Appendix B: Code for analysis comparing to historical data

2019-06-05

Section 1: Model estimates from data

Computes model parameter estimates for selected stocks in RAM using NIMBLE.

```
# devtools::install_github("boettiger-lab/sarsop")  ## install package first if necessary.
library(tidyverse)
library(sarsop)
library(nimble)
library(parallel)
library(gridExtra)
library(tictoc)
library(furrr)
tic()
```

```
path <- "../data/appendixB/ramlegacy_v3.0.xlsx"
sheets <- readxl::excel_sheets(path)
ram <- lapply(sheets, readxl::read_excel, path = path)
names(ram) <- sheets

ramlegacy <-
  ram$timeseries_values_views %>%
  select(assessid, stockid, stocklong, year, SSB, TC) %>%
  left_join(ram$stock) %>%
  left_join(ram$area) %>%
  select(assessid, stockid, scientificname,
         commonname, areaname, country, year,
         SSB, TC) %>%
  left_join(ram$timeseries_units_views %>%
              rename(TC_units = TC, SSB_units = SSB)) %>%
  select(scientificname, commonname,
         stockid, areaname, country, year,
         SSB, TC, SSB_units, TC_units)
```

Let's filter out missing data, non-matching units, and obvious reporting errors (catch exceeding total spawning biomass), then we re-scale each series into the 0,1 by appropriate choice of units:

```
df2 <- ramlegacy %>%
  filter(!is.na(SSB), !is.na(TC)) %>%
  filter(SSB_units == "MT", TC_units=="MT") %>%
  filter(SSB > TC) %>%
  select(-SSB_units, -TC_units) %>%
  group_by(stockid) %>%
  mutate(scaled_catch = TC / max(SSB),
         scaled_biomass = SSB / max(SSB))
```

```
stock_ids <- c("PLAICNS", "ARGHAKENARG")
examples <- df2 %>%
  filter(stockid %in% stock_ids) %>%
  ungroup() %>%
  group_by(commonname)
```

```
## Model does not estimate sigma_m; data is insufficient to do so.
gs_code  <- nimble::nimbleCode({
  r ~ dunif(0, 2)
  K ~ dunif(0, 2)
  sigma ~ dunif(0, 1)

  x[1] <- x0
  for(t in 1:(N-1)){
    mu[t] <- x[t] + x[t] * r * (1 - x[t] / K) - min(a[t],x[t])
    x[t+1] ~ dnorm(mu[t], sd = sigma)
  }
})

fit_models <- function(fish, code){
  # fish <- examples %>% filter(stockid == stock_ids[1])

  ## Rescale data
  N <- dim(fish)[1]
  scaled_data <- data.frame(t = 1:N,
                            y = fish$scaled_biomass,
                            a = fish$scaled_catch)
  data = data.frame(x = scaled_data$y)

  ## Compile  model
  constants <- list(N = N, a = scaled_data$a)
  inits <- list(r = 0.5, K = 0.5, sigma = 0.02, x0 = scaled_data$y[1])
  model <- nimbleModel(code, constants, data, inits)
  C_model <- compileNimble(model)

  mcmcspec <- configureMCMC(model, thin = 1e2)
  mcmc <- buildMCMC(mcmcspec)
  Cmcmc <- compileNimble(mcmc, project = model)
  Cmcmc$run(1e6)


  samples <- as.data.frame(as.matrix(Cmcmc$mvSamples))
  burnin <- 1:(0.05 * dim(samples)[1]) # drop first 5%
  samples <- samples[-burnin,1:(length(inits) - 1)] # drop raised vars, burnin
  #gather(samples) %>% ggplot() + geom_density(aes(value)) + facet_wrap(~key, scale='free')

  ## Return fit
  data.frame(stockid = fish$stockid[1],
             commonname = fish$commonname[1],
             r = mean(samples$r),
             K = mean(samples$K),
             sigma_g = mean(samples$sigma),
             r_sd = sd(samples$r),
             K_sd = sd(samples$K),
             sigma_g_sd = sd(samples$sigma),
             stringsAsFactors = FALSE)
}
```

```
set.seed(123)
fits <- examples %>% do(fit_models(., code=gs_code))
```

```
|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
|-------------|-------------|-------------|-------------|
|-------------------------------------------------------|
```

```
fits
```

```
# A tibble: 2 x 8
# Groups:   commonname [2]
  stockid     commonname          r     K sigma_g   r_sd  K_sd sigma_g_sd
  <chr>       <chr>           <dbl> <dbl>   <dbl>  <dbl> <dbl>      <dbl>
1 ARGHAKENARG Argentine hake  1.04   1.20   0.112 0.179  0.192     0.0267
2 PLAICNS     European Plaice 0.906  1.78   0.127 0.0743 0.156     0.0130
```

```
pars <- fits %>% ungroup() %>% select(commonname, r, K, sigma_g)
pars
```

