Simulate catch comparison data with sampling fractions

Mollie Brooks

12 Aug 2020

This is a simulation study based on Holst & Revill 2009. We assume that two types of gear have different selection patterns and simulate catch comparison data from those selection patterns. Then we fit a model with an offset to account for the sampling fractions or raising factors. For simplicity, we assume that psplit is constant across all pairs of hauls.

```
library(boot) #for inv.logit
library(ggplot2); theme_set(theme_bw())
library(plyr)
library(selfisher)
set.seed(11)
```

Simulate data

Set up parameters

```
length=1:100 #measured length classes
pars1=c(-6, .1) #selection curve parameters for gear 1
pars2=c(-11, .3) #selection curve parameters for gear 2
nhaul=10 #number of hauls with each type of gear
q1=runif(nhaul, 0.6, 0.8) #sampling fractions for gear 1
q2=runif(nhaul, 0.2, 0.4) #sampling fractions for gear 2
psplit=0.5 # psplit will be obsorbed by the intercept
nfish=c(rep(0, 20),
                rep(5, 20),
                rep(10, 20),
                rep(20, 20),
                rep(10, 10),
                rep(1, 10)) # avg number of fish encountered in each length class in each haul
```

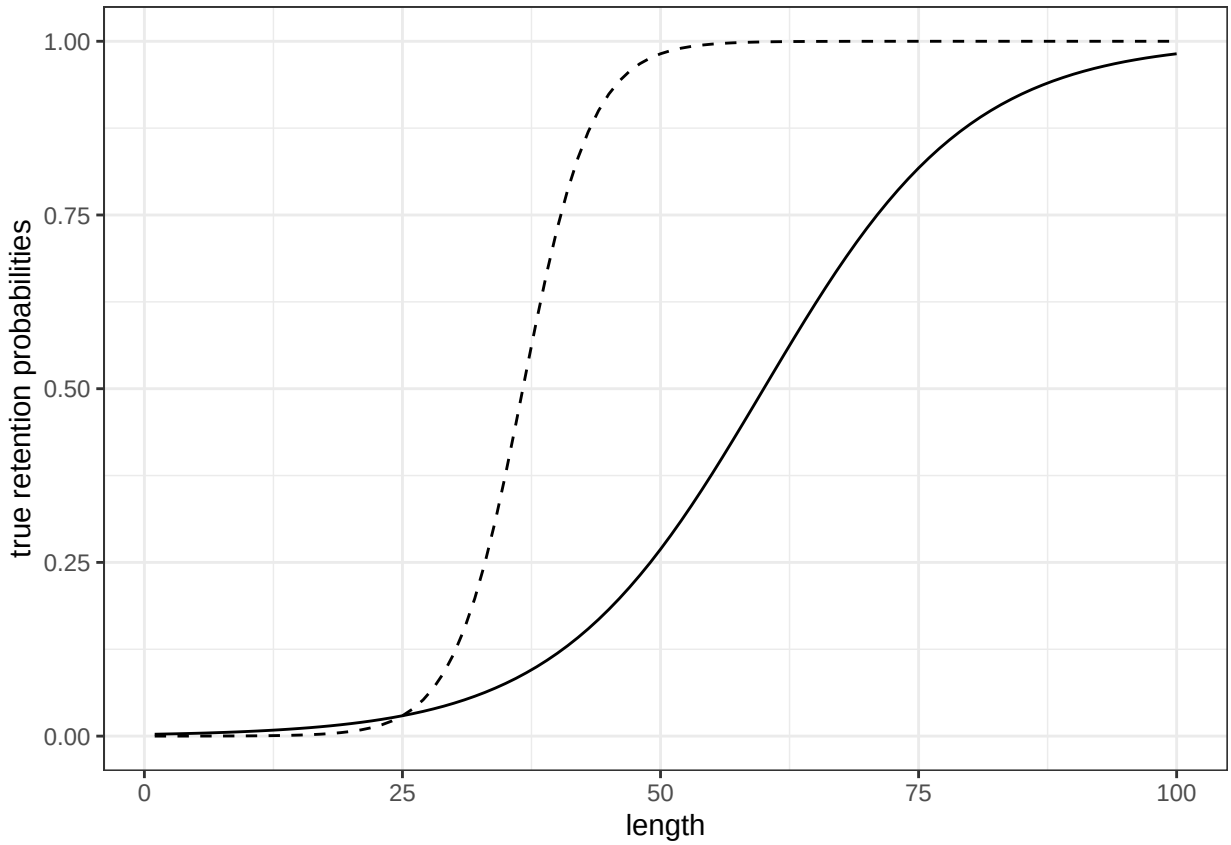
Organize the model parameters. Simulate the total number of fish of each length class by haul that are encountered (but not necessarily caught).

```
bdat=expand.grid(length = length, haul=1:nhaul)
bdat$nfish=nfish

bdat=transform(bdat,
            sampling_test1 =q1[haul],
            sampling_test2 =q2[haul],
            r1=inv.logit(pars1[1]+ pars1[2]* length),
```

```
            r2=inv.logit(pars2[1]+ pars2[2]* length),
            simtotal=rpois(nrow(bdat), nfish)
            )
```

