VAST workshop 2020

2020/02/07

金森由妃　研究支援@中央水研

Version 1.1.9000

Part I: 年効果だけのモデル

まずは年効果だけが入ったモデルを単一種のデータに適用してみる．．
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．

- プログラムコードは『part1.txt』
- 数式は付録の(3)(4)式

---

0. フォルダとデータの作成

1. ワークショップ用のディレクトリ『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

```
# 必須
require(VAST)
require(TMB)

# 自分がデータ形成で必要なもの
require(tidyverse)
```

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ファイルを読み込む場合は
   `df = read.csv("####.csv")`

6. データを確認する

```
summary(df)
```

7. 各列に『年，CPUE（あるいは，アバンダンスと努力量），経度，緯度，種名』が入ったデータフレーム（tidyデータ）を作成し，各列名を『year, cpue (abundanceとeffort), lon, lat, spp』に変更する．オブジェクト名はdfのままでよい

注意点

- 単一種のモデルを解析するので，**dfに複数種のデータ入っている場合は解析する種を1種決めて抽出する**
- 長期データの場合は計算に時間がかかるので，**ワークショップで解析するデータは5年分までとする．** つまり，dfに6年以上のデータが入っている場合には，5年分を抽出する

```
# 解析する種がマサバ，解析する年が2015~の場合
df = df %>%
  # マサバの2015~2019年のデータを抽出
  dplyr::filter(魚 == "マサバ", dplyr::between(年, 2015, 2019)) %>%
  # 解析に必要な列を選択
  select(年, cpue, 緯度, 経度, 魚) %>%
  # 列名を変更
  dplyr::rename(year = 年, lon = 緯度, lat = 経度, spp = 魚)
```

CPUEデータの例

year	cpue	lon	lat	spp
2015	2.5	135	30	マサバ
2015	0.2	135.5	30	マサバ
2019	1.2	135.5	30	マサバ

アバンダンスと努力量の例

year	abundance	effort	lon	lat	spp
2015	2.5	2	135	30	マサバ
2015	0.2	4	135.5	30	マサバ
2019	1.3	5	135.5	30	マサバ

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追加されている（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，テストコードが走ったバージョンを以下のように指定する

```
# バージョンを指定する場合
Version = "VAST_v4_2_0"
```

- 経験上，`VAST_v4_2_0`あたりが安定しているように感じる

1.2 空間の設定

K平均法を用いたknot決めについての設定を行う．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，有限要素法などの知識が必要になる．VASTを動かすだけならば深い理解がなくても大丈夫なので，とりあえず指示通りに設定することをお勧めする．

```
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

```
Kmeans_Config = list("randomseed" = 1, "nstart" = 100, "iter.max" = 1000)
```

- 変更の必要なし

```
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

因子分析の因子数

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

- カテゴリー（種，年齢，銘柄など）に共通する要因の数をいくつ推定するのかを設定する部分
- 上限はカテゴリーの数
- **今回は単一種を解析するため，最大数は1**

時間の扱い

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

- ここでは年を固定効果，時空間のランダム効果の年は独立と考えているため，**全てに0を入れる**

- BetaとEpsilonの選択肢（参考までに）
  - Beta: 分散が年で変わる(= 1)，ランダムウォーク(= 2)，定数(= 3)，AR(= 4)
  - Epsilon: ランダムウォーク(= 2)，AR(= 4)

Overdispersion

```
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

観察誤差とリンク関数

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

- 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，代表的なものを紹介する

	生物量	PostDist	Link	ObsModel
1	CPUE	Lognormal	遭遇率にlogit 生物量にlog	c(1, 0)
2	CPUE	Gamma	遭遇率にlogit 生物量にlog	c(2, 0)
3	重量	Lognormal	遭遇率にlogit 生物量にlog	c(1, 0)
4	重量	Gamma	遭遇率にlogit 生物量にlog	c(2, 0)
5	個体数	Poisson	生物量にlog	c(7, 1)
6	個体数	Negative bimomial	生物量にlog	c(5, 1)
(7)	遭遇率100%の年がある時			c(__, 3)
(8)	遭遇率100% or 0%の年がある時 (個体数)			c(__, 4)

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"
- 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

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

- 例）`paste0(getwd(), "/vast", Sys.Date(), "_lnorm_log", n_x, sakana)`
    - 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") # 加筆した部分
```

