p          mu         rho
1000.0000000   0.2600000   0.9897658   0.5169940
```

4.9 showAlpha

Show quantile confidence levels α for bootstrapped biomass. This function assumes that `bootBG()` has been run and that the data list object `PBStool` exists in the global position. The figure generated (Figure 25) follows the format of Figure 3 in Schnute and Haigh (2003). The panels show:

(a) histogram of bootstrapped biomass;
(b) cumulative probability distribution of bootstrapped biomass;
(c) normal quantiles of the estimates $\hat{B}$;
(d,e) observed probabilities that observed value B lies in the j^{th} confidence interval;
(f,g) $\hat{\alpha}_j$ vs. α_j for chosen confidence limit types.

Example (showAlpha):

```
ppbsfun=function(SID=1,S=1000){
  bootBG(dat=pop.pmr.qcss,SID=SID,S=S)
  showAlpha()
  invisible() }
pbsfun()
```

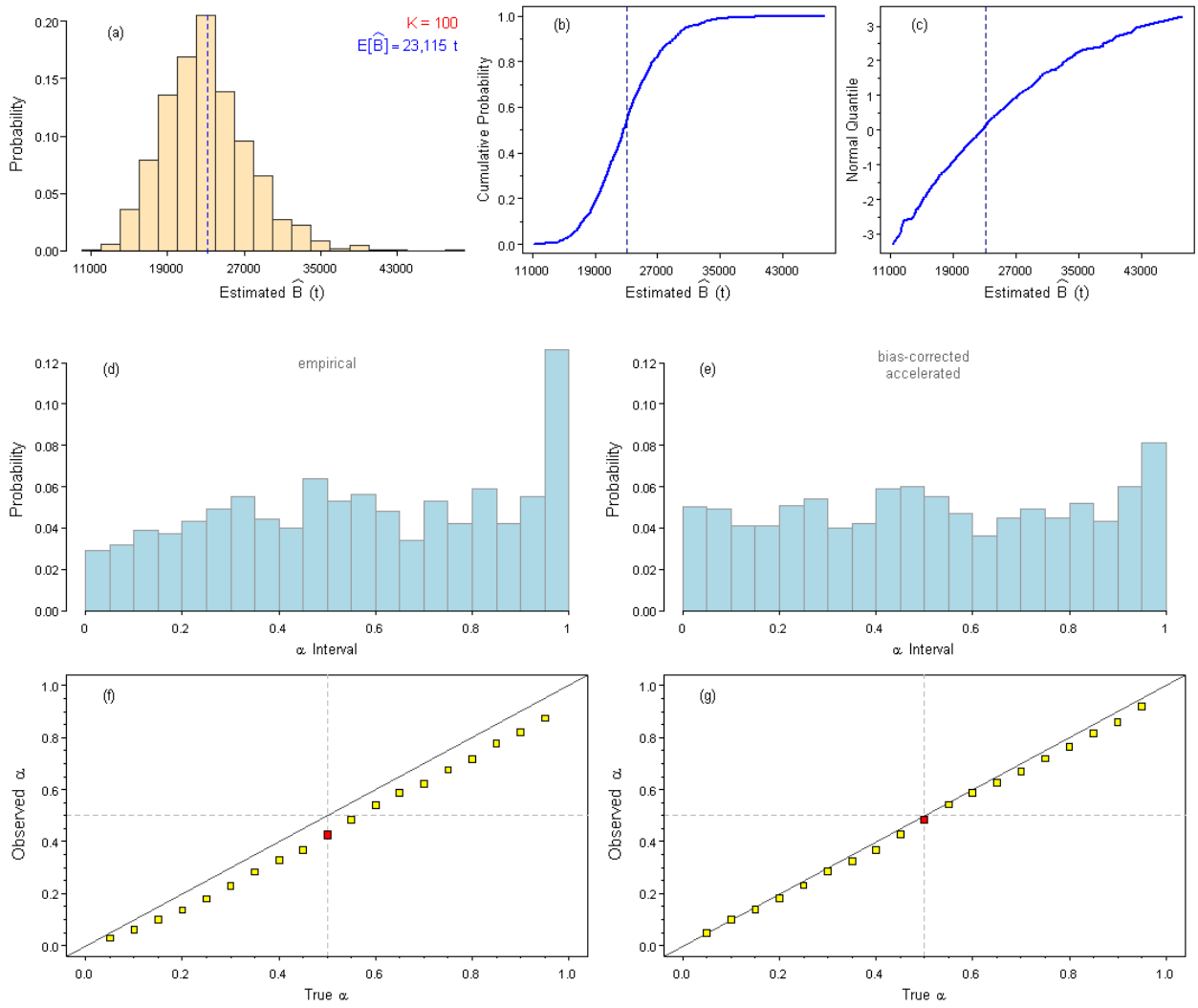

Figure 25. (*PBStools-showAlpha*) Results from S=1000 simulations, R=500 bootstraps, and tow budget K=100 for POP caught in eight strata of the Queen Charlotte Sound survey in 2003 (235 tows). (**Panels a-c**) (a) histogram, (b) cumulative probability distribution, and (c) normal quantiles for the S estimates $\hat{B}$. Vertical dashed lines in (b) and (c) denote the observed value B = 23t. (**Panels d-e**) Observed probabilities that the observed value B lies in the j^{th} confidence interval, based on (d) empirical confidence limits and (e) limits adjusted for bias correction and acceleration. (**Panels f-g**) plot of $\hat{\alpha}_j$ vs. α_j based on (f) empirical confidence limits and (g) limits adjusted for bias correction and acceleration. Vertical and horizontal dashed lines in (f g) show median levels at 50%, and solid lines denote the equality $\hat{\alpha} = \alpha$.

4.10 showIndices

Show survey indices for a species from bootstrapped biomass estimates stored in the SQL database GFBioSQL. The function relies on a short query function called `getBootRuns` which in turn queries tables of survey information (`BOOT_HEADER`) and relative biomass estimates by species for each survey (`BOOT_DETAIL`). The results are plotted with their confidence intervals (Figure 26). The user should be aware that any survey ID `survID` can have multiple boot runs `bootID` (e.g., re-stratification trials).

Example (showIndices):

```
pbsfun=function(strSpp="602",region="QCS",dfo=TRUE){
  if (dfo) {
    choices=c("WCVI","QCS","WQCI","HS")
    i=match(region,choices,nomatch=0)
    if (i==0) showError(paste("Choose region from\n'",
      paste(choices,collapse="', '"),"'",sep=""),as.is=TRUE)
    survID=switch(i,
      c(16,58,70,124,125,126,128),
      c(1,2,3,59,121,127),
      c(57,122,123,129),
      c(15,111,162,163) )
    showIndices(strSpp,survID) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
```

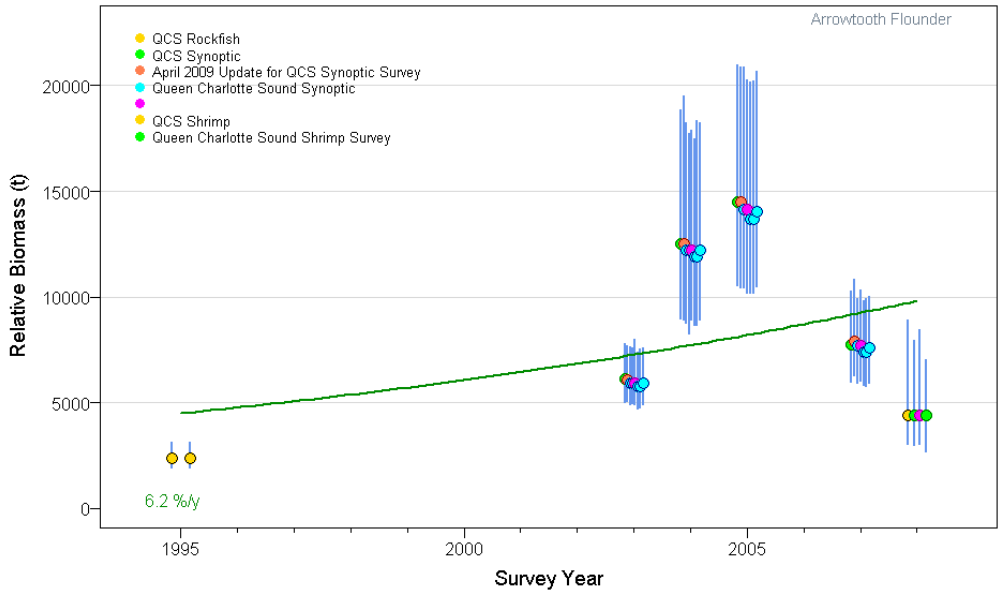

Figure 26. (*PBStools-showIndices*) Relative abundance indices from surveys in Queen Charlotte Sound for Arrowtooth Flounder (*Atheresthes stomias*). The bootstrapped results are stored on DFO's SQL Server database GFBioSQL. Duplicated points usually indicate a restratification exercise but can occasionally represent two surveys in the same year.

4.11 simBGtrend

Simulate a population projection based on prior binomial-gamma parameters and display the results. This function attempts to simulate a population past the observed year by randomly sampling the binomial-gamma distribution using weighted prior values of the population parameters `p`, `mu`, and `rho`.

The forward simulation is essentially based on momentum without reference to age classes, environment, or mortality. Users should be aware that this simulation remains naïve.

Example (simBGtrend):

```
pbsfun=function(Nsim=1,alpha=1) {
  simBGtrend(Nsim=Nsim,alpha=alpha)
  invisible() }
pbsfun(Nsim=7)
```

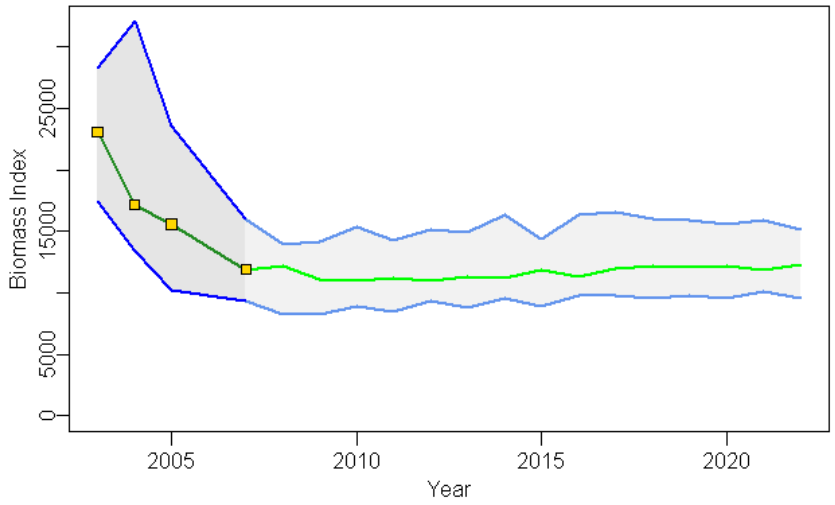

Figure 27. (*PBStools-simBGtrend*) Forward simulation (based on 7 trajectories) of POP abundance in Queen Charlotte Sound using past values of stratum population parameters (p, μ, ρ).

4.12 trend

An interactive GUI menu (Figure 28) created by a call to `trend` facilitates the exploration of trends in survey data. Once the user chooses a dataset, pressing the TREND button displays boxplots of catch or CPUE grouped by year. A trend line (Figure 29) fit through annual points, summarized by a user-specified summary function (e.g., `mean`), yields an annual rate of change r and the accumulated change R over the period of the fit (Schnute *et al.*, 2004b). Pressing the BOOT button generates and displays bootstrapped estimates of slope b and annual rate r.

The package includes three sample datasets from the International Pacific Halibut

Commission (IPHC)[8]: `iphc.rbr`, `iphc.rer`, and `iphc.yyr` for Redbanded Rockfish (*Sebastes babcocki*), Rougheye Rockfish (*S. aleutianus*), and Yelloweye Rockfish (*S. ruberrimus*), respectively. These data have been summarized extensively, the latest by Obradovich *et al.* (2008). The IPHC director, Dr. Bruce Leaman, has kindly granted PBS permission to include them here.

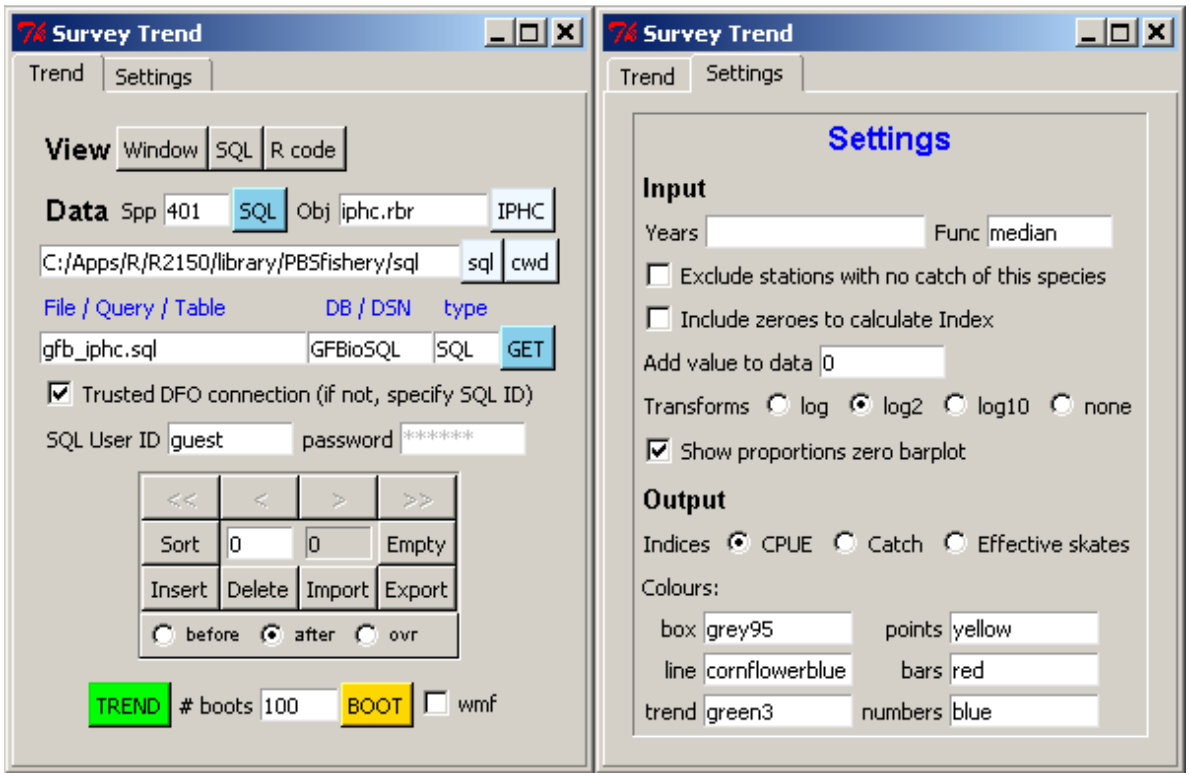

Figure 28. (*GUI-trend*) GUI menu system created by a call to `trend`. The user can either specify a data object or launch an SQL query to GFBioSQL (DFO only).

[8]International Pacific Halibut Commission (IPHC), P.O. Box 95009, Seattle, WA 98145-2009, U.S.A

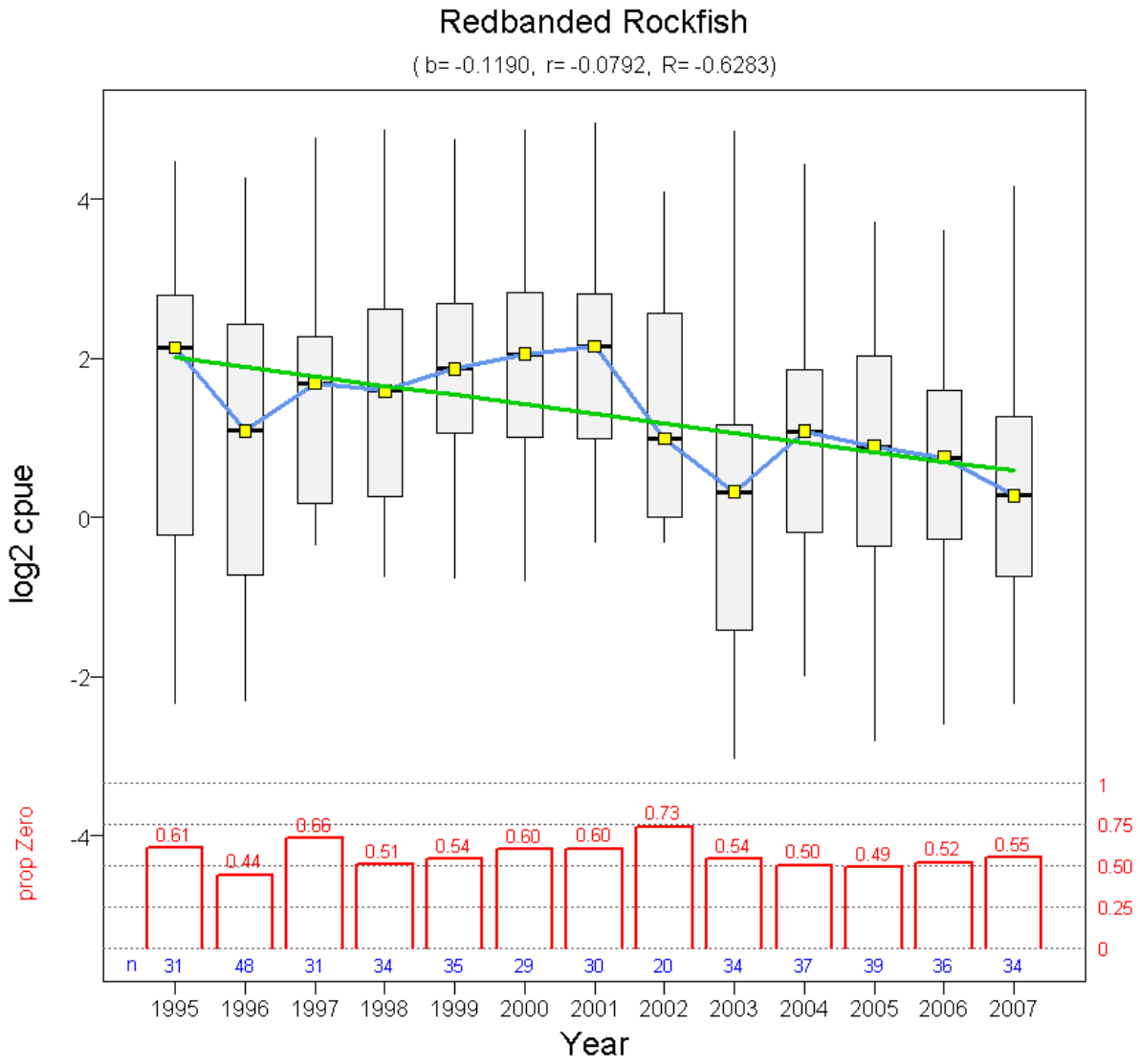

Figure 29. (*PBStools-trend*) IPHC survey CPUE index ($\log_2$(number of fish/effective skate)) for Redbanded Rockfish (*Sebastes babcocki*). The boxplots summarize annual non-zero data, where box whiskers show the extent of the data and box horizontal division indicates the median. Square symbols show the annual values of the specified summary function (in this case `median`). Superimposed on the boxplots is a regression line of the summary function. Barplots below show the proportion of non-zero values in the original dataset, and values at the base of the bars indicate the number of points used in the index calculation. Values in the subtitle summarize the trend line, where b = slope of the regression, r = annual rate of change, and R = accumulated change over the period of the trend line.

5 Spatial functions

Spatial functions arose primarily from work that DFO contributed to COSEWIC[9] in collaboration with the federal Ministry of Environment. Species-at-risk legislation requires that we know the spatial extent of marine species flagged for concern.

5.1 calcHabitat

This function calculates potential habitat using specified bathymetry limits. The default topography dataset `bctopo` was downloaded from the site http://topex.ucsd.edu/cgi-bin/get_data.cgi after specifying these boundaries: west (-134.5), east (-124.5), south (48), north (55). A user from another region could download a similar data file with different boundaries.

`calcHabitat` uses the **PBSmapping** function `makeTopography`, which slows down as the precision chosen for the longitude-latitude data increases. Setting `digits=1` provides a fast but roughly-estimated habitat region. Increasing the precision to `digits=2` slows the code markedly but provides a much more detailed bathymetry outline (Figure 30).

Example (calcHabitat):

```
calcHabitat(isob=c(150,435), digits=2)
```

5.2 calcOccur

This function calculates the percent occurrence of events (`EventData`) in a set of labelled polygons (`PolySet`). Consequently, the `PolySet` must have a `PolyData` attribute with a category field called `label` that provides categories or bins in which to count events and calculate percent occurrence. The results for the tow subset `testdatC` in various categories of surficial geology for the Queen Charlotte Basin (Barrie *et al.*, 1991; Sinclair *et al.*, 2005) are displayed in Figure 31.

Example (calcOccur):

```
pbsfun=function(){
  data(testdatC)
  dat=testdatC[!is.na(testdatC$X) & !is.na(testdatC$Y),]
  edat=as.EventData(dat,projection="LL",zone=9)
  calcOccur(polyset="qcb", events="edat", mess=TRUE)
  invisible() }
pbsfun()
```

5.3 calcSRFA

Determine the slope rockfish assessment areas from combinations of PMFC major and minor areas as well as locality codes for fishing grounds within PMFC minor areas. The function either accepts three vectors (`major`, `minor`, `loc`) of equal length or one matrix/data frame `major` with three columns corresponding to the first three arguments. Slope rockfish assessment subareas or gullies can be determined by setting the argument `subarea=TRUE`.

[9] Committee on the Status of Endangered Wildlife in Canada (http://www.cosewic.gc.ca/)

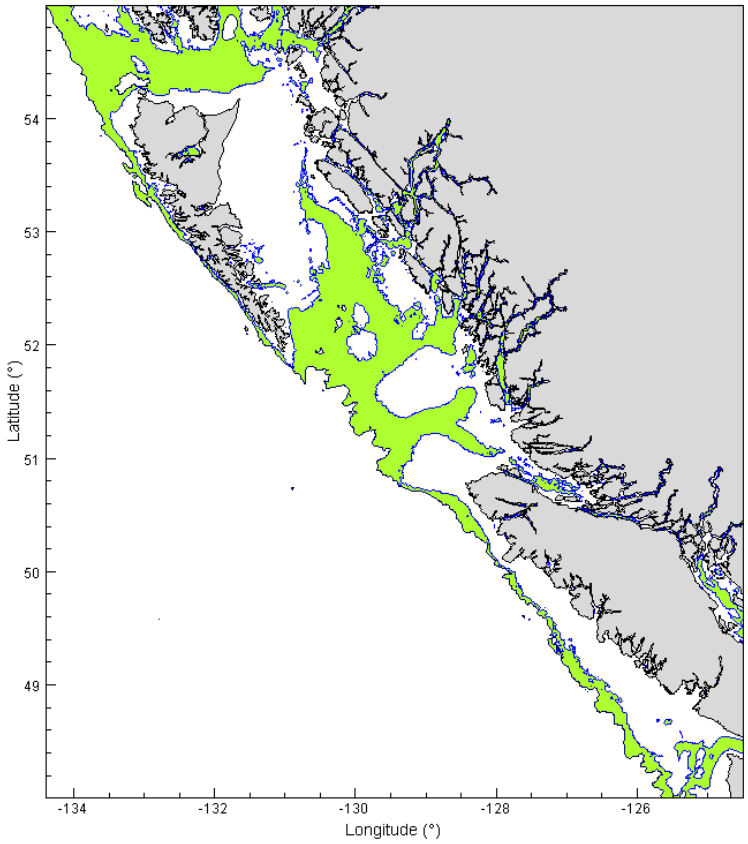

Figure 30. (*PBStools-calcHabitat*) Highlighted bathymetry (green) between 150 and 435 m serves as a proxy for Darkblotched Rockfish (*Sebastes crameri*) habitat along the BC coast.

Percent Occurrence:

Holocene Sand & Gravel = 51.5%
Sand & Gravel / Bedrock = 13.4%
Outwash Sand & Gravel = 12.7%
Holocene Mud = 8.7%
Bedrock = 4.0%
Glaciomarine Mud = 3.9%
Sand & Gravel / Glaciomarine Mud = 2.8%
Till = 2.7%
Sand & Gravel = 0.2%

Figure 31. (*PBStools-calcOccur*) Percent occurrence of tows from `testdatC` in the surficial geology categories of the Queen Charlotte Basin `PolySet`.

Example (calcSRFA):

```
pbsfun=function(){
  dat=data.frame(major=c(3:7,9,9),
    minor=c(23,26,11,8,8,35,31),loc=c(1,1,1,3,6,1,1))
  sdat=cbind(dat,srfa=calcSRFA(dat),srfs=calcSRFA(dat,subarea=TRUE))
  print(sdat); invisible() }
pbsfun()
```

Table 5. Output from `calcSRF` function.

	major	minor	loc	srfa	srfs
1	3	23	1	3C	
2	4	26	1	3D	
3	5	11	1	5AB	GS
4	6	8	3	5AB	MI
5	7	8	6	5CD	MR
6	9	35	1	5EN	
7	9	31	1	5ES	

5.4 calcSurficial

Calculate the intersection of two `PolySets`, specifically surficial geology and potential habitat using a bathymetry interval. The results of the intersection are displayed on a map, including the area (km^2) of surficial geology categories that occurs in the bathymetry interval or potential habitat (Figure 32).

Example (calcSurficial):

```
pbsfun=function() {
  calcHabitat(isob=c(150,435), digits=1, plot=FALSE) # DBR
  habitat = PBStool$habitat
  calcSurficial(surf="qcb", hab=habitat)
  invisible() }
pbsfun()
```

5.5 clarify

This function summarizes large datasets of catch represented by many species into n CLARA[10] clusters using the function `clara` in the package **cluster** (Kaufman and Rousseeuw, 1990). Essentially, `clara` sub-samples the large data set, identifying the best k medoids (centre is defined as the item which has the smallest sum of distances to the other items in the cluster) using a dissimilarity metric. In the end, all records are assigned to one of the k clusters. Further routines in the R-package **PBSmapping** locate fishing events in grid cells and display the results (Figure 33).

The demo `EventData` set used below is a subset of a potentially much larger file that might comprise hundreds of thousands of tows and 80+ fish species. This dataset has been pared down to the region bounded by 130.5°W, 128°W, 50.8°N, and 52°N. Additionally, it excludes all depths less than 100 m and only reports catch for 13 fish species.

[10] **C**lustering **Lar**ge **A**pplications

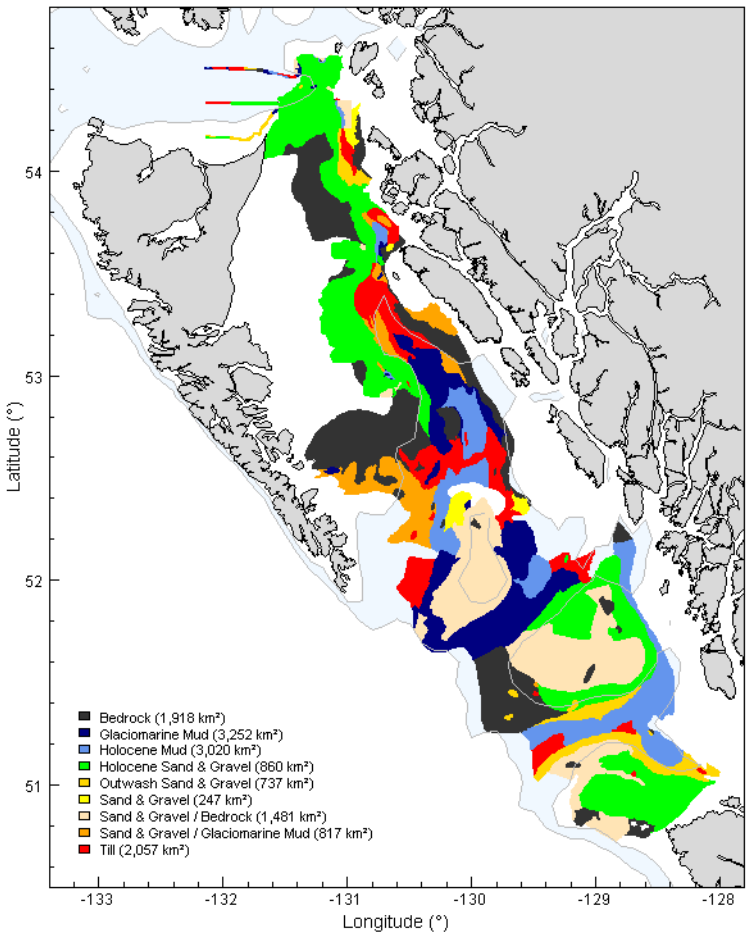

Figure 32. (*PBStools-calcSurficial*) Surficial geology of the Queen Charlotte basin and Hecate Strait. A rough bathymetry contour (150–145 m, derived from `calcHabitat` with `digits=1`) lies beneath the highlighted geology. The legend shows geology categories and the area (km^2) intersecting the bathymetry interval.

Example (clarify):

```
pbsfun=function(){
  clarify(claradat, xlim=c(-132,-127.8), ylim=c(50.6,52.1),
    cell=c(.05,.03),nG=7)
  invisible() }
pbsfun()
```

5.6 findHoles

Identifies polygons within polygons of the same PID, and transforms them into holes, if they are not already set that way. Also weeds out small polygons before the hole identification routine runs. This function can become very slow when there are hundreds of polygons. Hence the argument `minVerts` can reduce the number of candidate polygons. Ideally, `findHoles` should be programmed in C.

The output is a PolySet that identifies parents and holes based on the position of polygons

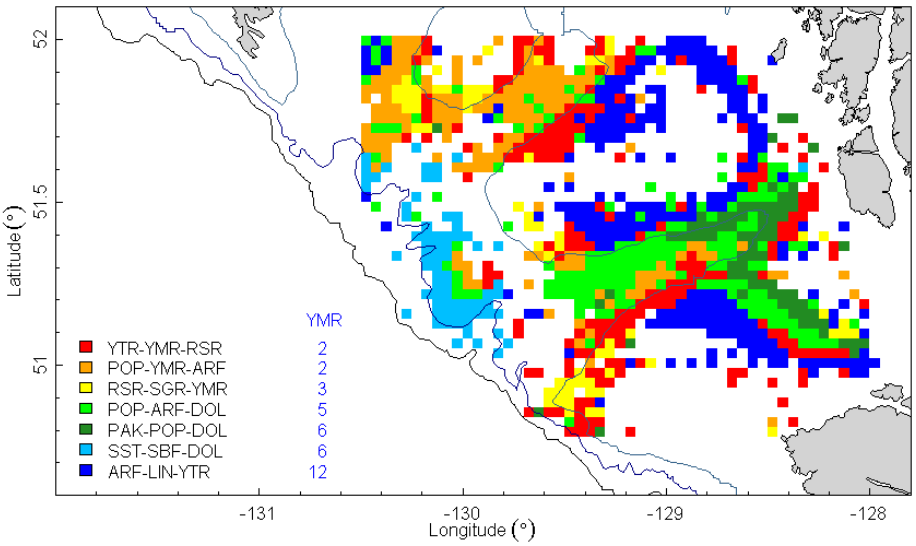

Figure 33. (*PBStools-clarify*) Groups identified by `clara` in R's package **cluster** and summarized by **PBSmapping** spatial functions called by `clarify`. Isobaths trace the 200, 1000, and 1800 m depth contours. The legend identifies seven clusters by the top three species comprising the medoids; the clusters are ordered by the contribution of Yellowmouth Rockfish (YMR) to each medoid.

Table 6. Species codes displayed in Figure 33.

ARF	Arrowtooth Flounder	*Atheresthes stomias*
BIS	Big Skate	*Raja binoculata*
DOL	Dover Sole	*Microstomus pacificus*
LIN	Lingcod	*Ophiodon elongatus*
PAK	Pacific Hake	*Merluccius productus*
POP	Pacific Ocean Perch	*Sebastes alutus*
ROL	Rock Sole	*Lepidopsetta bilineatus*
RSR	Redstripe Rockfish	*Sebastes proriger*
SBF	Sablefish	*Anoplopoma fimbria*
SGR	Silvergrey Rockfish	*Sebastes brevispinis*
SST	Shortspine Thornyhead	*Sebastolobus alascanus*
YMR	Yellowmouth Rockfish	*Sebastes reedi*
YTR	Yellowtail Rockfish	*Sebastes flavidus*

within polygons.

5.7 plotGMA

Plot the groundfish management areas (GMA) for Pacific Ocean Perch (*Sebastes alutus*) and Yellowmouth Rockfish (*S. reedi*) used by Pacific groundfish managers at RHQ in Vancouver. The PolySet `gma.popymr` can be found in the R package **PBSvault**, which is only available to DFO personnel, and contains an attribute called `PolyData` that defines

colours and labels for each of the GMAs.

Example (plotGMA):

