PacFIN Comps Package

Documentation updated March, 2014

Contents:

Examples (in this document)

- Full_Petrale_Comps_Example.R
- EF2_example_code.R
- Petrale_Transcript.txt
- Figures generated in Petrale example.

Files included in this package:

- PacFIN Comps Package Documentation.docx
- PacFIN_Utilities.R
- Full_Petrale_Comps_Example.R
- EF2_example_code.R
- Petrale_Transcript.txt
- GearTable.csv
- Catch.DBRK.csv
- PacFIN.Expansion.Methods.xlsx
- PacFIN.PTRL.bds.11.May.2011.dmp
- PetraleCALCOM_Query2011.csv

PacFIN Comps Package Overview

Goals:

- Provide a user-friendly, flexible suite of tools to manipulate PacFIN biological data to prepare assessment input files reflecting those in recent assessments and with flexibility for future assessments.
- Provide diagnostics and summaries
- Allow custom stratification of the data
- Allow standardized expansions of the records
- Create Length, Age, and Age-at-Length comps according to desired stratifications of the data
- Write out resulting compositions in SS3 input format

Description

This package breaks out common tasks used to clean, combine, stratify, expand and create composition summaries from PacFIN data. The PaFIN data referred to here are those constructed by the BDS extraction query and R code that John Wallace maintains.

The functions that provide these services work by creating new data columns in the dataset, retaining the original columns with original values. For standardization within the package, some data columns are created as copies of originals, but with different names.

For example, some assessments use the standard calendar year, but others (e.g., Petrale) define a "fishing year" that treats months over the winter season as belonging to the same year. To permit this flexibility, the "fishyr" column is created as a copy of "SAMPLE_YEAR" for use throughout the package, and may be customized.

Each function's result must be explicitly assigned to a new variable; this permits the result of a function to be compared to its input, and aids in selection of function arguments to achieve the desired result. The first set of seven functions manipulates the dataset to add data columns used throughout the package.

This simplistic example filters the raw data and removes bad records, using the defaults for the many filtering arguments, then assigns the State (CA, OR or WA) to each record, using the default arguments for getState.

Example:

```
Pdata = cleanPacFIN(rawdata)
Pdata = getState(Pdata)
```

More elaborate examples follow as an appendix to this document. It was a design goal that all functions can be run in sequence using only the default arguments and produce a usable (though perhaps suboptimal) composition output.

A note on package development:

Some parts of this package are undergoing further development, particularly the expansions, which continue to be an active research project. It is expected, however, that the work-flow structure of the package will remain largely the same as described here.

Other extensions have been suggested, such as length-at-age comps, which are not yet in the existing package. Ultimately the code and documentation will become an R package.

This work represents input from the whole of the Assessment Team, without which this project could not have gone forward.

Individual Function Descriptions

CombineCALCOM = function (RawData, CALCOM)

CombineCALCOM converts the fieldnames and date formats in CALCOM data to be compatible with PacFIN data, and appends the rows of converted CALCOM data to the PacFIN dataset.

Returns: The combined data.

Arguments:

- Pdata the PacFIN dataset
- CALCOM the CALCOM dataset

The cleanPacFIN function should be run on the combined dataset. Note that it sets INPFC_AREA, SAMPLE_AGENCY, SOURCE_AGID and PSMFC_ARID to "CALCOM", since these fields must be populated, and so that records originating from CALCOM data can be readily identified.

cleanPacFIN = function(RawData,
only_USINPFC = FALSE,
keep_INPFC = NULL,
remove_INPFC = NULL,
badRecords = NULL,
keep_sample_type = c("", "M"),
keep_sample_method = "R",
keep_missing_lengths = FALSE,
CLEAN=TRUE) {

The cleanPacFIN function filters out unusable data, and standardizes lengths and weights. It initializes a number of fields used by other functions in the suite, and generates a brief report of the records removed.

"Standardizing" here involves preferentially assigning values to the new data columns, for example, the lengthcm column will contain the FORK_LENGTH, or, if it is missing, FISH_LENGTH, floored to the nearest centimeter.

Returns The input dataset with new fields appended and records and with bad or incomplete values removed. This function is intended for use on raw PacFIN data, as the first pass over the data.

Arguments:

- Rawdata PacFIN data extraction, with or without the addition of CALCOM data.
- USINPFC If TRUE, use only records with INPFC_AREA in US waters.
- keep_INPFC A list of INPFC areas to retain; all records from other areas removed.
- remove_INPFC A list of INPFC areas to remove; all others kept.
- badRecords Optional vector of SAMPLE_NOs to exclude

- sample_type A list of sample types to retain. Default: "" or **M** SAMPLE_TYPEs.
- sample_method A list of sample methods to retain. Default: only records with **R** SAMPLE_METHOD
- keep_missing_lengths If TRUE, keep records with no length, but which may have ages.
- CLEAN If FALSE, report bad records but do not remove them (for initial investigations of the data).

The original fields in the returned data are left untouched, with the exception of SEX, which is modified so that unidentified fish are labeled "U".

SAMPLE_TYPEs may be M=Market, R=Research, S=Special request, C=Commercial on-board. Only samples of type M are generally used.

SAMPLE_METHODs may be (R=Random, S=Stratified, N=Systematic, P=Purposive, X=Special). Only samples collected in random sampling are generally used.

Fields created by cleanPacFIN and available for use post-cleaning are:

lengthcm	FORK_LENGTH where available, otherwise FISH_LENGTH, in floored cm.
fishyr	initialized to SAMPLE_YEAR
fleet	initialized to 1.
fishery	initialized to 1.
season	initialized to 1.
exp_wgt	initialized to EXP_WGT and further addressed in the expansion functions.

These are used by the other functions in this suite.

```
getState = function ( Pdata,
                      source="SOURCE_AGID",
                      CLEAN=TRUE,
                      keep_PW = FALSE)
```

The getState function creates a state field that is developed from SOURCE_AGID by default.

Returns: the dataset with **state** encoded as a new column.

Arguments:

- Pdata PacFIN dataset
- source one of {SOURCE_ARID, PSMFC_ARID, SAMPLE_AGENCY}
- CLEAN TRUE by default. Removes records for which a state could not be assigned.
- keep_PW retain records with SAMPLE_AGENCY "PW". These are from the hake (pacific whitefish) fishery

getState is intended to be run after running cleanPacFIN and before doing the data expansions.

Prints a report of the number of records removed for lack of State information.

getSeason = function (Pdata,
season_type=-1,
yearUp=NULL,
yearDown=NULL,
plotResults=F))

The getSeason function modifies the 'season' field created in cleanPacFIN according to SAMPLE_MONTH. Additional seasonal schemes may be provided, but for now it only encodes the Petrale fishing seasons (1 = winter months, 2 otherwise).

Returns: the dataset with **season** encoded.

Arguments:

- Pdata PacFIN dataset
- season_type one of {-1, 0 , 1}
- yearUp A list of months for which the fishyr should be incremented.
- yearDown A list of months for which the fishyr should be decremented.
- plotResults create a stacked barplot of records by year and season.

Season types:

- -1 Pdata$season = 1
- 0 assigns numeric values 1-12 from SAMPLE_MONTH
- 1 Encodes November-February as season 1; 2 for the remaining months

For example, petrale fished in the winter months are considered as the next year's catch, so the fishyr associated with records from months 11 and 12 (November and December) is incremented by one.

getGearGroups = function(Pdata)

Returns: the dataset with the **geargroup** associated with each GRID encoded

Requires: the file GearTable.csv, which is provided with this package.

