# VAST workshop 2020

執筆: 金森由妃（研究支援@中央水研）
kana.yuki@fra.affrac.go.jp

# Part 1: 年効果だけのモデル

まずは年効果のみを入れたモデルを単一種のデータに適用してみる．モデルは，・・・．
Part1で必要な情報は**『年，CPUE（or アバンダンスと努力量），緯度，経度』のみ**である．

---

## 0. フォルダとデータの作成

1. ワークショップ用のフォルダ『vastws』を任意の場所に作成し，パスを確認する．
   VASTのアウトプットは容量が大きいため，**フォルダをデスクトップに作成することはお勧めしない．**
2. 作成したフォルダに，解析で用いるデータ（csvファイルなど）を入れる
3. 確認したパスを以下のように入力し，作成したフォルダを作業ディレクトリとして設定する

```
dirname = "/Users/Yuki/Dropbox/vastws"
setwd("dir = dirname")
```

4. 解析で用いるパッケージを呼び出す

```
require(VAST)
require(TMB)
```

5. データを読み込み，オブジェクト名をdfとする．例えばcsvファイルでは，
   `df = read.csv("####.csv")`
6. 各列に『年，CPUE（あるいは，アバンダンスと努力量），緯度，経度』が入ったデータフレーム（tidyデータ）を作成する．オブジェクト名は，dfとしたままでよい．

# 1. 各種設定

## 1.1 cppファイルのバージョンを指定

- cppファイルとはTMBを動かすコードのことで，C++言語で書かれている
- cppファイルのバージョンは，**VASTやTMBなどのバージョンとは別物**
- 最新版では色々なオプションがあるが，CPUE標準化では使わない場合が多い

```
# 最新版のcppファイルを指定する時
Version = get_latest_version(package = "VAST")
```

- MacとLinuxでは最新版のcppファイルをコンパイルできないバグが発生しているため，テストコードを走らせたバージョンを以下のように指定する

```
Version = "VAST_v4_2_0"
```

- `VAST_v4_2_0`あたりが安定しているっぽい

## 1.2 空間の設定

K平均法によるknot決めを行う

```
Method = c("Grid", "Mesh", "Spherical_mesh")[#]
```

- データの観測点が空間的に均一な場合（例えば，格子点上に観測点が存在する）には，`Method = c("Grid", "Mesh", "Spherical_mesh")[1]`を選択する
- データの観測点が空間的に不均一な場合には，

```
# 観測点が狭い範囲にある場合（例えば，日本近海）
Method = c("Grid", "Mesh", "Spherical_mesh")[2]
# 観測点が全球に渡る場合
Method = c("Grid", "Mesh", "Spherical_mesh")[3]
```

とする．Thorson (2019)は，Meshを使った場合でも感度分析的にgridでも解析することを勧め

ている.

- ここでの設定について理解するためには，Gaussian field，Gaussian Markov Random Field，Matérn関数，INLA，SPDE，有限要素法などを勉強する必要がある（とっても難しい）.

```
# 変更の必要なし
Kmeans_Config = list("randomseed" = 1, "nstart" = 100, "iter.max" = 1000)
```

```
grid_size_km = 2.5
```

- MethodがGridの場合に必要な情報
- Meshの場合には関係ないが，NULLとすると『2.3 derived objects for spatio-temporal estimation』でエラーが出るため**触らない**

```
# knotの数の指定
n_x = 100
```

- Thorson (2019)は100以上を推奨
- knot数が多いほど滑らかに近似されるためAICは下がるが，計算負荷が大きくなる

## 1.3 モデルの設定

- プログラムコードの中でもっとも重要な部分で，解析するモデルについて『因子分析の因子数・時間の扱い・分散・観測誤差とリンク関数』を設定する.
- ギリシャ文字とギリシャ文字の直後の数字はVASTのモデル式と対応している．例えば，Beta1は遭遇確率の年効果，Beta2は遭遇確率が >0である場合の密度の年効果を表す.

```
FieldConfig = c(Omega1 = 1, Epsilon1 = 1, Omega2 = 1, Epsilon2 = 1)
```