```
pbsfun=function(){
  if(is.element("PBSvault",.packages(all.available=TRUE))) {
    require(PBSvault); data(gma.popymr)
    pdata=attributes(gma.popymr)$PolyData
    plotMap(gma.popymr,polyProps=pdata)
    addLabels(pdata,cex=2,font=2) }
  invisible() }
pbsfun()
```

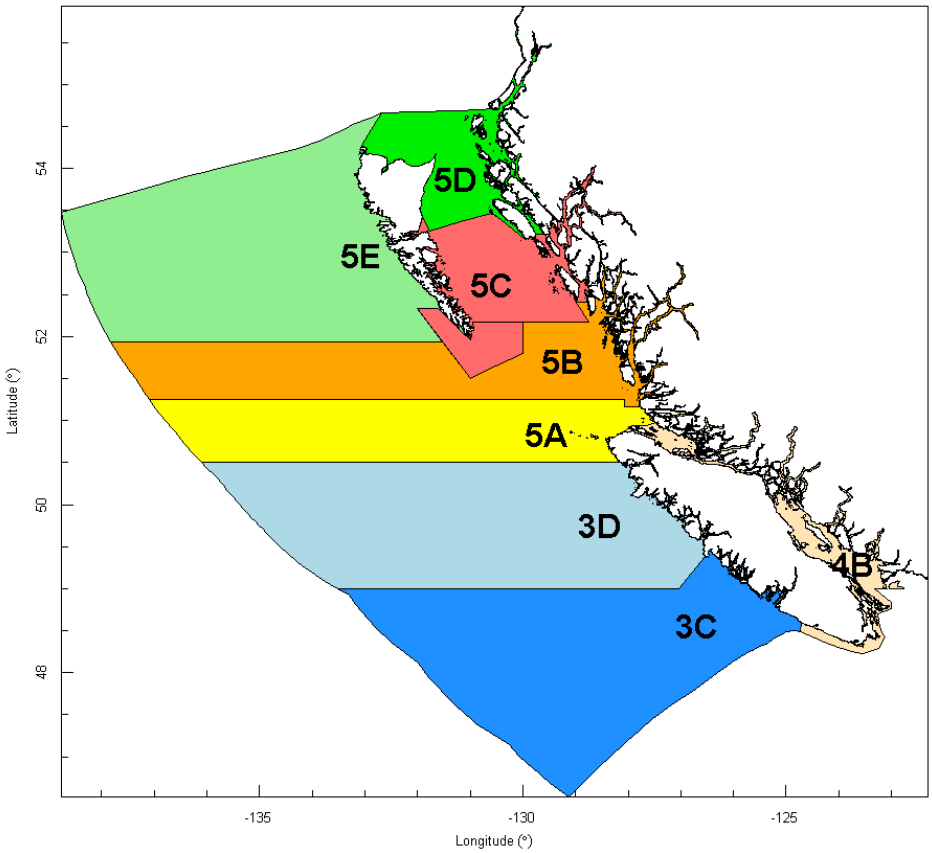

Figure 34. (*PBStools-plotGMA*) Groundfish Management Areas for Pacific Ocean Perch and Yellowmouth Rockfish.

5.8 plotTernary

Plot a ternary diagram using compositional data classified into three categories. When population components are amalgamated into $g = 3$ groups, vectors of proportions can be portrayed in a graph called a ternary diagram (Aitchison, 1986, p. 5). The diagram begins with an equilateral triangle that has vertices labelled "1", "2", and "3". A vector $\mathbf{p}$ (p_1, p_2, p_3) of proportions can then be represented as a point within this triangle, where the perpendicular distance to the side opposite vertex "i" is proportional to p_i.

Example (plotTernary):

```
pbsfun=function(){
  plotTernary(c(1/2,1/3,1/6))
  invisible() }
pbsfun()
```

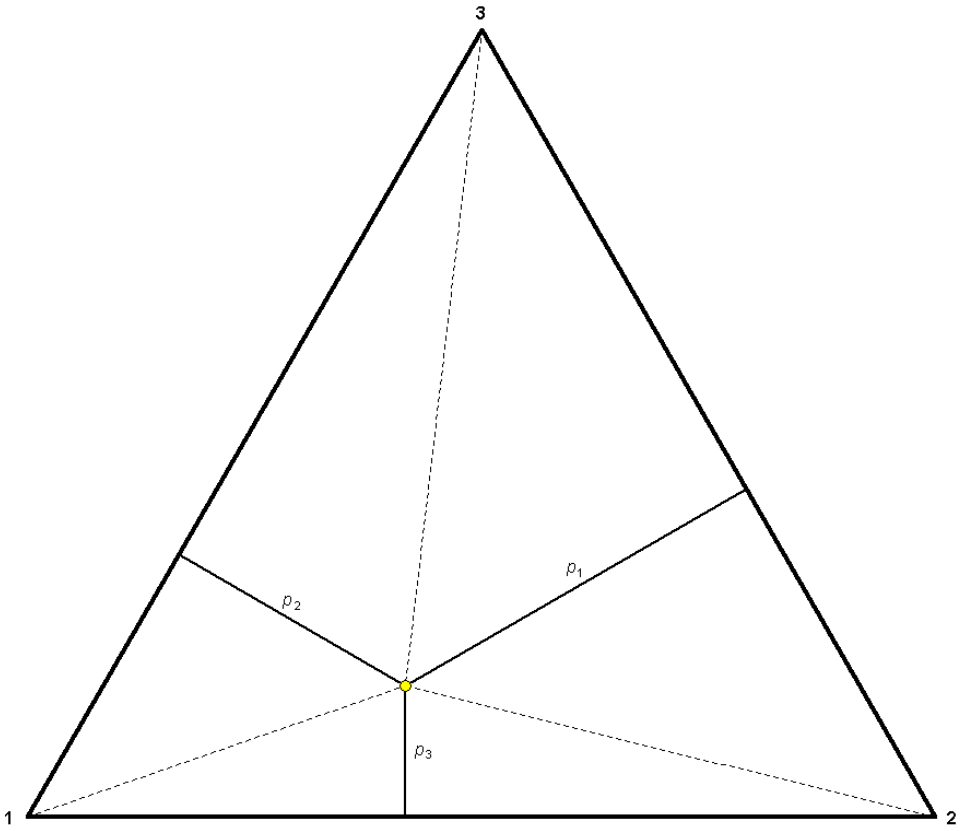

Figure 35. (*PBStools-plotTernary*) Ternary diagram for compositions amalgamated into $g = 3$ groups. The indicated point represents a vector of proportions. Solid lines perpendicular to the sides of an equilateral triangle have lengths proportional to $p_i (i = 1, 2, 3)$. Dotted lines facilitate a proof that this method actually works, given the constraint $\Sigma_{i=1}^{3} p_i = 1$ (Schnute and Haigh, 2007, Fig. 2).

5.9 plotTertiary

Plot a compositional diagram using data classified into n categories. This function seeks to extend the ternary diagram from 3 to n proportions. However, the geometry of mutiple vertices (squares, pentagons, hexagons, etc.) does not translate into deterministic solutions as it does for triangles (Aitchison, 1986; Schnute and Haigh, 2007). The algorithm currently yields a compromise solution because the constraint $\Sigma_{i=1}^{n} p_i = 1$ rarely yields a common point within polygons that are not equilateral triangles. The problem still needs more thought.

When only one proportion set (vector or single-row matrix) is supplied, the function

shows lines that connect the mode to the vertices and to each side of the polygon. When multiple proportion sets are supplied, the modes are connected together to form a trace diagram.

Example (plotTertiary):

```
pbsfun=function(){
  plotTertiary(c(1,2,2,5,5))
  invisible() }
pbsfun()
```

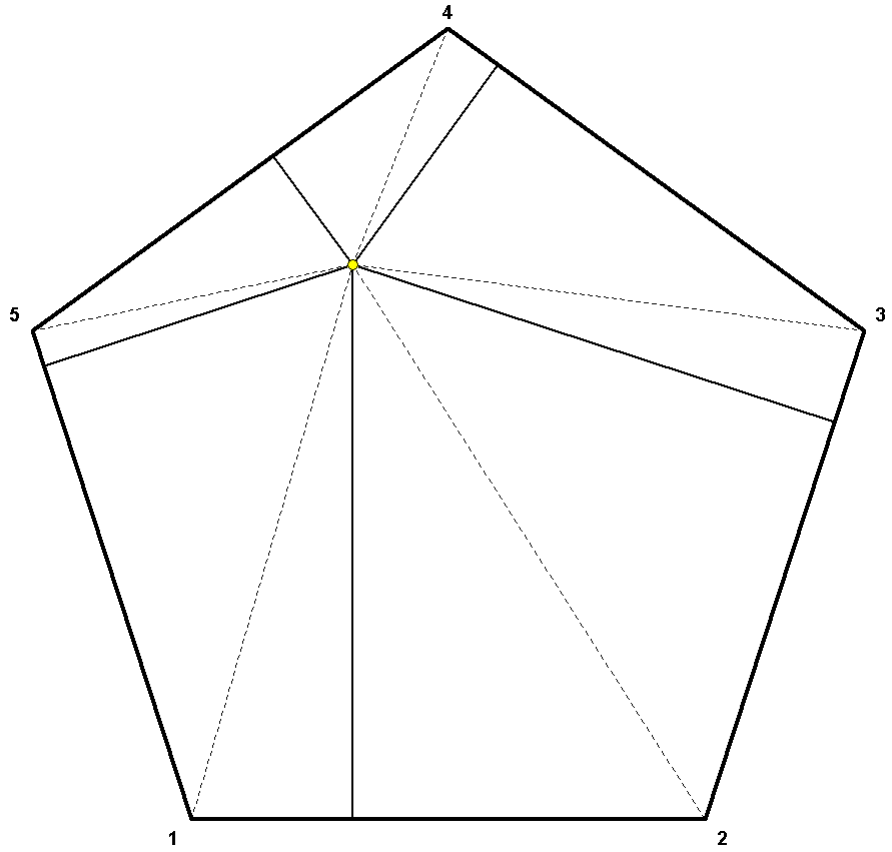

Figure 36. (*PBStools-plotTertiary*) Tertiary diagram for compositions amalgamated into $g = 5$ groups. The indicated point represents a vector of proportions. Solid lines perpendicular to the opposite sides of a polygon have lengths roughly proportional to $p_i (i = 1, ..., n)$.

5.10 preferDepth

This function displays the depth distribution for a selected species using an interactive GUI (Figure 37). The SQL code `pht_fdep.sql` grabs the depth table from the remote SQL Server database PacHarvest and the Oracle database GFFOS. The user may also specify an R object or a data file. The menu control offers some ability to specify years and/or areas (if available in the file). The depth-of-capture frequency for the selected species appears as a histogram (Figure 38). The cumulative catch is superimposed to show the depths at which the species is caught in relation to the tow frequency histogram. The

total fleet effort is displayed as a shaded histogram in the background.

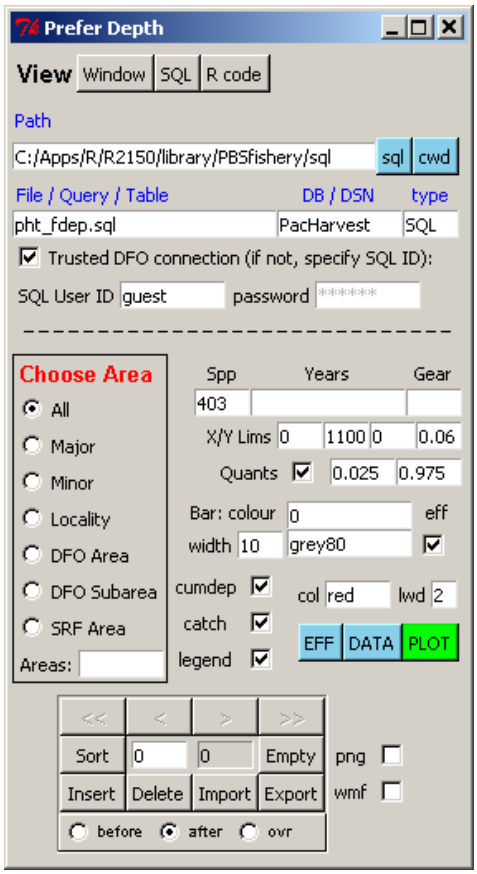

Figure 37. (*GUI-preferDepth*) GUI menu system created by a call to `preferDepth`. The SQL queries will only work for users on the DFO network. Alternatively, a data object or file can be specified in the File box and the type must be set to `FILE`.

5.11 prepClara

Prepare a data object for use by `clarify`, which uses the **cluster** function `clara`. The grouping variable `gfld` will determine how event data are collapsed into spatial units. The Lon-Lat default is likely equivalent to a sinlge field identifying each fishing event unless events happen to occupy the same Lon-Lat coordinate. A user may wish to create a field that delimits grid cells, but these should be the same as or smaller than those specified in the function `clarify`.

The function assumes that the input data object is always called 'dat', but a source file can be called anything as long as its import to R creates `dat`. Similarly, the output data object is always called 'claradat' but can be saved to a system file with any name. The function automatically names the output file containing `claradat` depending on whether an input name `fnam` was specified or SQl code was executed to get the input data.

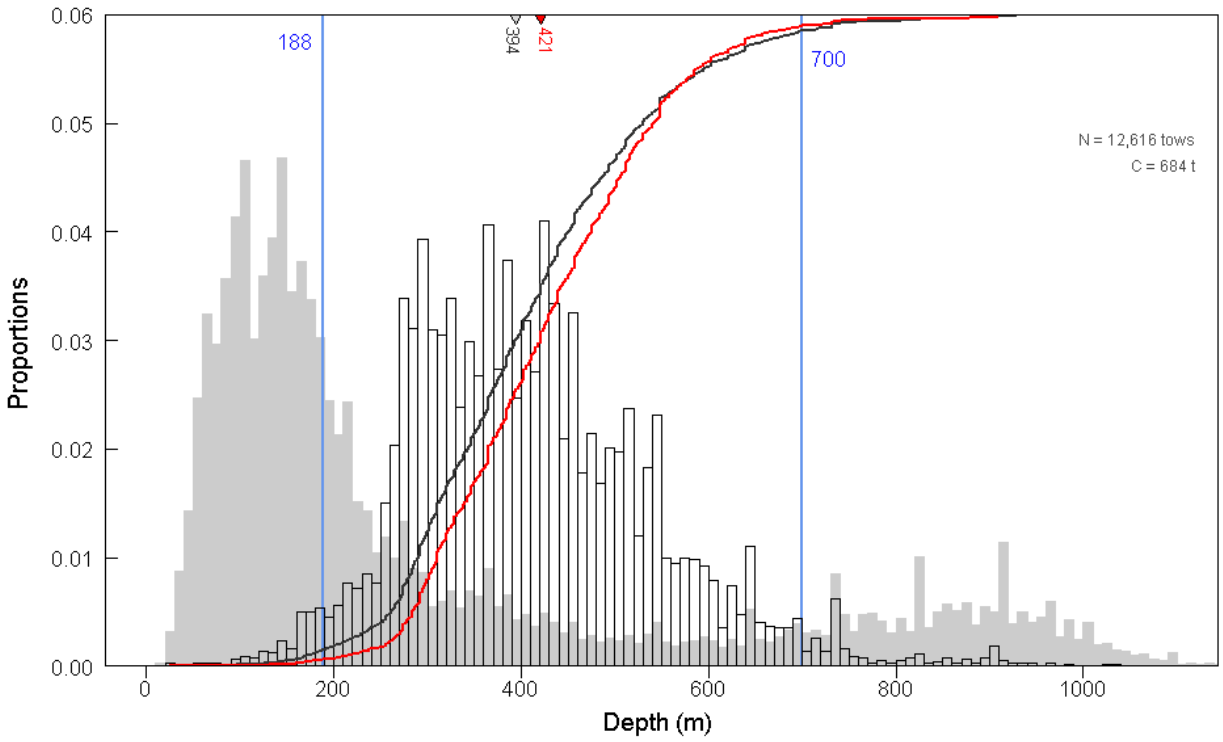

Figure 38. (*PBStools-preferDepth*) Relative frequency of tows in 10-m depth bins that capture Shortraker Rockfish (SKR, *Sebastes borealis*) from commercial trawl records (1996–2007). The vertical solid lines denote the 2.5% and 97.5% quantiles. The shaded histogram in the background reports the relative trawl effort on all species. The cumulative catch of SKR, superimposed on the histogram in relative space (0 to 1), provides confirmation that the bulk of SKR comes from these depths. The depth of median cumulative catch is indicated by an inverted red triangle at top. 'N' reports the total number of tows; 'C' reports the total catch (t).

5.12 zapHoles

The function attempts to remove (zap) holes that will ultimately be filled anyway. It uses centroid matching between holes and solids to identify candidate holes for removal. The value returned is a `PolySet` potentially modified to remove holes that will be filled. Non-essential attributes will be retained and supplemented with two additional ones created by the **PBSmapping** functions `calcCentroid` and `calcArea`: (i) `keep` – data frame summarizing the kept polygons, and (ii) `zap` – data frame summarizing the holes removed.

The original rationale for this function is to groom a `PolySet` before using in `addPolys(pset,...,colHoles="white")`. The `colHoles` option was designed to get rid of retrace lines that appear in `.pdf` files made from metafiles.

That said, the `PolySet` created by this function essentially comprises layers. To see all the layers, the user must add them sequentially from largest to smallest, which is not terribly efficient.

Example (zapHoles):

```
pbsfun=function(){
  data(qcb); qcbz=zapHoles(qcb)
  pdata=attributes(qcb)$PolyData
  pdata=pdata[rev(order(pdata$area)),]
  expandGraph(mfrow=c(1,1),plt=c(.05,1,.05,1))
  plotMap(qcbz)
  for (i in 1:nrow(pdata)) {
    pid=pdata$PID[i]
    qcbi=qcbz[is.element(qcbz$PID,pid),]
    if (nrow(qcbi)==0) next
    addPolys(qcbi,col=pdata$col[i]) }
  invisible() }
pbsfun()
```

6 Temporal functions

Temporal functions help scientists see how populations change over time. Here we present a few that visualize changes in species composition and diversity for phytoplankton samples. The functions could be applied equally to any taxonomic collection. We often think of such collections as marine species portfolios. Monitoring the diversity of species portfolios can alert us to changes in an ecosystem's resilience. Haigh and Schnute (2003) explored diversity patterns for fish species composition of commercial catch tows in the longspine thornyhead (*Sebastolobus altivelis*) fishery.

6.1 boxSeason

Summarize time-dependent data using boxplots to group by a specified period pattern (Figure 39). The data file imported from a Microsoft Access table or query called `fqtName` must contain a field called either `YMD` or `Date` that depicts the date values as strings with format `YYYY-MM-DD`.

Example (boxSeason):

```
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows")
    boxSeason(fld="S",brks="M",path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
```

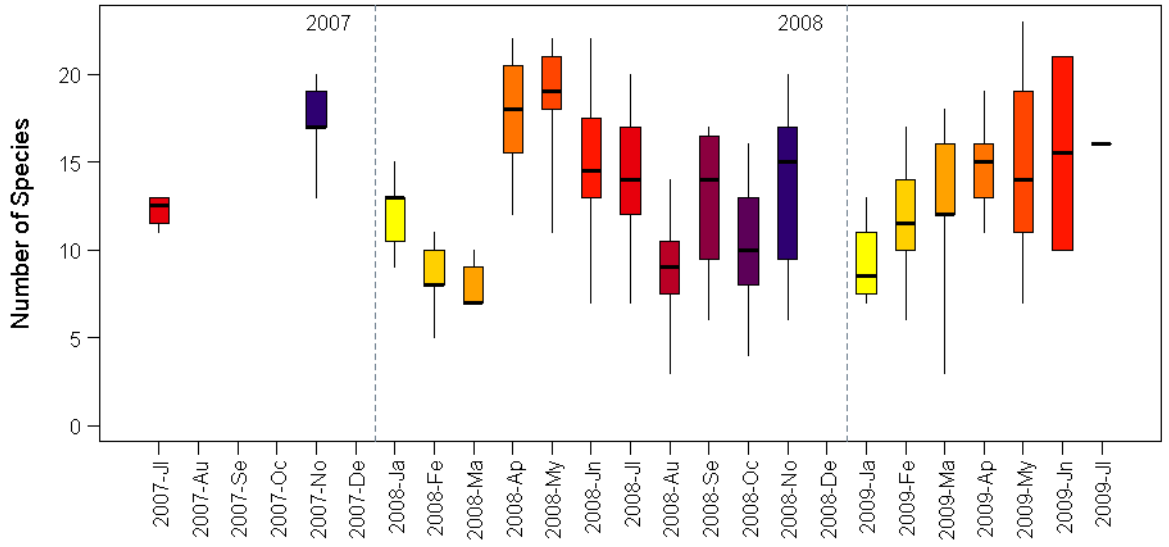

Figure 39. (*PBStools-boxSeason*) Number of phytoplankton species observed in samples grouped by month.

6.2 calcMA

Calculate a moving average of a series using a fixed period occurring every number of specified base units (usually days). The moving average is accumulated backwards from

the last observation.

Example (calcMA):

```
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows") {
    getData("Ex03_Portfolio","Examples",type="MDB",path=.getSpath())
    # Annual average every 2 months:
    ma=calcMA(PBSdat[[2]],PBSdat[[3]],period=180,every=30)
    dat=attributes(ma)$dat
    plot(dat$x,dat$y,type="l")
    lines(ma$x,ma$y,col="blue",lwd=2) }
  invisible() }
pbsfun()
```

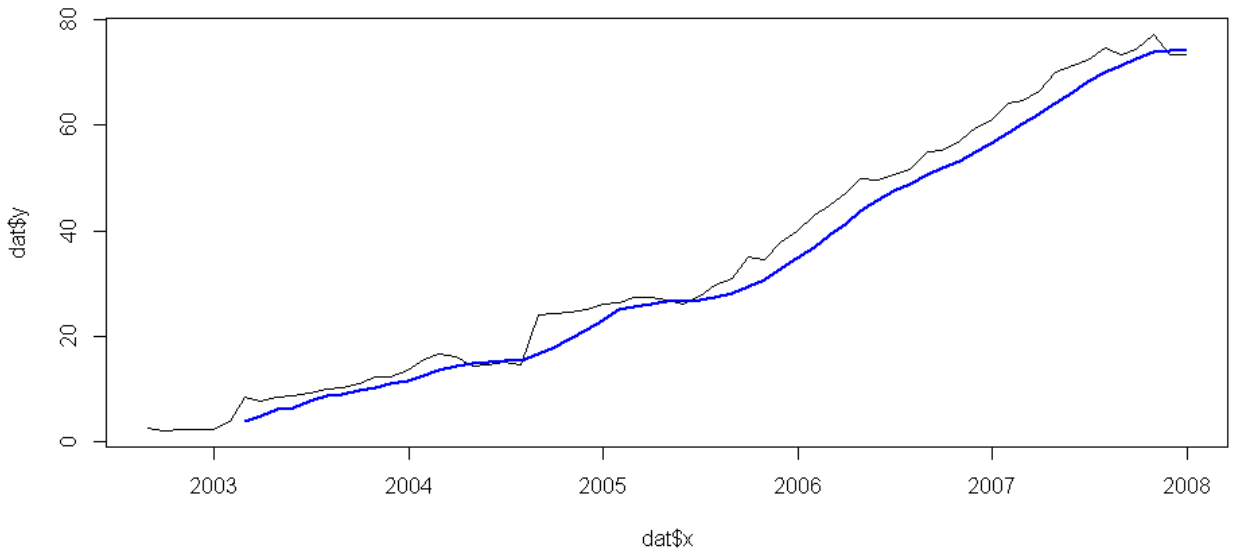

Figure 40. (*PBStools-calcMA*) Semi-annual moving average calculated every 30 days.

6.3 imputeRate

Impute the rate of return for an investment (existing account or simulated) that experiences regular or irregular contributions and/or withdrawals. This function creates an interactive GUI (Figure 41) to facilitate the process. The code adopts a variant of a formula in Microsoft Excel's XIRR function where the "internal rate of return" r is derived iteratively when the net present value (NPV) is set to 0.

$$\text{NPV} = \sum_{i=1}^{N} \frac{C_i}{(1-r)^{(\frac{d_i - d_1}{365})}} \tag{11}$$

where:

d_i = the i^{th} contribution date;
d_1 = the initial date;
C_i = the i^{th} contribution; and

C_1 = initial value of the portfolio (includes 1st contribution).

Here we use the **PBSmodelling** function `calcMin` to minimize the squared difference between the calculated NPV of the contribution stream and the final value of the portfolio (Figure 42). User's can either specify an MDB table in the current working directory or `Examples.mdb` in the package's `sql` directory. There is also an option to simulate a dataset using the random pareto distribution.

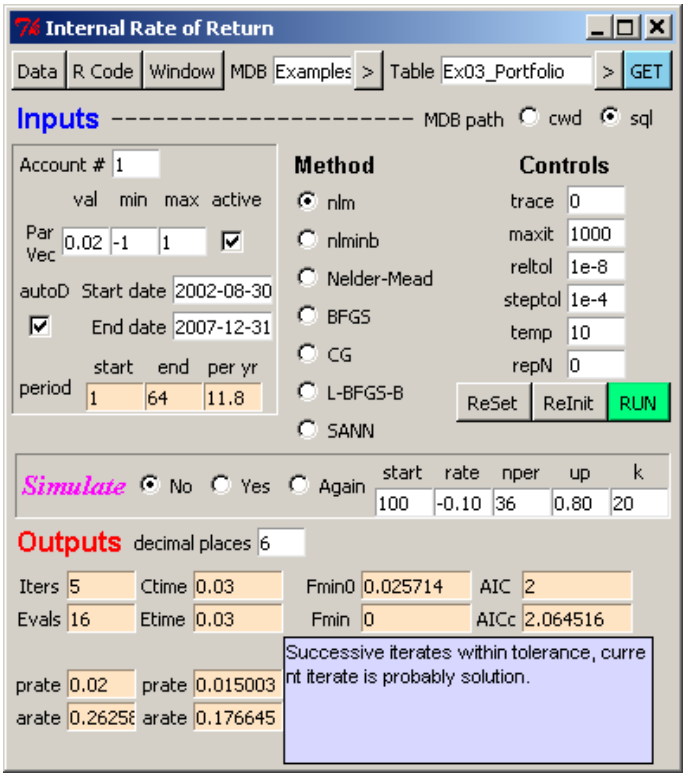

Figure 41. (*GUI-imputeRate*) GUI menu Internal Rate of Return to estimate the annualized return on an investment that experiences episodic contributions.

6.4 plotDiversity

Plot sample diversity (e.g., Shanon-Wiener diversity index H) as a barplot with an overlay of points using some other index (e.g., species richness S). The arguments `bars` and `pnts` can each be one of `H` = Shannon-Diversity index, `S` = species richness, or `E` = species evenness. The user can also choose to add lowess-fitted lines through the bars and the points (Figure 43).

Example (plotDiversity):

```
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows")
    plotDiversity(bars="H",pnts="S",xnames="Batch",path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
```

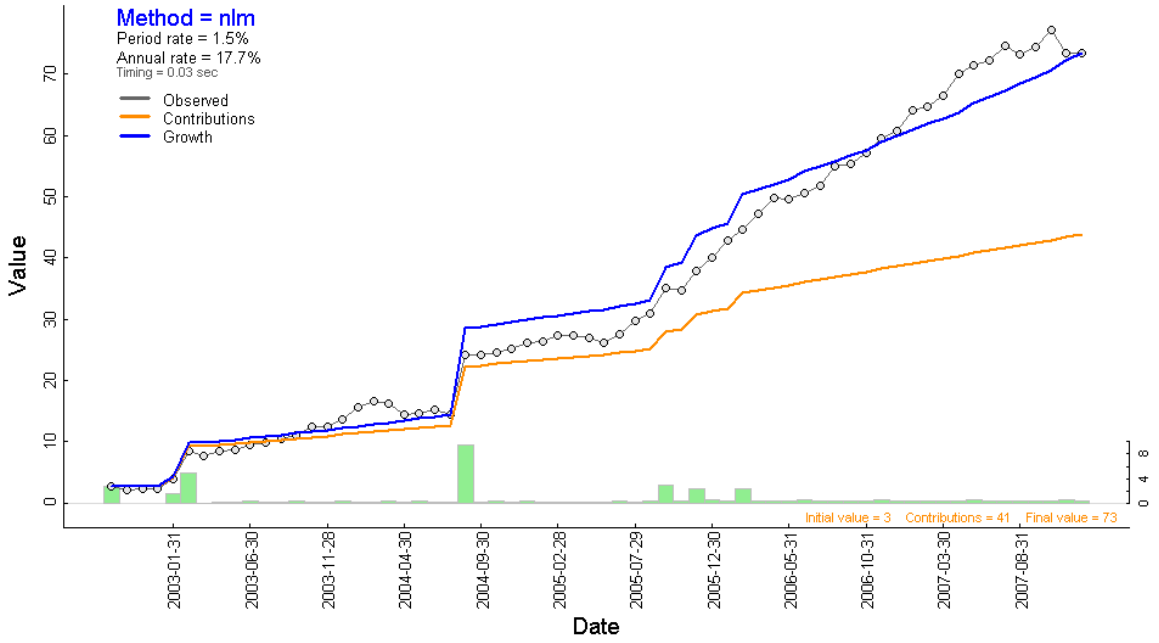

Figure 42. (*PBStools-imputeRate*) Portfolio progress where green bars represent episodic contributions, the orange line shows the cumulative contributions, the grey dots report the monthly portfolio values, and the blue line shows a calculated fit using the estimated internal rate of return.

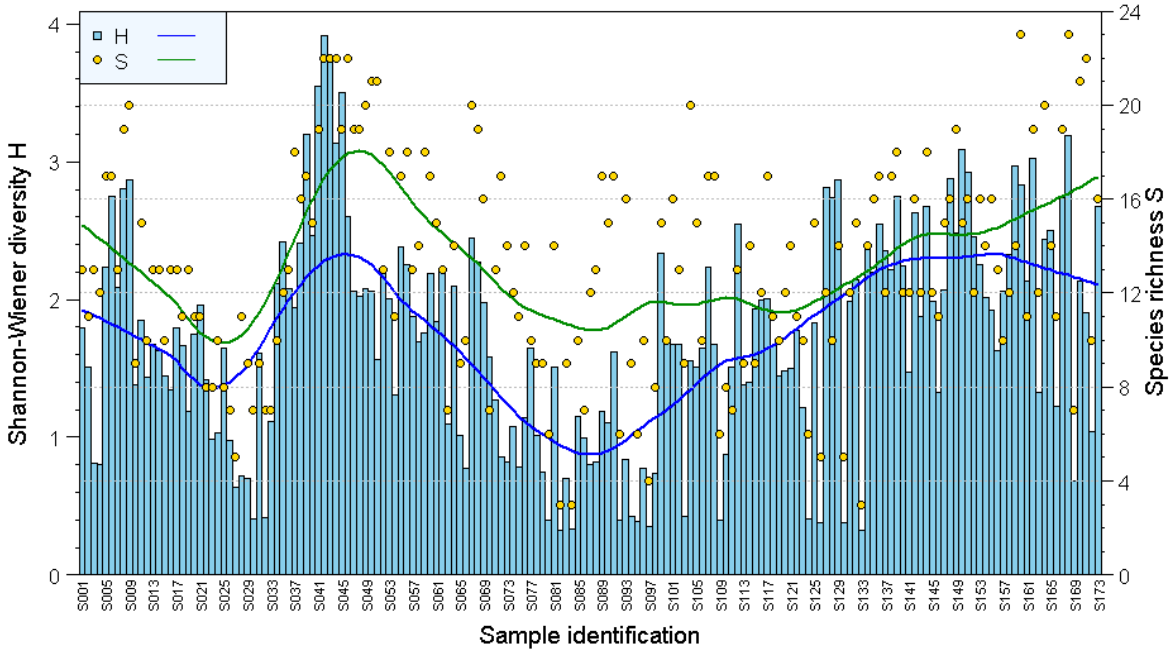

Figure 43. (*PBStools-plotDiversity*) Shannon-Wiener diversity index H (bars) for phytoplankton samples. Species richness S (numbers of species observed) appear as points. Smooth trend lines through H and S are calculated using the **stats** function `lowess`, which uses a locally-weighted polynomial regression.

6.5 trackComp

Track the mean composition of phytoplankton groups over time using a specified binning interval. The function creates a stacked line plot of relative composition frequencies for each group (Figure 44).

Example (trackComp):

```
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows") trackComp(ndays=15,path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
```

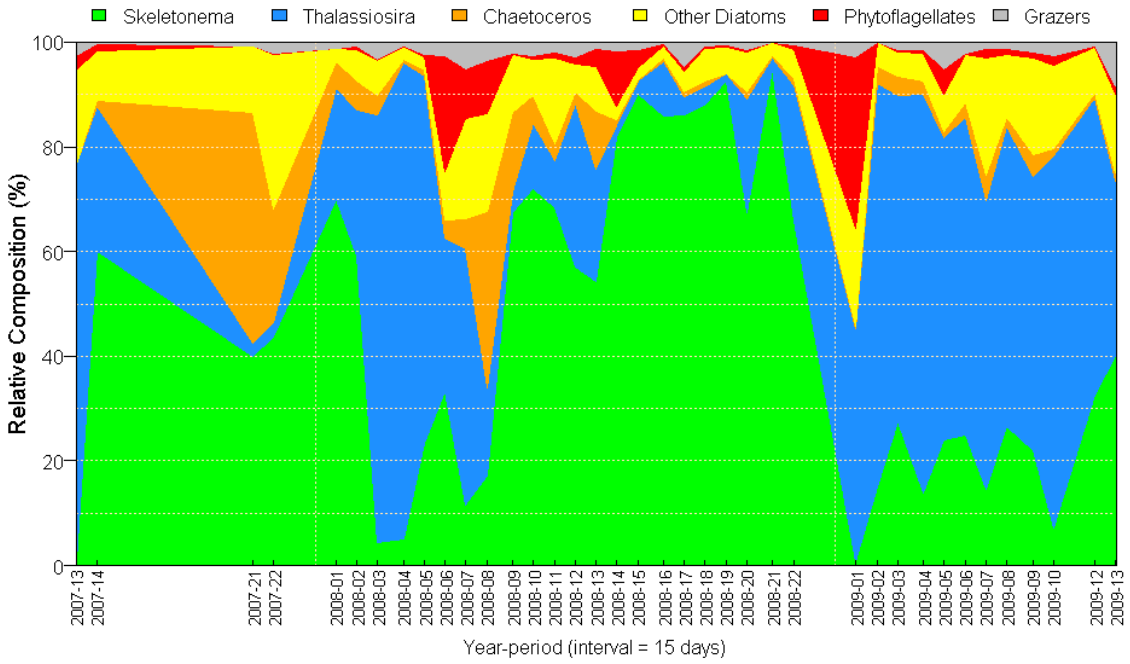

Figure 44. (*PBStools-trackComp*) Relative composition frequency of major phytoplankton groups in samples over time. The binning interval spans 15 days.

7 Catch Reconstruction

7.1 buildCatch

The function `buildCatch` offers an automated algorithm for reconstructing rockfish (RRF) catches in British Columbia (BC); full details are given in Haigh and Yamanaka (2011). The catch reconstruction uses: (i) reliable landings from modern databases, (ii) estimated landings using ratios of RRF catch to other rockfish (ORF) or total rockfish (TRF) catch, (iii) estimated discards using discard ratios from observer logs.

The reconstruction algorithm estimates annual removals (landed catch + discards) in metric tonnes (t) of specified rockfish, by:
(i) calendar year,
(ii) Pacific Marine Fisheries Commission (PMFC[11]) area:
4B = (major 1) inside waters including Strait of Georgia,
3C = (major 3) SW coast of Vancouver Island,
3D = (major 4) NW coast of Vancouver Island,
5A = (major 5) southern Queen Charlotte Sound,
5B = (major 6) northern Queen Charlotte Sound,
5C = (major 7) southern Hecate Strait,
5D = (major 8) northern Hecate Strait and Dixon Entrance,
5E = (major 9) west coast Haida Gwaii (formerly Queen Charlotte Islands); and
(iii) fishery type (FID):
1 = groundfish trawl,
2 = Halibut longline,
3 = Sablefish trap/longline,
4 = schedule II comprising primarily Dogfish and Lingcod, and
5 = hook and line (H&L) rockfish, eventually regulated under ZN licensing.

Because historical catch data do not provide landings of individual rockfish species, we use broad rockfish categories (POP = Pacific Ocean Perch, ORF = rockfish other than POP, and TRF = total rockfish) and apply ratios that we calculate from modern catches (e.g., POP/TRF for Pacific Ocean Perch (*Sebastes alutus*) or YYR/ORF for Yelloweye Rockfish (*S. ruberrimus*)) to disaggregate historical catches. This assumes that modern ratios reflect historical ones. Many factors may invalidate this assumption – selective fishing, environmental shifts, habitat destruction; however, we have little choice if we want historical catch estimates for specific rockfish species.

The reconstruction ratios calculated by `buildCatch`:
α proportion of RRF caught in a PMFC area for each fishery,
β proportion of RRF caught in a PMFC area for Hook and Line (H&L) fisheries,
γ ratio of RRF to a larger group like ORF or TRF,
δ discard rate of RRF per other catch category from observer logs, and
λ proportion of early ORF or TRF catch by general gear type
are applied to the historical data to estimate historical landings of the RRF species. For

[11] see footnote in Forrester (1969)

POP the historical trawl landings are known back to 1956 and so these are used instead of deriving estimates.

Most sources of historical landings provide data that overlap in some years. Two of the sources provide unique and consequently additive information, regardless of the year. The first additive series is the very early landed catch from 1918 to 1950. It reflects a basal low exploitation rate that occurred prior to World War II. The second additive series is that of the foreign (Russian and Japanese) fleets that fished heavily along the BC coast from 1965 to 1976 until the inception of the 200-mile exclusive economic zone (EEZ). All other historical series may contain redundant information where they overlap, and so we take the maximum annual value from these.

For the early time period 1918–1949, landings of ORF and TRF are identical because catch agencies did not identify Pacific Ocean Perch (POP). POP started showing up in catch records in 1950, but appear artificially low from 1950–1952. Therefore, for the period 1918–1952 we predict ORF and POP using a linear regression of ORF *vs.* TRF landings from the 1953–1995 data (excluding an anomalous 1966 point):

$$O = 2^a T^b \tag{12}$$

where a = -0.8063372, b = 1.006260, O = ORF, T = TRF, and POP = TRF - ORF.

Landings (metric tonnes) from eight modern catch databases (GFCatch, PacHarvest, PacHarvHL, PacHarvSable, PacHarvHL, PacHarvHL, and GFFOS) are summarized in a 5-dimensional array where i = calendar year, j = PMFC major area code, k = fishery ID, l = species or catch group, and m = the database. Then for each element (i, j, k, l), the maximum landing across database m is taken. This methodology is utilized due to the non-contiguous nature of the data sources. This also means that compiling individual catch records across the database sources, as performed for the trawl fishery by `getCatch`, is too complicated for the hook and line fisheries. It is possible if definite cut points can be identified for the various time series (which we no longer do for the reconstruction).

The reconstruction executes a complex series of rules using SQL (Structured Query Language) queries (shown in the DFO boxes below) combined with R code (R Development Core Team, 2012) to derive an annual catch series for five fisheries (Trawl, Halibut, Sablefish, Dogfish-Lingcod, H&L (hook and line) Rockfish) in BC's eight PMFC areas (4B, 3C, 3D, 5A, 5B, 5C, 5D, 5E). This yields 40 annual series (48 when fisheries are combined).

DFO:

PacHarv3 catch summary for fishery IDs 1:5 and 0 (unknown):

