Introducing selfisher: open source software for statistical analyses of fishing gear selectivity

Brooks, Mollie Elizabeth; Melli, Valentina; Savina, Esther Anne Charlotte Marie; Santos, Juan; Millar, Russell B.; O'Neill, Finbarr G; Veiga-Malta, Tiago; Krag, Ludvig Ahm; Feekings, Jordan P.

Published in:
Canadian Journal of Fisheries and Aquatic Sciences

Link to article, DOI:
10.1139/cjfas-2021-0099

Publication date:
2022

Document Version
Other version

Supplementary material A: Covered codend analyses of four codends catching haddock

15 Oct 2021

This example deals with data from an experiment published by O'Neill et al. (2016) that investigated the selectivity of haddock (*Melanogrammus aeglefinus*) in four codends made from netting materials with different twine bending stiffnesses and mesh sizes. Three of the codends had a nominal mesh size of 120mm and one a nominal mesh size of 130mm. The twine bending stiffness values were in the range 0.64 to 1.1kN mm^2. We label the codends as 120low, 120med, 120high and 130med to reflect their mesh size and bending stiffness (as categorised by the netting manufacturers). As in the original analysis, we show that selection is dependent on both of these parameters and the total codend catch weight.

Preliminaries

```
library(selfisher)
library(plyr) #for aggregating data
library(ggplot2); theme_set(theme_bw())
library(parallel) #for bootstrapping in parallel
library(bbmle) #for AICtab BICtab
library(splines)
library(reshape)
```

Data structure

We load the data each row of which corresponds to the fish of a given length from a given haul. The length (cm), haul number, mesh size (mm), twine bending stiffness ($kNmm^2$), number of fish measured from the codend, codend raising factor, number of fish measured from the cover, cover raising factor, catch size (kg) and codend label are specified respectively.

```
data("coverhaddock")
head(coverhaddock)
```

```
##   Length haul  mesh stiffness codend cod_rf cover cov_rf catch   gear
## 1   10.5   36 119.3      0.69      0      1     0   1.00   617 120low
## 2   11.5   36 119.3      0.69      0      1     0   1.00   617 120low
## 3   12.5   36 119.3      0.69      0      1     0   1.00   617 120low
## 4   13.5   36 119.3      0.69      0      1     1   4.78   617 120low
## 5   14.5   36 119.3      0.69      0      1     4   4.78   617 120low
## 6   15.5   36 119.3      0.69      0      1    12   4.78   617 120low
```

Here we can see that the raising factor varied by length class, which is not a problem in `selfisher`.

```
summary(coverhaddock)
```

```
##      Length          haul           mesh         stiffness
##  Min.   :10.5   Min.   : 6.0   Min.   :119.3   Min.   :0.6400
##  1st Qu.:21.5   1st Qu.:13.0   1st Qu.:119.3   1st Qu.:0.6900
##  Median :33.0   Median :22.5   Median :119.6   Median :0.8000
##  Mean   :33.0   Mean   :22.0   Mean   :122.5   Mean   :0.8087
##  3rd Qu.:44.5   3rd Qu.:31.0   3rd Qu.:129.4   3rd Qu.:0.8000
##  Max.   :55.5   Max.   :39.0   Max.   :129.4   Max.   :1.1000
##      codend           cod_rf          cover            cov_rf
##  Min.   :  0.00   Min.   :1.000   Min.   :  0.00   Min.   : 1.000
##  1st Qu.:  0.00   1st Qu.:1.000   1st Qu.:  0.00   1st Qu.: 1.000
##  Median :  0.00   Median :1.000   Median :  3.00   Median : 2.580
##  Mean   : 11.89   Mean   :1.349   Mean   : 24.82   Mean   : 3.437
##  3rd Qu.:  5.00   3rd Qu.:1.701   3rd Qu.: 38.00   3rd Qu.: 4.097
##  Max.   :156.00   Max.   :2.989   Max.   :272.00   Max.   :59.330
##      catch            gear
##  Min.   :267.0   120high:322
##  1st Qu.:482.0   120low :368
##  Median :535.5   120med :276
##  Mean   :556.7   130med :414
##  3rd Qu.:630.0
##  Max.   :979.0
```

We can also see that all the hauls are contained in one data frame. The data is organized into what is called "long format".

Transforming data

For a model in `selfisher`, we need to convert counts into proportions and totals. Unlike other GLM functions for binomial regression, it is **not** possible to specify the binomial variable as a two-column response variable, e.g. `cbind(N_test, N_cover)`.

Because not all fish in the samples were counted, we will account for this in the model (Millar 1994). The values we need in the model are calculated as `cov_rf/cod_rf`. If instead we had a sampling fraction, we would calculate `qratio = sampling_test/sampling_cover` because a sampling fraction is the inverse of a raising factor. We create a new column in the data with the value of `qratio = cov_rf/cod_rf` for each row. An easy way to compute this value by row is to use the `transform` function.

```
coverhaddock = transform(coverhaddock,
    total = codend + cover,
    prop = codend / (codend+cover),
    qratio = cov_rf / cod_rf)
```

We drop rows of data where no fish were observed because they don't contain any information (i.e. where total = 0). This doesn't affect the model except to allow for bootstrapping later.

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

Single gear model with covered codend

We will start out with a simple example of one gear type ("120low"). So we need to subset the data.

```
coverhaddock_120low = subset(coverhaddock, gear=="120low")
```

The following is a model for multiple haul data from a covered codend experiment with subsampling. It could be argued that there should be a random effect of haul in the models to account for variation among hauls and avoid pseudoreplication, but we will keep it simple for this first example and only include a fixed effect of length.

```
mod_base = selfisher(prop ~ Length, qratio=qratio, total=total, haul=haul, data=coverhaddock_120low)
```

