Single Species Example from Gulf of Alaska with Catchability Covariates RACE Bottom Trawl Survey

Dr. Curry J. Cunningham

January 23, 2018

Contents

1 Purpose

The purpose of this document is to describe how to generate a model-based index of abundance unsing the spatio-temporal delta-GLMM in the VAST package, while incorporating **catchability covariates**.

1.1 Background

Catchability covariates attempt to explain residual variance in either the **encounter probability** or **positive catch rate** components of the delta model, given conditions influencing observation uncertainty at the time of sampling. In this way **catchability** covariates which are represented at the haul level, may be contrasted with **density** covariates which attempt to explain the underlying spatial distribution of the observed species and are represented at the knot level.

1.1.1 Example notation

Within the delta-model, the linear predictor for encounter probability can be written as:

$$p_1(i) = \beta_1(c_i, t_i) + \sum_{f=1}^{n_{\omega 1}} L_{\omega 1}(c_i, f)\omega 1(s_i, f) + \sum_{f=1}^{n_{\epsilon 1}} L_{\epsilon 1}(c_i, f)\epsilon_1(s_i, f, t_i)$$
$$+ \sum_{f=1}^{n_{\delta 1}} L_{\delta 1}(v_i, f)\delta_1(v_i, f) + \sum_{p=1}^{n_p} \gamma_1(c_i, t_i, p)X(x_i, t_i, p) + \sum_{k=1}^{n_k} \lambda(k)Q(i, k)$$

where $p_1(i)$ is the predicotr for observation i, $Q(i,k)$ are measured catchability covariates that explain variation in catchability and $\lambda_1(k)$ is the estimated impact of catchability covariates for this linear predictor, and $X(x_i, t_i, p)$ are measured **density** covariates that explain variation in density and $\gamma_1(c_i, t_i, p)$ is the estimated impact of density covariates.

1.2 Hypotheses

We will be testing two hypotheses: * Tow duration (measured in hours) influences the catchability of species in the survey, under the assumption that longer tows should increase encounter probability and/or the positive catch rate if individuals outswim the survey net but tire over time. + Note: Trawl distance (positively correlated with encounter probability) is already accounted for when effort is calculated as area swept (square km). * Measured gear temperature influences catchability, due to potential shifts in the vertical distribution of species.

Specifics of this Example:

- Uses RACE bottom trawl survey data.
  - Data are available from the /data folder
- Single species implementation.
- Gulf of Alaska survey data.
- Haul-level catchability covariates: (1) tow duration (hours), (2) gear temperature.

---

2 Setup

2.1 Install required packages

```
devtools::install_github("nwfsc-assess/geostatistical_delta-GLMM")
devtools::install_github("james-thorson/VAST")
devtools::install_github("james-thorson/utilities")
```

2.2 Load required packages

```
require(dplyr)
require(VAST)
require(TMB)
require(FishData)
# require(tidyverse)
```

2.3 Setup model

2.3.1 Define species of interest (based on species code) and survey name.

Species are selected by defining the vector `species.codes` in combination with the `combineSpecies` variable. While most species will have a single species code, there are some examples (i.e. GOA Dusky Rockfish) that require multiple species codes to be combined for a single species index. In this later case `combineSpecies = FALSE` would be specified.

Here are some examples to choose from:

number	name	species.code	include	survey	Region
1	Pacific ocean perch	30060	Y	GOA	Gulf_of_Alaska
2	Pacific ocean perch	30060	Y	AI	Aleutian_Islands
3	Walleye pollock	21740	Y	GOA	Gulf_of_Alaska
4	Walleye pollock	21740	Y	AI	Aleutian_Islands
5	Pacific cod	21720	Y	GOA	Gulf_of_Alaska
6	Pacific cod	21720	N	EBS_SHELF	Eastern_Bering_Sea
7	Pacific cod	21720	Y	AI	Aleutian_Islands
8	Northern rockfish	30420	Y	GOA	Gulf_of_Alaska
9	Northern rockfish	30420	Y	AI	Aleutian_Islands
10	Dover sole	10180	Y	GOA	Gulf_of_Alaska
11	Big skate	420	Y	GOA	Gulf_of_Alaska
12	Atka mackerel	21921	Y	AI	Aleutian_Islands
13	Harlequin rockfish	30535	Y	GOA	Gulf_of_Alaska
14	Arrowtooth flounder	10110	Y	GOA	Gulf_of_Alaska
15	Arrowtooth flounder	10110	Y	EBS_SHELF	Eastern_Bering_Sea
16	Spiny dogfish	310	Y	GOA	Gulf_of_Alaska

Example: Pacific Cod in the Gulf of Alaska

```
species.codes = c(21720)
survey = "GOA"
combineSpecies = FALSE
```

```
if(survey=="GOA") { Region = 'Gulf_of_Alaska' }
if(survey=="EBS_SHELF") { Region = "Eastern_Bering_Sea" }
if(survey=="AI") { Region = "Aleutian Islands" }
```