Arguments:

- Pdata the PacFIN dataset

getGearGroups reads the gear table, if present, and matches the GRID values to their geargroups in the table, creating an extra column in Pdata.

The geargroup can reduce the GRID by a factor of 4 or more, and can be useful for creating gear-based comps and for the level 2 expansion.

The geargroup field is not required by any other functions.

```
cleanAges = function( Pdata,
                      keep_age_methods=c("B","S",""),
                      minAge=0,
                      maxAge=NULL,
                      CLEAN=TRUE ) {
```

The cleanAges function filters out unusable data, and standardizes the ages, and agemethods. It initializes the **age** and **agemethod** fields used by other functions in the suite, and generates a brief report of the records removed. cleanAges depends on the data having first been filtered with cleanPacFIN, and is intended to be run *after* length comps have been generated and before age or age-at-lengths are compiled.

Returns: the input data with ages modified according to the arguments given, and prints a brief report of the records removed. No new columns are created.

Arguments:

- Pdata PacFIN dataset
- keep_age_methods Vector of the agemethods (originally AGE_METHODS)
- minAge Records with age less than minAge will be excluded. Note that cleanPacFIN has set age to -1 where no age was found; these ages will be less than the default minAge=0.
- maxAge If given a value, any age greater than maxAge is set to maxAge.

Fields created by cleanAges and available for use post-cleaning are:

age created using FISH_AGE_YEARS_FINAL, age1, age2, or age3, in order of preference.

agemethod recodes AGE_METHODs 1 and 2 to “B” and “S”, respectively

The AGE_METHODs possibly present in the original data are:

WDFW: B-break and burn; L-length; N-not aged; O-optical scanner; X-sectioning

ODWF: 1-break and burn; 2-surface; 3-scales; 4-thin section; 5-optical scanner; 6-length; 9-unable)

CDFW: B-break and burn; S-surface; T-thin section

In order to investigate different aging strategies, you may want to create several copies of the data:

```
Pdata_ageTest1 = cleanAges(Pdata, minAge=1)
Pdata_ageTest2 = cleanAges(Pdata, keep_methods=c(“S”))
```

Stratify = function (inVector=NULL, splits=NULL, names=NULL, numeric=F)

Unlike the other functions in this suite, Stratify does not work on the PacFIN dataset; instead it works on a single input column (field) from the PacFIN dataset, and returns a numeric- or character-encoded column of the same length as the input.

Columns created by Stratify may be used for stratification in generating comps.

Returns: a vector the length of the input vector encoding the user-defined strata ("splits"). Intended as a column to be appended to the PacFIN dataset.

Arguments:

- inVector Vector of values to encode
- splits list of values to be encoded as {1,2,3…} . Either a list of character strings, or a list of numeric breaks. Input values that are not among those in 'splits', or are smaller than the smallest numerical value of the splits, are encoded as 0.
- names list of character strings to be substituted for the values in splits.
- numeric if TRUE, treat 'splits' as numerical ranges rather than character strings. The input data expected are data that have already been aggregated by the getComps function.

Stratify works on either text (character) valued fields, or numerical fields. For text-based data, **numeric** should be FALSE. If **numeric** is TRUE, findInterval is called, and any values smaller than the first value in **splits** will be encoded as 0.

Text-based example:

```
# Assign fleet ID based on GEAR.  This creates four fleets; the second and fourth
# fleets each composed of two gears.  Any records with GEAR other than the
# ones input will be assigned to fleet 0.
```

```
Pdata$fleet = Stratify( Pdata$GEAR,
                        splits=list("GFS", c("MDT","TB"), "TR", c("LGL","FTS")) )

head(Pdata$fleet)
[1] 0 0 0 1 1 2 1 2 1 3 3 4 1 1 0

# Substituting names for groupings of GEARS.  Note that “0” values appear
# where the GEARS didn’t match the groupings given in splits.

Pdata$fleet = Stratify( Pdata$GEAR,
                        splits=list("GFS", c("MDT","TB"), "TR", c("LGL","FTS")),
                        names=list(“Happy”,”Sleepy”,”Doc”,”Grumpy”) )

sort(unique(Pdata$fleet))
[1] "0"      "Doc"    "Grumpy" "Happy"  "Sleepy"
```

Numerically-based example:

```
# Stratify by DEPTH_AVG range

Pdata$use_depth = Stratify( Pdata$DEPTH_AVG, splits=c(0, 50, 100, 250, 500),
numeric=T)
```

getExpansion_1 = function(Pdata, maxExp = 0.95, Indiv_Wgts = T,
fa=2e-06, fb=3.5, ma=2e-06, mb=3.5, ua=2e-06, ub=3.5)

getExpansion_1 prepares the values that are used to calculate the level-1 expansion factor, which is the per-trip ratio of the sampled pounds of the species to the total weight of the sampled fish.

This function became unwieldy, so getExpansion_1 now calls two helper functions for the numerator and denominator calculations. These are what is documented here.

Returns:

Pdata with new columns including Expansion_Factor_1

There is a major difficulty in working with this data to expand the sampled weight of fish to total weights for the species. This is addressed by a series of calcuations intended to provide the maximum number of possible candidates for missing values for the final calculation.

These candidate values are then used according to preference based on the way the data are entered. Note that these preferences differ among the data provided by different state agencies.

Arguments:

- Indiv_Wgts controls whether or not individual weight calculations (where W = aL^b) will be used for the expansion, which is intended to account for the unsampled fish in the tow.

- maxExp is either a number or a quantile determining the maximum value the expansion factor can take. Numbers smaller than 1 are interpreted as quantiles.

- a and b arguments are the coefficient and exponent values for the length-weight relationships for females (fa, fb), males (ma, mb) and unsexed fish (ua, ub). **Note** that the default values are those for petrale sole, so that at least one assessment is served by the defaults.

The total weight of the fish is calculated three ways, by the helper function EF1_Denominator.

1. Wt_Sampled_1: FEMALES_WGT and MALES_WGT are summed per SAMPLE_NO.

2. Wt_Sampled_2: The SPECIES_WGT is summed across all clusters in a sample per SAMPLE_NO.

3. Wt_Sampled_3: weights of the male and female fish are calculated from their lengths. These are summed per SAMPLE_NO. Zero weights might occur for some fish; these are filled in with the median weight of fish in the sample.

Wt_Sampled is Wt_Sampled_1 with missing values filled by Wt_Sampled_2, and any remaining missing values filled by Wt_Sampled_3. Wt_Method records which method was used to calculate each weight.

Next, the helper function EF1_Numerator calculations are made towards finding the sampled weight of the species per trip, Trip_Sampled_Lbs:

- Use_acs is all_cluster_sums per SAMPLE_NO, with missing values filled in with year-specific, state-specific medians (or year-specific overall median if these are missing).
- Species_Percent_Sampled is the per SAMPLE_NO Wt_Sampled divided by Use_acs.

Trip_Sampled_Lbs is set to the default value TOTAL_WGT. This is then replaced by calculations specific to each state:

- California: Trip_Sampled_Lbs = Species_Percent_Sampled * TOTAL_WGT.

- Oregon: Trip_Sampled_Lbs = exp_wt. Where missing, use Species_Percent_Sampled * TOTAL_WGT.

- Washington: use RWT_LBS as first preference, then TOTAL_WGT, year- and state-specific median(RWT_LBS) and finally, year- and state-specific median(TOTAL_WGT).

- If all else fails, use year- and state-specific medians.

If none of these values is available, TOTAL_WGT remains as the default value.

