VPA-II(2023)

frasyrを用いたridge-VPA実践編

国立研究開発法人
水産研究・教育機構

動画作成者　漁業情報解析部　宮川光代
(miyagawa_mitsuyo88@fra.go.jp)

これまでの実践編のふりかえり

動画名	内容
VPA-02（2020）	データの読み込み・チューニングなしVPA結果のプロットの仕方
VPA-03（2020）	二段階法、選択率後進法、全F推定法の実行
VPA-05（2020）	モデル診断
VPA-06（2021）	データの扱いとVPAの実行
VPA-07（2021）	最適化手法による推定結果の違い
VPA-08（2021）	資源量－資源量指標値間の非線形性と全F推定
VPA-09（2021）	より詳細なモデル

Frasyrにおける**設定方法**を主に解説

実データを用いて、その場で**実際にコードを実行しながら**の解説

今回：VPA-10（2023）frasyrを用いたリッジVPA**の実行の仕方**

frasyrでのridge-VPAの基本的な設定

資源管理研修VPA編スクリプト.Rmd

Knit on Save | Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

全F推定法の設定の一例

通常の全F推定法のVPA関数の設定に（動画VPA-03, VPA-08参照）

①lambda
②penalty
③beta

の3つの引数を追加する

引数の値はどのように決めればよいの？

frasyrでのridge-VPAの引数（lambda）

資源管理研修VPA編スクリプト.Rmd

Knit on Save　Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}|F_{a,Y}|^{\beta}$$

ペナルティー項

引数の説明：
①lambda：ペナルティーの程度をコントロールするパラメータ(0以上1以下)
【動画VPA-10にあるλに対応】
入力する値はautocalc_ridgevpa関数を使って別途探索する（後述）

frasyrでのridge-VPAの引数（penalty）

資源管理研修VPA編スクリプト.Rmd

Knit on Save　Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}\left|F_{a,Y}\right|^{\beta}$$

ペナルティー項

引数の説明：
②penalty：ペナルティー項の指定で、pかfかsの3種類から選択可能（デフォルトはp）

p：指定した年齢範囲のターミナルFのbeta乗の和

→ $+\lambda\sum_{a=1}^{A}\left|F_{a,Y}\right|^{beta}$

frasyrでのridge-VPAの引数（penalty=“f”）

1_script.R × 資源管理研修VPA編スクリプト.Rmd* ×

Knit on Save | Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "f", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}|F_{a,Y}|^{\beta}$$

ペナルティー項

f：{a歳のターミナルF－(tf.yearで指定した年のa歳の平均のF)}のbeta乗の和

→ $+\lambda\sum_{a=1}^{A}\left|F_{a,Y}-(1/n)\sum_{y=Y-n}^{Y-1}F_{a,y}\right|^{beta}$

frasyrでのridge-VPAの引数（penalty=“s”）

1_script.R × 資源管理研修VPA編スクリプト.Rmd ×

Knit on Save | Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "s", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}|F_{a,Y}|^{\beta}$$

ペナルティー項

S：{最終年のa歳の選択率S－(tf.yearで指定した年のa歳の平均のS)}のbeta乗の和

→ $+\lambda\sum_{a=1}^{A}\left|S_{a,Y}-(1/n)\sum_{y=Y-n}^{Y-1}S_{a,y}\right|^{beta}$

frasyrでのridge-VPAの引数（beta）

資源管理研修VPA編スクリプト.Rmd

Knit on Save　Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}|F_{a,y}|^{\beta}$$

ペナルティー項

引数の説明：
③beta:正則化の種類（1ならラッソ回帰、2ならリッジ回帰）
【動画VPA-10にあるβに対応】
デフォルトは2のリッジ回帰

資源管理研修VPA編スクリプト.Rmd

Knit on Save | Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

ridge VPAを実行する上で
大事な3つの引数の設定に
ついては理解できました

lambdaの値はどのよう
に探索すればよいの？

lambdaの探索をしてくれる自動化関数

autocalc_ridgevpaという関数

資源管理研修VPA編スクリプト.Rmd

Knit on Save | Knit

Source | Visual

```
675
676
677  ridge_res<-autocalc_ridgevpa(
678                input = res5.1$input,
679                target_retro="F",
680                n_retro=5,
681                b_fix=TRUE,
682                bin=0.1)
683
```