2.3.2 Observation reference location settings

```
lat_lon.def = "start"
```

2.3.3 Spatial settings

The following settings define the spatial resolution for the model (defined by number of knots `n_x`), and whether to use a grid or mesh approximation through the `Method` variable.

```
Method = c("Grid", "Mesh", "Spherical_mesh")[2]
grid_size_km = 25
n_x = c(100, 250, 500, 1000, 2000)[1]
Kmeans_Config = list( "randomseed"=1, "nstart"=100, "iter.max"=1e3 )

#Strata Limits
#Basic - Single Area
strata.limits = data.frame(STRATA = c("All_areas"))

#VAST Version - latest!
Version = "VAST_v4_0_0"
```

2.3.4 Model settings

```
bias.correct = FALSE

FieldConfig = c(Omega1 = 1, Epsilon1 = 1, Omega2 = 1, Epsilon2 = 1)
RhoConfig  = c(Beta1 = 0, Beta2 = 0, Epsilon1 = 0, Epsilon2 = 0)
OverdispersionConfig  = c(Delta1 = 0, Delta2 = 0)

#Observation Model
ObsModel = c(1,0)
```

2.3.5 Save settings

DateFile is the folder that will hold my model outputs.

```
DateFile = paste0(getwd(), "/VAST_output/")

#Create directory
dir.create(DateFile, recursive=TRUE)
```

2.4 Specify model outputs

The following settings define what types of output we want to calculate.

```r
Options = c(SD_site_density = 0, SD_site_logdensity = 0,
            Calculate_Range = 1, Calculate_evenness = 0, Calculate_effective_area = 1,
            Calculate_Cov_SE = 0, Calculate_Synchrony = 0,
            Calculate_Coherence = 0)
```

---

3 Prepare the data

- Note: This section can be replace by function `create_VAST_input()`, from `R/create-VAST-input.r`

3.1 Load RACE data

To create the input data files for VAST model, first we must load RACE survey data. Two data files are necessary **(1)** catch data and **(2)** haul data.

3.1.1 Load and join data

```
catch = readRDS("data/race_base_catch.rds")

haul = readRDS("data/race_base_haul.rds")
haul = haul[haul$ABUNDANCE_HAUL == "Y", ]

# Join datasets
catchhaul = right_join(x = catch, y = haul, by = c("HAULJOIN"))

# Add in zero observations for catch weight, for no
# catches.
catchhaul.2 = FishData::add_missing_zeros(data_frame = catchhaul,
    unique_sample_ID_colname = "HAULJOIN", sample_colname = "WEIGHT",
    species_colname = "SPECIES_CODE", species_subset = species.codes,
    if_multiple_records = "First", Method = "Fast")
```

```
## Species to include: 21720

## Number of samples to include for each species: 31306

## Finished processing for 21720
```

```
# Load and attach cruise info
cruise.info = read.csv("data/race_cruise_info.csv",
    header = TRUE, stringsAsFactors = FALSE)

catchhaul.3 = inner_join(x = catchhaul.2, y = cruise.info[,
    c("Cruise.Join.ID", "Year", "Survey")], by = c(CRUISEJOIN.x = "Cruise.Join.ID"))

# Limit to survey of interest
catchhaul.3 = catchhaul.3[catchhaul.3$Survey == survey,
    ]

# Aggregate multiple `species.codes`, if we are
# combining into a single index.
if (combineSpecies == TRUE) {
    catchhaul.4 = data.frame(catchhaul.3 %>% group_by(HAULJOIN) %>%
        mutate(WEIGHT = sum(WEIGHT, na.rm = TRUE)))
    # Since we have aggregated, only retain rows for
    # 1st listed species code
    catchhaul.5 = catchhaul.4[catchhaul.4$SPECIES_CODE ==
        species.codes[1], ]
} else {
    catchhaul.5 = catchhaul.3
}
```

Lets see what `catchhaul.5` contains. . . .

```
names(catchhaul.5)
```

```
##  [1] "CRUISEJOIN.x"        "HAULJOIN"
##  [3] "CATCHJOIN"           "REGION.x"
##  [5] "VESSEL.x"            "CRUISE.x"
##  [7] "HAUL.x"              "SPECIES_CODE"
##  [9] "WEIGHT"              "NUMBER_FISH"
## [11] "SUBSAMPLE_CODE"      "VOUCHER"
## [13] "AUDITJOIN.x"         "CRUISEJOIN.y"
## [15] "REGION.y"            "VESSEL.y"
## [17] "CRUISE.y"            "HAUL.y"
## [19] "HAUL_TYPE"           "PERFORMANCE"
## [21] "START_TIME"          "DURATION"
## [23] "DISTANCE_FISHED"     "NET_WIDTH"
## [25] "NET_MEASURED"        "NET_HEIGHT"
## [27] "STRATUM"             "START_LATITUDE"
## [29] "END_LATITUDE"        "START_LONGITUDE"
## [31] "END_LONGITUDE"       "STATIONID"
## [33] "GEAR_DEPTH"          "BOTTOM_DEPTH"
## [35] "BOTTOM_TYPE"         "SURFACE_TEMPERATURE"
## [37] "GEAR_TEMPERATURE"    "WIRE_LENGTH"
## [39] "GEAR"                "ACCESSORIES"
## [41] "SUBSAMPLE"           "ABUNDANCE_HAUL"
## [43] "AUDITJOIN.y"         "Year"
## [45] "Survey"
```

3.2 Standardize data

In order to standardize the survey catch data, we must calculate effort as area swept per tow.

