Simulate paired gear data with sampling fractions

Mollie Brooks

12 Aug 2020

This is a simulation study to confirm that the model can recover the parameters. We assume that sampling fraction varies by haul and gear. To keep it simple, we assume that relative fishing power (psplit) is constant across hauls, but in reality it usually varies randomly by haul.

```
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 # length classes
pars1=c(-6, .1) #selection curve parameters for gear 1
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=0.6
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) #stacked length classes and hauls
bdat$nfish=nfish

bdat=transform(bdat,
            sampling_test =q1[haul], # sampling fraction varies by haul and gear
            sampling_control =q2[haul], # sampling fraction varies by haul and gear
            psplit=psplit, #relative fishing power is constant by haul
            r1=inv.logit(pars1[1]+ pars1[2]* length), #retention of test gear
            r2=1, #retention of control gear
            simtotal=rpois(nrow(bdat), nfish) #total number caught but not observed
            )
```

Simulate retained and sampled fish in each gear type.

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

Organize data

Here we do a few transformations:

- calculate the ratio of the sampling fractions
- calculate the binomial response (proportion and total)
- keep track of true parameters (`r1`, `psplit`) to compare with estiamates
- drop rows of data with no observed fish

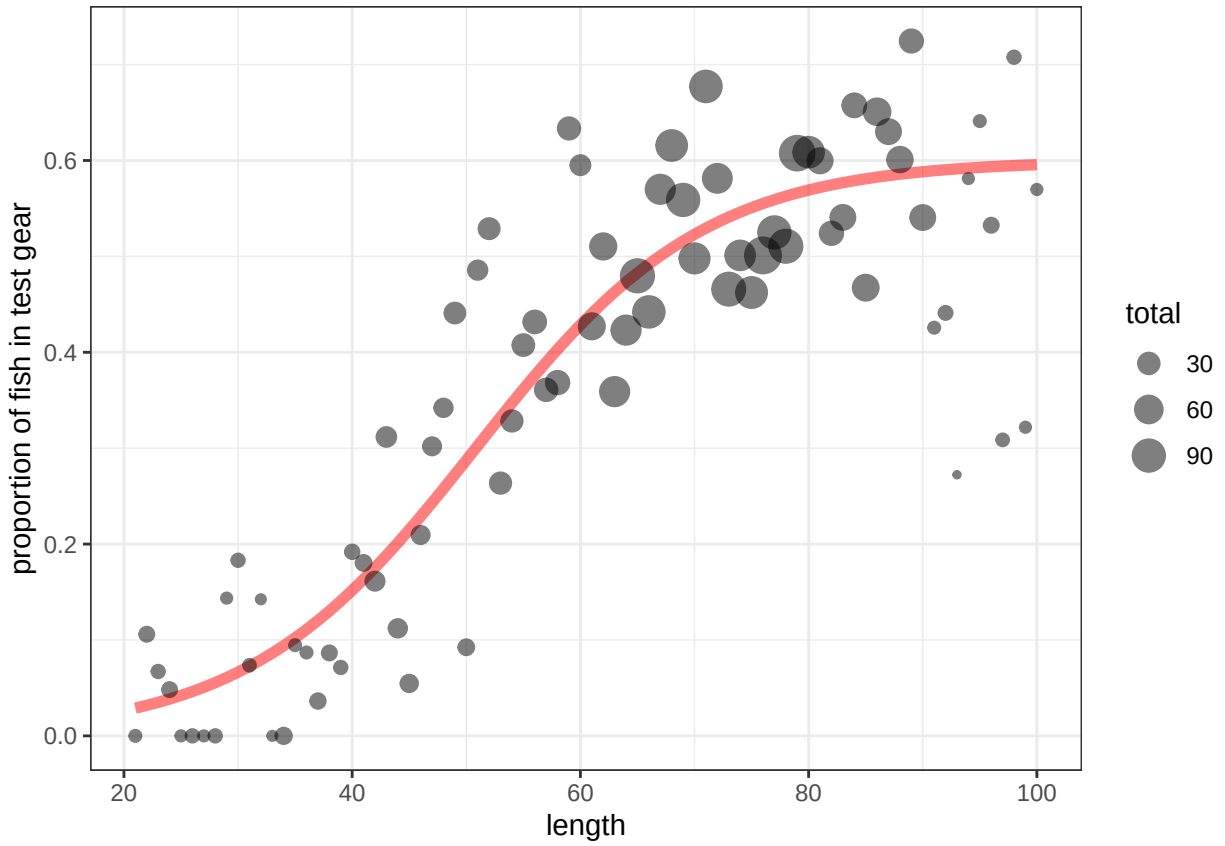
```
dat=transform(dat,
      qratio=sampling_test/sampling_control,
      prop=test/(test+control),
      total=test+control,
      r1=r1,
      true_resp=r1*psplit/ (1-psplit + psplit*r1)
)

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

Aggregate the data across hauls for plotting. Plot true selectivity pattern with observations.

```
sumdat=ddply(dat, ~length + r1 , summarize,
      prop=sum(test)/sum(total),
      total=sum(total),
      raised_prop=sum(test/sampling_test)/
                        (sum(test/sampling_test)+sum(control/sampling_control)),
      raised_total=sum(test/sampling_test)+sum(control/sampling_control),
      true_resp=true_resp[1]
      )

p1=ggplot(sumdat, aes(x=length))+
    geom_line(data=sumdat, aes(length, true_resp), colour="red", alpha=.5, lwd=2)+
      geom_point(data=sumdat, aes(size=total, y=raised_prop), alpha=.5)+
    ylab("proportion of fish in test gear")
p1
```



Model

Fit the model and check if estimates are close to true values.