```
getData("ph3_fcatORF.sql",dbName="HARVEST_V2_0",strSpp=strSpp,path=.getSpath(),
   server="ORAPROD",type="ORA",trusted=FALSE,uid=uid[1],pwd=pwd[1])
   assign("ph3dat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   dimnames(ph3dat)[[1]]=1:nrow(ph3dat)
   save("ph3dat",file="ph3dat.rda")
```

GFCatch records for fishery IDs (1,3,5):

```
getData("gfc_fcatORF.sql","GFCatch",strSpp=strSpp,path=.getSpath())
   assign("gfcdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   gfcdat$year=as.numeric(substring(gfcdat$date,1,4))
   dimnames(gfcdat)[[1]]=1:nrow(gfcdat)
   save("gfcdat",file="gfcdat.rda")
```

PacHarvest records for fishery ID (1):

```
getData("pht_tcatORF.sql","PacHarvest",strSpp=strSpp,path=.getSpath())
   assign("phtdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   save("phtdat",file="phtdat.rda")
```

PacHarvHL halibut validation records for fishery IDs (2,7):

```
getData("phhl_hcatORF.sql","PacHarvHL",strSpp=strSpp,path=.getSpath()) ### validated DMP
   assign("phhdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   save("phhdat",file="phhdat.rda")
```

PacHarvSable fisherlogs for fishery ID (3):

```
getData("phs_scatORF.sql","PacHarvSable",strSpp=strSpp,path=.getSpath(),
   fisheryid=3,logtype="FISHERLOG")
   assign("phsdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   save("phsdat",file="phsdat.rda")
```

PacHarvHL validation records for fishery IDs (2,4,5):

```
getData("phhl_vcatORF.sql","PacHarvHL",strSpp=strSpp,path=.getSpath()) ### validated DMP
   assign("phvdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   save("phvdat",file="phvdat.rda")
```

PacHarvHL fisherlog records for fishery IDs (4,5):

```
getData("phhl_fcatORF.sql","PacHarvHL",strSpp=strSpp,path=.getSpath(),
   logtype="FISHERLOG") ### fisherlog catch
   assign("phfdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   save("phfdat",file="phfdat.rda")
```

GFFOS catch from all fishery IDs (1:5):

```
getData("fos_vcatORF.sql",dbName="GFFOS",strSpp=strSpp,path=.getSpath(),
   server="GFSH",type="ORA",trusted=FALSE,uid=uid[2],pwd=pwd[2])
   assign("fosdat",PBSdat); rm(PBSdat,envir=.GlobalEnv) ### just to be safe
   dimnames(fosdat)[[1]]=1:nrow(fosdat)
   save("fosdat",file="fosdat.rda")
```

The final reconstructed catch array is invisibly returned. Additionally, if `saveinfo = TRUE`, various data objects internal to the function are saved to a list object called `PBStool`

(located in the environment `.PBStoolEnv`). Also, various data objects, including the eight database catch files consolidated and massaged, are saved as system binary files `.rda`. The final reconstructed catch array is also flattened to a comma-delimited text file `Catch-History-123.csv`, where `123` represents the species code. Summary catch barplots, specified by the argument `fidout`, are plotted on the R graphics device or sent to `wmf` file(s) (e.g., Figure 45).

The function only has one argument for an Oracle database user ID and one for its password. If the user's authentication credentials are different for PacHarv3 and GFFOS, specify `uid` and `pwd` as two-element vectors. The SQL servers assume a trusted relationship with the user's DFO logon identity.

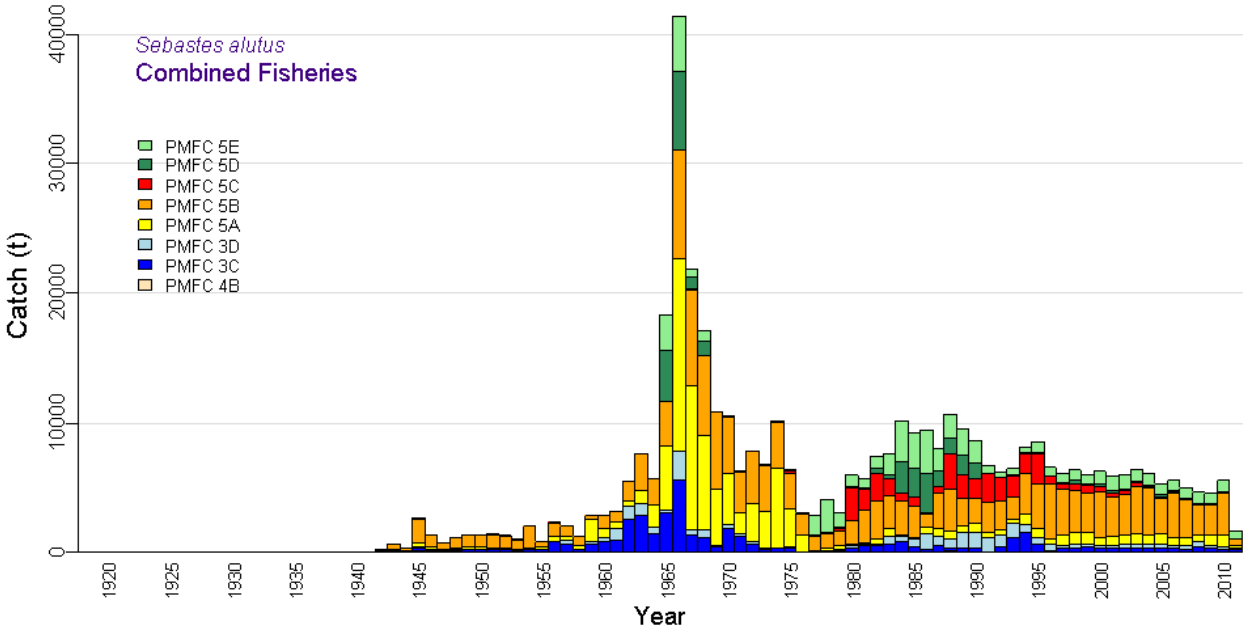

Figure 45. (*PBStools-buildCatch*) Reconstructed total (landed + discarded) catch (t) for Pacific Ocean Perch from all fisheries combined in PMFC major areas.

7.1.1 Argument details

`dbdat` — a list object of landing records from eight DFO databases:
(1) `ph3dat` = PacHarv3 (all fisheries) actually called HARVEST_V2_0 on the PROD Oracle server (host= `vsbciosxd76.ent.dfo-mpo.ca`,
(2) `gfcdat` = GFCatch (trawl, trap, h&l) on the SVBCPBSGFIIS SQL server,
(3) `phtdat` = PacHarvest (groundfish trawl) on the SVBCPBSGFIIS SQL server,
(4) `phhdat` = PacHarvHL (Halibut bycatch from DMP validated landings) on the SVBCPBSGFIIS SQL server,
(5) `phs dat` = PacHarvSable (Sablefish trap) on the SVBCPBSGFIIS SQL server,
(6) `phhdat` = PacHarvHL (validated landings Sched II & ZN) on the SVBCPBSGFIIS SQL server,

	(7) `phfdat` = PacHarvHL (fisherlog records Sched II & ZN) on the SVBCPBSGFIIS SQL server, (8) `fosdat` = GFFOS (all fisheries) on the GFSH Oracle server.
`sql`	logical: if `TRUE` query the databases, otherwise load catch records from binary files saved from a previous query run (saves time).
`strSpp`	character string specifying the Hart species code for the rockfish to be reconstructed (`RRF`).
`dfld`	field name of the denominator in the ratio of RRF to other rockfish (usually `ORF` but can be `TRF` if total rockfish is more appropriate).
`major`	major PMFC area codes in which catch is reconstructed (usually `c(1,3:9)`).
`refyrs`	reference years to use for ratio calculations.
`fidout`	fishery IDs for which an annual series barplot stacked by PMFC area is produced: `1` = groundfish trawl, `2` = Halibut longline, `3` = Sablefish trap, `4` = Dogfish-Lingcod, `5` = hook and line (H&L) rockfish, `10` = all fisheries combined.
`saveinfo`	logical: if `TRUE`, save various data objects created within the function to a list object called `PBStool`; setting to `FALSE` may speed reconstruction.
`wmf`	logical: if `TRUE` send the figures to `.wmf` files.
`uid, pwd`	user ID and password for Oracle DB account authentication.
`reconstruct`	logical: if `TRUE` (default) complete the reconstruction as previously; if `FALSE`, terminate the code once the modern catch array has been compiled and saved (to a binary file `cat123mod.rda`, where `123` = species code).

7.2 plotData

Plot data components of a catch reconstruction for user inspection and diagnostic checking. The function plots `z`-values of `y` for each `x`. The default plot shows lines because a typical `x` comprises years. An alternative is to show the `z`-values as bars for each `y` grouped by `x`.

This plot is used by the catch reconstruction algorithm `buildCatch` which automatically increments the plot number `pD` that is logged in the temporary list object `PBStool`. Additionally, the catch reconstruction creates a subdirectory `CRdiag` to which `plotData` sends image files.

7.3 plotRecon

Plot reconstructed catch using barplots stacked by catch in PMFC areas. This function allows greater control over the plotting details (see Figure 45) than is currently

offered by the `buildCatch` routine. Ideally, the latter should call `plotRecon`, but the implementation has not occurred yet.

Contact Information

Rowan Haigh
Research Biologist, Groundfish Section
Marine Ecosystems and Aquaculture Division
Fisheries and Oceans Canada
Pacific Biological Station, Nanaimo, BC V9T 6N7
Tel. +1 250-756-7123 fac simile +1 250-756-7053
Email rowan.haigh@dfo-mpo.gc.ca

References

Aitchison J (1986). *The Statistical Analysis of Compositional Data*. The Blackburn Press, Caldwell, NJ.

Barrie V, Bornhold BD, Conway KW, Luternauer JL (1991). "Surficial geology of the northwestern Canadian continental shelf." *Continental Shelf Research*, **19S(8–10)**, 701–715.

Boers NM, Haigh R, Schnute JT (2004). "PBS Mapping 2: developer's guide." *Canadian Technical Report of Fisheries and Aquatic Sciences*, **2550**, iv + 38. URL http://www.dfo-mpo.gc.ca/Library/285693.pdf.

Edwards AM, Starr PJ, Haigh R (2012). "Stock assessment for Pacific ocean perch (*Sebastes alutus*) in Queen Charlotte Sound, British Columbia." *Canadian Science Advisory Secretariat, Research Document*, **2011/111**, viii + 172. URL http://www.dfo-mpo.gc.ca/csas-sccs/Publications/ResDocs-DocRech/2011/2011_111-eng.html.

Forrester CR (1969). "Results of English Sole tagging in British Columbia waters." *Bulletin of Pacific Marine Fisheries Commission*, **7**, 1–10.

Haigh R, Schnute JT (2003). "The longspine thornyhead fishery along the west coast of Vancouver Island, British Columbia, Canada: portrait of a developing fishery." *North American Journal of Fisheries Management*, **23**, 120–140.

Haigh R, Yamanaka KL (2011). "Catch history reconstruction for rockfish (*Sebastes* spp.) caught in British Columbia coastal waters." *Canadian Technical Report of Fisheries and Aquatic Sciences*, **2943**, viii + 124 p. URL http://www.dfo-mpo.gc.ca/Library/344242.pdf.

Kaufman L, Rousseeuw PJ (1990). "Finding Groups in Data: An Introduction to Cluster Analysis."

Obradovich SG, Yamanaka KL, Cooke K, Lacko LC, Dykstra C (2008). "Summary of non-halibut catch from the Standardized Stock Assessment Survey conducted by the International Pacific Halibut Commission in British Columbia from June 4 to July 7, 2007." *Canadian Technical Report of Fisheries and Aquatic Sciences*, **2807**, x + 83. URL http://www.dfo-mpo.gc.ca/Library/335613.pdf.

Orr JW, Hawkins S (2008). "Species of the rougheye rockfish complex: resurrection of *Sebastes melanostictus* (Matsubara, 1934) and a redescription of *Sebastes aleutianus* (Jordan and Evermann, 1898) (Teleostei: Scorpaeniformes)." *Fisheries Bulletin*, **106**, 111–134.

R Development Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Schnute JT (1981). "A versatile growth model with statistically stable parameters." *Canadian Journal of Fisheries and Aquatic Sciences*, **38**, 1128–1140.

Schnute JT, Boers NM, Haigh R (2004a). "PBS Mapping 2: user's guide." *Canadian Technical Report of Fisheries and Aquatic Sciences*, **2549**, viii + 126. URL http://www.dfo-mpo.gc.ca/Library/285683.pdf.

Schnute JT, Couture-Beil A, Haigh R (2006). "PBS Modelling 1: user's guide." *Canadian Technical Report of Fisheries and Aquatic Sciences*, **2674**, viii + 114. URL http://www.dfo-mpo.gc.ca/Library/326794.pdf.

Schnute JT, Haigh R (2003). "A simulation model for designing groundfish trawl surveys." *Canadian Journal of Fisheries and Aquatic Sciences*, **60**(6), 640–656. URL http://www.nrcresearchpress.com/doi/abs/10.1139/f03-041.

Schnute JT, Haigh R (2007). "Compositional analysis of catch curve data, with an application to *Sebastes maliger*." *ICES Journal of Marine Science*, **64**, 218–233.

Schnute JT, Haigh R, Krishka BA, Sinclair A, Starr PJ (2004b). "The British Columbia Longspine Thornyhead fishery: analysis of survey and commercial data (1996–2003)." *Canadian Science Advisory Secretariat, Research Document*, **2004/059**, iii + 75. URL http://www.dfo-mpo.gc.ca/csas/CSAS/publications/ResDocs-DocRech/2004/2004_059_E.htm.

Schnute JT, Haigh R, Krishka BA, Starr PJ (2001). "Pacific Ocean Perch assessment for the west coast of Canada in 2001." *Canadian Science Advisory Secretariat, Research Document*, **2001/138**, iv + 90. URL http://www.dfo-mpo.gc.ca/csas/Csas/publications/ResDocs-DocRech/2001/2001_138_e.htm.

Sinclair AF, Conway KW, Crawford WR (2005). "Associations between bathymetric, geologic and oceanographic features and the distribution of the British Columbia bottom trawl fishery." *The Spatial Dimension of Ecosystem Structure and Dynamics*, **ICES CM 2005/L:25**.

Package 'PBStools'

April 29, 2013

Version 1.24

Date 2013-04-29

Title Tools for Semi-Serious Marine Scientists et al.

Author Rowan Haigh, Jon T. Schnute, Norm Olsen, and Lisa Lacko

Maintainer Rowan Haigh <rowan.haigh@dfo-mpo.gc.ca>

Depends R (>= 3.0.0), PBSmapping, PBSmodelling, PBSdata, RODBC

Suggests boot, BRugs, cluster, MASS, reshape, sp, utils

Description Provides tools for exploring fisheries data. This package provides screening and summary tools for fisheries stock assessments,species at risk reports, and exploratory data analyses.

License GPL (>= 2)

URL http://code.google.com/p/pbs-tools/, http://code.google.com/p/pbs-data/, http://code.google.com/p/pbs-mapx/

R topics documented:

`biteData` *Subset Data Matrix/Frame Using a Vector Object*

Description

Subset a matrix or data frame using a vector object with the same name as one of the fields in the data matrix/frame.

Usage

```
biteData(dat, vec)
```

Arguments

`dat`	matrix or data frame with named fields (row names).
`vec`	vector object of values to match in the field of `dat` with the same name as `vec` (e.g., `major=c(3:9)`).

Details

The data matrix/frame is subset using the values in `vec` and the subset data matrix/frame is returned. If there are no fields with the name of `vec`, the data matrix/frame is returned unaltered.

You cannot supply an *ad hoc* vector directly in the argument; `vec` must be an existing object.

NO CAN DO: `biteData(swiss,Education=1:5)`

CAN DO: `Education=1:5; biteData(swiss,Education)`

Value

A subset of dat after qualifying by vec.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

chewData, getData, getFile

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  cat("Records in original swiss object by Education\n")
  print(sapply(split(swiss$Education,swiss$Education),length))
  Education=1:5; test=biteData(swiss,Education)
  cat("Records in swiss object subset where Education=1:5\n")
  print(sapply(split(test$Education,test$Education),length))
  invisible() }
pbsfun()
})
```

---

bootBG *Bootstrap Biomass Using Binomial-Gamma Population Parameters*

Description

Bootstrap stratified biomass estimates using random draws from the binomial-gamma distribution using strata population parameters supplied.

Usage

```
bootBG(dat=pop.pmr.qcss, K=100, S=100, R=500,
   SID=NULL, ci=seq(0.1,0.9,0.1),
   lims=c("emp","norm","basic","perc","bca"), allo=0, ioenv=.GlobalEnv)
```

Arguments

dat	data frame with fields:
	SID = survey ID number assigned by the GFBio team;
	h = strata ID (alpha and/or numeric);
	p = proportion of zero measurements;
	mu = mean of nonzero measurements;
	rho = CV of nonzero measurements;
	A = bottom area (sq.km) of each stratum;
	n = actual number of tows performed to derive (p,mu,rho);
	k = preference/weighting of new tows.

K	number of proposed survey tows.
S	number of simulations.
R	number of bootstrap replicates.

SID	survey ID to simulate and bootstrap.
ci	confidence intervals for simulation; e.g., ci=c(0.90,0.95) gives confidence limits (0.025,0.05,0.95,0.975)
lims	bootstrap confidence limit types to compute (see boot::boot.ci).
allo	tow allocation scheme: 0 = existing, 1 = optimal.
ioenv	input/output environment for function input data and output results.

Details

This function depends on the R package **boot**. The user supplies a data frame dat summarizing a survey's strata population parameters, as outlined in Schnute and Haigh (2003). These parameters are used to estimate the survey population using random draws from the binomial-gamma distribution.

Value

No value is explicitly returned by the function. A global list object PBStool provides results of the analysis:

dat	data frame: a subset from the user input that contains population parameters for each unique stratum.
bgtab	matrix: parameter values and moment estimates for each stratum.
dStab	matrix: sampled density estimates for each stratum.
BStab	matrix: sampled biomass estimates for each stratum.
bgsamp	list: final simulated set of densities sampled from the binomial-gamma distribution and listed by stratum.
Bboot	list: results of boot::boot() listed for each simulation.
Bqnt	array: bootstrap quantiles for each simulation and type of confidence interval calculation.
Best	vector: biomass estimate for each simulation.
Bobs	scalar: observed biomass estimate (moment calculation).
group	vector: stratum grouping showing allocation of K tows.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

getPMR, calcPMR, sampBG, showAlpha
boot: boot.ci

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(SID=1,S=50){
  bootBG(dat=pop.pmr.qcss,SID=SID,S=S)
  showAlpha()
  invisible() }
pbsfun()
})
```

boxSeason *Display Seasonal Patterns Using Boxplots*

Description

Summarize time-dependent data using boxplots to group by a specified period pattern.

Usage

```
boxSeason(fqtName="Ex01_Sample_Info", dbName="Examples.mdb",
   type="MDB", path=getwd(), fld="S", brks="M",
   pdf=FALSE, wmf=FALSE, ioenv=.GlobalEnv)
```

Arguments

fqtName	name of a file, query or table in the specified database dbName; (can also be a local data object if type="FILE".)
dbName	name of a database.
type	type of database (one of "XLS", "MDB", "SQL").
path	directory path to the database.
fld	field name in fqtName table that contains data to summarise.
brks	type of period breaks; choose one of one of "Q" (quarterly), "S" (seasonal), "B" (bimonthly), or "M" (monthly).
pdf	logical: if TRUE, send figure to a postscript document file .pdf.
wmf	logical: if TRUE, send figure to a windows metafile .wmf.
ioenv	input/output environment for function input data and output results.

Details

The data file imported from a Microsoft Access table or query called fqtName must contain a field called either YMD or Date that depicts the date values as strings with format YYYY-MM-DD.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

See Also

trackComp, plotDiversity, histMetric, imputeRate

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows")
    boxSeason(fld="S",brks="M",path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
})
```

buildCatch *Catch Reconstruction Algorithm for BC Rockfish*

Description

A catch reconstruction algorithm for BC rockfish that uses:
(i) reliable landings from modern databases,
(ii) estimated landings using ratios of species catch to ORF/TRF catch,
(iii) estimated discards using discard ratios from observer logs.

Usage

```
buildCatch(dbdat, sql=FALSE, strSpp="424", dfld="ORF",
   major=c(1,3:9), refyrs=1997:2005, fidout=c(1:5,10),
   saveinfo=TRUE, eps=FALSE, pix=FALSE, wmf=FALSE,
   uid=Sys.info()["user"], pwd=uid, only.sql=FALSE,
   reconstruct=TRUE, diagnostics=TRUE, ioenv=.GlobalEnv)
```

Arguments

dbdat	a list object of landing records from eight DFO databases: (1) ph3dat = PacHarv3 (all fisheries) actually called HARVEST_V2_0 on the ORAPROD Oracle server, (2) gfcdat = GFCatch (trawl, trap, h&l) on the SVBCPBSGFIIS SQL server, (3) phtdat = PacHarvest (groundfish trawl) on the SVBCPBSGFIIS SQL server, (4) phhdat = PacHarvHL (Halibut bycatch from DMP validated landings) on the SVBCPBSGFIIS SQL server, (5) phs dat = PacHarvSable (Sablefish trap) on the SVBCPBSGFIIS SQL server, (6) phhdat = PacHarvHL (validated landings Sched II & ZN) on the SVBCPBSGFIIS SQL server, (7) phfdat = PacHarvHL (fisherlog records Sched II & ZN) on the SVBCPBSGFIIS SQL server, (8) fosdat = GFFOS (all fisheries) on the GFSH Oracle server.
sql	logical: if TRUE query the databases, otherwise load catch records from binary files saved from a previous query run (saves time).
strSpp	character string specifying the Hart species code for the rockfish to be reconstructed ("RRF").
dfld	field name of the denominator in the ratio of RRF to other rockfish (usually "ORF" but can be "TRF" if total rockfish is more appropriate).
major	major PMFC area codes in which catch is reconstructed (usually c(1,3:9)).
refyrs	reference years to use for ratio calculations.
fidout	fishery IDs for which an annual series barplot stacked by PMFC area is produced: 1 = groundfish trawl, 2 = Halibut longline, 3 = Sablefish trap, 4 = Dogfish-Lingcod, 5 = hook and line (H&L) rockfish, 10 = all fisheries combined.
saveinfo	logical: if TRUE, save various data objects created within the function to a list object called PBStool; setting to FALSE may speed reconstruction.
eps	logical: if TRUE send the figures to .eps files.
pix	logical: if TRUE send the figures to .png files.

wmf	logical: if TRUE send the figures to .wmf files.
uid, pwd	user ID and password for Oracle DB account authentication.
only.sql	logical: if TRUE, then only execute the queries to download catch data from remote databases.
reconstruct	logical: if TRUE (default) complete the reconstruction as previously, if FALSE, terminate the code once the modern catch array has been compiled and saved (to a binary file cat123mod.rda, where 123 = species code).
diagnostics	logical: if TRUE (default) create automatically-numbered diagnostic images files (prefaced with pD000, where 000 is the automatic number) to a subdirectory called CRdiag. Subsequent calls to buildCatch remove all pD... files created by previous builds.
ioenv	input/output environment for function input data and output results.

Details

For the time period 1918–1949, landings of other rockfish (ORF) and total rockfish (TRF) are identical because catch agencies did not identify Pacific Ocean Perch (POP). POP started showing up in catch records in 1950, but appear artificially low from 1950–1952. Therefore, for the period 1918–1952 we predict ORF and POP using a linear regression of ORF *vs*. TRF landings from the 1953–1995 data (excluding an anomalous 1966 point):

$$O = 2^a T^b$$

where a = -0.8063372, b = 1.006260, O = ORF, T = TRF, and POP = TRF - ORF.

The ratios calculated by buildCatch (α proportion of RRF caught in a PMFC area for each fishery, β proportion of RRF caught in a PMFC area for H&L fisheries, γ ratio of RRF to a larger group like ORF or TRF, δ discard rate of RRF per other catch category from observer logs, λ proportion of early ORF/TRF catch by general gear type) are applied to the historical data to estimate historical landings of the RRF species. For POP the historical trawl landings are known back to 1956 and so these are used instead of deriving estimates.

The eight modern catch databases (detailed above) summarize landings as metric tonnes in a 5-dimensional array where i = calendar year, j = PMFC major area code, k = fishery ID, l = species or catch group, and m = the database. Then for each element (i, j, k, l), the maximum landing across database m is taken. This methodology is utilized due to the non-contiguous nature of the data sources. This also means that compiling individual catch records across the database sources, as performed for the trawl fishery by getCatch, is too complicated for the hook and line fisheries. It is possible if definite cut points can be identified for the various time series (which we no longer do for the reconstruction).

Value

The final reconstructed catch array is invisibly returned. Additionally, if saveinfo = TRUE, various data objects internal to the function are saved to a global list object called PBStool, and if diagnostics = TRUE, numerous plots of the various intermediary data objects are stored as image files in a subdirectory called CRdiag. Also, various data objects, including the eight database catch files consolidated and massaged, are saved as system binary files .rda. The final reconstructed catch array is also flattened to a comma-delimited text file Catch-History-123.csv, where 123 represents the species code.

Note

The function only has one argument for an Oracle database user ID and one for its password. If the user's authentication credentials are different for PacHarv3 and GFFOS, specify uid and pwd as two- element vectors. The SQL servers assume a trusted relationship with the user's DFO login identity.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getCatch, getData, orfhistory
Available SQL queries: SQLcode

---

calcHabitat *Calculate Potential Habitat Using Bathymetry*

Description

Calculate potential habitat using specified bathymetry limits.

Usage

```
calcHabitat(topofile="bctopo", isob=c(150,435),
   digits=1, minVerts=10, col.hab="greenyellow", col.land="moccasin",
   xlim=NULL, ylim=NULL, areas=list(), col.areas="red", isolab=TRUE, labtit="",
   plot=TRUE, pin=c(7,8), eps=FALSE, pix=FALSE, wmf=FALSE, ioenv=.GlobalEnv)
```

Arguments

topofile	topographical ASCII XYZ file downloaded from the Scripps Institution of Oceanography http://topex.ucsd.edu/WWW_html/mar_topo.html.
isob	isobath limits (shallowest, deepest).
digits	the number of decimal places to round geographical coordinates.
minVerts	the minimum number of vertices per polygon (parent or hole) to retain the polygon (becomes critical when digits>1).
col.hab	colour to fill the bathymetry interval (potential habitat).
col.land	colour to fill the land area.
xlim	longitude limits of the x-axis
ylim	latitude limits of the y-axis
areas	list of area objects
col.areas	vector of colours for the boundary lines of area objects (recycled if necessary)
isolab	logical: if TRUE, add a legend label to the plot
labtit	title for the legend label
plot	logical: if TRUE, plot the results.
pin	numeric vector: width and height of a figure device plot in inches.
eps	logical: if TRUE, send plot to a postscript (.eps) file.
pix	logical: if TRUE, send plot to a raster (.png) file.
wmf	logical: if TRUE, send plot to a metafile (.wmf) file.
ioenv	input/output environment for function input data and output results.

Details

The function uses the dataset bctopo downloaded from the site http://topex.ucsd.edu/cgi-bin/get_data.cgi using the boundaries: west (-134.5), east (-124.5), south (48), north (55).

The PBSmapping function makeTopography can become very slow as the precision of the longitude-latitude data is increased. Setting digits=1 provides a very fast but roughly estimated habitat region. Note that you may need to reduce the value of minVerts to get get all the major shapes. Increasing the precision to digits=2 slows the code markedly but provides a much more detailed bathymetry outline.

Some choice of bathymetry limits, noticeably shallow depths, cause the function joinPolys to crash. Therefore, calcHabitat taps into a patch function findHoles which weeds out small polygons (less than minVerts vertices) and identifies holes within parents. The function findHoles can become very slow when there are hundreds of polygons. Ideally, findHoles should be programmed in C.

Value

Returns a a bathymetry habitat PolySet. Additionally, a global list object PBStool contains the object bathy created from the call to makeTopography, the final bathymetry habitat habitat, and a variety of PolySets created in transforming bathy to habitat.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

calcSurficial, getFile, findHoles
PBSmapping: makeTopography, convCP, joinPolys, calcArea
grDevices: contourLines

Examples

```
## Not run:
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  calcHabitat(isob=c(150,435),digits=1) # DBR
  invisible() }
pbsfun()
})

## End(Not run)
```

---

calcLenWt *Calculate Length-Weight Relationship*

Description

Calculate the length-weight relationship for a fish species.

Usage

```
calcLenWt(dat=bio440, strSpp="440", areas=NULL,
    ttype=NULL, stype=NULL, plotit=FALSE)
```

Arguments

dat	data frame of biological information (usually generated by query gfb_bio.sql).
strSpp	character string specifying Hart species code (e.g., "440" = Yellowmouth Rockfish).
areas	area list named by area type (e.g., major, srfa, etc.).
ttype	list of trip types (1=non-observed commercial, 2=research, 3=survey, 4=observed commercial).
stype	list of sample types (see GFBioSQL for codes).
plotit	logical: if TRUE, plot the results to a raster image file (.png).

Details

The functions spits results out to a comma-delimited text file (.csv). If the user specifies plotit=TRUE then image files are also generated. The code is currently set to spew .png files but with some code tweaking, other file types could be generated.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

calcVB, estOgive, histMetric, mapMaturity

---

calcMA *Calculate Moving Average of a Series*

Description

Calculate a moving average of a series using a fixed period occurring every number of specified base units (usually days).

Usage

```
calcMA(x, y, y2, period = 270, every = 10)
```

Arguments

x	x-values that define the units over which the series is observed (e.g., dates, years, months, etc.). Must be one of the following classes: numeric, integer, ts, POSIXt, POSIXct, POSIXlt, or Date.
y	y-values that measure the values to be smoothed by a moving average.
y2	optional y-values that can be used should any y be missing.
period	period in base units of x (usually days) over which each average in the moving series is calculated.
every	base units of x (usually days) to move the period forward for successive calculations of the average y.

Details

The moving average is accumulated backwards from the last observation.

If your series does not depend on any particular x (e.g., date), set x=1:nrow(y).

Value

A data frame that represents the moving average series. This object comprises three columns:
x = the end x-value of each period of the moving average;
y = the average y-value for each period;
z = the numeric value of x (above) taken from the cumulative sum of the input x-units starting at 0 (see below).

