

VPA-03(2020)

Frasyrを用いたVPA:実践編②

～チューニングありVPA～

- 二段階法, 選択率更新法, 全F推定法の実行方法の紹介



動画作成者 漁業情報解析部 宮川光代
(mmiyagawa@affrc.go.jp)

Frasyrを用いたVPAの実践（チューニングありVPA編）

※ここは二段階法，選択率更新法，全F推定法に共通の手順

1) 基礎的な設定

- Rのインストール（インストール方法は動画R-01参照）
- Rstudioのインストール（必須ではないですが，インストールすると便利：インストール方法も動画R-01参照）
- Frasyrのインストール方法は <https://github.com/ichimomo/frasyr> に従って下さい。（動画Tool-03を参照）
- インストール後，library(frasyr) でパッケージを呼び出しすることで使えるようになります。

2) VPAに用いるデータファイルの作成

- 色々やり方がありますが，今回は，5つのcsvファイルを作成してRで読み込むようにします
 1. caa.csv ---年別年齢別漁獲尾数のデータファイル
 2. waa.csv ---年別年齢別体重のデータファイル
 3. maa.csv ---年別年齢別体重成熟率のデータファイル
 4. M.csv ---年別年齢別自然死亡係数のデータファイル
 5. **Index.csv** ---資源量指標値(CPUEや資源量指数など)のデータファイル

この他にも，必要に応じて次のようなファイルを生成する：

6. maa_tune.csv ---チューニングに使う成熟率が異なる場合
7. waa_catch.csv ---資源量を計算するときと，漁獲量を計算するときとで年別年齢別重量が異なる場合

5つのcsvファイルの中身の例

※二段階法，選択率更新法，全F推定法に共通

- 例:0-3+歳まで，2011-2020年までのデータで，indexは6つの異なる指標がある場合

caa.csv：年別年齢別漁獲尾数（↓）

caa.csv											
	A	B	C	D	E	F	G	H	I	J	K
1		2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2	0	199.8743	216.5965	267.6906	218.6984	165.5603	249.2485	187.9007	130.2869	79.9196	77.11111
3	1	129.4618	121.4153	162.9626	158.6663	154.1357	95.75337	144.1552	91.39456	62.34938	36.7481
4	2	72.37195	71.29575	82.08322	85.4628	98.07561	76.98423	47.82475	61.086	39.04364	26.22548
5	3	26.55117	32.79557	38.99656	33.76093	40.75214	36.62719	28.75043	15.40572	20.81536	13.74911

waa.csv：年別年齢別体重（↓）

waa.csv											
	A	B	C	D	E	F	G	H	I	J	K
1		2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2	0	0.035833	0.035833	0.035833	0.035833	0.035833	0.035833	0.035833	0.035833	0.035833	0.035833
3	1	0.166985	0.166985	0.166985	0.166985	0.166985	0.166985	0.166985	0.166985	0.166985	0.166985
4	2	0.341248	0.341248	0.341248	0.341248	0.341248	0.341248	0.341248	0.341248	0.341248	0.341248
5	3	0.508367	0.508367	0.508367	0.508367	0.508367	0.508367	0.508367	0.508367	0.508367	0.508367

maa.csv：年別年齢別成熟率（↓）

maa.csv											
	A	B	C	D	E	F	G	H	I	J	K
1		2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
2	0	0	0	0	0	0	0	0	0	0	0
3	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
4	2	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
5	3	1	1	1	1	1	1	1	1	1	1

M.csv：年別年齢別自然死亡係数（↓）

M.csv											
	A	B	C	D	E	F	G	H	I	J	K
		2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
0	0	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
1	1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
2	2	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
3	3	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4

Index.csv：6つの異なる資源量指標値（↓）

Index.csv - 保存しました											
	A	B	C	D	E	F	G	H	I	J	K
		2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
1	1	1719.08	1602.804	1325.97	1449.229	853.9004	1402.377	1391.298	753.3395	552.8876	776.5378
2	2	1200.55	1100	900	1000	600	700	800	700	600	700
3	3	1.429189	1.451814	1.305684	1.11257	1.066373	1.497432	1.129647	0.767521	0.624667	0.860991
4	4	1.529189	1.351814	1.705684	1.112571	1.066373	1.297432	0.829647	1.067521	0.824667	1.060991
5	5	1888.823	1605.954	1392.562	1687.528	1000.92	1497.385	1456.42	856.6916	1038.909	890.8244
6	6	1676.366	1554.339	1267.788	1048.131	516.6027	1040.247	1222.32	720.3847	504.8838	442.2306

データの読み込み（チューニングありVPA編）

※二段階法，選択率更新法，全F推定法に共通

3) VPAに用いるデータファイルの読み込み

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

extract_spe_catch_JABBA.R x extract_spe_cpue.R x 13

Source on Save

```
1 #2020年VPA実践研修
2
3 devtools::install_github("ichimomo/frasyr@dev") #最新版のFrasyrのインストール
4 library(frasyr) #Frasyrの呼び出し
5
6 #データの読み込み----
7
8 caa <- read.csv("caa.csv", row.names=1) #caaの読み込み
9 waa <- read.csv("waa.csv", row.names=1) #waaの読み込み
10 maa <- read.csv("maa.csv", row.names=1) #maaの読み込み
11 M <- read.csv("M.csv", row.names=1) #Mの読み込み
12 index <- read.csv("Index.csv", row.names=1) #indexの読み込み
13
14 dat <- data.handler(caa=caa, waa=waa, maa=maa, M=M, index=index)
15 |
16 dat
17
```