Plot the true selection curves for the two gear types.

```
ggplot(bdat)+
    geom_line(aes(length, r1))+
    geom_line(aes(length, r2), lty=2)+
    ylab("true retention probabilities")
```



Simulate retained and sampled fish in each gear type.

```
dat=transform(bdat, length= length, haul=haul,
    test1=rbinom(nrow(bdat), size= simtotal, prob=r1*q1*psplit),
    test2=rbinom(nrow(bdat), size= simtotal, prob=r2*q2*(1-psplit)))
```

Organize data

Here we do a few transformaitons:

- scale the length classes to avoid overflow in the polynomial
- calculate the ratio of the sampling fractions to use as an offset or as qratio
- calculate the binomial response (proportion and total) for comparing test 1 vs test2

- calculate the true ratio of the retention probabilities for later
- drop rows of data with no observed fish

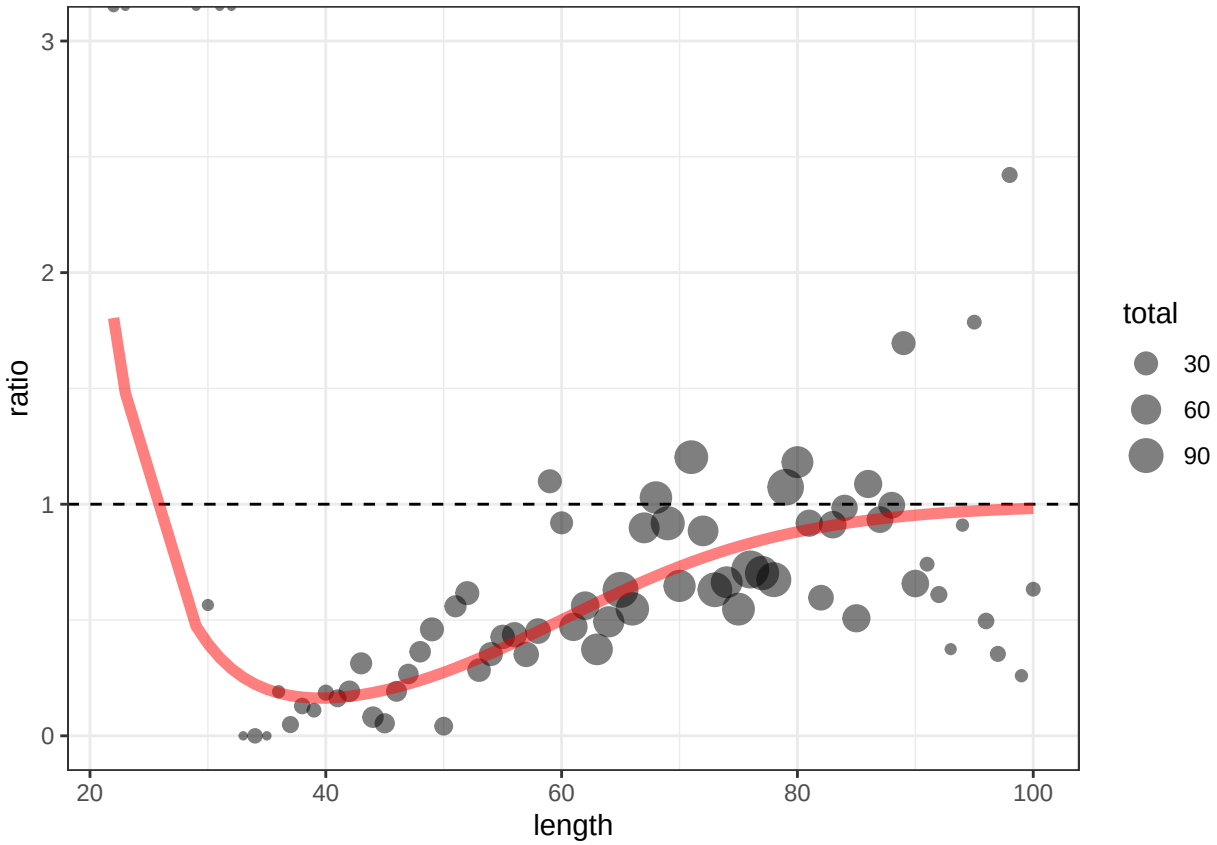
```
mu=mean(rep(dat$length, dat$test1+dat$test2))
var=var(rep(dat$length, dat$test1+dat$test2))

dat=transform(dat,
      sl=(length-mu)/sqrt(var),
      q_ratio=sampling_test1/sampling_test2,
      prop=test1/(test1+test2),
      total=test1+test2,
      true_raised_ratio=r1/r2)

dat=subset(dat, !is.na(prop))
```