commonname	r	K	sigma_g
Argentine hake	1.0379274	1.197693	0.1121662
European Plaice	0.9064288	1.777442	0.1273774

Calculations of the Decision Policies for Historical Data

```
options(mc.cores = 6) # Reserve ~ 10 GB per core
log_dir <- "../data/appendixB"
```

```
## Classic Gordon-Schaefer. Note that recruitment occurs *before* harvest
gs <- function(r,K){
  function(x, h){
    x + x * r * (1 - x / K) - pmin(x,h)
  }
}
reward_fn <- function(x,h) pmin(x,h)
discount <- .95
```

Discretize space

Note that the large values of K require we carry the numerical grid out further.

```
states <- seq(0,4, length=150)
actions <- states
observations <- states
```

Consider all parameter values combinations for which we want solutions (both species at each of three possible levels of measurement uncertainty; though we will focus on the `0.1` level for simplicity as overall pattern is the same at `0.15`):

```r
meta <- expand.grid(commonname = pars$commonname,
                    sigma_m = c(0, 0.1, 0.15),
                    stringsAsFactors = FALSE) %>%
  left_join(pars) %>%
  mutate(scenario  = as.character(1:length(sigma_m)))

meta
```

commonname	sigma_m	r	K	sigma_g	scenario
Argentine hake	0.00	1.0379274	1.197693	0.1121662	1
European Plaice	0.00	0.9064288	1.777442	0.1273774	2
Argentine hake	0.10	1.0379274	1.197693	0.1121662	3
European Plaice	0.10	0.9064288	1.777442	0.1273774	4
Argentine hake	0.15	1.0379274	1.197693	0.1121662	5
European Plaice	0.15	0.9064288	1.777442	0.1273774	6

Create the model matrices (transition, observation, and reward matrix):

```r
plan(multiprocess)

models <-
  furrr::future_map(1:dim(meta)[1],
                    function(i){
                      fisheries_matrices(
                        states = states,
                        actions = actions,
                        observed_states = observations,
                        reward_fn = reward_fn,
                        f = gs(meta[i, "r"][[1]], meta[i, "K"][[1]]),
                        sigma_g = meta[i,"sigma_g"][[1]],
                        sigma_m = meta[i,"sigma_m"][[1]],
                        noise = "normal")
                    })
```

Here's the slowest part: computing POMDP alpha vectors.

```r
dir.create(log_dir, FALSE)
plan(multiprocess)
## POMDP solution
system.time(
  alphas <-
    furrr::future_map(1:length(models),
    function(i){
      log_data <- data.frame(model = "gs",
                             r = meta[i, "r"][[1]],
                             K = meta[i, "K"][[1]],
                             sigma_g = meta[i,"sigma_g"][[1]],
                             sigma_m = meta[i,"sigma_m"][[1]],
                             noise = "normal",
                             commonname = meta[i, "commonname"][[1]],
                             scenario = meta[i, "scenario"][[1]])

      sarsop(models[[i]]$transition,
             models[[i]]$observation,
             models[[i]]$reward,
             discount = discount,
             precision = 2e-6,
             timeout = 25000,
             log_dir = log_dir,
             log_data = log_data)
    })
)
```

```
   user  system elapsed 
  3.456   0.585   9.316 
```

Comparison to the static models

```
pars <- examples %>%
  group_by(commonname) %>%
  summarise(N = max(SSB)) %>%
  right_join(
    meta %>%
      select(commonname, r, K) %>%
      distinct())
```