```
m1=selfisher(prop~ length, pformula=~1, psplit=TRUE, total=total, dat, haul=haul, qratio=qratio)

summary(m1)
```

```
##  Family: binomial  ( logit )
## Selectivity formula:          prop ~ length
## Relative fishing power formula:        ~1
## Data: dat
## Total: total
##
##           AIC           BIC        logLik      deviance Pearson.ChiSq
##        1384.5        1397.8        -689.2         641.1         574.8
##      df.resid
##           613
##
##
## Selectivity model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.48613    0.59877 -10.832  < 2e-16 ***
## length       0.11409    0.01401   8.142 3.89e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Relative fishing power model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.32701    0.08728   3.747 0.000179 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Size at retention probability:
##      p   Lp.Est Lp.Std.Err
## 1 0.25 47.22348   1.681468
## 2 0.50 56.85320   2.458740
## 3 0.75 66.48293   3.472919
##
## Selectivity range (SR):
##   Estimate Std. Error
##  19.259452   2.365461
```

```
fixef(m1)$r
```

```
## (Intercept)      length
##  -6.4861270   0.1140855
```

```
pars1
```

```
## [1] -6.0  0.1
```

```
psplit
```

```
## [1] 0.6
```

```
boot::inv.logit(fixef(m1)$p) #very close
```

```
## (Intercept)
##   0.5810317
```

Calculate predictions from the fitted model.

```
newdata=data.frame(length=unique(dat$length))
newdata=transform(newdata,
    total=1,
    haul=NA,
    qratio=1
)

newdata$est_resp=predict(m1, newdata=newdata, type="response")
```

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="response")})
```

Code for Windows

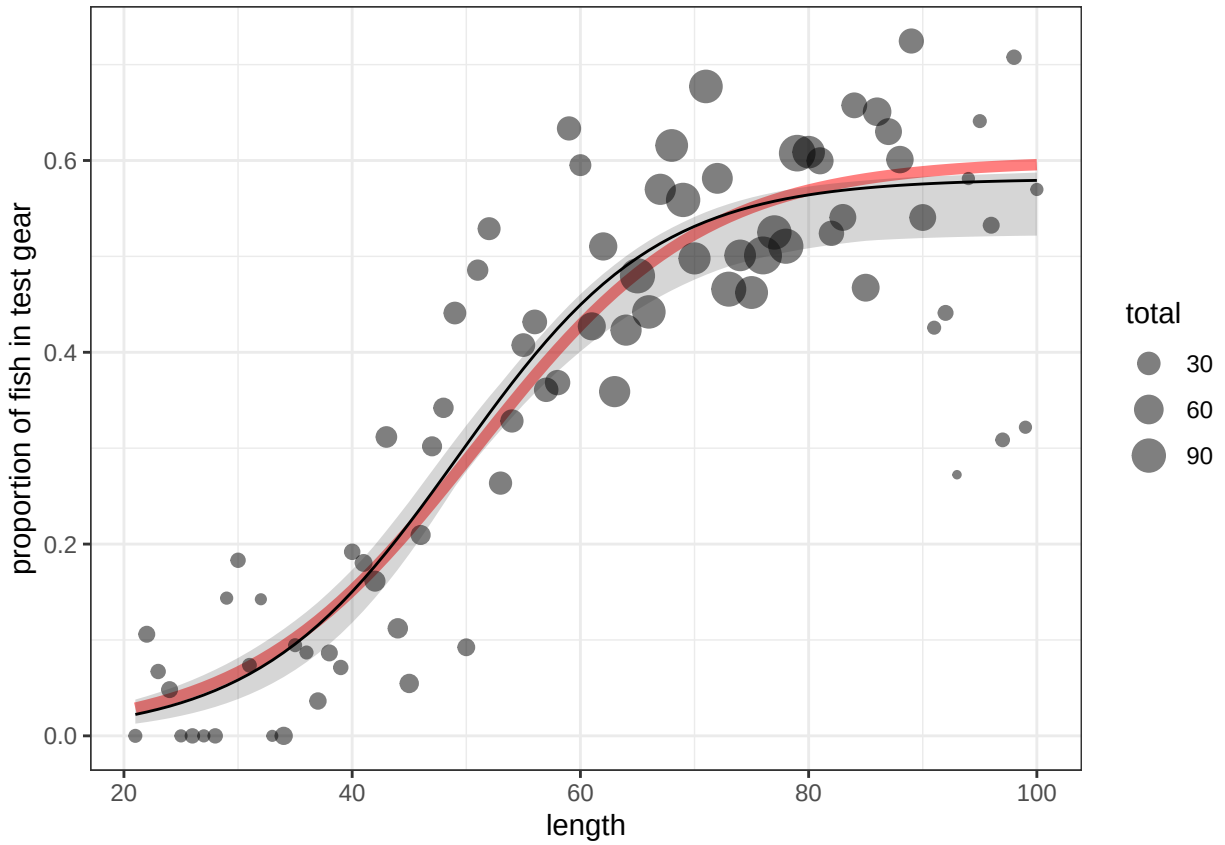
```
library(snow)
ncpus = 4
cl = makeCluster(rep("localhost", ncpus)) clusterExport(cl, "newdata")
bs = bootSel(m1, nsim=10, parallel = "snow", cl=cl,
    FUN=function(mod){predict(mod, newdata=newdata, type="response")})
stopCluster(cl)
```

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_resp))
```



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.