R Code Supplemental Material

1. Introduction

This supplement contains R code used for modelling predictors of catch per unit effort (CPUE) in an experimental summer longline fishery in Cumberland Sound, along with related supplemental tables and figures. Custom functions used throughout are also included in the Annex of this supplement.

The dataset consists of numbers of individuals of three species caught in 89 longline sets deployed across two years:

```
# Load in packages and data

pacman::p_load(tidyverse,ggplot2,mgcv,lubridate,MuMIn,DHARMa,sf,knitr,
               kableExtra,gratia,patchwork)

cpue = read_csv("Cumberland Sound CPUE.csv",show_col_types = F)
cpue
```

```
## # A tibble: 89 x 10
##    date          year   lat   lon soak.time hooks depth halibut_n shark_n skate_n
##    <chr>        <dbl> <dbl> <dbl>     <dbl> <dbl> <dbl>     <dbl>   <dbl>   <dbl>
##  1 Aug. 01, 2~  2010  65.7 -65.9      14.5  1500   856       216      17      70
##  2 Aug. 02, 2~  2010  65.7 -65.9      21.2  1500  1097        12       6      20
##  3 Aug. 03, 2~  2010  65.8 -66.1      22.9   900  1006        50      25       8
##  4 Aug. 10, 2~  2010  65.7 -65.9      17.3   750   796       129       1      25
##  5 Aug. 10, 2~  2010  65.7 -65.9      16     750   942       120       6      13
##  6 Aug. 11, 2~  2010  65.7 -65.9       9.5   750   914       135       1      61
##  7 Aug. 11, 2~  2010  65.7 -65.9      17.3   750   951       172      13      22
##  8 Aug. 11, 2~  2010  65.7 -65.9      15     750   896       129       2      15
##  9 Aug. 15, 2~  2010  65.7 -65.9      11.3  1500  1024       351       2      19
## 10 Aug. 17, 2~  2010  65.7 -65.9      18.5   750  1052       136       3      10
## # ... with 79 more rows
```

2. Data inspection and processing

First we carry out a few basic data processing steps including re-labelling species, formatting date and year columns, and rearranging the data to long format. We also calculate nominal CPUE as number of individuals caught per 1000 hooks:

```
cpue =
  rename(cpue,`Greenland halibut` = halibut_n, `Greenland shark` = shark_n,
         `Arctic skate` = skate_n) %>%
```

```
  dplyr::select(!contains('_kg'))          %>%
  mutate(date = mdy(date), year=as.factor(year),day = yday(date)) %>%
  gather(key='species',value='catch',-(c(date:depth,day))) %>%
  mutate(cpue = catch * (1000/hooks)) %>%
  dplyr::select(year,date,day,species,catch,cpue,everything())
```

Inspection of the data shows that CPUE is a non-negative continuous variable, and is strongly right skewed for Greenland shark and Arctic skate:

```
ggplot(cpue,aes(x = cpue)) +
  geom_histogram(fill='grey80',colour='black',bins=20) +
  facet_wrap(~species,scales = 'free_x') +
  labs(y = 'Frequency', x = bquote('CPUE'~(individuals.hooks^-3)))
```

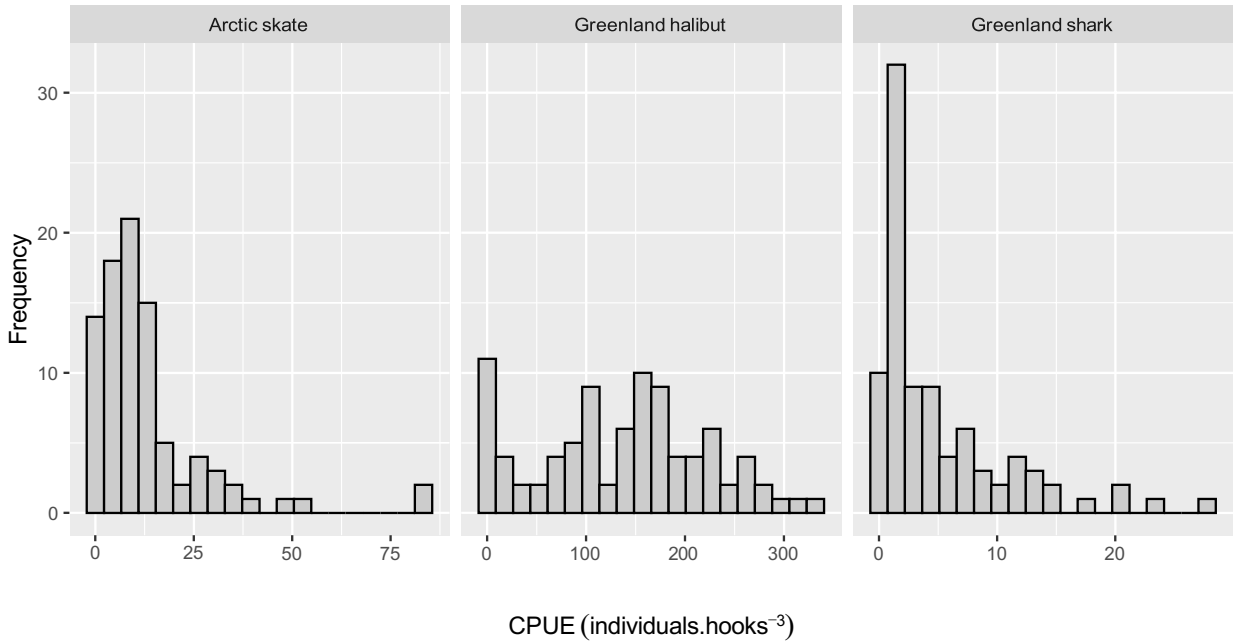

Figure S1: Frequency distribution of catch per unit effort (CPUE) in the Cumberland Sound summer longline fishery.

A cursory inspection of the data suggests that there is significant seasonal variation in nominal CPUE for all species:

```
ggplot(cpue,aes(x = day, y = cpue, colour = year)) +
  facet_wrap(~species,scales='free_y') +
  labs(x = 'Day of year', y = bquote('CPUE'~(individuals.hooks^-3))) +
  geom_point() + geom_smooth(method='gam',size=0.5) +
  theme(legend.position = 'top')
```

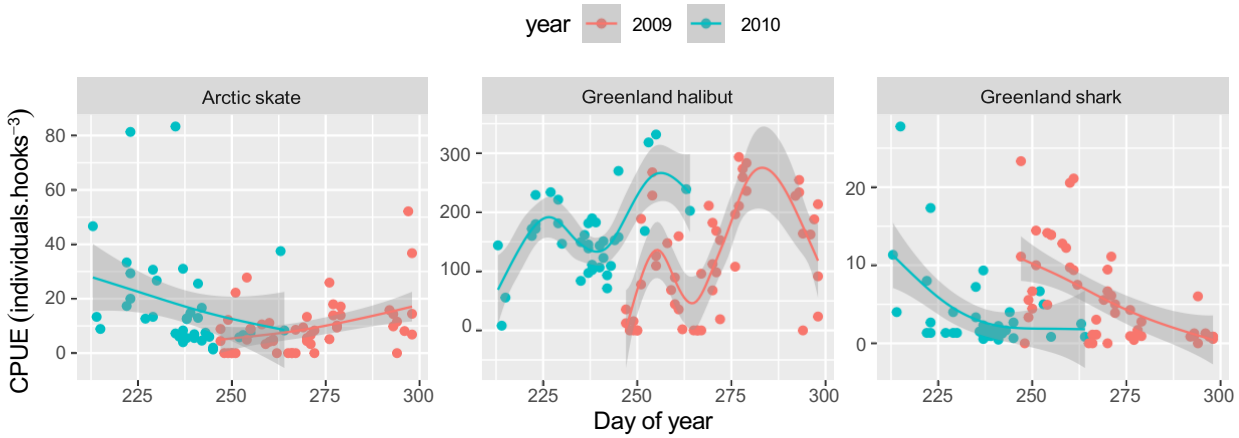

Figure S2: Seasonal variation in nominal CPUE in the Cumberland Sound summer longline fishery. Trend lines are centred thin plate regression splines plotted on the scale of linear predictor, along with 95% CIs (shaded ribbon).