3.2.1 Calculate effort

Input effort is in $kilometers^2$

```
catchhaul.5$effort = catchhaul.5$NET_WIDTH*catchhaul.5$DISTANCE_FISHED/1000
```

3.3 Add species names

First, we load the list describing both species names and `species.codes`

```
species.code.data = read.csv("data/race_species_codes.csv",
                             header=TRUE, stringsAsFactors=FALSE)

#Next, join this to our overall survey data list
load.data = merge(x=catchhaul.5, y=species.code.data[,c("Species.Code","Common.Name")],
                  by.x="SPECIES_CODE", by.y="Species.Code")
```

3.4 Build Data_Geostat

Now, we will create the list Data_Geostat which is the input for the VAST model.

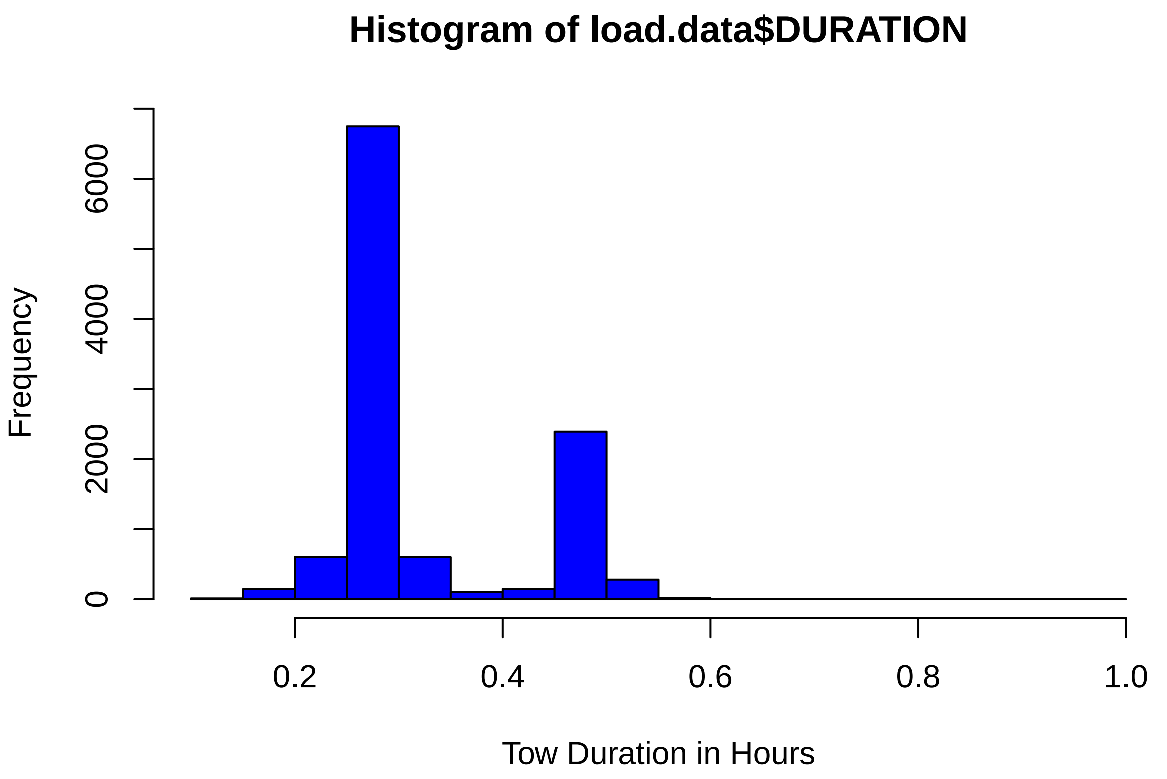
```
Data_Geostat = NULL

if(length(species.codes) > 1) {
  Data_Geostat$spp = load.data$Common.Name
}
Data_Geostat$Catch_KG = as.numeric(load.data$WEIGHT)
Data_Geostat$Year = as.integer(load.data$Year)
Data_Geostat$Vessel = "missing"
Data_Geostat$AreaSwept_km2 = as.numeric(load.data$effort)
Data_Geostat$Pass = 0
```