Extracting residuals and other standard methods

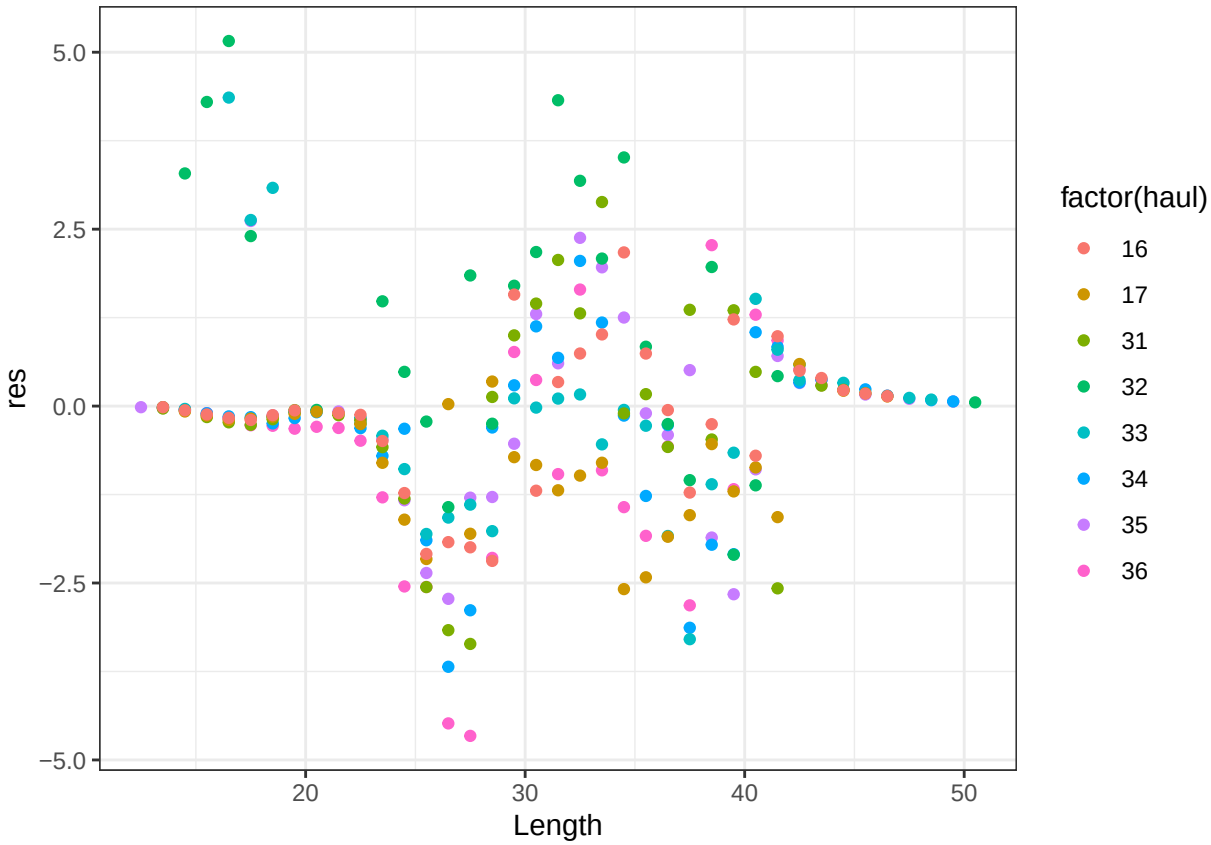
We can check the model residuals for patterns. There's a `residuals` function for `selfisher` models. All methods for `selfisher` models can be displayed as follows:

```
methods(class="selfisher")
```

```
##  [1] anova       confint     df.residual extractAIC  family      fitted
##  [7] fixef       getME       logLik      model.frame nobs        predict
## [13] print       ranef       residuals   simulate    summary     terms
## [19] VarCorr     vcov
## see '?methods' for accessing help and source code
```

Here is one way to plot the residuals.

```
coverhaddock_120low$res = residuals(mod_base, type="deviance")
ggplot(coverhaddock_120low, aes(Length, res, colour=factor(haul)))+geom_point()
```



Residuals from GLMs are notoriously opaque, but in the future, we will try to make `selfisher` compatible with the `DHARMa` package to make it easier to assess residuals (Hartig 2020).

Links other than `logit`

It is also possible to consider other link functions, or use a spline. See the function documentation (`?selfisher`) for a list of implemented link functions. Here we fit the logistic, probit and Richard's curve and a spline.

```
mod_probit = selfisher(prop ~ Length, qratio=qratio, total=total,
    link="probit", haul=haul, coverhaddock_120low)

mod_richards = selfisher(prop ~ Length , qratio=qratio, total=total,
    link="richards", haul=haul, coverhaddock_120low)
```

```
## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation

## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation
```

```
mod_spline = selfisher(prop ~ bs(Length, 3), qratio=qratio, total=total,
    haul=haul, coverhaddock_120low)
```

Fitting the model with `link="richards"` produced some warnings, but this is ok. The model is valid if the `summary` function is able to produce non-NA standard-errors as seen below.

```
summary(mod_richards)
```

```
##  Family: binomial  ( Richards )
## Selectivity formula:          prop ~ Length
## Data: coverhaddock_120low
## Total: total
##
##           AIC           BIC        logLik      deviance Pearson.ChiSq
##         953.2         963.9        -473.6         491.9       22746.8
##      df.resid
##           255
##
##
## Richards exponent parameter (delta): 0.542
##
## Selectivity model:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.65570    0.64953  -17.95   <2e-16 ***
## Length        0.35000    0.01549   22.60   <2e-16 ***
## ---
## 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 32.98060 0.05570708
## 2 0.50 35.54755 0.07331880
## 3 0.75 38.38777 0.15044341
##
## Selectivity range (SR):
##   Estimate Std. Error
##  5.4071721  0.1568771
```

Then we could find the model with the best fit, using information criteria (e.g. BIC).

```
BICtab(mod_base, mod_probit, mod_richards, mod_spline)
```

```
##              dBIC df
## mod_spline    0.0 4
## mod_richards 16.4 3
## mod_base     50.5 2
## mod_probit   81.8 2
```

Predictions

To see how the model fits the data, in addition to residuals as above, it helps to plot observations and predictions together. This could be done to examine any of the models above. We could have compared them using log-likelihoods or information criteria, but we'll demonstrate how to do that in the next section.