Figure S2 also highlights the different fishing periods in each year, which is linked to summer sea ice cover in Cumberland Sound. Sea ice retreated earlier in 2010 compared to 2009 allowing fishing to commence approximately 1 month earlier.

```
ice = read.csv("Cumberland Sound ice cover.csv") %>%
      mutate(date = dmy(date),day = yday(date), year = year(date))

fish.season = group_by(cpue,year) %>%
              summarise(start = min(day),end = max(day))

ggplot(ice,aes(x=day,y=percent.ice)) +
  geom_ribbon(aes(ymin=0,ymax=percent.ice),fill='#99FFFF',alpha=.7) +
  geom_line(colour = '#14959e') +
  geom_segment(aes(y=25,yend=25,x = start,xend=end),data = fish.season) +
  labs(x = 'Day of year',y = 'Percent sea ice cover') +
  facet_wrap(~year,ncol=1)
```

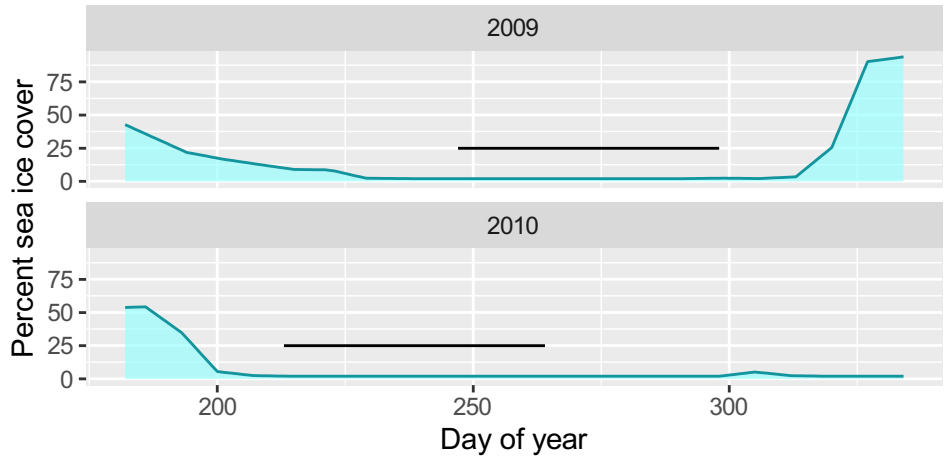

Figure S3: Timing of the Cumberland Sound longline halibut fishery (black bar) in relation to sea ice cover (blue polygon) across two years of the study. Sea ice data obtained from the Canadian Ice Service.

Seasonal variation in CPUE could be driven by several different factors, which can be modeled in subtly different ways:

1) Seasonal trends in local abundance or catchability, which we model as day of the year (day). These may be constant or vary between years e.g. due to the timings of sea ice retreat which have been shown to effect migrations of Arctic species.
2) Local stock depletion by the fishery, which we model as the cumulative number of individuals caught since the beginning of each fishing season (cumulative.catch).
3) Movement of the fishery between patches (e.g. deplete and move), which we model as the longline set number within a given a season (set).

```
cpue =   group_by(cpue,species,year) %>%
         arrange(day) %>%
         mutate(cumulative.catch = lag(cumsum(catch),default = 0)) %>%
         mutate(fish.days = day - min(day)) %>%
         mutate(set = row_number()) %>%
         ungroup()
```

For modelling spatial effects we convert our data to projected coordinates.

```
# Convert to projected coordinates (UTM zone 20N)
cpue   =   st_as_sf(cpue,coords=c('lon','lat'),crs=4326)   %>%
        st_transform(32620) %>%
        st_coordinates %>% as.data.frame %>%
        setNames(c('X','Y')) %>%
        bind_cols(cpue,.) %>% dplyr::select(-(lat:lon))
```

We then nest our data frame by species to facilitate model fitting to each.

```
cpue = nest(cpue,data = -species)
cpue
```

```
## # A tibble: 3 x 2
##   species              data
##   <chr>                <list>
## 1 Greenland halibut <tibble [89 x 13]>
## 2 Greenland shark   <tibble [89 x 13]>
## 3 Arctic skate      <tibble [89 x 13]>
```

3. Model fitting and selection

We build a global model containing each of our predictor variables and check for concurvity (the non-parametric equivalent of collinearity) between terms using the mgcv::concurvity function. The function is wrapped in a plotting function to visualise the results (see the Annex).

```
global.model = gam(cpue ~ s(day) + s(set) + s(cumulative.catch) + s(fish.days) +
                           s(depth) + s(soak.time) + year + s(X,Y),
                   data = cpue$data[[1]])

concurvity_plot(global.model)
```

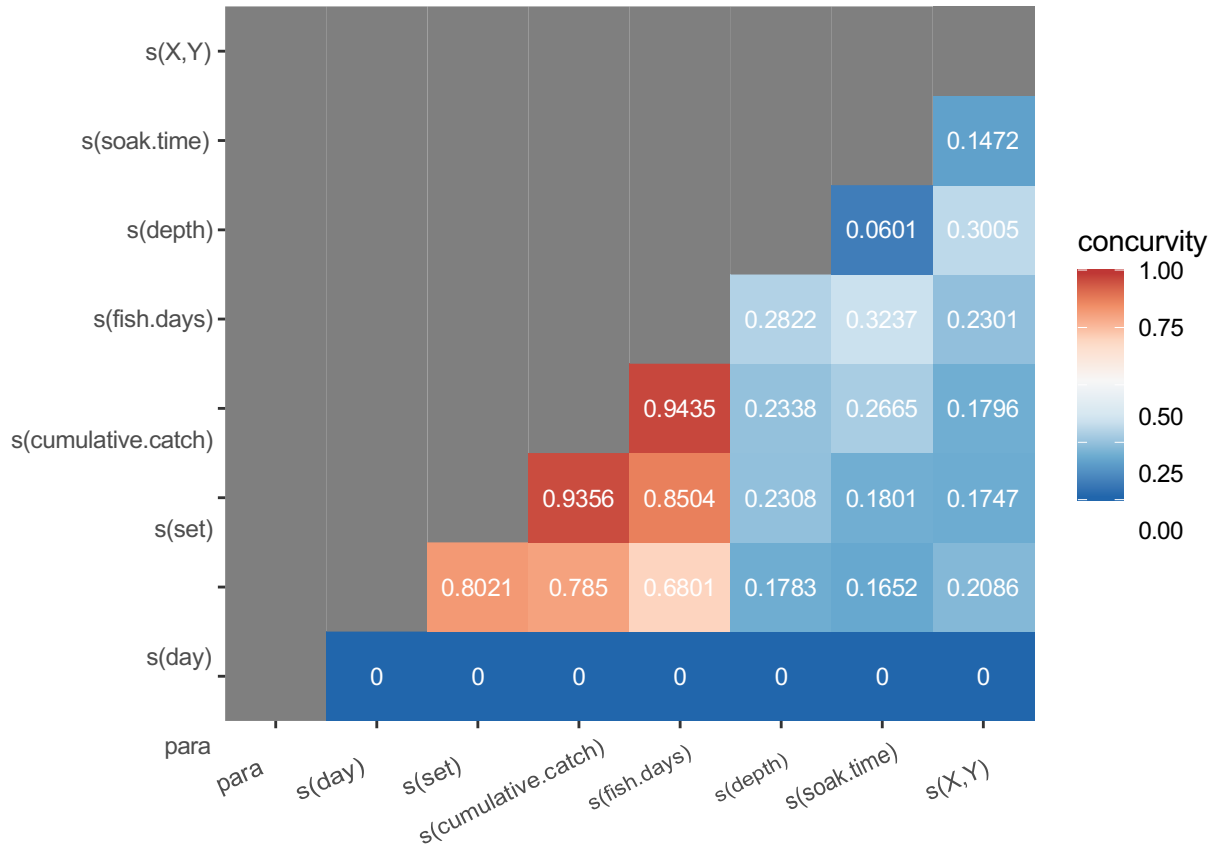

Figure S4: Pairwise concurvity estimates for variables used to model CPUE in the Cumberland Sound longline fishery.

