MT5767 Project 1

Lachlan MacLean, Misha Tseitlin, Liam Gerrity

23/10/2023

This report is the joint product of all of the above mentioned members with each member producing the code, analysis and write up for separate questions. Question 1 was completed by LM, Question 2 by MT and Question 3 by LG.

Question 1

Part a)

For this BAS model, we first look at the potential for stochastic components modelling the sub-process abundances. For survival, we use a binomial distribution to model this sub-process abundance for the first three age classes, with parameters $n_{i,t-1}$ and ϕ_i for $i = 1, 2, 3$, respectively. The last age class abundance for this sub-process is deterministically 0, as we are told that: "Fourth year individuals always die". Next we look at the ageing sub-process. As all animals must necessarily age between time periods (years), we know this must be a deterministic process where all individuals from the previous sub-process for each age class move into to age class above, with the final fourth year individuals staying in this absorbing state. However, this absorbing state is trivial as none of the fourth year individuals have survived into the time period. Finally for reproduction, we choose a Poisson distribution for the first sub-process age class, whilst keeping all other age classes deterministically the same as in the previous sub process. A Poisson distribution is chosen as we know one of the reproduction rates is greater than 1 ($\rho_3 = 1.9$) so a binomial distribution is not appropriate.

The only other stochastic part of this model are the observations. Given we are told the probability of detection is a static $p = 0.5$ and that no double counting can occur, a binomial distribution for the observations of each age class seems appropriate. Here for time t, we use the abundance for class i at time t, $n_{i,t}$, for the number of trials and $p = 0.5$ for the probability of success.

Part b)

In this part we functionalise the theory from a) in order to simulate the dynamics of this model.

```
# b)

# Write an R script to simulate dynamics from this model.
# we do this in a function.

# INPUT :
# n0 = initial populations
# phi = survival probabilities
# rho = reproduction rates
# p = probability of detection
# nyears = number of years over which population is projected
# PROCESS :
# calculate the stochastic population dynamics and observations for each year
# OUTPUT :
```

```r
# y = array of observations for each year, across all age classes
# n = array of abundances for each year, across all age classes

BAS_stoch <- function(n0, phi, rho, p, nyears) {

  # initialise

  # matrix of all abundances, each column is a year, row is age class
  n <- matrix(data = NA, nrow = length(n0), ncol = (nyears+1))

  # matrix of all observations
  y <- n #replicate n

  # initial abundance
  # let the first set of abundances (at t = 0) be the initial population size
  n[,1] <- n0

  # initial observation
  # ASSUMPTION: binomial distribution as constant probability of detection (see Newman)
  y[1,1] <- rbinom(n = 1, size = n[1,1], prob = p)
  y[2,1] <- rbinom(n = 1, size = n[2,1], prob = p)
  y[3,1] <- rbinom(n = 1, size = n[3,1], prob = p)
  y[4,1] <- rbinom(n = 1, size = n[4,1], prob = p)

  # loop over all (other) years
  for (i in 2:(nyears+1)) {

    # calculate stochastic sub-processes for BAS model

    # Survival process
    u_1s1t <- rbinom(n = 1, size = n[1,i-1], prob = phi[1])
    u_1s2t <- rbinom(n = 1, size = n[2,i-1], prob = phi[2])
    u_1s3t <- rbinom(n = 1, size = n[3,i-1], prob = phi[3])
    u_1s4t <- 0

    # Ageing process
    u_2a1t <- 0
    u_2a2t <- u_1s1t
    u_2a3t <- u_1s2t
    u_2a4t <- u_1s3t + u_1s4t

    # Reproduction/Birth process
    u_3b1t <- rpois(n = 1, lambda = rho[1] * u_2a2t) +
              rpois(n = 1, lambda = rho[2] * u_2a3t)
    u_3b2t <- u_2a2t
    u_3b3t <- u_2a3t
    u_3b4t <- u_2a4t

    # new abundances
    n[,i] <- c(u_3b1t, u_3b2t, u_3b3t, u_3b4t)

    # new observations
    y[1,i] <- rbinom(n = 1, size = n[1,i-1], prob = p)
```

```
    y[2,i] <- rbinom(n = 1, size = n[2,i-1], prob = p)
    y[3,i] <- rbinom(n = 1, size = n[3,i-1], prob = p)
    y[4,i] <- rbinom(n = 1, size = n[4,i-1], prob = p)
  }

  # return
  return(list(n=n,y=y))

}

# parameters from specification
phi <- c(0.45, 0.7, 0.7)
rho <- c(0.9, 1.9)
p <- 0.5
n0 <- c(150, 70, 50, 30)
```

Part c)