※maa_tuneやwaa_catchを指定している場合は，
下記のようにその部分も加える

```
maa_tune<-read.csv("maa_tune.csv", row.names=1)
waa_catch<-read.csv("waa_catch.csv", row.names=1)

dat <- data.handler(caa=caa, waa=waa, maa=maa, M=M, index=index,
                    maa.tune=maa_tune,
                    waa.catch=waa_catch) #データをVPA用の関数に変換する
```

R上にこのように7行
書いて，実行させる

#以降はコメント文な
ので省略してよい

データ読み込みの確認

※二段階法、選択率更新法、全F推定法に共通

4) データがきちんと読み込まれたことの確認

```
13 index <- read.csv("index.csv", row.names=1) # indexの読み込み
14
15 #maa_tune <- read.csv("maa_tune.csv", row.names=1)
16 #waa_catch <- read.csv("waa_catch.csv", row.names=1)
17
18 dat <- data.handler(caa=caa, waa=waa, maa=maa, M=M, index=index)
19
20 dat
21
22 ↑ datと書いて実行する
23
24
25
26
27
28
29
```

datの中身が表示される

\$caa

\$maa

\$waa

\$M

\$index

(\$maa.tune, \$waa.catch)

が与えたデータと一致しているか確認

一致していればOK

	index	M	maa	waa
	6 obs. of 10 variables	4 obs. of 10 variables	4 obs. of 10 variables	4 obs. of 10 variables

```
2 72.37195 71.29575 82.08322 85.46280 98.07561 76.98423 47.82475 61.08600 39.04364 26.22548
3 26.55117 32.79557 38.99656 33.76093 40.75214 36.62719 28.75043 15.40572 20.81536 13.74911
```

```
$maa
 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3
2 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8 0.8
3 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
```

```
$waa
      2011      2012      2013      2014      2015      2016      2017      2018      2019
0 0.03583254 0.03583254 0.03583254 0.03583254 0.03583254 0.03583254 0.03583254 0.03583254 0.03583254
1 0.16698471 0.16698471 0.16698471 0.16698471 0.16698471 0.16698471 0.16698471 0.16698471 0.16698471
2 0.34124750 0.34124750 0.34124750 0.34124750 0.34124750 0.34124750 0.34124750 0.34124750 0.34124750
3 0.50836731 0.50836731 0.50836731 0.50836731 0.50836731 0.50836731 0.50836731 0.50836731 0.50836731
2020
0 0.03583254
1 0.16698471
2 0.34124750
3 0.50836731
```

```
$index
      2011      2012      2013      2014      2015      2016      2017      2018
1 1719.079998 1602.803898 1325.969967 1449.228810 853.900375 1402.376696 1391.297808 753.3395461
2 1200.550000 1100.000000 900.000000 1000.000000 600.000000 700.000000 800.000000 700.000000
3 1.429189 1.451814 1.305684 1.112570 1.066373 1.497432 1.129647 0.7675213
4 1.529189 1.351814 1.705684 1.112571 1.066373 1.297432 0.829647 1.0675213
5 1888.822675 1605.954237 1392.561657 1687.527624 1000.919915 1497.384805 1456.419877 856.6915698
6 1676.365753 1554.338761 1267.787659 1048.131268 516.602688 1040.247162 1222.319546 720.3846515
2019 2020
1 552.8875843 776.5377895
2 600.0000000 700.0000000
3 0.6246666 0.8609907
4 0.8246666 1.0609907
5 1038.9088640 890.8244353
6 504.8837941 442.2305584
```

```
$M
 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
0 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4
1 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4
2 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4
3 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4 0.4
```