Once all this has been calculated, Expansion_Factor_1 is Trip_Sampled_Lbs / Wt_Sampled. If this value is less than 1, it is corrected to 1.

The maximum value is then capped at the value (or quantile) given in maxExp. If no maximum is desired, then call the function with maxExp = Inf.

No original data columns are altered in this process. The original data are supplemented with new columns containing the values to be used.

New columns:

- Wt_Sampled_1 per-SAMPLE_NO summed MALES_WGT + FEMALES_WGT
- Wt_Sampled_2 per-SAMPLE_NO summed cluster_wt
- LW_Calc_Wt individual weights predicted from L-W
- Wt_Sampled_3 per-SAMPLE_NO summed LW_Calc_Wts

- Wt_Sampled per-SAMPLE_NO combined weights.

  This is preferentially Wt_Sampled_1, with NAs replaced with values from Wt_Sampled_2, and remaining NAs replaced with values from Wt_Sampled_3.

- Wt_Method per-SAMPLE_NO method by which Wt_Sampled was obtained: 1,2, or 3.

- Use_acs all_cluster_sum with NAs replaced by year- and state-specific medians.

- Species_Percent_Sampled Percentage of this species in samples.
- Trip_Sampled_Lbs Calculated sampled weights. Where medians were used, they are year- and state-specific.
- Expansion_Factor_1 Trip_Sampled_Lbs / Wt_Sampled

Summary tables and plots are generated throughout this process.

getExpansion_2 = function (Pdata, Catch, Convert = FALSE, maxExp=0.95)

Expansion_Factor_2 is the return value; this is the second-stage per-trip, per-year, per-state, per-gear expansion; the catch / sampled catch.

Prepare for the level 2 expansion by encoding a column called “usegear” with the Stratify function so that the names, for example, “TWL” and “NONTWL” or “POT” and “TWL” will match the names given to the catch.

Names as in usegear: TWL POT NONTWL HKL

will be joined with the value in the “state” column, to match the names in the Catch data.

Names given to the Catch: TWL.CA TWL.OR and TWL.WA

Returns:

Pdata with a new column for Expansion_Factor_2.

Arguments:

- Pdata Dataset
- Catch Catch data already read in separately.
- Convert if TRUE, convert from metric tons to pounds.
- maxExp maximum expansion as a quantile (if less than one), or as a maximum value.

To avoid capping the return value, call the function with maxExp = Inf.
Example Catch Table:

Here there are six fisheries. The names of the fisheries in line 1 must match the names of the fisheries in the data that are created by the function when it pastes together “state” and “usegear” (with a “.” between).

Year, CA.TWL, OR.TWL, WA.TWL, CA.NONTWL,OR.NONTWL, WA.NONTWL
1977,117.5965435,130.2747312,62.2,5.635431605,0.000183558,0
1978,45.37964866,155.9795058,199.4,14.27727344,0.086554369,0
1979,135.0262839,496.3244793,87.6,13.04123423,0.386793722,0

getComps = function(Pdata, strat=NULL, Comps="AAL")

The getComps function expands the records by Final_Sample_Size, and aggregates the data by length, by age, or by age-at-length according to the given stratification.

Note that the aggregation is of the Pdata$Final_Sample_Size value, which should be set to the desired expansion before running getComps, e.g.

Pdata$Final_Sample_Size = Pdata$Expansion_Factor_1

or

Pdata$Final_Sample_Size =
Pdata$Expansion_Factor_1 * Pdata$Expansion_Factor_2

The **default stratification** is by fleet, fishyr, and season, and the lengthcm, age or both are appended to the depending on the "Comps" argument.

Returns: a “comps” object, consisting of the compositions expanded by Final_Sample_Size and aggregated by stratum and gender. This comprises the numbers in each stratum, the number of samples (individual fish) contributing to each stratum, and the number of tows in each stratum for female, male, and unsexed fish.

Arguments:

- Pdata Dataset
- strat List of additional column headings for stratification
- Comps One of “LEN”, “AGE”, or “AAL”

doSexRatio = function(CompData,
findRatio = FALSE,
Rvector=0.5,
Bins=NULL,
ratioU=.5,
maxsizeU=0,
GTsizeU=Inf)

doSexRatio determines gender for unsexed fish. Inspired by (stolen from?) Allan Hicks' similar code for NW survey data. The input data expected are data that have already been aggregated by the getComps function.

Returns: CompData object with all fish assigned to a gender. Note that comps are returned with original sizes for unsexed {fish, sample sizes and tows}, however the values for male and female {fish, sample sizes and tows} are increased.

Arguments:

- Compdata Comps object returned by getComps
- findRatio Calculate the per-stratum ratios and apply them to the unsexed fish.
- Rvector A ratio or vector of ratios of female to male fish to be applied over Lbins.
- Bins If Rvector is of length > 1, a vector of corresponding length bins to which the Rvector ratios should be applied. Unused if Rvector is a single number. If working up AGE comps, this should be a vector of ages.
- ratioU Ratio applied to fish at or below maxsizeU, which is a small slze below which fish cannot reliably be sexed. Usually 0.5.
- maxsizeU A small size below which fish cannot be reliably sexed. At or below this length, ratioU applies.
- GTsizeU the size above which all fish are female (big mamas).

doSexRatio is meant to be used in one of four ways:

1. Use findRatio = TRUE to allow the existing ratios to be applied per stratum.

2. Designate a single Rvector to apply to all lengths (e.g., 0.5).

3. Designate a small size with maxsizeU below which ratioU is applied, a middle range to which a single Rvector value is applied, and an optional GTsizeU above which all fish are female.

4. Designate a vector of Lbins over which to apply Rvector ratios. It is user error in this case to leave out lengths that occur in the data and which have unsexed fish. These fish will remain unsexed.

Future work: Test for the user error mentioned above and add a warning. Add a value for MMM – Massive Mature Males, analogous to GTsizeU.

writeComps = function(inComps, fname="out.csv", abins=NULL, lbins=NULL, maxAge=Inf, partition=0, ageErr=0)

Writes out comps with length (and/or) age bins, or using lengthcm and age columns from the data. The comps written will depend on the input comps – Length, Age, or Age-at-Length comps generated by getComps, with unsexed fish having first been assigned a gender by doSexRatio.

Returns: Returns nothing.
Writes comps to an output file, by default called, “out.csv”.

Arguments:

- InComps Comps from getComps, with doSexRatio applied
- fname Output filename
- abins Age bins
- lbins Length bins
- maxAge maximum age
- ageErr Age error type (1,2, etc.)

The comps are written out for:

- Females followed by males
- Females only
- Males only
- Combined sexes

doDiags = function(Pdata, fname=NULL)

Write a set of diagnostics of raw PacFIN data to the screen, and plots to a pdf file.

Note that the diagnostics can be explicitly written to a file using the “sink” command, which sends output to both console and file:

```
sink("My.log.text", split=T)

… < do stuff > …

sink()  # Closes the file.
```

plotCleaned = function(Pdata)

Plot a set of diagnostics for the post-processed dataset.

Plots:

- Lengths by state and year.
- Boxplot of lengths per year.
- Gears by year.
- Average Depth by year.

If getGearGroups has been run:

- GearGroup by year.

plotCleanedAges = function(Pdata)

Plot a set of diagnostics for the post-processed dataset.

Plots:

- Ages by state and year.
- Boxplot of ages per year.

capValues = function(DataCol, maxVal)

Arguments:

- DataCol a single column of numeric values
- maxVal a quantile (< 1) or maximum value (> 1) at which to cap the value of the input data column

Returns:

the input data column with outsize values replaced by the maximum.