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)), # カテゴリー数
  b_i = Data_Geostat[, 'Catch_KG'], # 応答変数（生物量）
  a_i = Data_Geostat[, 'AreaSwept_km2'], # 努力量
  a_i = rep(1, nrow(Data_Geostat)), # CPUEの場合
  s_i = Data_Geostat[, 'knot_i'] - 1, # knot
  t_i = Data_Geostat[, 'Year'], # 年
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE # 空間相関の歪みを考えるか否か
)
```

```
# 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:
```

遭遇率が100%でエラーが出た場合

- > 0データのみを解析することになる（デルタ型のモデルではなくなる）
- 1.3に戻りモデルの設定を変更し，1.4以降を実行する

```
FieldConfig = c(Omega1 = 0, Epsilon1 = 0, Omega2 = 1, Omega2 = 1)
```

```
ObsModel = c(PostDist = ___, Link = 3)
```

遭遇率が0%でエラーが出た場合

- 0に戻りデータが無い年を除去し，1以降を実行する

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，合っているかを確認
  - 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？）
- 不要なパラメータ入っていた場合，推定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し，TmbListを作成し直す

```
#"Extract Map and modify it to turn off lambda1_k
Map = TmbList$Map
Map[["lambda1_k"]]=rep(NA, length(TmbList$Parameters$lambda1_k))
Map[["lambda1_k"]] = factor(Map[["lambda1_k"]])

# Rebuild TMB object with user-specified Map
TmbList = VAST::make_model(Map = Map,
                           TmbData = TmbData,
                           RunDir = DateFile,
                           Version = Version,
                           RhoConfig = RhoConfig,
                           loc_x = Spatial_List$loc_x,
                           Method = Spatial_List$Method)
```

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"))
```

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 推定相対密度のマップ

`Dens.png`

- 算出式は付録(13)-(15)式を参照
- 赤いほど相対密度が高いことを表す

5.4 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
- 算出式は付録(16)式を参照

`Table_for_SS3.csv`

- 『Index-Biomass.png』の元データ

5.5 有効面積

`Effective_Area.png`

- 算出式は付録(17)-(18)式を参照

5.6 重心の変化

`center_of_gravity.png`

- 算出式は付録(19)式を参照

5.7 anisotropy

`Aniso.ping`

- 空間相関の方向と方強度を表す

Part II: 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

- プログラムコードは『part2.txt』

---

0. ggvastのインストール

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

1. 重心を地図上にプロットする

* ノミナルの重心を地図上にプロットしたい場合は，`get_cog()`で重心を計算してから`map_cog()`で作図する

map_cog()

```
# please change here ----------------------------------------------
vast_output_dirname = "////" # vastの推定結果が入っているディレクトリ
data_type = c("VAST", "nominal")[1]
category_name = c("spotted") #カテゴリーの名前（魚種名や銘柄など）  nominalの場合はNULL
#category_name = c("spotted","chub") #複数カテゴリーの場合

unique(map_data("world")$region)
region = "Japan" #作図する地域を選ぶ

ncol = 5 #横にいくつ図を並べるか（最大数 = カテゴリー数）
shape = 16 #16はclosed dot
size = 1.9 #shapeの大きさ

package = c("SpatialDeltaGLMM", "FishStatsUtils")[2]
map_output_dirname = "////" #作図を入れるディレクトリ
fileEncoding = "CP932"

# load data ------------------------------------------------------
setwd(dir = vast_output_dirname)
load("Save.RData")
DG = read.csv("Data_Geostat.csv")

# make figures ---------------------------------
map_cog(data_type = data_type,
        category_name = category_name,
        region = region,
        ncol = ncol,
        shape = shape,
        size = size,
        package = package,
        map_output_dirname = map_output_dirname,
        fileEncoding = fileEncoding)
```

get_cog()

```
# please change here ----------------------------------------------
vast_output_dirname = "///" #vastの推定結果が入っているディレクトリ

# load data ------------------------------------------------------
setwd(dir = vast_output_dirname)
DG = read.csv("Data_Geostat.csv")
```

```
# make data-frame ---------------------------------
cog_nom = get_cog(data = DG)
```

