Lab 5

Sawyer Balint

Fall 2024; Marine Semester Block 3

Contents

1 Introduction **1**

2 Part I **2**

2.1 Task 1 2

2.2 Task 2 4

2.3 Task 3 6

3 Model Comparison **8**

3.1 Task 4 8

4 Extension **13**

5 Reflection **15**

1 Introduction

This document is available at https://github.com/sjbalint/BI521/tree/main/scripts/labs

```r
#import packages
library(tidyverse) #for data wrangling
library(here) #for filepath management
library(ggsci) #for colors
library(scales) #for log axis breaks
library(bbmle) #for stats

#custom graphing theme

update_geom_defaults("point", list(shape = 21, fill="grey", stroke=0.8))

theme <- list(
  theme_classic(),
  scale_color_viridis_d(option="inferno", end=0.8),
  scale_fill_viridis_d(option="inferno", end=0.8),
  theme(legend.position="right",
        strip.placement="outside",
        strip.background=element_blank()),
  labs(x="Time")
)
```

2 Part I

2.1 Task 1

```
#function for S-Ricker model
gR_SRicker <- function(S,mu){
  with(mu,S^(1+delta)*exp(r*(1-S/K)))
}

SRicker_sim <- function(S0, mu, n_iter){

  R0 <- gR_SRicker(S0, mu)

  df <- data.frame(S=S0, R=R0, t=0)

  #empty list to store results
  result.list <- list(df)

  i_model <- 1

  while(i_model<n_iter){

    i_model <- i_model + 1

    R <- gR_SRicker(S0, mu)
    S <- with(mu, R + Sa*S0)

    df <- data.frame(S=S, R=R, t=i_model)

    S0 <- S
    R0 <- R

    #log results
    result.list <- c(result.list, list(df))
  }

  #compile results
  result.df <- bind_rows(result.list)

  return(result.df)

}

#try it out
mu <- list(r=2.6, K=3/4, delta=1, Sa=0.6)
df <- SRicker_sim(S0=0.04, mu=mu, n_iter=50)
```

```
#looks good
head(df)
```

```
##            S          R t
## 1 0.04000000 0.01875269 0
## 2 0.04275269 0.01875269 2
## 3 0.04687066 0.02121905 3
## 4 0.05326450 0.02514210 4
## 5 0.06371637 0.03175768 5
## 6 0.08205658 0.04382675 6
```

```
#loop over a range of S0
S0.list <- seq(0.01,0.1, by=0.01)

result.list <- list()

for (S0_i in S0.list){

  df <- SRicker_sim(S0=S0_i, mu=mu, n_iter=20) %>%
    mutate(S0 = S0_i)

  result.list <- c(result.list, list(df))

}

result.df <- bind_rows(result.list) %>%
  mutate(S0 = as.factor(S0))

ggplot(result.df, aes(t,S,color=S0, group=S0))+
  theme+
  geom_line()+
  geom_hline(yintercept=0, linetype="dashed")
```