Examples:

Pdata$Expansion_Factor_1 = capValues(Pdata$Expansion_Factor_1, 300)

This caps Expansion_Factor_1 at 300.

or

Pdata$Final_Sample_Size = capValues(Pdata$Final_Sample_Size, 0.90)

This caps Final_Sample_Size at its 90^{th} percentile.

Utility functions

- paste.col
- find.matching.rows

Full_Petrale_Comps_Example.R

```
################################################################################
#
# Example run of comps creation, start to finish.
#
################################################################################

setwd("~/Desktop/PacFIN_Comps_Beta_V1")

source("PacFIN_Utilities.R")

# Get data
load("PacFIN.PTRL.bds.11.May.2011.dmp")
CALCOM = read.csv("PetraleCALCOM_Query2011.csv", as.is=T)

# Combine
Pdata = combineCALCOM(PacFIN.PTRL.bds.11.May.2011, CALCOM)

# Clean and filter for lengths

Pdata = cleanPacFIN(Pdata)

# Further filtering and preparation for expansions

Pdata = getState(Pdata)
Pdata = getSeason(Pdata, season_type=1, yearUp=c(11,12))
Pdata = getGearGroups(Pdata)

# Plot the length data

plotCleaned(Pdata)

# Get the level-1 expansion

Pdata = getExpansion_1(Pdata)

# Run the example for the second-level expansion

source("EF2_example_code.R", echo=T)

# set the final sample size, evaluate it and cap it at the 90th percentile value.

Pdata$Final_Sample_Size = Pdata$Expansion_Factor_1 * Pdata$Expansion_Factor_2
summary(Pdata$Final_Sample_Size)
Pdata$Final_Sample_Size = capValues(Pdata$Final_Sample_Size, maxVal = 0.90)
summary(Pdata$Final_Sample_Size)

# Filter the length data for ages and plot them, AFTER getting the Final_Sample_Size

AgeData = cleanAges(Pdata, minAge = 1)
plotCleanedAges(AgeData)


# From here, the steps for each comps file is:  get the comps, assign unsexed fish
# to be males or females, then write out the comps.
```

```
# Get the Length comps.  Let the sex ratio be a stratum-by-stratum value
# as seen in the data.

Lcomps = getComps(Pdata, Comps="LEN")
Lcomps = doSexRatio(Lcomps, findRatio=T)
Lcomps = writeComps(Lcomps, fname="Lcomps.csv", lbins=seq(0,90,5))

# Now get Age comps stratified by state and usegear as well as the usual stratifying fields.
# Assign sex ratio according to a ramp up from 0.5 to 1.

Acomps = getComps(AgeData, strat=c("state", "usegear"), Comps="AGE")

# Need (age) Bins and Rvector the same length

length(seq(0,40,4))
length(seq(0.5, 1, 0.1))
length(seq(0.5, 1, 0.05))
length(seq(0.5, 1, 0.051))

Acomps = doSexRatio(Acomps, Rvector = seq(0.5, 1, 0.051), Bins=seq(0,90,10))

writeComps(Acomps, fname=("Acomps.csv"))


# Now the Age-at-Length comps, stratified by state in addition to the usual stratifying fields.

AALcomps = getComps(AgeData, strat=c("state"), Comps="AAL")

# Sex ratio is 0.5 for unsexed fish up to 15 cm, 0.7 until 75 cm, then 1.

AALcomps = doSexRatio(AALcomps, Rvector=0.7, ratioU = 0.5, maxsizeU = 15, GTsizeU = 75)

writeComps(AALcomps, fname="AALcomps.csv", lbins=seq(0,90,5))

#
# That's All, Folks!
#
################################################################################
```

EF2_example_code.R

```
################################################################################
#
# Expansion 2 example.  Example catch file is from DBRK, supplied by Vlada.

Catch = read.csv("Catch.DBRK.csv", skip=1, as.is=T)

# Rename Catch columns

names(Catch) = c("Year",
"CA.TWL","OR.TWL","WA.TWL","CA.NONTWL","OR.NONTWL","WA.NONTWL")

# Encode the two gears to use in this example
# Note that the state value will be pasted to the usegear to create the full name.

Pdata$usegear = "NONTWL"
Pdata$usegear[Pdata$geargroup %in% c("TWL","TWS")] = "TWL"

# No WA.NONTWL in example Pdata (from PTRL), remove column 7 for this example to run.

Catch = Catch[,-7]

# Get the expansion, converting the metric tonnes of catch to pounds

Pdata = getExpansion_2(Pdata, Catch, Convert=T, maxExp=0.9)

#
# That's All, Folks!
#
################################################################################
```

Petrale_Transcript.txt

```
> source("Full_Petrale_Comps_Example.R", echo=T)

>
################################################################################
#
> #
> # Example run of comps creation, start to finish.
> #
> ####### .... [TRUNCATED]

> source("PacFIN_Utilities.R")

> # Get data
> load("PacFIN.PTRL.bds.11.May.2011.dmp")

> CALCOM = read.csv("PetraleCALCOM_Query2011.csv", as.is=T)

> # Combine
> Pdata = combineCALCOM(PacFIN.PTRL.bds.11.May.2011, CALCOM)

Combining CALCOM and PacFIN data

PacFIN records: 179078

CALCOM records: 53794

Combined dataset: 232872


> # Clean and filter for lengths
>
> Pdata = cleanPacFIN(Pdata)

Cleaning data


Removal Report

Records in input:                232872
Records not in INPFC_AREA:         0
Records in badRecords list:        0
Records with bad SAMPLE_TYPE       4078
Records with bad SAMPLE_METHOD     0
Records with no SAMPLE_NO          0
Records with no usable length      1107
Records remaining:               227687

In addition, these values have been initialized for use when comps are generated.
Use Stratify and getSeason to reset them to appropriate values.

Pdata$fishyr = Pdata$SAMPLE_YEAR
Pdata$fleet = 1
Pdata$fishery = 1
Pdata$season = 1
```

```
> # Further filtering and preparation for expansions
>
> Pdata = getState(Pdata)

Getting state information from SOURCE_AGID


Original data:

    C CALCOM      O      W
 19128  53781  62061  92717


Data retained:

   CA    OR    WA
72909 62061 92717


 0  records were removed because no state id could be assigned


> Pdata = getSeason(Pdata, season_type=1, yearUp=c(11,12))

Default season = 1

Assigning season ala Petrale; winter is season 1, summer is 2.

Incremented fishyr for months 11 12 to the next year.


> Pdata = getGearGroups(Pdata)

> # Plot the length data
>
> plotCleaned(Pdata)

> # Get the level-1 expansion
>
> Pdata = getExpansion_1(Pdata)

Getting level-1 expansions


Individual weights will be generated from the following values:

Females: 2e-06 3.5 Males: 2e-06 3.5 Unknowns: 2e-06 3.5


Getting the summed weights of the sexed fish for each SAMPLE_NO


Summing the SPECIES_WEIGHTS for all clusters in a SAMPLE_NO
```

Calculating individual weights from lengths.

Done calculating sample weights

```
      M+F           SPECIES_WT          L-W           Final Wt_Sampled
 Min.   : 10.4    Min.   :  1.00   Min.   :   1.336   Min.   :   1.0
 1st Qu.: 50.5    1st Qu.: 25.00   1st Qu.: 186.866   1st Qu.: 123.5
 Median : 84.8    Median : 47.00   Median : 359.139   Median : 271.2
 Mean   :111.2    Mean   : 39.51   Mean   : 931.676   Mean   : 870.9
 3rd Qu.:156.0    3rd Qu.: 50.00   3rd Qu.:1399.556   3rd Qu.:1399.6
 Max.   :885.0    Max.   :104.00   Max.   :8941.261   Max.   :8941.3
 NA's   :179607   NA's   :208559
```

Wt_Methods:

```
    1      2      3
48080  19128 160479
```

Sample Wts found for all samples.

