SOSH GAMLSS

Max Czapanskiy

August 15, 2016

Goal: demonstrate improved quality of seabird distribution models by incorporating individual tracking data Method: Locations from tracked sooty shearwaters have been aggregated into kernel density estimates. These utilization distributions will be used in conjunction with more commonly used variables to create distribution models. The model of choice is GAMLSS, for its ability to handle non-linear relationships and errors.

1. Load packages

```
library(gamlss)
library(dplyr)
library(foreach)
library(doParallel)
library(MASS)
library(ggplot2)
```

2. Load data

```
sosh.data <- read.csv('TelemetryTransect17May2016_SOSH-PFSH-COMU.csv') %>%
  dplyr::select(SOSHcount, Binarea, Month,
                Latitude, DistCoast, Dist200, DepCI,
                SSTmean, STD_SST, MEAN_Beaufort, L10CHLproxy, STD_CHL_log10, Watermass,
                L10CHLsurvey, CHLsurvanom, L10CHLsurvclim, FCPI) %>%
  filter(Month > 2) %>% # exclude tiny January, February surveys
  mutate(SOSHcount = as.integer(SOSHcount),
         Month = factor(Month)) %>%
  na.omit
head(sosh.data)
```

```
##   SOSHcount   Binarea Month Latitude DistCoast  Dist200     DepCI  SSTmean   STD_SST MEAN_Beaufort L10CHLproxy
## 1         0 0.6070346     7 45.74943 90.730011 39.84471 0.3441763 15.10290 0.1543211             3 -0.13471743
## 2         0 1.0199987     7 45.75098 31.043267 11.65864 0.1656442 14.72605 0.5041291             3 -0.09102068
## 3         0 1.0199989     7 45.75140 24.261040 17.33907 0.1140940 15.30328 0.1757790             3  0.00484971
## 4         0 1.0164059     7 45.75116 17.711237 22.74802 0.2592593 15.38545 0.2115198             3  0.07141375
## 5         0 1.0199985     7 45.75159 11.183655 28.56728 0.2884615 15.70420 0.1791671             3  0.16948388
## 6         0 1.0199988     7 45.75038  4.532045 34.69352 0.6710526 15.70320 0.1887719             3  0.31959755
##   STD_CHL_log10 Watermass L10CHLsurvey CHLsurvanom L10CHLsurvclim      FCPI
## 1    0.02840480         3   -0.7665744  -1.2941501     -0.2932137 0.1824324
## 2    0.06258219         3    0.3587903  -1.5384094      0.6951271 0.2905405
## 3    0.02031565         1    0.4023360  -1.5200367      0.7013794 0.2770270
## 4    0.05416992         1    0.2798753  -1.5658311      0.7424677 0.2567568
## 5    0.02248318         2    0.5573132  -0.8559388      0.8132451 0.2905405
## 6    0.04746003         2    0.5720522  -0.3275567      0.7162526 0.2364865
```

```
summary(sosh.data)
```

```
##    SOSHcount         Binarea       Month       Latitude       DistCoast          Dist200           DepCI        
##  Min.   :  0.000   Min.   :0.0122   6 :512   Min.   :39.25   Min.   :  1.974   Min.   : 0.01411   Min.   :0.01885  
##  1st Qu.:  0.000   1st Qu.:0.5100   7 :561   1st Qu.:41.25   1st Qu.: 11.506   1st Qu.: 7.87697   1st Qu.:0.23184  
##  Median :  0.000   Median :1.0200   9 :569   Median :44.00   Median : 22.894   Median :17.27896   Median :0.33641  
##  Mean   :  3.879   Mean   :0.8249   10:538   Mean   :43.59   Mean   : 33.173   Mean   :19.79308   Mean   :0.42077  
##  3rd Qu.:  0.000   3rd Qu.:1.0200            3rd Qu.:45.55   3rd Qu.: 50.999   3rd Qu.:29.18043   3rd Qu.:0.57890  
##  Max.   :595.000   Max.   :1.0200            Max.   :47.05   Max.   :125.649   Max.   :80.15171   Max.   :1.00000  
##     SSTmean          STD_SST        MEAN_Beaufort    L10CHLproxy        STD_CHL_log10       Watermass    
##  Min.   : 8.371   Min.   :0.0000   Min.   :0.000   Min.   :-0.60703   Min.   :0.00000   Min.   :1.000  
##  1st Qu.:12.785   1st Qu.:0.1609   1st Qu.:2.000   1st Qu.:-0.04165   1st Qu.:0.02528   1st Qu.:1.000  
##  Median :13.956   Median :0.1940   Median :2.729   Median : 0.17588   Median :0.04042   Median :3.000  
##  Mean   :13.705   Mean   :0.2534   Mean   :2.575   Mean   : 0.14823   Mean   :0.05143   Mean   :2.444  
##  3rd Qu.:14.861   3rd Qu.:0.2822   3rd Qu.:3.037   3rd Qu.: 0.32251   3rd Qu.:0.06532   3rd Qu.:3.000  
##  Max.   :16.916   Max.   :1.3345   Max.   :6.000   Max.   : 1.06842   Max.   :0.39497   Max.   :4.000  
##   L10CHLsurvey       CHLsurvanom       L10CHLsurvclim          FCPI        
##  Min.   :-0.7825   Min.   :-2.2561   Min.   :-0.53229   Min.   :0.03378  
##  1st Qu.:-0.1748   1st Qu.:-0.8558   1st Qu.:-0.00885   1st Qu.:0.17568  
##  Median : 0.1007   Median :-0.3395   Median : 0.32011   Median :0.21622  
##  Mean   : 0.1674   Mean   :-0.3365   Mean   : 0.32061   Mean   :0.21730  
##  3rd Qu.: 0.4927   3rd Qu.: 0.2051   3rd Qu.: 0.64963   3rd Qu.:0.25676  
##  Max.   : 1.4523   Max.   : 1.8245   Max.   : 1.13857   Max.   :0.31757  
```