Unsurprisingly, our four 'seasonal' predictors defined above are all highly concurved (0.68 - 0.95). Other covariates in the model less concurvity (< 0.32). We therefore fit models to seasonal variables separately. We also fit models in which the shape of the seasonal curve can vary between years using a factor 'by' smooth:

```
# Define a base model
base.mod = formula(cpue ~ s(depth) + s(soak.time) + year + s(X,Y))

# Update base model to include each of our seasonal predictors
gams = list(
  m1  =  update(base.mod,~.+s(day)),
  m2 = update(base.mod,~.+s(day, by = year)),
  m3  =  update(base.mod,~.+s(set)),
  m4  =  update(base.mod,~.+s(set, by = year)),
  m5  =  update(base.mod,~.+s(cumulative.catch)),
  m6  =  update(base.mod,~.+s(cumulative.catch, by = year))
)
```

Next, we fit models to CPUE of each species in our dataset. First we define a function that iteratively applies the MuMIN::dredge function across each of the model formula defined above. The resulting model selection tables are then merged and ranked based on AICc, allowing us to compare models containing all allowable combinations of predictor variables. Note that the subsetting syntax in dredge ensures that the year term is retained in all models that also contain a smooth x year interaction.

```
AICselect = function(data){

  map(gams,function(.f){

    m = gam(.f, family   = tw(), method = 'REML', data = data, na.action=na.fail)
    dredge(m,subset = dc(year, s(set, by = year)) &&
             dc(year, s(cumulative.catch, by = year)))

  }) %>% Reduce(merge,.)

}

cpue = mutate(cpue, select.table = map(data,AICselect))
cpue
```

```
## # A tibble: 3 x 3
##   species             data                 select.table
##   <chr>               <list>               <list>
## 1 Greenland halibut <tibble [89 x 13]> <model.selection [176 x 16]>
## 2 Greenland shark   <tibble [89 x 13]> <model.selection [176 x 16]>
## 3 Arctic skate      <tibble [89 x 13]> <model.selection [176 x 16]>
```

Inspect the model selection tables using custom function draw.model.selection (defined in the Annex). For presentation purposes we limit our selection to a 99% confidence set (i.e. cumulative Akaike weight < 0.99):

```
map2(cpue$select.table,cpue$species,draw.model.selection,conf.set=0.99)
```

[[1]]

Table S1: Model selection table for Greenland halibutModels are ranked based on the difference in AICc (dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

dAICc	Akaike weight	s(depth)	s(set)	s(X, Y)	year	s(soak.time)	s(day)	s(day x year)	s(cumulative.catch)	s(set x year)	s(cumulative.catch x year)
0.00	0.51	+	+	+	+	+					
1.78	0.35	+	+	+	+	+					
4.02	0.08	+	+	+	+						
6.49	0.02	+	+	+	+	+					
9.33	0.01	+	+								

makecell[l]Models are ranked based on the difference in AICc (dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

[[2]]

Table S2: Model selection table for Greenland sharkModels are ranked based on the difference in AICc (dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

dAICc	Akaike weight	s(soak.time)	s(depth)	year	s(cumulative.catch x year)	s(day)	s(day x year)	s(X, Y)	s(set x year)	s(set)	s(cumulative.catch)
0.00	0.51	+	+	+	+						
4.39	0.10	+	+	+		+	+				
6.45	0.03	+	+	+							
8.11	0.01	+	+	+			+		+		

makecell[l]Models are ranked based on the difference in AICc (dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

Table S3: Model selection table for Arctic skateModels are ranked based on the difference in AICc

(dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

dAICc	Akaike weight	s(depth)	year	s(cumulative.catch x year)	s(soak.time)	s(set)	s(set x year)	s(day)	s(day x year)	s(cumulative.catch)	s(X, Y)
2.67	0.10	+	+	+	+	+					
3.39	0.07	+	+				+	+			
4.91	0.03	+	+		+	+			+		
5.77	0.02	+	+		+		+				
5.85	0.02	+							+		
5.86	0.02	+	+					+			
6.81	0.01	+			+			+			
7.07	0.01		+							+	+
7.60	0.01	+	+								
7.60	0.01	+	+								
7.60	0.01	+	+								
7.60	0.01	+	+								
7.60	0.01	+	+								
7.74	0.01	+	+		+				+		
8.10	0.01	+	+		+			+			
8.20	0.01	+	+		+				+		
8.66	0.00	+	+			+				+	
9.37	0.00	+	+		+						
9.27	0.00	+	+		+						
9.27	0.00	+	+		+						
9.93	0.00		+	+							+
10.29	0.00	+									
10.29	0.00	+									
10.29	0.00	+									
10.29	0.00	+									
10.29	0.00	+									
10.29	0.00	+	+		+					+	
11.19	0.00	+	+							+	+

makecell[l]Models are ranked based on the difference in AICc (dAICc) relative to the best-performing model. The Akaike weights represent the relative degree of support for a given model.

Select the top model for each species for inspection of goodness fit and prediction

```
cpue$best.model = vector(mode = "list", length = nrow(cpue))
for(i in 1:nrow(cpue)) {
  cpue$best.model[[i]]  <-  get.models(cpue$select.table[[i]],subset=1,
                                       data=cpue$data[[i]])[[1]]
}
names(cpue$best.model)  <-  cpue$species

cpue
```

```
## # A tibble: 3 x 4
##   species              data                 select.table                 best.model
##   <chr>                <list>               <list>                       <named list>
## 1 Greenland halibut    <tibble [89 x 13]>   <model.selection [176 x 16]> <gam>
## 2 Greenland shark      <tibble [89 x 13]>   <model.selection [176 x 16]> <gam>
## 3 Arctic skate         <tibble [89 x 13]>   <model.selection [176 x 16]> <gam>
```

4. Model diagnostics

We use a simulation approach for model diagnostics using the DHARMa package. DHARMa uses a simulation to build up an empirical distribution of the residuals when the model is correct and compares this to to observed residuals to test for deviations. Our models use a Tweedie error distribution which is not supported by the

current version of DHARMa::simulateResiduals, so we use the package mgcViz to generate the simulated response instead.

```
simulate_residuals = function(model) {
    createDHARMa(simulatedResponse = mgcViz::simulate.gam(model,1000),
                 observedResponse = model$y,integerResponse=F,
                 fittedPredictedResponse = predict(model))
}

cpue = mutate(cpue, diagnostics = map(cpue$best.model,simulate_residuals))
```

Now check diagnostics for each model:

```
list(~plot(cpue$diagnostics[[1]],title = 'Greenland halibut'),
     ~plot(cpue$diagnostics[[2]],title = 'Greenland shark'),
     ~plot(cpue$diagnostics[[3]], title = 'Arctic skate')) %>%
cowplot::plot_grid(plotlist = .,ncol=1)
```