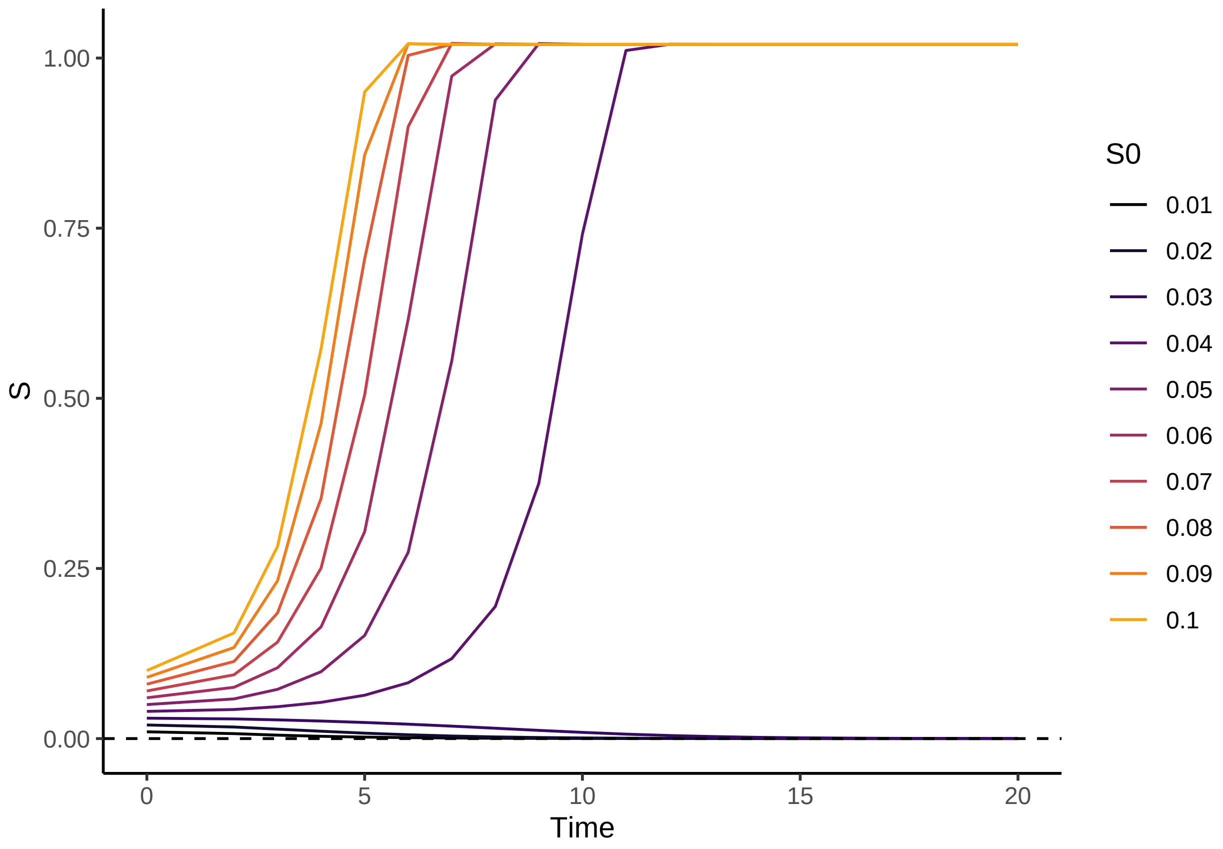


Fig. 1: Sigmodal Ricker with *S_0* values ranging from 0.01 to 0.1 (color). We find that S_0 equal to 0.01, 0.02, and 0.03 result to a population that crashes to zero.

2.2 Task 2

```
#new function that includes fishing mortality
SRicker_Fm_sim <- function(S0, mu, Fm, n_iter){

  R0 <- gR_SRicker(S0, mu)

  df <- data.frame(S=S0, R=R0, t=0)

  #empty list to store results
  result.list <- list(df)

  i_model <- 1

  while(i_model<n_iter){

    i_model <- i_model + 1

    R <- gR_SRicker(S0, mu)
    S <- with(mu, (R + Sa*S0)*(1-Fm))
    df <- data.frame(S=S, R=R, t=i_model)

    S0 <- S
    R0 <- R

    #log results
    result.list <- c(result.list, list(df))
  }

  #compile results
  result.df <- bind_rows(result.list)

  return(result.df)

}

Fm.list <- seq(0,0.8, by=0.1)
S0=0.5

result.list <- list()

for (Fm_i in Fm.list){

  df <- SRicker_Fm_sim(S0=S0, mu=mu, Fm=Fm_i, n_iter=20) %>%
    mutate(Fm = Fm_i)

  result.list <- c(result.list, list(df))

}

result.df <- bind_rows(result.list) %>%
  mutate(Fm = as.factor(Fm))

#make a plot
ggplot(result.df, aes(t,S,color=Fm, group=Fm))+
  theme+
  geom_line()+
  geom_hline(yintercept=0, linetype="dashed")
```