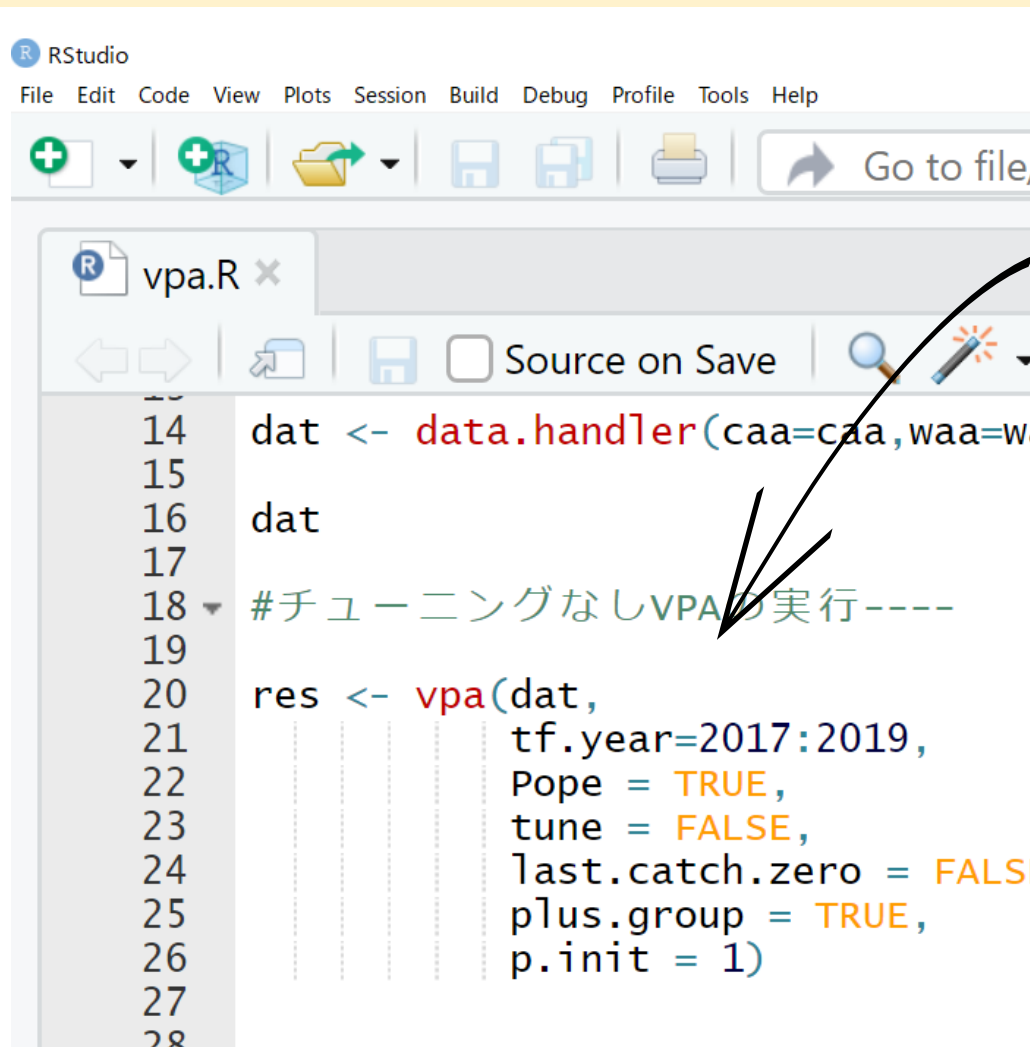
\$maa.tune
NULL

\$waa.catch
NULL

\$catch.prop
NULL

VPAの実行（チューニングありVPA編：二段階法）

5) VPAを走らせる（第一段階）



```
14 dat <- data.handler(caa=caa, waa=waa)
15
16 dat
17
18 #チューニングなしVPAの実行----
19
20 res <- vpa(dat,
21           tf.year=2017:2019,
22           Pope = TRUE,
23           tune = FALSE,
24           last.catch.zero = FALSE,
25           plus.group = TRUE,
26           p.init = 1)
27
28
```

第一段階：チューニングなしVPAを行う

R上にこのように7行書いて、実行させる
（#以降はコメント文なので省略してよい）

動画VPA-01『**frasyr**を用いたVPA：概要編』で解説しているように、
自分のしたい解析に合わせて以下のことを指定する：

1. 最終年の漁獲係数(F)はどの年の平均に等しいとするか
2. Pope近似式を使うか、Baranov方程式にするか
3. チューニングはするのかしないか
4. 最高齢はプラスグループなのか

#VPAを行うデータ

#ターミナルFをどの年の平均にするか

#Popeの近似式を使うならTRUE, Baranovを用いるならFALSE

#チューニングはしないのでFALSE（デフォルトはFALSE）

#最終年の漁獲尾数が全部0の場合はTRUE（デフォルトはFALSE）

#最高齢はプラスグループか否か（デフォルトはTRUE）

#ターミナルFに与える初期値

VPA第一段階の結果をみる（二段階法）

6) VPA第一段階目の結果をみる

The screenshot displays the RStudio interface with a script file named 'vpa.R' open. The script contains several commented-out lines of R code related to VPA (Virtual Population Analysis) results. A yellow callout box points to the 'Run' button in the toolbar, stating: 'ここを実行すると推定された様々な値が確認出来る' (When you execute here, various estimated values can be confirmed). On the right side, the Console window shows the output of the command 'res\$term.f', which is '[1] 1.046609'. A blue callout box points to this value, stating: '推定されたターミナルF' (Estimated terminal F).

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ +R Folder Save Print Go to file/function

vpa.R x

Source on Save Run Source

```
28  
29 #チューニングなしVPAの結果----  
30  
31 res$term.f      #推定されたターミナルFの値をみる  
32  
33 res$faa         #推定された年別年齢別漁獲係数  
34  
35 res$naa        #推定された年別年齢別資源尾数  
36  
37 res$saa        #推定された年別年齢別選択率  
38  
39 res$baa        #推定された年別年齢別資源重量（バイオマス）  
40  
41 res$sbb        #推定された年別年齢別産卵親魚量  
42
```

Console Jobs

~/勉強会/2020

```
> res$term.f  
[1] 1.046609  
>  
>  
>  
>  
>  
>
```

推定されたターミナルF

二段階目の準備：どの年代の選択率を仮定するか

7) 二段階目のチューニングVPAに向けて、ターミナルFを推定するための選択率の仮定を決める

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

madara_reproduce.R test-data.handler.R vpa.R vpa_2_nidan.R

```
47 res$bbb #推定された年別年齢別産卵親魚量
48
49
50
51 #選択率の仮定をどうするか----
52
53 sel_1 <- res$saa$`2018` #2018年の選択率と同じと仮定する場合
54 sel_1
55
56 sel_2 <- res$saa$`2019` #2019年の選択率と同じと仮定する場合
57 sel_2
58
59 sel_3 <- rowMeans(res$saa[,c("2017", "2018", "2019")]) #2017年～2019年の選択率と同じと仮定する場合
60 sel_3
61
```

