

D. Rの基礎知識

Rの基礎知識

- Rとは？
- エディタ：Rstudio
- Rプログラムの基本形（関数・引数・オブジェクト）
- ワーキングディレクトリの設定
- 「生のR」と「tidyverse」の違い(応用)

Rとは



You

プログラミング言語のRとはどのような言語ですか？



ChatGPT

Rは統計計算やデータ解析のために設計されたプログラミング言語および環境です。以下は、Rの主な特徴です：

1. **統計分析とデータ処理**: Rは統計学的な計算やデータ解析のために広く使用されます。統計的手法や機械学習アルゴリズムを実装し、データの可視化や処理を行うことができます。
2. **オープンソース**: Rはオープンソースであり、無料で利用できます。これは、コミュニティによって広くサポートされ、拡張や改良が進んでいることを意味します。
3. **豊富なパッケージとライブラリ**: Rにはさまざまな統計的手法やデータ可視化のためのパッケージやライブラリが豊富に存在します。これらのパッケージを使用することで、特定のタスクに対する高度な機能を追加することができます。
4. **ベクトル演算**: Rはベクトル演算をサポートしており、同じ操作を複数のデータポイントに対して一度に実行できます。これにより、効率的かつ簡潔なコードの記述が可能です。
5. **グラフィカルなデータ表示**: Rはデータの視覚化に強みがあり、さまざまなグラフやプロットを作成することができます。ggplot2などのパッケージを使用することで、高度なデータ可視化が可能です。
6. **スクリプティング言語**: Rはスクリプティング言語であり、コードをスクリプトとして保存し、再利用することができます。これにより、繰り返しのタスクやデータ分析の自動化が容易になります。

Rは主に統計家、データサイエンティスト、研究者、およびデータ解析を行う他の分野の専門家によって広く利用されています。



我が国の資源評価のためのパッケージ(frasyr, frasyr23, frapmr)

- 資源計算、再生産関係推定、ABC計算、結果の図の作成

一連の手順を統一式スクリプトにして配布 (frasyr_tool)



Rによる資源計算の演習

エディタ (Rstudio) : スクリプトを書き実行するためのソフ

You
スクリプトって何ですか？

ChatGPT
スクリプト (Script) は、一連のコンピュータプログラムの命令や手順がテキスト形式で記述されたファイルのことを指します。スクリプトは通常、特定のタスクや機能を実行するためのプログラムであり、解釈型の言語で書かれることがよくあります。スクリプトは通常、人が理解しやすく、テキストエディタで開いて編集できる形式で保存されます。

The screenshot shows the RStudio environment with several components and annotations:

- Source Editor:** Contains R code for a simulation. A blue box highlights the code area with the text "スクリプトを書く場所" (Place to write the script).
- Environment Pane:** Shows the current environment with objects like 'all_yp', 'Fmat', and 'x'. A blue box highlights the list of objects with the text "作ったオブジェクトのリストなど" (List of created objects, etc.).
- Console:** Shows the output of the R script. A blue box highlights the console area with the text "コードを実行する場所" (Place to execute the code).
- Plots Pane:** Shows a plot of the simulation results. A blue box highlights the plot area with the text "プロットやファイル操作、パッケージ管理など" (Plots, file operations, package management, etc.).

The R code in the Source Editor is as follows:

```
1 waa <- c(0.2, 4.4, 16.1, 34.5, 58.4, 85.2)
2 maa <-
3 M <- c(
4
5 # YPRと
6 YPRを返
7 calc_yp
8 ypr <
9 number
10 for(i in 1:5){
11   number[i+1] <- number[i]*exp(-FA[i]-M[i])
12   number0[i+1] <- number0[i]* exp(-M[i])
13 }
```

Rプログラムの基本形

```
> sumtest1 <- sum(1:10)
```

オブジェクト
(生成されたデータ
や情報が入ってい
る)

<-

代入記号
(=も可)

関数と引数
(命令)

- 関数が実行されて、無事代入が終わると、「オブジェクト」ができる
- 「オブジェクト」には関数の実行結果が保存されている
- 「引数」は関数に与える様々な「オプション」
- 作ったオブジェクトを別の関数に入れて再計算を繰り返す手順＝スクリプト

💡 ポイント 💡 使おうとする関数が何をするための関数なのか？
どんな引数（オプション）があるのか？どんな結果を返すのか？を知ることが
基本

Rの「オブジェクト」の型

```
x1 <- c(2,3,4,5) # ベクトル
x2 <- matrix(1:6,2,3) # 行列
x3 <- data.frame(x=1:2, y=c("a","b")) # データフレーム
x4 <- tibble(x=1:2, y=c("a","b")) # データフレーム(tidyverse)
x5 <- list(x1, x2, x3, x4) # リスト
x6 <- lst(x1, x2, x3, x4) # リスト(tidyverse)
```

- 複雑な計算をする関数の返り値はたいてい「リスト」の中に、いろいろな型のオブジェクトがつまってる
- リストの中の要素を見るには、\$を使う （リストx6の中のx1を見たい → x6\$x1）
- names(x6)とすれば、リストの中にどんな名前のオブジェクトが入ってるか確認できる

演習① 次の関数を使い、結果をオブジェクトに入れよう

1. sum (足し算をする)

- 1から10の足し算をし、結果をオブジェクトに格納

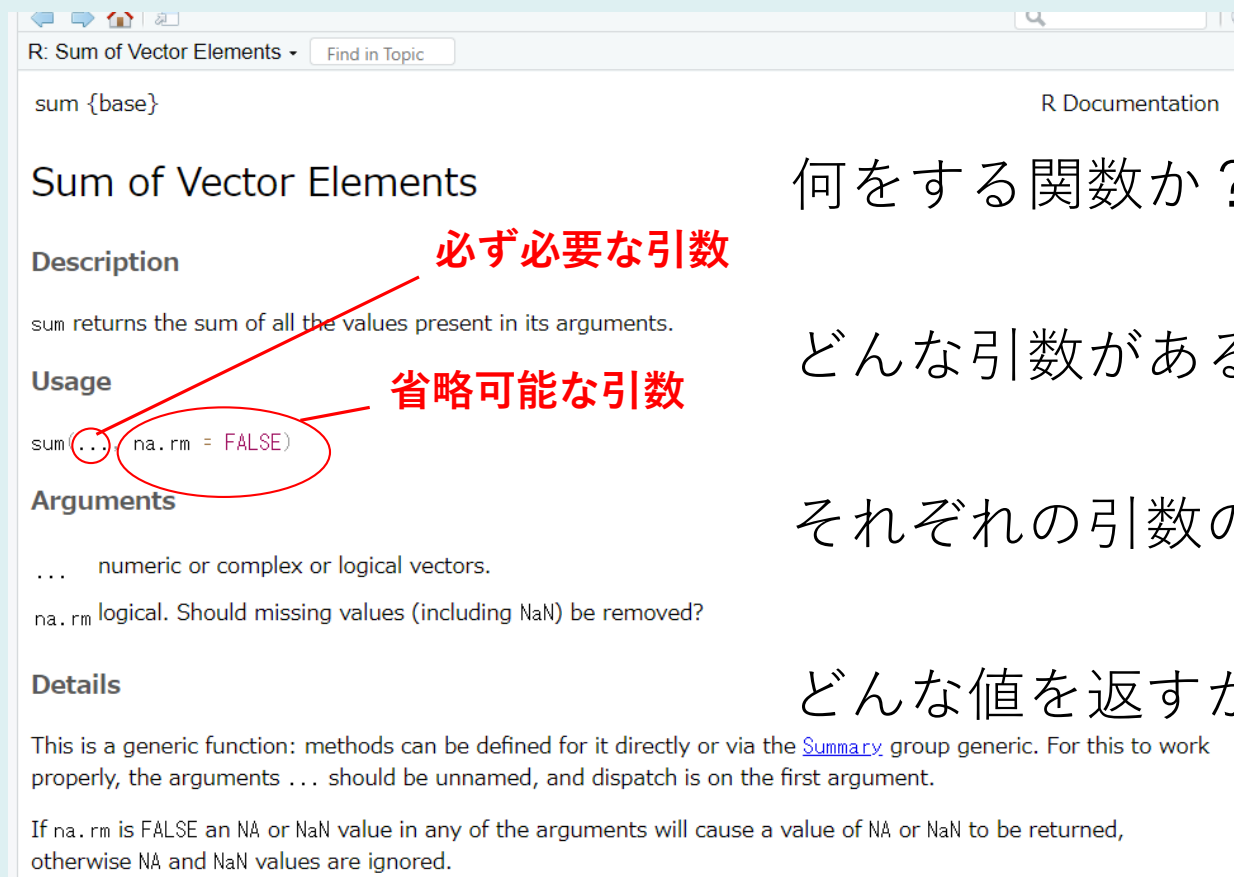
2. rnorm (正規乱数を発生させる)