- 因子分析の因子数．上限はカテゴリー数（種，年齢，体長などの数）．今回は単一種を解析するため，最大数は1.

```
RhoConfig = c(Beta1 = 0, Beta2 = 0, Epsilon1 = 0, Epsilon2 = 0)
```

- 今回は年を固定効果，時空間のランダム効果の年は独立と考えている
- Betaには，分散が年で変わる(= 1)，ランダムウォーク(= 2)，定数(= 3)，AR(= 4)が選択できる．
- Epsilonには，ランダムウォーク(= 2)，AR(= 4)が選択できる．

```
OverdispersionConfig = c("Eta1" = 0, "Eta2" = 0)
```

- 詳細はPart 2で紹介するため，とりあえず0にする

```
ObsModel = c(PostDist = , Link = )
```

- 観察誤差の分布とリンク関数についての設定．非常にたくさんの選択肢がある．詳細は?make_dataを参照されたい
- ここでは簡単のため，代表的な場合を紹介する

# データの種類とパラメータの選択を表にまとめる

## 1.4 データの範囲1

```
strata.limits = data.frame('STRATA'="All_areas")
```

- 変更の必要はない

## 1.5 データの範囲2

```
Region = "other"
```

- 自分のデータを解析する場合は，"other"に変更
- FishStatsUtilsに入っているテストデータを解析する時のみ，適切な地域を選択する．

## 1.6 設定の保存

```
DateFile = paste0(getwd(),'/VAST_output/')
dir.create(DateFile)
Record = list(Version = Version,
              Method = Method,
              grid_size_km = grid_size_km,
              n_x = n_x,
              FieldConfig = FieldConfig,
              RhoConfig = RhoConfig,
              OverdispersionConfig = OverdispersionConfig,
              ObsModel = ObsModel,
              Kmeans_Config = Kmeans_Config,
              Region = Region,
              strata.limits = strata.limits)
setwd(dir = DateFile)
save(Record, file = file.path(DateFile, "Record.RData"))
capture.output(Record, file = paste0(DateFile, "/Record.txt"))
```

- 作業ディレクトリの直下に，`VAST_output`というフォルダが作成され，結果が入れられていく．
- デフォルトのままだとフォルダ名が解析ごとに同じになるため，**解析結果が上書き保存されてしまう**
- `paste0(getwd(), "/vast", Sys.Date(), "_lnorm_log", n_x, sakana)`などどしておくと，フォルダ名を見ただけで『いつ，どんなモデルで，knot数がいくつで，どの魚種を解析した結果なのか』が分かる

# 2. VASTに合わせたデータセットの準備

## 2.1 データフレームの作成

```
head(df)

# CPUEデータの時
Data_Geostat = df %>%
  mutate(Year = year,
         Lon = lon,
         Lat = lat,
         Catch_KG = cpue)
# アバンダンスと努力量データの時
Data_Geostat = df %>%
  mutate(Year = year,
         Lon = lon,
         Lat = lat,
         Catch_KG = abundance,
         AreaSwept_km2 = effort)
```

- **VASTに渡すデータのオブジェクト名は，必ずData_Geostat**
- **列名はオリジナルで作成せず，VAStのデフォルトに合わせる．また列名はキャメルケース（大文字始まり）で書く**
- オブジェクト名がData_Geostatでない場合，列名をオリジナルで作成した場合，列名がキャメルケースでない場合は，以降のコードを修正する必要が出てくる（関数の中身も修正しなければいけないので，めちゃくちゃ大変）

## 2.2 データフレームから位置情報を取得

```
# コード確認！
Extrapolation_List = FishStatsUtils::make_extrapolation_info(
  Regio = Region, #zone range in Japan is 51:56
  strata.limits = strata.limits,
  observations_LL = Data_Geostat[, c("Lat", "Lon")]
  )
```