Console Jobs

~/勉強会/2020資源管理研修会/vpa_2_nidan/

```
> sel_1 <- res$saa$`2018` #2018年の選択率と同じと仮定する場合
> sel_1
[1] 0.5600520 0.7358278 1.0000000 1.0000000
>
> sel_2 <- res$saa$`2019` #2019年の選択率と同じと仮定する場合
> sel_2
[1] 0.5176832 0.6682636 1.0000000 1.0000000
>
> sel_3 <- rowMeans(res$saa[,c("2017", "2018", "2019")]) #2017年～2019年の選択率と同じと仮定する場合
> sel_3
      0      1      2      3
0.5172491 0.6712784 1.0000000 1.0000000
>
>
```

2018年の選択率を用いる場合

2019年の選択率を用いる場合

2017～2019年の3年平均の
選択率を用いる場合

それぞれの選択率

ここで指定した選択率の仮定を用いて、今度は**チューニングVPA**を行う



二段階目であるチューニングVPAの実行

8) VPAを走らせる (第二段階)

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ + + + + Go to file/funct
madara_reproduce.R x test-data.handler.R x
Source on Save
62
63
64 #チューニングありVPAの実行 (二段階法) ----
65
66 res <- vpa(dat,
67   tf.year=2017:2019,
68   Pope = TRUE,
69   tune = TRUE,
70   last.catch.zero = FALSE,
71   plus.group = TRUE,
72   p.init = 1,
73   term.F = "max",
74   sel.f = sel_1,
75   abund = c("N","N","N","N","N","N"), #チューニングの際の資源量指標値の属性
76   min.age=c(0,0,0,0,0,0), #チューニングの際の最低年齢
77   max.age=c(3,3,0,0,3,3), #チューニングの際の最高年齢
78   est.method = "ls",
79   b.est = FALSE
80 )
```

R上にこのように書いて、実行させる
(#以降はコメント文なので省略してよい)

動画VPA-01『**frasyrを用いたVPA：概要編**』で解説しているように、自分のしたい解析に合わせて以下のことを指定する：

指標値が資源量尾数なら “N”
資源重量なら “B”
産卵資源重量なら “SSB”

3番目の指標だけ “B” とかしたいときは `abund=c(“N” ,” N” ,” B” ,” N” ,” N” ,” N”)`

0 なら, (仮定している最低年齢)

つまり, 本解析では最低年齢は0歳で, ここが0ならチューニング指標の年齢参照下限は0歳。
もし, 解析で用いている最低年齢が1歳で, チューニング指標の年齢参照下限も1歳にしたいときも, ここはやはり0と指定 (このように **年齢そのものに該当しない場合があるので注意!**)
`abund= “SSB”` のときは, `min.age=NA` でよい

3 なら, (仮定している最低年齢+3歳) まで

つまり, 本解析では最低年齢は0歳で, ここが3ならチューニング指標の年齢参照上限は3歳。
もし, 解析で用いている最低年齢が1歳なら, ここが3ならチューニング指標の年齢参照上限は4歳に該当する。 `abund= “SSB”` のときは, `min.age=NA` でよい

二段階目であるチューニングVPAの実行 (続)

8) VPAを走らせる (第二段階) (続)

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ +R + Save Go to file/function Addins

madara_reproduce.R x test-data.handler.R x vpa.R x vpa_2_nidan.R x

Source on Save Run Source

```
62
63
64 #チューニングありVPAの実行 (二段階法
65
66 res <- vpa(dat,
67   tf.year=2017:2019,
68   Pope = TRUE,
69   tune = TRUE,
70   last.catch.zero = FALSE,
71   plus.group = TRUE,
72   p.init = 1,
73   term.F = "max",
74   sel.f = sel_1,
75   abund = c("N", "N", "N", "N", "N", "N"), #チューニングの際の資源量指標値の属性
76   min.age=c(0,0,0,0,0,0), #チューニング指標の年齢参照範囲の下限
77   max.age=c(3,3,0,0,3,3), #チューニング指標の年齢参照範囲の上限
78   est.method = "ls", #推定方法 (ls=最小二乗, ml=最尤法)
79   b.est = FALSE #bを推定するか否か
80 )
```

推定方法：最小二乗法なら“ls”，最尤法なら“ml”

上記の方法は，指標値の対数をとっている。

(例外として，対数をとらない指標値の絶対値の最小二乗法のオプションもある→ “ls_nolog”

VPAの結果をみる

9) VPAの結果をみる

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

Addins

madara_reproduce.R × test-data.handler.R × vpa.R × vpa_2_nidan.R ×

Source on Save