---

2. 局所密度を地図上にプロットする

- VASTの推定結果の場合は，まず`get_dens()`で`Save.RData`から推定結果を抽出し，その後`map_dens()`でプロットする

get_dens()

```
# please change here ------------------------------------------------
vast_output_dirname = "///" #vastの推定結果が入っているディレクトリ
category_name = c("spotted") #カテゴリーの名前（魚種名や銘柄など）
#category_name = c("spotted","chub") #複数カテゴリーの場合

# load data ---------------------------------------------------------
setwd(dir = vast_output_dirname)
load("Save.RData")
DG = read.csv("Data_Geostat.csv")

# get data-frame ----------------------------------
df_dens = get_dens(category_name = category_name)
```

map_dens()

```
# load data ---------------------------------------------------------
vast_output_dirname = "///" #vastの推定結果が入っているディレクトリ
setwd(dir = vast_output_dirname)
load("Save.RData")
DG = read.csv("Data_Geostat.csv")
#DG = DG %>% filter(Catch_KG > 0) #> 0データのみをプロットしたい場合

# please change here ------------------------------------------------
data = df_dens #VASTの結果ならdf_dens  ノミナルならDG = read.csv("Data_Geostat.csv")
unique(map_data("world")$region)
region = "Japan" #作図する地域を選ぶ
scale_name = "Log density" #凡例　色の違いが何を表しているのかを書く
ncol = 5 #横にいくつ図を並べるか（最大数 = 年数）
shape = 16 #16はclosed dot
size = 1.9 #shapeの大きさ
map_output_dirname = "///" # 作図を入れるディレクトリ
```

```
# make figures ---------------------------------
map_dens(data = data,
         region = region,
         scale_name = scale_name,
         ncol = ncol,
         shape = shape,
         size = size,
         map_output_dirname =  map_output_dirname)
```

---

3. 資源量指標値の年トレンド

- **ノミナルの資源量指標値とVASTで標準化した推定資源量指標値を比較する**
- mutate(type = "##")部分は凡例に反映される．必要に応じて適宜変更することができる

plot_index()

```
# please change here ----------------------------------------
vast_output_dirname = "///" #vastの推定結果が入っているディレクトリ
category_name = c("spotted") #カテゴリーの名前（魚種名や銘柄など）
fig_output_dirname = "///" #作図を入れるディレクトリ

# load data and make data_frame ----------------------------------
setwd(dir = vast_output_dirname)
vast_index = read.csv("Table_for_SS3.csv") %>%
  mutate(type = "Standardized") # 名前変更可

# vastの結果が複数ある場合
setwd(dir = ////)
vast_index2 = read.csv("Table_for_SS3.csv") %>%
  mutate(type = "Standardized2") # 名前変更可
vast_index = rbind(vast_index, vast_index2)

#ノミナルデータ
DG = read.csv("Data_Geostat.csv")

# make figures ---------------------------------
plot_index(vast_index = vast_index,
           DG = DG,
           category_name = category_name)
```

Part III: 複雑なモデル

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
(i) **catchabilityの違い**
(ii) **overdispersion**
(iii) **複数カテゴリー（種，年齢，銘柄が複数ある場合）の解析**
(iv) **環境の影響**

を紹介する．Part IIIでは，**Part1から変更しなければならないプログラムコードのみを紹介する**

(i) catchabilityの違い

ここでは，年効果に加えて，**catchability（採集率）が漁具や船，月によって異なる**というモデルを単一種に適用してみる．

- プログラムコードは『part3_catchability.txt』
- 数式は付録の(5)(6)式

なお漁具や船，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，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と，固定効果として推定する点である