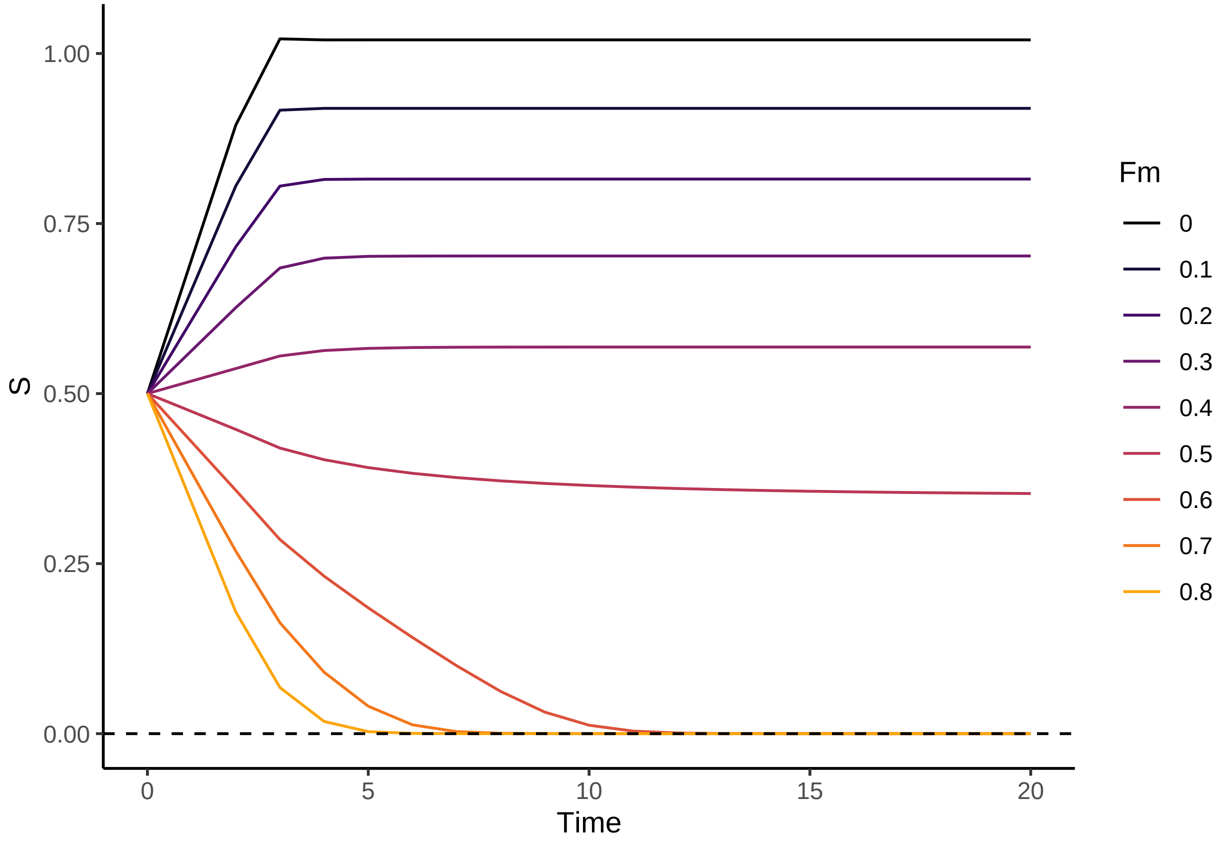


Fig. 2: Higher fishing mortality results in a lower equilibrium population level and a longer time to reach equilibrium. At mortalities of 0.6, 0.7, and 0.8, the population crashes to zero.

This simulation implements a constant effort model of fishing, where the yield is dependent on the size of the population.

2.3 Task 3

```
#function for S-Ricker model with noise
gR_SRicker_noise <- function(S,mu){
  with(mu,rlnorm(1, meanlog=(1+delta)*log(S)+(r*(1-(S^(1))/K)), sdlog=tau_r))
}

#new function that includes noise
SRicker_Fm_noise_sim <- function(S0, mu, Fm, n_iter){

  R0 <- gR_SRicker_noise(S0, mu)

  df <- data.frame(S=S0, R=R0, t=0)

  #empty list to store results
  result.list <- list(df)
  i_model <- 1

  while(i_model<n_iter){

    i_model <- i_model + 1

    R <- gR_SRicker_noise(S0, mu)
    S <- with(mu, (R + Sa*S0)*(1-Fm))

    df <- data.frame(S=S, R=R, t=i_model)

    S0 <- S
    R0 <- R

    #log results
    result.list <- c(result.list, list(df))
  }

  #compile results
  result.df <- bind_rows(result.list)

  return(result.df)

}

S0=0.2

result.list <- list()

for (Fm_i in seq(0,0.6, by=0.2)){

  for (taur_r_i in c(0,0.1)){
  mu <- list(r=2.6, K=3/4, delta=1, Sa=0.6, tau_r=taur_r_i)

  df <- SRicker_Fm_noise_sim(S0, mu, Fm=Fm_i, n_iter=20) %>%
    mutate(tau_r=taur_r_i,
           Fm=Fm_i)

  result.list <- c(result.list, list(df))

  }

}

result.df <- bind_rows(result.list) %>%
  mutate(tau_r=factor(tau_r),
         Fm = factor(Fm))

#make a plot
ggplot(result.df, aes(t,S,color=Fm, linetype=tau_r))+
  theme+
  geom_line()+
  geom_hline(yintercept=0, linetype="dashed")
```