- 緯度経度をUTM(Universal Transverse Mercator)座標へ変換している
- データフレームから検出した位置情報（zone）を教えてくれるので確認する

```
# 出力例
# この表示はエラーではない
# 日本は51~56の範囲に入る
```

```
Using strata 1
convUL: For the UTM conversion, automatically detected zone 9.
convUL: Converting coordinates within the northern hemisphere.
```

## 2.4 観測点をknotに変換

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

- 『1.2 空間の設定』の情報を使っている

```
# 出力例
# これもエラーではない
convUL: Converting coordinates within the northern hemisphere.
convUL: For the UTM conversion, used zone 9 as specified
convUL: Converting coordinates within the northern hemisphere.
convUL: For the UTM conversion, used zone 9 as specified
Num=1 Current_Best=Inf New=172166.9
 .
 .
 .
convUL: Converting coordinates within the northern hemisphere.
convUL: Converting coordinates within the northern hemisphere.
```

## 2.5 データフレームの保存

ggvastで描画するためのオリジナルコード

```
Data_Geostat = cbind(Data_Geostat,
```

```
                      knot_i = Spatial_List[["knot_i"]],
                      zone = Extrapolation_List[["zone"]] # 加筆した部分
                      )
write.csv(Data_Geostat, "Data_Geostat.csv")
```

## 3. パラメータの設定

### 3.1 TMBに渡すデータを作成する

```
TmbData = make_data(
  Version = Version,
  FieldConfig = FieldConfig,
  OverdispersionConfig = OverdispersionConfig,
  RhoConfig = RhoConfig,
  ObsModel = ObsModel,
  c_iz = rep(0, nrow(Data_Geostat)),
  b_i = Data_Geostat[, 'Catch_KG'],
  a_i = Data_Geostat[, 'AreaSwept_km2'], # CPUEデータの場合は不要
  s_i = Data_Geostat[, 'knot_i'] - 1,
  t_i = Data_Geostat[, 'Year'],
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE
)
```

# 引数の表を入れる

- その他については，?make_dataを参照

```
# 出力例
FieldConfig_input is:
Component_1 Component_2
Omega Epsilon
Beta OverdispersionConfig_input is: Eta1 Eta2
1 1 1 1
-2 -2
Calculating range shift for stratum #1:
```

# 100%の時について入れる

### 3.2 パラメータリストを作成

```
TmbList = VAST::make_model(TmbData = TmbData,
                           RunDir = DateFile,
                           Version = Version,
                           RhoConfig = RhoConfig,
                           loc_x = Spatial_List$loc_x,
                           Method = Spatial_List$Method)
```

- 『1.1 cppファイルのバージョン』で指定したcppファイルをコンパイルする.
- 推定するパラメータが列挙されるので，合っているか確認
- positive catchのモデルでは，{ギリシャ文字}2しか推定する必要が無いにも関わらず，{ギリシャ文字}1も推定パラメータとして列挙されることがある（make_model()のバグ？）．その場合，解析がうまくいかなくなる可能性があるので，以下のようにして不要なパラメータを除去する

# パラメータの抜き方

# パラメータについて表？

### 3.3 パラメータの推定

```
# 何も変更しない
Obj = TmbList[["Obj"]]
Opt = TMBhelper::fit_tmb(obj = Obj,
                           lower = TmbList[["Lower"]],
                           upper = TmbList[["Upper"]],
                           getsd = TRUE,
                           savedir = DateFile,
                           bias.correct = TRUE)
```

```
# 出力例
Constructing atomic D_lgamma
```

```
Optimizing tape... Done
iter: 1 value: 13012.14 mgc: 36.81998 ustep: 1
iter: 2 value: 12951.89 mgc: 9.56431 ustep: 1
iter: 3 value: 12949.05 mgc: 2.199174 ustep: 1
Matching hessian patterns... Done
outer mgc: 3081.279
.
.
.
iter: 1 mgc: 2.867521e-11
outer mgc: 0.004092186
Optimizing tape... Done
iter: 1 mgc: 2.867521e-11
Matching hessian patterns... Done
outer mgc: 31832.82
#########################
The model is likely not converged
#########################
```