Finally, we run the function over 25 years. Storing the values for the population dynamics and observations, we then convert this into an appropriate format. This data can then be used to produce clear visualisations in the form of line plots.

```
# c)

# Simulate 25 years of age-specific population dynamics and observations and
# produce an informative visualisation of the data.

# specify how many years to project over
nyears <- 25

# run function for 25 years
BAS_proj <- BAS_stoch(n0 = n0, phi = phi, rho = rho, p = p, nyears = nyears)

# visualise the data using a faceted ggplot.

# convert both outputs to dataframes
proj_n_df <- as.data.frame( cbind(0:nyears, t(BAS_proj$n),
                    rep(2,(nyears+1))) )
proj_y_df <- as.data.frame( cbind(0:nyears, t(BAS_proj$y),
                    rep(1,(nyears+1))) )
# rename
colnames(proj_n_df) <- c("Year", "First Year", "Second Year", "Third Year",
                         "Fourth Year", "State")
colnames(proj_y_df) <- c("Year", "First Year", "Second Year", "Third Year",
                         "Fourth Year", "State")
# pivot longer
proj_n_df_long <- pivot_longer(data = proj_n_df,
                               cols = c("First Year", "Second Year",
                                        "Third Year", "Fourth Year"))
proj_y_df_long <- pivot_longer(data = proj_y_df,
                               cols = c("First Year",  "Second Year",
                                        "Third Year", "Fourth Year"))
proj_df_long <- rbind(proj_n_df_long, proj_y_df_long)
colnames(proj_df_long) <- c("Year", "State", "Age Class", "Abundance")
```

```r
# change variable types
proj_df_long$State <- as.factor(proj_df_long$State)
proj_df_long$`Age Class` <- as.factor(proj_df_long$`Age Class`)

# Can now easily use faceted ggplot
plot_BAS <- ggplot(data = proj_df_long, aes(x = Year, y = Abundance)) +
  geom_line(aes(colour = State), size = 1) +
  scale_colour_manual("State", breaks = c(2, 1),
                      values = c("blue", "grey"),
                      labels = c("Dynamics","Observations")) +
  facet_wrap(~factor(`Age Class`, levels = c("First Year", "Second Year",
                                             "Third Year", "Fourth Year")),
             scales = "free_y") +
  ylab("Abundance") +
  ggtitle("Population Dynamics and Observations Projected over Time")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
plot_BAS
```

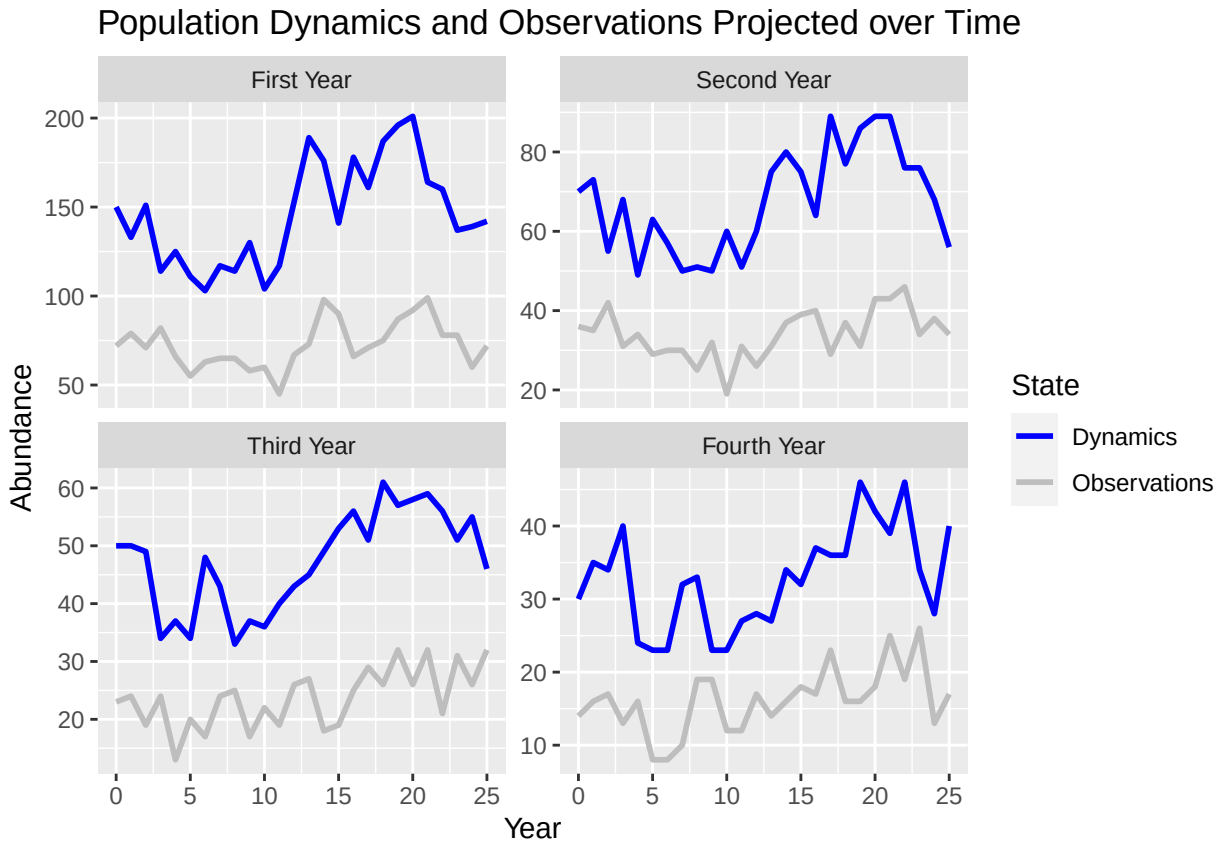

Figure 1: INSERT TITLE HERE

Question 3

We will fit models to the data using values for growth rate, r, and carrying capacity, K, dependent on the recorded rainfall, R, according to the data set. We will then look to test the impact of time lag when fitting models to the observed data by finding the coefficients of r and K with respect to R_t and R_{t-1}.