Additionally, the moving average data frame above has an attribute called `dat` that contains the original input data as a data frame comprising three columns:
x = the x-value of the input series;
y = the y-value of the input series;
z = the numeric cumulative sum of the x-units starting at 0.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`glimmer`, `plotDiversity`, `simBGtrend`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows") {
    getData("Ex03_Portfolio","Examples",type="MDB",path=.getSpath())
    # Annual average every 2 months:
    ma=calcMA(PBSdat[[2]],PBSdat[[3]],period=365,every=60)
    dat=attributes(ma)$dat
    plot(dat$x,dat$y,type="l")
    lines(ma$x,ma$y,col="blue",lwd=2) }
  invisible() }
pbsfun()
})
```

---

calcMoments *Calculate Survey Moments from Raw Data*

Description

Calculate survey moments, including relative biomass, for a species from raw survey data.

Usage

```
calcMoments(strSpp="396", survID=c(1,2,3,121,167))
```

Arguments

`strSpp`	string specifying species code (`species$code`).
`survID`	numeric survey ID assigned to each survey by the GFBio team.

Details

The function uses `getData` to calculate survey specs using the SQL queries:
`gfb_survey_stratum.sql`
`gfb_survey_catch.sql`.

These queries are modified SQL procedures originally designed by Norm Olsen to build SQL tables:
`BOOT_HEADER` and `BOOT_DETAIL`.

Value

Invisibly returns a list of survey moments. Additionally, a global list object `PBStool` provides the following:

`doors`	Vector of default doorspread values for each survey from `BOOT_DEFAULTS`.
`speed`	Vector of default vessel trawling speeds for each survey from `BOOT_DEFAULTS`.
`moments`	List of moment calculations by strata, listed by `survID`.

Author(s)

Norm Olsen, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`calcPMR`, `showIndices`
Available SQL queries: `SQLcode`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(strSpp="455", survID=c(1,121),dfo=FALSE){
  if (dfo) {
    momList=calcMoments(strSpp,survID)
    print(momList) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
})
```

---

calcOccur *Calculate Percent Occurrence of EventData in PolySet*

Description

Calculate the percent occurrence of `EventData` in a `PolySet`.

Usage

```
calcOccur(polyset="qcb", events, wt=1, mess=FALSE, ioenv=.GlobalEnv)
```

Arguments

polyset	name of a PolySet object (e.g., qcb).
events	name of an EventData object.
wt	weight to give each fishing event (default = 1); weight wt can also be a string specifying a field in events (e.g., "gcat" might be the catch of the species).
mess	logical: if TRUE, display the percent occurrence on the current device.
ioenv	input/output environment for function input data and output results.

Details

The function calculates the percent occurrence of events in a set of polygons. The PolySet must have a PolyData attribute with a category field called label.

Value

A vector of percent occurrences by the category label contained in the PolyData attribute of PolySet. Additionally, the global object PBStool is created and contains:

pd	the PolyData set modified.
loc	the LocationSet created by findPolys().
eidpid	a vector showing number of events in each PID.
occur	a vector of total occurrences by the category label.
poccur	a vector of percent occurrences by the category label.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

calcHabitat, calcSurficial, qcb
PBSmapping: findPolys

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(testdatC,envir=.PBStoolEnv)
  dat=testdatC[!is.na(testdatC$X) & !is.na(testdatC$Y),]
  edat=as.EventData(dat,projection="LL",zone=9)
  calcOccur(polyset="qcb", events="edat", mess=TRUE)
  invisible() }
pbsfun()
})
```

`calcPMR` *Calculate (p, mu, rho) from a Sample Population*

Description

Calculate `p`, `mu`, and `rho` values for a sample population.

Usage

```
calcPMR(x, na.value=NULL)
```

Arguments

`x`	vector: sample population.
`na.value`	specify a value to assign to `NA` values in the sample; the default `NULL` means that `NA` values will be removed before the calculation.

Details

This function calculates the binomial-gamma parameters of a sample, as described by Schnute and Haigh (2003):
`p` = proportion of zero values,
`mu` = the mean of the non-zero values,
`rho` = the CV of the non-zero values.

Value

A named vector with four elements: `n` (sample size), `p`, `mu`, and `rho`.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

`getPMR`, `sampBG`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(P=list(n=1000,p=0.25,mu=1,rho=0.5)) {
  unpackList(P)
  x = sampBG(n,p,mu,rho) # create a sample population
  y = calcPMR(x)
  cat("True parameters for sampling the binomial-gamma distribution:\n")
  print(unlist(P))
  cat("Estimated parameters of the sampled binomial-gamma distribution:\n")
  print(y)
  invisible() }
pbsfun()
})
```

`calcRatio` *Calculate the Ratio of One Catch to Another*

Description

Calculate the annual ratios of one catch to another (or one field to another) for various PMFC major areas and coastwide.

Usage

```
calcRatio(dat, nfld, dfld, nzero=TRUE, dzero=TRUE, sumF=mean,
   major=NULL, startM=1, plot=FALSE, ylim=NULL, wmf=FALSE,
   quiet=FALSE, ioenv=.GlobalEnv)
```

Arguments

`dat`	catch data file, either generated by a call to SQL or supplied by the user.
`nfld`	numerator field for the ratio calculations.
`dfld`	denominator field for the ratio calculations.
`nzero`	logical: if `TRUE`, retain zero-value numerators, otherwise remove.
`dzero`	logical: if `TRUE`, retain zero-value denominators, otherwise remove.
`sumF`	function that summarises ratios (e.g., `mean`).
`major`	Pacific Marine Fisheries Commission major area codes (e.g., 3:9).
`startM`	scalar integer from 1 to 12 specifying which month starts the fishing/fiscal year.
`plot`	logical: if `TRUE`, plot the annual ratios for the various PMFC areas.
`ylim`	explicit limits for the y-axis.
`wmf`	logical: if `TRUE`, send the plot to a `.wmf` file.
`quiet`	logical: if `TRUE`, do not interupt code with `showError` but return `NULL`. (This is handy when using `calcRatio` in other functions.)
`ioenv`	input/output environment for function input data and output results.

Details

For each numerator/denominator pair in the set, the following actions are taken:

If the numerator or the denominator contains `NA`, discard the observation.

If zero-value denominators are allowed (`dzero=TRUE`) then the folowing conditions apply:

(a) if the numerator > 0 and the denominator = 0, then set the denominator to the numerator (e.g., entire catch was discarded);

(b) if the numerator and the denominator = 0, then set the numerator to 0 and set the denominator to the small value 0.0001 (e.g, discard rate is effectively zero).

Value

Invisibly returns the mean ratios in a matrix where rows are years and columns are PMFC area codes.

Other objects are saved to the global list object `PBStool`.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

makeCATtables, pmfc, convFY

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(dfo=FALSE){
  if (dfo) {
    getData("pht_catORF.sql",strSpp="440",path=.getSpath())
    ymr.pop.orf = PBSdat
    pmean=calcRatio(dat=ymr.pop.orf, nfld="catch", dfld="ORF",
      major=3:9, plot=TRUE, ylim=c(0,.35))
    print(pmean) }
  else cat("If you are logged onto the DFO network,\nrun 'pbsfun(dfo=TRUE)'\n.")
  invisible() }
pbsfun()
})
```

---

calcSG *Calculate Fit using a Schnute Growth Model*

Description

Calculate the length or weight *vs.* age relationship by fitting the data using a versatile Schnute growth model.

Usage

```
calcSG(dat=pop.age, strSpp="", yfld="len", tau=c(5,40),
     areas=list(major=NULL, minor=NULL, locality=NULL, srfa=NULL,
     srfs=NULL, popa=NULL), ttype=list(c(1,4),c(2,3)), stype=NULL,
     year=NULL, ameth=3, allTT=TRUE, jit=c(0,0),
     pix=FALSE, wmf=FALSE, singles=FALSE, pages=FALSE, ioenv=.GlobalEnv)
```

Arguments

dat	biological data set with fields yfld and age.
strSpp	string code for species. If dataset has attribute spp, this will be used before strSpp.
yfld	string name of the field in dat that contains y-values for the fit with age.
tau	numeric vector specifying two fixed parameters: the age of a young fish and that of an old fish.
areas	list of listed area codes; choices can be any/all of major, minor, locality, srfa, srfs, popa.
ttype	list of trip type codes to use from field ttype.
stype	list of sample type codes to use from field stype.
year	numeric integer vector of years used to subset dat.
ameth	code specifying ageing method protocol (defaults to 3, specifying break & burn only).
allTT	logical: if TRUE, force the display of all trip types either specified or available.
jit	two-element vector specifying amount of jittering to apply in the x- and -y-directions respectively. (See jitter in the **base** package.)

pix	logical: if TRUE, send plot to a .png file.
wmf	logical: if TRUE, send plot to a .wmf file.
singles	logical: if TRUE and pix|wmf, send each area/trip type combination to separate files.
pages	logical: if TRUE and pix|wmf, send each page of area plots to separate files.
ioenv	input/output environment for function input data and output results.

Details

The function plots versatile Schnute fits to length-age or weight-age data where each column shows the fits for Males, Females, and M+F. If the data are plotted to the screen device, each row shows the fits by trip type, and each page shows the fits by area. If plots are sent to image files (.png or .wmf), each trip type and area combination goes to separate image files (handy for stacking in a .doc file).

The versatile Schnute fit requires initial parameter estimates and bounds. These are taken from the data object parVec. If the species code you are using is not contained in this file, the function uses the initial estimates for Pacific Ocean Perch (*Sebastes alutus*).

Value

No value is explictly returned by the function. A global list object PBStool provides the following plus some labels:

out	Output array of model parameter estimates and other metrics.
pVec	The initial parameter vector used for the model.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.
fits	List of x- and y-values that describe the versatile Schnute fits to the data.

The fits for females, males, and sexes combined (M+F) are also written to a comma-separated .csv file using a name that includes areas, trip types, and years (if specified).

Note

A suitable data set can be obtained by running the SQL query gfb_bio.sql.

```
getData("gfb_bio.sql","GFBioSQL","396",path=.getSpath())
```

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

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

See Also

calcVB, pop.age, parVec, getData, biteData, pmfc, species
PBSmodelling: calcMin
base: jitter

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  calcSG(areas=list(srfa=list(c("5AB","5CD"))),ttype=list(2:3))
  invisible() }
pbsfun()
})
```

`calcSRFA` *Get Slope Rockfish Assessment Areas*

Description

Determine slope rockfish assessment areas from combinations of PMFC major and minor areas as well as locality codes for fishing grounds within PMFC minor areas.

Usage

```
calcSRFA(major, minor=NULL, loc=NULL, subarea=FALSE)
```

Arguments

`major`	numeric: Pacific Marine Fisheries Commission major area codes or matrix/data frame with three columns containing codes `major`, `minor`, and `loc`
`minor`	numeric: Pacific Marine Fisheries Commission minor area codes
`loc`	numeric: locality codes denoting fishing grounds within PMFC minor areas
`subarea`	logical: if `TRUE` then determine SRF subareas (Queen Charlotte Sound gullies) instead of SRF areas.

Details

The function either accepts three vectors (`major`, `minor`, `loc`) of equal length or one matrix/data frame `major` with three columns corresponding to the first three arguments.

Value

A character vector specifying slope rockfish assessment areas/subreas.

See Also

`srfa`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  dat=data.frame(major=c(3:7,9,9),
    minor=c(23,26,11,8,8,35,31),loc=c(1,1,1,3,6,1,1))
  sdat=cbind(dat,srfa=calcSRFA(dat),srfs=calcSRFA(dat,subarea=TRUE))
  print(sdat); invisible() }
pbsfun()
})
```

`calcSurficial` *Calculate Intersection of Surficial Geology and Bathymetry Habitat*

Description

Calculate the intersection of surficial geology and potential habitat using a bathymetry interval.

Usage

```
calcSurficial(surf="qcb", hab,
    xlim=c(-133.4,-127.8), ylim=c(50.5,54.8),
    col.hab=c("aliceblue","grey"), col.cst="grey85",
    pix=FALSE, wmf=FALSE, ioenv=.GlobalEnv)
```

Arguments

`surf`	surficial geology data object (e.g., `qcb`).
`hab`	bathymetry habitat data object created by `calcHabitat()`.
`xlim`	X-limits of the mapping region.
`ylim`	Y-limits of the mapping region.
`col.hab`	2-element vector of colours for `hab` (fill, border).
`col.cst`	scalar specifying fill colour for the coast.
`pix`	logical: if `TRUE`, send plot to a `.png` file.
`wmf`	logical: if `TRUE`, send plot to a `.wmf` file.
`ioenv`	input/output environment for function input data and output results.

Details

The function uses available datasets `surf` (surficial geology) and `hab` (bathymetry habitat) and calculates the intersection. The results are displayed on a map and stored in the global list object `PBStool`.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`calcHabitat`, `preferDepth`, `qcb`, `bctopo`
PBSmapping: `makeTopography`, `convCP`, `joinPolys`, `calcArea`

Examples

```
## Not run:
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  habitat=calcHabitat(isob=c(150,435), digits=1, plot=FALSE) # DBR
  calcSurficial(surf="qcb", hab=habitat)
  invisible() }
pbsfun()
})

## End(Not run)
```

`calcVB` *Calculate Fit using a von Bertalanffy Growth Model*

Description

Calculate the length or weight *vs.* age relationship by fitting the data using a von Bertalanffy growth model.

Usage

```
calcVB(dat=pop.age, strSpp="", yfld="len", fixt0=FALSE,
     areas=list(major=NULL, minor=NULL, locality=NULL, srfa=NULL,
     srfs=NULL, popa=NULL), ttype=list(c(1,4),c(2,3)), stype=NULL,
     year=NULL, ameth=3, allTT=TRUE, jit=c(0,0),
     pix=FALSE, wmf=FALSE, singles=FALSE, pages=FALSE, ioenv=.GlobalEnv)
```

Arguments

`dat`	biological data set with fields `yfld` and `age`.
`strSpp`	string code for species. If dataset has attribute `spp`, this will be used before `strSpp`.
`yfld`	string name of the field in `dat` that contains y-values for the fit with `age`.
`fixt0`	logical: if `TRUE`, fix the parameter `t0` to equal 0.
`areas`	list of listed area codes; choices can be any/all of `major`, `minor`, `locality`, `srfa`, `srfs`, `popa`.
`ttype`	list of trip type codes to use from field `ttype`.
`stype`	list of sample type codes to use from field `stype`.
`year`	numeric integer vector of years used to subset `dat`.
`ameth`	code specifying ageing method protocol (defaults to `3`, specifying break & burn only).
`allTT`	logical: if `TRUE`, force the display of all trip types either specified or available.
`jit`	two-element vector specifying amount of jittering to apply in the x- and -y-directions respectively. (See `jitter` in the **base** package.)
`pix`	logical: if `TRUE`, send plot to a `.png` file.
`wmf`	logical: if `TRUE`, send plot to a `.wmf` file.
`singles`	logical: if `TRUE` and `pix|wmf`, send each area/trip type combination to separate files.
`pages`	logical: if `TRUE` and `pix|wmf`, send each page of area plots to separate files.
`ioenv`	input/output environment for function input data and output results.

Details

The function plots von Bertalanffy fits to length-age or weight-age data where each column shows the fits for Males, Females, and M+F. If the data are plotted to the screen device, each row shows the fits by trip type, and each page shows the fits by area. If plots are sent to image files (`.png` or `.wmf`), each trip type and area combination goes to separate image files (handy for stacking in a `.doc` file).

The von Bertalanffy fit requires initial parameter estimates and bounds. These are taken from the data object `parVec`. If the species code you are using is not contained in this file, the function uses the initial estimates for Pacific Ocean Perch (*Sebastes alutus*).

Value

No value is explictly returned by the function. A global list object `PBStool` provides the following plus some labels:

`out`	Output array of model parameter estimates and other metrics.
`pVec`	The initial parameter vector used for the model.
`xlim`	Limits of the x-axis.
`ylim`	Limits of the y-axis.
`fits`	List of x- and y-values that describe the von-Bertalanffy fits to the data.

The fits for females, males, and sexes combined (M+F) are also written to a comma-separated `.csv` file using a name that includes areas, trip types, and years (if specified).

Note

A suitable data set can be obtained by running the SQL query `gfb_bio.sql`.

```
getData("gfb_bio.sql","GFBioSQL","396",path=.getSpath())
```

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`calcSG`, `pop.age`, `parVec`, `getData`, `biteData`, `pmfc`, `species`
PBSmodelling: `calcMin`
base: `jitter`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  calcVB(areas=list(srfa=list(c("5AB","5CD"))))
  invisible() }
pbsfun()
})
```

---

chewData *Remove Sparse Category Records from Data Matrix/Frame*

Description

Remove records from a data frame or matrix that contribute little information to unique categories of the factor specified.

Usage

```
chewData(dat, fac, nmin=3, na.rm=TRUE)
```

Arguments

dat	matrix or data frame with named fields (row names).
fac	string scalar corresponding to a field name in dat.
nmin	minimum number of records in a factor category; if reached, records are discarded from dat.
na.rm	logical: if TRUE, remove records with unlabelled categories.

Details

Records are removed from a data frame/matrix if a specified factor's categories don't number more than nmin. If there are no fields with the name of fac, the data matrix/frame is returned unaltered.

Value

A subset of dat after thinning fac by nmin.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

biteData, getData, getFile

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  cat("Unique records in 'iris$Petal.Width'\n")
  print(sapply(split(iris$Petal.Width,iris$Petal.Width),length))
  test=chewData(iris,"Petal.Width",5)
  cat("'Petal.Width' categories fewer than 5 removed\n")
  print(sapply(split(test$Petal.Width,test$Petal.Width),length))
  invisible() }
pbsfun()
})
```

---

clarify *Summarize Catch Data Into CLARA clusters*

Description

Summarize large datasets of catch represented by many species into *n* CLARA clusters.

Usage

```
clarify(dat, cell=c(0.1,0.075), nG=8,
   xlim=c(-134.2,-127), ylim=c(49.6,54.8), zlim=NULL, targ="YMR",
   clrs=c("red","orange","yellow","green","forestgreen","deepskyblue",
   "blue","midnightblue"), land="ghostwhite", spp.names=FALSE,
   wmf=FALSE, eps=FALSE, hpage=10, ioenv=.GlobalEnv)
```

Arguments

dat	an EventData obejct with fields EID, X, Y, optional Z, and species fields named by 3-letter codes.
cell	cell size in degrees Lon/Lat (UTMs currently not implemented).
nG	number of CLARA groups to end up with.
xlim	X-limits of the mapping region.
ylim	Y-limits of the mapping region.
zlim	optional Z-limits (depth) to qualify data (if field available).
targ	target species to identify within the CLARA groups.
clrs	vector of colours to use for displaying nG CLARA groups.
land	colour to shade land masses.
spp.names	logical: if TRUE, replace species codes (if they are used in the data) with species names in the legend.
wmf	logical: if TRUE, send plot to a windows meta file .wmf.
eps	logical: if TRUE, send plot to an encapsulated postscript .eps file.
hpage	dimension in inches of the longest side of a plot page sent to an output file.
ioenv	input/output environment for function input data and output results.

Details

The function uses the function clara in the package **cluster**. Essentially, clara sub-samples the large data set, identifying the best *k* medoids (centre is defined as the item which has the smallest sum of distances to the other items in the cluster) using a dissimilarity metric. In the end, all records are assigned to one of the k clusters. Further routines in the R-package PBSmapping locate fishing events in grid cells and display the results.

The results are displayed on a map and stored in the global list object PBStool.

Author(s)

Rowan Haigh and Norm Olsen, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Kaufman, L. and Rousseeuw, P.J. (1990) Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

See Also

getFile, prepClara
PBSmapping: makeGrid, findCells
cluster: clara (clustering large applications)
PBSdata: claradat for an example dataset.

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  clarify(claradat,xlim=c(-132,-127.8),ylim=c(50.6,52.1),
    cell=c(.05,.03),nG=7)
  invisible() }
pbsfun()
})
```

collectFigs *Collect Postscript Figures Into One Document*

Description

Collect figures (currently only encapsulated postscript `.eps` supported) into one document using a latex compiler (`latex.exe`, `dvips.exe`, `ps2pdf.exe`).

Usage

```
collectFigs(path=".", ext="eps", is.fnum=FALSE,
   fout="collectFigs", width=6, capskip=0)
```

Arguments

`path`	path to `.eps` files; can be relative or absolute.
`ext`	extension of figure files (currently only `"eps"` supported).
`is.fnum`	logical: if `TRUE`, a figure number or identifier occurs in the first string sequence of a file name delimited by `"-"`.
`fout`	file name prefix of the output file.
`width`	width (inches) to render each figure in the collection.
`capskip`	space (pt) between each figure and its caption (negative values reduce the space).

Details

The code constructs a `.tex` file that uses all available `.eps` files in the specified directory. The final result is a bookmarked PDF file called `<fout>.pdf`.

Value

Returns a string vector of latex commands that form the `.tex` file when written to the system directory specified by the user.

Note

The latex `bin` directory (e.g., `C:\Apps\MiKTeX\miktex\bin`) must be on the system path.

The tex code requires the latex packages: `geometry`, `epsfig`, `caption`, and `hyperref`.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

http://latex-community.org/forum/viewtopic.php?f=5&t=21405

See Also

`buildCatch`, `writeLines`

compCsum *Compare Cumulative Relative Frequency Curves*

Description

Compare cumulative relative frequency curves in a common plot space.

Usage

```
compCsum(dat=pop.age, pro=TRUE, strSpp="", xfld="age", plus=60,
     yfac="year",areas=list(major=NULL, minor=NULL, locality=NULL,
     srfa=NULL, srfs=NULL, popa=NULL), ttype=list(c(1,4),c(2,3)),
     stype=c(1,2,5:8), ameth=3, allTT=TRUE, years=1998:2006,
     pix=FALSE, wmf=FALSE, singles=FALSE, pages=FALSE, ioenv=.GlobalEnv)
```

Arguments

dat	Biological data set with fields xfld and yfac.
pro	logical: if TRUE, transform the raw age observations to proportions-at-age.
strSpp	String code for species. If dataset has attribute spp, this will be used before strSpp.
xfld	String name of *x*-field for calculating cumulative frequencies.
plus	Plus class (accumulated) for xfld.
yfac	Vector of string names that correspond to field names in dat. Special cases "area" and "trip" create new fields from input arguments areas and ttype.
areas	List of listed area codes; choices can be any/all of major, minor, locality, srfa, srfs, popa.
ttype	List of trip type codes to use from field ttype.
stype	List of sample type codes to use from field stype.
ameth	Code specifying ageing method protocol (defaults to 3, specifying break & burn only).
allTT	Logical: if TRUE, force the display of all trip types either specified or available.
years	Vector of numeric years used in special cases of yfac (see above).
pix	Logical: if TRUE, send plot to a .png file.
wmf	Logical: if TRUE, send plot to a .wmf file.
singles	Logical: if TRUE and pix|wmf, send each area/trip type combination to separate files.
pages	Logical: if TRUE and pix|wmf, send each page of area plots to separate files.
ioenv	input/output environment for function input data and output results.

Details

The function compares cumulative relative frequency curves for specified factors yfac. Typically, factors are fields in dat and comparisons are performed for each specified area and trip type. If the data are plotted to the screen device, each panel contains curves by combinations of trip type (rows) and factor (columns), and each page shows these combinations by area. If plots are sent to image files (.png or .wmf), either each row is sent to separate files when singles=TRUE (handy for stacking in a .doc file) or each page is sent to a separate file when pages=TRUE.

In special cases yfac="area" or yfac="trip", each panel depicts curves by year, and the factors (curves) are created as new fields from inputs areas or ttype. (The option yfac="trip" is not implemented at present.)

Value

No value is explictly returned by the function. A global list object PBStool provides the following plus some labels:

dat	Qualified data used for the analysis.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.
clrs	Colour vector used to plot cumulative curves for each factor.
plotname	Current plot name used by this function, but also by .plotDev().

Note

A suitable data set can be obtained by running the SQL query gfb_bio.sql.

```
getData("gfb_bio.sql","GFBioSQL","396",path=.getSpath())
```

See Also

pop.age, calcVB, histMetric, getFile, biteData

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  compCsum(areas=list(srfa=list(c("5AB","5CD"))),ttype=list(c(2,3)))
  invisible() }
pbsfun()
})
```

---

confODBC *Configure an ODBC User Data Source Name*

Description

Use a command line utility found in Windows OS called 'odbcconf.exe' that configures an ODBC Data Source Name from the system's command line.

Usage

```
confODBC(dsn="PacHarvest", server="GFDB", db="PacHarvest",
         driver="SQL Server", descr="", trusted=TRUE)
```

Arguments

dsn	Data Source Name for the ODBC connection.
server	string specifying name of server that hosts the SQL Server databases.
db	string specifying the name of the default database for the DSN connection.
driver	string specifying the ODBC driver.
descr	string specifying optional description of the DSN connection.
trusted	logical: if TRUE, allow SQL Server to use a trusted DFO login ID.

Details

confODBC() runs the Windows command line utility with the user-supplied information and configures a User DSN on the user's computer.

In XP, the new User DSN can be seen by navigating to:
<Control Panel><Administrative Tools><Data Sources (ODBC)>.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

getData, runModules

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type,dfo=FALSE) {
  if (os=="windows" && dfo)
    confODBC(dsn="Popsicle", server="GFDB", db="PacHarvest",
      driver="SQL Server", descr="Icy sweet nonsense", trusted=TRUE)
  else showMessage("Only functional for DFO personnel using Microsoft Windows OS")
  invisible() }
pbsfun()
})
```

---

convFY *Convert Dates into Fishing/Fiscal Years*

Description

Convert a vector of dates into a vector of fishing or fiscal years based on a specified month to start the fishing/fiscal year.

Usage

```
convFY(x, startM=4)
```

Arguments

x	vector of date limits with class POSIXt or a character vector of dates specified in the format "yyyy-mm-dd" that can be converted to dates.
startM	scalar integer from 1 to 12 specifying which month starts the fishing/fiscal year.

Value

Returns a numeric vector of fishing/fiscal years of equal length to the input vector of dates. The output vector has names representing year-month "yyyy-mm". If startM=1, the output dates also represent calendar years.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

convYM, convYP

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  cat("Fishing years starting in April\n")
  print(convFY(paste("2009-",pad0(1:12,2),"-15",sep="")))
  invisible() }
pbsfun()
})
```

---

convYM *Convert Date Limits into a Vector of Year-Months*

Description

Convert a two-element vector of date limits into a vector of year-months where every month within the limits is represented.

Usage

```
convYM(x)
```

Arguments

x	vector of date limits with class POSIXt or a character vector of dates specified in the format "yyyy-mm-dd" that can be converted to dates.

Value

Returns an expanded character vector of ordered year-months "yyyy-mm" between and including the specified date limits. This character vector can be used for labelling and other factor-like routines.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

convYP, convFY

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  cat("Year-month labels between the specified date limits\n")
  print(convYM(c("2009-07-01","2011-11-11")))
  invisible() }
pbsfun()
})
```

convYP *Convert Dates into Binned Year Periods*

Description

Convert a vector of dates into one of year-periods through binning the dates into intervals specified in days.

Usage

```
convYP(x, ndays=90)
```

Arguments

x	date vector with class POSIXt or a character vector of dates specified in the format "yyyy-mm-dd" that can be converted to dates.
ndays	integer value specifying a time interval in days to partition a year.

Details

The routine actually fudges the integer day interval to a real number that will create even breaks.

Value

Returns a vector (with same length as input date vector) that specifies dates as real numbers where the integer part is the year and the fraction part is the proportion of the year corresponding to the bin's right-most value. Each element of the vector also has a name that specifies the year and bin number.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

trackComp, convYM, convFY

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  x=paste("2009-",c("03-31","06-30","09-30","12-30"),sep="")
  cat("Year periods based on numeric value of input dates\n")
  print(convYP(x,30))
  invisible() }
pbsfun()
})
```

createDSN *Create User DSNs for PBS Groundfish Databases*

Description

Create a suite of User DSNs for PBS groundfish databases, currently hosted on the server SVBCPBSGFIIS.

Usage

```
createDSN(trusted=TRUE)
```

Arguments

trusted	logical: if TRUE, allow SQL Server to use a trusted DFO login ID.

Details

createDSN() automatically creates (overwrites) six User DSNs:
GFBioSQL
GFCatch
GFCruise
PacHarvest
PacHarvHL
PacHarvSable

In XP, the new User DSN can be seen by navigating to:
<Control Panel><Administrative Tools><Data Sources (ODBC)>.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

confODBC, runModules

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type,dfo=FALSE) {
  if (os=="windows" && dfo) createDSN()
  else showMessage("Only functional for DFO personnel using Microsoft Windows OS")
  invisible() }
pbsfun()
})
```

crossTab *Run a Cross Tabulation*

Description

Run a cross tabulation on a data frame, summarizing z-values by specified 'factors'.

Usage

```
crossTab(x=PBSdat, y=c("year","major"),
     z="landed", func=function(x){sum(x)/1000.}, ...)
```

Arguments

x	data frame for cross tabulation.
y	field names in x that contain factor-like levels to build a contigency table.
z	field name in x that contains the values to summarize in the contigency table.
func	function to summarize z at each factor combination found in y.
...	additional arguments to pass to **reshape**'s function cast.

Details

This function requires the R package **reshape**, and uses the function melt.data.frame and cast therein.

Value

A cross tabulation of y with values z summarized by func. The function crossTab is useful for summarizing data query results (*e.g.*, total catch by year and PMFC major area) when exploring / debugging functions like buildCatch.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

buildCatch, getCatch, getData
reshape: melt.data.frame, cast

dtget *Get/Print Objects From or Put Objects Into Temporary Work Environment*

Description

These functions are wrappers to the PBSmodelling accessor functions that get/print objects from or put objects into a temporary work environment, in this case .PBSdataEnv.

Usage

```
dtget(...)
dtcall(...)
dtprint(...)
dtput(...)
dlisp(...)
```

Arguments

...	For `dtget` through to `dtput`, the only free argument is: `x` – name (with or without quotes) of an object to retrieve or store in the temporary environment; cannot be represented by a variable. Fixed arguments: `penv = parent.frame(), tenv = .PBSdataEnv` See `tget` for additional information. For `dlisp`, there is only one fixed argument: `pos = .PBSdataEnv` All other arguments are available – see `lisp`

Details

These accessor functions were developed as a response to the CRAN repository policy statement: "Packages should not modify the global environment (user's workspace)."

Value

Objects are retrieved from or sent to the temporary working environment to/from the place where the function(s) are called. Additionally, `dtcall` invisibly returns the object without transferring, which is useful when the object is a function that the user may wish to call, for example, `dtcall(myfunc)()`.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

CRAN Repository Policy: http://cran.r-project.org/web/packages/policies.html

See Also

`tget` and `lisp` in **PBSmodelling**

---

dumpMod *Dump Modern Catch Used by Reconstruction into Tables*

Description

Dump catch from modern databases that has been collected and organised during a catch reconstruction (see `buildCatch`) into CSV tables.

Usage

```
dumpMod(dat, catch=c("landed","discard"), fid=1:5,
    strSpp="396", dbs=TRUE)
```

Arguments

dat	numeric array of catch (e.g., catmod1 with dimensions named c("year", "major", "fid", "catch", "dbs"
catch	character vector of elements in the dimension labelled "catch".
fid	numeric vector of fishery ID codes (1=Trawl, 2=Halibut, 3=Sablefish, 4=Dogfish-Lingcod, 5=H&L Rockfish).
strSpp	character string specifying species Hart code (e.g., "396" = Pacific Ocean Perch).
dbs	logical: if TRUE, the input array dat has the dimension labelled "dbs", and the catch dump will report values from all available database sources.

Details

See the catch reconstruction algorithm (buildCatch) for more details.

Value

The data in the array dat are flattened from four or five dimensions into two dimensions and sent to a comma-delimited text file (.csv).

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

References

Haigh, R. and Yamanaka, K.L. (2011) Catch history reconstruction for rockfish (*Sebastes* spp.) caught in British Columbia coastal waters. *Canadian Technical Report of Fisheries and Aquatic Sciences* **0000**: xx + 00 p.

See Also

buildCatch, dumpRat, plotRecon

---

dumpRat *Dump Ratios from Catch Reconstruction into Tables*

Description

Dump catch ratios calculated during a catch reconstruction into CSV tables.

Usage

```
dumpRat(strSpp="396", rats=c("alpha","beta","gamma","delta","lambda"),
   ioenv=.GlobalEnv)
```

Arguments

strSpp	character string specifying species Hart code (e.g., "396" = Pacific Ocean Perch).
rats	character vector of named ratios used in a catch reconstruction.
ioenv	input/output environment for function input data and output results.

Details

See the catch reconstruction algorithm (buildCatch) for more details.
Note: POP = Pacific Ocean Perch, ORF = all rockfish other than POP, TRF = total rockfish, TAR = target species/group, RRF = reconstructed rockfish species.
Briefly:
α = proportion RRF caught in PMFC areas by all fisheries;
β = proportion RRF caught in PMFC areas by H&L fisheries;
γ = ratio of RRF to a prominent group (POP, ORF, TRF, or TAR);
δ = discard rate of RRF from observer logs;
λ = proportion of early ORF/TRF catch by general gear type.

Value

The ratio arrays are stored in a list object PBStool123 spat out by the reconstruction procedure. The arrays are extracted and sent to a comma-delimited text file (.csv).

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

References

Haigh, R. and Yamanaka, K.L. (2011) Catch history reconstruction for rockfish (*Sebastes* spp.) caught in British Columbia coastal waters. *Canadian Technical Report of Fisheries and Aquatic Sciences* **0000**: xx + 00 p.

See Also

buildCatch, dumpMod, plotRecon

---

estOgive *Estimate Ogive Curves for Maturity*

Description

Estimate ogive curves using three methods:
(i) calculate the cumulative distribution of the proportion of mature fish smoothed by binned ages,
(ii) use a glm fit with binomial error linked to a logit function, and
(iii) fit proportion-at-age data to a double normal curve using the minimizer nlm.

Usage

```
estOgive(dat=pop.age, strSpp="", method=c("empir","dblnorm"),
     sex=list(1,2,1:2), mos=list(1:12,1:12,1:12), mat=3:7,
     ofld="age", obin=1, xlim=c(0,60), fg=c("blue","red","black"),
     bg=c("cyan","pink","grey85"), wmf=FALSE, ioenv=.GlobalEnv, ...)
```

Arguments

dat	biological data set with fields mat and age.
strSpp	string code for species. If dataset has attribute spp, this will be used before strSpp.

method	string vector specifying methods used to fit the data: "empir" - empirical fit to cumulative curves of proportion maturity (only when specified alone), "logit" - binomial logit fit where maturity is specified as 0 or 1 for each specimen, "dblnorm" - fit proportion-at-ages using a double normal curve.
sex	sex code: 1 = males, 2 = females.
mos	list of months from which maturity data are used; list length matches that of list argument sex.
mat	codes that define maturity (e.g., 3+).
ofld	ogive field (usually age or len).
obin	bin size (in x-units) to group age or len.
xlim	range of the x-metric (x-axis).
fg	vector of three foreground colours for the sexes listed in order.
bg	vector of three background colours for the sexes listed in order.
wmf	logical: if TRUE, send plot to a .wmf file.
ioenv	input/output environment for function input data and output results.
...	additional arguments for use in estOgive.