TMB=TRUEとする場合は，事前にTMBのパッケージをインストールし，
library(TMB)
use_rvpa_tmb()
の2行を実行しておく必要がある

引数の説明と設定：
①input ： vpa関数の引数をリスト形式で与える．このとき，TMBという引数がTRUEになっていたほうが計算が比べものにならないほど早くなるので，TMB=TRUEとしておくことをお勧め．ただし，TMB=TRUEで計算できないケースもあるので，その場合は，TMB＝FALSEで時間をかけて計算するしかない（TMB＝TRUEが使えるのは，全F推定法，POPE＝TRUE，alpha=1，プラスグループが途中で変わらない場合など現状では制限があるため）

```
res5.1 <- vpa(dat,
              tf.year = 2015:2017,
              tune = TRUE,
              term.F = "all",
              alpha = 1,
              abund = c("N","N","SSB","SSB"),
              min.age = c(0,0,0,0),
              max.age = c(0,0,6,6),
              est.method="ml", #重み推定（最尤法）
              b.est = TRUE, #b推定
              sel.def="max",
              #b.fix = c(1,1,NA,NA),
              last.catch.zero=TRUE,
              fc.year=2016:2018,
              plot = TRUE,
              plot.year = 2002:2018,
              use.index =1:4,
              sigma.constraint = c(1,1,2,2),
              TMB = TRUE
)
```

lambdaの探索をしてくれる自動化関数

autocalc_ridgevpaという関数

資源管理研修VPA編スクリプト.Rmd

Knit on Save　Knit

Source　Visual

```
675
676
677  ridge_res<-autocalc_ridgevpa(
678               input = res5.1$input,
679               target_retro="F",
680               n_retro=5,
681               b_fix=TRUE,
682               bin=0.1)
683
```

実行

引数の説明と設定：

②target_retro ：レトロスペクティブバイアスを何について計るか（mohn’s rhoのパラメータθのこと）　→　FかBかSSBかNかRを選択

おさらい:Mohn’s rho(モーンズロー) ρ

資源管理研修動画 VPA-06（2021）を参照

$$\rho = \frac{1}{P}\sum_{i=1}^{P}\left(\frac{\hat{\theta}_{Y-i}^{R_i} - \hat{\theta}_{Y-i}}{\hat{\theta}_{Y-i}}\right)$$

i年分のデータを除去したときのターミナル年の推定値

データを除去する年数

Y-i年におけるフルモデル（全年数分のデータを用いたモデル）の推定値

③n_retro ：レトロスペクティブ解析で遡る年数。
デフォルトは`5`

④b_fix ：レトロスペクティブ解析内でbを固定するか。
デフォルトは`TRUE`

⑤bin ： lambdaの探索の幅

⚠ 注意：VPAの計算を何度も繰り返し行うので，TMB=TRUEでない場合はかなりの計算時間がかかります

lambdaの探索

autocalc_ridgevpaという関数の中で行っていること

ステップ1:

λに0～1の間の値をbinで指定した幅で与え，n_retro年遡ったときのモーンズρの値を計算

計算途中で，今計算しているλの値と
そのときの各指標のモーンズρの値が，このように途中経過としてprint outされる

```
[1] 0.1
          N           B         SSB           R           F
0.12513035  0.14663842  0.24735364 -0.05935824 -0.13157609
[1] 0.2
          N           B         SSB           R           F
0.12494223  0.14767380  0.26428089 -0.05733478 -0.17138000
[1] 0.3
          N           B         SSB           R           F
0.13086423  0.15461534  0.28383689 -0.05143401 -0.19737257
[1] 0.4
          N           B         SSB           R           F
0.14149972  0.16648803  0.30818727 -0.04276048 -0.21952121
```

lambdaの探索

autocalc_ridgevpaという関数の中で行っていること

ステップ2:

Target_retroで指定した指標のモーンズρが一番小さくなるλの値の前後のbinの間を
0.01刻みでさらに細かく調べてモーンズρの値を計算する

⋮

```
[1] 0.11
          N           B         SSB           R           F
0.12465595  0.14632502  0.24900617 -0.05945729 -0.13697280
[1] 0.12
          N           B         SSB           R           F
0.1243214   0.1461369   0.2506546 -0.0594626 -0.1418933
[1] 0.13
          N           B         SSB           R           F
0.12410669  0.14605730  0.25230558 -0.05938823 -0.14642676
[1] 0.14
          N           B         SSB           R           F
0.12399586  0.14607285  0.25396769 -0.05924446 -0.15064027
```

⋮

この例では，ステップ1でFに関して最小のモーンズρを与えたのはλ=0.1のときだったので，0から0.2まで0.01刻みでさらに細かくモーンズρを計算している

```
> ridge_res
$min_penalty
[1] 0.01

$plot

$lambda_mat1
   lambda        mohn   delta_mohn
1     0.0 474.1689709 474.03739476
2     0.1  -0.1315761   0.00000000
3     0.2  -0.1713800   0.03980391
4     0.3  -0.1973726   0.06579648
5     0.4  -0.2195212   0.08794512
6     0.5  -0.2413027   0.10972659
7     0.6  -0.2651116   0.13353548
8     0.7  -0.2936536   0.16207753
9     0.8  -0.3310016   0.19942552
10    0.9  -0.3840003   0.25242417
11    1.0  -0.2523107   0.12073460

$lambda_mat2
   lambda          mohn    delta_mohn
1    0.00 474.16897085 4.741435e+02
2    0.01   0.02543685 0.000000e+00
3    0.02  -0.02971799 4.281145e-03
4    0.03  -0.05829535 3.285851e-02
5    0.04  -0.07713443 5.169758e-02
6    0.05  -0.09105232 6.561547e-02
7    0.06  -0.10204489 7.660805e-02
8    0.07  -0.11111651 8.567967e-02
9    0.08  -0.11884694 9.341009e-02
10   0.09  -0.12558923 1.001524e-01
11   0.10  -0.13157609 1.061392e-01
12   0.11  -0.13697280 1.115360e-01
13   0.12  -0.14189333 1.164565e-01
14   0.13  -0.14642676 1.209899e-01
15   0.14  -0.15064027 1.252034e-01
16   0.15  -0.15457621 1.291394e-01
17   0.16  -0.15828501 1.328482e-01
18   0.17  -0.16179441 1.363576e-01
19   0.18  -0.16513219 1.396953e-01
20   0.19  -0.16832072 1.428839e-01
21   0.20  -0.17138000 1.459432e-01
```

autocalc_ridgevpaの出力結果

①**$min_penalty**: target_retroで指定した指標のモーンズρが最小になるようなλの値

②**$lambda_mat1**:

lambda ： **ステップ1**におけるλの値

mohn ： target_retroで指定した指標のモーンズρの値

delta_mohn ： 最小のmohnの値との差（つまりここが0なのが最もモーンズρが低い）

②**$lambda_mat2**:

lambda ： **ステップ2**におけるλの値

ここのdelta_mohnが0になっているlambdaが最もモーンズρが低くなる

↳ 原理的には$min_pentyで得られたλの値をvpa関数のlambda引数に代入

※target_retroで選択していない指標のモーンズρが極端に悪くなっていないかなども確かめながら，総合的に選ぶことが大事です

frasyrでのridge-VPAの引数（lambda）

資源管理研修VPA編スクリプト.Rmd

Knit on Save　Knit

Visual

```
res5.1_ridge <- vpa(dat,
                    tf.year = 2015:2017,
                    tune = TRUE,
                    term.F = "all",
                    alpha = 1,
                    abund = c("N","N","SSB","SSB"),
                    min.age = c(0,0,0,0),
                    max.age = c(0,0,6,6),
                    est.method="ml", #重み推定（最尤法）
                    b.est = TRUE, #b推定
                    sel.def="max",
                    #b.fix = c(1,1,NA,NA),
                    last.catch.zero=TRUE,
                    fc.year=2016:2018,
                    plot = TRUE,
                    plot.year = 2002:2018,
                    use.index =1:4,
                    sigma.constraint = c(1,1,2,2),
                    lambda = 0.01, #ridge penaltyの大きさ
                    penalty = "p", #penalty項の与え方
                    beta =2  #penaltyの種類：1=lasso, 2=ridge
)
```