Add corresponding static policy levels on:

```
statics <- function(P){
  f <- gs(P$r, P$K)
  S_star <- optimize(function(x) -f(x,0) + x / discount, c(0, 2* P$K))$minimum
  B_MSY <- S_star
  MSY <- f(B_MSY,0) - B_MSY

  tibble(S_star, F_MSY = MSY / B_MSY, F_TAC = 0.8 * F_MSY,
         commonname = P$commonname, N = P$N)
}

policy_pars <-
  pars %>%
  transpose() %>%
  map_df(statics)
```

Convert example data into discrete index space.

```
index <- function(x, grid) map_int(x, ~ which.min(abs(.x - grid)))
## repeats each series for each static model
ex <- examples %>%
  mutate(biomass = index(scaled_biomass, states),
         catch = index(scaled_catch, actions))  %>%
  left_join(policy_pars) %>%
  left_join(pars) %>%
  ungroup()
```

Static policy calculations:

```
CE_f <- function(S_star, r, K, i)
  index(pmax(gs(r[[1]],K[[1]])(states,0) - S_star[[1]],0), actions)[i]
MSY_f <- function(F_MSY, i) index(states * F_MSY[[1]], actions)[i]
```

```
TAC_f <- function(F_TAC, i) index(states * F_TAC[[1]], actions)[i]
rescale <- function(x, N) states[x]*N

historical <- ex %>%
  group_by(commonname) %>%
  mutate(CE =  CE_f(S_star, r, K, biomass),
         MSY =  MSY_f(F_MSY, biomass),
         TAC =  TAC_f(F_TAC, biomass)) %>%
  select(year, biomass, catch, CE, MSY, TAC, commonname, N) %>%
  gather(model, stock, -year, -commonname, -N) %>%
  mutate(stock = states[stock] * N) %>%
  select(-N)
```

Compute POMDP policy for historical data:

```
set.seed(123456)
pomdp_sims <-
  pmap_dfr(list(models, alphas, 1:dim(meta)[[1]]),
           function(.x, .y, .z){

             ## avoid NSE
             who <- (ex$commonname == meta[.z,"commonname"])
             df <- ex[who,]

             hindcast_pomdp(.x$transition, .x$observation, .x$reward, discount,
                            obs = index(df$scaled_biomass, states),
                            action = index(df$scaled_catch,states),
                            alpha = .y)$df %>%
              mutate(method = "pomdp") %>% # include a column labeling method
              mutate(year = ex[who, "year"][[1]])
           },
           .id = "scenario")
```

Join records:

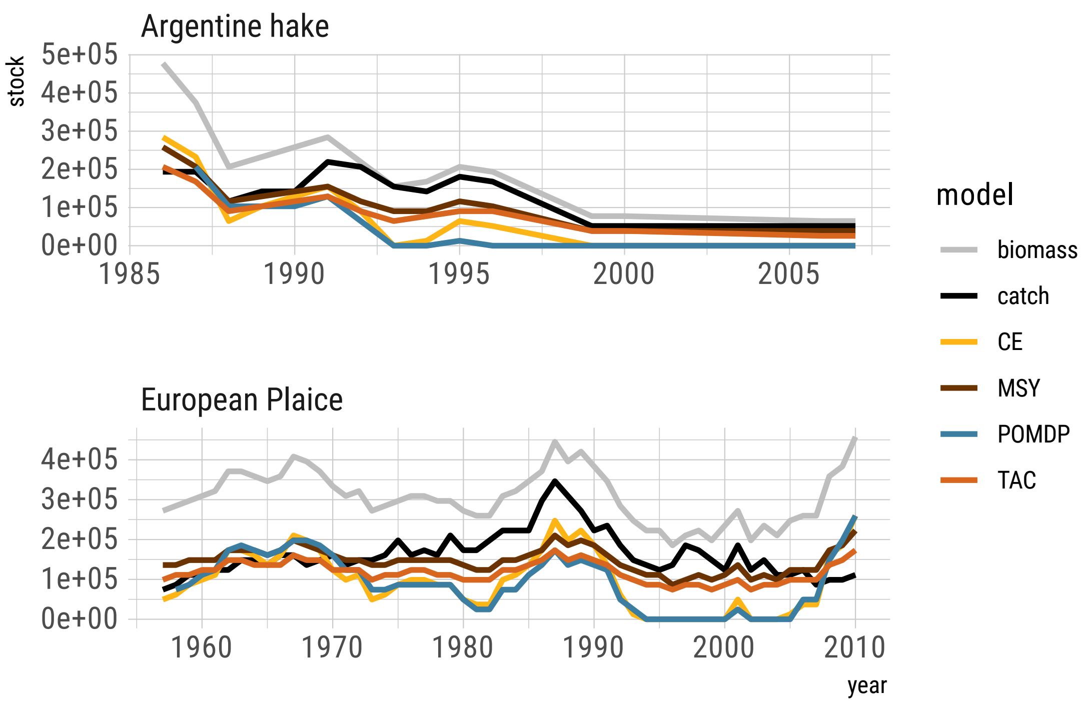
```
pomdp_sims <-
  meta %>%
  select(scenario, commonname,sigma_m) %>%
  left_join(pars) %>%
  right_join(pomdp_sims)
```

```
sims <- pomdp_sims %>%
  mutate(optimal = states[optimal] * N) %>% # original scale
  select(year, optimal, commonname, sigma_m) %>%
  rename(stock = optimal) %>%
  ## treat each sigma_m value as separate 'model'
  mutate(sigma_m = as.factor(sigma_m)) %>%
  mutate(model = recode(sigma_m,
                        "0" = "CE",
                        "0.1" = "POMDP",
                        "0.15" = "POMDP_0.15")) %>%
  select(-sigma_m) %>%
  bind_rows(historical)

write_csv(sims,file.path(log_dir, "appendixB.csv"))
```