3. Set up parallel processing

```
nCores <- detectCores()
gamlssCl <- makeCluster(nCores)
registerDoParallel(gamlssCl)
```

4. Create geographic and oceanographic models by month using canonical discrete distributions WITHOUT home range variable:

```
# Wrapper for gamlss. Error handling and refitting.
N_REFIT <- 5
try.fit <- function(formula, data, family, n_refit = N_REFIT) {
  # Try to fit model
  model <- try(gamlss(formula, data = data, family = family))
  fitAttempts <- 1
```

```
  # Keep refitting model until...
  while(fitAttempts <= n_refit &&              # ... we run out of attempts,
        class(model) != 'try-error' &&         # get an error,
        !getElement(model, 'converged')) {     # or model converges
    model <- try(refit(model))
    fitAttempts <- fitAttempts + 1
  }
  # Return fitting results. Hopefully it's a fitted model, may be a try-error
  model
}

# Create models using canonical discrete distributions.
# SOSHcount as function of FCPI, Dist200, SSTmean (all three cubic splines) with Month as random variable
# and log(Binarea) as offset
# Create models using canonical discrete distributions for each month.
# Geographic + oceanographic models
# geo = SOSHcount ~  cs(Latitude)+cs(DistCoast)+cs(Dist200)+cs(DepCI) + offset(log(Binarea))
# ocean = SOSHcount ~ cs(SSTmean)+cs(STD_SST)+cs(MEAN_Beaufort)+cs(L10CHLproxy)+cs(STD_CHL_log10)+Watermass+cs(L10CHLsurv
ey)+
#   cs(CHLsurvanom)+cs(L10CHLsurvclim)+cs(FCPI) + offset(log(Binarea))
discrete.dist <- c('PO', 'NBI', 'NBII', 'DEL', 'PIG', 'SI', 'SICHEL', 'ZIP', 'ZIP2')
months <- c(6, 7, 9, 10)
# For each discrete distribution...
monthly.geo.ocean.models <- foreach(dist = discrete.dist,
                                    .packages = c('foreach',
                                                  'doParallel',
                                                  'gamlss',
                                                  'dplyr')) %dopar% {
  # For each month's survey...
  foreach(month = months) %do% {
    print(sprintf('Distribution %s, month %i', dist, month))

    # Filter SOSH data to the month of interest
    month.data <- filter(sosh.data, Month == month)
    if(nrow(month.data) == 0) stop(sprintf('No data in month %i', month))

    # Fit geographic model
    print('Fitting geographic model')
```

```
    geo.model <- try.fit(SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) + offset(log(Binarea)),
                         data = month.data,
                         family = get('dist'))

    # Fit oceanographic model
    print('Fitting oceanographic model')
    ocean.model <- try.fit(SOSHcount ~ cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                                cs(STD_CHL_log10) + as.factor(Watermass) + cs(L10CHLsurvey) + cs(CHLsurvanom) +
                                cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                           data = month.data,
                           family = get('dist'))

    list(dist = dist,
         month = month,
         geo.model = geo.model,
         ocean.model = ocean.model)
  }
}

stopCluster(gamlssCl)
```