---

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を作成する．オブジェクト名はdfのままでよい

CPUEデータの例

year	cpue	lon	lat	gear
2015	2.5	135	30	Keta
2015	0.2	135.5	30	Beam

2019	1.2	135.5	30	Beam

アバンダンスと努力量の例

year	abundance	effort	lon	lat	gear
2015	2.5	2	135	30	Keta
2015	0.2	4	135.5	30	Beam
2019	1.3	5	135.5	30	Beam

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)), # カテゴリー数
  b_i = Data_Geostat[, 'Catch_KG'], # 応答変数（生物量）
  a_i = Data_Geostat[, 'AreaSwept_km2'], # 努力量（CPUEデータの場合は不要）
  s_i = Data_Geostat[, 'knot_i'] - 1, # knot
  t_i = Data_Geostat[, 'Year'], # 年
  Q_ik = model.matrix(as.formula(~0+Gear), data = Data_Geostat), # 加筆部分
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE # 空間相関の歪みを考えるか否か
)
```

注意点

- 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
- Q_ikに入れられる要因の数は，カテゴリーの数まで

(ii) overdispersion

ここでは，年効果に加えて，分散が漁具うあ船，月によって期待していたよりも大きくなる（overdispersion; 過分散）というモデルを単一種に適用してみる．

- プログラムコードは『part3_overdispersion.txt』
- 数式は付録の(7)(8)式

なお漁具や船，月の効果を考慮したい場合には，『1. catchabilitynへの影響』でも扱うことができる．『1. catchabilityへの影響』との違いは，漁具などは生物量の変動に影響すると考える点と，ランダム効果として推定する点である
年と月の交互作用を考えたい場合にも，overdispersionへの影響として扱うことになる．

---

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，overdispersionに影響する要因（漁具・船・月など）』が入ったデータフレームを作成する．オブジェクト名はdfのままでよい

- 年と月の交互作用を考えたい場合には，年と月を組み合わせたfactor型（Rのデータ型の一つ．因子型とも言う．numericとかcharacterとか，そーゆーやつ）を作る．例えば，

```
df = df %>% mutate(time = paste0("year", "month", sep = "_"))
```

CPUEデータの例

year	cpue	lon	lat	vessel
2015	2.5	135	30	A
2015	0.2	135.5	30	A
2019	1.2	135.5	30	B

アバンダンスと努力量の例

year	abundance	effort	lon	lat	vessel

2015	2.5	2	135	30	A
2015	0.2	4	135.5	30	A
2019	1.3	5	135.5	30	B

1.3 モデルの設定

Overdispersion

```
OverdispersionConfig = c("Eta1" = 1, "Eta2" = 1)
```

- 入れられる要因の数は，カテゴリーの数まで

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,
         Vessel = vessel) # 年と月の交互作用の場合はここを変える
# 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,
         Vessel = vessel) # 年と月の交互作用の場合はここを変える
```

- **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

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)), # カテゴリー数
  b_i = Data_Geostat[, 'Catch_KG'], # 応答変数（生物量）
  a_i = Data_Geostat[, 'AreaSwept_km2'], # 努力量（CPUEデータの場合は不要）
  s_i = Data_Geostat[, 'knot_i'] - 1, # knot
  t_i = Data_Geostat[, 'Year'], # 年
  v_i = matrix(Data_Geostat[, "Vessel"]), # 加筆部分．年×月の場合はここを変える
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE # 空間相関の歪みを考えるか否か
)
```

4. 描画

4.10 Plot overdispersion（追記）