- 『収束していない』と出るが，モデル診断で問題が無い場合でも出てくるメッセージなので，『終わったよ』の合図くらいに思っておけばよい

## 3.4 推定結果の保存

```
Report = Obj$report()
Save = list("Opt" = Opt,
            "Report" = Report,
            "ParHat" = Obj$env$parList(Opt$par),
            "TmbData" = TmbData)
save(Save, file = paste0(DateFile,"/Save.RData"))
```

- 作業ディレクトリに推定結果が`Save.RData`として保存される

# 4. 描画

何も考えずに全て実行する

```
# 4.1 Plot data
plot_data(Extrapolation_List = Extrapolation_List,
          Spatial_List = Spatial_List,
          Data_Geostat = Data_Geostat,
          PlotDir = DateFile)

# 4.2 Convergence
pander::pandoc.table(Opt$diagnostics[, c('Param','Lower','MLE',
                                         'Upper','final_gradient')])

# 4.3 Diagnostics for encounter-probability component
Enc_prob = plot_encounter_diagnostic(Report = Report,
                                     Data_Geostat = Data_Geostat,
                                     DirName = DateFile)

# 4.4 Diagnostics for positive-catch-rate component
Q = plot_quantile_diagnostic(TmbData = TmbData,
                             Report = Report,
                             FileName_PP = "Posterior_Predictive",
                             FileName_Phist = "Posterior_Predictive-Histogram",
                             FileName_QQ = "Q-Q_plot",
                             FileName_Qhist = "Q-Q_hist",
                             DateFile = DateFile )
# 4.5 Diagnostics for plotting residuals on a map
MapDetails_List = make_map_info("Region" = Region,
                                "spatial_list" = Spatial_List,
                                "Extrapolation_List" = Extrapolation_List)
Year_Set = seq(min(Data_Geostat[,'Year']), max(Data_Geostat[,'Year']))
Years2Include = which(Year_Set %in% sort(unique(Data_Geostat[,'Year'])))

# FishStatsUtils(2.3.4)を使っている場合は#の行も入れる
# それ以前のバージョンのFishStatsUtilsを使っている場合は#の行をコメントアウトする
plot_residuals(Lat_i = Data_Geostat[,'Lat'],
               Lon_i = Data_Geostat[,'Lon'],
               TmbData = TmbData,
               Report = Report,
               Q = Q,
               savedir = DateFile,
               spatial_list = Spatial_List, # ここ！
               extrapolation_list = Extrapolation_List, # ここ！
               MappingDetails = MapDetails_List[["MappingDetails"]],
               PlotDF = MapDetails_List[["PlotDF"]],
               MapSizeRatio = MapDetails_List[["MapSizeRatio"]],
               Xlim = MapDetails_List[["Xlim"]],
               Ylim = MapDetails_List[["Ylim"]],
               FileName = DateFile,
               Year_Set = Year_Set,
               Years2Include = Years2Include,
               Rotate = MapDetails_List[["Rotate"]],
```

```
                Cex = MapDetails_List[["Cex"]],
                Legend = MapDetails_List[["Legend"]],
                zone = MapDetails_List[["Zone"]],
                mar = c(0,0,2,0),
                oma = c(3.5,3.5,0,0),
                cex = 1.8)

# 4.6 Direction of "geometric anisotropy"
plot_anisotropy(FileName = paste0(DateFile,"Aniso.png"),
                Report = Report,
                TmbData = TmbData)

# 4.7 Density surface for each year
Dens_xt = plot_maps(plot_set = c(3),
                    MappingDetails = MapDetails_List[["MappingDetails"]],
                    Report = Report,
                    Sdreport = Opt$SD,
                    PlotDF = MapDetails_List[["PlotDF"]],
                    MapSizeRatio = MapDetails_List[["MapSizeRatio"]],
                    Xlim = MapDetails_List[["Xlim"]],
                    Ylim = MapDetails_List[["Ylim"]],
                    FileName = DateFile,
                    Year_Set = Year_Set,
                    Years2Include = Years2Include,
                    Rotate = MapDetails_List[["Rotate"]],
                    Cex = MapDetails_List[["Cex"]],
                    Legend = MapDetails_List[["Legend"]],
                    zone = MapDetails_List[["Zone"]],
                    mar = c(0,0,2,0),
                    oma = c(3.5,3.5,0,0),
                    cex = 1.8,
                    plot_legend_fig = FALSE)
Dens_DF = cbind("Density" = as.vector(Dens_xt),
                "Year" = Year_Set[col(Dens_xt)],
                "E_km" = Spatial_List$MeshList$loc_x[row(Dens_xt),'E_km'],
                "N_km" = Spatial_List$MeshList$loc_x[row(Dens_xt),'N_km'])
pander::pandoc.table(Dens_DF[1:6,], digits=3)

# 4.8 Index of abundance
Index = plot_biomass_index(DirName = DateFile,
                           TmbData = TmbData,
                           Sdreport = Opt[["SD"]],
                           Year_Set = Year_Set,
                           Years2Include = Years2Include,
                           use_biascorr = TRUE)
pander::pandoc.table(Index$Table[,c("Year","Fleet","Estimate_metric_tons",
                                    "SD_log","SD_mt")] )
# 4.9 Center of gravity and range expansion/contraction
plot_range_index(Report = Report,
                 TmbData = TmbData,
                 Sdreport = Opt[["SD"]],
                 Znames = colnames(TmbData$Z_xm),
                 PlotDir = DateFile,
                 Year_Set = Year_Set)
```