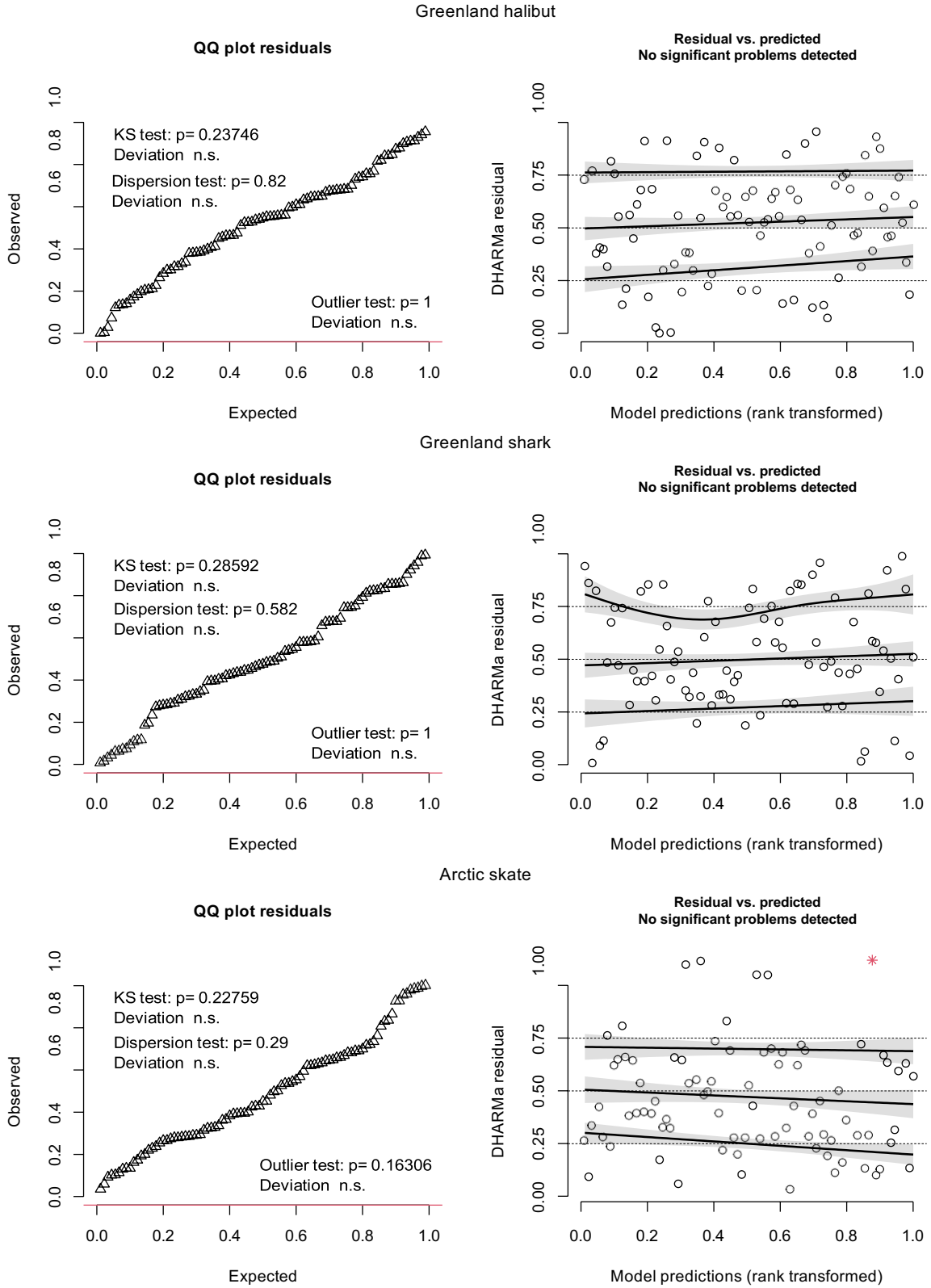

Figure S5: Residual diagnostic plots for GAMs fit to CPUE of Greenland halibut, Greenland shark and Artic skate. Expected residual distributions were generated by simulation using the R package DHARMa and compared to the observed residuals using standard diagnostic plots and statistical tests. Shaded sections around trendlines show 95% CIs. See help for DHARMa for further details.

QQplots and Kolmogorov–Smirnov tests reveal no significant deviations from the expected distribution of residuals. Residual vs. predicted plots show no evidence of significant heteroskedasicity. Outlier tests and disperson tests are also non-significant.

Next, we check each model for residual spatial autocorrelation using global Moran's I test implemented through DHARMa:

```
# We take X and Y coordinates of longline sets from the first model as they are
# the same for all species.
map(cpue$diagnostics,testSpatialAutocorrelation,
    x=cpue$data[[1]]$X,y=cpue$data[[1]]$Y,plot=F)
```

```
## $`Greenland halibut`
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  .x[[i]]
## observed = -0.036405, expected = -0.011364, sd = 0.026113, p-value =
## 0.3376
## alternative hypothesis: Distance-based autocorrelation
##
##
## $`Greenland shark`
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  .x[[i]]
## observed = 0.009983, expected = -0.011364, sd = 0.026126, p-value =
## 0.4139
## alternative hypothesis: Distance-based autocorrelation
##
##
## $`Arctic skate`
##
##  DHARMa Moran's I test for distance-based autocorrelation
##
## data:  .x[[i]]
## observed = 0.022073, expected = -0.011364, sd = 0.026117, p-value =
## 0.2004
## alternative hypothesis: Distance-based autocorrelation
```

Models do not exhibit any significant residual spatial autocorrelation (all $p > 0.2$). Finally, we check each model to make sure the maximum basis dimension we have set is sufficient to capture patterns in the data using the function mgcv::k.check. Here a $p < 0.05$ indicates that there is pattern in the residuals which may require a higher value of k.

```
map(cpue$best.model,k.check)
```

```
## $`Greenland halibut`
##             k'      edf   k-index p-value
## s(depth)     9 5.135785 0.8879105  0.1250
## s(set)       9 6.632901 1.1339050  0.9225
## s(X,Y)      29 10.629921 1.1548133  0.9475
```

```
##
## $`Greenland shark`
##                                k'      edf    k-index p-value
## s(cumulative.catch):year2009    9 3.893844 1.2177742  0.9800
## s(cumulative.catch):year2010    9 2.740627 1.2177742  0.9925
## s(depth)                        9 4.591514 0.8793102  0.1625
## s(soak.time)                    9 5.207523 0.9549037  0.3725
##
## $`Arctic skate`
##                                k'      edf    k-index p-value
## s(cumulative.catch):year2009    9 1.442212 0.8672555    0.12
## s(cumulative.catch):year2010    9 1.000229 0.8672555    0.14
## s(depth)                        9 4.323075 0.8752376    0.19
```

There are no significant problems here. All p-values are > 0.15 and the estimated degrees of freedom (edf) are all well below the maximum possible (k').

5. Model predictions

The top models for each species generally have a high degree of support and appear to fit the data reasonably well. Next, we examine the contributions of each predictor variable to the fitted models.

Next we examine the marginal effects of each smooth term in the top model for each species after controlling for other significant terms. We use the gratia::draw function for this and plot the partial residuals:

```
map(cpue$best.model,draw,residuals=T)
```