- 正規乱数を100個発生させ、結果をオブジェクトに格納

3. hist (ヒストグラムを作成する)

- 上で発生させた正規乱数をヒストグラムとしてプロット

Helpの見方 (sum)



The screenshot shows the R documentation for the `sum` function. The title is "Sum of Vector Elements". The "Description" section states: "sum returns the sum of all the values present in its arguments." The "Usage" section shows the function signature: `sum(..., na.rm = FALSE)`. A red circle highlights the `...` argument, with a red arrow pointing to it from the text "必ず必要な引数" (Required argument). Another red arrow points from the text "省略可能な引数" (Optional argument) to the `na.rm = FALSE` argument. The "Arguments" section describes `...` as "numeric or complex or logical vectors" and `na.rm` as a logical value indicating whether missing values should be removed. The "Details" section explains that `sum` is a generic function and that if `na.rm` is `FALSE`, NA or NaN values will cause the function to return NA or NaN.

R: Sum of Vector Elements ▾ Find in Topic

sum {base}

R Documentation

Sum of Vector Elements

Description

sum returns the sum of all the values present in its arguments.

Usage

```
sum(..., na.rm = FALSE)
```

Arguments

... numeric or complex or logical vectors.

na.rm logical. Should missing values (including NaN) be removed?

Details

This is a generic function: methods can be defined for it directly or via the [Summary](#) group generic. For this to work properly, the arguments ... should be unnamed, and dispatch is on the first argument.

If na.rm is FALSE an NA or NaN value in any of the arguments will cause a value of NA or NaN to be returned, otherwise NA and NaN values are ignored.

何をする関数か？ (Description)

どんな引数があるか？ (Arguments)

それぞれの引数の説明 (Detail)

どんな値を返すか？ (Value)

helpの見方 (rnorm)

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Arguments

<code>x, q</code>	vector of quantiles.
<code>p</code>	vector of probabilities.
<code>n</code>	number of observations. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>mean</code>	vector of means.

helpの見方 (hist)

Description

The generic function `hist` computes a histogram of the given data values. If `plot = TRUE`, the resulting object of [class](#) "histogram" is plotted by [plot.histogram](#), before it is returned.

Usage

```
hist(x, ...)
```

```
## Default S3 method:
```

```
hist(x, breaks = "Sturges",  
     freq = NULL, probability = !freq,  
     include.lowest = TRUE, right = TRUE,  
     density = NULL, angle = 45, col = "lightgray", border = NULL,  
     main = paste("Histogram of" , xname),  
     xlim = range(breaks), ylim = NULL,  
     xlab = xname, ylab,  
     axes = TRUE, plot = TRUE, labels = FALSE,  
     nclass = NULL, warn.unused = TRUE, ...)
```

Arguments

<code>x</code>	a vector of values for which the histogram is desired.
<code>breaks</code>	one of:

ワーキングディレクトリ(WD)とは？

You
ワーキングディレクトリとは何ですか？

ChatGPT
ワーキングディレクトリ（Working Directory）は、コンピュータ上で現在作業しているディレクトリ（フォルダ）のことを指します。プログラミングやデータ分析の文脈では、スクリプトやプログラムがファイルの読み込みや書き込み、実行などの操作を行う基準となるディレクトリとして重要です。

もしこのディレクトリで作業しているなら（=WDがC : ¥Users¥00006909）、ファイルの保存先も必要なファイルを探す時も、このフォルダを起点に行うよ

C :

