SELECT package demonstration  
unpaired-haul relative selectivity

## Summary

Here we re-use paired haul data but without the paired haul structure.

With unpaired data, a data storage strategy is used whereby the data are put into an artificial paired format that creates a paired-haul format structure by pairing each haul with a ghost zero-catch haul with the other gear.

Consequently, the SELECT curve fitting functions can be used without knowing whether the data are paired or unpaired. The paired, or unpaired, nature of the data only becomes relevant when using the bootstrap and permutation functions, which now have a paired argument (default=TRUE).

**This case study demonstrates:**

* Reading data from separate gear-specific files and stacking them into a single data frame.
* Working with sub-sampled data.
* Converting unpaired data into the paired SELECT format using the SELECT.FORMAT function.
* Fitting catch share (relative selectivity) curves using splines via the SplineSELECT function.
* Using the bootstrap function bootSELECT to estimate the uncertainty in the catch share curve.
* Using the permutation function permSELECT to quantify the evidence for a length effect on the catch share.

### Data source

The data are for school prawn relative selectivity in penaeid trawls from the experiments conducted by Broadhurst et al., (2018, T45 side panels improve penaeid-trawl selection. Fisheries Research, 204: 8-15).

Here, any actual paired haul structure is ignored, and the hauls with the 32 m and 35 mm square side panels are treated as unpaired.

### Load required packages

require(tidyverse)  
require(mgcv)  
require(SELECT)  
require(readxl) #This package is installed with tidyverse  
nsim=999 #Number of bootstrap or permutation simulations. Should be >=999 in practice.

### Read in the data

NOTES: In the code below

* The Day variable is actually the unique twin-haul identifier, and so is renamed to Haul.
* CommonHauls is the set of twin-hauls that have the 32 mm and 35 mm gears. These are thus the same data as used in the paired-haul case study, but here are analysed as unpaired.

GearA.df=read\_excel("SchoolPrawnLenFreqs.xlsx", sheet = "32 square trawl")  
GearB.df=read\_excel("SchoolPrawnLenFreqs.xlsx", sheet = "35 square trawl")  
CommonHauls=intersect(unique(GearA.df$Day),unique(GearB.df$Day))  
  
GearA.df = GearA.df |>   
 rename(Haul=Day, n=No.school) |>   
 filter(Haul %in% CommonHauls) |>   
 mutate(Haul=paste0(Haul,".A"),q=1/Sf.school,Gear="A") |>   
 select(-Sf.school)  
GearB.df = GearB.df |>   
 rename(Haul=Day, n=No.school) |>   
 filter(Haul %in% CommonHauls) |>   
 mutate(Haul=paste0(Haul,".B"),q=1/Sf.school,Gear="B") |>  
 select(-Sf.school)

### Stack the separate dataframes

Note the conversion from sub-sampling scaling factors to sampling fractions and removal of CLs that are outside of the range of measured data.

Df=rbind(GearA.df,GearB.df) #Stack the two dataframes  
Df = Df |> filter(CL>=5 & CL<=25)

### Put into SELECT format

The SELECT.FORMAT function converts the unpaired data into a paired format. Note that it includes a variable (column) indicating the actual gear for the haul (since the other element of the pair is a ghost zero-catch haul).

Gears.df=SELECT.FORMAT(Df,by=c("Haul","CL"),gear="Gear",freq="n",  
 q.name="q",paired=F)  
head(Gears.df)

## # A tibble: 6 × 7  
## Haul CL nA nB qA qB Gear   
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>  
## 1 D1H2.A 5 0 0 0.0839 0 A   
## 2 D1H2.A 6 0 0 0.0839 0 A   
## 3 D1H2.A 7 0 0 0.0839 0 A   
## 4 D1H2.A 8 1 0 0.0839 0 A   
## 5 D1H2.A 9 0 0 0.0839 0 A   
## 6 D1H2.A 10 1 0 0.0839 0 A

### Define variable names

names(Gears.df)

## [1] "Haul" "CL" "nA" "nB" "qA" "qB" "Gear"

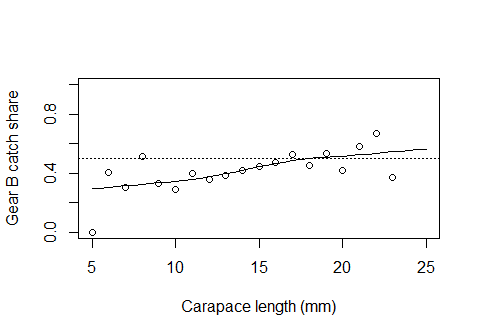
vNames=c("CL","nA","nB")  
qNames=c("qA","qB")

### Define a prediction function to be used with the bootstrap

This fit used the SplineSELECT defaults, but more generally one may want to try other values of k, say 5 and 10.