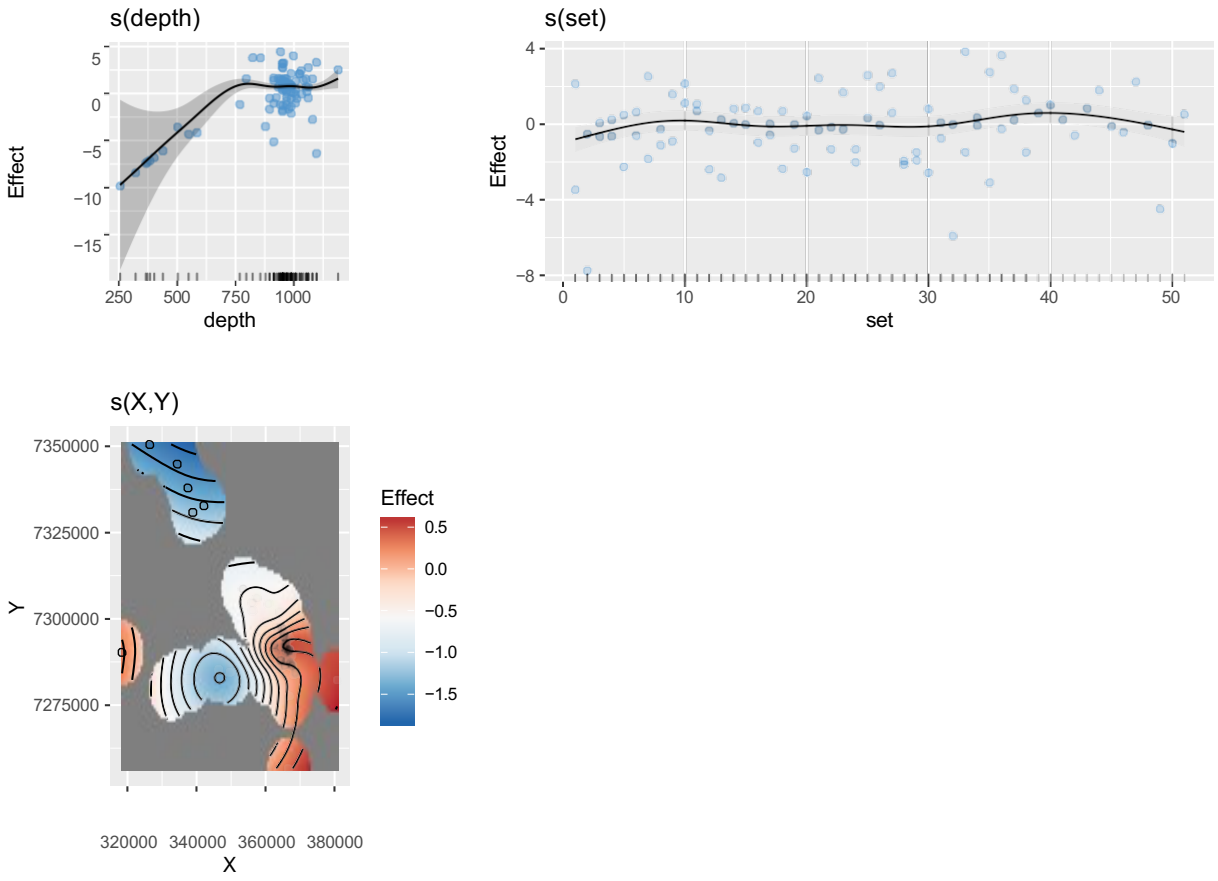

Figure S6: Partial effects of smooth terms in best-fitting GAM on CPUE of Greenland halibut. Trend lines are centred thin plate regression splines plotted on the scale of linear predictor, along with 95% CIs (shaded ribbon). Partial residuals are also plotted in blue.

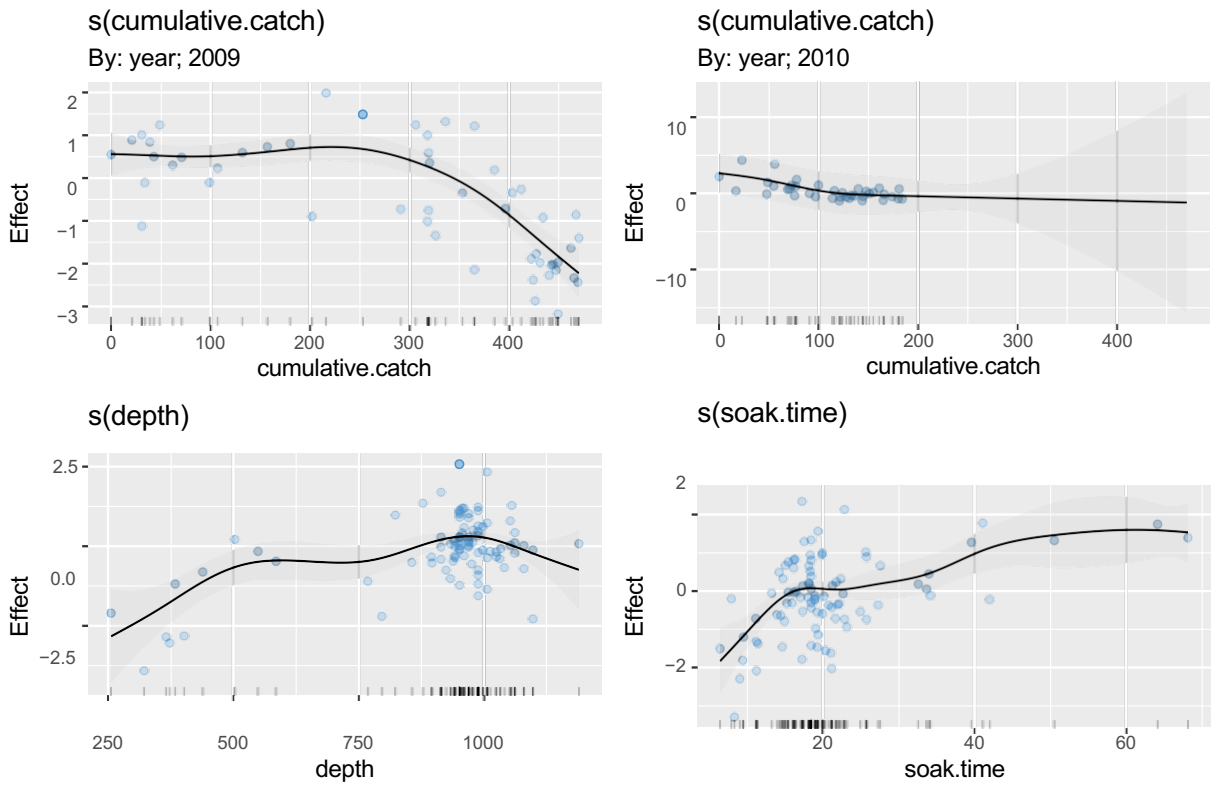

Figure S7: Partial effects of smooth terms in best-fitting GAM on CPUE of Greenland shark. Trend lines are centred thin plate regression splines and are plotted on the scale of linear predictor along with 95% CIs (shaded ribbon). Partial residuals are also plotted in blue.

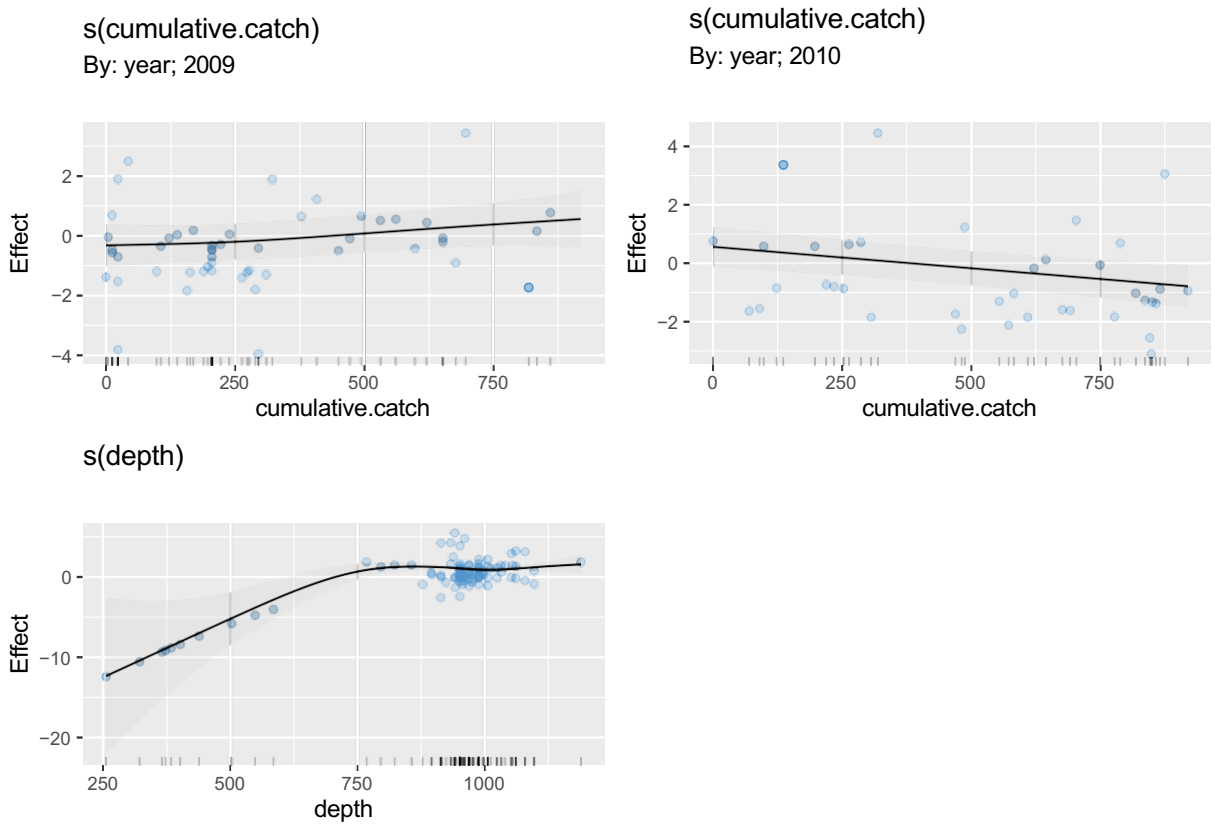

Figure S8: Partial effects of smooth terms in best-fitting GAM on CPUE of Arctic skate. Trend lines are centred thin plate regression splines plotted on the scale of linear predictor along with 95% Cis (shaded ribbon). Partial residuals are also plotted in blue.