modelling r dependent on R_t and R_{t-1}

The previously used model has now been extended to include temporal variation in r:

$r = exp(\alpha_0 + \alpha_1 R_t)$

$r = exp(\alpha_0 + \alpha_1 R_{t-1})$

By finding the likelihood of the model and using an optimisation function, parameter values of α_0 and α_1 can be found to calculate r for each time period. Each model can then be used to estimate N as shown by Figure 3. The AIC of each model is displayed in Table 1 and used for model comparison.

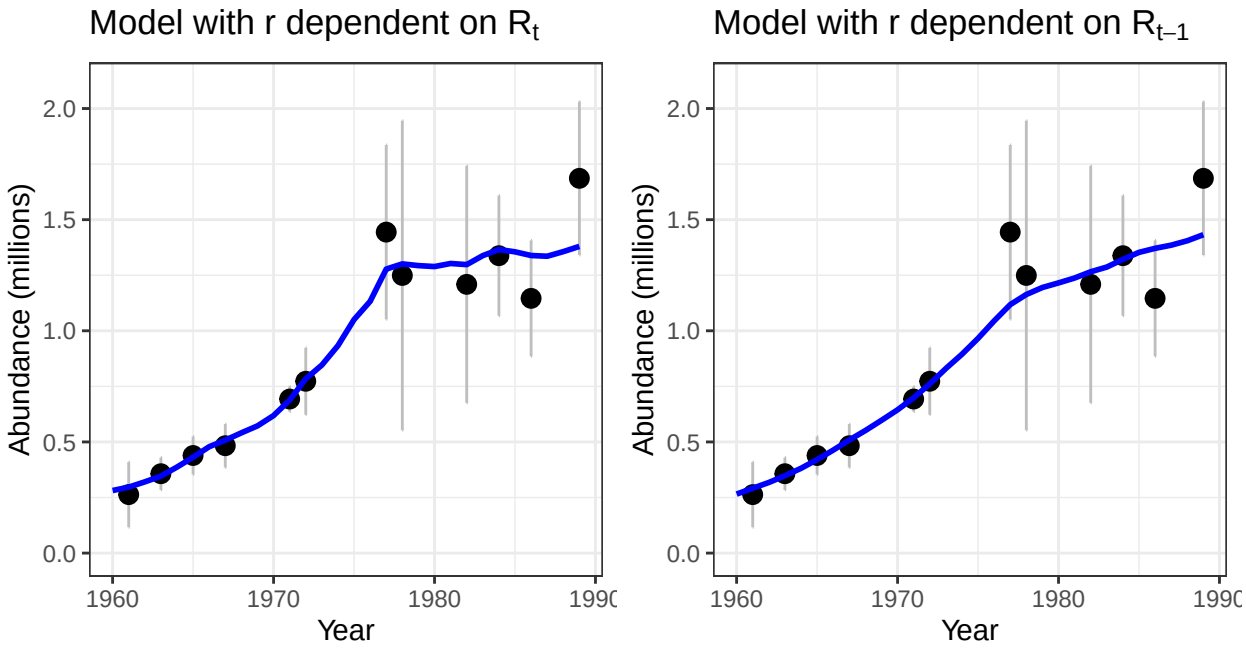

Figure 2: A plot of the model fits with r depending on R_t and R_{t-1} respectively

Table 1: Coefficients and model comparisons for time varying r

	ALPHA 0	ALPHA 1	AIC	BIC
r ~ Rt	-2.984800	0.6323085	-18.96679	-17.02716
r ~ Rt-1	-2.344128	0.1565288	-17.14583	-15.20620

By comparing AIC values, the first model $r = exp(\alpha_0 + \alpha_1 R_t)$is preferabe as it produces the lowest AIC value. Looking at the coefficent estimates we can see that α_0 is equal to -2.98 which is negative. This will mean that the intercept of the r equation when there is no rainfall is less than 1. WHY / WHAT DOES THIS SHOW???. It is also shown for this optimal model that $\alpha_1 = 0.632$. This positive value shows that rainfall

has a positive impact on the growth rate. To put this into context we can conclude that this model implies that as rainfall increases, the maximum growth rate will also increase causing a greater abundance estimate.

Modelling K dependent on R_t and R_{t-1}

Using the same optimization methods used previously when modelling the temporal variation of r, we can now extend the model to include temporal variation of k. The two formulations of K modeled are:

$K = exp(\beta_0 + \beta_1 R_t)$

$K = exp(\beta_0 + \beta_1 R_{t-1})$

Again through the use of optimisation, coefficient value for β_0 and β_1 were used to model estimates for N, plotted in Figure 3, alongside the respective AIC values displayed in table 2.