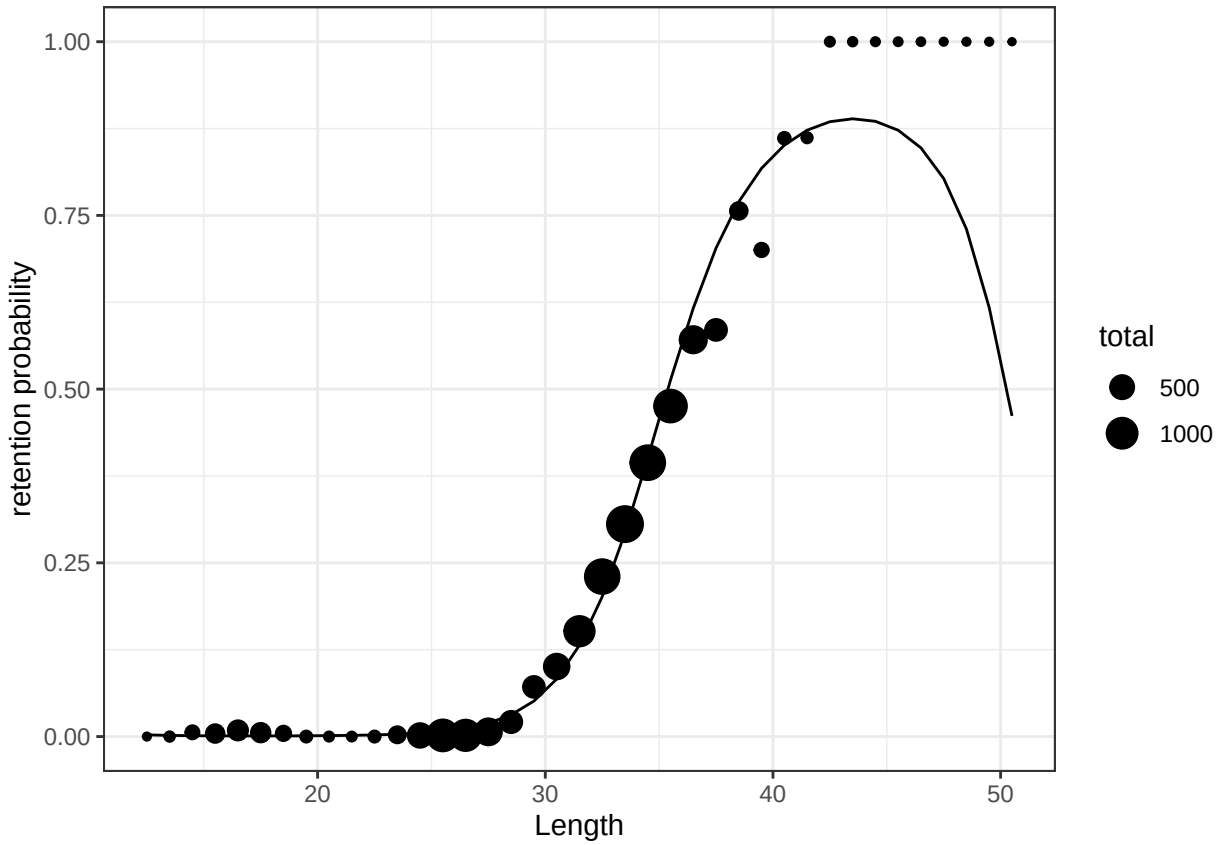
```
newdata = expand.grid(Length=unique(coverhaddock_120low$Length),
    total=1,
    haul=NA,
```

```
    qratio=1)

newdata$prop = predict(mod_spline, newdata=newdata, type="response")
```

For plotting observations, we need to aggregate the hauls and raise the data by the raising factor. The raised data is only used for plotting, not for statistical analyses.

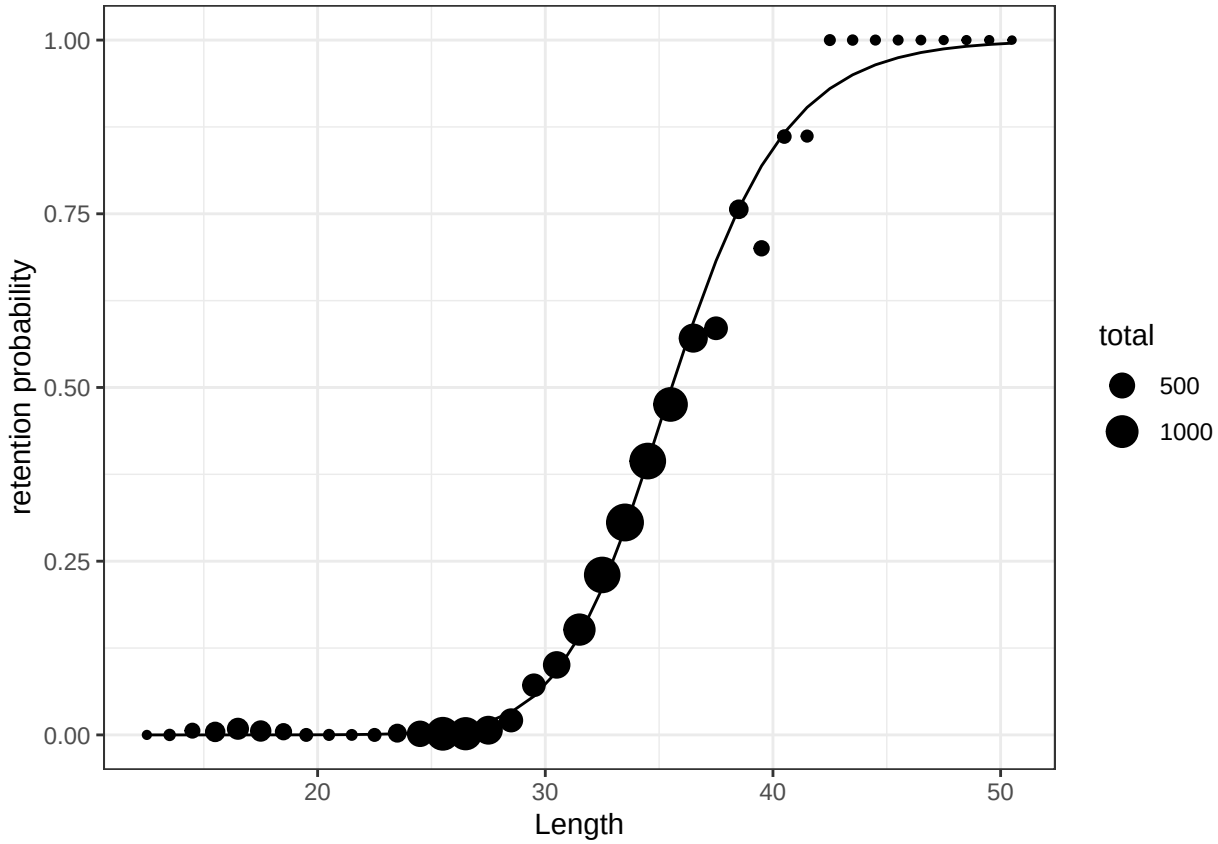
```
sumdat_base = ddply(coverhaddock_120low, ~Length, summarize,
    prop = sum(codend)/sum(total),
    total = sum(total),
    raised_prop = sum(codend * cod_rf)/
        sum(codend * cod_rf + cover* cov_rf),
    raised_total=sum(codend * cod_rf + cover* cov_rf)
)
```

```
ggplot(sumdat_base, aes(Length, prop))+
    geom_point(aes(size=total, y=raised_prop))+
    geom_line(data=newdata)+
    ylab("retention probability")
```