5. Summarize model results for ranking (parameters, convergence, EDF, AIC)

```
model.rankings <- foreach(dist.list = monthly.geo.ocean.models, .combine = rbind) %do% {
  foreach(models = dist.list, .combine = rbind) %do% {
    data.frame(Distribution = models$dist,
               Month = models$month,
               GeoConverged = tryCatch(getElement(models$geo.model, 'converged'), error = function(e) NA),
               GeoEDF = tryCatch(extractAIC(models$geo.model)[1], error = function(e) NA),
               GeoAIC = tryCatch(extractAIC(models$geo.model)[2], error = function(e) NA),
               OceanConverged = tryCatch(getElement(models$ocean.model, 'converged'), error = function(e) NA),
               OceanEDF = tryCatch(extractAIC(models$ocean.model)[1], error = function(e) NA),
               OceanAIC = tryCatch(extractAIC(models$ocean.model)[2], error = function(e) NA))
  }
}
model.rankings
```

```
##    Distribution Month GeoConverged   GeoEDF     GeoAIC OceanConverged OceanEDF  OceanAIC
## 1            PO     6         TRUE 17.37874 14429.1667           TRUE 39.99704 8720.4084
## 2            PO     7         TRUE 17.22401  2891.0454           TRUE 40.00081 2206.1990
## 3            PO     9           NA       NA         NA           TRUE 39.99669 2401.9413
## 4            PO    10           NA       NA         NA           TRUE 40.00109 1943.9565
## 5           NBI     6         TRUE 19.76089  1597.5923           TRUE 41.00034 1559.7594
## 6           NBI     7         TRUE 19.64882   854.0322           TRUE 41.11919  877.4572
## 7           NBI     9           NA       NA         NA           TRUE 41.00053 1373.9162
## 8           NBI    10         TRUE 19.58401   909.4861           TRUE 41.00000  893.7608
## 9          NBII     6         TRUE 19.57903  1641.2317           TRUE 41.00221 1630.9974
## 10         NBII     7         TRUE 19.35685   912.5372           TRUE 40.99794  952.9774
## 11         NBII     9         TRUE 20.09472  1389.9298           TRUE 41.00057 1431.6700
## 12         NBII    10         TRUE 19.07953   933.8088           TRUE 40.99842  942.9446
## 13          DEL     6           NA       NA         NA             NA       NA        NA
## 14          DEL     7         TRUE 19.65512   805.9848           TRUE 42.00058  848.9592
## 15          DEL     9         TRUE 20.45498  1317.7136           TRUE 41.99937 1371.0198
## 16          DEL    10         TRUE 19.89557   864.0192           TRUE 42.00111  872.7358
## 17          PIG     6         TRUE 19.54640  1578.3406           TRUE 41.00224 1547.9482
## 18          PIG     7         TRUE 19.31741   812.3070           TRUE 40.99844  846.7177
## 19          PIG     9         TRUE 19.86607  1306.4611           TRUE 41.00071 1356.8089
## 20          PIG    10         TRUE 18.99260   866.9973           TRUE 40.99824  876.4088
## 21           SI     6         TRUE 20.52629  1580.0869           TRUE 41.99769 1546.8304
## 22           SI     7         TRUE 20.37716   797.8217           TRUE 41.99955  844.9290
## 23           SI     9         TRUE 21.06846  1307.0433           TRUE 41.99905 1357.4700
## 24           SI    10         TRUE 20.06931   867.2903           TRUE 42.00210  882.2266
## 25       SICHEL     6         TRUE 20.54553  1580.4882           TRUE 41.99817 1546.7936
## 26       SICHEL     7         TRUE 20.36798   797.8504           TRUE 41.99945  845.1267
## 27       SICHEL     9         TRUE 21.07391  1307.0165           TRUE 41.99906 1357.5018
## 28       SICHEL    10         TRUE 20.07979   866.9274           TRUE 42.00003  879.3872
## 29          ZIP     6           NA       NA         NA             NA       NA        NA
## 30          ZIP     7           NA       NA         NA             NA       NA        NA
## 31          ZIP     9           NA       NA         NA           TRUE 41.00300 2205.6094
## 32          ZIP    10           NA       NA         NA             NA       NA        NA
## 33         ZIP2     6           NA       NA         NA             NA       NA        NA
## 34         ZIP2     7           NA       NA         NA             NA       NA        NA
## 35         ZIP2     9           NA       NA         NA             NA       NA        NA
## 36         ZIP2    10           NA       NA         NA             NA       NA        NA
```