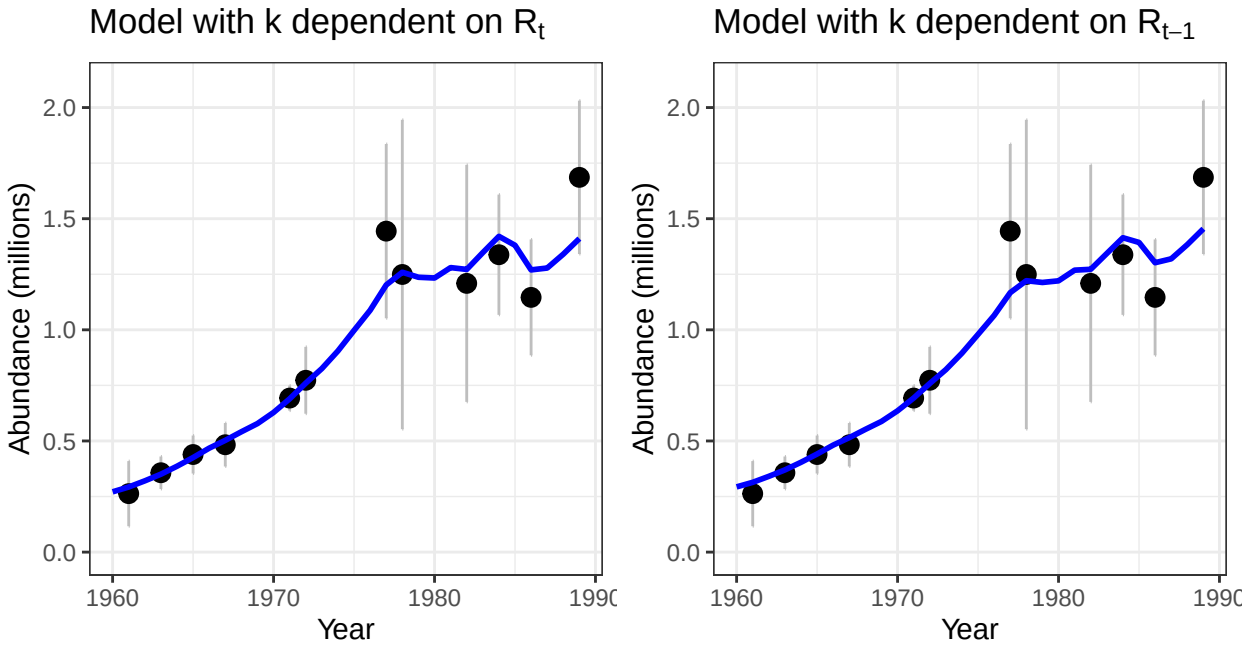

Figure 3: A plot of the model fits with K depending on R_t and R_{t-1} respectively

Table 2: Coefficients and model comparisons for time varying K

	BETA 0	BETA 1	AIC	BIC
k ~ Rt	-0.4969040	1.020015	-19.89674	-17.95711
k ~ Rt-1	-0.5030827	1.194308	-14.42609	-12.48647

From comparing the AIC values of the competing models in Table 2, the optimal choice appears to use K as a function of R_t. Within this model $\beta_0 = -0.497$ and $\beta_1 = 1.02$. This positive value of β_1 implies that rain has a positive effect on the carrying capacity and so more rain will lead to a greater carrying capacity.

The Optimal Model?

By comparing the AIC and BIC scores of all 4 models, it would appear that the optimal model according to the lowest score is the one which uses K dependent on R_t followed by the dependence of r on R_t.

modelling both temporal r and K

From the above model comparisons it has been shown that the models perform better when using r or K dependent on R_{t-1}, but what if we use a r and K as follows:

$r = exp(\alpha_0 + \alpha_1 R_{t-1})$

$K = exp(\beta_0 + \beta_1 R_{t-1})$

This arises to model fit displayed in Figure 4. Table 3 displays the coefficient values for the equations above for the model in addition to those found in the optimal models using only a single non constant variable to allow for comparison.