```
81
82
83 #チューニングありVPAの結果----
84
85 res$term.f #推定されたターミナルFの値をみる
86 res$faa    #推定された年別年齢別漁獲係数
87 res$naa    #推定された年別年齢別資源尾数
88 res$saa    #推定された年別年齢別選択率
89 res$baa    #推定された年別年齢別資源重量(バイオマス)
90 res$ssb    #推定された年別年齢別産卵親魚量
91
92
93 res$pred.index #モデルから推定された指標値
94 res$q          #推定された比例定数q
95 res$b          #資源量指標値の非線形性に関するパラメータb
96 res$logLik     #推定された対数尤度
97
98
99
100
```

ここを実行すると
推定された様々な値が確認出来る

Console

Jobs ×

~/勉強会/2020資源管理研修会/vpa_2_nidan/

> #チューニングありVPAの結果----

>

> res\$term.f #推定されたターミナルFの値をみる
[1] 0.3256853

> res\$faa #推定された年別年齢別漁獲係数
2012 2013 2014

推定された
ターミナルF

0.3731327 0.3610383 0.

0.5297792 0.5106938 0.

602

2 0.7025884 0.7011019 0.8389161 0.7908119 0.

222

3 0.7025884 0.7011019 0.8389161 0.7908119 0.

222

2017

2018

2019

2020

0 0.4503021 0.3576792 0.2705250 0.1824007

1 0.6041872 0.5322587 0.3667111 0.2396483

2 1.0107721 0.7434863 0.5939931 0.3256853

3 1.0107721 0.7434863 0.5939931 0.3256853

おまけ：resの中に入っているオブジェクト

前のスライドで紹介した代表的な結果以外にも、様々な情報を得ることが出来ます

res\$input	解析に用いたデータや仮定
res\$term.f	推定されたターミナルF
res\$np	推定されたターミナルFの数
res\$minimum	最適解における目的関数の値(合計)
res\$minimum.c	最適解における目的関数の値(個々)
res\$logLik	負の対数尤度
res\$gradient	最適解での傾き
res\$code	最適化法から返されるコード (どのような理由で最適化が停止したのかがわかる)
res\$q	推定されたq(資源量指標値の比例定数)
res\$b	推定されたb(資源量指標値の非線形性)
res\$sigma	資源量指標値の分散の平方根
res\$convergence	解が収束していれば1, そうでないと0
res\$message	最適化に関する注意(あれば)
res\$hessian	ヘッセ行列の値
res\$Ft	最終年のFの平均値
res\$Fc.at.age	Fcurrentで指定した年における平均のF
res\$Fc.mean	Fc.at.ageを平均したもの
res\$Fc.max	Fc.at.ageの最大値
res\$last.year	vpaを計算する最終年を別に指定した場合

res\$Pope	popeの近似式を用いたか否か
res\$ssb.coef	産卵親魚量の計算時期 (年始めなら0, 年中央なら0.5, 年最後なら1)
res\$pred.index	推定された資源量指標値
res\$wcaa	caa*waa.catch
res\$naa	推定された年別年齢別資源尾数
res\$faa	推定された年別年齢別漁獲係数
res\$baa	推定された年別年齢別資源重量
res\$ssb	推定された年別年齢別産卵親魚量
res\$saa	推定された年別年齢別選択率

※この情報はヘルプファイル(後述)でもみることが出来ます。

ヘルプファイルの活用

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ + + + + Go to file/function Addins

Console Jobs x

~/勉強会/2020資源管理研修会/vpa_2_nidan/

```
>  
> help(vpa)  
>
```

Files Plots Packages Help Viewer

R: VPAによる資源計算を実施する Find in Topic

```
penalty_age = NULL,  
no_eta_age = NULL,  
sdreport = FALSE,  
use.equ = "new"  
)
```

Arguments

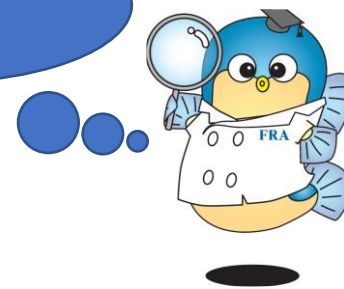
dat	data for vpa
sel.f	最終年の選択率
tf.year	terminal Fをどの年の平均にするか
rec.new	翌年の加入を外から与える
rec	rec.yearの加入

help(VPA)

と入力すると、

Helpというところに、先ほどと同じ内容
(resの中に入っているオブジェクト)
が紹介されますので、ご活用下さい

ヘルプファイルの使い方のついては、
動画R-05（関数（使い方編））
もご参照下さい



VPAの結果を可視化する (1)

※ 詳しくは、動画VPA-02「チューニングなしVPA」を参照

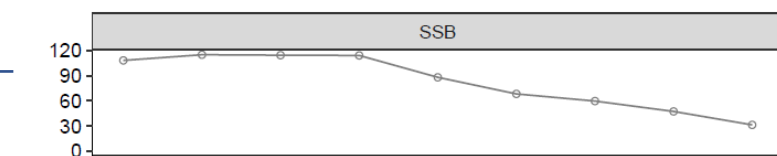
10) VPAの結果をプロットしてみる