新：正則化ありの負の対数尤度

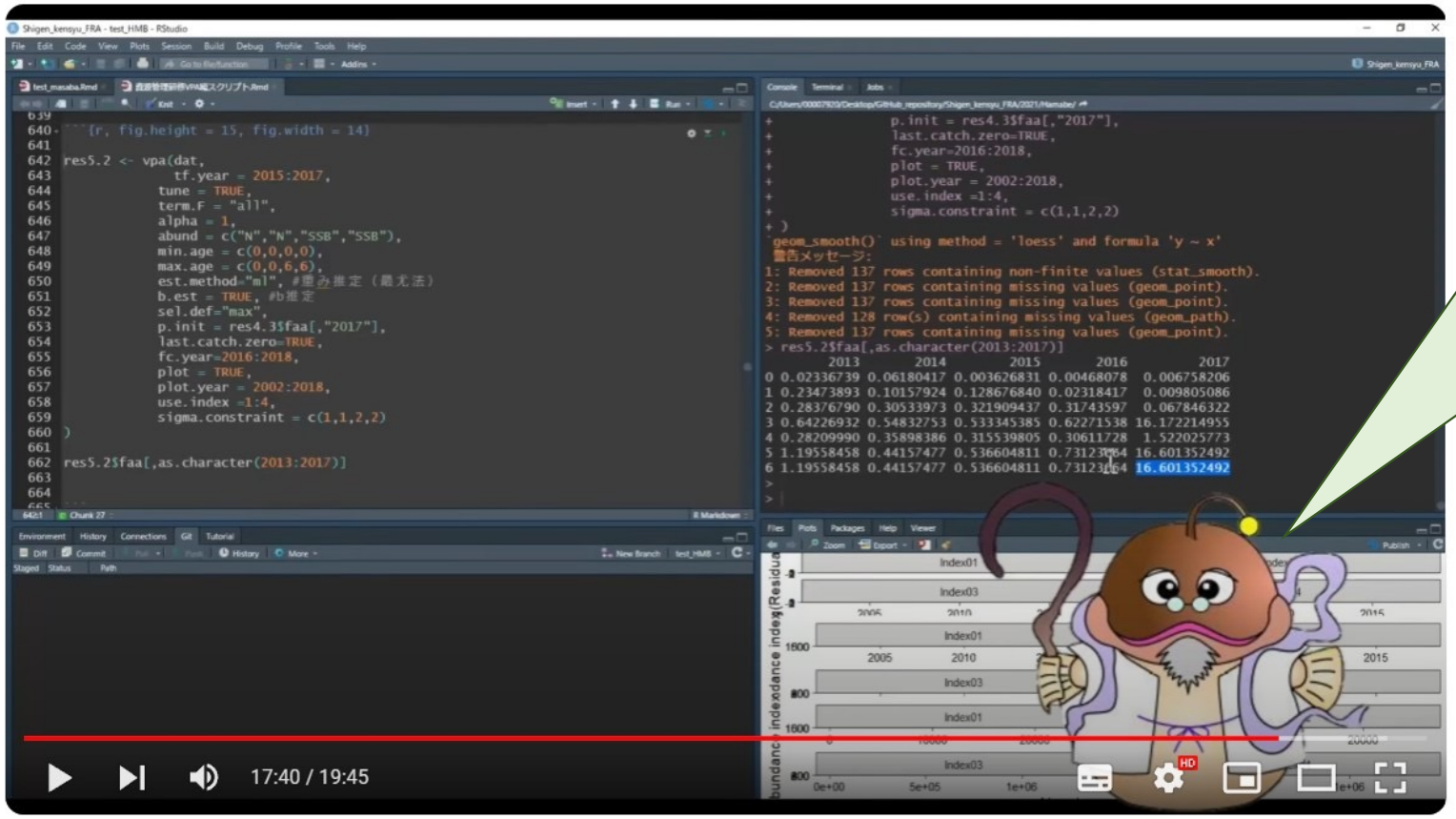
$$(1-\lambda)\sum_{k=1}^{K}\sum_{y=1}^{Y}\left[\log(\sigma_k)+\frac{\left(I_{k,y}-\log(q_k N_y)\right)^2}{2\sigma_k^2}\right]+\lambda\sum_{a=1}^{A}|F_{a,Y}|^{\beta}$$