C : ¥Users



C : ¥Users¥00006909

```
> getwd()  
[1] "C:/Users/00006909"
```

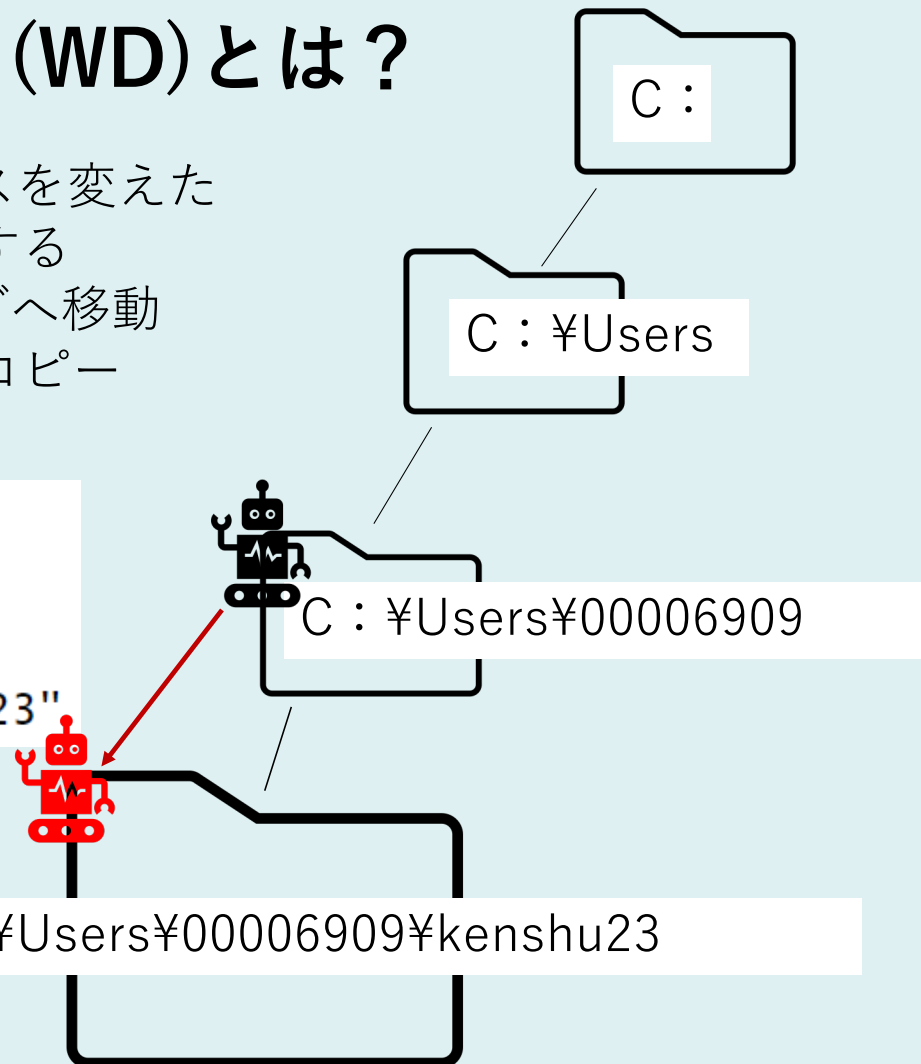
C : ¥Users¥00006909¥kenshu23

ワーキングディレクトリ(WD)とは？

- 仕事の内容によって、作業スペースを変えたい
1. その仕事用のフォルダを作成する
 2. setwdコマンドで、そのフォルダへ移動
 3. 必要ファイルをそのフォルダにコピー

```
> getwd()  
[1] "C:/Users/00006909"  
> setwd("kenshu2023")  
> getwd()  
[1] "C:/Users/00006909/kenshu2023"
```

※ウィンドウズではエクスプローラーのパスをバックスラッシュからスラッシュに置き換える必要あり



演習2 ----

1. 以下の関数を編集して、自分のワーキングディレクトリを設定しましょう

2. また、配布した疑似データ・スクリプトをこのワーキングディレクトリ下に保存しましょう

wdの移動

setwd("c:/Users/00006909/kenshu2023")

ちゃんと設定できたか確認

getwd()

ファイルがちゃんと置けているか確認

dir()

(応用) 「生のR」とtidyverse

 You

tidyverseとは何ですか？



ChatGPT

Tidyverse (タイディバース) は、統計解析やデータ操作、可視化などのためのR言語のパッケージの集まりです。Tidyverseは、データ分析のプロセスを簡潔で効果的に行うために設計されており、一貫性のある文法やデータ構造を提供します。Hadley Wickhamとその共同開発者たちによって開発されました。

Tidyverseの主なパッケージには以下が含まれます：

1. **ggplot2**: 高品質なグラフィックスを生成するためのパッケージ。
2. **dplyr**: データのフィルタリング、整形、結合などのデータ操作を行うためのパッケージ。
3. **tidyr**: データの整形や欠損値の処理を行うためのパッケージ。
4. **readr**: テキストデータやCSVファイルなどを効率的に読み込むためのパッケージ。
5. **purrr**: データの操作や関数の適用をベクトル化するためのパッケージ。
6. **tibble**: データフレームの改良版で、表示や処理が改善されています。

これらのパッケージは、統一されたデータモデルと文法を提供し、データサイエンスや統計解析のプロセスをより効果的に行うのに役立ちます。TidyverseはR言語の生態系を豊かにし、データ分析や可視化の作業をより直感的で効率的に進めることができますようになります。

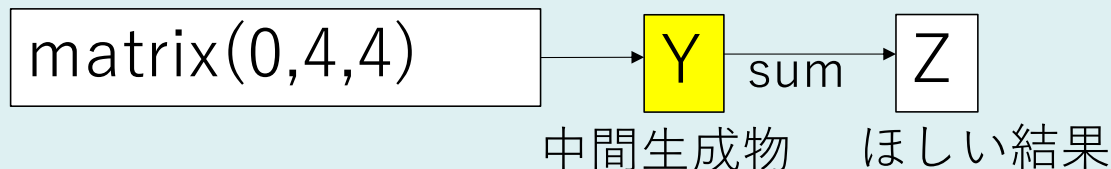
表：tidyverseが提供するパッケージと代表的な機能

パッケージ名	説明	代表的な関数(括弧内は、tidyverseを使わない場合のRの関数)
tibble	より柔軟な形式のデータフレーム <code>tibble</code> 等を提供	データフレームを作成: <code>tibble (data.frame)</code> , データをtibble形式に変換: <code>as_tibble</code> , リストを作成: <code>lst (list)</code>
tidyr	データを横長データから縦長データへ、縦長データから横長データへ変換する	横長データを縦長データに変換する: <code>gather (as.data.frame.table)</code> , 縦長データを横長データに変換する: <code>spread (tapply)</code>
dplyr	データを整形したり、変換したりする	列を抽出: <code>select</code> , 行を抽出: <code>filter (subset)</code> , 列を追加または変更: <code>mutate</code> , データの並び替え: <code>arrange (sort , order)</code> , パイプ演算子: <code>%>%</code> (詳細はBOX)
readr	データの読み込みと出力。 読んだデータはtibble形式となる。	<code>read_table (read.table)</code> , <code>read_csv (read.csv)</code> , <code>write_table (write.table)</code> , <code>write_csv (write.csv)</code>

stringr	文字列操作	<code>str_</code> で始まる関数群。文字列結合: <code>str_c (paste)</code>
forcats	カテゴリカル変数を取扱う	<code>fct_</code> で始まる関数群。他の変数を使ってカテゴリカル要素の順番 (level)を付け替える: <code>fct_reorder</code> , 登場頻度が多い (少ない) 順に levelを付け替える: <code>fct_infreq</code>
purrr	ループ処理やapply系関数	<code>map_</code> で始まる関数群。各要素に関数を適用: <code>map (lapply)</code> , mapしたのちにdata.frameとして結合: <code>map_dfr (sapply)</code>
ggplot2	グラフ作成 (詳細は後日)	<code>ggplot</code> , <code>geom_</code> で始まる関数群

※ 資源管理研修、R統計研修、frasyrの関数の中身は生のRとtidyverseが混在しています！
<https://github.com/ichimomo/main/tree/master/tidyverse#readme>

演算子 %>%



生のRを使う場合

```
Y <- matrix(0, 4, 4)
```

```
Z <- sum(Y)
```

生のRで中間生成物Yをつくらないやり方

```
Z <- sum(matrix(0, 4, 4))
```

%>%を使ったやり方

```
Z <- matrix(0, 4, 4) %>% sum()
```

%>%の前のオブジェクトが%>%
の関数の第一引数になる

余計な中間生成物 (Y)を作らずに、
かつ、matrixを作った後にsumをする、
という手順がわかりやすい

生のRとtidyverseの違い

生のR

```
datA <- data.frame(x=1:10, y=rnorm(10)) # datAを作る
```

```
datA$z <- datA$x * 2 # zという列を足す
```

```
datA2 <- datA[datA$x>5, ] # xが5よりも大きい行をとりだす
```

```
plot(datA$x, datA$y)
```

\$ (列の指定) や [,] (列、行の指定) を使った操作が多い

tidyverse

```
library(tidyverse)
```

```
datB <- tibble(x=1:10, y=rnorm(10)) %>%
```

```
  mutate(z=x*2) # datBを作ってzという列を足す
```

```
datB2 <- mutate(datB, x>5) # datBでxが5よりも大きい行をとりだす
```

```
datB %>% ggplot() +  
  geom_point(aes(x=x, y=y))
```