```
33 #VPAの結果を可視化する
34 plot_vpa(res, plot_year=2012:2020)
```

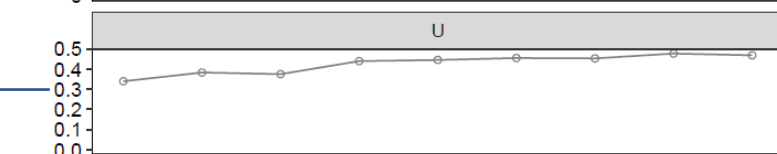
R上にこのように書いて、実行させる

2012年～2020年までの様々な結果をプロット (下図)

産卵親魚量



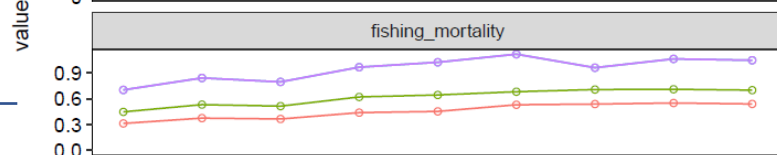
漁獲量/資源量



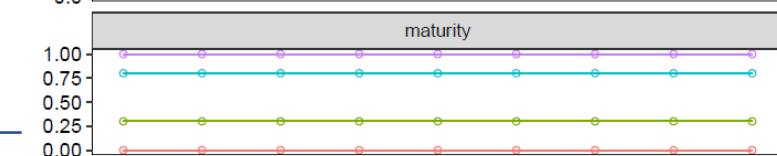
加入量



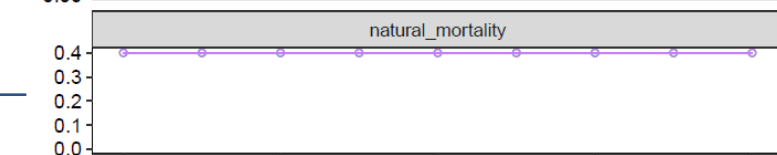
漁獲係数



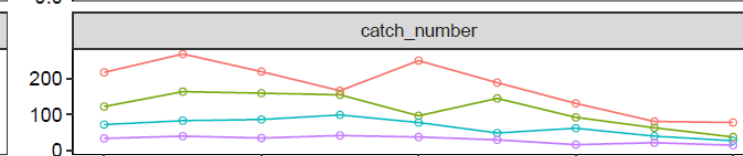
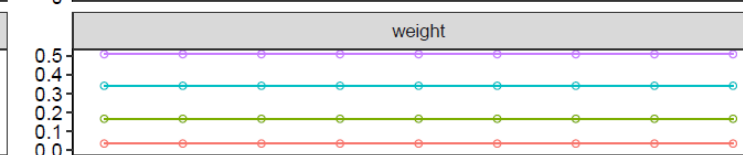
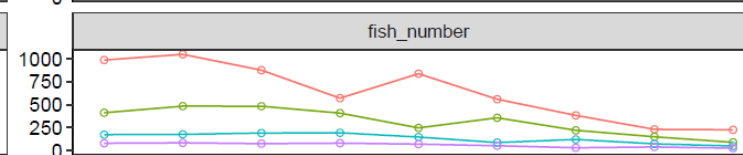
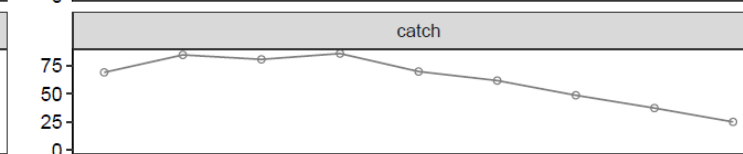
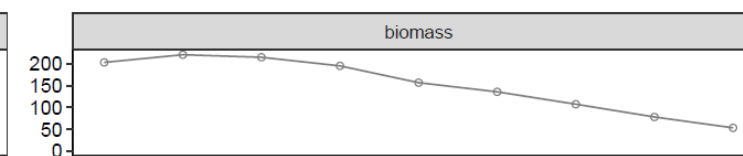
成熟率



自然死亡係数



id 1 age 0 1 2 3 NA



親魚量

漁獲量

資源尾数

体重

漁獲尾数

Year

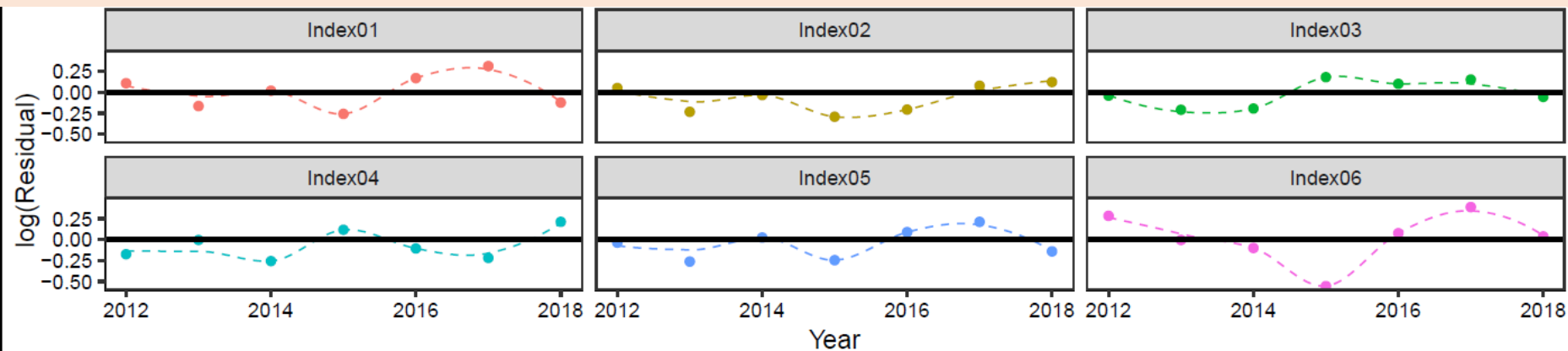
VPAの結果を可視化する (2)

11) 資源量指標値への当てはまりの単純な可視化

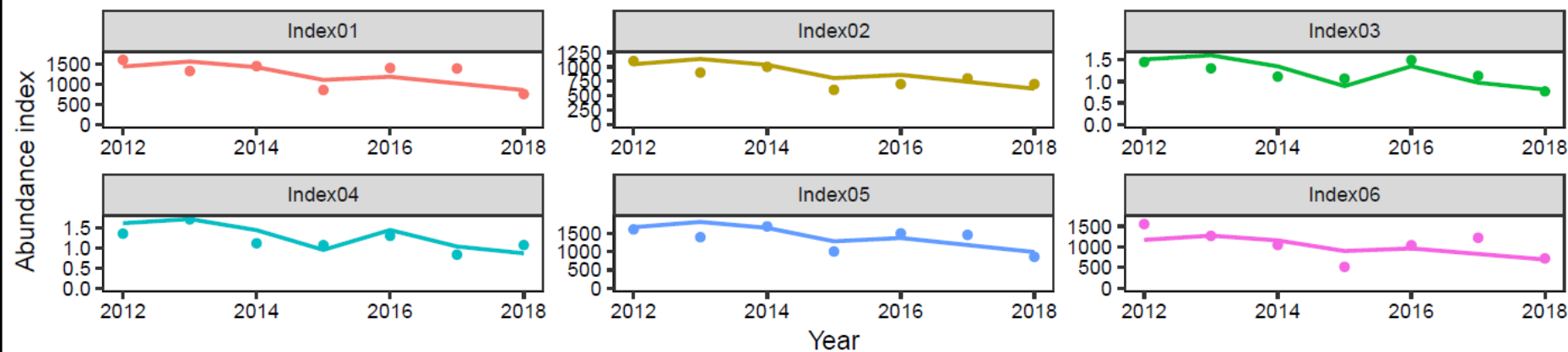
```
65  
66 res <- vpa(dat,  
67           tf.year=2017:2019,  
68           Pope = TRUE,  
69           tune = TRUE,  
70           last.catch.zero = FALSE,  
71           plus.group = TRUE,  
72           p.init = 1,  
73           term.F = "max",  
74           sel.f = sel_1,  
75           abund = c("N","N","N","N","N","N"), #チューニングの際の資源量指標値の属性  
76           min.age=c(0,0,0,0,0,0), #チューニング指標の年齢参照範囲の下限  