- 4.7では推定相対密度のマップが作成される．`plot_set = c()`を変えると，推定相対密度以外のマップも作成可能．詳細は`?plot_map`
- バイアスコレクションは必須（Thorson & ristensen 2016）なので，4.8では`use_biascorr = TRUE`にする
- 4.8と4.9で以下のようなメッセージが出るが，エラーではない

```
# 4.8
Using bias-corrected estimates for abundance index (natural-scale)...
Using bias-corrected estimates for abundance index (log-scale)...
```

```
# 4.9
Plotting center-of-gravity...
Using bias-corrected estimates for center of gravity...
Plotting effective area occupied...
Using bias-corrected estimates for effective area occupied (natural scale)...
Using bias-corrected estimates for effective area occupied (log scale)...
```

# 5. アウトプットの見方

『4. 描画』で作成されたアウトプットについていくつか紹介する．全てを紹介することはできないので，VASTのgithubの『deprecated_examples』フォルダに入っている資料（ワークショップHPのマニュアルのリンク先）を参照されたい

## 5.1 解析したデータの空間情報

**`Data_and_knots.png`**

- 上の図2つが解析した空間範囲のマップ
- 下の図がknotの位置

## 5.2　モデル診断

**`parameter_estimates.txt`**

- パラメータの推定値が入っている
- `$diagnostics`のMLE列の値がLowerとUpperに近くなっていないか，final_gradient列の値が0に近くなっているかが収束の判断材料となる