%>% という演算子が登場

最初にtidyverseが呼び出される

プロットするときはggplot () + から始まる

💡 ポイント 💡

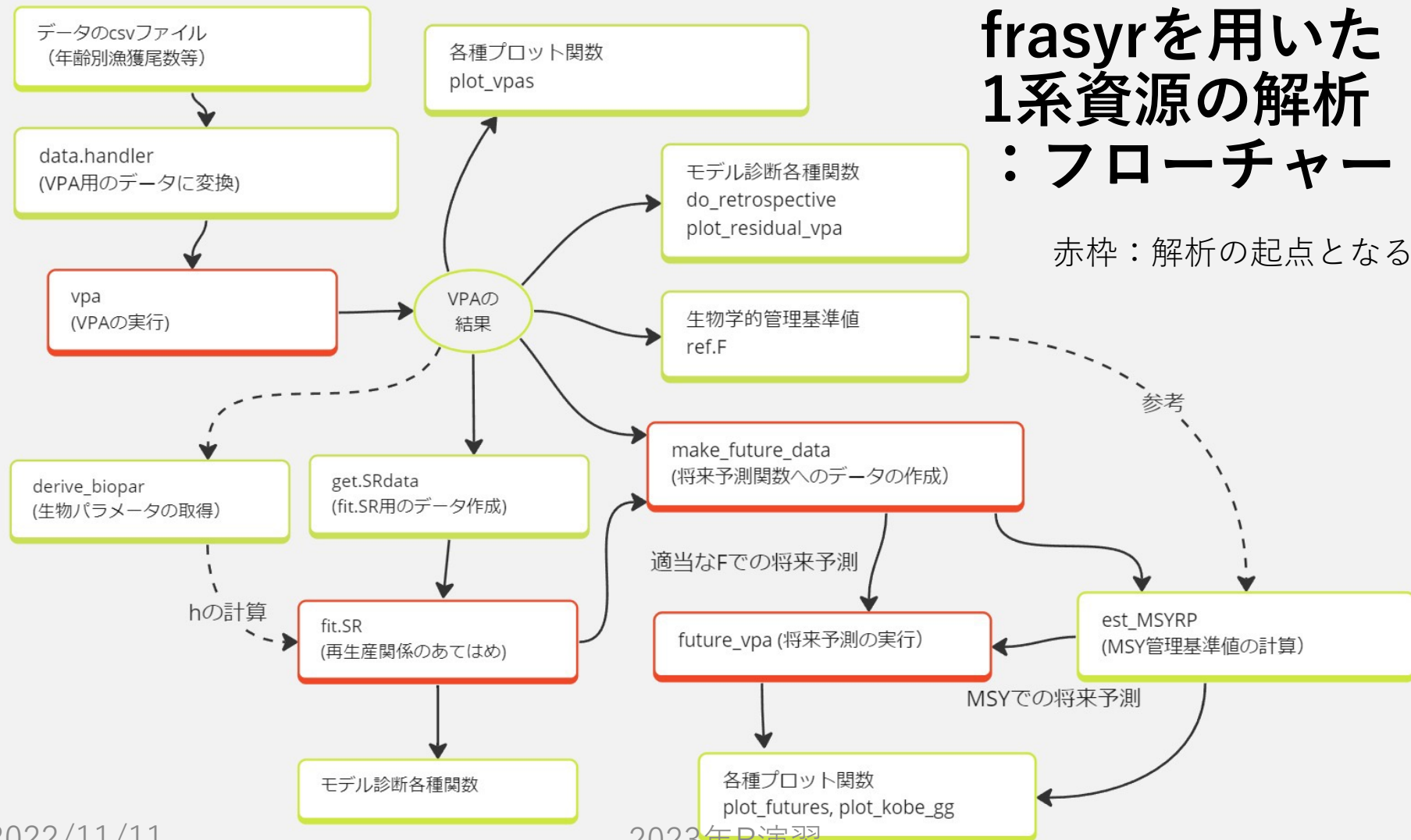
- 用いたい関数は何をするのか・どんな結果を返すのか・どんなオプションが可能なのかをチェックしよう
- 常にWDを意識して作業しよう
- 読みたいRコードが、生のRなのか、tidyverse構文なのか意識しよう

D. Rを使った演習

1. データの読み込み
2. VPAのあてはめ
3. 再生産関係
4. 管理基準値と将来予測

frasyrを用いた 1系資源の解析 ：フローチャート

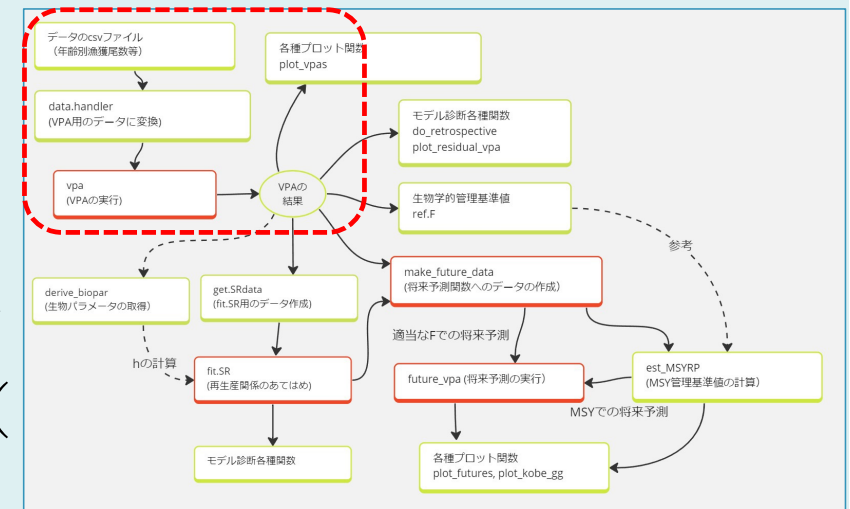
赤枠：解析の起点となる関数



1. データの読み込み

- 用意するもの：年齢別漁獲尾数・体重・成熟率・（あれば）資源量指数のデータ（
- データのフォーマット例

<https://github.com/ichimomo/frasyr/tree/dev/data-raw>



生データのダウンロード

frasyr / data-raw / ex1_caa.csv

ichimomo データファイルの名前変更と整理 723e696 · 4 years ago History

Preview Code Blame 8 lines (8 loc) · 1.25 KB

Search this file

3	1	129.461780481427	121.415250168908	162.962634012983	158.666333202136	154.13567216006	95.7533708182427	144.155190886353	91.3945563063534	62.3493810112294
4	2	72.3719525924633	71.295752908148	82.0832241749605	85.4627954042309	98.0756145304531	76.9842305941744	47.8247473536579	61.0860022903068	39.0436403342683

演習③ データを読み込んでVPAを一回実行しよう

- 配布データされたcsvファイルを読み込んでVPAを一回実行してみてください

```
setwd("c:/Users/00006909/kenshu2023")
```

自分の環境にあわせて変更してください

```
library(frasyr)
```

```
library(tidyverse)
```

```
library(patchwork)
```