NOTE: The code below is identical to the unpaired case.

#Define the bootstrap prediction function  
CLseq=seq(5,25,0.5) #Carapace lengths to use for predn  
PrednFnc=function(data,var.names,q.names) {  
 SplineFit=SplineSELECT(data,var.names,q.names) #Defaults  
 predict(SplineFit,newdata=data.frame(CL=CLseq),type="response") }  
#Check that it works  
predn=PrednFnc(Gears.df,vNames,qNames)  
  
#Plot predictions against observed proportions  
Tots.df=Raw2Tots(Gears.df,vNames,qNames) |>   
 transform(lgth=CL, y=nB/(nA+nB))  
plot(y~CL,data=Tots.df,ylim=c(0,1),xlab="Carapace length (mm)",  
 ylab="Gear B catch share")  
points(CLseq,predn,type="l")  
abline(h=0.5,lty=3)



### Do the bootstrap of the catch share curve

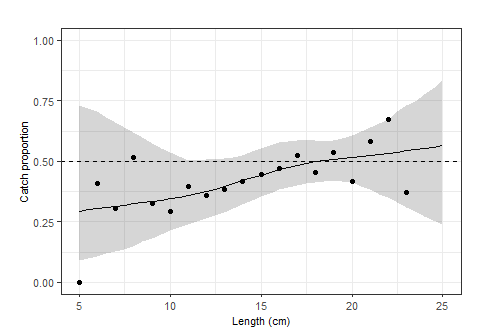
Note that it is necessary to provide the gear argument to SELECT when analyzing unpaired data.

BootPreds=bootSELECT(Gears.df,vNames,qNames,PrednFnc,haul="Haul",nsim=nsim,  
 paired=F,gear="Gear",verbose=F)

##   
## Bootstrap successfully completed

BootPlot(BootPreds,CLseq,predn,Data=Tots.df) +  
 geom\_hline(yintercept=0.5,linetype="dashed")

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



## Permutation test(s)

### Define a function to return permutation statistic(s)

For school prawns the commercial minimum landed carapace length is MLS=15 mm. In this example MLS=15 is passed to the SplineStatistics function so that it will also calculate the ratio of the proportion of commercial sized prawns in gear B versus gear A.

StatsFnc=function(data,var.names,q.names) {  
 SplineFit=SplineSELECT(data,var.names,q.names) #Defaults  
 SplineStatistics(SplineFit,MLS=15) }  
#Check that it works  
ObsStats=StatsFnc(Gears.df,vNames,qNames)  
ObsStats

## DevExpl DevExpl0.5 null null0.5 full model   
## 0.7180213 0.8881293 -83.0303441 -140.1892519 -45.4400569 -56.0397163   
## LRT LRT0.5 PropnRatio avg   
## 53.9812556 168.2990712 1.1949629 0.4301829

### Do permutations

PermStats=permSELECT(Gears.df,vNames,qNames,StatsFnc,haul="Haul",nsim=nsim,  
 paired=F,gear="Gear",verbose=F) #Use verbose=T to see progress

##   
## Permutations successfully completed

colnames(PermStats)=names(ObsStats) #To add column names to PermStats

The permPval function is used to calculate the permutation p-values

Stat="LRT" #Likelihood ratio test for a length effect  
cat("The observed",Stat,"is",ObsStats[Stat],"\n")

## The observed LRT is 53.98126

pval=permPval(ObsStats[Stat],PermStats[,Stat])  
cat("The permutational p-value for a length effect is",pval,"\n")

## The permutational p-value for a length effect is 0.087

Stat="LRT0.5" #LRT for equivalence, i.e., catch comparison=0.5 for all lengths  
cat("The observed",Stat,"is",ObsStats[Stat],"\n")

## The observed LRT0.5 is 168.2991

pval=permPval(ObsStats[Stat],PermStats[,Stat])  
cat("The permutational p-value for equivalence is",pval,"\n")

## The permutational p-value for equivalence is 0.124

Stat="PropnRatio" #Proportion of large fish in gear B compared to in gear A  
cat("The observed",Stat,"is",ObsStats[Stat],"\n")

## The observed PropnRatio is 1.194963

pval=permPval(ObsStats[Stat],PermStats[,Stat])  
cat("The permutational p-value for equal propns of large fish is",pval,"\n")

## The permutational p-value for equal propns of large fish is 0.028