Final plot, as in paper but including MSY:

```
appendixB <- read_csv(file.path(log_dir, "appendixB.csv"))
appendixB %>%
  filter(model %in% c("biomass", "catch", "POMDP", "CE", "TAC", "MSY")) %>%
  ggplot(aes(year, stock, col=model)) +
  geom_line(lwd=1) +
  scale_color_manual(values = colors) +
  facet_wrap(~commonname, scales = "free", ncol=1)
```



System Information

Total runtime:

```r
toc()
```

```
118.142 sec elapsed
```

Hardware:

```r
system2("grep", c("MemTotal", "/proc/meminfo"), stdout = TRUE)
```

```
[1] "MemTotal:       32938360 kB"
```

```r
system2('grep', '"model name" /proc/cpuinfo', stdout = TRUE)
```

```
 [1] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [2] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [3] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [4] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [5] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [6] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [7] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [8] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
 [9] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[10] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[11] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[12] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[13] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[14] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[15] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
[16] "model name\t: Intel Xeon E312xx (Sandy Bridge)"
```

Software:

```r
devtools::session_info()
```

```
- Session info ----------------------------------------------------------
 setting  value
 version  R version 3.6.0 (2019-04-26)
 os       Debian GNU/Linux 9 (stretch)
 system   x86_64, linux-gnu
 ui       RStudio
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       Etc/UTC
 date     2019-06-05

- Packages --------------------------------------------------------------
 package     * version date       lib
 assertthat    0.2.1   2019-03-21 [1]
 backports     1.1.4   2019-04-10 [1]
 broom         0.5.2   2019-04-07 [1]
 Cairo       * 1.5-10  2019-03-28 [1]
 callr         3.2.0   2019-03-15 [1]
 cellranger    1.1.0   2016-07-27 [1]
 cli           1.1.0   2019-03-19 [1]
 coda          0.19-2  2018-10-08 [1]
 codetools     0.2-16  2018-12-24 [2]
 colorspace    1.4-1   2019-03-18 [1]
 crayon        1.3.4   2017-09-16 [1]
 desc          1.2.0   2018-05-01 [1]
 devtools      2.0.2   2019-04-08 [1]
 digest        0.6.19  2019-05-20 [1]
 dplyr       * 0.8.1   2019-05-14 [1]
 evaluate      0.13    2019-02-12 [1]
 extrafont   * 0.17    2014-12-08 [1]
 extrafontdb   1.0     2012-06-11 [1]
 fansi         0.4.0   2018-10-05 [1]
 forcats     * 0.4.0   2019-02-17 [1]
 fs            1.3.1   2019-05-06 [1]
 furrr       * 0.1.0   2018-05-16 [1]
 future      * 1.13.0  2019-05-08 [1]
 gdtools       0.1.8   2019-04-02 [1]
 generics      0.0.2   2018-11-29 [1]
 ggplot2     * 3.1.1   2019-04-07 [1]