3.4.1 Add catchability covariates to Data_Geostat

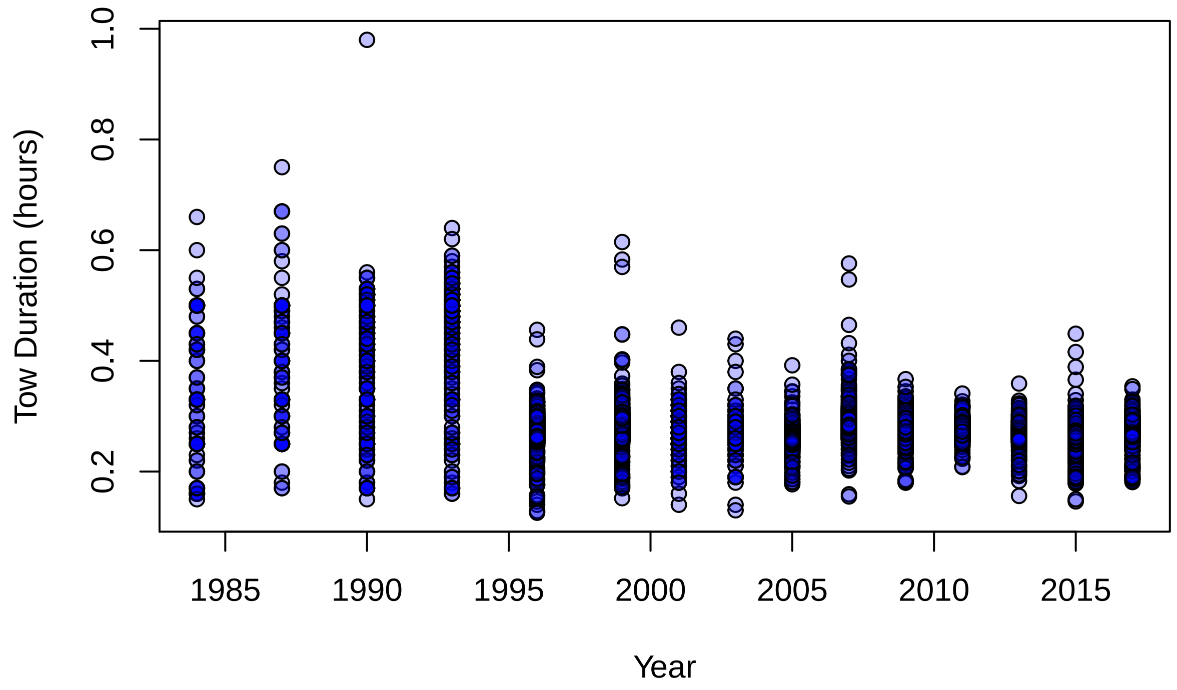
Here we can attach our two catchability covariates to our list of input data.

Hypothesis #1: Catchability is is influenced by tow duration in hours.

First, lets see what the distribution of tow durations look like...



So there seems to be a break down between 0.5 hour and 0.25 hour tows, but when did they occurr?

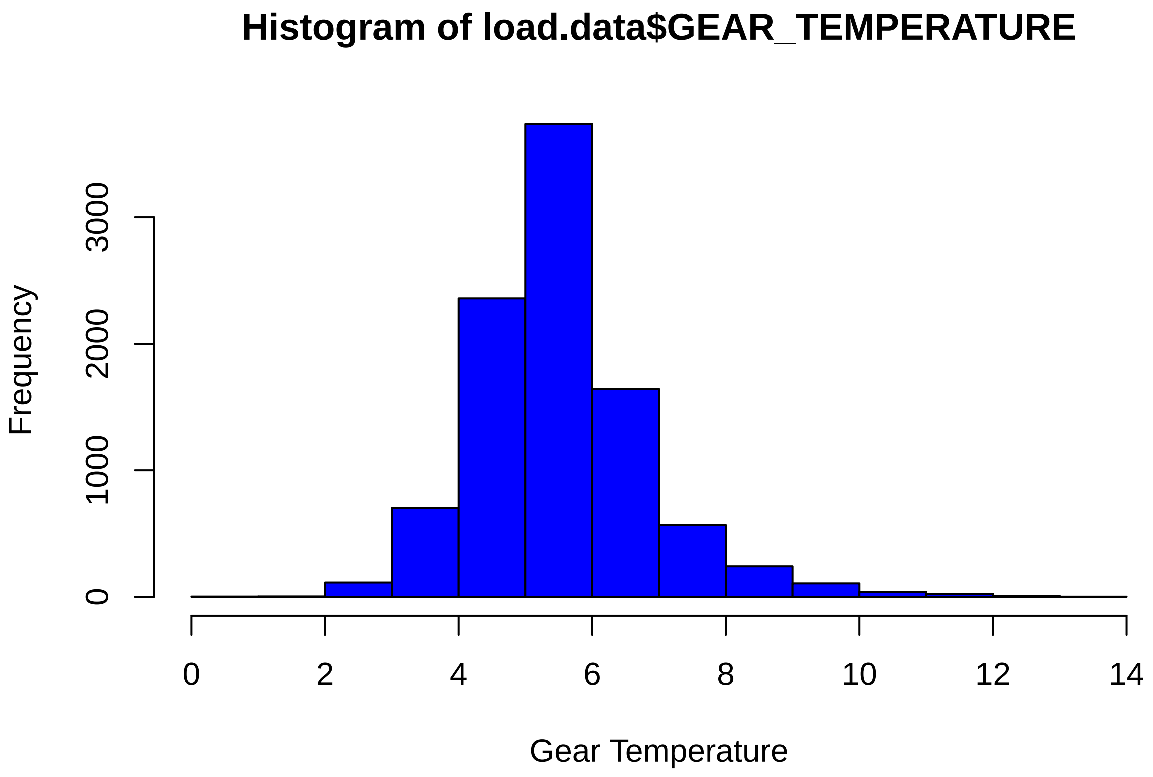


Ok lets attach these data to `Data_Geostat`

```
Data_Geostat$Duration <- as.numeric(load.data$DURATION)
```

Hypothesis #2: Catchability is is influenced by gear temperature.