Getting cluster sums

Getting total weights per sample

Warning: data does not contain column RWT_LBS required for WA data

Sampled pounds per trip:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0     644    2887    9048   10000  137000   29966
```

29966 NA Expansion_Factor_1 values replaced by 1.

Capping Expansion_Factor_1 at 0.95

Maximum expansion capped at 0.95 quantile: 103.68

Expansion Factor 1:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   5.169  16.960  17.380 103.700
```

```
> # Run the example for the second-level expansion
>
> source("EF2_example_code.R", echo=T)
```

```
>
################################################################################
#
> #
> # Expansion 2 example.  Example catch file is from DBRK, suppli .... [TRUNCATED]

> # Rename Catch columns
>
> names(Catch) = c("Year",
"CA.TWL","OR.TWL","WA.TWL","CA.NONTWL","OR.NONTWL","WA.NONTWL")

> # Encode the two gears to use in this example
> # Note that the state value will be pasted to the usegear to create the full name.
>
> Pdata$usegea .... [TRUNCATED]

> Pdata$usegear[Pdata$geargroup %in% c("TWL","TWS")] = "TWL"

> # No WA.NONTWL in example Pdata (from PTRL), remove column 7 for this example to run.
>
> Catch = Catch[,-7]

> # Get the expansion, converting the metric tonnes of catch to pounds
>
> Pdata = getExpansion_2(Pdata, Catch, Convert=T, maxExp=0.9)
Converting Catch to pounds (multiplying by 2204).

.....Assigned catch for CA.TWL
.....Assigned catch for OR.TWL
.....Assigned catch for WA.TWL
.....Assigned catch for CA.NONTWL
.....Assigned catch for OR.NONTWL
No Catch was found for these combinations:

      NoCatchYr NoCatchSG
 [1,] "1955"    "WA.TWL"
 [2,] "1956"    "WA.TWL"
 [3,] "1957"    "WA.TWL"
 [4,] "1958"    "WA.TWL"
 [5,] "1959"    "WA.TWL"
 [6,] "1960"    "WA.TWL"
 [7,] "1961"    "WA.TWL"
 [8,] "1964"    "WA.TWL"
 [9,] "1965"    "WA.TWL"
[10,] "1966"    "OR.TWL"
[11,] "1966"    "WA.TWL"
[12,] "1967"    "OR.TWL"
[13,] "1968"    "OR.TWL"
[14,] "1967"    "WA.TWL"
[15,] "1969"    "OR.TWL"
[16,] "1968"    "WA.TWL"
[17,] "1970"    "OR.TWL"
[18,] "1969"    "WA.TWL"
[19,] "1970"    "WA.TWL"
[20,] "1971"    "OR.TWL"
[21,] "1971"    "WA.TWL"
```

```
[22,] "1972"    "WA.TWL"
[23,] "1972"    "OR.TWL"
[24,] "1973"    "WA.TWL"
[25,] "1973"    "OR.TWL"
[26,] "1974"    "WA.TWL"
[27,] "1974"    "OR.TWL"
[28,] "1975"    "WA.TWL"
[29,] "1975"    "OR.TWL"
[30,] "1976"    "WA.TWL"
[31,] "1948"    "CA.NONTWL"
[32,] "1949"    "CA.NONTWL"
[33,] "1962"    "CA.NONTWL"
[34,] "1964"    "CA.NONTWL"
[35,] "1965"    "CA.NONTWL"
[36,] "1966"    "CA.NONTWL"
[37,] "1967"    "CA.NONTWL"
[38,] "1968"    "CA.NONTWL"
[39,] "1969"    "CA.NONTWL"
[40,] "1970"    "CA.NONTWL"
[41,] "1971"    "CA.NONTWL"
[42,] "1972"    "CA.NONTWL"
[43,] "1973"    "CA.NONTWL"
[44,] "1974"    "CA.NONTWL"
[45,] "1975"    "CA.NONTWL"
[46,] "1976"    "CA.NONTWL"


Summary of Expansion_Factor_2

   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
   1.00     1.00     1.00     8.04     5.98 11810.00   116142

 116142 NA Expansion_Factor_2 values replaced by 1.


Maximum expansion capped at 0.9 quantile: 7.620436

Summary of Expansion_Factor_2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   1.851   1.000   7.620

Remember to set (or reset) Pdata$Final_Sample_Size


> #
> # That's All, Folks!
> #
>
################################################################################
#

> # set the final sample size, evaluate it and cap it at the 90th percentile value.
>
> Pdata$Final_Sample_Size = Pdata$Expansion_Factor_1 * Pdata$Ex .... [TRUNCATED]
```

```
> summary(Pdata$Final_Sample_Size)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.026   6.261  36.520  29.190 790.100

> Pdata$Final_Sample_Size = capValues(Pdata$Final_Sample_Size, maxVal = 0.90)

Maximum expansion capped at 0.9 quantile: 103.68


> summary(Pdata$Final_Sample_Size)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.026   6.261  23.740  29.190 103.700

> # Filter the length data for ages and plot them, AFTER getting the Final_Sample_Size
>
> AgeData = cleanAges(Pdata, minAge = 1)

Cleaning Age data.


Removal report

Records in input:                227687
Records with age less than min:     156101
Records with bad agemethods:        17986
Records remaining:                53600

> plotCleanedAges(AgeData)

> # From here, the steps for each comps file is:  get the comps, assign unsexed fish
> # to be males or females, then write out the comps.
>
> # Get  .... [TRUNCATED]

Aggregating, stratification is by fleet, fishyr, season, lengthcm


Getting Males


Getting Females


Getting Unsexed

Getting tows and samples


> Lcomps = doSexRatio(Lcomps, findRatio=T)

Applying per-stratum sex ratio.


Summary of sex ratios observed per stratum:

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

0.0000 0.3292 0.7822 0.6519 1.0000 1.0000

> Lcomps = writeComps(Lcomps, fname="Lcomps.csv", lbins=seq(0,90,5))
Writing comps to file Lcomps.csv

Note that if you didn't run doSexRatio, all unsexed fish disappear at this point.

Bins:

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 91
4 9 14 19 24 29 34 39 44 49 54 59 64 69 74 79 84 89 91 92

Note that last bin is a dummy bin

110 unique keys for 3514 records

Assembling, sex is: m
Assembling, sex is: f
Assembling, sex is: b

Writing FthenM, dimensions: 110 46
Writing F only, dimensions: 110 46
Writing M only, dimensions: 110 46
Writing combined sexes as females, dimensions: 110 46

> # Now get Age comps stratified by state and usegear as well as the usual stratifying fields.
> # Assign sex ratio according to a ramp up from 0.5 to .... [TRUNCATED]

Aggregating, stratification is by state, usegear, fleet, fishyr, season, agemethod, age

Getting Males

Getting Females

Getting Unsexed

Getting tows and samples

> # Need (age) Bins and Rvector the same length
>
> length(seq(0,40,4))
[1] 11

> length(seq(0.5, 1, 0.1))
[1] 6

> length(seq(0.5, 1, 0.05))
[1] 11

> length(seq(0.5, 1, 0.051))

[1] 10

> Acomps = doSexRatio(Acomps, Rvector = seq(0.5, 1, 0.051), Bins=seq(0,90,10))

Applying sex ratio: 0.5,0.551,0.602,0.653,0.704,0.755,0.806,0.857,0.908,0.959 to numbers, samples and tows

Done.

> writeComps(Acomps, fname=("Acomps.csv"))
Writing comps to file Acomps.csv

Note that if you didn't run doSexRatio, all unsexed fish disappear at this point.

No age bins provided, using data as-is

Abins:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 32 33

Note that last bin is a dummy bin

140 unique keys for 1447 records

Assembling, sex is: m
Assembling, sex is: f
Assembling, sex is: b

Writing FthenM, dimensions: 140 73
Writing F only, dimensions: 134 73
Writing M only, dimensions: 135 73
Writing combined sexes as females, dimensions: 140 73

> # Now the Age-at-Length comps, stratified by state in addition to the usual stratifying fields.
>
> AALcomps = getComps(AgeData, strat=c("state"), ... [TRUNCATED]

Aggregating, stratification is by state, fleet, fishyr, season, agemethod, lengthcm, age

Getting Males

Getting Females

Getting Unsexed

Getting tows and samples