Aggregate the data across hauls for plotting.

```
sumdat=ddply(dat, ~length + sl + true_raised_ratio, summarize,
      prop=sum(test1)/sum(total),
      total=sum(total),
      raised_prop=sum(test1/sampling_test1)/
                        (sum(test1/sampling_test1)+sum(test2/sampling_test2)),
      raised_total=sum(test1/sampling_test1)+sum(test2/sampling_test2),
      raised_ratio = sum(test1/sampling_test1)/sum(test2/sampling_test2))
```

Plot aggregated data.

```
p1=ggplot(sumdat, aes(x=length,))+
  geom_point(aes(y= raised_ratio, size=total), alpha=.5)+
  geom_hline(yintercept = 1, lty=2)+
  geom_line(data=sumdat, aes(length, true_raised_ratio), colour="red", alpha=.5, lwd=2)+
  ylab("ratio")+
  coord_cartesian(ylim=c(0, 3))
p1
```



Model

Fit

Fit the model.

```
m1=selfisher(prop~offset(log(q_ratio))+sl+I(sl^2)+I(sl^3)+I(sl^4), total=total, dat, haul=haul)
```

Predict

Calculate predictions from the fitted model.

```
newdata=data.frame(length=unique(dat$length))
newdata=transform(newdata,
    sl=(length-mu)/sqrt(var),
    total=1,
    haul=NA,
    q_ratio=1
)

newdata$est_raised_ratio=predict(m1, newdata=newdata, type="ratio")
```