Again, lets see what the distribution of gear temperatures look like…



And lets add this to `Data_Geostat`

```
Data_Geostat$Gear_Temperature <- as.numeric(load.data$GEAR_TEMPERATURE)
```

3.4.2 Define location of samples

Depending on `lat_lon.def` specification we will either use the **start**, **end**, or **mean** location recorded for a survey haul.

- Note: Using the starting location of each haul is probably best, as: `lat_lon.def="start"`.

```
if(lat_lon.def=="start") {
  Data_Geostat$Lat = load.data$START_LATITUDE
  Data_Geostat$Lon = load.data$START_LONGITUDE
}
if(lat_lon.def=="end") {
  Data_Geostat$Lat = load.data$END_LATITUDE
  Data_Geostat$Lon = load.data$END_LONGITUDE
}
if(lat_lon.def=="mean") {
  Data_Geostat$Lat = rowMeans(cbind(load.data$START_LATITUDE,
                                    load.data$END_LATITUDE), na.rm=TRUE)

  Data_Geostat$Lon = rowMeans(cbind(load.data$START_LONGITUDE,
                                    load.data$END_LONGITUDE), na.rm=TRUE)
}
```

Next, we ensure this `Data_Geostat` is a proper data frame.

```
Data_Geostat = data.frame(Data_Geostat)
```

To double check lets see how `Data_Geostat` looks…

```
kable(head(Data_Geostat))
```

Catch_KG	Year	Vessel	AreaSwept_km2	Pass	Duration	Gear_Temperature	Lat	Lon
190.1	2005	missing	0.028	0	0.281	NA	52.6	-170
30.9	2005	missing	0.020	0	0.252	4.9	52.6	-170
242.2	2005	missing	0.024	0	0.255	4.7	52.7	-169
39.9	2005	missing	0.021	0	0.266	5.2	53.2	-168
16.2	2005	missing	0.019	0	0.255	5.1	53.2	-168
61.2	2005	missing	0.021	0	0.252	4.9	53.1	-168

3.5 Limit `Data_Geostat` to only tows with recorded `Gear_Temperature`

Upon closer inspection you will notice that some tows did not have a recorded `Gear_Temperature`, appearing as an `NA`. It appears to be ~ 13.7%

```
nrow(Data_Geostat[is.na(Data_Geostat$Gear_Temperature),])/nrow(Data_Geostat)*100
```

```
## [1] 13.7
```

Given we want to compare models fit to the same data, lets remove these tows without `Gear_Temperature` observations.

```
Data_Geostat = Data_Geostat[!is.na(Data_Geostat$Gear_Temperature),]
```

3.6 Create the extrapolation grid

We also generate the extrapolation grid appropriate for a given region. For new regions, we use Region="Other".

- Note: We are not defining strata limits, but could do so based on latitude and longitude definitions.

```
Extrapolation_List = SpatialDeltaGLMM::Prepare_Extrapolation_Data_Fn(Region = Region,
    strata.limits = strata.limits)
```

3.7 Create spatial list

Next, generate the information used for conducting spatio-temporal parameter estimation, bundled in list `Spatial_List`.

```
Spatial_List = SpatialDeltaGLMM::Spatial_Information_Fn(grid_size_km = grid_size_km,
    n_x = n_x, Method = Method, Lon = Data_Geostat[,
        "Lon"], Lat = Data_Geostat[, "Lat"], Extrapolation_List = Extrapolation_List,
    randomseed = Kmeans_Config[["randomseed"]], nstart = Kmeans_Config[["nstart"]],
    iter.max = Kmeans_Config[["iter.max"]], DirPath = DateFile,
    Save_Results = TRUE)
```

3.8 Update `Data_Geostat` with knot references

We then associate each of our haul observations with its appropriate knot.

```
Data_Geostat = cbind(Data_Geostat, knot_i = Spatial_List$knot_i)
```

---

4 Build and run models

Here we are going to build **3** models.

- **Null Model** which does **not** estimate catchability covariates.
- **Model 1** testing Hypothesis #1 with **tow duration** as a catchability covariate.
- **Model 2** testing Hypothesis #2 with **temperature** as a catchability covariate.