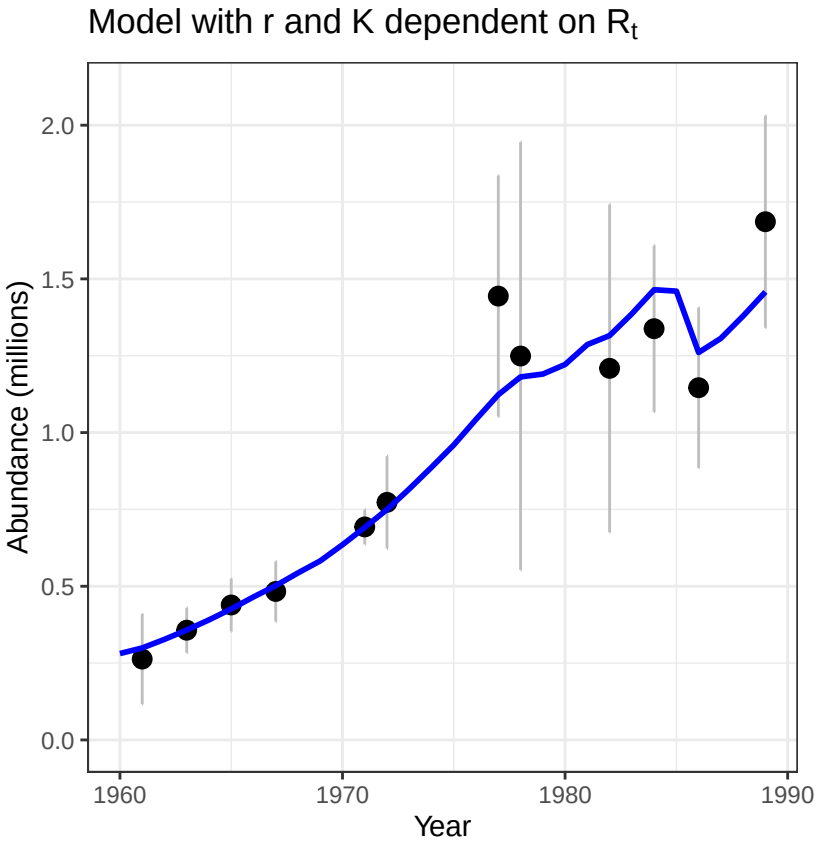

Figure 4: Model with r and K dependent on R_t

Table 3: Coefficient and model comparison values

	ALPHA 0	ALPHA 1	BETA 0	BETA 1	AIC	BIC
r ~ Rt	-2.984800	0.6323085	NA	NA	-18.96679	-17.02716
K ~ Rt	NA	NA	-0.496904	1.020015	-19.89674	-17.95711
r & K ~ Rt	-2.109316	-0.1507751	-1.350160	2.395443	-16.98160	-14.55707

From Table 3 it can be seen that both the AIC and BIC for the final model are greater than the models using only a single variable dependent on Rainfall. This would suggest that there is no immediate benefit to using a more complex model utilising multiple non constant coefficients.

It can also be seen by comparison the coefficient values for the model with both α_1 and β_1 appear to have a smaller magnitude than those of the models with only a single time varying variable suggesting that compared

to the individual models the impact of rainfall on each coeffiecient has been reduced. α_1 has now also taken a negative value implying a negative relationship between the maximum growth rate and rainfall, contradicting the observations made previously in the model with on r displaying temporal variation.