Details

The cumulative distribution of mature fish is plotted and the point at 50% maturity is interpolated. The x-axis (age or length) of empirical data can be binned to smooth out rough or sparse data. The user can also choose two methods to fit empirical data: glm fit of binary data (mature vs. not mature) using a linked logit function, or an nlm fit of the proportions-at-age data to a double normal curve (which is the default method used in Coleraine).

Value

No value is explictly returned by the function. A global list object PBStool provides the following:

mdxy	Last median x-y values.
mbin	Maturities binned by age (or length) group.
ld50	x-value at y = 0.50.
ldR	x-y values at the right end of the logit-fitted curve visible.
ldL	x-y values at the left end of the logit-fitted curve visible.
out	Results table showing proportions mature at each binned age or length.
pmat	Function calculating proportions mature.
obin	Bin size.
strSpp	String species code used.

Also, matrices by sex of empirical proportions-at-age and fits using a logit and double normal are dumped to a .csv file.

Note

A suitable data set can be obtained by running the SQL query gfb_bio.sql.

```
getData("gfb_bio.sql","GFBioSQL","396",path=.getSpath())
```

See Also

pop.age, fitLogit, mapMaturity, getData

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  estOgive(obin=2, xlim=c(0,30), sex=1:2, method=c("empir","dblnorm"), cex=0.9)
  invisible() }
pbsfun()
})
```

---

findHoles *Find Holes in Parent Polygons*

Description

Identifies polygons within polygons of the same PID, and transforms them into holes, if they are not already set that way. Also weeds out small polygons before the hole identification routine runs.

Usage

```
findHoles(polyset, minVerts = 10)
```

Arguments

`polyset`	a valid PBSmapping PolySet.
`minVerts`	minimum number of vertices required for a polygon to be retained before the algorithm identifying holes begins.

Details

This function can become very slow when there are hundreds of polygons. Hence the argument `minVerts` can reduce the number of candidate polygons. Ideally, `findHoles` should be programmed in C.

Value

A PolySet that identifies parents and holes based on the position of polygons within polygons.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`calcHabitat`
sp: `point.in.polygon`

fitLogit *Fit Binomial Data Using Logit Function*

Description

Fit binomial data using a logit link function in a `glm`.

Usage

```
fitLogit(dat, yfld="pmat", xflds="age")
```

Arguments

`dat`	data frame with binomial observations (y-values) and independent explanators (x-values).
`yfld`	string name of field containing binomial data.
`xflds`	string name(s) of field(s) containing explanatory data.

Details

The data are passed to the function `glm` in the `stats` package.

Value

The function returns a GLM fit of the binomial data using the specified independent observations. See the 'Value' section of the `glm` help file.

Author(s)

Rob Kronlund, Pacific Biological Station, Nanaimo BC

See Also

`estOgive`

flagIt *Flag a Coordinate*

Description

Take a coordinate `(a,b)` and label it using a diagonal line of radius `r` and angle `A`.

Usage

```
flagIt(a, b, A = 45, r = 0.2, n = 1, ...)
```

Arguments

`a,b`	midpoint coordinate of a circle
`A`	angle (degrees) to radiate out from coordinate
`r`	radius of the circle
`n`	number of times to decrement the radius (max=5)
`...`	additional `par` parameters for the `text` function.

Details

A diagonal dotted line in light grey radiates out from a central coordinate and the coordinate values are used to label the coordinate itself. The function adjusts for the aspect ratio created by different x- and y-limits, and for different x- and y-dimensions of the plot. This ensures that an angle looks correct in the plotting space.

Value

Invisibly returns a list of the vectors (a,x) and (b,y), the angle in radians, the original x-value calculated in a square Cartesian system, and the final x- and y-coordinates on the periphery of a circle.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`estOgive`, `scaleVec`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun = function() {
  plot(0,0,type="n",xlim=c(0,20),ylim=c(0,1))
  points(10,0.5,pch=20,col="blue")
  for (i in seq(10,360,10))
    flagIt(a=10, b=0.5, r=0.25, A=i,col="blue",cex=0.7)
}; pbsfun()
})
```

---

formatCatch *Format Table of Numeric Catch as Strings*

Description

Format a table of numeric values so that each cell displays *N* significant figures in character format.

Usage

```
formatCatch(dat, N=3, X=1, zero="0", K=",")
```

Arguments

`dat`	data frame/matrix of raw catch numbers
`N`	number of significant digits
`X`	numeric vector of columns to exclude (set to `NULL` or `0` if selecting all columns)
`zero`	character replacement for zero-values
`K`	big mark separator (see `format`)

Details

The algorithm relies on string manipulation to achieve a result that cannot be obtained by the **base** package's `format` function alone. The formatted table can then be passed to **xtable**'s function `xtable`, which produces output that can be passed to `print.xtable` for use in LaTeX.

Value

A data frame of formatted string values with dimensions and `dimnames` of the input matrix or data frame.

Note

If the input table has no `dimnames`, one will be added with format
`list(paste("R",1:nrow(dat),sep=""),paste("C",1:ncol(dat),sep=""))`

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`buildCatch`, `dumpMod`, `plotRecon`, `makeLTH`

Examples

```
local(envir=.PBStoolEnv,expr={
  print(formatCatch(genMatrix(20,5),X=0))
})
```

`genPa` *Generate Proportions-At-Age from Catch Curve Composition*

Description

Generate proportions-at-age using the catch curve composition of
Schnute and Haigh (2007, Table 2).

Usage

```
genPa( np = 40,
   theta=list(Z=0.1,mu=15,sigma=5,bh=c(10,20),rho=c(3,2),tau=1.5),
   sim = TRUE)
```

Arguments

`np`	number of age classes B.
`theta`	list of parameters $\theta = (Z;\ \mu,\sigma;\ b_1,...b_m,\rho_1,...\rho_m,\tau)$: `Z.....` total mortality ($Z = F + M$). `mu.....` age μ of full selectivity. `sigma...` variance parameter (rendered as the standard deviation σ) `.....` for the selectivity curve (left limb of a double normal distribution). `bh.....` ages b_h with anomalous recruitment, `.....` where $h = 1, ..., m$ and m = number of recruitment anomalies. `rho.....` recruitment anomaly parameter ρ_h at age b_h. `tau.....` standard deviation τ for recruitment anomalies.
`sim`	logical: if `TRUE`, return components of the simulated proportions-at age.

Details

Schnute and Haigh (2007) provide a catch curve model in Table 2 (p.219). Their model uses a design vector ϕ and a parameter vector θ; the latter contains the quantities to be estimated in a negative log likelihood using the complete parameter vector (θ, σ) for a logistic-normal model or (θ, n) for a Dirichlet model.

Here we use θ primarily as a parameter vector, but include the design elements $b_1, ...b_m$ for expediency.

Additionally, for the selectivity component, we use the parameterisation of Edwards et al. (2010) where μ represents the age of full selectivity and $\sigma = \sqrt{\nu}$ represents the variance of the left limb of a double normal (see equation F.7 in the Pacific Ocean Perch assessment). This parameretisation replaces (T2.2) in Schnute and Haigh (2007), which uses α and β_k.

Value

Proportions-at-age vector of length `np`.
If `sim=TRUE`, the components S_a, β_a, R_a, and p_a are writtten to the global list object `PBStool`.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

References

Edwards, A.M., Starr, P.J., and Haigh, R. (2010, under revision) Stock assessment for Pacific Ocean Perch (*Sebastes alutus*) in Queen Charlotte Sound, British Columbia. *Canadian Science Advisory Secretariat, Research Document* **2010/xxx**: iii + nnn p.

Schnute, J.T., and Haigh, R. (2007) Compositional analysis of catch curve data, with an application to *Sebastes maliger*. *ICES Journal of Marine Science* **64**: 218–233.

See Also

`scaleVec`, `sampBG`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  genPa(); unpackList(ttget(PBStool)); unpackList(theta)
  par0 = par(no.readonly=TRUE); on.exit(par(par0))
  expandGraph(mfrow=c(2,2),mar=c(1,4,1,0.5),oma=c(3,0,0,0),mgp=c(2,.5,0))
  for (i in c("Sa","Ba","Ra","pa")) {
    expr = paste(
      c("plot(1:np,",i,",type=\"l\",lwd=2,col=\"blue\",xlab=\"\",ylab=\"\");",
      "mtext(expression(italic(",substring(i,1,1),
      "[a])),side=2,line=2,cex=1.5,las=1);",
      "points(1:np,",i,",pch=21,bg=\"aliceblue\",cex=1.2)"),collapse="")
    eval(parse(text=expr))
    if (i=="Ba") lines(rep(mu,2),c(0.99,par()$usr[3]),lty=2,col="grey30")
    if (i=="Ra") for (j in 1:length(rho))
      lines(rep(bh[j],2),c(rho[j]+0.98,par()$usr[3]),lty=2,col="grey30")
    if (i=="pa") lines(1:np,scaleVec(Sa,pa[np],max(pa)),lty=2,col="grey30") }
  mtext("     Age",outer=TRUE,side=1,line=1.5,cex=1.5)
  invisible() }
pbsfun()
})
```

getBootRuns *Get Survey Bootstrap Results*

Description

Get the survey bootstrap results from GFBioSQL.

Usage

```
getBootRuns(strSpp)
```

Arguments

strSpp	character string code for a species

Details

This function calls the SQL query gfb_boot.sql to download the bootstrap results for the selected species from GFBioSQL tables BOOT_HEADER and BOOT_DETAIL. The bootstrap tables are maintained by Norm Olsen, PBS.

Value

A data frame containing the bootstrapped biomass estimates and confidence limits for strSpp from all surveys. The data frame is ordered by date and bootID.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getPMR, showIndices
Available SQL queries: SQLcode

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(strSpp="057",dfo=FALSE) {
  if (dfo) {
    bootTab=getBootRuns(strSpp)
    print(bootTab) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
})
```

`getCatch` *Get Catch from Various Databases and Create One Dataset*

Description

Extract catch records for a species from various databases and combine them into one catch file.

Usage

```
getCatch(strSpp="394", dbs=c("gfb","gfc","pht","fos"),
   sql=FALSE, proBio=TRUE, uid=Sys.info()["user"], pwd=uid,
   ioenv=.GlobalEnv)
```

Arguments

`strSpp`	character string specifying a species code.
`dbs`	character string vector of 3-letter codes specifying DFO databases. Currently, only the following are allowed: `"gfb"` = `GFBioSQL` (survey catch data), `"gfc"` = `GFCatch` (primarily trawl data), `"pht"` = `PacHarvest` (trawl data only), `"fos"` = `GFFOS` (all fisheries).
`sql`	logical: if `TRUE`, grab the pre-queried catch files stored as binaries.
`proBio`	logical: if `TRUE` run the function `processBio` on the combined dataset to add fields: `srfa` = slope rockfish assessment areas, `srfs` = slope rockfish assessment subareas or gullies, `popa` = Pacific ocean perch areas.
`uid, pwd`	user ID and password for Oracle DB account authentication.
`ioenv`	input/output environment for function input data and output results.

Details

This function provides a quick method of extracting contiguous datasets and combining them into one data object. The primary use for this dataset is to create the catch input for the function `weightBio`. Currently, `getCatch` only functions for the trawl fishery as the hook and line databases are messy and not contiguous. For a comprehensive catch reconstruction, see `buildCatch`.

Value

Returns a data object of survey catch records `Scat123.wB` and commercial catch records `Ccat123.wB` resulting from the merge of query results to various DFO databases. The results are also written to system binary files `Scat123.wB.rda, Ccat123.wB.rda`, where `123` specifies the species code and `wB` specifies creation for use in the function `weightBio`.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`buildCatch`, `processBio`, `weightBio`
Available SQL queries: `SQLcode`

getData *Get Data from a Variety of Sources*

Description

Attempt to get data from a variety of types (e.g., data objects/files, queries/tables) and remote sources (ORA, SQL, MDB, DBF, XLS, FILE).

Usage

```
getData(fqtName, dbName="PacHarvest", strSpp=NULL, server=NULL,
     type="SQL", path=getwd(), trusted=TRUE, uid="", pwd="",
     noFactors=TRUE, noLogicals=TRUE, rownum=0, mindep=NULL,
     maxdep=NULL, surveyid=NULL, survserid=NULL, fisheryid=NULL,
     logtype=NULL, doors=NULL, speed=NULL, mnwt=NULL, tarSpp=NULL,
     major=NULL, top=NULL, dummy=NULL, senv=NULL, tenv=.GlobalEnv, ...)
```

Arguments

fqtName	string specifying name of file, query, or table; can also be an explicit SQL statement when TYPE=SQLX or TYPE=ORAX.
dbName	string specifying the name of a remote database. Types supported: XLS (Excel), MDB (ACCESS), SQL (Sequel Server), ORA (Oracle).
strSpp	string specifying species code for the SQL variable @sppcode. If not specified, a table or query is retrieved.
server	string specifying the name of a remote server (e.g., SVBCPBSGFIIS for an SQL server, GFSH or ORAPROD for an Oracle server.
type	type of file: FILE = local data object, a saved binary file *.rda, a dumped ASCII file .r, a comma-delimited file .csv or .txt. Specify the file name without the extension. XLS = Microsoft Excel spreadsheet (specific worksheet is specified by fqtName). DBF = Dbase IV table. As each .dbf file acts like a separate table, use fqtName to specify the .dbf file without its extension. MDB = Microsoft ACCESS database query or table. SQL = SQL Server SQL query (code file) or table. ORA = Oracle SQL query (code file) or table. SQLX = SQL Server SQL query code (direct expression). ORAX = Oracle SQL query code (direct expression).
path	string specifying path to local file, MDB database, or SQL query code file.
trusted	logical: if TRUE, allow SQL Server to use a trusted DFO login ID.
uid, pwd	user ID and password for authentication (if required).
noFactors	logical: if TRUE, convert all factor fields to character fields.
noLogicals	logical: if TRUE, convert all logical fields to characters "T" or "F".
rownum	numeric indicating how many rows of a table to return. The default 0 means all rows are returned (entire table). This argument only affects downloads of remote database tables.
mindep	numeric specifying minimum depth for the SQL variable @mindep.
maxdep	numeric specifying maximum depth for the SQL variable @maxdep.
surveyid	numeric specifying survey ID in GFBio for the SQL variable @surveyid.

survserid	numeric specifying survey series ID in GFBio for the SQL variable @survserid.
fisheryid	numeric specifying fishery ID number for the SQL variable @fisheryid.
logtype	string specifying log type code for the SQL variable @logtypeval.
doors	numeric specifying door spread width for the SQL variable @doorsval.
speed	numeric specifying vessel trawling speed for the SQL variable @speedval.
mnwt	numeric specifying mean weight (g) of a species for the SQL variable @mnwt.
tarSpp	string specifying species code(s) for the SQL variable @tarcode.
major	numeric specifying PMFC major area codes(s) for the SQL variable @major.
top	numeric specifying top *N* records for the SQL variable @top.
dummy	numeric or character to use *ad hoc* wherever the SQL variable @dummy appears.
senv	source environment from which fqtName will be retrieved (if available). Only used when type=FILE.
tenv	target environment to which the result object PBSdat will be sent.
...	additional arguments for RODBC's function sqlQuery (specifically if the user wishes to specify rows_at_time=1).

Details

The data table retrieved is placed in the data frame object PBSdat. If the type is MDB, the query/table name and the database name are attached as attributes. If the type is SQL or ORA, the DB name, query name, and SQL code are attached as attributes.

Value

No value is explicitly returned; data frame is retrieved and place in the target environment as an object called PBSdat.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getCatch, getFile, listTables
Available SQL queries: SQLcode

---

getFile *Get Data from Binaries or ASCII*

Description

Attempt to get data from the specified sources (without specifying the extension).

Usage

```
getFile(..., list=character(0), path=getwd(),
   senv=NULL, tenv=.GlobalEnv, use.pkg=FALSE, reload=FALSE,
   try.all.frames=FALSE, use.all.packages=FALSE)
```

Arguments

...	a sequence of names or literal character strings.
list	a character vector of potential objects.
path	string specifying path to data files *.rda and *.r.
senv	source environment from which specified object(s) are retrieved.
tenv	target environment to which specified object(s) will be sent.
use.pkg	logical: if TRUE look for data binaries in loaded packages.
reload	logical: if TRUE, force a reloading of the data object into R's memory.
try.all.frames	logical: if TRUE, look for named object(s) in all frames, starting from the parent frame and working backwards.
use.all.packages	logical: if TRUE, look for data binaries in all the packages installed in the user's R library folder.

Details

The function was originally designed to retrieve data from system files; however, it now includes the ability to specify an R source environment (senv) from which objects might be available. Their is also an argument that specifies a target environment (tenv) to place the objects. Be aware that the transfer of objects between environments overwrites existing objects. Also, when senv is specified, the function currently looks no further for data. To search for data automatically, set senv=NULL, which is the default (following two paragraphs apply).

If a data object with this name exists in the specified target environment, the function will not attempt to reload it unless the users specifies reload=TRUE. Failing to find an R data object in memory (frames), the function tries the binary libraries for package data objects, but only if the user specifies use.pkg=TRUE. Even with this option, the code only looks in those packages that are currently loaded and attached. If the user wishes to search all R libraries installed on his system, he must further specify use.all.packages=TRUE. Searching through all libraries can be slow if the user has installed many packages.

Thereafter, the code searches for remote data files in the following order: (i) local binary files with the extension .rda, (ii) ASCII data files created using dump and with the extension .r, (iii) comma-data delimited files .csv, (iv) and ordinary text files .txt.

Value

No value is explicitly returned, data file is retrieved and placed in the target environment.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getData, getName

`getName` *Get String Names from Literals or Named Objects*

Description

Get string names from user supplied input. If the name supplied exists as an object in the parent frame, the object will be assessed for its potential as a source of names.

Usage

```
getName(fnam)
```

Arguments

`fnam`	file name(s) specified directly or through names in objects.

Details

If `fnam` exists as a list, the function returns the names of the list.
If `fnam` exists as a string vector, the function returns the strings in the vector.
If `fnam` does not exist as an object, it simply returns itself as a string.

Value

A vector of string names.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`getFile`, `getData`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  cat("Data object 'swiss' doesn't appear in the parent frame\n")
  print(getName(swiss))
  swiss=swiss
  cat("And now it does, so it acts like a source of names\n")
  print(getName(swiss))
  invisible() }
pbsfun()
})
```

getPMR *Get (p, mu, rho) Values for Survey Strata*

Description

Get p, mu, and rho values for populations by strata from survey data housed in GFBioSQL using the query gfb_pmr.sql stored in library/PBStools/sql.

Usage

```
getPMR(strSpp="396", survID=c(1,2,3,121,167))
```

Arguments

strSpp	string specifying species code (see species$code).
survID	numeric survey ID assigned to each survey by the GFBio team.

Details

The function executes the SQL query gfb_pmr.sql for each survey ID i:
`getData("gfb_pmr.sql","GFBioSQL",strSpp,path=.getSpath(),surveyid=i)`
and collects the results in a data frame.

Value

A data frame where each row describes the population parameters (p,mu,rho) for each unique survey ID and survey stratum. Attributes include fqt (query name), strSpp (string species code), h (strata), surveys.all (data frame from a call to getBootRuns), and surveys.selected (records from surveys.all with the specified survID).

Additionally, a global list object PBStool contains:

pmrTab	the data frame output object described above.
sTime	system and elapsed time to get the pmr data.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

getBootRuns, calcPMR, sampBG, showIndices
Available SQL queries: SQLcode

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(dfo=FALSE){
  if (dfo) {
    pmrTab=getPMR(strSpp="396", survID=c(1,121))
    print(pmrTab) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
})
```

`glimmer` *Perform a Standardised GLM Analysis*

Description

Perform a standardised GLM analysis and plot the results.

Usage

```
glimmer(file, facs=c("year","month","depth","vessel"),
     afld="latitude", yrs=1996:2010, mos=1:12, mo1=1,
     dbrks=seq(100,500,100), gear=c(0,1,3), catch="catch",
     areas=NULL, Vpro=0.03, Uplot=TRUE, Upanel=0.4,
     Ionly=FALSE, Imax=NULL, crange=c(-2,2), erange=c(0,5),
     facmin=10, effmax=1440, vline=NULL, wmf=FALSE, ioenv=.GlobalEnv)
```

Arguments

`file`	name of file or data object containing fields for GLM analysis: `year, month, depth, latitude, vessel, effort, catch`. See the SQL file `pht_glm.sql`.
`facs`	string vector: factors (fields) other than those that describe area.
`afld`	string scalar: factor describing spatial area, choose one of: `c("latitude", "major", "minor", "locality", "pfma", "pfms", "srfa", "srfs", "popa")`.
`yrs`	numeric vector of years.
`mos`	numeric vector of months.
`mo1`	first month in year (*e.g.*, 1=calendar year, 4=fishing year).
`dbrks`	either a numeric vector of two or more cut points or a single number (greater than or equal to 2) giving the number of intervals into which depth is to be cut.
`gear`	numeric vector: gear codes for trawl fishery (*e.g.*, 1=bottom, 3=midwater).
`catch`	string scalar: field containing catch numbers.
`areas`	vector: specific area names/codes within `afld`.
`Vpro`	numeric scalar: minimum vessel's proportion of catch to total catch.
`Uplot`	logical: if `TRUE`, calculate and plot CPUE index and effects on the CPUE index. If `FALSE`, plot the parameter coefficients.
`Upanel`	numeric scalar: proportion of space to plot index panel if `Uplot=TRUE`.
`Ionly`	logical: if `TRUE`, plot only the first panel (index).

Imax	numeric scalar: maximum index value for annual index plot.
crange	numeric vector: y-limits (min,max) for coefficient plots.
erange	numeric vector: y-limits (min,max) for factor effect plots.
facmin	numeric vector: minimum number of values before a category is used in the factor. Separate minima can be specified for each factor.
effmax	effort maximum; all records with effort greater than this are discarded.
vline	numeric vector: value(s) on x-axis to place a vertical dashed line.
wmf	logical: if TRUE, send plot to a .wmf file.
ioenv	input/output environment for function input data and output results.

Details

This function calculates parameter coefficients using the lm and aov functions in the stats package. (The aov results are not currently used for anything.). See Section 7 of Schnute *et al.* (2004) for a detailed discussion on 'Standardized CPUE Analyses'.

Value

List PBStool containing:

dat	Data frame of processed data to pass to lm and aov.
coeffs	Named vector of coefficients calculated by lm.
lmres	Results of lm analysis
lmsum	Summary (list) of lmres with correlation=TRUE.
aovres	Results of aov analysis
facmin	Vector of minimum number of records per category for the specified factor.
year	Data frame of parameter coefficients and standard errors for year factor.
month	Data frame of parameter coefficients and standard errors for month factor.
depth	Data frame of parameter coefficients and standard errors for depth factor.
latitude	Data frame of parameter coefficients and standard errors for latitude factor.
vessel	Data frame of parameter coefficients and standard errors for vessel factor.
brR	Slope (b), annualized rate (r), and accumulated rate (R) from the lm-fit through the $\log_2$ annual indices.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Schnute, J., Haigh, R., Krishka, B., Sinclair, A. and Starr, P. (2004) The British Columbia longspine thornyhead fishery: analysis of survey and commercial data (1996–2003). *Canadian Science Advisory Secretariat, Research Document* **2004/059**. 75 p.

See Also

getData
Available SQL queries: SQLcode
stats: lm, aov

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(testdatC,envir=.PBStoolEnv); x = testdatC
  glmdat=data.frame(
    year=as.numeric(substring(x$date,1,4)),
    month=as.numeric(substring(x$date,6,7)),
    depth=x$fdep,latitude=x$Y,vessel=x$cfv,
    effort=x$eff,catch=x$POP)
  attr(glmdat,"spp")="396"
  glimmer(glmdat,Vpro=.025,dbrks=seq(100,500,50))
  invisible() }
pbsfun()
})
```

---

histMetric *Plot Matrix of Histograms for Metric*

Description

Plot a matrix of histograms or cumulative frequencies for a specified biology metric by year and trip type.

Usage

```
histMetric(dat=pop.age, xfld="age", xint=1, minN=50,
  ttype=1:4, year=NULL, plus=NULL, ptype="bars", allYR=FALSE, allTT=FALSE,
  major=NULL, minor=NULL, locality=NULL, srfa=NULL, srfs=NULL,
  xlim=NULL, ylim=NULL, pxlab=c(.075,.85),axes=TRUE,
  fill=FALSE, bg="grey90", fg="black",
  hrow=0.75, hpage=8, wmf=FALSE, pix=FALSE, ioenv=.GlobalEnv)
```

Arguments

`dat`	data file of biology metrics (e.g., `pop.age`).
`xfld`	string specifying the field containing the metric of interest.
`xint`	x-interval specfying the bar width.
`minN`	minimum number of specimens per matrix cell to display histogram.
`ttype`	explicit trip type codes: 1 = non-observed domestic commercial, 2 = research, 3 = survey, 4 = observed domestic commercial.
`year`	set of years to display (handy when many years available).
`plus`	plus class value (accumulates all `xfld >= plus`).
`ptype`	string specifiying the plot type: `"bars"` histogram-like bars; `"csum"` cumulative sum curves with median point estimated; `"vlin"` vertical lines.
`allYR`	logical: if `TRUE`, force the display of all years either specified or available.
`allTT`	logical: if `TRUE`, force the display of all trip types either specified or available.
`major`	numeric code for major PMFC areas.
`minor`	numeric code for minor PMFC areas.

locality	numeric code for fishing localities in minor PMFC areas.
srfa	alpha-numeric code for slope rockfish assessment areas.
srfs	character code for slope rockfish assessment subareas.
xlim	explicit x-limits.
ylim	explicit y-limits.
pxlab	relative position on histogram plots to add summary information.
axes	logical: if FALSE, suppress plotting most of the axes (primarily for dense cumulative sum plots).
fill	logical: if TRUE, fill bars (if selected) with bg colour.
bg	background colour to fill bars or points.
fg	foreground colour for bar borders or lines.
hrow	height of each row in inches if figure is sent to an image file.
hpage	height of page in inches if figure is sent to an image file. The maximum height of a page will be the minimum of hpage and m*hrow, where m is the number of rows in a multi-panel figure.
wmf	logical: if TRUE, send plot to a .wmf file.
pix	logical: if TRUE, send plot to a .png file.
ioenv	input/output environment for function input data and output results.

Details

The function displays the relative frequency of a specified metric like length by year (row) and trip type (column). Cells where the number of specimens <= minN are left blank. If only one trip type is specified or available, the cells display the annual histograms by row with an automatically calculated number of columns.

Value

No value is explictly returned by the function. A global list object PBStool provides the following:

dat	Massaged input data.
xy	Output from the function hist().
YRS	Years used in the plots.
ttype	Trip type used in the plots.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.

See Also

histTail, getFile
PBSmodelling: drawBars
graphics: polygon

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  histMetric(xfld="len",year=2003:2007,xlim=c(0,60),fill=TRUE,bg="moccasin",xint=2)
  invisible() }
pbsfun()
})
```

`histTail` *Plot Histogram with Optional Tail Zoom*

Description

Plot a histogram for a specified biology metric with an option to zoom in on the tail detail.

Usage

```
histTail(dat=pop.age, xfld="age", tailmin=NULL,
    bcol="gold", tcol="moccasin", hpage=3.5,
    wmf=FALSE, pix=FALSE, ioenv=.GlobalEnv, ...)
```

Arguments

`dat`	data file of biology metrics (e.g., `pop.age`).
`xfld`	string specifying the field containing the metric of interest.
`tailmin`	numeric value of `xfld` to start the tail plot.
`bcol`	fill colour for the main histogram bars.
`tcol`	fill colour for the tail histogram bars.
`hpage`	height of page in inches if figure is sent to an image file.
`wmf`	logical: if `TRUE`, send plot to a `.wmf` file.
`pix`	logical: if `TRUE`, send plot to a `.png` file.
`ioenv`	input/output environment for function input data and output results.
`...`	additional arguments to pass to `truehist` and `hist`.

Details

The function displays the relative frequency of a specified metric like age. Its functionality is minimal, providing a quick histogram for a specified field in a data frame. The tail histogram appears as an inset on the right showing a *zoom-in* of the frequency of the upper tail data.

The user can specify whether to plot the relative probability distribution `prob=TRUE` or a frequency distribution `prob=FALSE`. The default provides the former for the main histogram and the latter for the tail zoom-in.

Value

No value is explictly returned by the function. A global list object `PBStool` provides the following:

`x`	Vector of raw data (`xfld` in `dat`) used in the main histogram.
`nx`	Total number of observations *N*.
`nz`	Number of observations in tail *n*.
`brks`	Breaks used to get histogram bars in tail.
`xlab`	Label used for x-axis.
`ylab`	Label used for y-axis.

See Also

histMetric
PBSmodelling: evalCall
MASS: truehist
graphics: hist

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  histTail(xfld="age",tailmin=60,prob=FALSE)
  invisible() }
pbsfun()
})
```

---

imputeRate *Impute Rate of Return for an Investment*

Description

Impute the rate of return for an investment that experiences regular or irregular contributions and/or withdrawals.

Usage

```
imputeRate(qtName="Ex03_Portfolio", dbName="Examples",
     AID=1, pathN=2, hnam=NULL)
```

Arguments

qtName	Name of query or table in a Microsoft ACCESS file (.mdb).
dbName	Name of the Microsoft ACCESS file (.mdb).
AID	Numeric specifying account ID.
pathN	Numeric specifying path: 1 = current working directory, 2 = SQL directory .../library/PBStools/sql.
hnam	Name of a history file.

Details

This function creates an interactive GUI that can be used to impute the rate of return for an investment account or for a simulated investment. The code adopts the formula for the "internal rate of return" used in Microsoft Excel.

The input data must contain the fields:
AID.....Account ID number;
date....Date of the account valuation;
value...Value of the account as of date;
cont....Total contributions/withdrawals from the previous date up to and including the current date.

The GUI controls:

Data	Open the .mdb database.
R code	View the function imputeRate.
Window	View the *window description file*.

MDB	Microsoft Access database name (no extesion).
>	Displays available .mdb files on the specified path (below).
Table	Table or query that contains account information.
>	Displays tables and queries in MDB, choose one.
GET	Get the data from the chosen table.
MDB path	Choice of cwd (current working directory) or sql (package SQL directory).
Inputs	
Account #	Account ID number.
parVec	Data frame specifying val, min, max, active for parameter rate.
autoD	If TRUE, get the date limits of the account automatically.
Start date	Starting date of the investment time series to calculate rate.
End date	Ending date of the investment time series to calculate rate.
period	Periods in data source corresponding to the date limits specified.
Estimation	
Method	Choose one of various non-linear estimation routines.
Controls	Various controls used by calcMin.
Reset	Button resets parVec and output boxes (below).
ReInit	Button sets parVec's val to the last estimated prate (below).
RUN	Runs the rate estimation for the investment interval chosen.
Simulate	
No Yes Again	Choose to simulate investment data using a random pareto distribution.
start	Starting value of the investment.
rate	True rate of return per period.
nper	Number of periods to run the simulation.
up	Proportion of the time that a contribution is made vs. a withdrawal.
k	Pareto distribution parameter (volatility decreases as k increases).
Outputs	
decimal places	Number of decimal places for display output.
Iters Evals	Number of iterations and evaluations for the estimation.
prate arate	Starting period rate and annualised rate.
Ctime Etime	Computer and evaluation times (seconds).
prate arate	Estimated period rate and annualised rate.
Fmin0 Fmin	Initial function evaluation and the final function value at minimization.
AIC AICc	Aikike Information Criterion (model fit) and corrected AIC.
message	Message box area reporting results of the minimization.

See Also

calcMA, glimmer, trend

---

isThere *Check Whether an Object Physically Exists*

Description

Check to see whether objects physically exist in the specified environment. This differs from the function exists in that the latter sees objects across environments.

Usage

```
isThere(x, envir=parent.frame())
```

Arguments

x	a variable name (given as a character string).
envir	specify an environment to look in (e.g., sys.frame(sys.nframe()))

Details

This function looks in the specified environment and the object must be physically present to elicit a TRUE response. This contrasts with the base function exists.

Value

A Boolean vector.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

getFile

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  cat("Data object 'swiss' appears to exist in pos=0\n")
  cat(paste("   exists(\"swiss\",envir=sys.frame(0))",
    exists("swiss",envir=sys.frame(0))),"\n")
  cat("But it isn't really there...\n")
  cat(paste("   isThere(\"swiss\",envir=sys.frame(0))",
    isThere("swiss",envir=sys.frame(0))),"\n")
  invisible() }
pbsfun()
})
```

---

lenv *Get the Local/Parent/Global Environment*

Description

Shortcut functions to get the local (lenv), parent (penv), and global (genv) environments.

Usage

```
lenv()
penv()
genv()
```

Value

An environment

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

base: environment

---

listTables *List Tables in Specified SQL/ORA/MDB Database*

Description

List the tables in a specified SQL, Oracle or Microsoft ACCESS database. User can choose table type and/or search table names through pattern matching.

Usage

```
listTables(dbName, pattern=NULL, path=getwd(), server=NULL,
   type="SQL", ttype=NULL, trusted=TRUE, uid="", pwd="",
   silent=FALSE, tenv=.GlobalEnv)
```

Arguments

dbName	string specifying the name of a remote SQL/ORA or local MDB database.
pattern	string specifying search pattern for table names.
path	string specifying path to MDB database.
server	string specifying server name (*e.g.*, GFDB, SVBCPBSGFIIS, ORADEV, or ORAPROD).
type	type of server: SQL, ORA for Oracle, or MDB for Microsoft ACCESS.
ttype	table type: SEQUENCE, SYNONYM, SYSTEM TABLE, TABLE, or VIEW; default NULL shows all table types.
trusted	logical: if TRUE, allow SQL/ORA server to use a trusted DFO login ID.
uid, pwd	user ID and password for authentication (if trusted=FALSE).
silent	logical: if TRUE then table names are not printed to the console.
tenv	target environment to which the result object PBSdat will be sent.

Details

The results of RODBC::sqlTables are dumped to a data frame object called PBSdat. If no pattern pattern is specified, all table names in dbName are returned. If pattern is provided, only table names containing the pattern are returned. Pattern matching is case-sensitive. Additionally, the user can choose table types, depending on the server type.