6. SI, SICHEL, and PIG distributions perform the best (which makes sense, they're all closely related), but only SI and SICHEL are in each month's top 4 distributions for both geographic and oceanographic models. SI has the slightly greater mean AIC weight than SICHEL, so we'll proceed using SI.

```
# Utility function for calculating AIC weight. vAIC is a vector of AIC values
calcAICw <- function(vAIC) {
  deltaAIC <- vAIC - min(vAIC, na.rm = TRUE)
  relLikelihood <- exp(-0.5 * deltaAIC)
  normalizingFactor <- sum(relLikelihood, na.rm = TRUE)
  relLikelihood / normalizingFactor
}

# Sort models by month and quality of fit
monthly.geo.ranks <- model.rankings %>%
  group_by(Month) %>%
  mutate(GeoDeltaAIC = GeoAIC - min(GeoAIC, na.rm = TRUE),
         GeoAICw = calcAICw(GeoAIC) %>% round(3)) %>%
  arrange(-GeoAICw) %>%
  slice(1:4) %>%
  ungroup %>%
  select(Distribution:GeoAICw)
monthly.ocean.ranks <- model.rankings %>%
  group_by(Month) %>%
  mutate(OceanDeltaAIC = OceanAIC - min(OceanAIC, na.rm = TRUE),
         OceanAICw = calcAICw(OceanAIC) %>% round(3)) %>%
  arrange(-OceanAICw) %>%
  slice(1:4) %>%
  ungroup %>%
  select(Distribution, Month, OceanConverged:OceanAICw)

# Identify best available distribution by product of AIC weight across months
monthly.geo.ranks %>%
  group_by(Distribution) %>%
  summarize(N = n(),
            meanAICw = mean(GeoAICw)) %>%
  ungroup %>%
  arrange(-N, -meanAICw)
```

```
## # A tibble: 5 x 3
##   Distribution     N  meanAICw
##         <fctr> <int>     <dbl>
## 1           SI     4 0.2880000
## 2       SICHEL     4 0.2822500
## 3          PIG     3 0.3676667
## 4          DEL     3 0.2046667
## 5           PO     2 0.0000000
```

```
monthly.ocean.ranks %>%
  group_by(Distribution) %>%
  summarize(N = n(),
            meanAICw = mean(OceanAICw)) %>%
  ungroup %>%
  arrange(-N, -meanAICw)
```

```
## # A tibble: 6 x 3
##   Distribution     N meanAICw
##         <fctr> <int>    <dbl>
## 1           SI     4  0.27425
## 2       SICHEL     4  0.27100
## 3          PIG     4  0.23300
## 4          DEL     2  0.44250
## 5          NBI     1  0.00100
## 6           PO     1  0.00000
```

7. Model culling

```
# We see SI and SICHEL are the only two distributions to show up in both models' top 4 across months
# SI has the slightly higher mean AIC weight, so we'll go with that.

# Cull models using selected distribution (SI)

# Recursively drop parameters by largest decrease in AIC until model degrades
cull.model <- function(model) {
  aic <- extractAIC(model)[2]
```

```
  # Re-fit model to n-1 terms
  model.terms <- model %>% formula %>% terms
  term.labels <- attr(model.terms, 'term.labels')
  submodels <- foreach(i = seq(term.labels)) %do% {
    dropped <- term.labels[i]
    submodel <- try(update(model, reformulate(sprintf('. - %s', dropped))))
    list(dropped = dropped,
         submodel = submodel)
  }

  # Which submodels converged? Which submodel has the greatest decrease in AIC?
  drop.results <- foreach(i = seq(submodels), .combine = rbind) %do% {
    tryCatch(with(submodels[[i]], data.frame(dropped = dropped,
                                              i = i,
                                              converged = submodel$converged,
                                              AIC = extractAIC(submodel)[2],
                                              deltaAIC = extractAIC(submodel)[2] - aic)),
             error = function(e) data.frame(dropped = NA,
                                             i = NA,
                                             converged = NA,
                                             AIC = NA,
                                             deltaAIC = NA))
  } %>%
    filter(converged) %>%
    arrange(deltaAIC)

  # If no submodels converge or if no submodels are an improvement, return the original model
  if(nrow(drop.results) == 0 || min(drop.results$deltaAIC, na.rm = TRUE) > 0) {
    return(model)
  } else {
    # Otherwise, repeat the process on the best submodel
    best.submodel <- submodels[[drop.results$i[1]]]$submodel
    return(cull.model(best.submodel))
  }
}

# Month 6
```

```
month6 <- filter(sosh.data, Month == 6)

# Geo
geo6global <- gamlss(SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) + offset(log(Binarea)),
                     data = month6,
                     family = SI,
                     control = gamlss.control(n.cyc = 100))
geo6dropterm <- dropterm(geo6global, test = 'Chisq')
geo6culled <- cull.model(geo6global)
geo6dropped <- setdiff(terms(geo6global) %>% attr('term.labels'), terms(geo6culled) %>% attr('term.labels'))

# Ocean
ocean6global <- gamlss(SOSHcount ~ cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                         cs(STD_CHL_log10) + as.factor(Watermass) + cs(L10CHLsurvey) + cs(CHLsurvanom) +
                         cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                       data = month6,
                       family = SI,
                       control = gamlss.control(n.cyc = 100))
ocean6dropterm <- dropterm(ocean6global, test = 'Chisq')
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
ocean6culled <- cull.model(ocean6global)
```

```
## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged
```

```r
ocean6dropped <- setdiff(terms(ocean6global) %>% attr('term.labels'), terms(ocean6culled) %>% attr('term.labels'))

# Month 7
month7 <- filter(sosh.data, Month == 7)

# Geo
geo7global <- gamlss(SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) + offset(log(Binarea)),
                     data = month7,
                     family = SI,
                     control = gamlss.control(n.cyc = 100))
geo7dropterm <- dropterm(geo7global, test = 'Chisq')
geo7culled <- cull.model(geo7global)
geo7dropped <- setdiff(terms(geo7global) %>% attr('term.labels'), terms(geo7culled) %>% attr('term.labels'))

# Ocean
ocean7global <- try(gamlss(SOSHcount ~ cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                             cs(STD_CHL_log10) + as.factor(Watermass) + cs(L10CHLsurvey) + cs(CHLsurvanom) +
                             cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                           data = month7,
                           family = SI,
                           control = gamlss.control(n.cyc = 100)))
ocean7dropterm <- dropterm(ocean7global, test = 'Chisq')
ocean7culled <- cull.model(ocean7global)
```

```
## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged
```

```
ocean7dropped <- setdiff(terms(ocean7global) %>% attr('term.labels'), terms(ocean7culled) %>% attr('term.labels'))

# Month 9
month9 <- filter(sosh.data, Month == 9)

# Geo
# For some reason month 9 geo model doesn't converge reliably
geo9global <- try.fit(SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) + offset(log(Binarea)),
                      data = month9,
                      family = SI)
```

```
## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations
```

```
## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
```

```
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations
```

```
## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations
```

```
## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations

## Warning in additive.fit(x = X, y = wv, w = wt * w, s = s, who = who, smooth.frame, : additive.fit convergence not
## obtained in 30 iterations
```

```
## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged
```

```
geo9dropterm <- dropterm(geo9global, test = 'Chisq')
```

```
## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'
```

```
## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'

## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'

## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'
```

```
geo9culled <- cull.model(geo9global)
```

```
## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'

## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'

## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'

## Warning in is.na(data): is.na() applied to non-(list or vector) of type 'closure'
```

```
geo9dropped <- setdiff(terms(geo9global) %>% attr('term.labels'), terms(geo9culled) %>% attr('term.labels'))

# Ocean
ocean9global <- gamlss(SOSHcount ~ cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                         cs(STD_CHL_log10) + as.factor(Watermass) + cs(L10CHLsurvey) + cs(CHLsurvanom) +
                         cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                       data = month9,
                       family = SI,
                       control = gamlss.control(n.cyc = 100))
ocean9dropterm <- dropterm(ocean9global, test = 'Chisq')
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
ocean9culled <- cull.model(ocean9global)
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged
```