必要なライブラリの読み込み

データの読み込み ----

年齢別漁獲尾数

```
caa <- read.csv("ex1_caa.csv", row.names=1)
```

配布したサンプルデータの読み込み

年齢別体重

```
waa <- read.csv("ex1_waa.csv", row.names=1)
```

年齢別成熟率

```
maa <- read.csv("ex1_maa.csv", row.names=1)
```

資源量指数

```
index <- read.csv("ex1_index.csv", row.names=1)
```

f rasyrの関数の利用

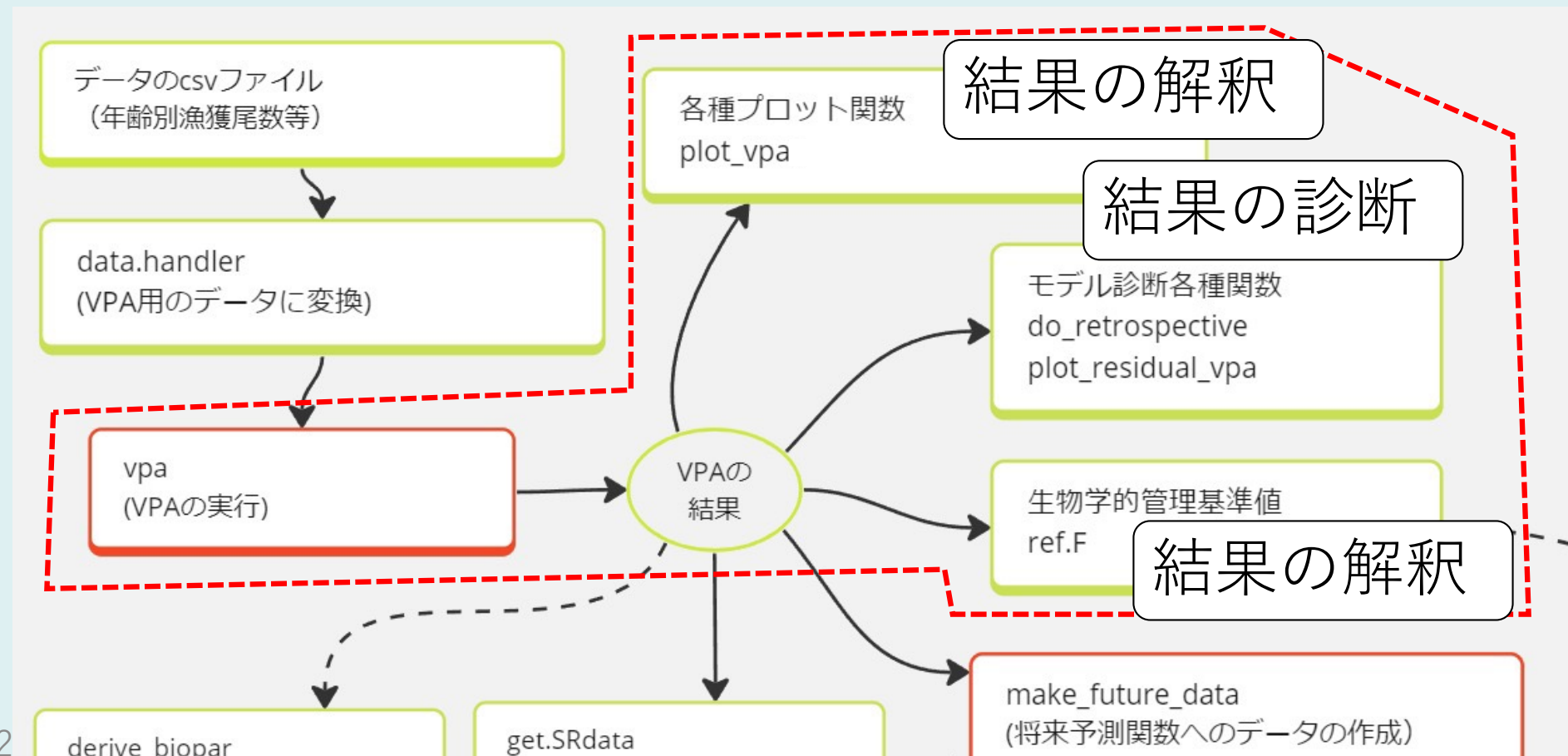
```
dat_vpa <- data.handler(caa=caa, waa=waa, maa=maa, M=0.5, index=index)
```

```
res_vpa <- vpa(dat_vpa) # 引数をつけないで実行 = デフォルト引数が使われる
```

💡 ポイント 💡

- 新しく自分の解析を始める場合でも、既存のデータ（動くもの）をダウンロード➡変えるべき部分を変更するのが良い
- Mが年・年齢によって一定の場合には、ファイルを用意せず data.handlerの引数 (M=XXX) で指定しても良い
- 省略可の引数を指定しない場合には、デフォルト引数が使われる➡どんな引数がデフォルトとして設定されているのか、確認するのが大事

VPAの実行&プロット&モデル診断



vpa関数の全貌

```
> args(vpa)
function (dat, sel.f = NULL, tf.year = 2008:2010, rec.new = NULL,
  rec = NULL, rec.year = 2010, rps.year = 2001:2010, fc.year = 2009:2011,
  last.year = NULL, last.catch.zero = FALSE, faa0 = NULL, naa0 = NULL,
  f.new = NULL, Pope = TRUE, p.pope = 0.5, tune = FALSE, abund = "B",
  min.age = 0, max.age = 0, link = "id", base = NA, af = NA,
  p.m = 0.5, omega = NULL, index.w = NULL, use.index = "all",
  scale = 1000, hessian = TRUE, alpha = 1, maxit = 5, d = 1e-04,
  min.caa = 0.001, plot = FALSE, plot.year = NULL, term.F = "max",
  plus.group = TRUE, stat.tf = "mean", add.p.est = NULL,
  add.p.ini = NULL, sel.update = FALSE, sel.def = "max",
  max.dd = 1e-06, ti.scale = NULL, tf.mat = NULL, eq.tf.mean = FALSE,
  no.est = FALSE, est.method = "ls", b.est = FALSE, est.constraint = FALSE,
  q.const = 1:length(abund), b.const = 1:length(abund), q.fix = NULL,
  b.fix = NULL, sigma.const = 1:length(abund), fixed.index.var = NULL,
  max.iter = 100, optimizer = "nlm", Lower = -Inf, Upper = Inf,
  p.fix = NULL, lambda = 0, beta = 2, penalty = "p",
  ssb.def = "i", ssb.lag = 0, TMB = FALSE, sel.rank = NULL,
  p.init = 0.2, sigma.constraint = 1:length(abund), eta = NULL,
  eta.age = 0, tmb.file = "rvpa_tmb", remove.abund = NULL,
  madara = FALSE, p_by_age = FALSE, penalty_age = NULL, no_eta_age = NULL,
  sdreport = FALSE, use.equ = "new", ave_S = TRUE)
```

help(vpa)

R: VPAによる資源計算を実施する ▾Find in Topic

vpa {frasyr}

R Documentation

VPAによる資源計算を実施する

Description

VPAによる資源計算を実施する

Usage

```
vpa(  
  dat,  
  sel.f = NULL,  
  tf.year = 2008:2010,  
  rec.new = NULL,  
  rec = NULL,  
  rec.year = 2010,  
  rps.year = 2001:2010,  
  fc.year = 2009:2011,  
  last.year = NULL,  
  last.catch.zero = FALSE,  
  faa0 = NULL,  
  naa0 = NULL,  
  f.new = NULL,  
  Pope = TRUE,  
  p.pope = 0.5,  
  tune = FALSE,  
  abund = "B",
```