77           max.age=c(3,3,0,0,3,3), #チューニング指標の年齢参照範囲の上限  
78           est.method = "ls", #推定方法 (ls=最小二乗, ml=最尤法)  
79           b.est = FALSE, #bを推定するか否か  
80           plot = TRUE, #チューニングに使った資源量指標値に対するフィットのプロット  
81           plot.year =2012:2018 #上のプロットの参照年  
82 )
```

この2行を加えて実行すると、
次のスライドのような図が生成される

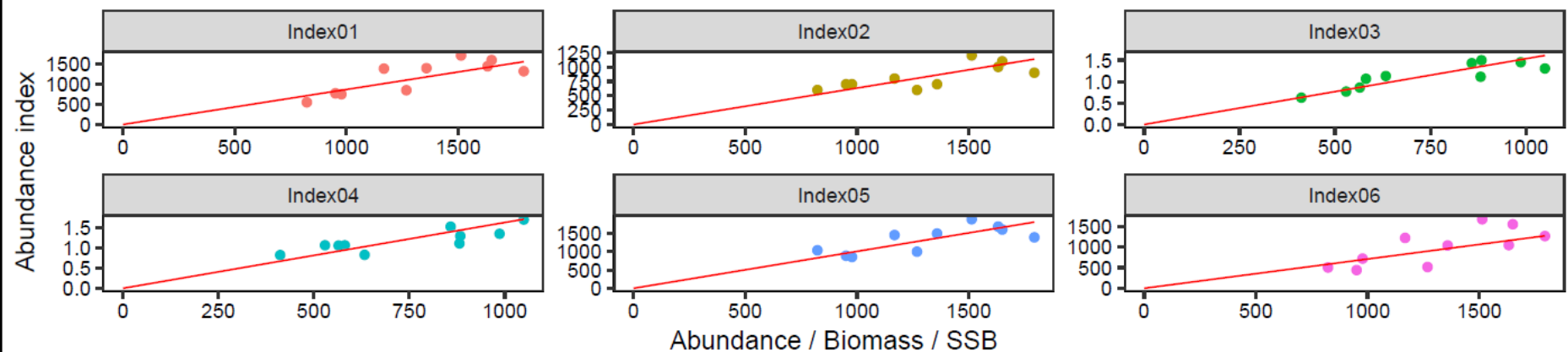
資源量指標値への当てはまり



指標値の残差の対数



点が観察された指標値
線が推定された資源尾数



横軸：推定された推定値
縦軸：観察された指標値
赤線が非線形性表す

この続きの，VPAのモデル診断に関しては，
動画VPA-04：VPAのモデル診断 概要編
動画VPA-05：VPAのモデル診断 実践編
を参照して下さい．



選択率更新法の場合の実行方法

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
+ + + + + Go to file/function
madara_reproduce.R x test-data.handler.R x
Source on Save
85 #チューニングありVPAの実行（選択率更新法） ----
86
87 res <- vpa(dat,
88           tf.year=2017:2019,
89           Pope = TRUE,
90           tune = TRUE,
91           last.catch.zero = FALSE,
92           plus.group = TRUE,
93           p.init = 1,
94           term.F = "max",
95           sel.update = TRUE,
96           sel.def = "max",
97           abun = c("N","N","N","N","N","N"),
98           min.age=c,
99           max.age=c,
100          est.method,
101          b.est = F,
102          )
```

