SELECT package demonstration  
Paired-haul relative selectivity

## Summary

**This case study demonstrates:**

* Reading data from separate gear-specific files and merging them pairwise into a single data frame.
* Working with sub-sampled data.
* Fitting catch share (relative selectivity) curves using splines via the SplineSELECT function.
* Fitting catch share curves using an improved version of the averaged polynomial approach via the PolySELECT function (see bottom part of the code).
* 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.
  + that the gears are not identical (i.e., catch comparison is not equal to 0.5 for all lengths).
  + that the proportion of commercial sized prawns differs between gears.

### 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). The dats used here are from the twin-trawl experiment that fished two trawls differing in square mesh sizes used for the side panel, 32 mm or 35 mm.

### Load required packages

require(tidyverse)  
require(mgcv)  
require(SELECT)  
require(readxl) #This package is installed with tidyverse  
nsim=999 #For bootstrapping and permutation tests, set to >=999 in practice

### Read in the data

The data for each mesh size are in individual xlsx files and require merging.

GearA.df=read\_excel("SchoolPrawnLenFreqs.xlsx", sheet = "32 square trawl")  
GearB.df=read\_excel("SchoolPrawnLenFreqs.xlsx", sheet = "35 square trawl")  
#The "Day" variable is actually the unique haul-pair identifier  
GearA.df = GearA.df |> rename(Haul=Day, nA=No.school, sfA=Sf.school)  
GearB.df = GearB.df |> rename(Haul=Day, nB=No.school, sfB=Sf.school)

### Merge the separate dataframes to create a dataframe for the twin hauls

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

#Merge the gears with the same identical haul-pair ID (i.e., twin tows)  
Pairs.df = inner\_join(GearA.df,GearB.df,by=c("Haul","CL"))  
#Convert scaling factors to sampling fractions  
Pairs.df = Pairs.df |>  
 transform(qA=1/sfA,qB=1/sfB) |> filter(CL>=5 & CL<=25)

### Define variable names

names(Pairs.df)

## [1] "Haul" "CL" "nA" "sfA" "nB" "sfB" "qA" "qB"

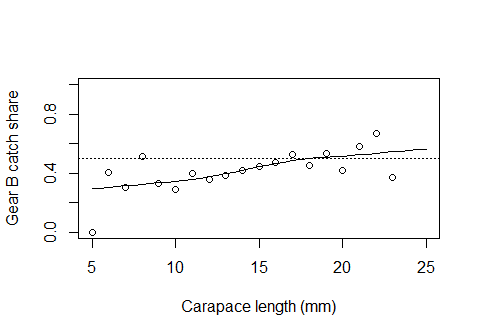
vNames=c("CL","nA","nB")  
qNames=c("qA","qB")

## Bootstrap

### 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.

#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(Pairs.df,vNames,qNames)  
  
#Plot predictions against observed proportions  
Tots.df=Raw2Tots(Pairs.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)



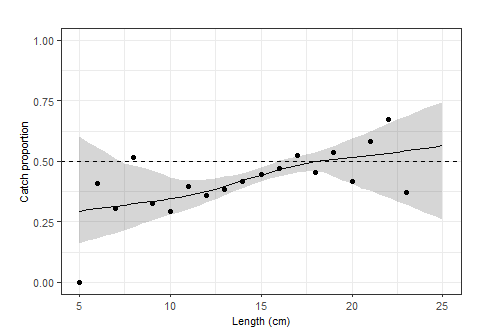
### Do the bootstrap of the catch share curve

BootPreds=bootSELECT(Pairs.df,vNames,qNames,PrednFnc,haul="Haul",nsim=nsim,  
 paired=T,verbose=F) #Use verbose=T to see progress

##   
## 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(Pairs.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(Pairs.df,vNames,qNames,StatsFnc,haul="Haul",nsim=nsim,  
 paired=T,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 0.003

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 0.003

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 0.001

## Demonstration of the averaged-polynomial catch curve

**This is included for completeness - we recommend using the spline approach instead of the averaged polynomial approach.**

**The default use of PolySELECT includes improvements to the Herrmann et al. (2017) implementation by reducing the tendency of the averaged polynomial to overfit. See PolySELECT documentation for details.**

### Averaged polynomial fits

require(MuMIn) #For MuMIn::dredge  
options(na.action = "na.fail") #To ensure dredge terminates if there are NAs  
#The default averaged-polynomial fit  
PolyFit=PolySELECT(Pairs.df,vNames,qNames)   
#The averaged-polynomial fit used by Herrmann et al. (2017)  
HerrmannFit=PolySELECT(Pairs.df,vNames,qNames,q.ODadjust=F,quasi=F,All=T)

PolyPredn=predict(PolyFit$avg.fit,newdata=data.frame(CL=CLseq),  
 type="response")  
HerrmannPredn=predict(HerrmannFit$avg.fit,newdata=data.frame(CL=CLseq),  
 type="response")  
plot(y~CL,data=Tots.df,ylim=c(0,1),xlab="Carapace length (mm)",  
 ylab="Gear B catch share")  
points(CLseq,predn,type="l",col="blue") #Spline  
points(CLseq,PolyPredn,type="l",col="blue",lty=2) #PolySELECT  
points(CLseq,HerrmannPredn,type="l",col="red",lty=2) #Herrmann et al. (2017)  
legend("topleft",  
 legend=c("Spline","PolySELECT default","Herrmann et al. (2017)"),  
 col=c("blue","blue","red"),lty=c(1,2,2),bty="n")  
abline(h=0.5,lty=3)