4.1 Null Model

First, create a subdirectory for the **Null Model**

```
DateFile_null = paste0(DateFile,"Null/")
dir.create(DateFile_null)
```

Building and compiling:

- Note: in `Build_TMB_Fn()` whether to estimate **catchability** covariates is specified by the `Q_Config` argument, and whether to estimate **density** covariates by `CovConfig`.

```
# Build
if (length(species.codes) > 1 & combineSpecies == FALSE) {
    # MULTISPECIES
    TmbData_null = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = as.numeric(Data_Geostat[, "spp"]) - 1,
        b_i = Data_Geostat[, "Catch_KG"], a_i = Data_Geostat[,
            "AreaSwept_km2"], v_i = as.numeric(Data_Geostat[,
            "Vessel"]) - 1, s_i = Data_Geostat[, "knot_i"] -
            1, t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options)
} else {
    # SINGLE SPECIES
    TmbData_null = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = rep(0, nrow(Data_Geostat)), b_i = Data_Geostat[,
            "Catch_KG"], a_i = Data_Geostat[, "AreaSwept_km2"],
        v_i = as.numeric(Data_Geostat[, "Vessel"]) -
            1, s_i = Data_Geostat[, "knot_i"] - 1,
        t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options)
}

# Compile TMB object
TmbList_null = VAST::Build_TMB_Fn(TmbData = TmbData_null,
    RunDir = DateFile_null, Version = Version, RhoConfig = RhoConfig,
    loc_x = Spatial_List$loc_x, Method = Method, Q_Config = FALSE,
    CovConfig = FALSE)
Obj_null = TmbList_null[["Obj"]]
```

Fit VAST model to the data by optimizing the TMB function.

```
Opt_null = TMBhelper::Optimize(obj = Obj_null, lower = TmbList_null[["Lower"]],
                          upper = TmbList_null[["Upper"]], getsd = TRUE,
                          savedir = DateFile_null,
                          bias.correct = bias.correct)
```

Save outputs from estimation

```
Report_null = Obj_null$report()

Save_null = list("Opt"=Opt_null, "Report"=Report_null,
                 "ParHat"=Obj_null$env$parList(Opt_null$par),
                 "TmbData"=TmbData_null)

save(Save_null, file=paste0(DateFile_null,"Save.RData"))
```

4.2 Model 1

First, create a subdirectory for the **Model 1**

```
DateFile_Mod1 = paste0(DateFile,"Model 1/")
dir.create(DateFile_Mod1)
```

Building and compiling:

`Q_ik` is the argument to Data_Fn for catchability covariates.

- Note: `Q_ik` expects a matrix so we specify `Q_ik=as.matrix(Data_Geostat[,'Duration'])`

```
# Build
if (length(species.codes) > 1 & combineSpecies == FALSE) {
    # MULTISPECIES
    TmbData_Mod1 = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = as.numeric(Data_Geostat[, "spp"]) - 1,
        b_i = Data_Geostat[, "Catch_KG"], a_i = Data_Geostat[,
            "AreaSwept_km2"], v_i = as.numeric(Data_Geostat[,
            "Vessel"]) - 1, s_i = Data_Geostat[, "knot_i"] -
            1, t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options,
        Q_ik = as.matrix(Data_Geostat[, "Duration"]))
} else {
    # SINGLE SPECIES
    TmbData_Mod1 = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = rep(0, nrow(Data_Geostat)), b_i = Data_Geostat[,
            "Catch_KG"], a_i = Data_Geostat[, "AreaSwept_km2"],
        v_i = as.numeric(Data_Geostat[, "Vessel"]) -
            1, s_i = Data_Geostat[, "knot_i"] - 1,
        t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options,
        Q_ik = as.matrix(Data_Geostat[, "Duration"]))
}

# Compile TMB object
TmbList_Mod1 = VAST::Build_TMB_Fn(TmbData = TmbData_Mod1,
    RunDir = DateFile_Mod1, Version = Version, RhoConfig = RhoConfig,
    loc_x = Spatial_List$loc_x, Method = Method, Q_Config = TRUE,
    CovConfig = FALSE)
Obj_Mod1 = TmbList_Mod1[["Obj"]]
```

Fit VAST model to the data by optimizing the TMB function.

```
Opt_Mod1 = TMBhelper::Optimize(obj = Obj_Mod1, lower = TmbList_Mod1[["Lower"]],
                          upper = TmbList_Mod1[["Upper"]], getsd = TRUE,
                          savedir = DateFile_Mod1,
                          bias.correct = bias.correct)
```

Save outputs from estimation

```
Report_Mod1 = Obj_Mod1$report()

Save_Mod1 = list("Opt"=Opt_Mod1, "Report"=Report_Mod1,
                 "ParHat"=Obj_Mod1$env$parList(Opt_Mod1$par),
                 "TmbData"=TmbData_Mod1)

save(Save_Mod1, file=paste0(DateFile_Mod1,"Save.RData"))
```

4.3 Model 2

First, create a subdirectory for the **Model 2**

```
DateFile_Mod2 = paste0(DateFile,"Model 2/")
dir.create(DateFile_Mod2)
```

Building and compiling:

`Q_ik` is the argument to Data_Fn for catchability covariates.