R上にこのように書いて、実行させる
(#以降はコメント文なので省略してよい)

動画VPA-01『**frasyrを用いたVPA：概要編**』で解説しているように、
自分のしたい解析に合わせて指定する

選択率の定義の仕方を指定する**sel.def**には3つのオプションがある：

- “**max**” → 選択率が一番大きい年齢の選択率を1として、それを基準に選択率を決める
- “**maxage**” → 最高齢の選択率を1として、それを基準に選択率を決める
- “**mean**” → 全体に対する割合として選択率を決める (i.e., $saa = faa / \text{sum}(faa)$)

※実行後の結果の見方やプロットなどは先ほどの二段階法と同じ

全F推定法の場合の実行方法

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ + R + Save Print Go to file/function Addins

madara_reproduce.R x test-data.handler.R x vpa_2_nidan.R x vpa.R x

Source on Save Run Sour

```
104
105 #チューニングありVPAの実行（全F推定法）----
106
107 res <- vpa(dat, #VPAを行うデータ
108             #tf.year=2017:2019, #ターミナルFをどの年の平均にするか
109             Pope = TRUE, #Popeの近似式を使うならTRUE, Baranovを用いるならFALSE
110             tune = TRUE, #チューニング
111             last.catch.zero = FALSE, #最終年
112             plus.group = TRUE, #最高齢
113             p.init = c(0.2, 0.4, 0.8, 0.8), #各年齢のFの初期値
114             term.F = "all", #全年齢分のターミナルFを推定する
115             #sel.update = TRUE, #選択率更新
116             #sel.def = "mean", #選択率の定義
117             abund = c("N", "N", "N", "N", "N", "N"), #年齢別の資源量指標値の属
118             min.age=c(0,0,0,0,0,0), #チューニング指標の年齢参照範囲の下限
119             max.age=c(3,3,0,0,3,3), #チューニング指標の年齢参照範囲の上限
120             est.method = "ls", #推定方法 (ls=最小二乗, m=最大尤度)
121             b.est = FALSE) #bを推定するか否か
122
```

全F推定法では、全年齢分のターミナルFを推定するので、
term.F = "all" とする

1歳のFの初期値

さらに、推定するFの初期値を全年齢分与えたほうがよいので、
p.init = c(0.2, 0.4, 0.8, 0.8) のように年齢分与える

0歳のFの初期値

選択率更新法で必要だったsel.updateやsel.defの設定は必要ないので消す

※実行後の結果の見方やプロットなどは先ほどの二段階法と同じ

おまけ：vpa関数で指定できるその他のこと

今までのスライドで紹介した実行方法の設定は、基本的なものであり、その他にも様々な状況に合わせて、設定を変えることができます。ここに役に立ちそうないくつかのvpa関数の引数を紹介しておきます

引数	意味
rec.new	last.catch.zero=TRUEのときに、翌年の加入を外から与える
rec	rec.yearで指定した年における加入
rec.year	加入を代入する際の年
p.pope	Popeの近似式にした場合、どこで漁獲が起こるか(デフォルトは0.5)
link	資源量指標値がどういう関係式で資源量と関係しているかを調整 (link=“id”だったらlogをとった指標値がlogをとった資源量に比例, だったらlogをとった指標値がlog(log(資源量))に比例) link=“log”
p.m	Popeの近似式で、チューニングの際にどこで漁獲が起こるかを指定 (デフォルトは0.5で年の真ん中)
index.w	チューニングに用いる指標の重みを外部から与える
use.index	“all” だったら全ての資源量指標値を用い, c(2:3)だったら最低齢+1歳～最低齢+2歳まで使用
alpha	最高齢と最高齢-1のFの比 $F(a) = \alpha * F(a-1)$
stat.tf	最終年のFを推定する統計量, “mean” だったら平均値, “median” だったら中央値
no.est	TRUEならパラメータ推定せず, 初期値として与えたFのもとでvpaを行う (Excelとの比較に便利)

おまけ：vpa関数で指定できるその他のこと（続き）

引数	意味
q.fix	資源量指標値の比例定数qを外部から与える
b.fix	資源量指標値の非線形性に関するbを外部から与える．指標値の2番目と5番目だけbを1にして，あとは推定させたいならb.fix=c(NA, 1, NA, NA, 1, NA)と指定
optimizer	最適化の手法に何を使うか．デフォルトは “nlm” ． “nlminb” など指定できる
lambda	ridgeVPAを行う際のlambda（ペナルティーの大きさ）の指定
pen	<div>help(VPA)で同様の情報が参照できます</div>
	“s” = {最終年の年齢aのs－（tf.yearで指定した年のa歳の平均のs）} のbeta乗の和
TMB	TRUEにするとTMBを用いて高速計算行う．（事前にuse_rvpa_tmb()を実行） 全F推定法，POPE=TRUE，alpha=1，途中でプラスグループが変化しない場合のみのときに使用可能
sigma.constraint	sigmaパラメータの制約. 使い方としては，指標が5つあり，2番目と3番目の指標のsigmaは同じとしたい場合はc(1, 2, 2, 3, 4)と指定する
eta	ridgeVPAを行う際にFのpenaltyを分けて与えるときにeta.ageで指定した年齢への相対的なpenalty（0～1）
eta.age	ridgeVPAを行う際にFのpenaltyを分けるときにetaを与える年齢（0 = 0歳（加入）, 0:1 = 0～1歳）

これで、チューニングありVPA実践編（動画VPA-03）の解説は終わりです
こちらでご紹介したデータやコードはファイル名：vpa_03_data_codeにあります。
ありがとうございました