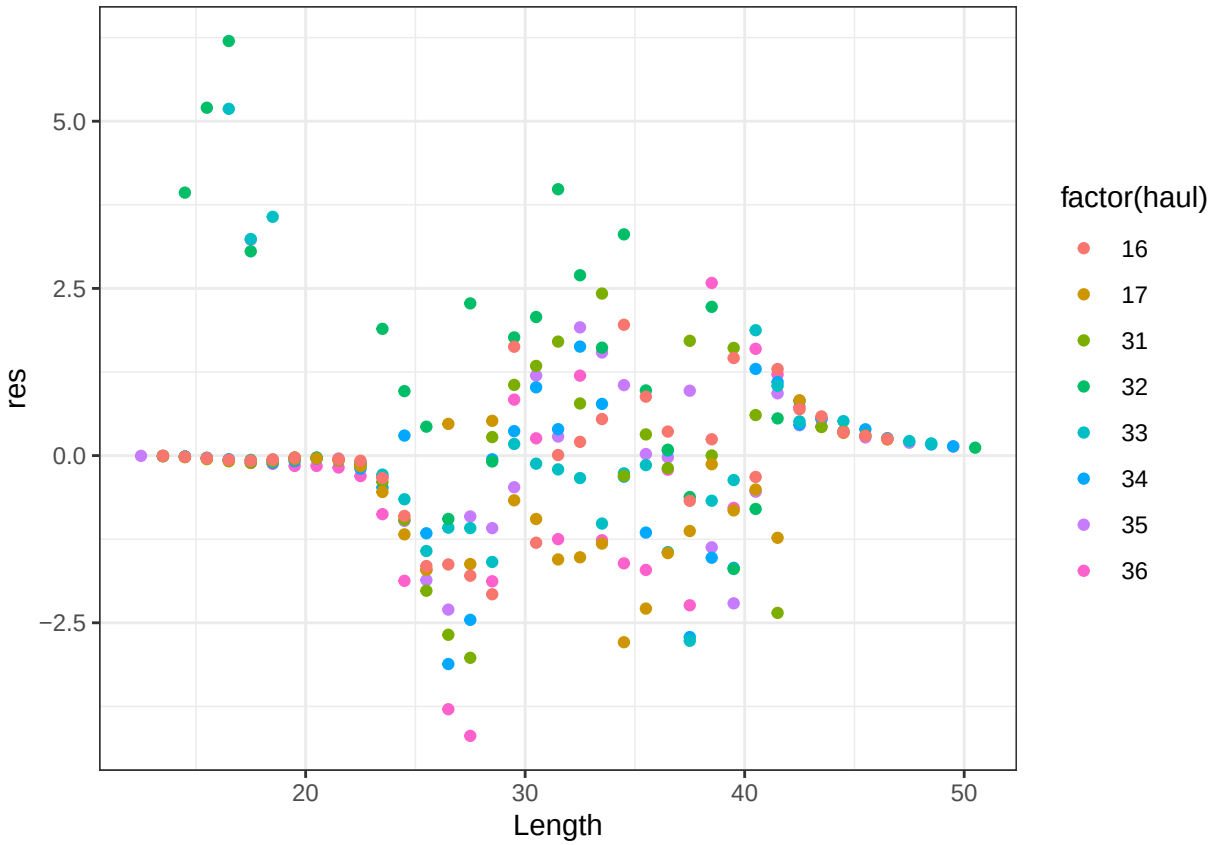
Even though it had the lowest BIC, the spline in this case gives unreasonable predictions for the larger length classes. So we check if the next best model (with Richard's link) looks closer to the data. Alternatively, we could try splines with more degrees of freedom (e.g. `df=4`), but the important thing is to get a model with unbiased predictions in the end.

```
newdata$prop = predict(mod_richards, newdata=newdata, type="response")


ggplot(sumdat_base, aes(Length, prop))+
    geom_point(aes(size=total, y=raised_prop))+
    geom_line(data=newdata)+
    ylab("retention probability")
```



The predictions look reasonable, and we can also check the residuals for the model with Richard's link.

```
coverhaddock_120low$res = residuals(mod_richards, type="deviance")
ggplot(coverhaddock_120low, aes(Length, res, colour=factor(haul)))+geom_point()
```



Confidence intervals by bootstrapping

Following the method of Millar (1994) the bootstrapping function `bootSel` resamples hauls, then resamples fish within hauls, fits the model to the resampled data, then applies a function `FUN` to each fitted model. In the code below, we define `FUN` to make predictions from each fitted model onto `newdata`. The type of predictions we want in this case are the retention probabilities, i.e. the estimated selection curve, so we specify `type="selection"`. To read about the `predict` function, type `?predict.selfisher` in the R console.