Bootstrap confidence intervals

Code for Mac and Linux

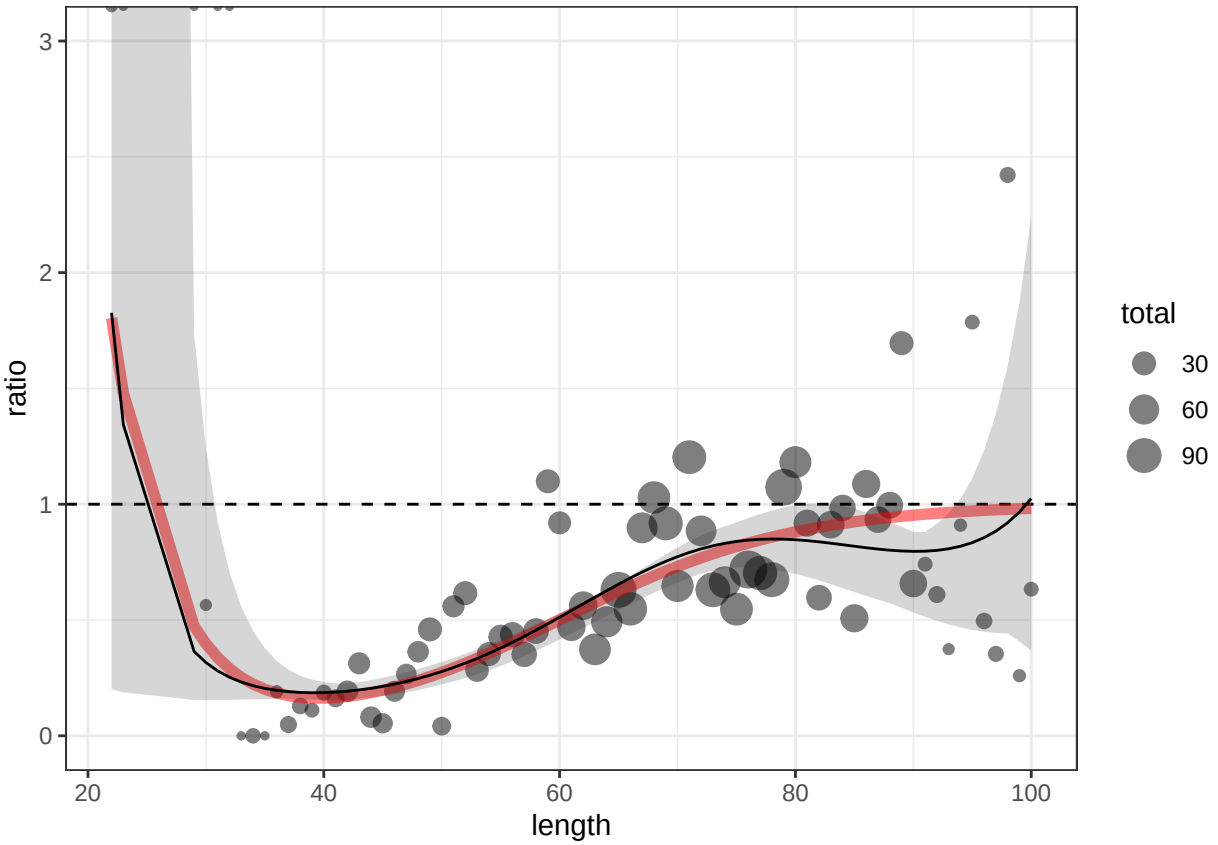
```
bs=bootSel(m1, nsim=10, parallel = "multicore", ncpus = 4,
        FUN=function(mod){predict(mod, newdata=newdata, type="ratio")})
```

Code for Windows

```
library(snow)
ncpus = 4
cl = makeCluster(rep("localhost", ncpus)) clusterExport(cl, "newdata")
bs = bootSel(m1, nsim=10, parallel = "snow", cl=cl,
    FUN=function(mod){predict(mod, newdata=newdata, type="ratio")})
stopCluster(cl)
```

Organize and plot bootstraps

```
#apply quantile function to bootstraps and match them with the newdata used for predictions
quants=apply(bs$t, 2, quantile, c(0.025, 0.5, 0.975))

newdata[,c("lo", "mid", "hi")]=t(quants)

p1+
    geom_ribbon(data=newdata, aes(ymin=lo, ymax=hi), alpha=0.2)+
    geom_line(data=newdata, aes(y=est_raised_ratio))
```



In the plot above, the black line is the model fit, the red line is the true pattern that we simulated from, the grey dots are the observations, and the grey ribbon is the bootstrapped CI.

Or we could use a spline instead of a polynomial

```
library(splines)

m1_sp=selfisher(prop~offset(log(q_ratio))+bs(sl, df=4), total=total, dat, haul=haul)
```