Value

Silently returns a vector of table names.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getFile, getData

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type,dfo=FALSE) {
  if (os=="windows") {
    if(dfo) {
      cat("Tables in 'PacHarvest' matching pattern 'Species'\n")
      listTables("PacHarvest",pattern="Species") }
    else {
      cat("Tables in 'Examples.mdb'\n")
      listTables("Examples",type="MDB",ttype=c("TABLE","VIEW"),path=.getSpath()) } }
  else showMessage("If logged onto DFO network, set argument 'dfo=T'")
  invisible() }
pbsfun()
})
```

---

makeCATtables *Make Catch Tables from Commercial Fishery Data*

Description

Create a CSV file containing tables of commercial catch by management fishing year. Various fisheries can be specified.

Usage

```
makeCATtables(strSpp, comm=1:7, path=.getSpath(), pout=FALSE)
```

Arguments

strSpp	string specifying species code used in GFBio.
comm	numeric code specifying various fisheries: 1 = GF Catch (GFC) trawl tows; 2 = PacHarvest (PHT) trawl tows (observer or fisher logs); 3 = PacHarvHL (PHHL) Zn hook and line from validation records; 4 = PacHarvHL (PHHL) Zn hook and line from fisherlog records; 5 = PacHarvHL (PHHL) Schedule II hook and line from validation records; 6 = PacHarvHL (PHHL) Schedule II hook and line from fisherlog records; 7 = PacHarvHL (PHHL) Halibut longline from validation records.
path	path leading to SQL code (available in **PBStools**'s directory sql).
pout	logical, if TRUE then print table results on the command line (results are sent to the output CSV regardless).

Details

The code scrolls through the specified fisheries, issuing the appropriate SQL query for each catch summary. The results are placed in a text file called cattab-XXX.csv, where XXX = the strSpp specified.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getData, makePMRtables
Available SQL queries: SQLcode

---

makeLTH *Make a Longtable Header for Sweave*

Description

Make a longtable header for printing an xtable in Sweave, source:
http://tex.stackexchange.com/questions/41067/caption-for-longtable-in-sweave?rq=1

Usage

```
makeLTH(xtab.table, table.caption, table.label)
```

Arguments

xtab.table	table object created by xtable.
table.caption	table caption for the xtable (no need to specify this in the xtable command).
table.label	table label used to reference the table from other places in the document.

Details

This code was provided by a clever Swedish fellow (pseudonym = chepec) to elegantly break a longtable across pages by specifying *Continued on next page* as a footer where the table breaks and changing the caption on the following page to "Table xx – continued from previous page".

Value

A longtable header formatted to the specifications of print.table's argument add.to.rows.

Note

Note the percent sign paired with the number of rows in the table; it will cause that trailing \hline command in your .tex file to be commented out. Not in any manual I've seen, by the way. Just a trick I use. (chepec, stackexchange user: 10824)

Author(s)

Modified by Rowan Haigh, Pacific Biological Station, Fisheries & Oceans Canada, Nanaimo BC
from user 10824 (chepec) on tex.stackexchange (http://tex.stackexchange.com/users/10824/chepec)

See Also

PBSawatea: runSweaveMCMC
xtable: xtable, print.xtable

Examples

```
## Not run:
<<echo=FALSE, results=tex>>=
# In a Sweave document, one might use makeLTH like this:
require(xtable)
# need the column name formats and the data to be character:
habtab1 = read.csv("concur396-3CD-gear1.csv",check.names=FALSE,colClasses="character")
ncol   = dim(habtab1)[[2]] - 1
saraSpp=c("027","394","410","424","435","437","440","442","453")
rows.sara = (1:nrow(habtab1))[is.element(habtab1$Code,saraSpp)] - 1
xtab1 = xtable(habtab1,align=paste(c("cc",rep("r",ncol)),collapse=""))
longtable.header=makeLTH(xtab1,"Top 25 species by catch weight.", "tab:concur396-3CD-gear1")
print(xtab1,
   floating = FALSE, # longtable never floats
   hline.after = NULL,
   add.to.row = list(pos = list(-1, rows.sara, nrow(xtab1)),
      command = c(longtable.header, "\\rowcolor{rowclr}", "%")),
   include.rownames = FALSE,
   include.colnames = FALSE,
   type = "latex",
   tabular.environment = "longtable",
   sanitize.text.function = function(x){x},
   math.style.negative = FALSE)
@

## End(Not run)
```

---

makePMRtables *Make pmr Tables for Surveys*

Description

Create `.csv` files containing (p, mu, rho) values for strata in the specified surveys.

Usage

```
makePMRtables(strSpp, surveys=list(qcss=c(1,2,3,121,167),
     wcvis=c(16,70,126), wqcis=c(79,122,129)),
     qName="gfb_pmr.sql", path=.getSpath())
```

Arguments

`strSpp`	string specifying species code used in `GFBioSQL`.
`surveys`	named list object containing vectors of survey ID numbers, where the names describe the surveys (e.g., `qcss` = Queen Charlotte Sound Synoptic; `wcvis` = West Coast Vancouver Island Synoptic; `wqcis` = West Queen Charlotte Islands Synoptic).
`qName`	string specifying name of the SQL query.
`path`	path leading to the SQL code `qName`.

Details

The code scrolls through the survey IDs, issuing sequential SQL queries to the DFO database `GFBioSQL`, and collects the debris in `.csv` files, one for each name in the list `surveys`.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

calcPMR, makeCATtables
Available SQL queries: SQLcode

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(strSpp="451",dfo=FALSE) {
  if (dfo) makePMRtables(strSpp)
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
})
```

---

makeSSID *Make a Data Object of Survey Series Information*

Description

Make a data object of survey series information called ssid for use in the package **PBSdata**.

Usage

```
makeSSID()
```

Details

The function imports the SQL table SURVEY from the DFO database GFBioSQL using:
`getData("SURVEY","GFBioSQL")`

The survey information is then summarised by its series number and saved to a binary .rda file.

Value

A list of data frames, one for each database, of survey series.

gfb (GFBioSQL):
Data frame with 118 series (rows) and 5 fields.

ssid	survey series ID number; also repeated as row names.
desc	survey description
nsurv	number of surveys in the series
original	logical: if TRUE, the series comprises an original set of surveys; if FALSE, the series has been derived for some special purpose.
trawl	logical: if TRUE, the series comprises trawl surveys.

Note

The data object `ssid` can be found in the package **PBSdata**.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`ssid`, `getData`

---

mapMaturity *Map Maturity by Month*

Description

Map each maturity code to show its relative occurrence by month.

Usage

```
mapMaturity(dat=pop.age, strSpp="", mats=1:7, sex=1:2,
    brks=c(0,.05,.1,.25,.5,1),
    clrs=c("aliceblue","lightblue","skyblue3","steelblue4","black"),
    hpage=6, wmf=FALSE, pix=FALSE, ioenv=.GlobalEnv)
```

Arguments

`dat`	biological data set with fields `mat` and `age`.
`strSpp`	string code for species. If dataset has attribute `spp`, this will be used before `strSpp`.
`mats`	maturity codes to map.
`sex`	sex code: 1 = males, 2 = females.
`brks`	relative breaks from 0 to 1 to partition the maturity data.
`clrs`	colours to use for relative intervals defined by `brks`.
`hpage`	height of page in inches if figure is sent to an image file.
`wmf`	logical: if `TRUE`, send plot to a `.wmf` file.
`pix`	logical: if `TRUE`, send plot to a `.png` file.
`ioenv`	input/output environment for function input data and output results.

Details

For each maturity code the function plots a row of monthly boxes colour- coded by the relative occurrence of the code in that month compared to the year.

Value

No value is explictly returned by the function. A global list object `PBStool` provides the following:

`dat`	Qualified data used for the analysis.
`xlim`	Limits of the x-axis.
`ylim`	Limits of the y-axis.
`x`	Current value of x.
`y`	Current value of y.
`sdat`	Last subset of `dat`.
`mday`	Number of days in each month.
`mcut`	Cut points in days for months in a year from 0 to 365.
`idat`	Last subset of `sdat`.
`ibin`	Days of the month (observations) occuring in each month.
`icnt`	Relative frequency of observations in each month.
`iclr`	Colour code assigned to each month.

Note

A suitable data set can be obtained by running the SQL query `gfb_bio.sql`.

```
getData("gfb_bio.sql","GFBioSQL","396",path=.getSpath())
```

See Also

`pop.age`, `estOgive`, `getData`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  mapMaturity(brks=c(0,.01,.05,.25,.5,1),sex=1:2)
  invisible() }
pbsfun()
})
```

`PBStools` *PBS Tools*

Description

PBStools provides tools for stock assessments, species-at-risk requirements, and stakeholder requests. The package depends on four other R packages: **PBSmapping**, **PBSmodelling**, **PBSdata**, and **RODBC**.

PBStools contains the following functions:

Utilities

`biteData`	Subset a data matrix/frame using a vector object.
`chewData`	Remove sparse category records from data matrix/frame.
`collectFigs`	Collect postscript figures into one document.
`confODBC`	Configure an ODBC user data source name (DSN).
`convFY`	Convert dates into fishing/fiscal years.

convYM	Convert date limits into expanded year-months.
convYP	Convert dates into binned year periods.
createDSN	Create user DSNs for PBS groundfish databases.
crossTab	Use package 'reshape' to summarize z using crosstab values y.
dtget	Get/print objects from or put objects into temporary work environment (.PBSdataEnv).
fitLogit	Fit binomial data using logit link function.
flagIt	Label a coordinate (a, b) using a diagonal line.
getData	Get data from a variety of sources.
getFile	Get data from binaries or ASCII.
getName	Get string names from literals or named objects.
isThere	Checks whether an object is physically present in a specified environment.
lenv	Get the local/parent/global environment.
listTables	List tables in specified SQL/ORA database.
makeLTH	Make a longtable header for Sweave.
prime	Report the prime numbers given an integer vector.
revStr	Reverse a string set.
runModules	Run GUI modules included in PBS Fishery.
scaleVec	Scale a vector to span a target minimum and maximum.
showError	Display an error message on the current device.
showMessage	Display an information message on the current device.
spooler	Spool information specified by a list into a new field of an existing data frame.
stdConc	Standardise a chemical concentration.
toUpper	Capitalise first letters of words.
ttget	Get/print objects from or put objects into temporary work environment (.PBStoolEnv).
wrapText	Wrap, indent, and prefix a long text string.
zapDupes	Remove records with duplicated indices from a data frame.

Biology

calcLenWt	Calculate length-weight relationship for a fish.
calcSG	Calculate growth curve using Schnute growth model.
calcVB	Calculate fits using a von Bertalanffy growth model.
compCsum	Compare cumulative frequencies (e.g., of ages).
estOgive	Estimate ogive curves for maturity.
genPa	Generate proportions-at-age using catch curve composition.
histMetric	Plot a matrix of histograms for a specified metric.
histTail	Plot histogram with optional tail zoom.
mapMaturity	Map species maturity by month.
plotProp	**GUI:** Create bubble plots showing proportions-at-age etc.
predictRER	Predict Rougheye Rockfish from biological data.
processBio	Process results from SQL query gfb_bio.sql.
reportCatchAge	**GUI:** Plot results from Jon Schnute's catch-at-age ADMB report file.
requestAges	Determine which otoliths to sample for ageing requests.
simBSR	Simulate Blackspotted Rockfish biological data.
simRER	Simulate Rougheye Rockfish biological data.
sumBioTabs	Summarize frequency occurrence of biological samples.
weightBio	Weight age/length frequencies/proportions by catch.

Fishery

calcRatio	Calculates ratios of numerator to denominator (e.g., 'discard'/'catch').
dumpMod	Dump catch from modern sources used in catch reconstruction.
dumpRat	Dump catch ratios calculated by a catch reconstruction.
formatCatch	Format table of numeric catch as strings.
getCatch	Get catch records for a species from various databases and combine.
glimmer	Perform a standardised GLM analysis.

`makeCATtables`	Make catch tables from commercial fishery data.
`plotCatch`	Plot catch history as an annual barplot.
`plotConcur`	Plot concurrent species in tows at depth.
`plotFOScatch`	Plot monthly catch from FOS as barplots.
`plotRecon`	Plot reconstructed catch using barplots stacked by PMFC area.
`runCCA`	Catch-curve model based on Schnute and Haigh (2006).
`sumCatTabs`	Summarize catch by year and PMFC from modern catch data used in catch reconstruction.
`trackBycat`	Track annual fish group catches between depth limits.

Survey

`bootBG`	Bootstrap biomass using binomial-gamma population parameters.
`calcMoments`	Calculate survey moments from raw data.
`calcPMR`	Calculate (p, μ, ρ) from a sample population.
`getBootRuns`	Get Norm's survey bootstrap results.
`getPMR`	Get (p, μ, ρ) values for survey strata.
`makePMRtables`	Make (p, μ, ρ) tables for surveys.
`makeSSID`	Make a data object of survey series information.
`sampBG`	Sample from the binomial-gamma distribution.
`showAlpha`	Show quantile confidence levels (α) for bootstraps.
`showIndices`	Show survey indices from bootstrap tables.
`simBGtrend`	Simulate population projection based on prior binomial-gamma parameters.
`trend`	**GUI:** Create boxplots of annual survey data, trend lines, and bootstraps.

Spatial

`calcHabitat`	Calculate potential habitat using bathymetry limits.
`calcOccur`	Calculate percent occurrence of EventData in PolySet.
`calcSRFA`	Determine slope rockfish assessment areas or subareas.
`calcSurficial`	Calculate intersection of surficial geology and bathymetry habitat.
`clarify`	Summarize catch data into CLARA clusters.
`findHoles`	Find holes and place them under correct parents.
`plotGMA`	Plot the Groundfish Management Areas.
`preferDepth`	**GUI:** Plot depth distribution of species.
`prepClara`	Prepare a data object for Clustering Large Applications.
`zapHoles`	Zap (remove) holes from polygons.

Temporal

`boxSeason`	Display seasonal patterns using boxplots.
`calcMA`	Calculate a moving average using a fixed period occurring every x units.
`imputeRate`	**GUI:** Impute the rate of return of an investment.
`plotDiversity`	Plot diversity of phytoplankton samples.
`trackComp`	Track composition of phytoplankton over time.

Catch Reconstruction

`buildCatch`	Build a catch history of BC rockfish 1918–present.
`plotData`	Plot diagnostic data for catch reconstructions.
`plotRecon`	Plot reconstructed annual catch series.

Other package resources:

`../library/PBStools/doc`	includes a User's Guide describing all functions.
`../library/PBStools/ADMB`	ADMB batch files and project files (`pop`, `vonB`).
`../library/PBStools/sql`	SQL code files for querying remote databases.
`../library/PBStools/win`	**PBSmodelling** *window description files* for GUIs.

Note

The temporary working list object PBStool (formerly PBSfish) is now stored in the temporary working environment .PBStoolEnv (see ttget functions to access the object).

See Also

getData, ttget
ServerParlance for differences between SQL Server and Oracle database organisation.
SQLcode for available SQL queries.

---

plotCatch *Plot Catch History as Annual Barplot*

Description

Display catch history where selected columns of catch are stacked by year.

Usage

```
plotCatch(dat="dbr.rem", flds=c("CAtrawl","UStrawl","TotalHL"),
   yrlim=NULL, wmf=FALSE, ioenv=.GlobalEnv, ...)
```

Arguments

dat	catch data object or string representing the object contained in file form (*.rda,*.r.
flds	string specifying fields of catch to use in barplot.
yrlim	numeric limits of the years to display; default uses all in dat.
wmf	logical: if TRUE, send plot to a .wmf file.
ioenv	input/output environment for function input data and output results.
...	additional arguments passed to barplot and legend.

Details

This function simply plots the catch history of the supplied dat file. Each bar shows the cumulative catch from flds. Decadal mean catch is displayed in horizontal bands.

If the user wishes to change the bar colours by passing in a vector col to the function, s/he must also pass in a similar vector fill for the legend.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

getData, getFile, dbr.rem

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  col=fill=c("yellow","green","blue")
  plotCatch(dat=ymr.rem,col=col,fill=fill)
  invisible() }
pbsfun()
})
```

---

plotConcur *Plot Concurrent Species in Fishing Events at Depth*

Description

Create a barplot of the top *N* concurrent species from fishing events capturing `strSpp` between `mindep` and `maxdep` in PMFC `major` areas.

Usage

```
plotConcur(strSpp="410", dbName="PacHarvest",
     mindep=150, maxdep=435, major=NULL, top=NULL, trawl=1,
     saraSpp=c("027","394","410","424","435","437","440","442","453"),
     reset.mf=TRUE, eps=FALSE, pix=FALSE)
```

Arguments

`strSpp`	string code specifying species of interest (often a Hart code for fish).
`dbName`	string name of database with species catch information. If `dbName="PacHarvest"`, then the `SQL` query `pht_concurrent.sql` is run. If `dbName="PacHarvHL"`, then the `SQL` query `phhl_concurrent.sql` is run.
`mindep`	minimum depth (m) where tows occur.
`maxdep`	maximum depth (m) where tows occur.
`major`	PMFC major areas; default selects `major=c(1,3:9)`.
`top`	top (first) *N* records to return from an `SQL` query.
`trawl`	trawl type code usually specified by gear subtype code: `1` = bottom, `2` = shrimp, `3` = midwater
`saraSpp`	string codes specifying species listed by COSEWIC as either: "Special Concern", "Threatened", or "Endangered".
`reset.mf`	logical: if `TRUE`, reset the multiple figure region to `c(1,1)` (only effective for plots sent to the graphics window device).
`eps`	logical: if `TRUE`, send the barplot to an `EPS` file.
`pix`	logical: if `TRUE`, send the barplot to a `PNG` file.

Details

This function uses Norm Olsen's `SQL` code `pht_concurrent.sql` (located in the directory `.../library/PBStools/sql`) to query the observer trawl database `PacHarvest`. A variant of this `phhl_concurrent.sql` taps into the hook and line database `PacHarvHL`.

Value

Invisibly returns a data frame of top *N* species ordered by percent catch weight.

Also, objects of interest are saved to the global list object PBStool.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

preferDepth, getData, toUpper
Available SQL queries: SQLcode

---

plotData *Plot Diagnostic Data for Catch Reconstructions*

Description

Plot data components of a catch reconstruction for user inspection and diagnostic checking.

Usage

```
plotData(x, description="something", col=c("red","coral","gold",
   "green2","skyblue","blue","blueviolet","purple4"), ...)
```

Arguments

x	numeric matrix or data frame (two-dimensional) of z-values for each x and y.
description	short description that will be used to name the output image file.
col	vector of colours to use when plotting columns of the data.
...	additional control data. Currently, the only value used is type="bars".

Details

The function plots z-values of y for each x. The default plot shows lines because a typical x comprises years. An alternative is to show the z-values as bars for each y grouped by x.

Note

This plot is used by the catch reconstruction algorithm which automatically increments the plot number pD that is logged in the global list object PBStool.

Additionally, the catch reconstruction creates a subdirectory CRdiag to which plotData sends image files.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans, Nanaimo BC

See Also

The catch reconstruction function buildCatch and plotRecon

`plotDiversity` *Plot Diversity of Phytoplankton Samples*

Description

Plot sample diversity (*e.g.*, Shanon-Wiener diversity index H) as a barplot with an overlay of points using some other index (*e.g.*, species richness S).

Usage

```
plotDiversity(fqtName="Ex01_Sample_Info",dbName="Examples.mdb",
   type="MDB", path=getwd(), bars="H", pnts="S", xnames=c("SID","Batch"),
   xint=40, clrs=c("skyblue","gold","blue","green4"), addlowess=TRUE,
   f=0.2, pdf=FALSE, wmf=FALSE, ioenv=.GlobalEnv)
```

Arguments

`fqtName`	string specifying name of file, query, or table; (can also be a local data object if `type="FILE"`.)
`dbName`	string specifying the name of a database.
`type`	type of database (e.g., `MDB` = Microsoft ACCESS database query or table).
`path`	directory path to the database.
`bars`	string value denoting field name with an index for barplots.
`pnts`	string value denoting field name with an index for points.
`xnames`	vector of strings denoting fields to use as barplot labels.
`xint`	integer value specifying number of labels to display (useful when number of bars is high).
`clrs`	vector of colours to use for (1) bars, (2) points, (3) lowess line through bars, (4) lowess line through points.
`addlowess`	logical: if `TRUE`, add lowess-fitted lines through bars and points.
`f`	numeric value for smoother span in `lowess`. This gives the proportion of points in the plot which influence the fit at each value. Larger values give more smoothness.
`pdf`	logical: if `TRUE`, send figure to a postscript document file (`.pdf`).
`wmf`	logical: if `TRUE`, send figure to a windows metafile (`.wmf`).
`ioenv`	input/output environment for function input data and output results.

Details

Creates a barplot of one index with overlaid points of another. The arguments `bars` and `pnts` can each be one of H = diversity index, S = species richness, or E = species evenness.

Value

No value is explicitly return; however, objects of interest are stored in the global list object `PBStool`.

Note

The Shannon-Wiener diversity index H is calculated

$$H = \sum_{i=1}^{S} p_i \log_2 p_i$$

where S = species richness (number of species observed) and
p_i = proportion of each species i in the sample.

Species evenness E simply measures how evenly the species are distributed and is calculated

$$E = \frac{H}{\log_2 S}$$

The value of E ranges from 0 for situations with only a single species present to 1 where all species are evenly distributed.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Haigh, R. and Schnute, J.T. (2003) The Longspine Thornyhead fishery along the west coast of Vancouver Island, British Columbia, Canada: portrait of a developing fishery. *North American Journal of Fisheries Management* **23**: 120–140.

See Also

trackComp, boxSeason

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows")
    plotDiversity(bars="H",pnts="S",xnames="Batch",path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
})
```

---

plotFOScatch *Plot Monthly Catch from FOS as Barplots*

Description

Plot monthly catch of specified species from FOS query `fos_catch.sql` as bars stacked by PMFC area.

Usage

```
plotFOScatch(strSpp="453", majors=c(1,3:9), space=0.5,
   fplot="Groundfish Trawl", xlim=c("2007-04","2011-06"),
   fyrM1=4, uid=Sys.info()["user"], pwd=uid, wmf=FALSE, ...)
```

Arguments

strSpp	string specifying species code (species$code)
majors	major PMFC area codes (pmfc$major)
space	space between bars of plot
fplot	name of fishery to plot: "Groundfish Trawl" "Halibut Longline" "Sablefish Trap" "Dogfish & Lingcod" "Rockfish H&L"
xlim	character: lower and upper x-limits of the plot specified as categories "YYYY-MM"
fyrM1	numeric: first month of a fishing year (assumed to run at 12-month intervals)
uid, pwd	user ID and password for Oracle DB account authentication.
wmf	logical: if TRUE, send plot to a WMF file
...	additional arguments for plotFOScatch()

Details

The function uses the SQL query fos_catch.sql to obtain catch from the main groundfish fisheries, and the user can specify which fishery to display. Barplot catch categories are combinations of year-month, with catch from PMFC areas stacked in each category. The total catch is reported at the end of each fishing year.

Additionally catch summaries by fishery, year, and PMFC area are dumped to a file called catFOSfyr.csv.

Internally derived objects of interest (such as catch-by-month and catch-by-year matrices) are save in the global list object PBStool.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

getFile, fos.fid, species, pmfc
Available SQL queries: SQLcode

---

plotGMA *Plot the Groundfish Management Areas*

Description

Plot the groundfish management areas (GMA) for Pacific Ocean Perch (*Sebastes alutus*) and Yellowmouth Rockfish (*S. reedi*) used by Pacific groundfish managers at RHQ in Vancouver.

Usage

```
plotGMA(gma=gma.popymr, xlim=c(-134,-123), ylim=c(48.05,54.95),
     dimfig=c(9,9), pix=FALSE, extra.labels=NULL)
```

Arguments

gma	PolySet of Groundfish Management Areas (e.g. **PBSdata**::gma.popymr GMAs).
xlim	numeric: latitude limits of the map region.
ylim	numeric: longitude limits of the map region.
dimfig	numeric: dimensions of the figure region (inches).
pix	logical: if TRUE, send the plot to a .png file.
extra.labels	EventData object that defines coordinates and labels for additional map annotation.

Details

The code assumes that the PolySet of GMA areas contains an attribute called PolyData that defines colours and labels for each of the GMAs.

If extra.labels = NULL, then no extra labels are added.
If extra.labels = "default", then function-supplied extra labels are added.
Otherwise, a user can specify an EventData object with fields EID, X, Y, and label. The object should have the projection attribute set to the same projection as the map (in this case "LL").

Value

Invisibly returns a list of the PolyData attribute and the EventData object called extra.labels (see Details section above).

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

See Also

The DFO proprietary package **PBSvault**: gma.popymr, pfma
(contact rowan.haigh@dfo-mpo.gc.ca for availability)

---

plotProp *Display Proportions-at-Age Using Bubble Plots*

Description

Display proportions-at-age (or perhaps proportions-at-length or proportions-at-weight) *vs.* year using the **PBSmodelling** function plotBubbles controlled by an interactive GUI.

Usage

```
plotProp(fnam="pop.age", hnam=NULL, ioenv=.GlobalEnv, ...)
```

Arguments

fnam	string specifying file name
hnam	string name of a history file
ioenv	input/output environment for function input data and output results.
...	additional arguments for plotProp.

Details

This function creates an interactive GUI that can be used to display porportions-at-metric data.

The GUI controls:

`Window`	Button to display the window description file.
`Reset`	Reset all settings in the GUI to an initial state.
`File`	Name of data file containing biological (morphometric) measurements.
`Stratify`	Field name of information to stratify proportions data (e.g, `SID`).
`weighted`	Check to weight proportions by the stratifier field (e.g, `catch`).
`Spp`	Species code in field `spp`.
`Sex`	ALL (codes 0:3), Male (1), Female (2), Uncertain (3), Unknown (0).
`field`	Field names of x and y.
`limit1`	Lower value limits of x and y.
`limit2`	Upper value limits of x and y.
`interval`	Grouping interval for x and y values.
`Agg Y`	Logical: if checked, aggregate the y values outside. the y limits to their lower or upper limit.
`Reset`	Reset all settings in the (x,y) table on the left.
`B&B`	Check for specimens with otoliths broken & burnt (ageing method = 3).
`Gear`	Gear types (see `pop.age` for details).
`Trip`	Trip types (see `pop.age` for details).
`Sample`	Sample types (see `pop.age` for details).
`PMFC`	Pacific Marine Fisheries Commission `major` areas.
`SRF`	Slope rockfish assessment areas `srfa`.
`Gullies`	Slope rockfish assessment subareas `srfs`

History widget and various plot controls:

`psize`	Maximum size of bubbles (proportional to the difference of x-width)
`powr`	Transformation of z using an exponential power to use for radius of bubbles (`powr`=0.5 creates bubble areas proportional to z)
`lwd`	Line thickness for bubbles
`pos`	Colour for bubbes representing positive values
`neg`	Colour for bubbes representing negative values
`zero`	Colour for bubbes representing zero values
`line`	Check for horizontal dotted lines across major y-values
`hide0`	Check to hide zero-value bubbles
`ltype`	Legend type: 1 = Detailed showing number of specimes by `ttype`, `sstype`, `gear`, `srfs`, `srfa`, `major` 2 = Number of specimens only 3 = Number of samples only
`GO`	Execute the `plotProp` procedure with all current settings
`Plot`	Replot without recalculating the current settings
`wmf`	Make a windows metafile image file (`.wmf`) from the current display

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

`pop.age`, `getData`,

PBSmodelling: plotBubbles

---

plotRecon *Plot Reconstructed Annual Catch Series*

Description

Plot reconstructed catch using barplots stacked by catch in PMFC areas.

Usage

```
plotRecon(dat=cat440rec, strSpp="440", major=c(1,3:9), fidout=10,
    years=1918:2011, xlab=seq(1920,2010,5),
    eps=FALSE, pix=FALSE, wmf=FALSE, PIN=c(10,5))
```

Arguments

dat	numerix array of annual catch by PMFC area and fishery ID.
strSpp	character string specifying Hart species code (e.g., "440" = Yellowmouth Rockfish).
major	numeric codes specifying PMFC areas (1=4B, 3=3C, 4=3D, 5=5A, 6=5B, 7=5C, 8=5C, 9=5E).
fidout	numeric code indicating for which fishery the catch reconstruction will be plotted (1=Trawl, 2=Halibut, 3=Sablefish, 4=Dogfish-Lingcod, 5=H&L Rockfish, 10=Combined).
years	numeric vector of years for which the catch reconstruction will be plotted.
xlab	numeric vector of years to use for major tick labels.
eps	logical: if TRUE, send the figure to a postscript file (.eps).
pix	logical: if TRUE, send the figure to a raster file (.png).
wmf	logical: if TRUE, send the figure to a windows metafile (.wmf).
PIN	numeric vector indicating figure size (width, height) specified in inches; only applies if the figure is sent to an output file.

Details

See the catch reconstruction algorithm (buildCatch) for more details.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

References

Haigh, R. and Yamanaka, K.L. (2011) Catch history reconstruction for rockfish (*Sebastes* spp.) caught in British Columbia coastal waters. *Canadian Technical Report of Fisheries and Aquatic Sciences* **0000**: xx + 00 p.

See Also

buildCatch, plotFOScatch, plotCatch

`plotTernary` *Plot a Ternary Diagram*

Description

Plot a ternary diagram using compositional data classified into three categories.

Usage

```
plotTernary(x=c(3,2,1), connect=FALSE, show.geometry=TRUE,
   bw=FALSE, eps=FALSE, wmf=FALSE)
```

Arguments

`x`	compositional data: single three-element vector or a matrix/data frame with three columns. The latter can have row and column names which will be used in the plot.
`connect`	logical: if `TRUE`, multiple events (rows) will be connected together in the ternary diagram.
`show.geometry`	logical: if `TRUE`, the vectors p_1, p_2, and p_3 will be drawn, as well as lines connecting (x, y) to each vertex.
`bw`	logical: if `TRUE`, the plot is rendered without colour.
`eps`	logical: if `TRUE`, the plot is sent to a postrcript `.eps` file.
`wmf`	logical: if `TRUE`, the plot is sent to a Windows metafile `.wmf`.

Details

When population components are amalgamated into $g = 3$ groups, vectors of proportions can be portrayed in a graph called a ternary diagram (Aitchison, 1986, p. 5). The diagram begins with an equilateral triangle that has vertices labelled "1", "2", and "3". A vector **p** (p_1, p_2, p_3) of proportions can then be represented as a point within this triangle, where the perpendicular distance to the side opposite vertex "i" is proportional to p_i.

Value

Nada at present.

Author(s)

Rowan Haigh, Pacific Biological Sattion, Fisheries and Oceans Canada, Nanaimo BC

References

Aitchison, J. (1986, reprinted in 2003). The Statistical Analysis of Compositional Data. The Blackburn Press, Caldwell, NJ. 416 p.

Schnute, J.T., and Haigh, R. (2007). Compositional analysis of catch curve data, with an application to *Sebastes maliger*. *ICES Journal of Marine Science* **64**: 218–233.

See Also

`runCCA`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  plotTernary(c(1/2,1/3,1/6))
  invisible() }
pbsfun()
})
```

---

plotTertiary *Plot a Compositional Diagram Within a Polygonal Space*

Description

Plot a compositional diagram using data classified into n categories.

Usage

```
plotTertiary(x=c(100,5,25,10,50), pC=c(0.5,0.5), r=0.5,
   diag=FALSE, eps=FALSE, wmf=FALSE)
```

Arguments

x	compositional data: single n-element vector or a matrix/data frame with n columns. The latter can have row and column names which will be used in the plot.
pC	(x, y) coordinates of the polygon centroid.
r	radial arm length from the centroid to each vertex.
diag	logical: if TRUE, add some diagnostic vectors to the polygon (mostly for debugging).
eps	logical: if TRUE, the plot is sent to a postrcript .eps file.
wmf	logical: if TRUE, the plot is sent to a Windows metafile .wmf.

Details

This function seeks to extend the ternary diagram from 3 to n proportions. However, the geometry of mutiple vertices (squares, pentagons, hexagons, etc.) does not translate into deterministic solutions as it does for triangles (see Aitchison 1986, Schnute and Haigh 2007).

When only one proportion set (vector or single-row matrix) is supplied, the function shows lines that connect the mode to the vertices and to each side of the polygon. When multiple proportion sets are supplied, the modes are connected together to form a trace diagram.

Value

Invsibly returns the modal point(s).

Additionally, a number of calculated values within the function are written to the list object PBStool located in the .PBStoolEnv environment. Use ttget(PBStool) to transfer the object to your local working environment, or use ttprint(PBStool) to print the contents on the command console.

Author(s)

Rowan Haigh, Pacific Biological Sattion, Fisheries and Oceans Canada, Nanaimo BC

References

Aitchison, J. (1986, reprinted in 2003). The Statistical Analysis of Compositional Data. The Blackburn Press, Caldwell, NJ. 416 p.

Schnute, J.T., and Haigh, R. (2007). Compositional analysis of catch curve data, with an application to *Sebastes maliger*. *ICES Journal of Marine Science* **64**: 218–233.

See Also

plotTernary, ttget

Examples

```
local(envir=.PBStoolEnv,expr={
  pbsfun=function(){
    plotTertiary(c(1,1,2,2,5,5))
    invisible() }
  pbsfun()
})
```

---

predictRER *Predict Classification of Rougheye Rockfish*

Description

Discriminant function for the accurate classification of Rougheye Rockfish (RER, *Sebastes aleutianus*) and Blackspotted Rockfish (BSR, *Sebastes melanostictus*).

Usage

```
predictRER(S, L, N)
```

Arguments

S	numeric scalar: Standard length (mm) of RER or BSR
L	numeric vector: Six morphometrics (lengths in mm)
N	numeric vector: Two meristics (observed numbers)

Details

Exploring 35 morphometric and 9 meristic characters, Orr and Hawkins (2008) provide a discriminant function (using only 6 morphometrics L and 2 meristics N) that claims to correctly classify the two species 97.8% of the time.

The discriminant function:

$$D = 101.557\lambda_1 + 52.453\lambda_2 + 0.294N_1 + 51.920\lambda_3 + 0.564N_2 - 38.604\lambda_4 - 22.601\lambda_5 - 10.203\lambda_6 - 10.445$$

where, $\lambda_n = 100L_n/S$, i.e., percent Standard Length

$$\begin{aligned} S &= \text{standard fish length measured from tip of snout,} \\ L_1 &= \text{length of dorsal-fin spine 1,} \\ L_2 &= \text{snout length,} \\ L_3 &= \text{length of gill rakers,} \end{aligned}$$

L_4 = length of pelvic-fin rays,
L_5 = length of soft-dorsal-fin base,
L_6 = preanal length,
N_1 = number of gill rakers,
N_2 = number of dorsal-fin rays.