Marginal effects of each term on CPUE are easier to interpret when plotted on the response scale. We use a series of custom functions to do this which allows us to tailor a few aspects of the output, as well as identify regions of significant change using gratia::derivatives() (see the Annex). When predicting partial effects on the response scale, all other covariates are fixed at their medians (continuous) or modal categories (factors).

```
# Greenland halibut
map(c('depth','set'),
      ~PE_plot(cpue$best.model[[1]],.x)) %>%
      cowplot::plot_grid(plotlist=.)
```

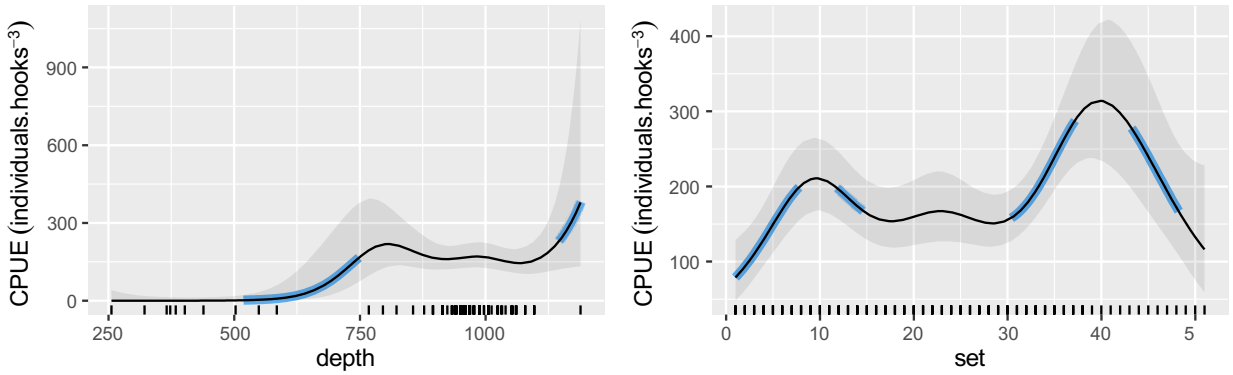

Figure S9: Partial effects of smooth terms in best-fitting GAM on CPUE of Greenland halibut. Smooths are plotted on the response scale. Trend lines are centred thin plate regression splines plotted on the scale of linear predictor along with 95% CIs (shaded ribbon). Areas of significant change (i.e. where first derivatives of the fitted smooth are significantly different than zero) are highlighted in blue. Rug plots show the distribution of data on the x-axis.

```
# Greenland shark
map(c('depth','cumulative.catch','soak.time'),
      ~PE_plot(cpue$best.model[[2]],.x)) %>%
      cowplot::plot_grid(plotlist=.)
```

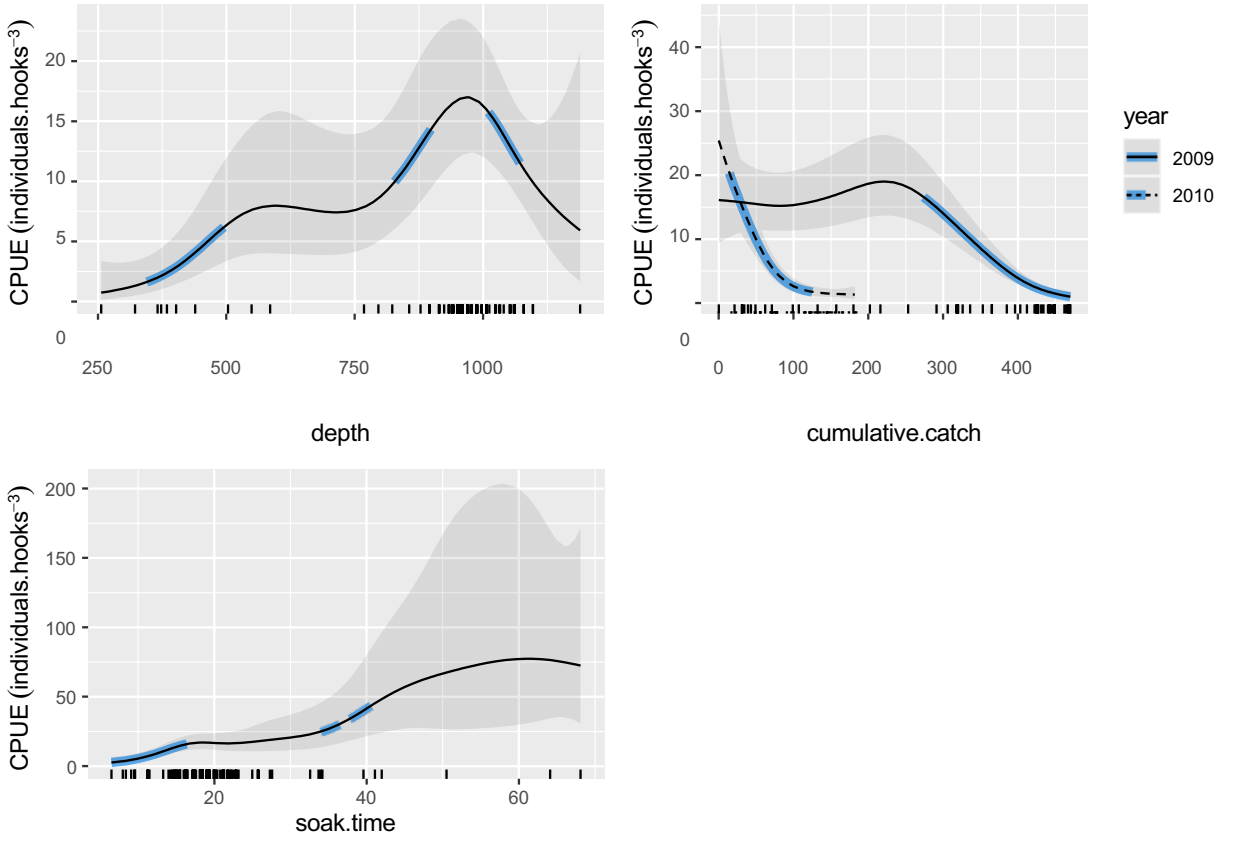

Figure S10: Partial effects of smooth terms in best-fitting GAM on CPUE of Greenland shark. Smooths are plotted on the response scale. Trend lines are centred thin plate regression splines plotted on the scale of linear predictor along with 95% CIs (shaded ribbon). Areas of significant change (i.e. where first derivatives of the fitted smooth are significantly different than zero) are highlighted in blue. Rug plots show the distribution of data on the x-axis.

```
# Arctic skate
map(c('depth','cumulative.catch'),
      ~PE_plot(cpue$best.model[[3]],.x)) %>%
      cowplot::plot_grid(plotlist=.)
```

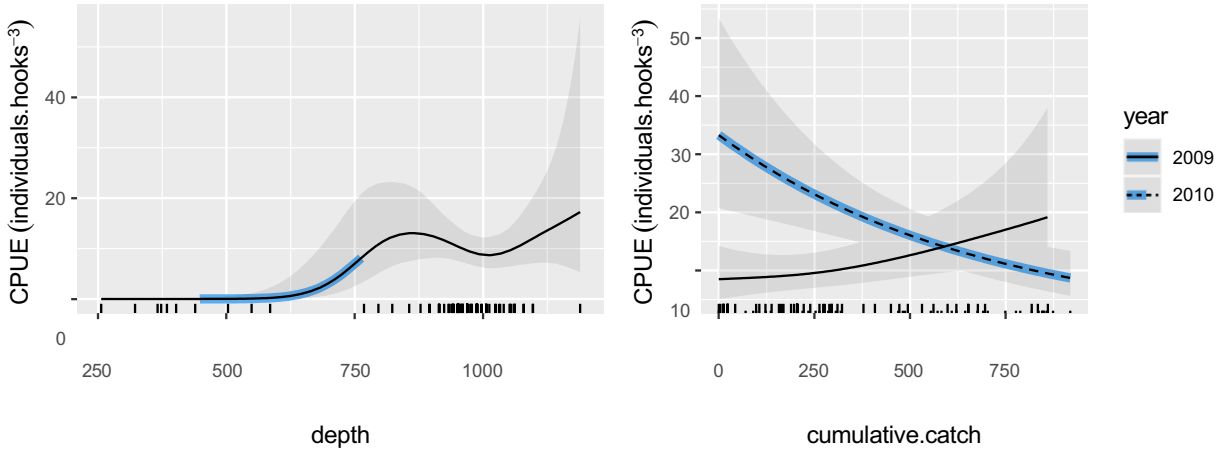

Figure S11: Partial effects of smooth terms in best-fitting GAM on CPUE of Arctic skate. Smooths are plotted on the response scale. Areas of significant change (i.e. where first derivatives of the fitted smooth are significantly different than zero) are highlighted in blue. Rug plots show the distribution of data on the x-axis