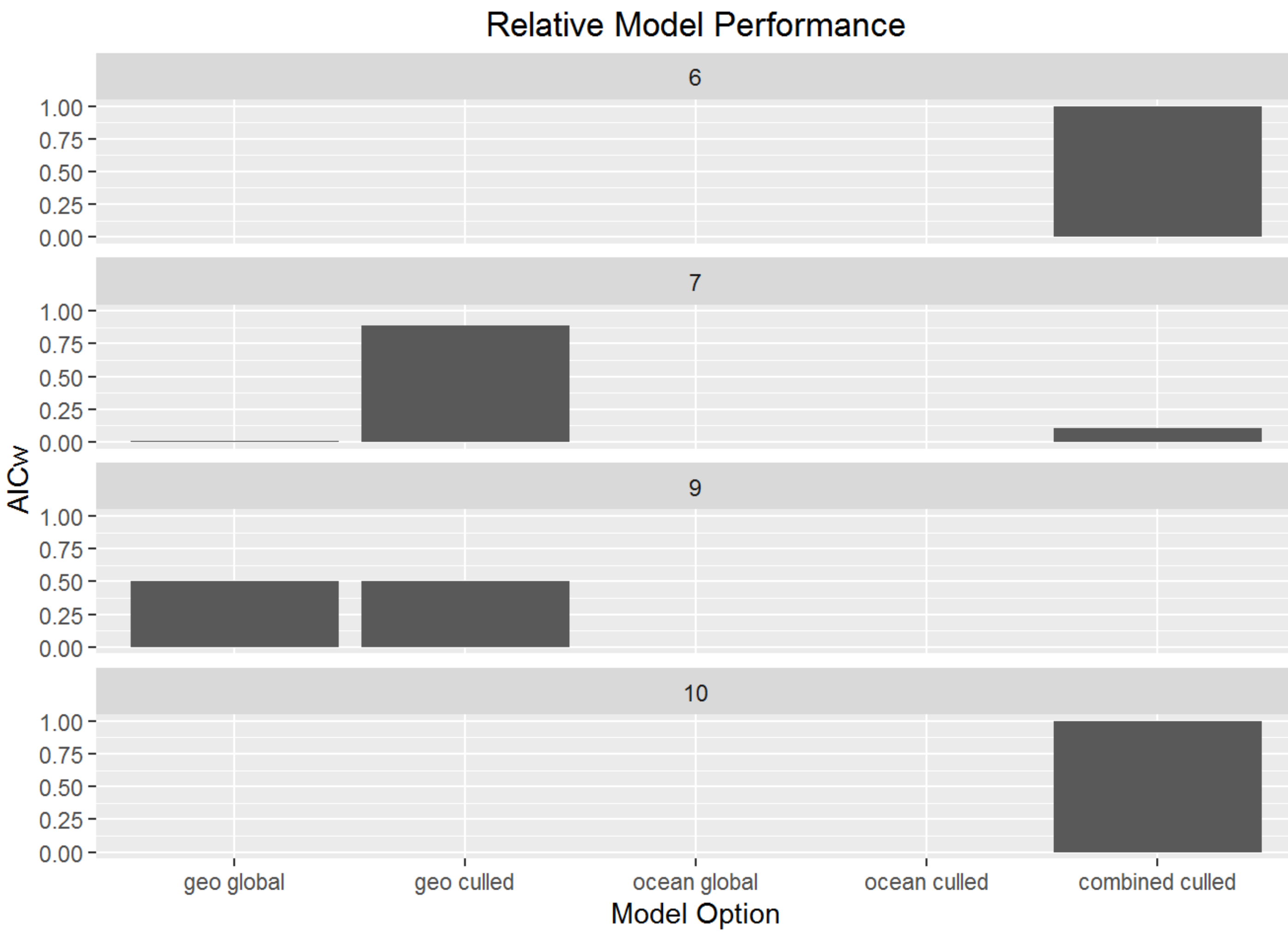
```
ocean9dropped <- setdiff(terms(ocean9global) %>% attr('term.labels'), terms(ocean9culled) %>% attr('term.labels'))

# Month 10
month10 <- filter(sosh.data, Month == 10)

# Geo
geo10global <- gamlss(SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) + offset(log(Binarea)),
                     data = month10,
                     family = SI,
                     control = gamlss.control(n.cyc = 100))
geo10dropterm <- dropterm(geo10global, test = 'Chisq')
geo10culled <- cull.model(geo10global)
geo10dropped <- setdiff(terms(geo10global) %>% attr('term.labels'), terms(geo10culled) %>% attr('term.labels'))

# Ocean
ocean10global <- try(gamlss(SOSHcount ~ cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                             cs(STD_CHL_log10) + as.factor(Watermass) + cs(L10CHLsurvey) + cs(CHLsurvanom) +
                             cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                           data = month10,
                           family = SI,
                           control = gamlss.control(n.cyc = 100)))
ocean10dropterm <- dropterm(ocean10global, test = 'Chisq')
ocean10culled <- cull.model(ocean10global)
ocean10dropped <- setdiff(terms(ocean10global) %>% attr('term.labels'), terms(ocean10culled) %>% attr('term.labels'))
```