When $D < 0$, the prediction is that the rockfish is *S. aleutianus*.
When $D > 0$, the prediction is that the rockfish is *S. melanostictus*.

Value

Numeric scalar representing the calculated discriminant value.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries & Oceans Canada, Nanaimo BC.

References

Orr, J.W. and Hawkins, S. (2008) Species of the rougheye rockfish complex: resurrection of *Sebastes melanostictus* (Matsubara, 1934) and a redescription of *Sebastes aleutianus* (Jordan and Evermann, 1898) (Teleostei: Scorpaeniformes). *Fisheries Bulletin* **106**: 111–134.

See Also

simRER, simBSR

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun = function(){
  test = simRER(1000)
  Dval = apply(test,1,function(x){predictRER(x[1],x[2:7],x[8:9])})
  print(paste("RER correctly predicted ",
    round((length(Dval[Dval<0])/length(Dval)*100),1),"% of the time",sep=""))
  test = simBSR(1000)
  Dval = apply(test,1,function(x){predictRER(x[1],x[2:7],x[8:9])})
  print(paste("BSR correctly predicted ",
    round((length(Dval[Dval>0])/length(Dval)*100),1),"% of the time",sep=""))
  invisible() }
pbsfun()
})
```

---

preferDepth — *Display Depth Distribution for a Single Species*

Description

Display depth distribution for a selected species using an interactive GUI. The SQL code grabs the depth table from a remote SQL Server database. The user can specify years and various areas, if desired.

Usage

```
preferDepth(strSpp="410", fqtName="pht_fdep.sql", dbName="PacHarvest",
     spath=NULL, type="SQL", hnam=NULL, get.effort=TRUE)
```

Arguments

strSpp	string specifying the code for a species that appears in the target database
fqtName	string specifying the name of a local file (f), an SQL query file (q), or an MDB table (t)
dbName	string specifying the name of an SQL DSN for an ODBC call to the SQL Server database SVBCPBSGFIIS, or an MDB database
spath	string specifying the path to the file fqtName
type	string specifying the file type: "SQL", "MDB", or "FILE"
hnam	string specifying the name of a history file
get.effort	logical: if TRUE get the trawl effort data right away.

Details

The function preferDepth() creates an interactive GUI that can be used to display histograms of depth distribution for specified species that appear in SQL Server database SVBCPBSGFIIS. The use may also specify a local file (see getFile). The cumulative catch is superimposed to show the depths at which the species is caught in relation to the tow frequency histogram. The total fleet effort is displayed as a shaded histogram in the background.

The GUI controls:

Window	Button to display the window description file.
SQL	Button to display the SQL code (hard-wired to that at start-up).
R code	Button to display the R code used to generate preferDepth.
path	Path to the data/code file.
sql	Button to set path to the sql directory in PBStools.
cwd	Button to set path to the current working directory.
File/Query/Table	Name of (i) data file, (ii) SQL code file or MDB query, or (iii) MDB table.
DB/DSN	Descriptor Name Service that provides an ODBC connection to the SQL Server database SVBCPBSGFIIS.
type	Type of file: SQL, MDB, or FILE.
Trusted	Check if using a trusted DFO identity.
SQL User ID	SQL user ID assigned by the database server administrators.
password	Password assigned by the database server administrators.

Choose Area

All	Use depth record from all areas.
Major	Use depth records with a PMFC major area code.
Minor	Use depth records with a PMFC minor area code.
Locality	Use depth records with a locality (fishing ground) code.
DFO Area	Use depth records with a DFO management area code.
DFO Subarea	Use depth records with a DFO management subarea code.
SRF Area	Use depth records with a slope rockfish assessment area code.
Areas:	Specify area codes within the area type chosen above, either singly (e.g. 3C), as a range (e.g. 5:8, numeric only) or as a group (e.g. 5AB,5CD,5E).
Spp	Species code that appears in the target SQL Server database.
Years	Specify years either singly (e.g. 2000), as a range (e.g. 2000:2003), as a group (e.g. 1996,2000,2006), or as a sequence (e.g. seq(1997,2007,2)).
X/Y Lims	Control the X- and Y-limits of the plot.
Quants	Specify the bounding quantiles of the depth distribution. that will characterise the species depth range.
Bar: colour	Fill colours for histograms; note: specify '0' (transparent) for species depth.
width	Specify the width in X-units of the histogram bars.
eff	If checked, shaded histogram of trawl effort is displayed in background.

catch	If checked, display cumulative catch curve for selected species.
legend	If checked, display the legend information.
wmf	If checked, send the PLOT to a WMF file.
col	Colour of cumulative catch curve.
lwd	Line thickness (line width device) of cumulative catch curve.
EFF	Button to explicitly refresh the trawl effort dataset.
DATA	Button to explicitly refresh the depth dataset given the GUI settings.
PLOT	Button to plot the depth distribution given the GUI settings.

History widget

Value

No value is explicitly return; however, objects of interest are stored in the global list object PBStool.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

See Also

getData, calcHabitat, histMetric
Available SQL queries: SQLcode

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(dfo=FALSE,strSpp="LST"){
  if (dfo) {
    preferDepth(strSpp=strSpp, get.effort=FALSE) }
  else {
    data(testdatC,envir=.PBStoolEnv)
    x=testdatC[testdatC[strSpp]>0 & !is.na(testdatC[strSpp]),]
    testdep=data.frame(year=as.numeric(substring(x$date,1,4)),depth=x$fdep,catch=x$LST)
    attr(testdep,"spp")=strSpp
    save("testdep",file=paste(tempdir(),"/testdep.rda",sep=""))
    preferDepth(strSpp=strSpp, fqtName="testdep", type="FILE",
      spath=convSlashes(tempdir(), os="unix"), get.effort=FALSE) }
  invisible() }
pbsfun()
})
```

---

prepClara *Prepare a Data Object for Clustering Large Applications*

Description

Prepare a data object for use by clarify, which uses the **cluster** function clara.

Usage

```
prepClara(ssid=7, gfld=c("X","Y"), sfld="spp", mfld="catKg",
   fnam=NULL, ioenv=.GlobalEnv)
```

Arguments

ssid	survey series ID
gfld	spatial grouping field name; the default is a special case where Lon-Lat is combined as a unique identifier.
sfld	species field name
mfld	metric field name (determines the impact of the species)
fnam	if NULL, the function queries the GFBioSQL database using the SQL query gfb_clara_survey.sql (at present only queries invertebrates)
ioenv	input/output environment for function input data and output results.

Details

The grouping variable gfld will determine how event data are collapsed into spatial units. The Lon-Lat default is likely equivalent to a sinlge field identifying each fishing event unless events happen to occupy the same Lat-Lon coordinate. A user may wish to create a field that delimits grid cells, but these should be the same as or smaller than those specified in the function clarify.

The function assumes that the input data object is always called 'dat', but a source file can be called anything as long as its import to R creates dat. Similarly, the output data object is always called 'claradat' but can be saved to a system file with any name. The function automatically names the output file containing claradat depending on whether an input name fnam was specified or SQl code was executed to get the input data.

Value

Returns the data frame claradat for use in clarify.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

clarify, crossTab, getData, getFile

---

prime *Report Prime Numbers from an Integer Vector*

Description

Report the prime numbers given an integer vector. If none are found, the function returns NULL.

Usage

```
prime(x = 2:100)
```

Arguments

x	an integer vector that may or may not contain prime numbers.

Details

The function will reject non-vector objects, non-numeric vectors, and numeric vectors that do not comprise integers.

Value

A vector of prime numbers (including duplicates) that appear in the input vector. If no prime numbers are present, the function returns NULL.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

http://en.wikipedia.org/wiki/Prime_number

See Also

biteData, isThere, revStr

---

processBio *Process Results from SQL Query to GFBio*

Description

Process further the global data object PBSdat that results from the SQL query gfb_bio.sql.

Usage

```
processBio(dat=PBSdat, addsrfa=TRUE, addsrfs=TRUE, addpopa=TRUE,
    maxrows=5e4)
```

Arguments

dat	data frame, usually PBSdat, resulting from an SQL query: getData("gfb_bio.sql","GFBioSQL","396")
addsrfa	logical: if TRUE, calculate slope rockfish assessment areas from fields major, minor, and locality, and add as a field srfa to dat.
addsrfs	logical: if TRUE, calculate slope rockfish assessment subareas (gullies) from fields major, minor, and locality, and add as a field srfs to dat.
addpopa	logical: if TRUE, determine Pacific Ocean Perch areas using the PolySet popa and the fields X and Y in dat, then add as a field popa to dat.
maxrows	numeric: number of rows of dat to process at any one time when addpopa=TRUE. Finding POP areas uses the function findPolys, which bogs down when the number of events gets very high.

Details

Currently, this function can add fields srfa (slope rockfish assessment areas), srfs (slope rockfish assessment subareas or gullies), and popa (Pacific Ocean Perch areas identified by Schnute *et al.* 2001). Not all these areas exist as fields in GFBIO.

Value

The data frame dat plus additional fields.

References

Schnute, J.T., Haigh, R., Krishka, B. and Starr, P. (2001) Pacific Ocean Perch assessment for the west coast of Canda in 2001. *Canadian Science Advisory Secretariat, Research Document* **2001/138**, 90 pp.

See Also

`getData`, `getCatch`

`reportCatchAge` *GUI to Display Results from Catch-Age Analysis*

Description

Display various results plots for catch-age analysis from ADMB using an interactive GUI.

Usage

```
reportCatchAge(prefix="pop", path=getwd(), hnam=NULL, ...)
```

Arguments

`prefix`	string specifying the name of an ADMB project.
`path`	string specifying the path to the ADMB files (`*.tpl, *.dat, *.pin, *.rep`); default is current working directory.
`hnam`	string specifying the name of a history file.
`...`	additional arguments for `reportCatchAge()`.

Details

The function `reportCatchAge()` creates an interactive GUI that can be used to display the ADMB results from catch-age analysis.

Original graphs designed by Jon Schnute in Schnute *et al.* (2001).

GUI controls:

`Prefix`	Prefix for an ADMB project.
`WDF`	Button to display the window description file.
`R code`	Button to display the R code for this function.
`TPL`	Button to display the ADMB input files (`*.tpl`, `*.dat`, `*.pin`).
`REP`	Button to display the ADMB output report file.
`GET`	Button to get the ADMB report `*.rep` file.
`path`	Path to the ADMB files: `*.tpl, *.dat, *pin, *.rep`.
`admb`	Button to set path to the library `admb` directory.
`cwd`	Button to set path to the current working directory.
`time steps`	Number of time steps for the simulation.
`catch steps`	Number of catch steps for the simulation.
`catch range`	Limits of the simulated catch.
`recruit productivity`	Recruitment productivity for the simulation.
`r range`	Recruitment productivity range and number of steps.
`recruit offset`	Number of years from recruitment age class to display.
`HS trace`	Check button for tracing the harvest strategy trajectory.
`HS regime`	Check button for showing regime changes in harvest strategy.

History widget	
SMW	Button to display selectivity, maturity, and weight.
Age	Button to display age bubble plots.
Rec	Button to display recruitment and productivity.
Sim	Button to display forward simulation.
Risk	Button to display sustainable catch and productivity.
BPA	Button to display biomass and precautionary approach (harvest strategy) diagram.
PLOT	Button to display plots checked off below.
Menu	Button to display a menu of codes for plots.
AA	Ages actual
AP	Ages predicted
BE	Biomass estimates
CP	Catch (sustainable) vs. productivity
HS	Harvest strategy (precautionary approach)
PP	Probability of achieving productivity
RN	Recruitment numbers
RP	Recruitment productivity
SB	Simulation: biomass at fixed catches
SG	Simulation: growth rate vs. catch
SM	Selectivity and maturity vs. age
WA	Weight vs. age
plot name	File name for figure sent to PDF, PNG, or WMF format.

References

Schnute, J.T., Haigh, R., Krishka, B.A., and Starr, P. (2001) Pacific ocean perch assessment for the west coast of Canada in 2001. *Canadian Science Advisory Secretariat, Research Document* **2001/138**. 90 pp. http://www.dfo-mpo.gc.ca/csas/Csas/publications/ResDocs-DocRech/2001/2001_138_e.htm

See Also

plotProp, pop.age
PBSadmb: admb

---

requestAges *Request Otoliths for Ageing*

Description

Query the DFO database GFBioSQL for available otoliths to age. Choose random otoliths based on commercial or survey catch weights of the specified species.

Usage

```
requestAges(strSpp, nage=500, year=2011,
     areas=list(major=3:9, minor=NULL), ttype=c(1,4),
     sex=1:2, nfld = "nallo", sql=TRUE, bySID=FALSE,
     spath=.getSpath(), uid=Sys.info()["user"], pwd=uid, ...)
```

Arguments

strSpp	string HART code of a species with sampled otoliths.
nage	number of otoliths allocated for this species given the years and areas specified.
year	numeric vector of years.
areas	list of area vectors: those specified first in the list take precedence over those specified later, if overlap occurs.
ttype	numeric vector specifying trip type: 1=non-observed commercial, 2=research, 3=charter, 4=observed commercial. Note that you can either choose purely commercial samples (`ttype=c(1,4)`) OR survey samples (`ttype=c(2,3)`), not a mixture.
sex	numeric sex code where 1=males, 2=females, c(1,2)=males + females.
nfld	string specifying field with number of otoliths to sample: `"nfree"` . . . number of otoliths available (not aged); `"ncalc"` . . . number of otoliths calculated after weighting by species catch, ... less the number of otoliths already aged by break & burn; `"nallo"` . . . default: essentially `ncalc` scaled to the user-specified `nage`. Both `ncalc` and `nallo` are constrained by `nfree`.
sql	logical: if `TRUE` query the databases for all otolith and catch information, otherwise grab the information from stored binaries (e.g., `Sdat396.rda` for POP otolith data, and `Ccat396.rda` for POP catch from `PacHarvest` and `GFFOS`.
bySID	logical: if `TRUE` group the chosen otoliths by sample ID (SID) for presentation in the CSV file (assumes that one tray houses one sample); otherwise assume that many small samples occur in one tray.
spath	string specifying system path to the SQL queries; default points to the SQL directory in **PBStools**.
uid, pwd	user ID and password for Oracle DB account authentication (no longer need as queries are coded in SQL with embedded Oracle queries that use the SQL server's credentials).
...	allows users to request specific trip IDs (either `TID_gfb` or `TID_fos`); also user can specify a minimum number of free otoliths per sample (e.g., `nfree=32`) when selecting.

Value

Invisibly returns the otolith data with supplementary quarterly commercial / survey catch. Also, writes this otolith data to binary `.rda` and ascii `.csv` files.

An ASCII file called `oto123-C(YYYY)-areas(0+0+0)--sex(1+2)-N999.csv` (where `123` = species code, `C` = Commercial or `S` = Survey, `YYYY` = year, `0` = PMFC area code, and `999` = no. of otoliths) writes out the random samples for each trip (if commercial samples) / fishing event (if survey samples) in a format that simulates an otolith tray.

The global list object `PBStool` contains:

module	string	**PBStools** source code module that contains `requestAges`;
call	call	details of the last last call to `requestAges`;
Sdat	data.frame	all sample data for `strSpp`;
catch	data.frame	catches for `strSpp` (either commercial or survey);
C	vector	`strSpp` catch per quarter (commercial) or stratum (surveys);
samp	data.frame	sample data relevant to the user's specification;
describe	string	used to label output files;
Opool	list	pool of available otoliths for samples in `samp`;
Nsamp	vector	number of ages to sample from `Opool`;
Osamp	list	Otoliths chosen randomly from `Opool`.

The otiliths in Opool are identified by SPECIMEN_SERIAL_NUMBER and are taken from the GFBioSQL table B05_SPECIMEN:

```
SELECT
  B5.SAMPLE_ID AS SID,
  B5.SPECIMEN_SERIAL_NUMBER AS SN
FROM B05_SPECIMEN B5
WHERE
  B5.SAMPLE_ID IN (@SIDS) AND
  B5.AGEING_METHOD_CODE IS NULL AND
  B5.SPECIMEN_SEX_CODE IN (@sex)
```

Author(s)

Rowan Haigh & Brian Krishka
Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

calcVB, estOgive, plotProp, weightBio
Available SQL queries: SQLcode

---

revStr *Reverse a String*

Description

Reverse a string character by character.

Usage

```
revStr(x)
```

Arguments

x	vector of strings

Value

A vector strings that have been reversed.

Author(s)

base::strsplit

See Also

wrapText

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  cat(revStr("Nardwuar the Human Serviette"),"\n")
  invisible() }
pbsfun()
})
```

`runCCA` *Run Catch-Curve Analysis on Age Frequency Data*

Description

Perform a catch-curve analysis using a model formulated by Schnute and Haigh (2006). The code allows a user to perform both frequentist (NLM) and Bayesian (BRugs) analyses.

Usage

```
runCCA(fnam="nage394", hnam=NULL, ioenv=.GlobalEnv, ...)
```

Arguments

`fnam`	string specifying file name
`hnam`	string name of a history file
`ioenv`	input/output environment for function input data and output results.
`...`	additional arguments (not currently used).

Details

The function opens up a notebook GUI with two tabs named "NLM" and "BRugs". The user needs to find a decent modal fit using `NLM` before attempting to run the Bayesian analysis, which employs Markov chain Monte Carlo (MCMC) techniques using the Gibbs sample algorithm.

Currently, the NLM fit can use one of three sampling distributions: multinomial, Dirichlet, and logistic-normal. The BRugs model is only coded for four cases using the Dirichlet distribution:

`M.....`case 1: survival only, which depends only on mortality,
`MS....`case 2: survival and selectivity only,
`MA....`case 3: survival and recruitment anomalies only,
`MSA...`case 4: survival, selectivity, and recruitment anomalies.

The Bayesian model is coded in the OpenBUGS language and can be found in the examples subdirectory `library/PBStools/examples`.

Value

Output is primarily visual as directed by the GUI. A number of objects are dumped to the R environment

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

Schnute, J.T., and Haigh, R. (2007) Compositional analysis of catch curve data, with an application to *Sebastes maliger*. *ICES Journal of Marine Science* **64**: 218–233.

See Also

plotProp, nage394

---

runModules *Run GUI Modules Included in PBStools*

Description

Display an interactive GUI to run the various modules in PBStools.

The modules' window description files can be found in the directory PBStools/win, located in R's directory library.

Usage

```
runModules()
```

Details

Some modules can tap into PBS's SQL Server databases.

Note

This is a work in progress so bear with me.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

PBSmodelling: runDemos

---

sampBG *Sample from the Binomial-Gamma Distribution*

Description

Take n random samples from the binomial-gamma distribution $BG(p, \mu, \rho)$.

Usage

```
sampBG(n, p=0.25, mu=1, rho=0.5)
```

Arguments

n	number of random samples to draw.
p	proportion of zero values.
mu	mean of the non-zero values.
rho	coefficient of variation (CV) of the non-zero values.

Details

This function generates n random samples from the binomial $B(p)$ and the gamma $G(\mu, \rho)$ distributions, and returns the product of the two.

Value

A vector of length n of values drawn from the binomial-gamma distribution.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

calcPMR, getPMR

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(P=list(n=1000,p=0.25,mu=1,rho=0.5)) {
  unpackList(P)
  x = sampBG(n,p,mu,rho) # create a sample population
  y = calcPMR(x)
  cat("True parameters for sampling the binomial-gamma distribution:\n")
  print(unlist(P))
  cat("Estimated parameters of the sampled binomial-gamma distribution:\n")
  print(y)
  invisible() }
pbsfun()
})
```

---

scaleVec *Scale Vector to Lie Between Target Minimum and Maximum*

Description

Scale a vector to span a target minimum and maximum.
(See **PBSmodelling** functions scalePar and restorePar.)

Usage

```
scaleVec(V, Tmin = 0, Tmax = 1)
```

Arguments

V	numeric vector (NAs ignored).
Tmin	numeric target minimum value.
Tmax	numeric target maximum value.

Details

This function combines the utility of **PBSmodelling**'s scalePar and restorePar so that a user can rescale a vector to lie between any minimum and maximum value (including c(0,1)).

The target minimum must be less than the target maximum, otherwise the function stops execution.

Value

The numeric vector V re-scaled to range from Tmin and Tmax.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

See Also

scalePar, restorePar

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function() {
  V = runif(50,100,1000)
  T = scaleVec(V, Tmin=-3, Tmax=5)
  lmf = lm(T~V)
  par0 = par(no.readonly=TRUE); on.exit(par(par0))
  expandGraph(cex=1.2,mgp=c(1.8,0.5,0))
  plot(V,T,type="n",cex.lab=1.5)
  abline(lmf,col="dodgerblue",lwd=2)
  points(V,T,pch=21,cex=1.2,col="navyblue",bg="yellow")
  expr=paste(c("addLabel(0.1,0.85,expression(T == ",round(lmf$coeff[1],5),
    " + ",round(lmf$coeff[2],5),"* V~~~~R^2 == ",
    round(summary(lmf)$adj.r.squared,2),"),adj=0)"),collapse="")
  eval(parse(text=expr))
  invisible(expr) }
pbsfun()
})
```

ServerParlance *Info: Server Parlance – SQL Server vs. Oracle*

Description

PBStools allows a user to easily query SQL Server and Oracle databases. At PBS, both types reside on a single server SVBCPBSGFIIS. The two database systems are similar but the terminology is potentially confusing. This package attempts to equalize the two database systems for the end user by representing the Oracle hierarchy in terms of the SQL Server system. Specifically, we use the term server to refer to an Oracle database, and the term database to refer to an Oracle schema.

Details

In SQL Server parlance we have:

1. A server (SVBCPBSGFIIS).
2. One or more SQL Server instances.
   The default instance is unnamed but if you run multiple instances they have to be named.
3. Each instance has one or more databases (e.g. PacHarvest).
4. Each database contains one or more schemas (the default schema is dbo).
5. Each schema consists of various logical objects (e.g. tables, views, procedures).

In Oracle parlance we have:

1. A server (SVBCPBSGFIIS).
2. One or more databases (e.g. GFSH).
3. Each database contains one or more schemas (e.g. GFFOS).
4. Each schema consists of various logical objects (e.g. tables, views, procedures).

In comparison, an SQL Server instance is equivalent to an Oracle database, and an SQL Server database is equivalent to an Oracle schema.

SQL Server schemas are quite a bit different than Oracle schemas in that they are chiefly a way of organizing objects and controlling permissions within a single database, whereas Oracle schemas are basically separate logical structures in the same way that SQL Server databases are.

Author(s)

Norm Olsen, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

PBStools for package functions and data.
SQLcode for available SQL queries.
getData to execute the SQL queries.

---

showAlpha *Show Quantile Confidence Levels (alpha) for Bootstraps.*

Description

Show quantile confidence levels α for bootstrapped biomass.

Usage

```
showAlpha(lims=c("emp","bca"))
```

Arguments

lims	bootstrap confidence limit types from choices in bootBG.

Details

This function assumes that `bootBG()` has been run and that the data list object `PBStool` exists in the global position. The figure generated follows the format of Figure 3 in Schnute and Haigh (2003). The panels show:
(a) histogram of bootstrapped biomass;
(b) cumulative probability distribution of bootstrapped biomass;
(c) normal quantiles of the estimates $\hat{B}$;
(d) observed probabilities that the true value B lies in the j^{th} confidence interval;
(e) $\hat{\alpha}_j$ *vs.* α_j for chosen confidence limit types.

Value

No value is explicitly returned by the function. Objects generated are appended to the global list object `PBStool`:

`plev`	vector: probability levels used in the `bootBG` analysis.
`xy`	histogram: output from the last α histogram.
`xcount`	vector: count of α values in Confidence Interval (CI) for the last graph.
`pobs`	vector: probability of α values in CI bins for the last graph.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

See Also

`bootBG`, `getPMR`, `calcPMR`, `sampBG`

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(SID=1,S=50){
  bootBG(dat=pop.pmr.qcss,SID=SID,S=S)
  showAlpha()
  invisible() }
pbsfun()
})
```

`showIndices` *Show Survey Indices from Bootstrap Tables*

Description

Show survey indices for a species from bootstrapped biomass estimates stored in the SQL database `GFBioSQL`.

Usage

```
showIndices(strSpp="396", survID=c(1,2,3,121,167), bootID)
```

Arguments

strSpp	string specifying species code (species$code).
survID	numeric survey ID assigned to each survey by the GFBio team.
bootID	numeric boot run ID assigned to each record in the BOOT header table.

Details

The function relies on a short query function called getBootRuns which in turn queries tables of survey information (BOOT_HEADER) and relative biomass estimates by species for each survey (BOOT_DETAIL).

Species relative biomass indices and their bootstrapped confidence intervals are plotted.

Note: Any survey ID survID can have multiple boot runs bootID (e.g., re-stratification trials).

Value

Annual biomass indices for the specified survIDs are return invisibly. A global list object PBStool provides the following:

surveys	Data frame of survey information for the specified species.
sppBoot	Subset of surveys that has records with the specified survey IDs.
survBoot	Subset of surveys that shows boot run description for bootID.
xy	List of x- and y-vectors for the biomass points on the plot.
cil	List of x- and y-vectors for the lower points of the confidence limits.
cih	List of x- and y-vectors for the upper points of the confidence limits.
clr	List of x- and y-vectors for the colour numbers of biomass points.
llab	List of boot run labels for each point.
ulab	Vector of unique boot run labels for the legend.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

getBootRuns, calcMoments, spn, species

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(strSpp="602",region="QCS",dfo=FALSE){
  if (dfo) {
    choices=c("WCVI","QCS","WQCI","HS")
    i=match(region,choices,nomatch=0)
    if (i==0) showError(paste("Choose region from\n'",
      paste(choices,collapse="', '"),"'",sep=""),as.is=TRUE)
    survID=switch(i,
      c(16,58,70,124,125,126,128),
      c(1,2,3,59,121,127),
      c(57,122,123,129),
      c(15,111,162,163) )
    showIndices(strSpp,survID) }
  else showMessage("If connected to DFO network, set argument 'dfo=TRUE'")
  invisible() }
pbsfun()
})
```

showMessage *Display a Message on the Current Device*

Description

Display a message (possibly error) on the current device. If `err==TRUE`, stop code execution.

Usage

```
showMessage(str, type="", as.is=FALSE, err=FALSE, ...)
```

Arguments

`str`	string indicating the message or error.
`type`	code indicating header string before the message.
`as.is`	logical: if `TRUE`, turn off the automatic line breaking.
`err`	logical: if `TRUE`, treat the message as an error message and stop code execution.
`...`	other graphical parameters to pass to `addLabel`.

Details

The user-specified message will be displayed on the current device. If `err==TRUE`, code execution will cease using `stop`.

A wrapper function called `showError` is included for backwards compatibility.

Value

No value is explicitly returned, error message is displayed.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`getData`, `wrapText`

simBGtrend *Simulate Population Projection Based on Prior Binomial-Gamma Parameters.*

Description

Simulate a population projection based on prior binomial-gamma parameters (p, μ, ρ), and display the results.

Usage

```
simBGtrend(pmr=pop.pmr.qcss, Npred=15, genT=NULL, Nsim=1,
   yrs=NULL, alpha=1, ioenv=.GlobalEnv)
```

Arguments

pmr	data frame with fields:
	SID = survey ID number assigned by the GFBio team;
	h = strata ID (alpha and/or numeric);
	p = proportion of zero measurements;
	mu = mean of nonzero measurements;
	rho = CV of nonzero measurements;
	A = bottom area (sq.km) of each stratum;
	n = actual number of tows performed to derive (p,mu,rho);
	k = preference/weighting of new tows.

Npred	number of predicted years past the observed year.
genT	generation time; if specified, Npred is automatically set to 3 times genT.
Nsim	number of population simulations.
yrs	years of the observed parameters (normally, getPMR() attaches this information to the pmr data frame as an attribute called survey).
alpha	coefficient to weight past observations of population parameters: alpha=0 applies equal weighting, increasingly positive alpha values apply exponentially heavier weights to latter observations, and negative alpha values weight earlier observations more heavily.
ioenv	input/output environment for function input data and output results.

Details

This function attempts to simulate a population past the observed year by randomly sampling the binomial-gamma distribution $BG(p, \mu, \rho)$ using weighted prior values of the population parameters (p, μ, ρ).

Value

No value is explicitly returned by the function. Objects generated are appended to the global list object PBStool (which contains the last call to bootBG):

pmrhist	array: the p, mu, rho values used for each year of each simulation.
Btrend	array: estimated annual biomass and CI values for each simulation.
Bmean	data frame: mean annual biomass and CI limits of the multiple simulations.
Bseen	data frame: mean annual biomass and CI limits of the observed surveys.
brR	data frame: for each simulation (one record): slope b of the linear trend through annual indices; annualized rate of change r implied by the slope b; accumulated rate of change R implied by the slope b.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Schnute, J.T. and R. Haigh. (2003) A simulation model for designing groundfish trawl surveys. *Canadian Journal of Fisheries and Aquatic Sciences* **60**, 640–656.

Schnute, J., Haigh, R., Krishka, B., Sinclair, A. and Starr, P. (2004) The British Columbia longspine thornyhead fishery: analysis of survey and commercial data (1996–2003). *Canadian Science Advisory Secretariat, Research Document* **2004/059**, 75 pp.

See Also

getPMR, calcPMR, sampBG

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(Nsim=1,alpha=1) {
  simBGtrend(Nsim=Nsim,alpha=alpha)
  invisible() }
pbsfun()
})
```

---

simBSR *Simulate Blackspotted Rockfish Biological Data*

Description

Simulate biological data that best characterizes Blackspotted Rockfish (BSR, *Sebastes melanostictus*), using parameter estimates from Table 2 in Orr and Hawkins (2008).

Usage

```
simBSR(Nfish)
```

Arguments

Nfish	Number of BSR fish to simulate.

Details

Exploring 35 morphometric and 9 meristic characters, Orr and Hawkins (2008) provide a discriminant function (using only 6 morphometrics L and 2 meristics N) that claims to correctly classify *Sebastes melanostictus* and *S. aleutianus* 97.8% of the time (see predictRER).

Table 2 in Orr and Hawkins (2008) provides a range of BSR standard lengths (mm) and distributions for morphometrics and meristics. This function samples from a random uniform distribution for S and from random normal distributions for model inputs of λ and N – table below gives (μ, σ).

S	=	standard fish length measured from tip of snout	95.5–539
λ_1	=	length of dorsal-fin spine 1	(7.8, 0.7)
λ_2	=	snout length	(8.0, 0.6)
λ_3	=	length of gill rakers	(5.6, 0.6)
λ_4	=	length of pelvic-fin rays	(21.4, 1.2)
λ_5	=	length of soft-dorsal-fin base	(21.4, 1.5)
λ_6	=	preanal length	(70.2, 2.6)
N_1	=	number of gill rakers	(33.0, 1.2)
N_2	=	number of dorsal-fin rays	(13.7, 0.5)

where, $\lambda_n = 100L_n/S$, i.e., percent Standard Length

Value

A numeric matrix with dimensions c(Nfish,9) where columns are labelled c('S', 'L1', 'L2', 'L3', 'L4', 'L5', 'L6', 'N
The values are described above in **Details**, but generally, S = standard length of the fish (mm), L = six diagnostic

length measurements (mm), and N = numbers of gill rakers and dorsal fin rays.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries & Oceans Canada, Nanaimo BC.

References

Orr, J.W. and Hawkins, S. (2008) Species of the rougheye rockfish complex: resurrection of *Sebastes melanostictus* (Matsubara, 1934) and a redescription of *Sebastes aleutianus* (Jordan and Evermann, 1898) (Teleostei: Scorpaeniformes). *Fisheries Bulletin* **106**: 111–134.

See Also

`simRER`, `predictRER`

---

simRER *Simulate Rougheye Rockfish Biological Data*

Description

Simulate biological data that best characterizes Rougheye Rockfish (RER, *Sebastes aleutianus*), using parameter estimates from Table 2 in Orr and Hawkins (2008).

Usage

```
simRER(Nfish)
```

Arguments

`Nfish`	Number of RER fish to simulate.

Details

Exploring 35 morphometric and 9 meristic characters, Orr and Hawkins (2008) provide a discriminant function (using only 6 morphometrics L and 2 meristics N) that claims to correctly classify *Sebastes aleutianus* and *S. melanostictus* 97.8% of the time (see `predictRER`).

Table 2 in Orr and Hawkins (2008) provides a range of RER standard lengths (mm) and distributions for morphometrics and meristics. This function samples from a random uniform distribution for S and from random normal distributions for model inputs of λ and N – table below gives (μ, σ).

S	=	standard fish length measured from tip of snout	63.4–555.2
λ_1	=	length of dorsal-fin spine 1	(5.8, 0.6)
λ_2	=	snout length	(7.5, 0.7)
λ_3	=	length of gill rakers	(4.9, 0.6)
λ_4	=	length of pelvic-fin rays	(22.1, 1.1)
λ_5	=	length of soft-dorsal-fin base	(22.8, 1.2)
λ_6	=	preanal length	(71.8, 2.3)
N_1	=	number of gill rakers	(31.2, 1.0)
N_2	=	number of dorsal-fin rays	(13.5, 0.5)

where, $\lambda_n = 100L_n/S$, i.e., percent Standard Length

Value

A numeric matrix with dimensions `c(Nfish,9)` where columns are labelled `c('S', 'L1', 'L2', 'L3', 'L4', 'L5', 'L6', 'N`
The values are described above in **Details**, but generally, S = standard length of the fish (mm), L = six diagnostic length measurements (mm), and N = numbers of gill rakers and dorsal fin rays.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries & Oceans Canada, Nanaimo BC.

References

Orr, J.W. and Hawkins, S. (2008) Species of the rougheye rockfish complex: resurrection of *Sebastes melanostictus* (Matsubara, 1934) and a redescription of *Sebastes aleutianus* (Jordan and Evermann, 1898) (Teleostei: Scorpaeniformes). *Fisheries Bulletin* **106**: 111–134.

See Also

`simBSR`, `predictRER`

---

spooler *Spool List Object Information Into New Fields*

Description

Spool a named list or list of lists into a new field of an existing data frame.

Usage

```
spooler(xlist, newfld="area", target)
```

Arguments

`xlist`	Named list or list of lists.
`newfld`	String name of a new field to be created in the target data frame.
`target`	Target data frame object.

Details

Spool a named list or list of lists into a new field of an existing data frame where the names of the lists match existing fields in the target data frame. The elements of the list are values to match in the existing fields. The values placed in `newfld` are either matched values (if a simple list is supplied) or list names concatenated with matched values (if a list of lists is supplied).