Mac and Linux bootstrapping in parallel

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

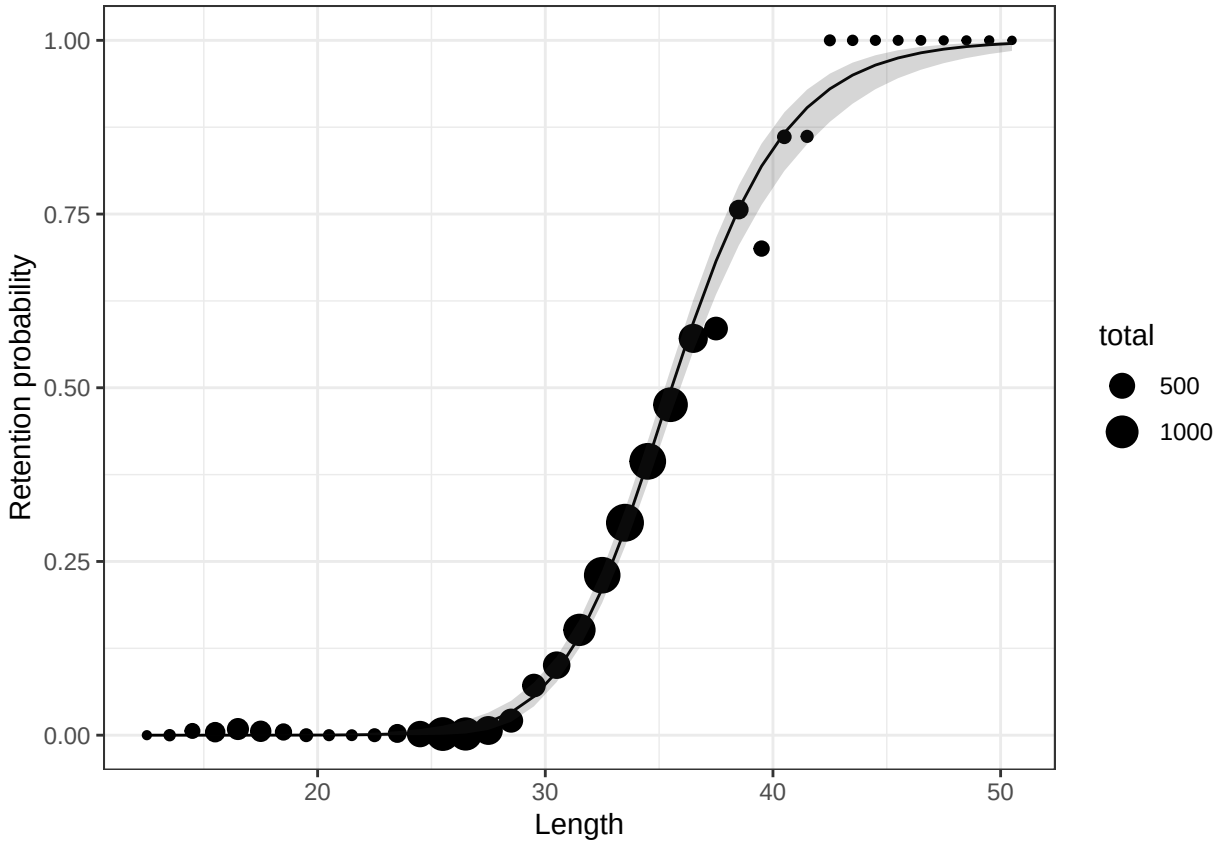
Windows bootstrapping in parallel

```
ncpus = 4
cl = makeCluster(rep("localhost", ncpus))
clusterExport(cl, "newdata")
bs = bootSel(mod_richards, nsim=1000, parallel = "snow", cl=cl,
    FUN=function(mod){predict(mod, newdata=newdata, type="selection")})
stopCluster(cl)
```

Then we calculate quantiles across bootstraps for each row of `newdata`.

```
quants = apply(bs$t, 2, quantile, c(0.025, 0.5, 0.975))
newdata[,c("lo", "mid", "hi")] = t(quants)
```

```
ggplot(sumdat_base, aes(Length, prop))+geom_point(aes(size=total, y=raised_prop))+
   geom_line(data=newdata)+
   geom_ribbon(data=newdata, aes(ymin=lo, ymax=hi), alpha=0.2)+
    ylab("Retention probability")
```



All four codends in one model

We can analyze the data from all gear types together and test if the gear type affects selectivity by comparing models of varying complexity. If we want to use models directly (i.e. before bootstrapping) for testing the significance of a variable, we need to account for variability among hauls and avoid pseudoreplication (Hurlbert 1984) by including a random effect of haul. The random effect is only needed when models are used directly for hypothesis testing (e.g. to test our hypothesis that the gear types differ in their selectivity).

For a thorough investigation, we could try different link functions for this analysis just as we did above, but first we'll try Richard's link because it worked well for the subset of the data that we analyzed above. Then it's important to check for bias in the residuals, or by looking at the model predictions vs observations.

```
mod0 = selfisher(prop~ Length +(1|haul), qratio=qratio, total=total, coverhaddock, link = "richards")
```

```
## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation
```

```
mod1 = selfisher(prop~ Length * gear +(1|haul), qratio=qratio, total=total,
    coverhaddock, link = "richards")
```

```
## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation

## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation
```

```
BICtab(mod0, mod1)
```

```
##      dBIC df
## mod1  0.0 10
## mod0 33.5 4
```

This tells us that `mod1` is more parsimonious, which means that the 4 codends are somehow different. We can explore which aspect of the gears made them different. We'll assume that Baranov's Principle Of Geometric Similarity applies, i.e. selectivity is a function of fish length scaled by mesh size. It could also be reasonable to fit models with a main effect of mesh size and length in addition to an interaction term, but we do not attempt to fit all possible models here and leave it to users to decide what applies to their particular study.

```
mod_all = selfisher(prop ~ Length + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_c = selfisher(prop ~ Length + catch + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_s = selfisher(prop ~ Length + stiffness + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_m = selfisher(prop ~ Length:I(1/mesh) + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_c_s = selfisher(prop ~ Length + catch + stiffness + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_c_m = selfisher(prop ~ Length:I(1/mesh)  + catch + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_s_m = selfisher(prop ~ Length:I(1/mesh) + stiffness + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_c_s_m = selfisher(prop ~ Length:I(1/mesh) + catch + stiffness + (1|haul),
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

mod_all_c_s_m_g = selfisher(prop ~ Length:I(1/mesh) + catch + stiffness + (1|haul),
    dformula = ~gear, #let the d parameter of the Richards curve vary by gear
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")

BICtab(mod_all, mod_all_c, mod_all_s, mod_all_m, mod_all_c_s, mod_all_c_m,
    mod_all_s_m, mod_all_c_s_m, mod_all_c_s_m_g)
```

```
##                 dBIC df
## mod_all_c_s_m    0.0 6
## mod_all_c_s_m_g  2.1 9
## mod_all_s_m     10.4 5
## mod_all_s       22.1 5
## mod_all_c_s     28.9 6
## mod_all_c_m     30.8 5
## mod_all         36.3 4
## mod_all_c       42.4 5
## mod_all_m         NA 4
```

Several models gave warning messages as often occurs with Richard's link. A feature of the `ICtab` methods in `bbmle` is that any models that do not converge will have their IC reported as NA and will drop to the bottom of the list as occurred with `mod_all_m`. It is probably fine to ignore the unconverged model as long as the top model appears to adequately capture the patterns in the data while examining the residuals.