8. Per-month combined models

```
combined6culled <- gamlss(formula = SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) + cs(DepCI) +
                      cs(SSTmean) + cs(STD_SST) + cs(MEAN_Beaufort) + cs(L10CHLproxy) +
                      as.factor(Watermass) + cs(L10CHLsurvey) + cs(L10CHLsurvclim) +
                      cs(FCPI) + offset(log(Binarea)),
                    family = SI,
                    data = month6,
                    control = gamlss.control(n.cyc = 100))

combined7culled <- gamlss(formula = SOSHcount ~ cs(Latitude) + cs(DistCoast) +
                      cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                    family = SI,
                    data = month7,
                    control = gamlss.control(n.cyc = 100))

# Combined model fails for month 9
# combined9culled <- gamlss(formula = SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) +
#                       cs(SSTmean) + cs(MEAN_Beaufort) +
#                       cs(STD_CHL_log10) + cs(CHLsurvanom) + cs(L10CHLsurvclim) +
#                       cs(FCPI) + offset(log(Binarea)),
#                     family = SI,
#                     data = month9,
#                     control = gamlss.control(n.cyc = 100))

combined10culled <- gamlss(formula = SOSHcount ~ cs(Latitude) + cs(DistCoast) + cs(Dist200) +
                       cs(SSTmean) + cs(STD_SST) + cs(L10CHLproxy) +
                       as.factor(Watermass) + cs(L10CHLsurvclim) + cs(FCPI) + offset(log(Binarea)),
                    family = SI,
                    data = month10,
                    control = gamlss.control(n.cyc = 100))
```