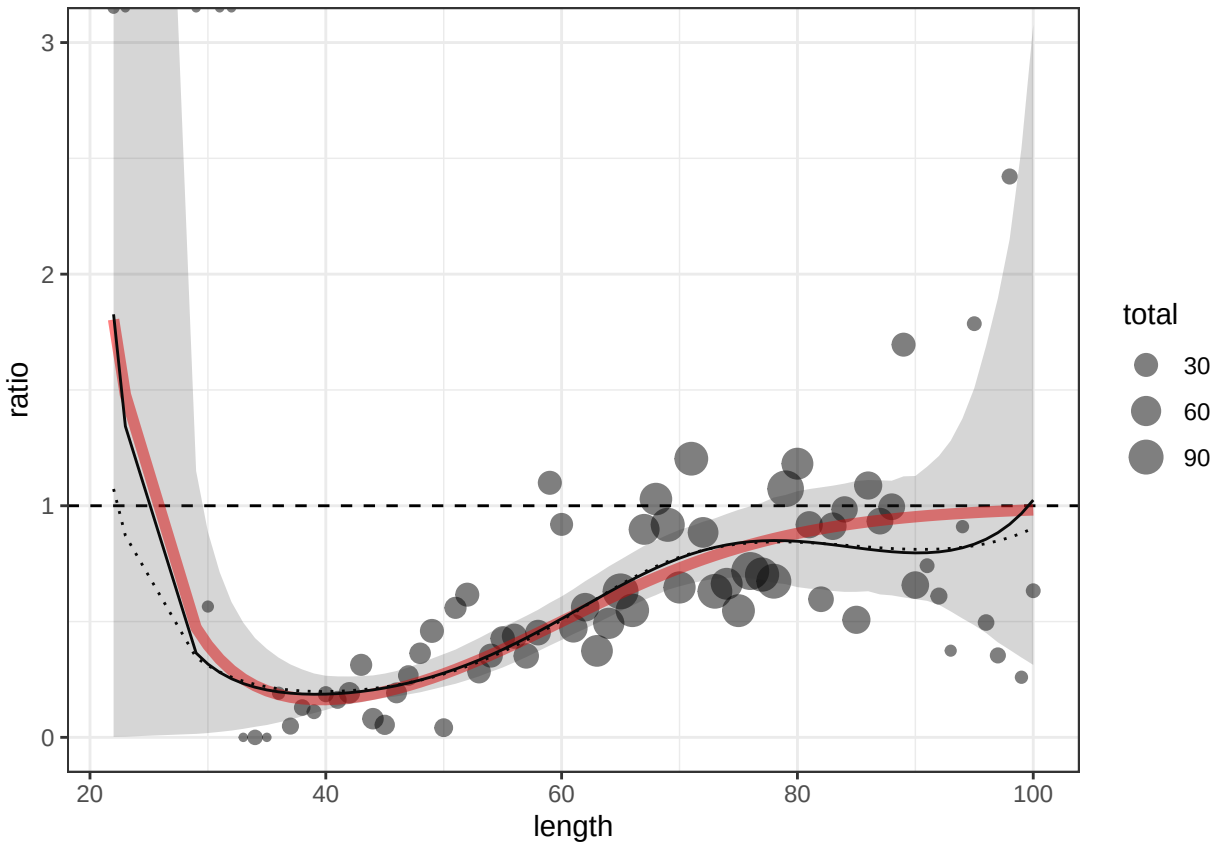
Calculate predictions from the fitted model.

```
newdata$raised_ratio_sp=predict(m1_sp, newdata=newdata, type="ratio")
```

Bootstrap confidence intervals

```
bs=bootSel(m1_sp, nsim=1000, parallel = "multicore", ncpus = 4,
        FUN=function(mod){predict(mod, newdata=newdata, type="ratio")})

quants=apply(bs$t, 2, quantile, c(0.025, 0.5, 0.975))

newdata[,c("lo", "mid", "hi")]=t(quants)

p1+
```

```
  geom_line(data=newdata, aes(length, raised_ratio_sp), lty=3)+
  geom_line(data=newdata, aes(y=est_raised_ratio))+
  geom_ribbon(data=newdata, aes(ymin=lo, ymax=hi), alpha=0.2)
```



In the plot above, the black line is the model fit with a polynomial, the dotted line is the model fit with a spline of length, the red line is the true pattern that we simulated from, the grey dots are the observations, and the grey ribbon is the bootstrapped CI. The spline and polynomial give pretty much the same answer. Maybe the AIC are different?

```
library(bbmle)
```

```
## Loading required package: stats4
```

```
AICtab(m1, m1_sp)
```

```
##       dAIC df
## m1    0.0  5
## m1_sp 1.5  5
```

They're basically equivalent, but the polynomial is slightly better.