Bootstrapping

We drop the random effect of haul from the most parsimonious model before bootstrapping for two reasons (1) the bootstrapping method resamples among and within hauls and thereby accounts for variation among hauls, and (2) random effects make model fitting much slower which can be burdensome when refitting the model 1000 times or more.

```
mod_all_c_s_m_FE = selfisher(prop ~ Length:I(1/mesh) + catch + stiffness,
    qratio=qratio, total=total, haul=haul, data=coverhaddock, link = "richards")
```

```
## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation

## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation

## Warning in nlminb(start = par, objective = fn, gradient = gr, control =
## optControl): NA/NaN function evaluation
```

```
summary(mod_all_c_s_m_FE)
```

```
##  Family: binomial  ( Richards )
## Selectivity formula:          prop ~ Length:I(1/mesh) + catch + stiffness
## Data: coverhaddock
## Total: total
##
##           AIC           BIC        logLik      deviance Pearson.ChiSq
##        3704.9        3729.2       -1847.4        2035.6      157172.2
##      df.resid
##           958
##
##
## Richards exponent parameter (delta): 0.438
##
## Selectivity model:
##                    Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)      -11.732852   0.435121  -26.96   <2e-16 ***
## catch             -0.001531   0.000103  -14.86   <2e-16 ***
## stiffness          2.305502   0.083235   27.70   <2e-16 ***
## Length:I(1/mesh)  40.905610   1.203304   33.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then we create a new data set to use for predictions. It must include all variables that appear in the model. Even though `haul` is not used in the mathematics behind the predictions, it must be included in the new data for technical reasons. We include `gear` just because it makes it easy to organize and plot the data further down. For each gear, we will make predictions over the range of catch weights observed for that gear. In this particular example, we need to expand the range of length classes beyond those observed in the data to be wide enough to calculate SR for all bootstraps.

```
tmp = unique(coverhaddock[,c("stiffness", "mesh", "gear", "catch")])

newdata_v2 = expand.grid.df(data.frame("Length"=0:100), tmp) #wider than observed range

newdata_v2 = transform(newdata_v2,
    qratio=1,
    total=1,
    haul=NA)

set.seed(111)
```

Windows bootstrapping code

```
ncpus = 4
cl = makeCluster(rep("localhost", ncpus))
clusterExport(cl, "newdata")
bs = bootSel(mod_all_c_s_m_FE, nsim=1000, parallel = "snow", cl=cl,
    FUN=function(x){predict(x, newdata= newdata_v2, type="selection")})
stopCluster(cl)
```

Mac and linux bootstrapping code

```
bs= bootSel(mod_all_c_s_m_FE, nsim=1000, parallel = "multicore", ncpus = 4,
    FUN=function(x){predict(x, newdata= newdata_v2, type="selection")})
```

Organize bootstrap predictions with the predictor variables

Then we organize the bootstrap results and join them with the newdata used for predictions.

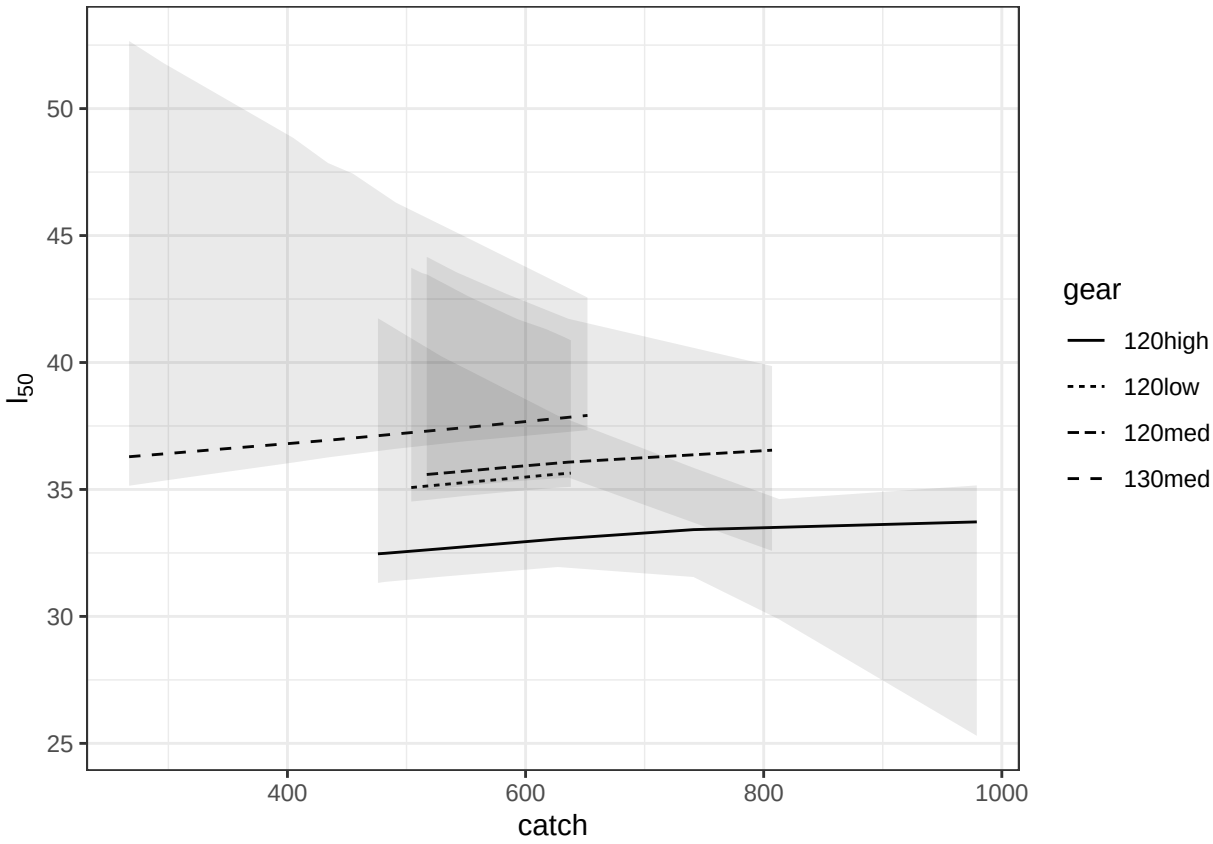
```
quants = apply(bs$t, 2, quantile, c(0.025, 0.5, 0.975))

newdata_v2[,c("lo", "mid", "hi")]=t(quants)