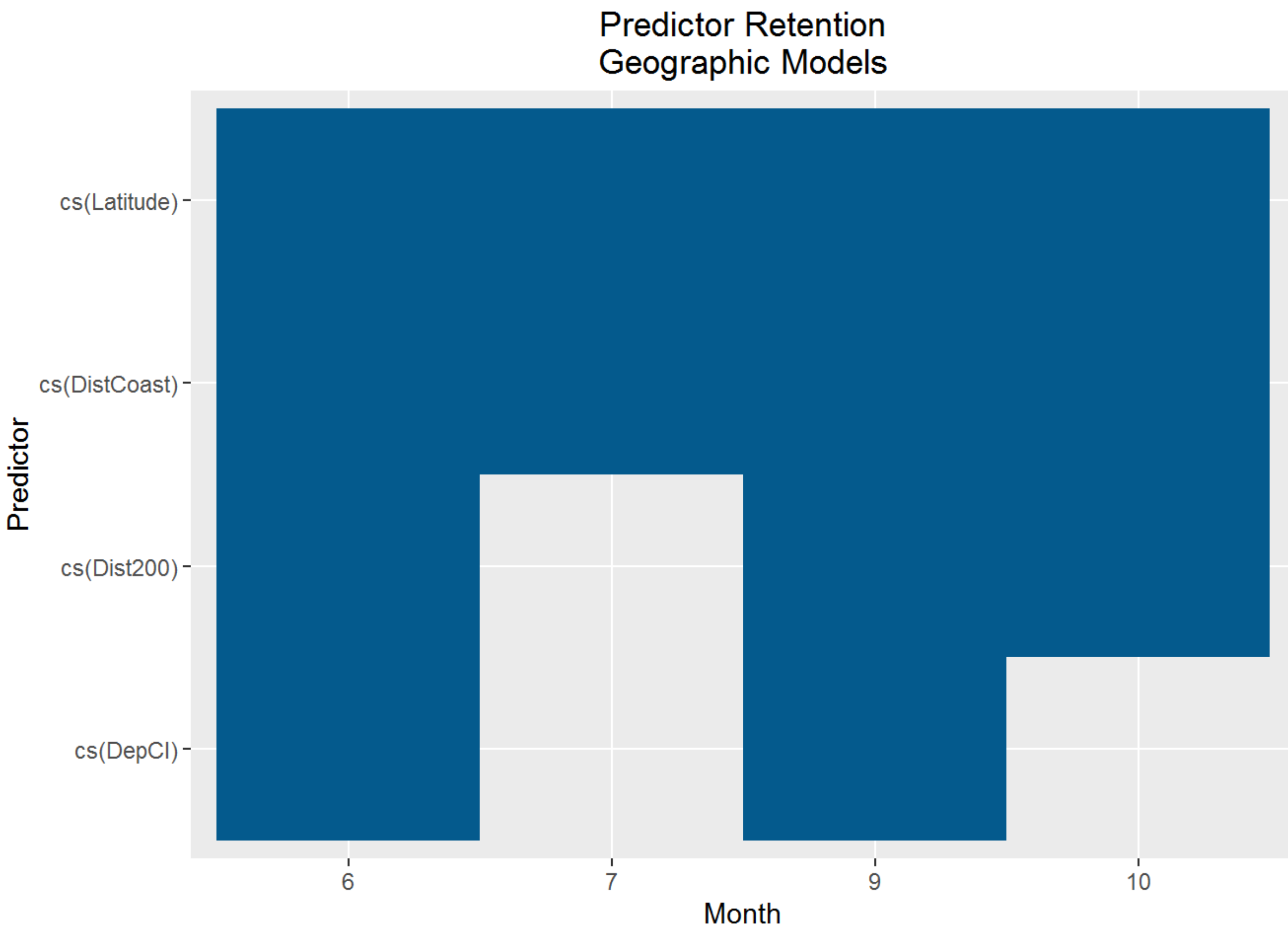
9. Model analysis

```
model.options <- foreach(type = c('geo', 'ocean', 'combined'), .combine = rbind) %do% {
  foreach(month = c(6, 7, 9, 10), .combine = rbind) %do% {
    foreach(scope = c('global', 'culled'), .combine = rbind) %do% {
      if(type == 'combined' && scope == 'global')
        return(NULL)
      model <- try(get(paste(type, month, scope, sep = '')))
      if(class(model)[1] == 'try-error')
        return(NULL)
      aic <- extractAIC(model)[2]
      predictors <- model %>% formula %>% terms %>% attr('term.labels') %>% paste(collapse = ', ')
      data.frame(type = type,
                 month = month,
                 scope = scope,
                 AIC = aic,
                 predictors = predictors)
    }
  }
} %>%
  group_by(month) %>%
  mutate(AICw = calcAICw(AIC)) %>%
  ungroup %>%
  arrange(month, type, scope)

ggplot(model.options,
       aes(x = paste(type, scope),
           y = AICw)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~ month,
             nrow = 4) +
  scale_x_discrete('Model Option',
                   limits = c('geo global', 'geo culled',
                              'ocean global', 'ocean culled',
                              'combined culled')) +
  ggtitle('Relative Model Performance')
```