ペナルティー項

引数の説明：
①lambda：ペナルティーの程度をコントロールするパラメータ(0以上1以下)
【動画VPA-10にあるλに対応】
入力する値はautocalc_ridgevpa関数を使って別途探索する（後述）

こちら

Ridge-vpaの効果の確かめ



第8回 実データを用いたfrasyrによるVPA解析③　－資源量－資源量指標値間の非線形性と全F推定－

限定公開

FRA　水産研究・教育機構
チャンネル登録者数 983人

チャンネル登録　3　共有　クリップ

以前の動画VPA-08(2021)【実データを用いたfrasyrによるVPA解析③】の全F推定のところで，一部の年齢のターミナルFが大きくなってしまうという問題があり，その解決にはリッジVPAという手法があると解説

Ridge-vpaの効果の確かめ(1)

Ridge-vpaなしの場合のres5.1の全F推定法の直近5年の年齢別Fの推定値

```
> res5.1$faa[,as.character(2013:2017)]
       2013       2014        2015        2016         2017
0 0.0233673 0.06180418 0.003626864 0.004680797 6.758223e-03
1 0.2347387 0.10157883 0.128676848 0.023184386 9.805120e-03
2 0.2837677 0.30533928 0.321907833 0.317435997 6.784699e-02
3 0.6422689 0.54832694 0.533344175 0.622710344 1.615977e+02
4 0.2820997 0.35898353 0.315539287 0.306116236 1.521987e+00
5 1.1955841 0.44157438 0.536604017 0.731234516 1.242000e+01
6 1.1955841 0.44157438 0.536604017 0.731234576 1.242000e+01
```