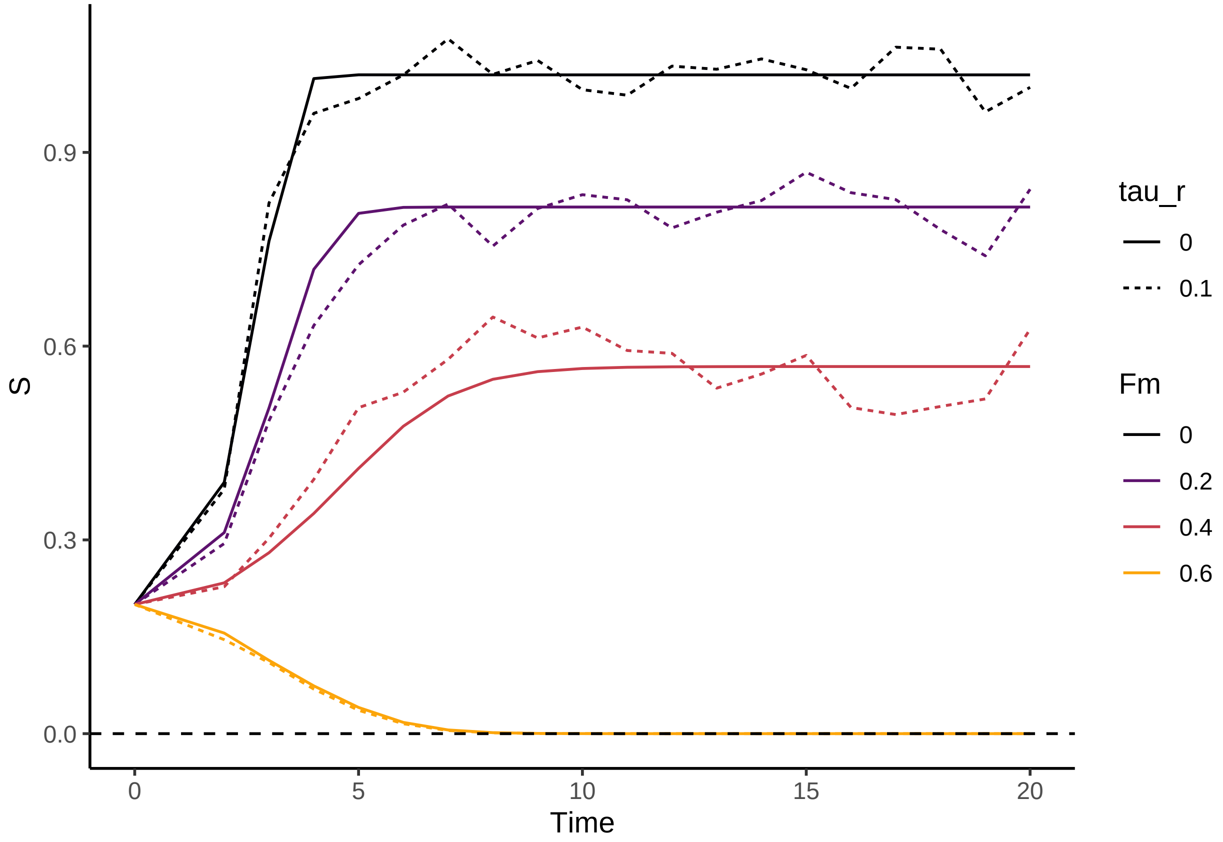


Fig. 3: I modified S0 and the range of fishing mortality tested. The additional “noise” of *tau_r* = 1 is observed in the dashed lines.

This version is including process error - variations in the actual population of the fishery, rather than noise stemming from incomplete counting.

3 Model Comparison

3.1 Task 4

```
#make functions
gR_allee <- function(S,mu){
  with(mu,rlnorm(1, meanlog=(1+delta)*log(S) + (r*(1-(S^(1))/K)), sdlog=tau_r))
}
gS_allee <- function(S,Fm,mu){
  with(mu, (gR_allee(S,mu)+Sa*S)*(1-Fm))
}

gR_ricker <- function(S,mu){
  with(mu, rlnorm(1, meanlog=(1)*log(S) + (r*(1-(S^(1))/K)), sdlog=tau_r))
}
gS_ricker <- function(S,Fm,mu){
```

```r
  with(mu,(gR_ricker(S,mu) + Sa*S)*(1-Fm))
}

simulation <- function(S, Fm, dFm, mu, n_burn, n_iter){

  R <- gR_allee(S,mu)

  result.list <- list()

  for (model_i in 1:(n_iter+n_burn)){

    R <- gR_allee(S,mu)
    S <- with(mu, (R+Sa*S)*(1-Fm))

    Fm <- Fm + dFm

    df <- data.frame(R=R, S=S, Fm=Fm, t=model_i)

    result.list <- c(result.list, list(df))

  }

  result.df <- bind_rows(result.list)

  return(result.df)

}

#run the simulation
mu <- list(r=2.1, K=1, delta=0.6, Sa=0.65, tau_r=0.15)
df <- simulation(S=0.8, Fm=0, dFm=0.005, mu=mu, n_burn=50, n_iter=150)

plot.df <- df %>%
  pivot_longer(!t)

#visualize the simulation results
ggplot(plot.df, aes(t, value, color=name))+
  theme+
  geom_line(show.legend=FALSE)+
  geom_hline(yintercept=0, linetype="dashed")+
  facet_wrap(.~name, ncol=1, strip.position="left", scales="free_y")+
  labs(y=NULL)+
  scale_color_jama()
```

```
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
```