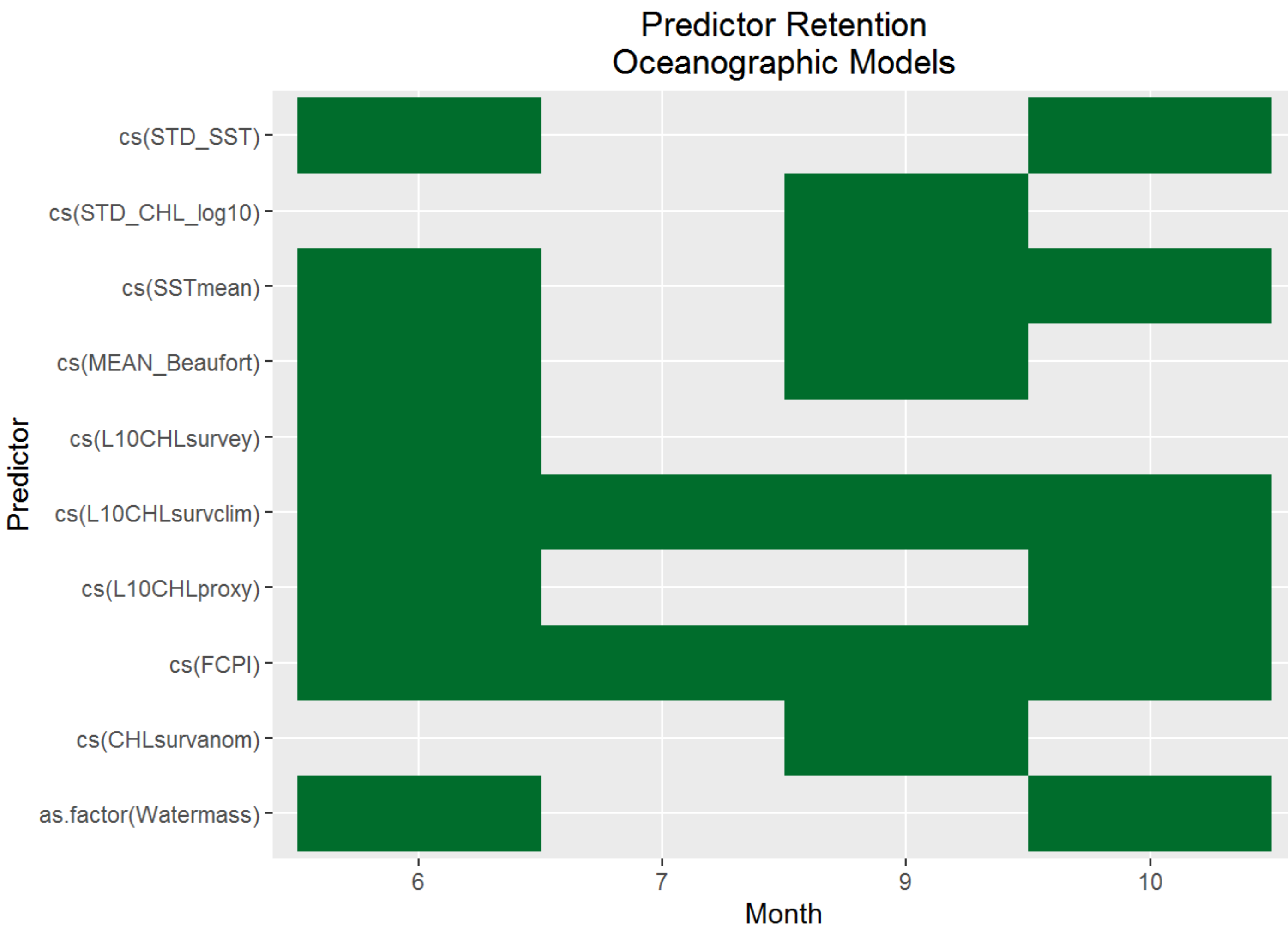


Relative Model Performance
6
7
9
10
1.00
0.75
0.50
0.25
0.00
AICw
geo global
geo culled
ocean global
ocean culled
combined culled
Model Option

```r
predictor.comparison <- foreach(type = c('geo', 'ocean', 'combined'), .combine = rbind) %do% {
  foreach(month = c(6, 7, 9, 10), .combine = rbind) %do% {
    foreach(scope = c('global', 'culled'), .combine = rbind) %do% {
      if(type == 'combined' && scope == 'global')
        return(NULL)
      model <- try(get(paste(type, month, scope, sep = '')))
      if(class(model)[1] == 'try-error')
        return(NULL)
      predictors <- model %>% formula %>% terms %>% attr('term.labels')
      data.frame(type = type,
                 month = month,
                 scope = scope,
                 predictor = predictors)
    }
  }
}

predictor.comparison %>%
  filter(type == 'geo', scope == 'culled') %>%
  ggplot(aes(x = factor(month),
             y = predictor)) +
  geom_tile(fill = '#045a8d') +
  labs(title = 'Predictor Retention\nGeographic Models',
       x = 'Month',
       y = 'Predictor')
```




```
predictor.comparison %>%
  filter(type == 'ocean', scope == 'culled') %>%
  ggplot(aes(x = factor(month),
             y = predictor)) +
  geom_tile(fill = '#006d2c') +
  labs(title = 'Predictor Retention\nOceanographic Models',
       x = 'Month',
       y = 'Predictor')
```



Predictor Retention
Oceanographic Models
cs(STD_SST)
cs(STD_CHL_log10)
cs(SSTmean)
cs(MEAN_Beaufort)
cs(L10CHLsurvey)
cs(L10CHLsurvclim)
cs(L10CHLproxy)
cs(FCPI)
cs(CHLsurvanom)
as.factor(Watermass)
Predictor
6
7
9
10
Month