#all the bootstrap predictions with the variables used to create them
newdata_v3 = cbind(newdata_v2[,c("Length", "gear", "mesh", "stiffness", "catch")], t(bs$t))
```

```
#put them in long format (i.e. separate the different bootstrap replicates)
newdata_v3 = melt(newdata_v3, id.vars=1:5)

names(newdata_v3)[6:7] = c("rep", "predicted_r")
```

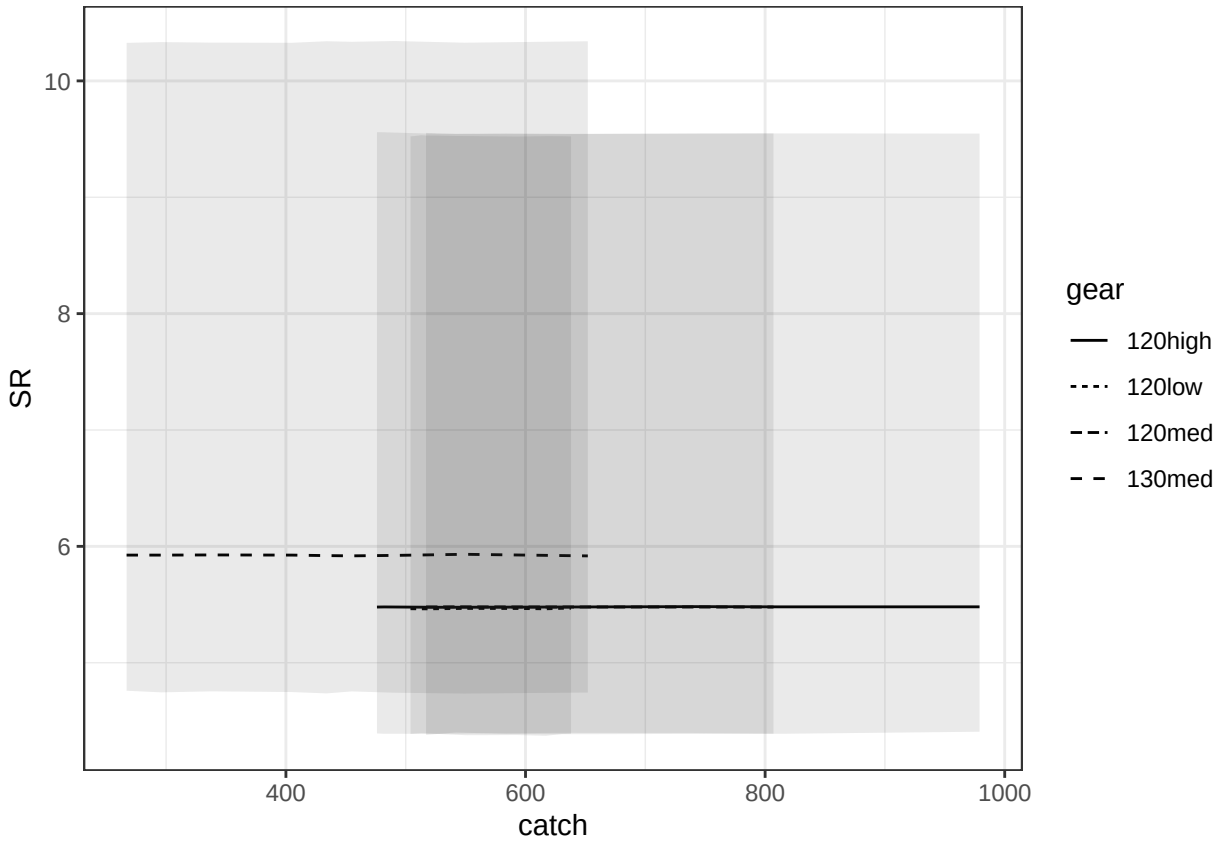
Calculate l_{50} and SR from bootstraps

Here we define a function to evaluate the length at which a given proportion of fish (p) are retained using interpolation. We use this function to find the l_{50} and SR for each bootstrap while varying over mesh size, twine bending stiffness and catch size and subsequently find the mean l_{50}, mean SR and their 95% confidence limits. We plot the results against catch size, which is how they were presented in the original analysis of O'Neill et al (2016).

```
# Function to interpolate lengths - assumes y strictly increases with x
findx=function(x,y,p=0.5) {
  n=1:length(x)
  lo.obs=sum(y<p)
  hi.obs=lo.obs+1
  delta=(p-y[lo.obs])/(y[hi.obs]-y[lo.obs])
  x[lo.obs]+delta*(x[hi.obs]-x[lo.obs])
}

#find l50 and SR for each bootstrap, still varying by gear characteristics and catch
sums0 = ddply(newdata_v3, ~gear + mesh + stiffness + catch + rep, .inform = TRUE,
    summarize,
    SR = findx(Length, predicted_r, 0.75)-findx(Length, predicted_r , 0.25),
    l50 =  findx(Length, predicted_r, 0.5)
)

#summarize over bootstraps, but still varying by gear characteristics and catch
sums1 = ddply(sums0, ~gear + mesh + stiffness + catch , summarize,
    l50_lo=quantile(l50, 0.025),
    l50_mid=quantile(l50, 0.5),
    l50_hi=quantile(l50, 0.975),
    SR_lo=quantile(SR, 0.025),
    SR_mid=quantile(SR, 0.5),
    SR_hi=quantile(SR, 0.975)
)

ggplot(sums1, aes(x=catch, group=gear))+
    geom_line(aes(y=l50_mid, lty=gear))+
    geom_ribbon(aes(ymin=l50_lo, ymax=l50_hi), alpha=0.1)+
    ylab(expression(l[50]))
```