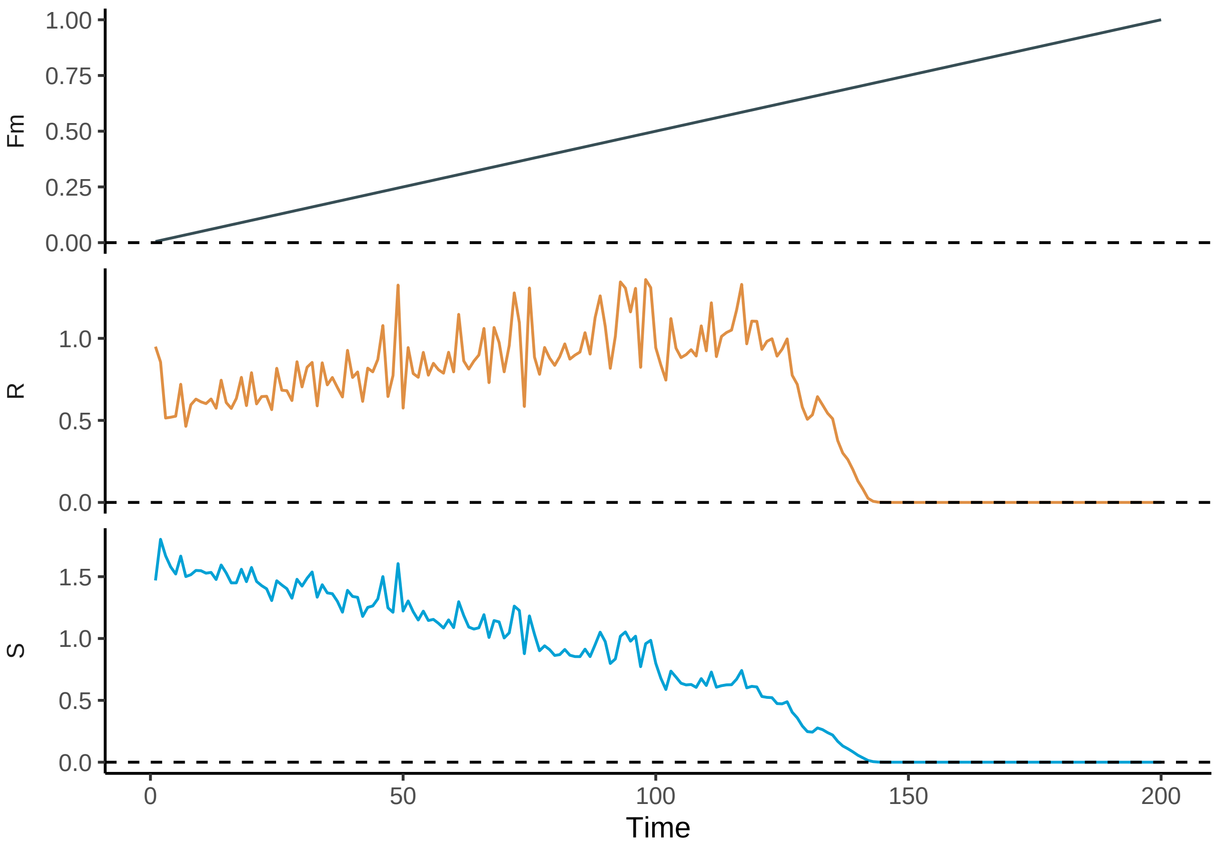


Fig. 4: Changes in `R` and `S` as fishing nortality (`Fm`) increases. We find that `S` decreases as `Fm` increases, but `R` also increases with `Fm` until mortality begins to excees ∼ 0.5 at around the 100 time step.

```
#make two models with different deltas
data_fit_1 <- data.frame(time=61:110, S=df$S[61:110], R=df$R[62:111])
data_fit_2 <- data.frame(time=61:140, S=df$S[61:140], R=df$R[62:141])

suppressWarnings(
  mle2.m0 <- mle2(R ~ dlnorm((1+0)*log(S) + alpha - beta*(S^(1+0)),sdlog=tau_r),
                            # method = "SANN",
                            method = "BFGS",
                            control = list(maxit = 5000),
                            data = data_fit_1,
                            start = list(alpha=2.5,beta=1/2.2,tau_r=0.1))
)

suppressWarnings(
 mle2.m1 <- mle2(R ~ dlnorm((1+delta)*log(S) + alpha - beta*(S^(1)),sdlog=tau_r),
                            # method = "SANN",
                            method = "BFGS",
                            control = list(maxit = 5000),
                            data = data_fit_1,
                            start = list(alpha=2.5,beta=1/2.2,delta=1.0,tau_r=0.1))
)
```

```
anova <- anova(mle2.m0,mle2.m1)
print(anova)
```

```
## Likelihood Ratio Tests
## Model 1: mle2.m0, R~dlnorm((1+0)*log(S)+alpha-beta*(S^(1+0)),sdlog=tau_r)
## Model 2: mle2.m1, R~dlnorm((1+delta)*log(S)+alpha-beta*(S^(1)),sdlog=tau_r)
##   Tot Df Deviance  Chisq Df Pr(>Chisq)
## 1      3  -36.109
## 2      4  -36.583 0.4735  1     0.4914
```

```
summary(anova)
```

```
##      Tot Df        Deviance          Chisq              Df      Pr(>Chisq)
##  Min.   :3.00   Min.   :-36.58   Min.   :0.4735   Min.   :1   Min.   :0.4914
##  1st Qu.:3.25   1st Qu.:-36.46   1st Qu.:0.4735   1st Qu.:1   1st Qu.:0.4914
##  Median :3.50   Median :-36.35   Median :0.4735   Median :1   Median :0.4914
##  Mean   :3.50   Mean   :-36.35   Mean   :0.4735   Mean   :1   Mean   :0.4914
##  3rd Qu.:3.75   3rd Qu.:-36.23   3rd Qu.:0.4735   3rd Qu.:1   3rd Qu.:0.4914
##  Max.   :4.00   Max.   :-36.11   Max.   :0.4735   Max.   :1   Max.   :0.4914
##                                  NA’s   :1        NA’s   :1   NA’s   :1
```