最終年の3歳，5歳，6歳のFが非現実的に大きい

Ridge-vpaありの場合のres5.1_ridgeの全F推定法の直近5年の年齢別Fの推定値

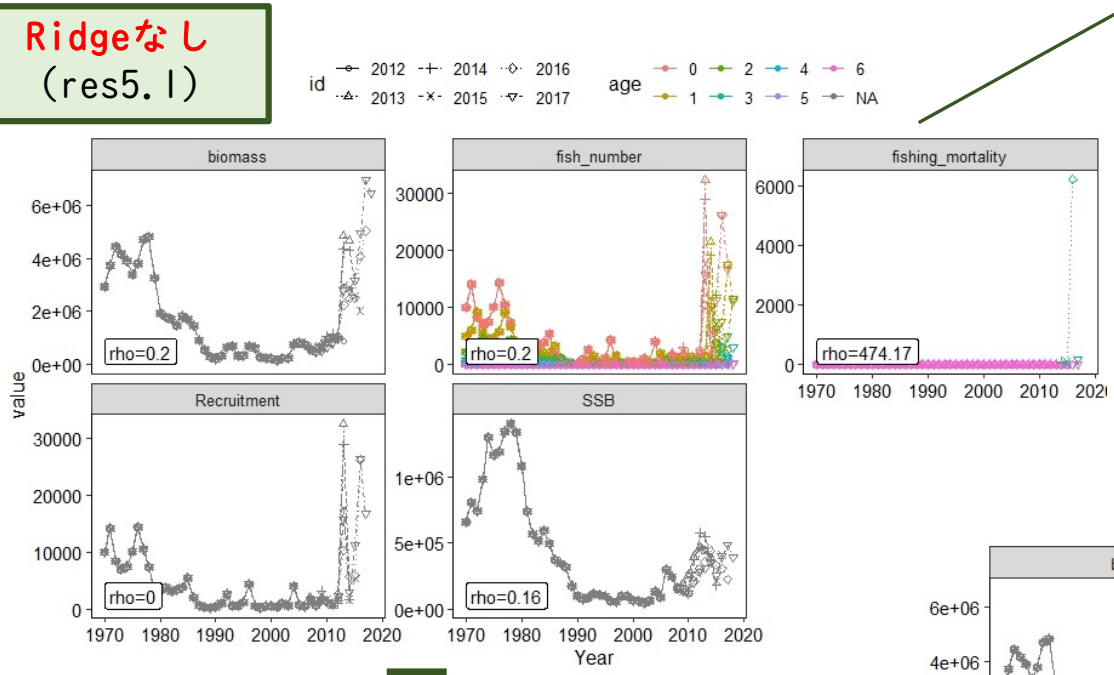
```
> res5.1_ridge$faa[,as.character(2013:2017)]
       2013       2014        2015        2016        2017
0 0.02248816 0.05852434 0.004124662 0.004835536 0.006945882
1 0.22523411 0.09751916 0.121202196 0.026415544 0.010131698
2 0.27650241 0.28932339 0.306104555 0.294655509 0.077810159
3 0.62871310 0.52715286 0.491677328 0.574666007 2.491217522
4 0.27563432 0.34678794 0.297291116 0.271368713 1.212711194
5 1.17712812 0.42712399 0.508462066 0.661697508 2.043979270
6 1.17712812 0.42712399 0.508462066 0.661697608 2.043979270
```

まだ少し高めだが，λ=0.01でリッジVPAを行うと，Fがだいぶ現実的な値になった

※　配布したVPA-II_script.Rを実行してみてください

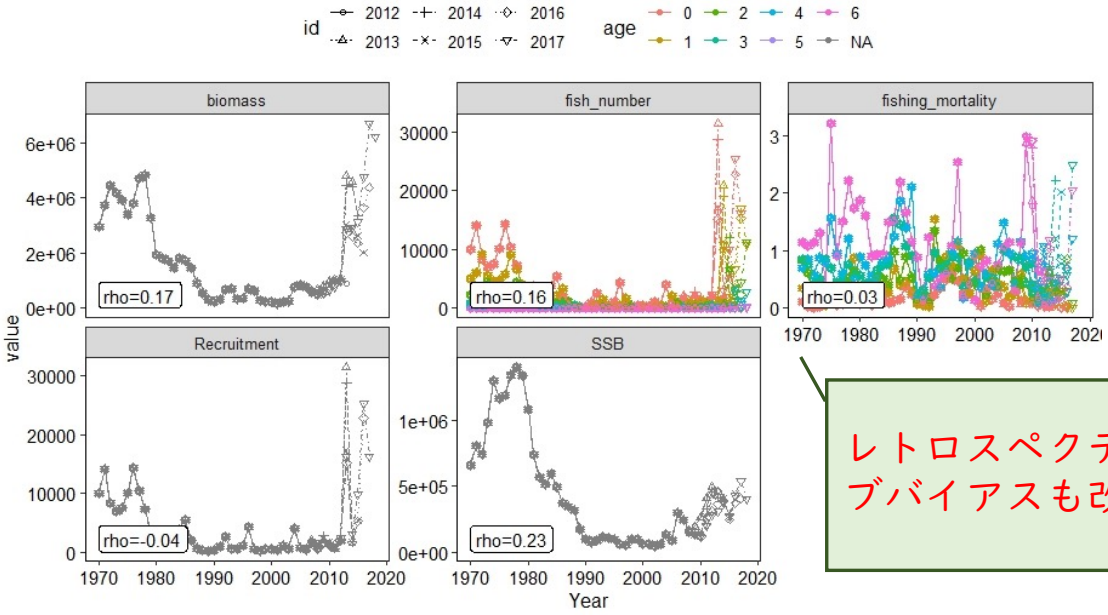
Ridge-vpaの効果の確かめ（2）

Ridgeなし
(res5.1)




```
retro_5.1 <- do_retrospective_vpa(res5.1,
                                  n_retro=5,
                                  b_reest =FALSE)
```