Question 3 Code

```
#QUESTION 3A

#Create a datafrae to store the optimal alpha values as well as AIC and BIC values
alpha_values <- data.frame(matrix(0, nrow = 2, ncol = 4))
rownames(alpha_values ) = c('r ~ Rt', 'r ~ Rt-1')               #each row represents a model
colnames(alpha_values )= c('ALPHA 0' , 'ALPHA 1', 'AIC', 'BIC') #each column represents a value

#combined_values will store data from the optimal model in a) and b) to provide a final
#comparison
combined_values <- data.frame(matrix(0, nrow = 3, ncol = 6))
rownames(combined_values ) = c('r ~ Rt', 'K ~ Rt', 'r & K ~ Rt')
colnames(combined_values )= c('ALPHA 0' , 'ALPHA 1','BETA 0', 'BETA 1', 'AIC', 'BIC')

#rainr_t will find the negative log-likelihood for the model constructed with the input
#parameters. Here r varies with time.
rainr_t <- function(pars, years, removals, Nhat, SEhat, rain, t){
  #set the initial parameters
  N0 <- exp(pars[1])
  alpha0 <- pars[2]
  alpha1 <- pars[3]
  k <- exp(pars[4])

  #create a list the length of the number of years we ish to project to store the projections
  N <- numeric(years)
  r <- numeric(years)
  N[1] <- N0
  r[1] <- NA

  #generate population dynamics
  for(i in 2:years){
    #using the input parameter t the function can specify the time lag we wish to consider when
    #finding r with respect to R
    if (t == 't-1'){
      r[i] <- exp(alpha0 + alpha1*rain[(i-1)])
    }
    else{
      r[i] <- exp(alpha0 + alpha1*rain[i])
    }
    N[i] = N[i-1] + r[i] * N[i-1] * (1-N[i-1]/k) - removals[i-1]
  }

 #findthe negative log likelihood
  negloglik <- -sum(dnorm(Nhat,N,SEhat, log=TRUE), na.rm=TRUE)

  return(negloglik)   #return the negative log likelihood
}

#to run the function we must now set some initil values for the unknown parameters we wish to
#test
N0 <- log(0.1)
alpha0 <- log(1.5)
alpha1 <-  0.2
```

```r
k <- log(1.5)
parsr <- c(N0,alpha0,alpha1,k)          #store these values in a list

#run the optimiser function to find the optimal coefficient values
optimised_values <- optim(parsr,
                  fn = rainr_t,
                  years = nrow(wildebeest),
                  removals = wildebeest$Catch,
                  Nhat = wildebeest$Nhat,
                  SEhat = wildebeest$sehat,
                  rain = wildebeest$rain,
                  t = 't')

# set up parameters using the optimised values above
N0 <- exp(optimised_values$par[1])
alpha0 <- optimised_values$par[2]
alpha1 <- optimised_values$par[3]
k <- exp(optimised_values$par[4])
pars <- c(N0, alpha0, alpha1,k)
N <- numeric(nrow(wildebeest))
r <- numeric(nrow(wildebeest))

#store these values in the previously mentioned dataframes for later comparison to other models
alpha_values[1,1] = alpha0
alpha_values[1,2] = alpha1
alpha_values[1,3] <- 2 * optimised_values$value + 2*(length(optimised_values$par))
alpha_values[1,4] <- 2 * optimised_values$value + log(12) * (length(optimised_values$par))

combined_values[1,1] = alpha0
combined_values[1,2] = alpha1
combined_values[1,3] = NA
combined_values[1,4] = NA
combined_values[1,5] <- 2 * optimised_values$value + 2*(length(optimised_values$par))
combined_values[1,6] <- 2 * optimised_values$value + log(12) * (length(optimised_values$par))

#first year
N[1] <- N0
r[1] <- NA #1st K not in the model

#subsequent years
for(i in 2:nrow(wildebeest)){
  r[i] <- exp(alpha0 + alpha1*wildebeest$rain[i])
  N[i] = N[i-1] + r[i] * N[i-1] * (1-N[i-1]/k) - wildebeest$Catch[i-1]
}

tmp_wilde <- data.frame(Nhat = wildebeest$Nhat,
                        Nproj = N,
                        Year = wildebeest$year,
                        lci = wildebeest$lci,
                        uci = wildebeest$uci)

#plot the projections and the estimates
plot1 <- ggplot(tmp_wilde, aes(x=Year, y=Nproj)) +
```

```
  geom_errorbar(aes(ymin=lci,ymax=uci), width=0, color="grey") +
  geom_point(aes(x=Year,y=Nhat), size=3) +
  geom_line(aes(x=Year,y=Nproj),color="blue", size=1) +
  ylim(0,2.1) + ylab("Abundance (millions)") +
  labs(title = expression("Model with r dependent on R"[t])) +
  theme_bw() +
  theme(aspect.ratio = 1)

#---- rt ~ Rt-1 ----
#reset the intial parameter estimates. These values have been used as other combination caused
#false results due to the optimisation finding local minima for the negative log likelihood
parsr <- c(N0,optimised_values$par[2],optimised_values$par[3],optimised_values$par[4])

#rerun the optimiser
optimised_values <- optim(parsr,
                          fn = rainr_t,
                          years = nrow(wildebeest),
                          removals = wildebeest$Catch,
                          Nhat = wildebeest$Nhat,
                          SEhat = wildebeest$sehat,
                          rain = wildebeest$rain,
                          t = 't-1')

# set up parameters using the optimised values above
N0 <- exp(optimised_values$par[1])
beta0 <- optimised_values$par[2]
beta1 <- optimised_values$par[3]
k <- exp(optimised_values$par[4])
N <- numeric(nrow(wildebeest))
r <- numeric(nrow(wildebeest))

#store optimal values for later comparison beteen models
alpha_values[2,1] = beta0