The differences between these two models are not significant, indicating that the model comparison is unable to distinguish between D = 0.6 and D = 0.0. However, we will run multiple iterations to confirm that result:

```
library(foreach) #for parellel processing
```

```
##
## Attaching package: ’foreach’
```

```
## The following objects are masked from ’package:purrr’:
##
##     accumulate, when
```

```
library(doParallel) #for parallel processing
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
#initialize parallel computing
n_cores <- detectCores()-1 #we'll keep one core free
cluster <- makeCluster(n_cores, type="FORK") #fork works better than snow on my machine
registerDoParallel(cl = cluster)

n_iter=500

#run for loop in parallel
p_values <- foreach (i=1:n_iter, .combine=c) %dopar%{
```

```r
  df <- simulation(S=0.8, Fm=0, dFm=0.005, mu=mu, n_burn=50, n_iter=150)

  data_fit <- data.frame(time=61:110, S=df$S[61:110], R=df$R[62:111])

  suppressWarnings(
       mle2.m0 <- mle2(R ~ dlnorm((1+0)*log(S) + alpha - beta*(S^(1+0)),sdlog=tau_r),
                         # method = "SANN",
                         method = "BFGS",
                         control = list(maxit = 5000),
                         data = data_fit,
                         start = list(alpha=2.5,beta=1/2.2,tau_r=0.1))
  )

  suppressWarnings(
       mle2.m1 <- mle2(R ~ dlnorm((1+delta)*log(S) + alpha - beta*(S^(1)),sdlog=tau_r),
                         # method = "SANN",
                         method = "BFGS",
                         control = list(maxit = 5000),
                         data = data_fit,
                         start = list(alpha=2.5,beta=1/2.2,delta=1.0,tau_r=0.1))
  )

  #do the anova
  anova <- anova(mle2.m0,mle2.m1)

  anova[2,5]

}


plot.df <- data.frame(p_value=p_values)

ggplot(plot.df, aes(p_value))+
  theme+
  geom_histogram(color="black", fill="grey")+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(y="Count", x="p value")+
  geom_vline(xintercept=0.05, linetype="dashed")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

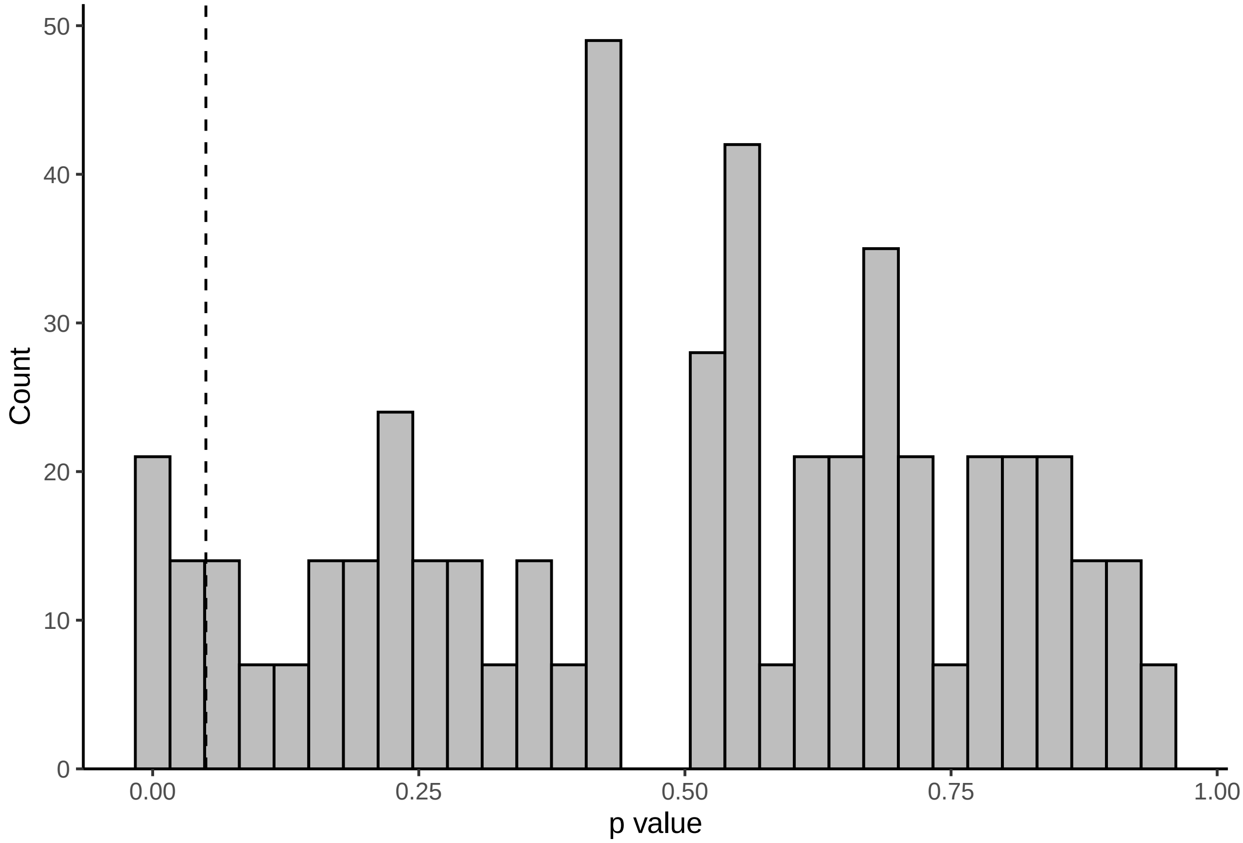
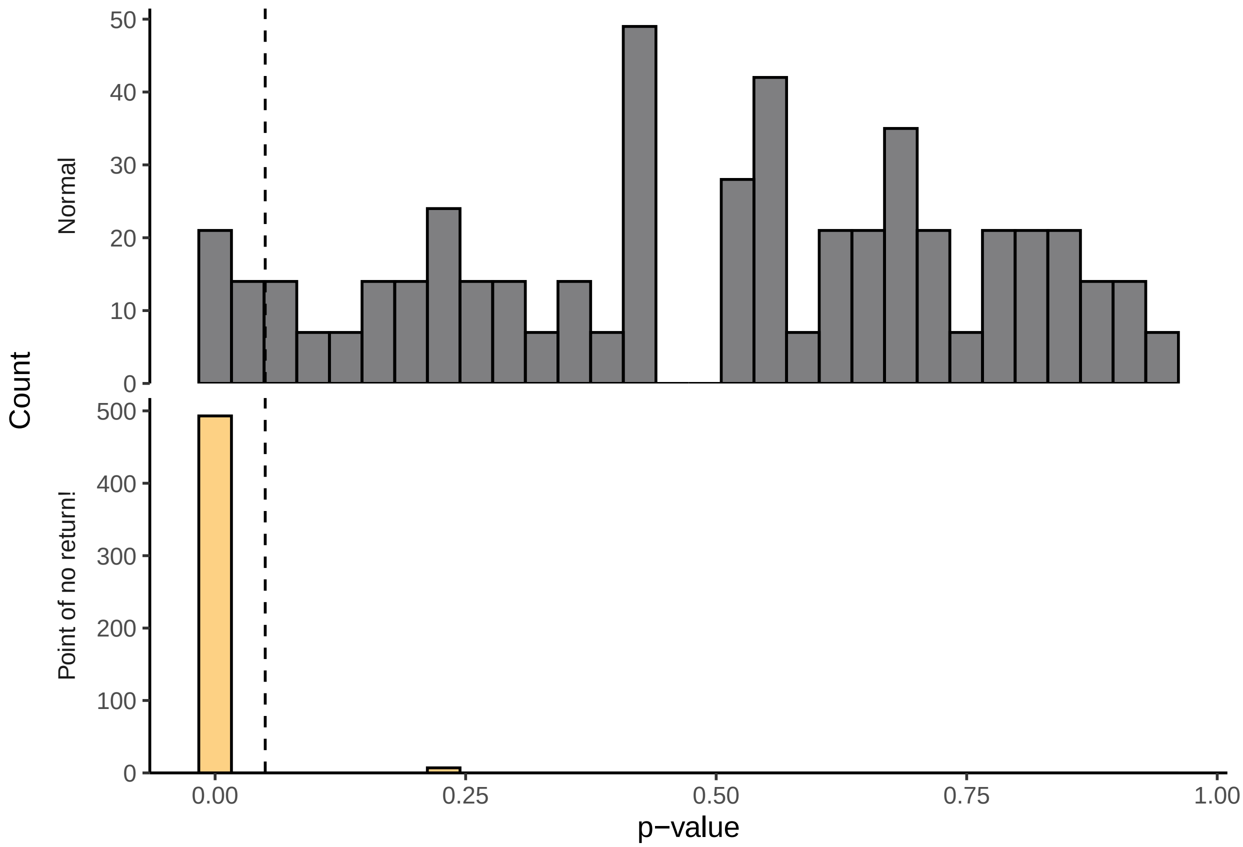


Fig. 5: Histogram of p values from an ANOVA between a model with `D` = 0.0 and `D` = 0.6. Statistical significance ($p < 0.05$) is indicated by the vertical dashed line. In almost all cases, the models are not significantly different.

4 Extension