```
Plot_Overdispersion(filename1 = paste0(DateDir, "Overdispersion"),
                    filename2 = paste0(DateDir, "Overdispersion--panel"),
                    Data = TmbData,
                    ParHat = ParHat,
                    Report = Report,
                    ControlList1 = list(Width = 5, Height = 10,
                                        Res = 200, Units = "in"),
                    ControlList2 = list(Width = TmbData$n_c,
                                        Height = TmbData$n_c,
                                        Res = 200, Units = "in"))
```

(iii) 複数カテゴリーの解析　　（修正必要）

ここでは，年効果だけが入ったモデルを複数カテゴリー（種，年齢，銘柄など）のデータに適用してみる．

- プログラムコードは『part3_multispecies.txt』
- 数式は付録の(9)(10)式

---

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，カテゴリー』が入ったデータフレームを作成する．オブジェクト名は，dfのままでよい

CPUEデータの例

year	cpue	lon	lat	category
2015	2.5	135	30	masaba
2015	0.2	135.5	30	gomasaba
2019	1.2	135.5	30	masaba

アバンダンスと努力量の例

year	abundance	effort	lon	lat	category
2015	2.5	2	135	30	masaba
2015	0.2	4	135.5	30	gomasaba
2019	1.3	5	135.5	30	masaba

1.3 モデルの設定

因子分析の因子数

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

- カテゴリー（種，年齢，銘柄など）に共通する要因の数をいくつ推定するのかを設定する部分
- 上限はカテゴリーの数
- 多いほど計算負荷が大きくなる

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,
         spp = category) # 加筆部分
# 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,
         spp = category) # 加筆部分
```

- **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．**カテゴリーに関する列は例外的にキャメルケースではない
- 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 = as.numeric(as.factor(Data_Geostat[, "spp"])) - 1, # カテゴリー数
  b_i = Data_Geostat[, 'Catch_KG'], # 応答変数（生物量）
  a_i = Data_Geostat[, 'AreaSwept_km2'], # 努力量（CPUEデータの場合は不要）
  s_i = Data_Geostat[, 'knot_i'] - 1, # knot
  t_i = Data_Geostat[, 'Year'], # 年
  spatial_list = Spatial_List,
  Options = Options,
  Aniso = TRUE # 空間相関の歪みを考えるか否か
)
```

4. 描画

4. 11 Plot factors（追記）

```
Plot_factors(Report = Report,
             ParHat = Obj$env$parList(),
             Data = TmbData,
             SD = Opt$SD,
             mapdetails_list = MapDetails_List,
             Year_Set = Year_Set,
             category_names = levels(DF[,"Sci"]),
             plotdir = DateFile)
```

(iv) 環境の影響

ここでは，年効果に加えて，様々な環境要因を共変量として持つモデルを単一種に
VASTでは様々な環境要因を共変量として入れることができるが，Part IIIの(i)-(iii)に比べてプログラミング技術が必要である．なぜなら，`[knot, 年, 環境変数]`といった配列データを作成してTMBに渡さなければならないからである．調査・漁業と同時に観測され，生物データと調査データが同一のファイルに保存されている場合もあれば，衛星データのように調査とは独立して観測され，生物データと調査データが別のファイルで保存されている場合もあるため，一般的なプログラミングコードを紹介することは難しい．そのため，ここではTMBへの渡し方のみを紹介する．（何のヒントにもならないが，配列データにはknotの情報が必要であるため，環境データの作成は『2.5 データフレームの保存』と『3.1 TMBに渡すデータを作成する』の間で行うことになる）

- プログラムコードは『part3_env.txt』
- 数式は付録の(11)(12)式

---

3. パラメータの設定

3.1 TMBに渡すデータを作成する

```
# 環境データをenv_dataとした時
TmbData = make_data(
 Version = Version,
 FieldConfig = FieldConfig,
 OverdispersionConfig = OverdispersionConfig,
 RhoConfig = RhoConfig,
 ObsModel = ObsModel,
 c_iz = rep(0, nrow(Data_Geostat)), # カテゴリー数
 b_i = Data_Geostat[, 'Catch_KG'], # 応答変数（生物量）
 a_i = Data_Geostat[, 'AreaSwept_km2'], # 努力量（CPUEデータの場合は不要）
 s_i = Data_Geostat[, 'knot_i'] - 1, # knot
 t_i = Data_Geostat[, 'Year'], # 年
 X_itp = array(env_data, dim = c(n_knot, n_yr, n_env)), #環境要因
 spatial_list = Spatial_List,
 Options = Options,
 Aniso = TRUE # 空間相関の歪みを考えるか否か
)
```

- 1年程前に解析した時，共変量の引数はX_xtpで，NAが入ったデータは解析できなかった．しかし現在はX_gtpとX_itpの2種類があり，X_gtpには『if missing, assumed to not include covariates』と書かれている．X_itpならばNAが入っていても解析できるのかもしれない