alpha_values[2,2] = beta1
alpha_values[2,3] <- 2 * optimised_values$value + 2*(length(optimised_values$par))
alpha_values[2,4] <- 2 * optimised_values$value + log(12) * (length(optimised_values$par))

#first year
N[1] <- N0
r[1] <- NA #1st K not in the model

#subsequent years
for(i in 2:nrow(wildebeest)){
  r[i] <- exp(beta0 + beta1*wildebeest$rain[i-1])
  N[i] = N[i-1] + r[i] * N[i-1] * (1-N[i-1]/k) - wildebeest$Catch[i-1]
}

tmp_wilde <- data.frame(Nhat = wildebeest$Nhat,
                        Nproj = N,
                        Year = wildebeest$year,
                        lci = wildebeest$lci,
                        uci = wildebeest$uci)
```

```r
#create a plot of the projections and the estimates
plot2 <-ggplot(tmp_wilde, aes(x=Year, y=Nproj)) +
  geom_errorbar(aes(ymin=lci,ymax=uci), width=0, color="grey") +
  geom_point(aes(x=Year,y=Nhat), size=3) +
  geom_line(aes(x=Year,y=Nproj),color="blue", size=1) +
  ylim(0,2.1) + ylab("Abundance (millions)") +
  labs(title = expression("Model with r dependent on R"[t-1])) +
  theme_bw() +
  theme(aspect.ratio = 1)

#plot the projections of the models using different time lags side by side for comparison
grid.arrange(plot1, plot2, ncol = 2)
#output the table of coefficient values and comparison metrics
kable(alpha_values, caption = "Coefficients and model comparisons for time varying r")
```

```r
# QUESTION 3B


#create a new dataframe to store coefficient values, AIC and BIC
beta_values <- data.frame(matrix(0, ncol = 4, nrow = 2))
rownames(beta_values ) = c('k ~ Rt', 'k ~ Rt-1')
colnames(beta_values )= c('BETA 0' , 'BETA 1', 'AIC', 'BIC')

#---- k ~ Rt ----
#create a new function to find the negative log likelihood of models based on in put parameter
#values. This function is the same as the one from 3a) however this time K is dependent on Rt
#instead of r.
rainK_t <- function(pars, years, removals, Nhat, SEhat, rain, t){

  N0 <- exp(pars[1])
  r <- exp(pars[2])
  beta0 <- pars[3]   #not transformed |
  beta1 <- pars[4]   #not transformed |-> note extra parameter now
  N <- numeric(years)
  k <- numeric(years)
  N[1] <- N0
  k[1] <- NA          #1st K not in the model

  for(i in 2:years){    #generate population dynamics:
    if (t == 't-1'){
      k[i] <- exp(beta0 + beta1*rain[(i-1)]) #link fn of the linear predictor
    }
    else{
      k[i] <- exp(beta0 + beta1*rain[i]) #link fn of the linear predictor
    }
    N[i] = N[i-1] + (r * N[i-1] * (1-N[i-1]/k[i])) - removals[i-1]
  }
  negloglik <- -sum(dnorm(Nhat,N,SEhat, log=TRUE), na.rm=TRUE)

  return(negloglik)
}

#input starting estimates for the parameters
N0 <- log(0.1)
```

```
r <- log(0.25)
beta0 <-  log(0.5)
beta1 <-  log(0.5)
parsk <- c(N0,r,beta0,beta1)

#optimise the parameter estimates according to the observations
fit_rainK <- optim(parsk,
                   fn = rainK_t,
                   years = nrow(wildebeest),
                   removals = wildebeest$Catch,
                   Nhat = wildebeest$Nhat,
                   SEhat = wildebeest$sehat,
                   rain = wildebeest$rain,
                   t = 't')

# set up optimised parameters
N0 <- exp(fit_rainK$par[1])
r <- exp(fit_rainK$par[2])
beta0 <- fit_rainK$par[3]
beta1 <- fit_rainK$par[4]
pars <- c(N0, r, beta0,beta1)

#store the values for later comparison in the previously created dataframes
beta_values[1,1] = beta0
beta_values[1,2] = beta1
beta_values[1,3] <- 2 * fit_rainK$value + 2*length(fit_rainK$par)
beta_values[1,4] <- 2 * fit_rainK$value + log(12)*length(fit_rainK$par)

combined_values[2,1] = NA       #note that the model has no alpha values
combined_values[2,2] = NA
combined_values[2,3] = beta0
combined_values[2,4] = beta1
combined_values[2,5] <- 2 * fit_rainK$value + 2*length(fit_rainK$par)
combined_values[2,6] <- 2 * fit_rainK$value + log(12)*length(fit_rainK$par)

#create vectors to store the projections
N <- numeric(nrow(wildebeest))
k <- numeric(nrow(wildebeest))
#first year
N[1] <- N0
k[1] <- NA #1st K not in the model

#subsequent years
for(i in 2:nrow(wildebeest)){
  k[i] <- exp(beta0 + beta1*wildebeest$rain[i])
  N[i] = N[i-1] + r * N[i-1] * (1-N[i-1]/k[i]) - wildebeest$Catch[i-1]
}

tmp_wilde <- data.frame(Nhat = wildebeest$Nhat,
                        Nproj = N,
                        Year = wildebeest$year,
                        lci = wildebeest$lci,
                        uci = wildebeest$uci)
```

```r
#create a plot of the projections and the estimates
plot3 <- ggplot(tmp_wilde, aes(x=Year, y=Nproj)) +
  geom_errorbar(aes(ymin=lci,ymax=uci), width=0, color="grey") +
  geom_point(aes(x=Year,y=Nhat), size=3) +
  geom_line(aes(x=Year,y=Nproj),color="blue", size=1) +
  ylim(0,2.1) + ylab("Abundance (millions)") +
  labs(title = expression("Model with k dependent on R"[t])) +
  theme_bw() +
  theme(aspect.ratio = 1)

#---- Kt ~ Rt-1 ----

# with the same initial conditions as above rerun the optimiser this time specifyingthe time
#lag as t-1 to find new optimal coefficient estimates for the model

fit_rainK <- optim(parsk,
                   fn = rainK_t,
                   years = nrow(wildebeest),