演習④ いろいろな設定のVPAを試そう

- # - 配布された"dat_vpa1.rds"ファイルをreadRDSコマンドで読み込もう
- # - どんなデータか確認しよう
- # - いろいろなVPAの設定を試して、結果を比較しよう

#

#+

```
dat_vpa <- readRDS("dat_vpa1.rds")  
#dat_vpa <- readRDS("dat_vpa2.rds")  
#dat_vpa <- readRDS("dat_vpa3.rds")  
#dat_vpa <- readRDS("dat_vpa4.rds")
```

データの確認

```
par(mfrow=c(2,2))  
matplot(t(dat_vpa$caa), type="b")  
matplot(t(dat_vpa$index[1,]), type="b")  
matplot(t(dat_vpa$index[2,]), type="b")
```

まず、どんなデータをこれから解析するのか、
確認してみましょう

```

# tuningなし
res_vpa_no <- vpa(dat_vpa, tf.year=45:49, tune=FALSE,
                 p.init=0.4,Pope=FALSE,fc.year=48:50)
plot_vpa(res_vpa_no, what.plot=c("SSB","Recruitment","fishing_mortality"))

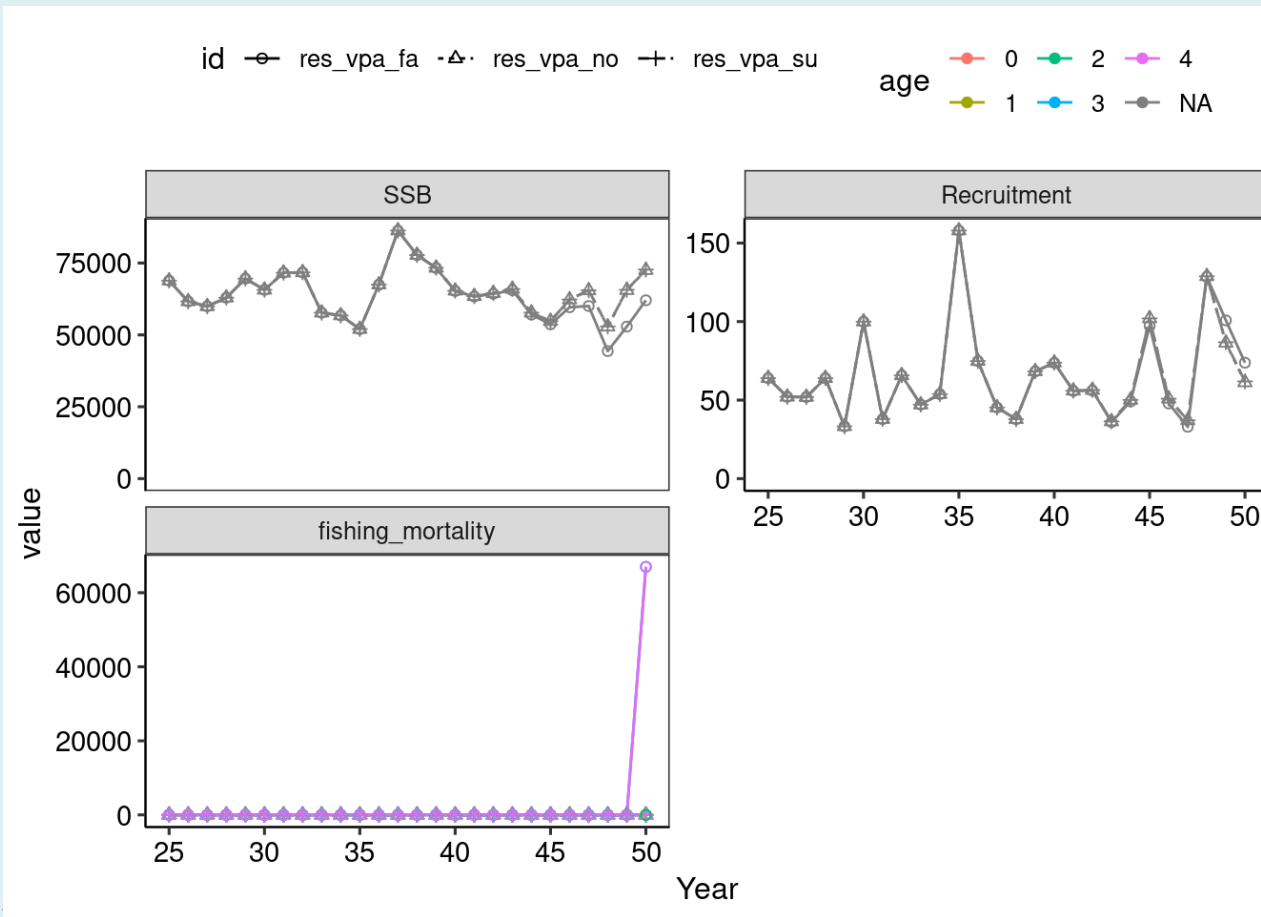
# 選択率更新法
res_vpa_su <- vpa(dat_vpa, tf.year=45:49, sel.f=NULL,
                 tune=TRUE, sel.update=TRUE,
                 abund=c("N","SSB"),min.age=c(0,0), max.age=c(0,5),
                 p.init=0.4,Pope=FALSE,fc.year=40:45)
plot_vpa(res_vpa_su, what.plot=c("SSB","Recruitment","fishing_mortality"))

# 全F推定
res_vpa_fa <- vpa(dat_vpa, tf.year=NULL, sel.f=NULL,
                 tune=TRUE, sel.update=FALSE, term.F="all",
                 abund=c("N","SSB"),min.age=c(0,0), max.age=c(0,5),
                 p.init=0.4,Pope=FALSE,fc.year=40:45)
plot_vpa(res_vpa_fa, what.plot=c("SSB","Recruitment","fishing_mortality"))

```

3つの結果の比較

```
plot_vpa(lst(res_vpa_no, res_vpa_su, res_vpa_fa),  
  what.plot=c("SSB", "Recruitment", "fishing_mortality"))
```



```
# ### 演習⑤ モデル診断しよう  
# - 残差プロット・レトロスペクティブ解析・ブートストラップ解析を行って、  
#   3つのモデルを比較しよう
```

```
# 残差のプロット
```

```
gg <- plot_residual_vpa(res_vpa_su)  
wrap_plots(gg[1:3], ncol=1)
```

```
gg <- plot_residual_vpa(res_vpa_fa)  
wrap_plots(gg[1:3], ncol=1)
```

レトロスペクティブ解析

```
res_retro_no <- do_retrospective_vpa(res_vpa_no)
```

```
res_retro_su <- do_retrospective_vpa(res_vpa_su)
```

```
res_retro_fa <- do_retrospective_vpa(res_vpa_fa)
```

Mohn's rho: 正なら過大傾向気味、負なら過小推定気味

```
bind_rows(res_retro_no$mohn_rho,  
          res_retro_su$mohn_rho,  
          res_retro_fa$mohn_rho)
```