```
#run for loop in parallel
p_values_2 <- foreach (i=1:n_iter, .combine=c) %dopar%{

  df <- simulation(S=0.8, Fm=0, dFm=0.005, mu=mu, n_burn=50, n_iter=150)

  data_fit <- data.frame(time=61:140, S=df$S[61:140], R=df$R[62:141])

  suppressWarnings(
       mle2.m0 <- mle2(R ~ dlnorm((1+0)*log(S) + alpha - beta*(S^(1+0)),sdlog=tau_r),
                         # method = "SANN",
                         method = "BFGS",
                         control = list(maxit = 5000),
                         data = data_fit,
                         start = list(alpha=2.5,beta=1/2.2,tau_r=0.1))
  )

  suppressWarnings(
       mle2.m1 <- mle2(R ~ dlnorm((1+delta)*log(S) + alpha - beta*(S^(1)),sdlog=tau_r),
                         # method = "SANN",
                         method = "BFGS",
                         control = list(maxit = 5000),
                         data = data_fit,
                         start = list(alpha=2.5,beta=1/2.2,delta=1.0,tau_r=0.1))
  )

  #do the anova
  anova <- anova(mle2.m0,mle2.m1)

  anova[2,5]

}

plot.df <- data.frame(run1=p_values,
                      run2=p_values_2) %>%
  pivot_longer(cols=c("run1", "run2")) %>%
  mutate(name=factor(name, labels=c("Normal","Point of no return!")))

ggplot(plot.df, aes(value, fill=name))+
  theme+
  geom_histogram(color="black", alpha=0.5, show.legend=FALSE)+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(y="Count", x="p-value")+
  geom_vline(xintercept=0.05, linetype="dashed")+
  facet_wrap(.~name, ncol=1, strip.position="left", scales="free_y")
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



```
#end the cluster to keep my computer happy
stopCluster(cluster)
```

Fig. 6: Histogram of p values from an ANOVA between a model with `D` = 0.0 and `D` = 0.6 when using data closer to the "point of no return" (bottom). Statistical significance ($p < 0.05$) is indicated by the vertical dashed line. Closer to the point of no return, we can observe a significant difference between the models most of the time, indicating that the semisiodal growth model is producing different results.

5 Reflection

Stochastic variability has a strong effect on whether and when a population crashes in response to fishing pressures, particularly in the context of a semosiodal growth model with a minimum viable population size. Quantifying the natural variability that would be expected in a population, and potentially adjusting quotas to match that variability on an annual basis, is crucially important for fishery managers.