 ggthemes    * 4.2.0   2019-05-13 [1]
 globals       0.12.4  2018-10-11 [1]
 glue          1.3.1   2019-03-12 [1]
 gridExtra   * 2.3     2017-09-09 [1]
 gtable        0.3.0   2019-03-25 [1]
 haven         2.1.0   2019-02-19 [1]
 highr         0.8     2019-03-20 [1]
 hms           0.4.2   2018-03-10 [1]
 hrbrthemes  * 0.6.0   2019-01-21 [1]
 htmltools     0.3.6   2017-04-28 [1]
 httr          1.4.0   2018-12-11 [1]
 igraph        1.2.4.1 2019-04-22 [1]
 jsonlite      1.6     2018-12-07 [1]
 knitr         1.23    2019-05-18 [1]
 labeling      0.3     2014-08-23 [1]
 lattice       0.20-38 2018-11-04 [2]
 lazyeval      0.2.2   2019-03-15 [1]
 linprog       0.9-2   2012-10-17 [1]
 listenv       0.7.0   2018-01-21 [1]
 lubridate     1.7.4   2018-04-11 [1]
```

```
magrittr      1.5     2014-11-22 [1]
Matrix        1.2-17  2019-03-22 [2]
MDPtoolbox    4.0.3   2017-03-03 [1]
memoise       1.1.0   2017-04-21 [1]
modelr        0.1.4   2019-02-18 [1]
munsell       0.5.0   2018-06-12 [1]
nimble      * 0.7.1   2019-03-12 [1]
nlme          3.1-139 2019-04-09 [2]
packrat       0.5.0   2018-11-14 [1]
pillar        1.4.0   2019-05-11 [1]
pkgbuild      1.0.3   2019-03-20 [1]
pkgconfig     2.0.2   2018-08-16 [1]
pkgload       1.0.2   2018-10-29 [1]
plyr          1.8.4   2016-06-08 [1]
prettyunits   1.0.2   2015-07-13 [1]
printr      * 0.1     2017-05-19 [1]
processx      3.3.1   2019-05-08 [1]
ps            1.3.0   2018-12-21 [1]
purrr       * 0.3.2   2019-03-15 [1]
R6            2.4.0   2019-02-14 [1]
Rcpp          1.0.1   2019-03-17 [1]
readr       * 1.3.1   2018-12-21 [1]
readxl        1.3.1   2019-03-13 [1]
remotes       2.0.4   2019-04-10 [1]
reshape2      1.4.3   2017-12-11 [1]
rlang         0.3.4   2019-04-07 [1]
rmarkdown     1.12    2019-03-14 [1]
rprojroot     1.3-2   2018-01-03 [1]
rstudioapi    0.10    2019-03-19 [1]
Rttf2pt1      1.3.7   2018-06-29 [1]
rvest         0.3.4   2019-05-15 [1]
sarsop      * 0.6.0   2019-06-04 [1]
scales        1.0.0   2018-08-09 [1]
sessioninfo   1.1.1   2018-11-05 [1]
stringi       1.4.3   2019-03-12 [1]
stringr     * 1.4.0   2019-02-10 [1]
testthat      2.1.1   2019-04-23 [1]
tibble      * 2.1.1   2019-03-16 [1]
tictoc      * 1.0     2014-06-17 [1]
tidyr       * 0.8.3   2019-03-01 [1]
tidyselect    0.2.5   2018-10-11 [1]
tidyverse   * 1.2.1   2017-11-14 [1]
tinytex       0.13    2019-05-14 [1]
usethis       1.5.0   2019-04-07 [1]
utf8          1.1.4   2018-05-24 [1]
vctrs         0.1.0   2018-11-29 [1]
withr         2.1.2   2018-03-15 [1]
xfun          0.7     2019-05-14 [1]
xml2          1.2.0   2018-01-24 [1]
yaml          2.2.0   2018-07-25 [1]
zeallot       0.1.0   2018-01-28 [1]
source
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
CRAN (R 3.6.0)
```

```
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 Github (boettiger-lab/sarsop@6d6d2f8)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)
 CRAN (R 3.6.0)

[1] /usr/local/lib/R/site-library
[2] /usr/local/lib/R/library
```