Figures S6 and S8 suggest that the shape of the depth smooths for Greenland halibut and Arctic skate are strongly influenced by a single deep set at the extreme of the fished range (depth > 1100 m). When this datapoint is excluded an apparent increase in CPUE at depths > 1000 m is no longer supported. More data at depths > 1100 m are needed to properly constrain the curve in this region so we limit our analysis to the depth range between 250-1100 m.

```
cpue =
mutate(cpue,best.model = map2(best.model,data,
                              ~update(.x,data = subset(.y,depth!=max(depth)))))


  PE_plot(cpue$best.model[[1]],'depth') + ggtitle('Greenland halibut') +
  PE_plot(cpue$best.model[[2]],'depth') + ggtitle('Greenland shark') +
  PE_plot(cpue$best.model[[3]],'depth') + ggtitle('Arctic skate')
```

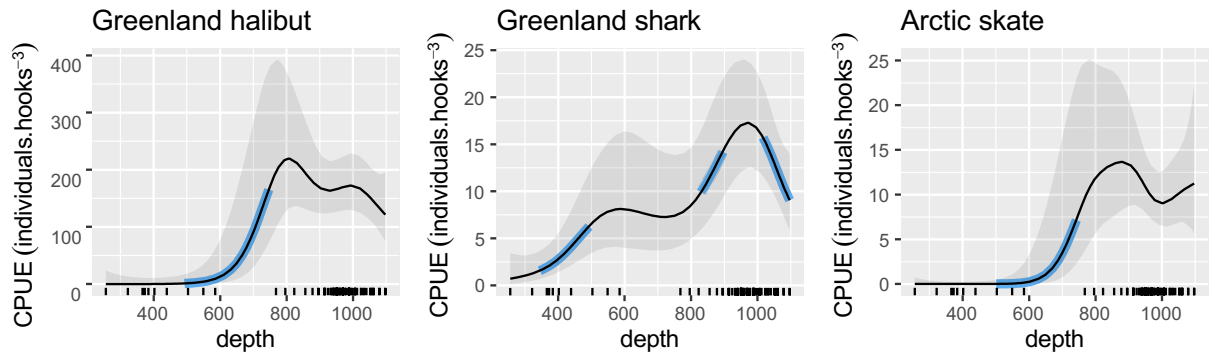

Figure S12: Partial effect of depth on CPUE of Greenland halibut, Greenland shark and Arctic skate when excluding a single influential set at depth > 1100 m

More data at depths > 1100 m are needed to properly constrain the curve in this region so we limit our analysis to the depth range between 250-1100 m.

This yields our final models which are summarised in tables and figures presented in the published manuscript.

6. Model summary

For each smooth term in the final model we extract the estimated degrees of freedom (wiggliness of the spline), the Bayesian approximate p-values estimated internally by package 'mgcv' and associated Wald test statistics:

```
map(cpue$best.model,anova)
```

```
## $`Greenland halibut`
##
## Family: Tweedie(p=1.224)
## Link function: log
##
## Formula:
## cpue ~ s(depth) + s(set) + s(X, Y) + 1
##
## Approximate significance of smooth terms:
##                edf Ref.df      F  p-value
## s(depth)     4.493  5.290  3.214  0.017483
## s(set)       6.763  7.845  5.193  6.12e-05
## s(X,Y)      10.865 13.819  3.545 0.000268
##
## $`Greenland shark`
##
## Family: Tweedie(p=1.176)
## Link function: log
##
## Formula:
## cpue ~ s(cumulative.catch, by = year) + s(depth) + s(soak.time) +
##     year + 1
##
```

```
## Parametric Terms:
##        df     F p-value
## year    1 0.565   0.455
##
## Approximate significance of smooth terms:
##                                   edf Ref.df      F  p-value
## s(cumulative.catch):year2009 3.932  4.826 25.116  < 2e-16
## s(cumulative.catch):year2010 2.743  3.261 30.504  < 2e-16
## s(depth)                     4.713  5.698  8.850 1.07e-06
## s(soak.time)                 5.153  6.254  9.004 5.11e-07
##
## $`Arctic skate`
##
## Family: Tweedie(p=1.466)
## Link function: log
##
## Formula:
## cpue ~ s(cumulative.catch, by = year) + s(depth) + year + 1
##
## Parametric Terms:
##        df     F p-value
## year    1 8.432 0.00477
##
## Approximate significance of smooth terms:
##                                   edf Ref.df      F p-value
## s(cumulative.catch):year2009 1.152  1.288  2.597 0.11301
## s(cumulative.catch):year2010 1.000  1.001 11.596 0.00103
## s(depth)                     4.201  5.100  2.574 0.03520
```

We also assess the relative contribution of each term by calculating the proportion of null deviance explained by models with and without the term of interest. When calculating partial deviances, the smoothing parameters of all other terms in the reduced model are fixed to those estimated in the full model to minimize any compensatory effects.

```
partial_DE = function(model,data){

  vars = getAllTerms(model)
  # Find the term for each smooth
  tms  = map_chr(model$smooth,~paste(.x$term,collapse=', '))
  b0   = update(model,~1,data=data) # Null model

  sapply(vars,function(i){
  # Find which smoothing parameters relate to the term being removed and
  #remove these before fixing remaining params.
  b = update(model,as.formula(paste0('~.-',i)),data=data,sp =
               model$sp[!stringr::str_detect(i,tms)],fixed=T)
    (deviance(b)-deviance(model))/deviance(b0) * 100
  })

}

map2(cpue$best.model,cpue$data,partial_DE)
```

```
## $`Greenland halibut`
```

```
##   s(depth)      s(set)    s(X, Y)
## 18.313592  8.235475 12.940146
##
## $`Greenland shark`
## s(cumulative.catch, by = year)                 s(depth)
##                      48.2764919               16.0754361
##                     s(soak.time)                     year
##                      17.0744316                0.7902698
##
## $`Arctic skate`
## s(cumulative.catch, by = year)                 s(depth)
##                       2.4282479               54.3599173
##                            year
##                       0.8379969
```

7. Extracting derived metrics

By simulating from the posterior of the final, fitted GAMs we can also derive estimates to summarize changes in CPUE as a function of different predictor variables. Simulations are performed using a wrapper around gratia::simulate.gam (see the Annex).

For example, we are interested in quantifying the magnitude of decline in CPUE of Greenland shark as a function of cumulative catch in each of our study years. First need to extract the points at which CPUE began to decline significantly and stopped declining based on our spline derivatives:

```
# We first need to extract the points at which CPUE began to decline
# significantly and stopped declining:
chpts      =
partial_effect(cpue$best.model[[2]],'cumulative.catch')      %>%
group_by(year) %>%
summarise(start_decline = min(data[!is.na(change)]),
          end_decline = max(data[!is.na(change)]))

chpts
```

```
## # A tibble: 2 x 3
##    year  start_decline  end_decline
##    <chr>          <dbl>        <dbl>
## 1  2009           276.          469
## 2  2010            11.8         125.
```

We then simulate 1000 times from the fitted GAM and calculate the mean CPUE before and after periods of change for each iteration.

```
simulate_gam(cpue$best.model[[2]],nsim=1000,'cumulative.catch')      %>%
left_join(chpts)  %>%
mutate(stage = case_when(cumulative.catch <= start_decline ~ 'before',
                         cumulative.catch >= end_decline ~ 'after')) %>%
subset(!is.na(stage)) %>%
group_by(year,iter,stage)  %>%  summarise(est = mean(est))  %>%
group_by(year,stage) %>%
tidybayes::mean_qi(est) %>%
dplyr::select(-(.width:.interval))
```

```
## # A tibble: 4 x 5
##   year  stage    est .lower .upper
##   <chr> <chr>  <dbl>  <dbl>  <dbl>
## 1 2009  after   1.03  0       3.24
## 2 2009  before 17.0  15.7    18.3
## 3 2010  after   1.47  0.864   2.14
## 4 2010  before 24.4  17.6    31.3
```

We use the same approach to quanitfy changes for other species and variables of interest (e.g. soak time & depth).