```
ggplot(sums1, aes(x=catch, group=gear))+
    geom_line(aes(y=SR_mid, lty=gear))+
    geom_ribbon(aes(ymin=SR_lo, ymax=SR_hi), alpha=0.1)+
    ylab("SR")
```

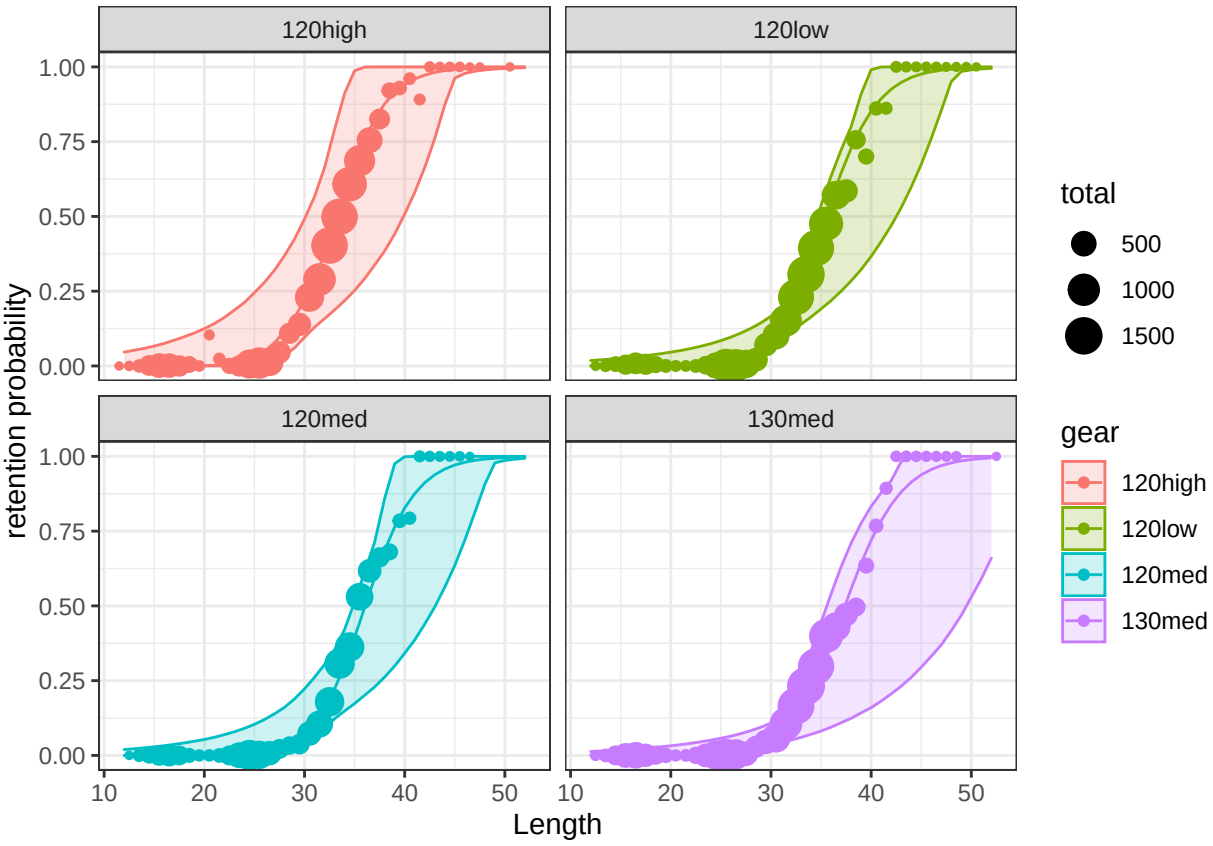


The plots presented here are very comparable to Fig 1 from the original manuscript, which finds a similar dependence of haddock l_{50} on mesh size, twine bending stiffness and total codend catch weight but finds that SR is a constant (O'Neill et al., 2016). We should not, however, expect the results to be identical as there are many differences between the analyses. In the original, it was assumed that the slope and intercept of the logistic link functions vary randomly from haul to haul and that l_{50} and log SR mean selection curve were linearly dependent on the explanatory variables. Whereas, here we have assumed geometric similarity and that overall retention is related to the explanatory variables.

Below we plot retention curves for each gear aggregated over catches.

```
sumdat = ddply(coverhaddock, ~Length+gear, summarize,
    prop=sum(codend)/sum(total), total=sum(total),
    raised_prop=sum(codend * cod_rf)/sum(codend * cod_rf + cover* cov_rf),
    raised_total=sum(codend * cod_rf + cover* cov_rf)
)

newdata_v4 = ddply(newdata_v3,~Length + gear + mesh + stiffness , summarize,
    lo = quantile(predicted_r, 0.025),
    mid = quantile(predicted_r, 0.5),
    hi = quantile(predicted_r, 0.975)
)

p1 = ggplot(sumdat, aes(colour=gear))+geom_point(aes(size=total, x=Length, y=raised_prop))+
  geom_line(data=newdata_v4, aes(x=Length, y=mid))+
  geom_ribbon(data=newdata_v4, aes(x=Length, ymin=lo, ymax=hi, fill=gear), alpha=0.2)+
    ylab("retention probability")+
    xlim(range(sumdat$Length))
```

```
p1+facet_wrap(~gear)
```



References

Hartig F. (2020). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.2.7. https://CRAN.R-project.org/package=DHARMa

Hurlbert S.H. (1984) Pseudoreplication and the design of ecological field experiments. Ecol. Monogr. 54:187–211.

O'Neill F.G., Kynoch R.J., Blackadder L., Fryer, R.J., Eryasar A.R., Notti, E., and Sala A., 2016. The influence of twine tenacity, thickness and bending stiffness on codend selectivity. Fisheries Research. 76, 94-99.

Millar R.B. (1994) Sampling from trawl gears used in sized selectivity experiments, ICES Journal of Marine Science, 51:293–298, https://doi.org/10.1006/jmsc.1994.1030