**`QQ_Fnフォルダ`**

- `Posterior_Predictive-Histogram-1.jpg`が$y = x$に近いかどうかが収束の判断材料となる

**`Diag--Encounter_prob.png`**

- ピンクのリボンは95%信頼区間

## 5.3 推定資源量指標値の年変化

**`Index-Biomass.png`**

- 推定資源量指数の平均値とSD

- 推定資源量指数とは各knotの推定相対密度に各knotの面積を掛けたもの．詳細はThorson(2019)を参照されたい

`Table_for_SS3.csv`

- 『Index-Biomass.png』の元データ

## 5.4 推定相対密度のマップ

`Dens.png`

- 赤いほど相対密度が高いことを表す

## 5.5 重心の変化

`center_of_gravity.png`

- 『Dens.png』のデータから重心を計算し，年変化を描画したもの
- 重心の算出式はThorson(2019)を参照されたい

## 5.6 有効面積

`Effective_Area.png`

- 算出式はThorson(2019)を参照されたい

## 5.7 anisotropy

`Aniso.ping`

- 空間相関の強度と歪みを表す

# Part 2: ggvastパッケージを使った描画

ggvastとは，VASTの推定結果を作図するためのパッケージ．VASTではFishStatsUtilsを用いて作図をしているが，

- 後日，Save.RDataを使って作図をすることができない
- VASTやFishStatsUtilsが変更されると，これまでのコードで作図ができなくなることがある
- 軸の名前が変更できない
  - 推定指標値の年トレンドでは，y軸名が必ずmetric tonnesになる
  - 推定密度のマップでは，NorthtingやEastingで表示される
- 推定密度のマップとリジェンドが別々のファイルになる
- COGの変化がkmで表示される
  などの不便な点がある．ggvast はこれらの問題を解決し，様々なハビタット，生物，研究分野でVASTを使いやすくすることを目標としている

---

## 0. ggvastのインストール

```
require(devtools)
devtools::intrall_packeage("ggvast")
```

## 1. VASTの推定結果

### 1.1

# Part 3: 応用モデル

Part1では年の効果のみを入れた単純なモデルを単一種に適用した.
Part3ではより複雑なモデルとして

- **catchabilityへの影響**
- **overdispersionへの影響**
- **複数カテゴリー（種，年齢，銘柄が複数ある場合）の解析**
- **環境の影響** を紹介する.
    - プログラムコードは，**Part1から変更しなければならない部分のみを紹介する**

---

## 1. catchabilityへの影響

catchability（魚の採集率）は，漁具や船，月によって変化していることがある.
ここではそのような現象をモデリングしてみる.
数式は，・・・を参照されたい.
**なお月の効果を考慮したい場合には，『2. overdispersionへの影響』でも扱うことができる.**
**『2. overdispersionへの影響』との違いは，固定効果として推定する点と，漁具などは（直接生物量に影響するのではなく）catchabilityに影響すると考える点である.**

### 0. データの作成

各列に『年，CPUE（あるいは，アバンダンスと努力量），緯度，経度，catchabilityに影響する要因（漁具・船・月など）』が入ったデータフレーム（tidyデータ）を作成する．オブジェクト名は，dfとしたままでよい.

### 2. VASTに合わせたデータセットの準備

### 2.1 データフレームの作成

```
head(df)

# CPUEデータの時
Data_Geostat = df %>%
  mutate(Year = year,
         Lon = lon,
         Lat = lat,
         Catch_KG = cpue,
         Gear = gear)
# アバンダンスと努力量データの時
```

```
Data_Geostat = df %>%
  mutate(Year = year,
         Lon = lon,
         Lat = lat,
         Catch_KG = cpue,
         Gear = gear)
```

- **VASTに渡すデータのオブジェクト名は，必ずData_Geostat**
- **列名はオリジナルで作成せず，VASTのデフォルトに合わせる．また列名はキャメルケース（大文字始まり）で書く**
- オブジェクト名がData_Geostatでない場合，列名をオリジナルで作成した場合，列名がキャメルケースでない場合は，以降のコードを修正する必要が出てくる（関数の中身も修正しなければいけないので，めちゃくちゃ大変）

## 3. パラメータの設定

## 3.1 TMBに渡すデータを作成する

```
TmbData = make_data(
  Version = Version,
  FieldConfig = FieldConfig,
  OverdispersionConfig = OverdispersionConfig,
  RhoConfig = RhoConfig,
  ObsModel = ObsModel,
  c_iz = rep(0, nrow(Data_Geostat)),
  b_i = Data_Geostat[, 'Catch_KG'],
  a_i = Data_Geostat[, 'AreaSwept_km2'], # CPUEデータの場合は不要
  s_i = Data_Geostat[, 'knot_i'] - 1,
  t_i = Data_Geostat[, 'Year'],
  Q_ik = model.matrix(as.formula(~0+Gear), data = Data_Geostat) # 加筆部分
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE
```

- Q_ikには数値しか入らないため，カテゴリカル変数の場合はダミー変数を作成する必要がある

####

赤文字