Ridgeあり
(res5.1_ridge)




```
retro_5.1ridge <- do_retrospective_vpa(res5.1_ridge,
                                       n_retro=5,
                                       b_reest =FALSE)
```

レトロスペクティブバイアスも改善

frasyrでのridge-VPAの特殊な設定(etaの設定)

ペナルティー項の重みを年齢によって変える場合（例：マイワシ太平洋，スケトウダラ太平洋など）

```
res5.1_ridge2 <- vpa(dat,
                     tf.year = 2015:2017,
                     tune = TRUE,
                     term.F = "all",
                     alpha = 1,
                     abund = c("N","N","SSB","SSB"),
                     min.age = c(0,0,0,0),
                     max.age = c(0,0,6,6),
                     est.method="ml", #重み推定（最尤法）
                     b.est = TRUE, #b推定
                     sel.def="max",
                     #b.fix = c(1,1,NA,NA),
                     last.catch.zero=TRUE,
                     fc.year=2016:2018,
                     plot = TRUE,
                     plot.year = 2002:2018,
                     use.index =1:4,
                     sigma.constraint = c(1,1,2,2),
                     lambda = 0.01, #ridge penaltyの大きさ
                     penalty = "p", #penalty項の与え方
                     beta =2,  #penaltyの種類：1=lasso, 2=ridge
                     eta=0.99, #penaltyを年齢で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=0, #penaltyを年齢で分けるときにetaを与える年齢(0 = 0歳（加入）,0:1 = 0~1歳)
                     TMB =TRUE
)
```

etaとeta.ageを指定する

系群名	ペナルティー項（λとη）
マイワシ太平洋	$(1-\lambda)\sum_{k=1}^{3}\sum_{y}\left[\ln(I_{k,y})-\ln(q_k X_{k,y}^{b_k})\right]^2+\lambda\left[(1-\underline{\eta})\sum_{a=1}^{4}F_{a,2021}^2+\underline{\eta}F_{0,2021}^2\right]$
スケトウダラ太平洋	$(1-\lambda)\mathrm{lnL}+\alpha\lambda\left[(1-\underline{\eta})\Sigma_{a=4}^{9}F_{a,Y}^2+\underline{\eta}F_{3,Y}^2\right]$

マイワシ太平洋：親魚量のレトロバイアスを小さくすると、加入量のレトロバイアスが大きくなるというトレードオフ
（解決策）→ペナルティに対する重みを1歳以上（λ）と0歳魚（η）で変えた

スケトウ太平洋：3歳のFのレトロバイアスが特に強い
（解決策）→ペナルティに対する重みを4歳以上（λ）と3歳魚（η）で変えた

VPA-10(2023)参照

適切なetaとlambdaの探索は， autocalc_ridgevpaを用いて探索可能

リッジVPA実践編のまとめ

1. vpa関数の引数としてlambda，penalty，betaの値を設定する
2. レトロバイアスを小さくするようなlambdaの値はautocalc_ridgevpa関数で探索可能
3. do_retrospective_vpa関数を用いてレトロスペクティブ解析を行い，レトロバイアスに問題がないことを確認する
4. レトロバイアスにトレードオフなどがみられる場合は，ペナルティに対する重みを年齢によって変えるetaを導入したりして工夫する

本動画では，標準的なリッジVPAの実践法について解説しました．動画VPA-10で説明したように，一部の系群では，λの選択にモーンズρ以外の基準を用いていたり，また特殊なペナルティー項を与えている場合もあり，そのような特殊な場合に現行のfrasyrは対応していないため，個々のケースに合わせて，frasyrのコードを書きかえる必要があることに注意してください！
※frasyrは日進月歩で改良が進められているので利用の際には
https://github.com/ichimomo/frasyrに記載のあるバージョンと更新情報を確認してください．本動画は2023年12月現在のfrasyrをもとにして作成されています．