```
> # Sex ratio is 0.5 for unsexed fish up to 15 cm, 0.7 until 75 cm, then 1.
>
> AALcomps = doSexRatio(AALcomps, Rvector=0.7, ratioU = 0.5, maxsizeU = .... [TRUNCATED]

Applying sex ratio: 0.5,0.7,1 to numbers, samples and tows


Done.


> writeComps(AALcomps, fname="AALcomps.csv", lbins=seq(0,90,5))
Writing comps to file AALcomps.csv

Note that if you didn't run doSexRatio, all unsexed fish disappear at this point.

Bins:

0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 91
4 9 14 19 24 29 34 39 44 49 54 59 64 69 74 79 84 89 91 92

Note that last bin is a dummy bin


No age bins provided, using data as-is

Abins:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 32 33

Note that last bin is a dummy bin



Age-at-length takes awhile to assemble.  Be patient!
793 unique keys for 11478 records



Assembling, sex is: m
Assembling, sex is: f
Assembling, sex is: b

Writing FthenM, dimensions: 793 74
Writing F only, dimensions: 728 74
Writing M only, dimensions: 507 74
Writing combined sexes as females, dimensions: 793 74

> #
> # That's All, Folks!
> #
>
################################################################################
#
>
```

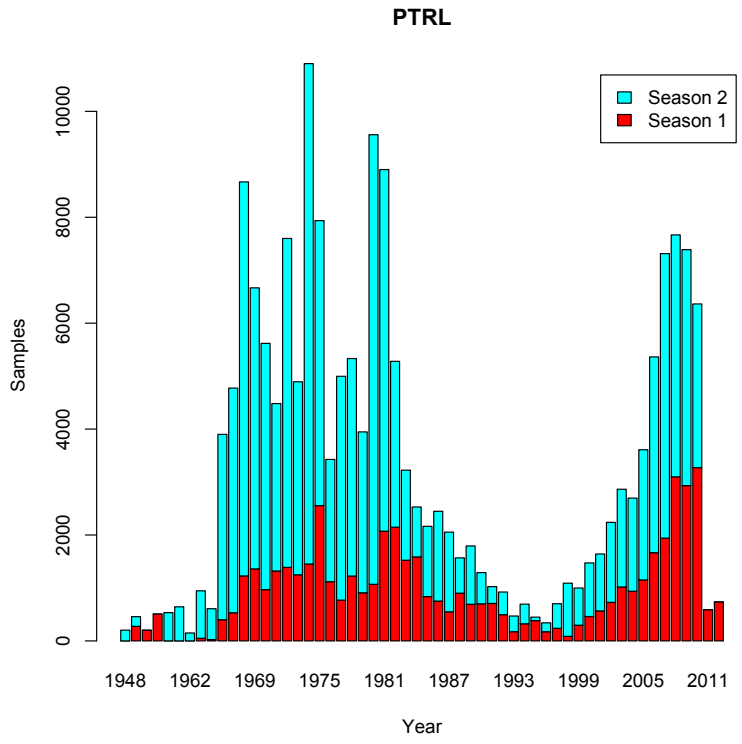

Figure 1. Season plot from Petrale example.