This function is most useful when multiple fields exist that describe a similar concept. For example `area` might be any combination of the fields `major`, `minor`, `locality`, `srfa`, `srfs`, and `popa`. Obviously some of these overlap and the user has to be mindful of which combinations to collapse into one field. If multiple matches are available, the code gives preference to the first search field in the list.

Value

The `target` data frame augmented with the new field `newfld`.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

biteData, compCsum

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(pop.age,envir=.PBStoolEnv)
  temp=pop.age[sample(1:nrow(pop.age),10),]
  spooler(list(major=list(3:4,7:8),srfs=list("GS","MI","MR")),"area",temp)
  print(temp)
  invisible() }
pbsfun()
})
```

SQLcode *Info: SQL Code in PBStools*

Description

PBStools contains numerous SQL code files to query DFO databases. The SQL code is executed by the function `getData`, which relies on the R package **RODBC**. The code files are found in the package library `..\library\PBStools\sql`.

Note:
The location of the SQL code files on a user's system can be determined by specifying `.getSpath()`, which returns a path string invisibly.

Details

The Fisheries and Oceans Canada (DFO) databases used by groundfish personnel comprise the following:

Database	code	Fishery	Dates
GFBioSQL	gfb	JV Hake, research, surveys, etc.	1946–2011
GFCatch	gfc	Commercial fisheries data	1954–1975
GFFOS	fos	Commercial fisheries (all sectors)	2002–2011
PacHarv3	ph3	Commercial sales slips data	1982–1994
PacHarvest	pht	Commercial trawl fisheries	1996–2007
PacHarvHL	phhl	Commercial hook & line fisheries	1984–2006
PacHarvSable	phs	Commercial sablefish fishery	1987–2006

Further details of the DFO databases can be found in Haigh and Yamanaka (2011).

GFBioSQL

The GFBioSQL database is an SQL copy of the Oracle database GFBIO. The SQL version is housed on the SQL server SVBCPBSGFIIS located at the Pacific Biological Station (PBS). The database contains biological sample and specimen information, as well as catch from fishing activity not housed in any other database (e.g., JV Hake, research, surveys, foreign fleets).

SQL queries:

gfb_age_request.sql	Summarize and choose otoliths taken but not aged.
gfb_bio.sql	Specimen biological data for a target species.
gfb_boot.sql	Bootstrapped survey indices from boot tables.
gfb_bycatch.sql	Annual fish group catches as bycatch to target species' depth range.
gfb_catch_records.sql	Research survey catch for getCatch and weightBio.
gfb_concat_rows_example.sql	Concatenate values when number of items is small and known upfront. http://www.projectdmx.com/tsql/rowconcatenate.aspx
gfb_fishwt.sql	Mean weight of species sampled from commercial and research trips.
gfb_fos_age_request.sql	Otoliths taken but not aged; show only those that can be identified by GFFOS' TRIP_ID.
gfb_fos_catch.sql	Get commercial catch from GFFOS; match trip IDs in GFBioSQL & GFFOS.
gfb_fos_tid.sql	Match GFBioSQL's TRIP_IDs with those in GFFOS.
gfb_gfb_catch.sql	Research survey catch for requestAges.
gfb_iphc.sql	IPHC survey data for target species.
gfb_iphc_deprecated.sql	**Deprecated** (use gfb_iphc.sql); query for IPHC survey data.
gfb_pht_catch.sql	Get commercial catch from PacHarvest; match trip IDs in GFBioSQL & GFFOS.
gfb_pmr.sql	Population parameters (p, μ, ρ) for target species in survey strata.
gfb_survey_activity.sql	Summarise the annual survey catch for specified species.
gfb_survey_catch.sql	Survey catch and density by species.
gfb_survey_stratum.sql	Survey stratum information.
gfb_tasu.sql	Specimen biological data for LL surveys in Tasu, Flamingo, Triangle, Brooks.
gfb_tid_catch.sql	**Deprecated** (use gfb_pht_catch.sql). Commercial catch from PacHarvest; match trip IDs in GFBioSQL & FOS.
gfb_YrWtAge.sql	Year, weight, age data for target species.

GFCatch

GFCatch was originally a FORTRAN data compilation but is now housed on the SQL server SVBCPBSGFIIS. This database contains catch and effort data from three sources: logbooks (skippers, onboard observers), landing records (sales slips or validation records), and anecdotal information (Rutherford 1999).

SQL queries:

gfc_catch_fyear.sql	Catch of target species by fishing year.
gfc_catch_records.sql	Species catch by tow (zero-catch records excluded).
gfc_fcatORF.sql	GFCatch landings for rockfish reconstruction.
gfc_glm.sql	Catch/effort data for a target species; used in a GLM analysis.

GFFOS

This database is a subset of the Fishery Operations System (FOS), an Oracle database on the server ORADEV in Vancouver. The subset GFFOS (Groundfish FOS) comprises Views to FOS that make the data more useful to stock assessment personnel, and is housed on the Oracle server GFSH located at PBS, Nanaimo.

SQL queries:

fos_bycatch.sql	Annual fish group catches as bycatch to target species' depth range.
fos_catch.sql	Commercial catch by year, month, and PMFC area.
fos_catch_records.sql	Species catch by tow (zero-catch records excluded).
fos_fcatORF.sql	Species landings format for catch history reconstructions.
fos_feid_catch.sql	Fishing event catch.
fos_fid_catch.sql	Total catch by weight of every Fishing Event ID.
fos_fishwt.sql	Mean weight for every species with catch and count info.
fos_glm.sql	Catch/effort data for a target species; used in a GLM analysis.

`fos_map_density.sql`	Catch/effort data for a target species; used to map spatial density.
`fos_sectors.sql`	Trips rationalised by sector and fishery ID.
`fos_tid_catch.sql`	Summarise `GFFOS` catch (kg) by trip ID, date, PMFC major and PMFC minor for a specified species.
`fos_vcatORF.sql`	Query catch from the merged catch table `GF_D_OFFICIAL_FE_CATCH`.
`fos_vess_year_trip.sql`	Tally number of years and trips/year for each vessel.

PacHarv3

This Oracle database is commonly called `PacHarv3`; however, it's real name is `HARVEST_V2_0` and it sits on the Oracle server `ORAPROD`. It contains sales slip data from landings records by trip. The data tables are numerous and the linkages obscure. Fortunately, there exists a summary table called `CATCH_SUMMARY`.

SQL queries:

`ph3_fcatORF.sql`	Species landings format for catch history reconstructions.
`ph3_orfhistory.sql`	Catch table for other rockfish used in catch reconstructions.
`ph3_target.sql`	Determine target species by year, month, area, and gear.

PacHarvest

The `PacHarvest` database (sometimes referred to as `PacHarvTrawl`) sits on the SQL server `SVBCPBSGFIIS` and houses observer and fisher log catch and effort information for most trawl fishing events from Feb 1996 to Mar 2007.
Details of the database can be found in Schnute et al. (1999).

SQL queries:

`pht_bycatch.sql`	Annual fish group catches as bycatch to target species' depth range.
`pht_catch_fyear.sql`	Catch of target species by fishing year using observer & fisher logs.
`pht_catch_gullies_cyear.sql`	Catch of target species by calendar year in QCS gullies.
`pht_catch_gullies_fyear.sql`	Catch of target species by fishing year in QCS gullies.
`pht_catch_records.sql`	Species catch by tow (zero-catch records excluded).
`pht_clara.sql`	Catch data for use in `clara` clustering algorithm.
`pht_concurrent.sql`	Top 20 species caught concurrently with a target species.
`pht_effort.sql`	Trawl effort used for background histogram in depth preference.
`pht_fdep.sql`	Catch of target species used to determine depth of capture.
`pht_fdep_pjs.sql`	As for `pht_fdep.sql`, but vessels present ≥ 3 years and ≥ 5 trips/y.
`pht_glm.sql`	Catch/effort data for a target species; used in a GLM analysis.
`pht_map.sql`	**Deprecated** (use `pht_map_density.sql`); data to map spatial density.
`pht_map_density.sql`	Catch/effort data for a target species; used to map spatial density.
`pht_tcatORF.sql`	Trawl landings of target species, POP, & ORF (other rockfish).
`pht_vess_year_trip.sql`	Tally number of years and trips/year for each vessel.

PacHarvHL

The `PacHarvHL` database comprises hook and line records from fisher log data (1986–2006 for ZN license, 2001-2006 for Schedule II), observer log data (1999–2004 for ZN and Halibut, 2001–2004 for Schedule II), and the Dockside Monitoring Program (1995–2005 for ZN, 1996–2005 for Schedule II, and 1991-2005 for Halibut). This database sits on the SQL server `SVBCPBSGFIIS`. See Haigh and Richards (1997) for some of the early history.

SQL queries:

`phhl_concurrent.sql`	Top 20 species caught concurrently with a target species.
`phhl_districts.sql`	Calculate proportion of PMFC major areas in each historical district.
`phhl_fcatch_fyear.sql`	Fisherlog catch of target species by fishing year & PMFC area.
`phhl_fcatORF.sql`	Fisherlog catch of target species, POP, & ORF (other rockfish).

phhl_hcatch_fyear.sql	Halibut bycatch of target species by fishing year & PMFC area.
phhl_hcatORF.sql	Halibut bycatch of target species, POP, & ORF (other rockfish).
phhl_map_density.sql	Catch/effort data for a target species; used to map spatial density.
phhl_ocatORF.sql	Offical (landed) catch of target species, POP, & ORF (other rockfish).
phhl_vcatch_fyear.sql	Verified landings of target species by fishing year & PMFC area.
phhl_vcatORF.sql	Verified landings of target species, POP, & ORF (other rockfish).

PacHarvSable

The database PacHarvSable (on the SQL server SVBCPBSGFIIS) includes commercial trap and longline Sablefish catch and effort records from fisher logbooks (1990–2006), observer logbooks (2000–2005), and the Dockside Monitoring Program (1995–2002).

SQL queries:

phs_scatORF.sql	Official landings of target species, POP, & ORF (other rockfish).

Author(s)

Rowan Haigh & Norm Olsen
Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Haigh, R., and Richards, L.J. (1997) A relational database for hook and line rockfish logbook data. *Canadian Manuscript Report of Fisheries and Aquatic Sciences* **2408**: vi + 46 p.

Haigh, R., and Yamanaka, K.L. (2011) Catch history reconstruction for rockfish (*Sebastes* spp.) caught in British Columbia coastal waters. *Canadian Technical Report of Fisheries and Aquatic Sciences* **2943**: viii + 124 p.

Rutherford, K.L. (1999) A brief history of GFCatch (1954-1995), the groundfish catch and effort database at the Pacific Biological Station. *Canadian Technical Report of Fisheries and Aquatic Sciences* **2299**. v + 66 p.

Schnute, J.T., Olsen, N., and Haigh, R. (1999) Slope rockfish assessment for the west coast of Canada in 1998. *Canadian Stock Assessment Secretariat, Research Document* **99/184**, 104 p.

See Also

PBStools for package functions and data.
getData to execute the SQL queries.
ServerParlance for differences between SQL Server and Oracle database organisation.

---

stdConc *Standardise Chemical Concentration*

Description

Standardise a chemical concentration from various aavailable unit combinations to a common concentration and adjust for moisture.

Usage

```
stdConc(dat, nUout="mg", dUout="kg", fac=1)
```

Arguments

dat	vector with 5 elements or a data frame with 1 row and 5 columns (labelled or not): namt. . . . numerator amount nunit. . . numerator unit damt. . . . denominator amount dunit. . . denominator unit moist. . . moisture content as a proportion
nUout	character: numerator unit desired
dUout	character: denominator unit desired
fac	numeric: multiplicative factor (*e.g.*, 75 for 75 mg)

Details

Note that the input object dat mixes data types (numeric and character). These types are maintained in a single-row data frame but a vector will change all numerics to character. The function stdConc automatically forces all inputs back to their intended mode.

Value

A one-row data frame with fields c(stdAmt, unit), which represents the standardised concentration as a numeric value and a string unit fraction.

Note

The available units for combinations in the numerator and denominator are:

mcg	microgram	mg	milligram	g	gram
kg	kilogram	L	litre	pct	percent (%)
pdw	% dry weight	ppm	parts per million	ppt	parts per thousand

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

calcMA, crossTab

---

sumBioTabs *Summarize Frequency of Biological Samples*

Description

Summarize frequency occurrence of biological samples and specimens, and send output to a data table.

Usage

```
sumBioTabs(dat, fnam="sumBioTab.csv", samps=TRUE, specs=TRUE,
     facs=list(c("year","major"), c("year","ttype"), c("year","stype"),
     c("year","ameth")) )
```

Arguments

dat	data object with biological sample and specimen information (e.g., pop.age).
fnam	string name of the output file to which summary tables are sent.
samps	logical, if TRUE, report the number of samples.
specs	logical, if TRUE, report the number of specimens.
facs	list of factors in pairs (two-element string vectors).

Value

No value is returned to the R console. The output is placed into the file name specified.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

sumCatTabs, processBio
base: table
reshape: melt.data.frame, cast

---

sumCatTabs *Summarize Catch by Year and PMFC Area*

Description

Summarize catch by year and PMFC area from modern catch arrays compiled during catch reconstruction.

Usage

```
sumCatTabs(dat, facs=list(c("year","major")),
     cflds=c("landed","discard"), fnam)
```

Arguments

dat	a 4-dimensional array object of catch created by the catch reconstruction algorithm buildCatch; the dimensions comprise: year = calendar year, major = PMFC major area codes (1=4B, 3=3C,4=3D, 5=5A, 6=5B, 7=5C, 8=5D, 9=5E), fid = fishery identifier code (1=trawl, 2=halibut, 3=sablefish, 4=dogfish/lingcod, 5=h&l rockfish, catch = catch of principal species and various groups (landed, discard, POP, ORF, TRF, TAR).
facs	list of factors in pairs (two-element string vectors).
cflds	string names of columns containing catch to sum and report.
fnam	string name of the output file to which summary tables are sent.

Value

No value is returned to the R console. The output is placed into the file name specified.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

buildCatch, processBio, sumBioTabs

---

toUpper *Capitalise First Letters of Words*

Description

Capitalise the first letter of each word in a sentence or phrase.

Usage

```
toUpper(x, exclude=c("&","and","exact","or","in","on","organic","pelagic",
   "sp.","spp.","species","undetermined","unidentified","birds","barnacles",
   "crabs","eggs","fishes","larvae","matter","objects","remains","sample",
   "shells","subfamily","tunicates","worms"))
```

Arguments

x	a vector of sentences or phrases
exclude	words to exclude from capitalisation

Details

The function uses strsplit and toupper along with a whole mess of sapply calls. A vector of sentences or phrases can be passed to x.

Value

The input vector where the first letter of all words are capitalised.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

revStr, scaleVec, plotConcur
PBSdata: species

trackBycat *Track Annual Fish Group Catches*

Description

Track fish group catches by year and PMFC area that occur between depths corresponding to a target species' depth-of-occurrence.

Usage

```
trackBycat(strSpp="396", major=5:7, mindep=70, maxdep=441,
   dbs=c("gfb", "pht", "fos"), trawl="bottom", spath=.getSpath(),
   pyrs=1996:2010, rda=NULL, ioenv=.GlobalEnv)
```

Arguments

strSpp	string Hart code for a target fish species.
major	numeric codes for a set of PMFC major areas.
mindep	numeric specifying the minimum depth that defines strSpp's depth-of-occurrence.
maxdep	numeric specifying the maximum depth that defines strSpp's depth-of-occurrence.
dbs	string vector codes that specify particular DFO databases; currently uses: "gfb" = GFBioSQL, "pht" = PacHarvTrawl, "fos" = GFFOS.
trawl	string specifying trawl type – either "bottom" or "midwater".
spath	path where SQL files are located; defaults to the package repository of SQL code.
pyrs	numeric vector of years used to plot the results.
rda	string name (without extension) of an R binary file (.rda) containing an array of catches called bycatch.
ioenv	input/output environment for function input data and output results.

Details

The function currently uses three SQL query files called gfb_bycatch.sql, pht_bycatch.sql, and fos_bycatch.sql, which tap into the DFO databases GFBioSQL, PacHarvTrawl, and GFFOS, respectively.

The function displays two barplots of annual catch, absolute and relative, where bars are sectioned by fish groups.

Value

If rda=NULL, then three SQL queries extract annual fish group catches from the databases mentioned above. These catches are transferred to a 3-dimensional array bycatch:
year... years spanning all datasets;
fish... fish groups: POP, rockfish, turbot, flatfish, hake, sharks, other;
db..... DFO databases: gfb = GFBioSQL, pht = PacHarvest, fos = GFFOS.
The catch array is saved to an .rda file with a name that indicates major areas and trawl type (bottom or midwater).

If the argument rda is given an .rda name, the bycatch array is loaded, by-passing the SQL calls.

The bycatch array is returned invisibly.
Additionally, a list object called PBStool, located in the user's global environment, is populated with:
module..... the name of the **PBStools** module to which this function belongs;
call..... the call to the function trackBycat;
plotname... name of the plot, should the user which to use it, e.g. .plotDev(act="png");

`bartab.....` matrix of catch (tonnes) by year and fish group, summed over the databases;
`amat.......` matrix of absolute catch (kt) used in the upper barplot;
`rmat.......` matrix of relative catch (0:1) used in the lower barplot;
`clrs.......` vector of colour names to distinguish the fish groups within any one bar.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC.

See Also

`getData`, `getCatch`, `buildCatch`
Available SQL queries: `SQLcode`

`trackComp` *Track Composition of Phytoplankton over Time*

Description

Track the mean composition of phytoplankton groups over time using a specified binning interval.

Usage

```
trackComp(fqtName=c("Ex01_Sample_Info","Ex02_Species_Abundance"),
   dbName="Examples.mdb", type="MDB", path=getwd(), ndays=15, groups=NULL,
   dlim=c("2007-01-01",substring(Sys.time(),1,19)),
   clrs=c("green","dodgerblue","orange","yellow","red","grey"),
   pdf=FALSE, wmf=FALSE, ioenv=.GlobalEnv, ...)
```

Arguments

`fqtName`	string vector specifying table names containing sample information and species abundance, respectively; (can also be two local data objects if `type="FILE"`.)
`dbName`	string specifying the name of a database.
`type`	type of database (e.g., `MDB` = Microsoft ACCESS database query or table).
`path`	Directory path to the database.
`ndays`	integer value specifying a time interval in days to partition a year.
`groups`	vector of groups for summary; the default is: `c("Skeletonema", "Thalassiosira", "Chaetoceros", "Other Diatoms", "Phytoflagellates",`
`dlim`	date limits to restrict time line of compositional changes.
`clrs`	vector of colours to shade `groups`.
`pdf`	logical: if `TRUE`, send figure to a postscript document file `PDF`.
`wmf`	logical: if `TRUE`, send figure to a windows metafile `WMF`.
`ioenv`	input/output environment for function input data and output results.
`...`	additional vectors to use for data selection (e.g., `co="CPA", use=1`).

Details

Creates a stacked line plot of relative composition frequencies `0:100` for each `group`. The polygons are coloured by `clrs`.

Value

No value is explicitly returned; however, objects of interest are store in the global list object PBStool.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

convYP, boxSeason

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(os=.Platform$OS.type) {
  if (os=="windows") trackComp(ndays=15,path=.getSpath())
  else showMessage("Only functional for Microsoft Access on Windows OS")
  invisible() }
pbsfun()
})
```

---

trend *Display Boxplots of CPUE and the Trend Line*

Description

Display boxplots of catch or CPUE grouped by year. A trend line is fit through the annual index points summarized by a user-specified summary function (e.g., mean). The interactive GUI also allows the generation and display of bootstrapped estimates of slope *b* and annual rate *r*.

Usage

```
trend(strSpp="442", fqtName="gfb_iphc.sql", dbName="GFBioSQL",
   spath=NULL, type="SQL", ioenv=.GlobalEnv, hnam=NULL)
```

Arguments

strSpp	string specifying the code for a species that appears in the target database
fqtName	string specifying the name of a local file (f), an SQL query file (q), or an MDB table (t)
dbName	string specifying the name of an SQL DSN for an ODBC call to the SQL Server database SVBCPBSGFIIS, or an MDB database
spath	string specifying the path to the file fqtName
type	string specifying the file type: "SQL", "MDB", or "FILE"
ioenv	input/output environment for function input data and output results.
hnam	string specifying the name of a history file

Details

The function `trend()` creates an interactive GUI that can be used to display boxplots, trend lines, and bootstrapped estimates of change.

GUI Left - Main Control:

`Window`	Button to display the window description file.
`SQL`	Button to display the SQL code.
`R code`	Button to display the R code used to generate `trend`.
`Spp`	Species code in target data object or SQL Server database.
`SQL`	Button to run Sql code to get the species data instead of specifying a data object.
`Obj`	Name of data file/object.
`IPHC`	Example IPHC SSA longline data objects in package: `iphc.rbr`, `iphc.rer`, `iphc.yyr`.
`path`	Path to the SQL code or data object.
`sql`	Button to set path to the library's `sql` directory.
`cwd`	Button to set path to the current working directory.
`File/Query/Table`	Name of (i) data file, (ii) SQL code file or MDB query, or (iii) MDB table.
`DB/DSN`	Database name on the SQL server `SVBCPBSGFIIS`.
`type`	Type of file: `"SQL"`, `"MDB"`, or `"FILE"`.
`GET`	Button to run the code specified on this line of the GUI.
`Trusted`	Check if using a trusted DFO identity.
`SQL User ID`	SQL user ID assigned by the database server administrators.
`password`	Password assigned by the database server administrators.
`History`	History widget for storing GUI settings.
`TREND`	Button to calculate and display the trend based on GUI choices.
`boots`	Number of bootstrap replicates.
`BOOT`	Button to calculate and display bootstrapped parameters.
`wmf`	Check box to send TREND or BOOT to a `.wmf` file.

GUI Right - Settings:

`Years`	Specify years either singly (e.g. `2000`), as a range (e.g. `2000:2003`), as a group (e.g. `1996,2000,2006`), or as a sequence (e.g. `seq(1997,2007,2)`)
`Func`	Summary function to represent annual index points in trend line estimation.
`Exclude stations`	Exclude stations that never catch the target species.
`Include zeroes`	Include zero-values in the calculation of the index (need to add a small value if using log transformations).
`Add value`	Add a constant value to the data points.
`Transforms`	Transformation of data (`log`, `log2`, `log10`, or no transformation.
`Show barplot`	Show the proportions-zero barplot.
`Indices`	Choose an index to plot (CPUE, Catch, or Effective skates).
`Colours`	Choose colours for: boxplots, line connecting index points, least squares trend line, index points, bars showing proportion zeroes, and number of data points used in the index calculation.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

Obradovich, S.G., Yamanaka, K.L., Cooke, K., Lacko, L.C. and Dykstra, C. (2008) Summary of non-halibut catch from the Standardized Stock Assessment Survey conducted by the International Pacific Halibut Commission in

British Columbia from June 4 to July 7, 2007. *Canadian Technical Report of Fisheries and Aquatic Sciences* **2807**, x + 83 p.

Schnute, J., Haigh, R., Krishka, B., Sinclair, A. and Starr, P. (2004) The British Columbia longspine thornyhead fishery: analysis of survey and commercial data (1996–2003). *Canadian Science Advisory Secretariat, Research Document* **2004/059**, 75 pp.

See Also

`iphc.rbr`, `getData`, `simBGtrend`
Available SQL queries: `SQLcode`
graphics: `boxplot`
stats: `lm`

---

ttget *Get/Print Objects From or Put Objects Into Temporary Work Environment*

Description

These functions are wrappers to the PBSmodelling accessor functions that get/print objects from or put objects into a temporary work environment, in this case `.PBStoolEnv`. Working objects include `PBStool`, which acts as a storage object for many of the functions.

Usage

```
ttget(...)
ttcall(...)
ttprint(...)
ttput(...)
tlisp(...)
```

Arguments

`...`	For `ttget` through to `ttput`, the only free argument is: `x` – name (with or without quotes) of an object to retrieve or store in the temporary environment; cannot be represented by a variable. Fixed arguments: `penv = parent.frame(), tenv = .PBStoolEnv` See `tget` for additional information. For `tlisp`, there is only one fixed argument: `pos = .PBStoolEnv` All other arguments are available – see `lisp`

Details

These accessor functions were developed as a response to the CRAN repository policy statement: "Packages should not modify the global environment (user's workspace)."

Value

Objects are retrieved from or sent to the temporary working environment to/from the place where the function(s) are called. Additionally, `ttcall` invisibly returns the object without transferring, which is useful when the object is a function that the user may wish to call, for example, `ttcall(myfunc)()`.

Note

Additional wrapper functions to access functions in `.PBStoolEnv` are named with the prefix `.win` (none at the moment).

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

References

CRAN Repository Policy: http://cran.r-project.org/web/packages/policies.html

See Also

`tget` and `lisp` in **PBSmodelling**

`weightBio` *Weight Age Frequencies/Proportions by Catch*

Description

Weight age or length frequencies||proportions by:
commercial: sample trip catch within quarters and quarterly catch within years, or
survey: sample species density (kg/km^2) in stratum and stratum area (km^2) in survey.

Usage

```
weightBio(adat, cdat, sunit="TID", sweight="catch",
     ttype=NULL, stype=c(0,1,2,5:8), ameth=3, sex=1:2, major=NULL,
     wSP=c(TRUE,TRUE), wN=TRUE, plus=60, Nmin=0, Amin=NULL,
     ctype="C", per=90, SSID=32, tabs=TRUE,
     plot=TRUE, ptype="bubb", size=0.15, powr=0.5, zfld="wp",
     clrs=list(c("blue","cyan"),c("green4","chartreuse")), cohorts=NULL,
     regimes=list(1900:1908,1912:1915,1923:1929,1934:1943,1957:1960,
                  1976:1988,1992:1998,2002:2006),  #PDO
     layout="portrait", wmf=FALSE, pix=FALSE, eps=FALSE, longside=10,
     ioenv=.GlobalEnv, ...)
```

Arguments

`adat`	data object of ageing information with fields that must include: `TID` = trip ID, `FEID` = fishing event ID, `date` = trip or fishing event date, `age` = age of fish, `ameth` = ageing method code, `sex` = sex code (1 = males, 2 = females), `major` = PMFC major area code, `ttype` = trip type code, `catch` = trip catch or fishing event catch.

cdat	data object of catch information with fields that must include: TID = trip ID, date = trip or fishing event date, year = trip or fishing event year, major = PMFC major area code, catch = trip catch or fishing event catch.
sunit	sample unit (e.g., TID for sampling at the trip level).
sweight	sample weighting field (e.g., catch or density).
ttype	trip type where: 1 = non-observed domestic, 2 = research, 3 = charter, 4 = observed domestic, 5 = observed j-v, 6 = non-observed j-v, 7 = polish commercial national, 8 = russian commercial national, 9 = russian commercial supplemental, 10 = polish commercial supplemental, 11 = recreational, 12 = japanese observed national, 13 = japanese commercial, 14 = unknown foreign.
stype	sample type where: 0 = unknown, 1 = total catch, 2 = random, 4 = selected, 5 = stratified, 6 = random from randomly assigned set, 7 = random from set after randomly assigned set, 8 = random from set requested by vessel master, 9 = selected juveniles, 10 = selected adults.
ameth	ageing method code (ameth=3 denotes break and burn).
sex	sex codes (1 = males, 2 = females).
major	major PMFC area codes (1=4A, 3=3C, 4=3D, 5=5A, 6=5B, 7=5C, 8=5D, 9=5E).
wSP	logical vector of length two for weighting by (i) sample catch and (ii) period (e.g., quarterly) commercial catch.
wN	logical scalar: if TRUE, weight the age frequencies else weight the age proportions.
plus	age of the plus group (accumulate ages >= plus).
Nmin	minimum number of specimens by year and sex in order to display bubbles.
Amin	minimum age to display on the y-axis if plot=TRUE; the default NULL means the function automatically finds the youngest age observed.
ctype	catch type where "C" = commercial and "S" = survey.
per	period length (days) within a year (e.g., 90 for quarterly periods).
SSID	survey series ID (can be a restratification series).
tabs	logical: if TRUE, create formatted tables for use in other applications.
plot	logical: if TRUE, plot the results as bubble plots.

ptype	plot type where "bubb" = bubbles and "bars" = horizontal bars.
size	specify size (inches) of largest bubble in all panels.
powr	power to scale bubbles, where powr=0.5 makes bubble area roughly proportional to the z-value.
zfld	field name containing values for output (e.g., "wp" for weighted proportions).
clrs	specify bubble colours indexed by sex code (0 = black).
cohorts	diagonal lines to plot indicating year-class cohorts (list of x- and y-values).
regimes	diagonal periods to plot indicating regime periods (e.g., positive PDO periods).
layout	string describing figure layout; one of: "portrait" for vertically stacked plots, "landscape" for plot placed horizontally side-by-side, "juxtapose" for plots overlaid on each other (not implemented at present).
wmf	logical: if TRUE, send figure to a .wmf file.
pix	logical: if TRUE, send figure to a .png file.
eps	logical: if TRUE, send figure to an .eps file.
longside	dimension (inches) of the longest side of the figure.
ioenv	input/output environment for function input data and output results.
...	additional arguments to pass to the function plotBubbles.

Details

The function uses values in the field sweight to weight the proportions-at-age (or frequencies-at-age). For example, if sweight="catch" then the first weighting for commercial samples might be by trip catch with a quarter year to yield a weighted proportion-at-age vector within each quarter. Similarly, the first weighting for a survey might be by sample catch within a stratum to yield a weighted proportion-at-age vector within each stratum. The second weighting would further weight these period/stratum vectors by total catch of the species in each period/stratum within a year/survey.

The function is not designed to scale up observations based on areal densities, though it can use this information to weight the proportions. For surveys in particular, if sweight="density", then the first weighting is by density of the species in each sample to derive a vector of weighted proportions for each stratum, and the second weightling is simply by the stratum area (km^2).

Weighting by density is treated differently here to 'emulate' a survey sample design. The code's algorithm usually takes data for the first weighting from the age object and data for the second weighting from the catch object. In the special case outlined in the previous paragragh, data for both weightings come from the age object.

Value

An array with the four dimensions: age, year, sex, and np = numbers/proportions.
The age array has two attributes attached:
wpatab = two-dimensional matrix of weighted proportions-at-age flattened by sex for input to the Awatea/Coleraine population model.
sumtab = supplementary array with dimensions year, per = period (quarters), and stat = statistic, where statistic is one of the following:
Nsid = no. samples,
Ntid = no. trips (which roughly equates to number of samples in the non-observed fishery),
Scat = sample unit catch (in our case the sample unit is the trip),
Fcat = commercial fishery catch,
Psamp = trip catch/fishery catch.

Author(s)

Rowan Haigh, Pacific Biological Station, Fisheries and Oceans Canada, Nanaimo BC

See Also

plotProp, reportCatchAge, requestAges
Available SQL queries: SQLcode

---

wrapText *Wrap, Mark and Indent a Long Text String*

Description

Wrap a long text string to a desired width, mark it with prefixes, and indent the lines following the first.

Usage

```
wrapText(string, width=50, prefix=c("> ","+ "), exdent=3)
```

Arguments

string	a text string.
width	a positive integer giving the target column for wrapping lines in the output.
prefix	character strings to be used as prefixes for the first and subsequent lines.
exdent	a non-negative integer specifying the indentation of subsequent lines.

Details

Using base::strwrap, this function splits a long text string into a target width, indents lines following the first, and adds a prefix string prefix[1] to the first line and prefix[2] to all indented lines.

Value

A text string with inserted prefixes and carriage returns. This output string can be sent to display functions like cat and text.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

revStr, showError

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  txt=wrapText(paste("USA state names:  ",
    paste(state.name,collapse=", "),sep=""),width=72,exdent=5)
  showMessage(txt,as.is=TRUE,adj=0,col="blue",cex=.9,x=.05)
  cat(txt,"\n"); invisible() }
pbsfun()
})
```

zapDupes *Zap Records with Duplicated Indices*

Description

Zap (remove) records from a data frame using an index to determine whether duplicated indexed records exist.

Usage

```
zapDupes(dat, index)
```

Arguments

dat	a data frame.
index	a vector of field names in dat that can be used to construct an index.

Details

The function orders the data frame based on the index, then removes records based on duplicated indices. There is no control over which of the duplicated records will be zapped. Pre-existing attributes other than names and row.names are transferred to the reduced object.

Value

An input data frame less records with duplicated indices.

See Also

zapHoles, processBio

Examples

```
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(ltmose07,envir=.PBStoolEnv)
  cat(paste("'ltmose07' has",nrow(ltmose07),"rows\n"))
  zap=zapDupes(ltmose07,"PID")
  print(zap,quote=FALSE)
  cat(paste("After zapping using index 'PID', there are",nrow(zap),"rows\n"))
  invisible() }
pbsfun()
})
```

zapHoles *Zap Holes That Will Be Filled Anyway*

Description

Zap (remove) holes from polygons that will ultimately be filled by solids from other polygons.

Usage

```
zapHoles(pset)
```

Arguments

pset	a valid PolySet object.

Details

The function attempts to remove holes that will ultimately be filled anyway. It uses centroid matching between holes and solids to identify candidate holes for removal.

The original rationale for this function is to groom a PolySet before using in `addPolys(pset,...,colHoles="white")`. The `colHoles` option was designed to get rid of retrace lines that appear in `.pdf` files made from metafiles.

That said, the PolySet created by this function essentially comprises layers. To see all the layers, the user must add them sequentially from largest to smallest, which isn't terribly efficient.

Value

A PolySet potentially modified to remove holes that will be filled. Non-essential attributes will be retained and supplemented with two additional ones created by `calcCentroid` and `calcArea`: `keep` = data frame summarizing the kept polygons, and `zap` = data frame summarizing the holes removed.

Author(s)

Rowan Haigh, Pacific Biological Station, Nanaimo BC

See Also

`calcSurficial`
PBSmapping: `addPolys`

Examples

```
## Not run:
local(envir=.PBStoolEnv,expr={
pbsfun=function(){
  data(qcb,envir=.PBStoolEnv); qcbz=zapHoles(qcb)
  pdata=attributes(qcb)$PolyData
  pdata=pdata[rev(order(pdata$area)),]
  expandGraph(mfrow=c(1,1),plt=c(.05,1,.05,1))
  plotMap(qcbz)
  for (i in 1:nrow(pdata)) {
    pid=pdata$PID[i]
    qcbi=qcbz[is.element(qcbz$PID,pid),]
    if (nrow(qcbi)==0) next
```

```
    addPolys(qcbi,col=pdata$col[i]) }
  invisible() }
pbsfun()
})

## End(Not run)
```

Index