- Note: `Q_ik` expects a matrix so we specify `Q_ik=as.matrix(Data_Geostat[,'Gear_Temperature'])`

```
# Build
if (length(species.codes) > 1 & combineSpecies == FALSE) {
    # MULTISPECIES
    TmbData_Mod2 = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = as.numeric(Data_Geostat[, "spp"]) - 1,
        b_i = Data_Geostat[, "Catch_KG"], a_i = Data_Geostat[,
            "AreaSwept_km2"], v_i = as.numeric(Data_Geostat[,
            "Vessel"]) - 1, s_i = Data_Geostat[, "knot_i"] -
            1, t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options,
        Q_ik = as.matrix(Data_Geostat[, "Gear_Temperature"]))
} else {
    # SINGLE SPECIES
    TmbData_Mod2 = VAST::Data_Fn(Version = Version,
        FieldConfig = FieldConfig, OverdispersionConfig = OverdispersionConfig,
        RhoConfig = RhoConfig, ObsModel = ObsModel,
        c_i = rep(0, nrow(Data_Geostat)), b_i = Data_Geostat[,
            "Catch_KG"], a_i = Data_Geostat[, "AreaSwept_km2"],
        v_i = as.numeric(Data_Geostat[, "Vessel"]) -
            1, s_i = Data_Geostat[, "knot_i"] - 1,
        t_i = Data_Geostat[, "Year"], a_xl = Spatial_List$a_xl,
        MeshList = Spatial_List$MeshList, GridList = Spatial_List$GridList,
        Method = Spatial_List$Method, Options = Options,
        Q_ik = as.matrix(Data_Geostat[, "Gear_Temperature"]))
}

# Compile TMB object
TmbList_Mod2 = VAST::Build_TMB_Fn(TmbData = TmbData_Mod2,
    RunDir = DateFile_Mod2, Version = Version, RhoConfig = RhoConfig,
    loc_x = Spatial_List$loc_x, Method = Method, Q_Config = TRUE,
    CovConfig = FALSE)
Obj_Mod2 = TmbList_Mod2[["Obj"]]
```

Fit VAST model to the data by optimizing the TMB function.

```
Opt_Mod2 = TMBhelper::Optimize(obj = Obj_Mod2, lower = TmbList_Mod2[["Lower"]],
                          upper = TmbList_Mod2[["Upper"]], getsd = TRUE,
                          savedir = DateFile_Mod2,
                          bias.correct = bias.correct)
```

Save outputs from estimation

```
Report_Mod2 = Obj_Mod2$report()

Save_Mod2 = list("Opt"=Opt_Mod2, "Report"=Report_Mod2,
                 "ParHat"=Obj_Mod2$env$parList(Opt_Mod2$par),
                 "TmbData"=TmbData_Mod1)

save(Save_Mod2, file=paste0(DateFile_Mod2,"Save.RData"))
```

---

5 Compare Models

5.1 Check convergence

To evaluate convergence of our three candidate models we will look at convergence and the maximum gradient, which can both be accessed from the **Opt** object as **Opt**$convergence * *and * *Opt$**max_gradient**.

- Note: **Opt$convergence = 0** indicates relative convergence.

```r
temp.table = NULL
temp.table$name = c('Null Model', 'Model 1', 'Model 2')
temp.table$convergence = c(Opt_null$convergence,
                           Opt_Mod1$convergence,
                           Opt_Mod2$convergence)
temp.table$message = c(Opt_null$message,
                       Opt_Mod1$message,
                       Opt_Mod2$message)
temp.table$max_gradient = c(Opt_null$max_gradient,
                            Opt_Mod1$max_gradient,
                            Opt_Mod2$max_gradient)
temp.table = data.frame(temp.table)
names(temp.table) = c('Model Name','Convergence','Message','Maximum Gradient')
#Print the table
kable(temp.table, digits=5)
```

Model Name	Convergence	Message	Maximum Gradient
Null Model	0	relative convergence (4)	0.00596
Model 1	1	false convergence (8)	0.00412
Model 2	0	relative convergence (4)	0.00357

5.2 Covariate effects

Now that we have fit these three alternative models and checked convergence, lets see what the estimates are for the effect of each covariate on catchability.

5.2.1 Model 1 with tow duration

We can recall that the parameters describing the effect of **catchability** covariates on the encounter probability component of our delta-model is $\lambda_1(k)$, and the effect on the positive catch rate component is $\lambda_2(k)$.

So, in our mode output below we are intersted in:

- `lambda1_k` - Effect of tow duration on encounter probability.
- `lambda2_k` - Effect of tow duration on positive catch rate.

```r
Opt_Mod1$SD
```

```
## sdreport(.) result
##               Estimate Std. Error
## ln_H_input     0.64851    0.20259
## ln_H_input     0.56346    0.20731
## beta1_ct       0.61398    0.48733
## beta1_ct       0.93182    0.51232
## beta1_ct      0.81314    0.50142
## beta1_ct      0.81279    0.48973
## beta1_ct      0.47392    0.42532
## beta1_ct     -0.01317    0.42426
## beta1_ct     -0.56960    0.44196
## beta1_ct     -0.08540    0.42061
## beta1_ct     -0.04397    0.41884
## beta1_ct     -0.00513    0.42679
## beta1_ct      0.37844    0.42434
## beta1_ct      0.38363    0.42684
## beta1_ct      0.38689    0.42714
## beta1_ct      0.33944    0.42198
## beta1_ct     -0.45818    0.42663
## lambda1_k     0.49192    0.62780
## L_omega1_z   -1.46618    0.16985
## L_epsilon1_z -0.56886    0.06042
## logkappa1    -4.45666    0.12561
## beta2_ct      7.48262    0.34104
## beta2_ct      7.40080    0.35772
## beta2_ct      7.48505    0.34541
## beta2_ct      7.64410    0.34278
## beta2_ct      7.53090    0.25577
## beta2_ct      7.39118    0.25761
## beta2_ct      7.00254    0.27492
## beta2_ct      7.19419    0.25197
## beta2_ct      7.19774    0.24965
## beta2_ct      7.10910    0.25809
## beta2_ct      7.53411    0.25550
## beta2_ct      7.52398    0.25763
## beta2_ct      7.61083    0.25796
## beta2_ct      7.27991    0.25101
## beta2_ct      6.90894    0.26380
## lambda2_k    -0.36331    0.58631
## L_omega2_z    0.54566    0.07721
## L_epsilon2_z  0.33622    0.04954
## logkappa2    -4.60930    0.17617
## logSigmaM     0.45724    0.00951
## Maximum gradient component: 0.00412
```

5.2.2 Model 2 with gear temperature

For our second model we are interested in:

- `lambda1_k` - Effect of gear temperature on encounter probability.
- `lambda2_k` - Effect of gear temperature on positive catch rate.

```
Opt_Mod2$SD
```

```
## sdreport(.) result
##               Estimate Std. Error
## ln_H_input       0.643    0.20830
## ln_H_input       0.579    0.21117
## beta1_ct        -1.101    0.43653
## beta1_ct        -0.687    0.45572
## beta1_ct        -0.706    0.45271
## beta1_ct        -0.708    0.42913
## beta1_ct        -1.128    0.42728
## beta1_ct        -1.492    0.42102
## beta1_ct        -2.336    0.45332
## beta1_ct        -1.845    0.43199
## beta1_ct        -1.736    0.42854
## beta1_ct        -1.466    0.42163
## beta1_ct        -1.088    0.42175
## beta1_ct        -1.216    0.42925
## beta1_ct        -1.170    0.42929
## beta1_ct        -1.458    0.43390
## beta1_ct        -2.172    0.43506
## lambda1_k        0.311    0.02985
## L_omega1_z      -1.411    0.16305
## L_epsilon1_z    -0.605    0.06087
## logkappa1       -4.497    0.12051
## beta2_ct         8.552    0.22234
## beta2_ct         8.454    0.23778
## beta2_ct         8.425    0.23635
## beta2_ct         8.616    0.21245
## beta2_ct         8.536    0.21117
## beta2_ct         8.287    0.20434
## beta2_ct         8.089    0.23694
## beta2_ct         8.328    0.22223
## beta2_ct         8.274    0.21700
## beta2_ct         7.999    0.20348
## beta2_ct         8.431    0.20271
## beta2_ct         8.518    0.21236
## beta2_ct         8.594    0.21435
## beta2_ct         8.419    0.22072
## beta2_ct         7.953    0.22573
## lambda2_k       -0.193    0.02346
## L_omega2_z      -0.508    0.07026
## L_epsilon2_z    -0.344    0.05023
## logkappa2       -4.353    0.19187
## logSigmaM        0.451    0.00957
## Maximum gradient component: 0.00357
```

5.3 Compare AIC across models

One way to compare across our candidate models with and without catchability covariates is to use AIC. The AIC for each model can be accessed from **Opt$AIC**.

Lets compare AIC across models…

```r
aic.table <- NULL
aic.table$name = c('Null Model', 'Model 1', 'Model 2')
aic.table$AIC = c(Opt_null$AIC, Opt_Mod1$AIC, Opt_Mod2$AIC)
#Calculate dAIC
aic.table$dAIC <- aic.table$AIC - min(aic.table$AIC)
#Data frame
aic.table <- data.frame(aic.table)
names(aic.table) <- c('Model Name', 'AIC', 'dAIC')
#Print the table
kable(aic.table, digits=5)
```

Model Name	AIC	dAIC
Null Model	66056	178
Model 1	66059	181
Model 2	65878	0

6 General conclusions

Here are some general conclusions regarding GOA Pacific Cod:

- For **Model 1** the effect of tow duration is highly uncertain with CV>1 estimated for effects of this covariate on both encounter probability `lambda1_k` and positive catch rate `lambda2_k`.
- For **Model 2** estimated catchability covariate effects have lower uncertainty
- Encounter probability is estimated to **increase** with gear temperature.
- Positive catch rate is estimated to **decrease** with gear temperature.
- It appears that **Model 2** which incorporates **gear temperature** as a **catchability** covariate provides a more parsimonious fit to the survey data.
- It should be noted that a Poisson-link delta-model may be a better way to correct for differences in tow duration.
- This may be specified with `ObsModel = c(1,1)`.