                   removals = wildebeest$Catch,
                   Nhat = wildebeest$Nhat,
                   SEhat = wildebeest$sehat,
                   rain = wildebeest$rain,
                   t = 't-1')

# set up optimised parameters
N0 <- exp(fit_rainK$par[1])
r <- exp(fit_rainK$par[2])
beta0 <- fit_rainK$par[3]
beta1 <- fit_rainK$par[4]
pars <- c(N0, r, beta0,beta1)

#store the values of beta and calculate AIC and BIC
beta_values[2,1] = beta0
beta_values[2,2] = beta1
beta_values[2,3] <- 2 * fit_rainK$value + 2*length(fit_rainK$par)
beta_values[2,4] <- 2 * fit_rainK$value + log(12)*length(fit_rainK$par)

N <- numeric(nrow(wildebeest))
k <- numeric(nrow(wildebeest))
N[1] <- N0
k[1] <- NA #1st K not in the model

#subsequent years
for(i in 2:nrow(wildebeest)){
  k[i] <- exp(beta0 + beta1*wildebeest$rain[i])
  N[i] = N[i-1] + r * N[i-1] * (1-N[i-1]/k[i]) - wildebeest$Catch[i-1]
}

tmp_wilde <- data.frame(Nhat = wildebeest$Nhat,
                        Nproj = N,
                        Year = wildebeest$year,
                        lci = wildebeest$lci,
                        uci = wildebeest$uci)
```

```r
#plot the projections and the estimates
plot4 <- ggplot(tmp_wilde, aes(x=Year, y=Nproj)) +
  geom_errorbar(aes(ymin=lci,ymax=uci), width=0, color="grey") +
  geom_point(aes(x=Year,y=Nhat), size=3) +
  geom_line(aes(x=Year,y=Nproj),color="blue", size=1) +
  ylim(0,2.1) + ylab("Abundance (millions)") +
  labs(title = expression("Model with k dependent on R"[t-1])) +
  theme_bw() +
  theme(aspect.ratio = 1)

#create a side by side plot of the different models projections of N
grid.arrange(plot3, plot4, ncol = 2)
#output the table to compare the coefficient values and AIC / BIC values
kable(beta_values, caption = "Coefficients and model comparisons for time varying K")
```

```r
# QUESTION 3 EXTRA


#----Modelling r & K ~ t-1 ----
#create a function to find the negative log likelihood for a model with both r and K dependent
#on Rt. This is an extension of the functions within 3a and 3b by simply calculating r and K
rainr_K_t <- function(pars, years, removals, Nhat, SEhat, rain){
  N0 <- exp(pars[1])
  alpha0 <- pars[2]
  alpha1 <- pars[3]
  beta0 <- pars[4]
  beta1 <- pars[5]

  N <- numeric(years)
  r <- numeric(years)
  k <- numeric(years)
  N[1] <- N0
  r[1] <- NA
  k[1] <- NA

  #generate population dynamics:
  for(i in 2:years){
    r[i] <- exp(alpha0 + alpha1*rain[(i)])
    k[i] <- exp(beta0 + beta1*rain[(i)])
    N[i] = N[i-1] + r[i] * N[i-1] * (1-N[i-1]/k[i]) - removals[i-1]
  }

  negloglik <- -sum(dnorm(Nhat,N,SEhat, log=TRUE), na.rm=TRUE)

  return(negloglik)   #return the negative log likelihood
}

#set initial parameter values
N0 <- log(0.1)
alpha0 <- log(0.5)
alpha1 <-  log(0.5)
beta0 <-  log(0.5)
beta1 <-  log(0.5)
```

```r
parsr <- c(N0,alpha0, alpha1, beta0, beta1)

#optimise parameter values for the model according to the observations
optimised_values <- optim(parsr,
                          fn = rainr_K_t,
                          years = nrow(wildebeest),
                          removals = wildebeest$Catch,
                          Nhat = wildebeest$Nhat,
                          SEhat = wildebeest$sehat,
                          rain = wildebeest$rain)

# set up parameters using the optimised values above
N0 <- exp(optimised_values$par[1])
alpha0 <- optimised_values$par[2]
alpha1 <- optimised_values$par[3]
beta0 <- optimised_values$par[4]
beta1 <- optimised_values$par[5]
pars <- c(N0,alpha0, alpha1, beta0,beta1,k)

#store the model coefficient estimates as well as the AIC and BIC
combined_values[3,1] = alpha0
combined_values[3,2] = alpha1
combined_values[3,3] = beta0
combined_values[3,4] = beta1
combined_values[3,5] <- 2 * optimised_values$value + 2*(length(optimised_values$par))
combined_values[3,6] <- 2 * optimised_values$value + log(12) * (length(optimised_values$par))

N <- numeric(nrow(wildebeest))
r <- numeric(nrow(wildebeest))
k <- numeric(nrow(wildebeest))
#first year
N[1] <- N0
r[1] <- NA
k[1] <- NA

#Calculate the projections for the subsequent years using the temproal r and K values
for(i in 2:nrow(wildebeest)){
  r[i] <- exp(alpha0 + alpha1*wildebeest$rain[(i)])
  k[i] <- exp(beta0 + beta1*wildebeest$rain[(i)])
  N[i] = N[i-1] + r[i] * N[i-1] * (1-N[i-1]/k[i]) - wildebeest$Catch[i-1]
}

tmp_wilde <- data.frame(Nhat = wildebeest$Nhat,
                        Nproj = N,
                        Year = wildebeest$year,
                        lci = wildebeest$lci,
                        uci = wildebeest$uci)

#create a plot of the projections of N
plot5 <- ggplot(tmp_wilde, aes(x=Year, y=Nproj)) +
  geom_errorbar(aes(ymin=lci,ymax=uci), width=0, color="grey") +
  geom_point(aes(x=Year,y=Nhat), size=3) +
  geom_line(aes(x=Year,y=Nproj),color="blue", size=1) +
```

```
  ylim(0,2.1) + ylab("Abundance (millions)") +
  labs(title = expression("Model with r and K dependent on R"[t])) +
  theme_bw() +
  theme(aspect.ratio = 1)

grid.arrange(plot5, ncol = 1)
#output the datagrame as a table for analysis
kable(combined_values, caption = "Coefficient and model comparison values")
```