8. ANNEX. Custom functions

draw.select.table: method for printing model.selection objects output from MuMIn::dredge as an RMarkdown table.

```
draw.model.selection = function(x, species, conf.set = 0.99){

opts <- options(knitr.kable.NA = "")
conf.set = subset(x,cumsum(weight) <= conf.set)[,-1]
var.imp  = importance(x)
conf.set  =  mutate(conf.set,across(where(is.numeric),round,2))  %>%
             dplyr::select(dAICc = delta, `Akaike weight` = weight,
                           names(var.imp)) %>%
             setNames(gsub(', by =',' x',names(.)))

kable(conf.set,align='c', caption = paste0('Model selection table for ',species,
'Models are ranked based on the difference in AICc (dAICc) relative to the
 best-performing model. The Akaike weights represent the relative degree of
support for a given model.'),
      escape=F,row.names = F,booktabs=T) %>%
      kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                    font_size = 9) %>%
footnote(general = 'Models are ranked based on the difference in AICc (dAICc)
          relative to the best-performing model. The Akaike weights represent the
          relative degree of support for a given model.',general_title = '')

}
```

Smooth data: generates new observations for prediction or simulation from individual smooth terms in fitted GAMs. The function either generates n new regularly spaced values between the minimum and maximum of the term in the original data, or accepts specific values at which predictions are to be made. All other terms in the model are set at their median (continuous variables) or modal (categorical variables) values. Returns a dataframe for passing to simulation/prediction function.

```
# Define a function for generating new observations for prediction/simulation.

smooth_data = function(model,term, n=100, values = NULL){

  # Find medians (numeric) and modes (factors) of covariates for prediction
  ref = summarise(model$model,across(where(is.numeric),.fns=median),
        across(where(is.factor),.fns=~names(which.max(table(.x)))))
  ref = ref[,-attr(model$terms, "response")]

  # Create a prediction dataframe for the term of interest with other variables
  # fixed at their medians or modes
  if(is.null(values)){
    newdata  =  tibble(seq(min(model$model[,term]),max(model$model[,term]),
                           length.out = n))
  } else {
    newdata = tibble(values)
  }

  newdata = setNames(newdata,term) %>%
            bind_cols(ref[rep(1,nrow(.)),-grep(term,colnames(ref)),drop=F])
```

```
  # Check if the term is part of a factor-smooth interaction:
  est = smooth_estimates(model,term,partial_match=T)
  is.by = !all(is.na(est$by))

  # And if so generate prediction data for all levels of the by variable,
  # ensuring we don't extrapolate beyond the original range for each level:
  if(is.by) {
    by.var = unique(est$by)
    by.levels = model$xlevels[[by.var]]
    newdata = tidyr::uncount(newdata,length(by.levels)) %>%
    mutate(!!by.var := rep(by.levels,nrow(newdata))) %>%
    left_join(group_by(model$model,!!sym(by.var)) %>%
                summarise(mint = min(!!sym(term)),maxt = max(!!sym(term)))) %>%
    rowwise() %>%
    dplyr::filter(between(!!sym(term),mint,maxt)) %>%
    dplyr::select(-mint,-maxt) %>%
    ungroup
    attr(newdata,'by.var') <- by.var
  }

  return(newdata)
}
```

partial_effect: wrapper around mgcv::predict.gam and gratia::derivatives which estimates partial effects of individual smooth terms in a GAM model on the response scale and identifies regions of significant change. Fitted values and confidence intervals are predicted on the response scale for new observations generated using smooth_data. A column is appended to the prediction dataframe indicating whether the first derivatives (slope) of the smooth differ significantly from zero at each point evaluated.

```
# Define function for estimating partial effects of individual smooth terms on
# the response scale.

partial_effect = function(model,term,n = 200,...){

  # Generate predictions and transform to response scale

  smdata = smooth_data(model,term,n,...)
  newdata = data.frame(predict(model,newdata = smdata,se.fit=T)) %>%
            bind_cols(smdata,.) %>%
            mutate(lcl = fit-1.96*se.fit, ucl = fit+1.96*se.fit) %>%
            mutate(across(c(fit,lcl,ucl),model$family$linkinv))

# Calculate derivatives. Note: in gratia 0.7, derivatives() does not return
# the level of the 'by' variable in smooth-factor interactions so we need to
# retrieve this manually

  by.var = attr(smdata,'by.var')
  deriv = derivatives(model,term=term,newdata=smdata,partial_match = T) %>%
          {if(!is.null(by.var))
            mutate(.,!!by.var := sub(paste0('.*:',by.var),'',smooth))
           else .} %>%
          dplyr::select(-crit,-se,-var) %>%
          rename(!!term:=data,deriv=derivative,der.lcl=lower,der.ucl = upper)
```

```
# Bind derivatives to fitted values and find regions of splines where confidence
# intervals do not overlap zero
 mutate(newdata,term=term,data=!!sym(term)) %>%
 left_join(deriv) %>%
 dplyr::select(term,data,everything(),-smooth,-any_of(names(smdata)),
                  any_of(by.var)) %>%
 mutate(change = ifelse(der.ucl<0 | der.lcl>0 ,fit,NA))


}
```

PE_plot: wrapper around for partial_effect for plotting the partial results of individual smooth terms in a fitted GAM.

```
# Convenience function for plotting partial effects.

PE_plot = function(model,term){
  pe = partial_effect(model,term)
  if(has_name(pe,'year')){
    pl = ggplot(pe,aes(y=fit,x=data,group=year,linetype=year))
   } else {
    pl = ggplot(pe,aes(y=fit,x=data))
  }
  pl +
  geom_line(aes(y=change,linetype=NULL),colour='steelblue2',size=2) +
  geom_ribbon(aes(ymin = lcl, ymax = ucl),alpha=.1) +
  geom_line() +
  geom_rug(aes_string(y=NULL, x = term), data = model$model) +
  labs(y = bquote('CPUE'~(individuals.hooks^-3)), x = term)
}
```

simulate_gam: wrapper around gratia::simulate.gam which formats the simulations and allows us to better control the new observations at which the posterior draws are evaluated using smooth_data.

```
simulate_gam    =    function(model,term,nsim,...){

newdata = smooth_data(model,term,...)
simulate(model,nsim = nsim, newdata = newdata) %>%
as.data.frame %>% setNames(1:nsim) %>%
bind_cols(dplyr::select(newdata,any_of(c(term,'year')))) %>%
gather(.,'iter','est',-any_of(c(term,'year')))

}
```

concurvity_plot: wrapper around the mgcv::concurvity function which extracts the estimated pairwise concurvity between terms in a GAM model and plots them in a visual way.

```
concurvity_plot = function(model){
if(inherits(model,'gamm')) model = model$gam
cc = concurvity(model,full=F)$estimate %>% round(4)
cc[lower.tri(cc,diag=T)]<-NA
as.data.frame(cc)      %>%
```

```
rownames_to_column('var1') %>%
gather('var2','concurvity',-var1)    %>%
mutate(across(c(var1,var2),factor,levels=rownames(cc))) %>%
ggplot(aes(y=var1,x=var2)) + geom_tile(aes(fill = concurvity)) +
geom_text(aes(label = concurvity),colour='white',size=3) +
labs(x=NULL,y=NULL) +
theme(axis.text.x = element_text(angle=25,hjust=1)) +
scale_fill_distiller(limits=c(0,1),palette = 'RdBu') +
coord_cartesian(expand=c(0,0))
}
```