ブートストラップ

```
res_boot_su <- plot_resboot_vpa(res_vpa_su, B_ite=30)
```

```
res_boot_fa <- plot_resboot_vpa(res_vpa_fa, B_ite=30)
```

```
wrap_plots(res_boot_su[c("plot_ssb", "plot_rec", "plot_biomass")], ncol=2)
```

```
wrap_plots(res_boot_fa[c("plot_ssb", "plot_rec", "plot_biomass")], ncol=2)
```



```
# 演習⑥ 資源の状態を把握しよう
# - SPR, YPRなどの管理基準値を計算し、現状の漁獲圧がどのくらいなのか
#   (高いのか・低いのか) 判断してみよう
```

```
# 生物学的管理基準値の計算
```

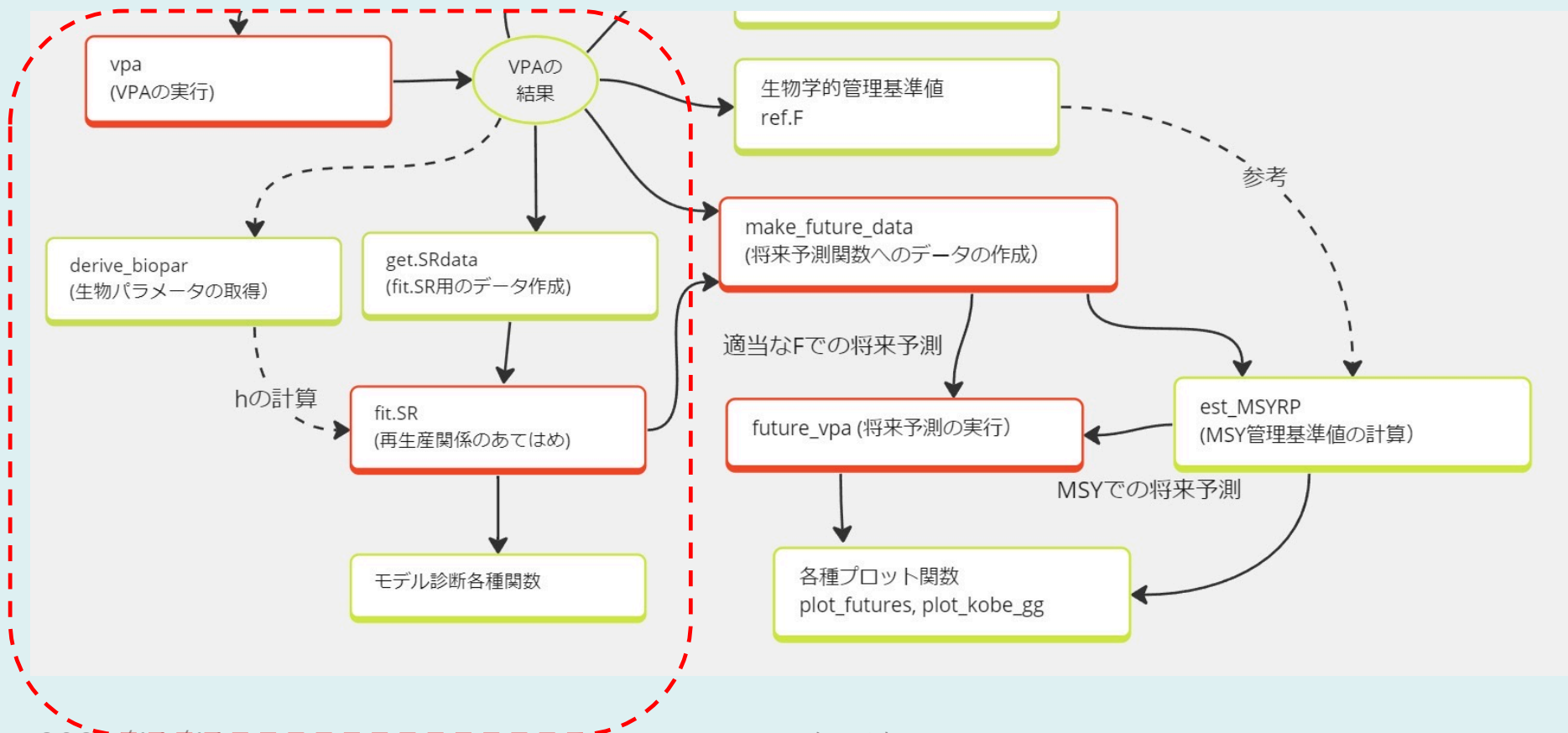
```
F4850 <- rowMeans(res_vpa_su$faa[, as.character(48:50)])
res_ref4850 <- ref.F(res_vpa_su, Fcurrent=F4850)
F4045 <- rowMeans(res_vpa_su$faa[, as.character(40:45)])
res_ref4045 <- ref.F(res_vpa_su, Fcurrent=F4045)
```

```
res_ref4850$summary
res_ref4045$summary
```

```
aa <- get.SPR(res_vpa_su)
plot(aa$ysdata$perSPR)
```

演習⑦ どのチューニング手法がおすすめか、判断しよう [難]
- `dat1.rds`から推定される3つのVPA結果のうち、どの結果が一番「良い」だろうか？その根拠は？[難]
- 別データ（`dat2.rds`）でも同様の解析を行って、`dat2.rds`の場合にはどの手法が一番良いか考えよう[難]
#

3. 再生産関係のあてはめ



演習⑧ 再生産関係のあてはめと選択

- dat1.rdsの選択率更新法の結果を用いて再生産関係の推定を行ってみよう

- 再生産ガイドラインに沿うと、HS, RI, BHのうちどの再生産関係を選ぶのが適切だろうか？[難]

- 再生産関係の選択によって、過去の資源のトレンドや資源状態がどのように説明できるようになるだろうか？[難]

- dat3.rdaデータについても同様の解析を行い、dat1.rdsの結果と比較してみよう
[難]

hの計算に必要な生物パラメータの取り出し

```
biopars <- derive_biopar(res_vpa_su, derive_year=45:50)
```

```
data_SR_su <- get_SRdata(res_vpa_su)
```

```
res_allSR_su <- tryall_SR(data_SR_su, bio_par=biopars)
```

```
res_allSR_su %>% arrange(AICc) %>%
```

```
  mutate(deltaAICc=AICc-min(AICc)) %>% View()
```

```
res_SR_BH <- fit_SR(data_SR_su, SR="BH", method="L2", AR=0,  
                    bio_par=biopars, max.ssb.pred=10)
```