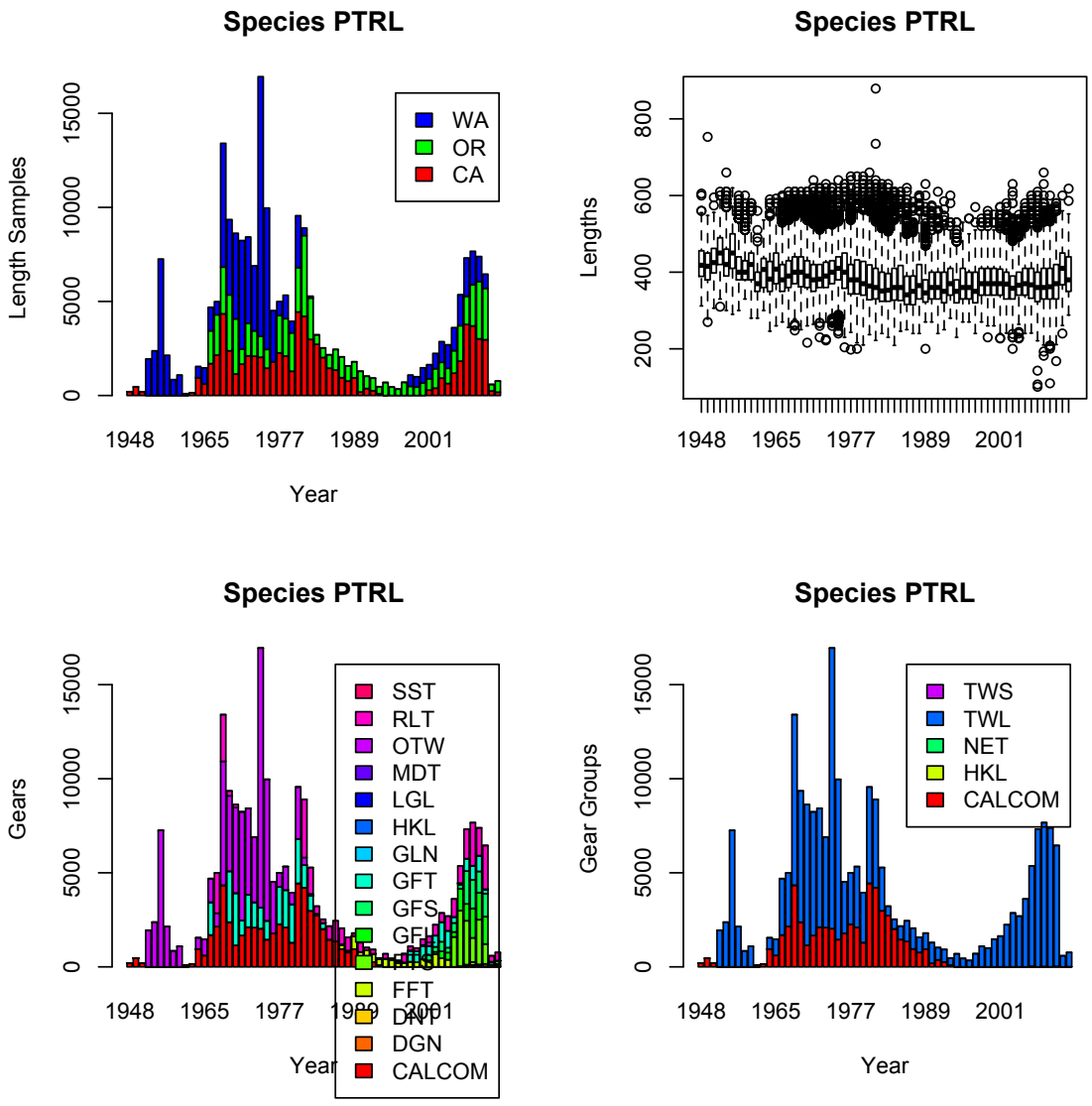

Figure 2. Page 1 of plotCleaned output.

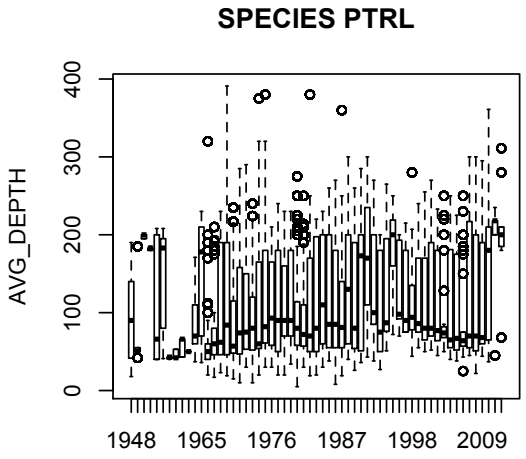

Figure 3. Page 2 of plotCleaned output.

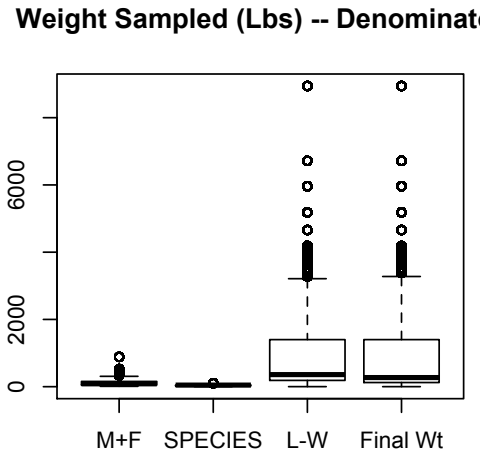

Figure 4. Weight_Sampled_Lbs plot from getExpansion_1.

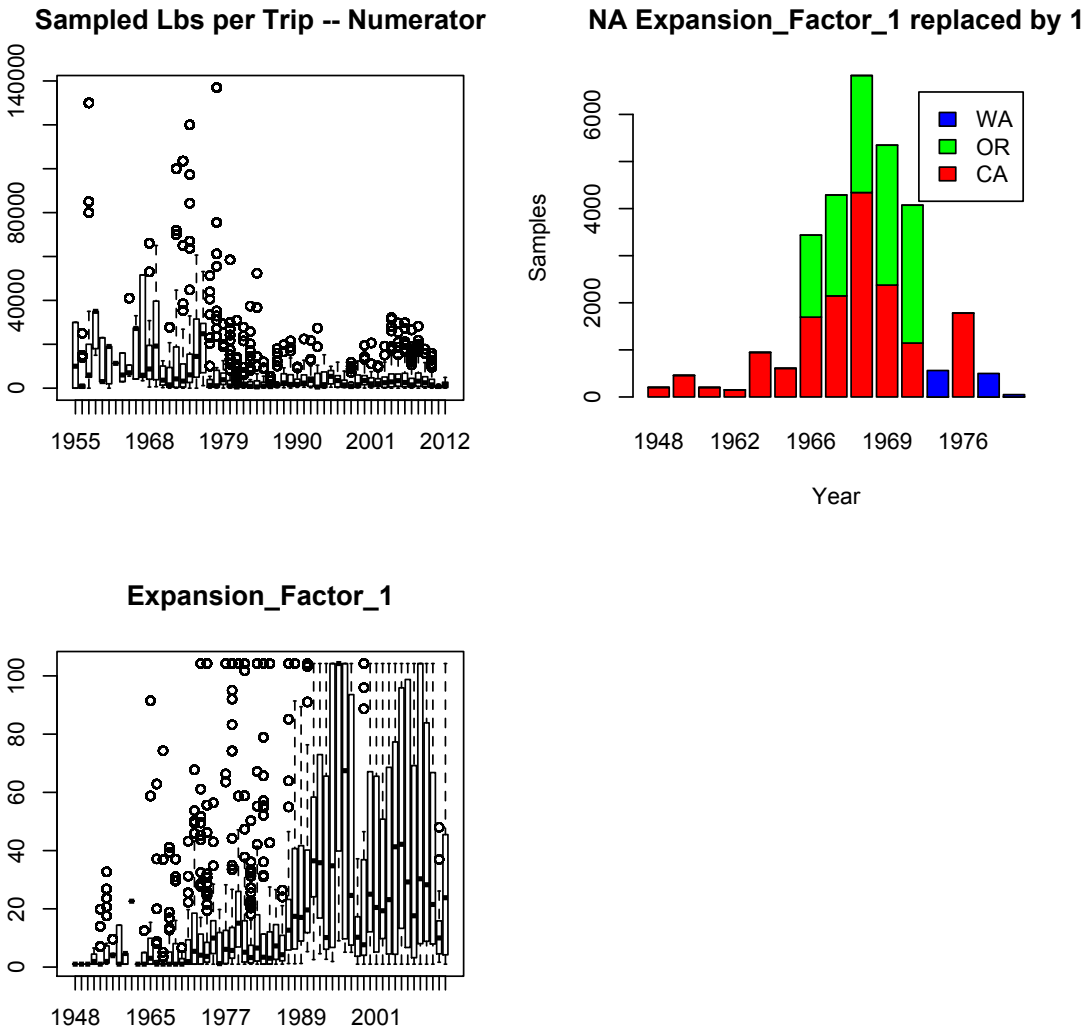

Figure 5. Expansion_factor_1 plots.

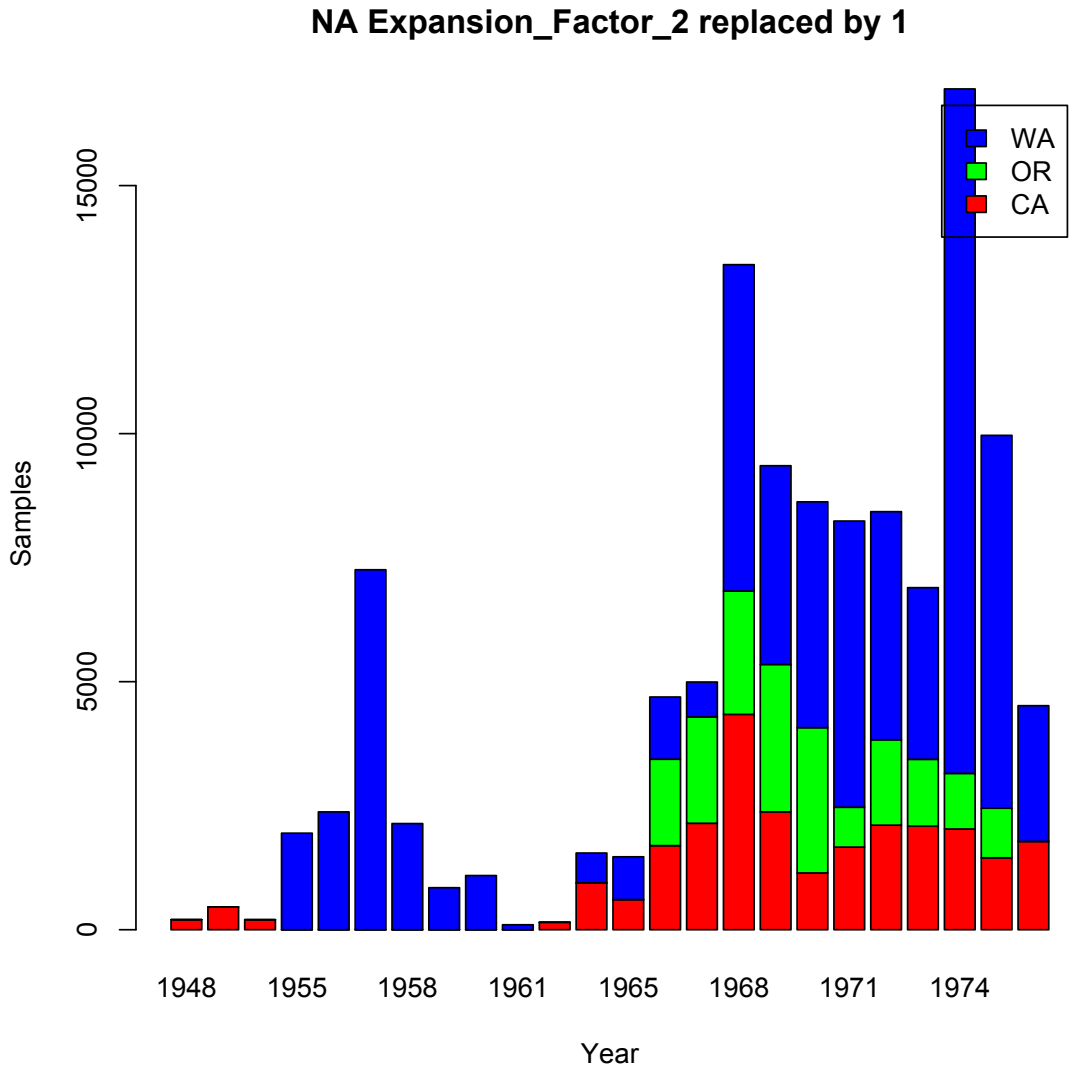

Figure 6. NAs in Expansion_Factor_2.

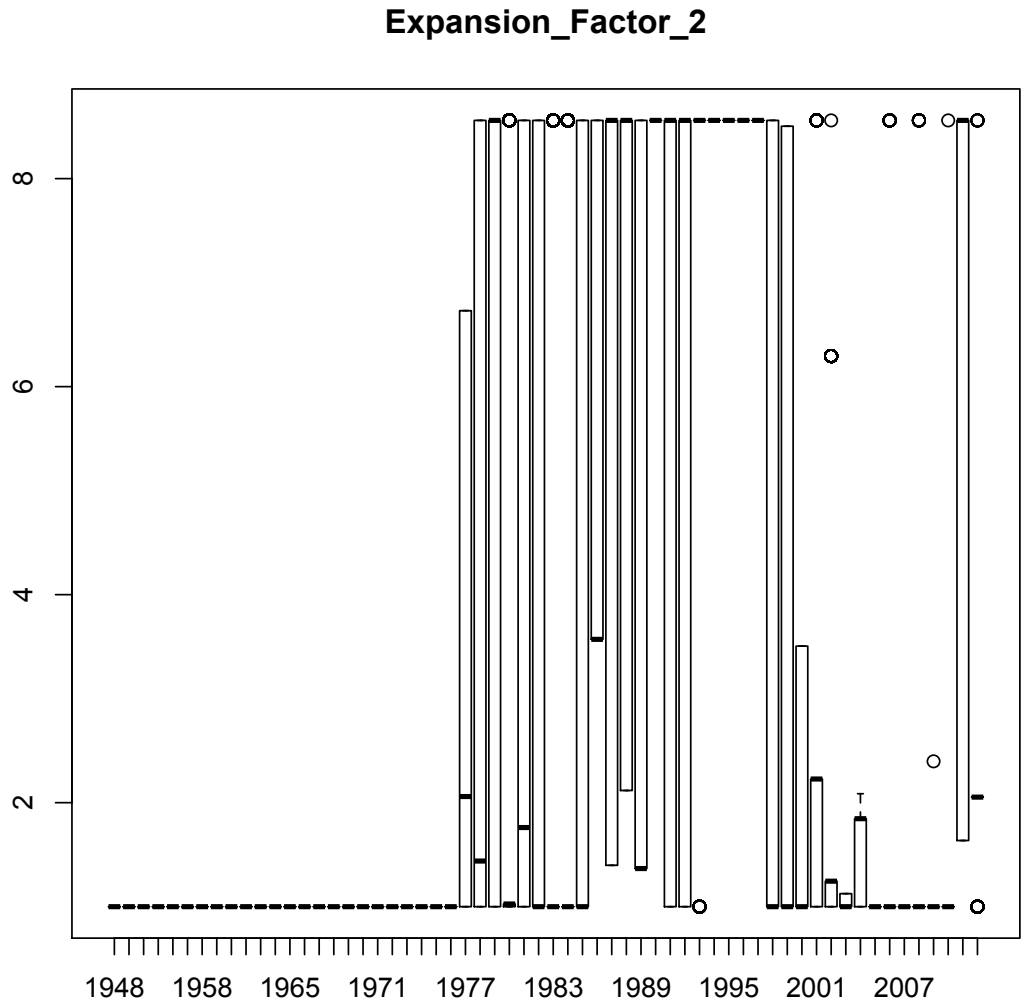

Figure 7. Epansion_Factor_2 after capping at .95 quantile (percentile) value.

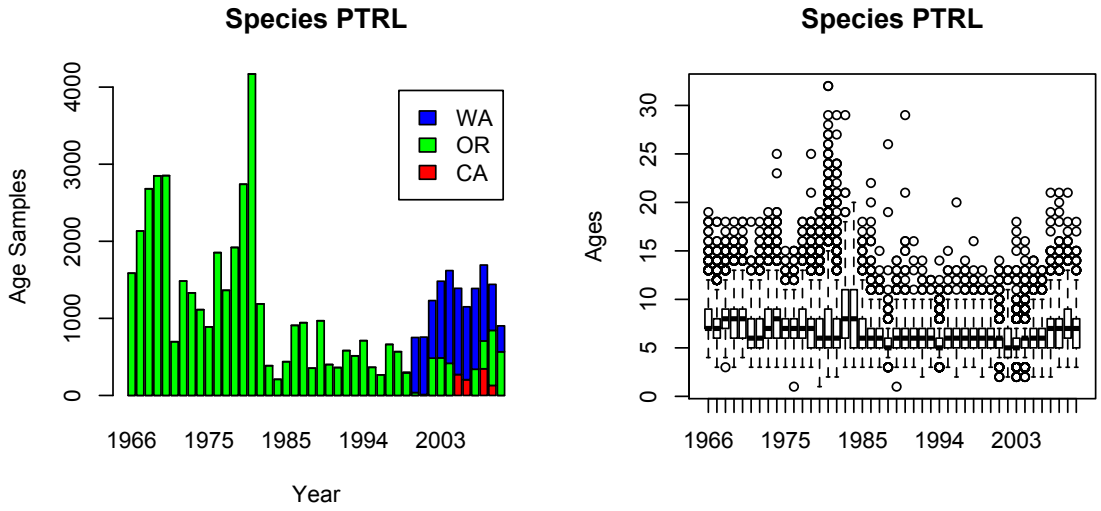

Figure 8. Age plots from plotCleanedAges.