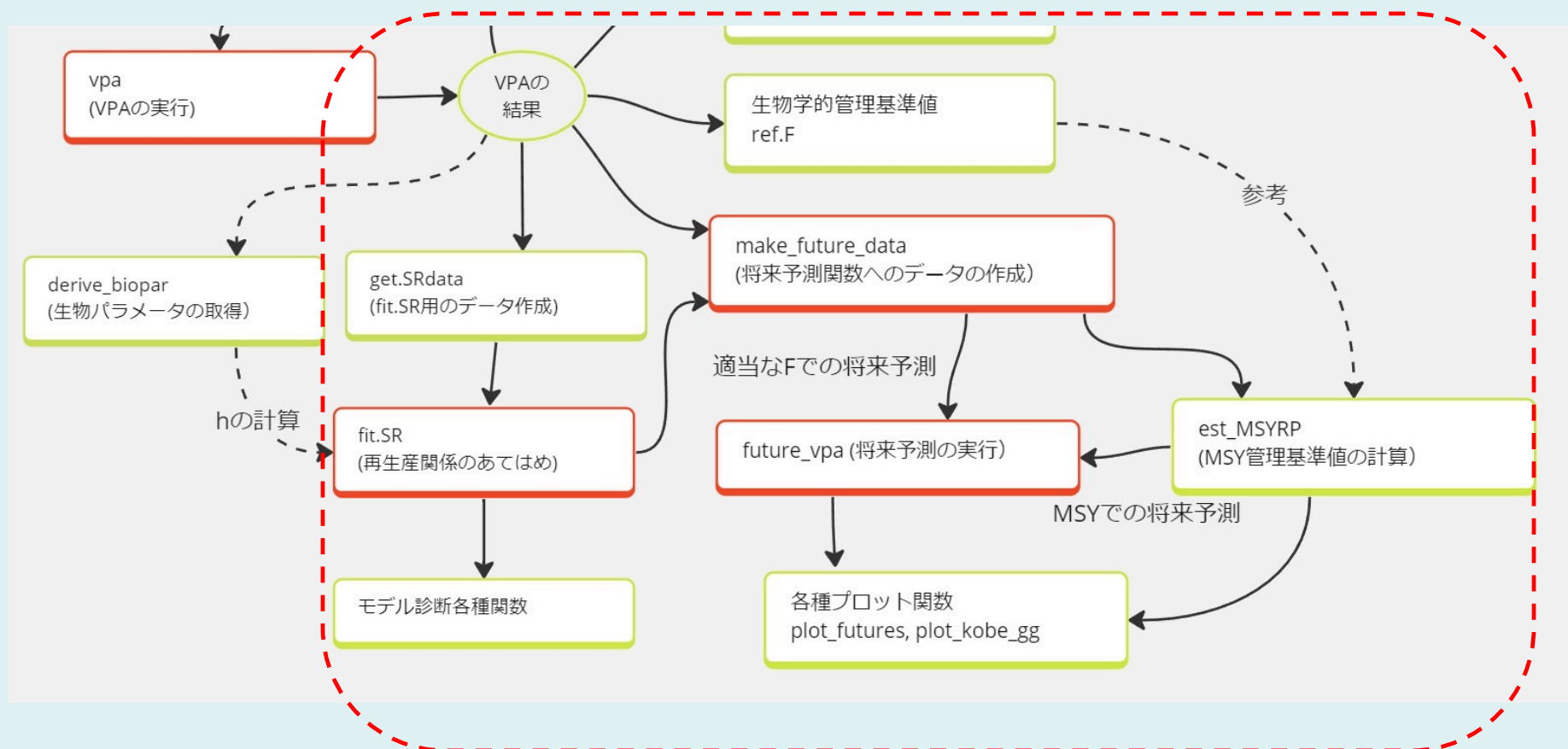
```
res_SR_RI <- fit_SR(data_SR_su, SR="RI", method="L2", AR=0,  
                    bio_par=biopars, max.ssb.pred=10)
```

```
res_SR_HS <- fit_SR(data_SR_su, SR="HS", method="L2", AR=0,  
                    bio_par=biopars, max.ssb.pred=10)
```

B0まで考慮した場合の再生産関係：どれがそれらしいか？頑健か？

```
(gg <- compare_SRfit(SRlist=lst(res_SR_BH, res_SR_RI, res_SR_HS)) +  
  geom_vline(xintercept=res_SR_BH$steepness$SB0/1000, col="red") +  
  geom_vline(xintercept=res_SR_RI$steepness$SB0/1000, col="blue") +  
  geom_vline(xintercept=res_SR_HS$steepness$SB0/1000, col="green") +  
  xlim(0, res_SR_BH$steepness$SB0/1000*1.2))
```

4. 管理基準値計算 & 将来予測



```
# 演習⑨ 将来予測をやってみよう
# - 以下のコマンドを実行し、将来予測～結果のプロット～神戸プロットの描画までを体験してみよう
# - dat3.rds、dat4.rdsについても将来予測をやってみよう。
#                               それぞれどのような特徴があるだろうか？[難]
currentF <- as.numeric(unlist(res_vpa_su$faa))

data_future <- make_future_data(res_vpa_su, nsim=100, nyear=30,
                                future_initial_year_name=50, start_F_year_name=51,
                                start_biopar_year_name=51, start_random_rec_year_name=51,
                                currentF=F4850,
                                futureF=F4850,
                                waa_year=49:50, waa_catch_year=49:50, maa_year=49:50,
                                M_year=49:50,
                                faa_year=NULL, start_ABC_year_name=52, res_SR=res_SR_HS)
res_MSY <- est_MSYP(data_future)
res_MSY$summary
```



```

SPRmsy <- res_MSY$summary$perSPR[1]
Fmsy <- res_MSY$Fvector[1,] %>% as.numeric()
SBlimit <- res_MSY$summary$SSB[4]
SBban <- res_MSY$summary$SSB[3]
SBtarget <- res_MSY$summary$SSB[1]

# make_data_futureのHCR関係の引数をMSYベースのものに変更する
data_future_msy <- redo_future(data_future,
                                input_data_list=list(futureF=Fmsy, HCR_Bban=SBban,
                                                       HCR_beta=0.8, HCR_Blimit=SBlimit),
                                only_data=TRUE)
res_future_msy <- future_vpa(data_future_msy$data, multi_init=1,
                              calc_SPR_year_name=as.character(1:100),
                              SPRtarget=SPRmsy*100)

plot_futures(res_vpa_su, list(res_future_msy), Btarget=res_MSY$summary$SSB[1],
              what.plot=c("Recruitment", "SSB", "biomass", "catch", "U", "Fratio"))

```

神戸プロット

```
kobe_ratio <- get.SPR(res_vpa_su, target.SPR=res_MSY$summary$perSPR[1]*100)$ysdata %>%  
  mutate(Bratio=unlist(colSums(res_vpa_su$ssb))/res_MSY$summary$SSB[1])%>%  
  mutate(Fratio=get("F/Ftarget")) %>%  
  mutate(year=as.numeric(colnames(res_vpa_su$faa)))  
  
plot_kobe_gg(FBdata=kobe_ratio,  
             refs_base=res_MSY$summary)
```

```

# ### 演習10 将来予測でABCの不確実性を考慮してみよう[難]
# - 通常の将来予測res_future_msyの結果とどこが違うだろうか？
# - dat1.rdsとdat4.rdsでどのように違うだろうか？その理由は？

# 引数を変えてdata_futureを作り直す（時間がかかるため、計算回数を減らす）
data_future_mse <- redo_future(data_future_msy, input_data_list=list(nsim=50,
nyear=10),
                                only_data=TRUE)
# do_MSEのオプションをTRUEにしてMSEモードにする
res_future_mse <- future_vpa(data_future_mse$data,
                              multi_init=res_MSY$summary$Fref2Fcurrent[1],
                              do_MSE=TRUE, MSE_input_data=data_future_mse,
                              calc_SPR_year_name=as.character(1:100),
                              SPRtarget=SPRmsy*100)
plot_futures(res_vpa_su, lst(res_future_mse, res_future_msy),
             what.plot=c("Recruitment", "SSB", "biomass", "catch", "U", "Fratio"))

```