Hog

0. Libraries

```
library(knitr)
library(tidyverse)
library(janitor)
library(broom)
library(reshape2)
```

1. Data

```
knitr::read_chunk('data.R')
```

```
# Parameters.
pars = read_csv("pars.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
##   `Name (Long)` = col_character(),
##   `Name (Short)` = col_character(),
##   Function = col_character(),
##   Default = col_double(),
##   Pessimistic = col_double(),
##   Optimistic = col_double(),
##   Units = col_character(),
##   Module = col_character(),
##   SourceDef = col_character(),
##   SourcePess = col_character(),
##   SourceOpt = col_character(),
##   SamePess = col_double(),
##   SameOpt = col_double()
## )
```

```
# Market
# Prices and characteristics of buche in end markets. Several sources.
ma_dat = read_csv("ma_dat.csv")
```

```
## Parsed with column specification:
## cols(
##   year = col_double(),
##   p = col_double(),
##   q = col_double(),
##   kg = col_double(),
##   cond = col_character(),
##   place = col_character(),
##   spp = col_character(),
##   g = col_double(),
##   bicond = col_double()
## )
```

```r
# Aquaculture
#  Marginal mortalities per month in aquaculture from COF (2017).
aq_mort_dat = read_csv("aq_mort_dat.csv")
```

```
## Parsed with column specification:
## cols(
##   a_months = col_double(),
##   m_months = col_double()
## )
```

```r
# Fishery
#  Biomass at age for 2017 from INAPESCA (2018).
fi_biom_dat = read_csv("fi_biom_dat.csv")
```

```
## Parsed with column specification:
## cols(
##   Edad = col_double(),
##   Biomasa = col_double()
## )
```

2. Functions

```r
knitr::read_chunk('functions.R')
```

```r
# Ages to lengths - Von Bertalanffy Growth Function.
fun_a_l = function(a, linf, k, t0){l = linf * (1 - exp(-k * (a - t0)))}

# Lengths to weights.
fun_l_w = function(a, l, b){w = a * l ^ b}

# Ages to natural mortalities.
fun_a_nmort = function(a, a_mat, a_old, m_juv, m_mat, m_old){s = ifelse(a < a_mat, m_juv, ifelse(a < a_

# Lengths to selectivities.
fun_l_s = function(l, a, b, m){s = a / (1 + exp(b - m * l))}

# Ages to bycatch mortalities.
fun_a_bmort = function(a, b_b, a_mat, n0){b = ifelse(a < a_mat, 0.20, 0)}#(b_b / round(a_mat)) / n0[rou

# Numbers at age to recruitment - Shepherd Recruitment Function.
fun_rec = function(n, a_rec, b_rec, d_rec){n0 = (a_rec * n) / ((1 + n / b_rec) ^ d_rec)}

# Production and grams to price in multivariate inverse demand specification.
fun_p = function(q, g, a_ma, b_ma, c_ma){p = q * a_ma + g ^ b_ma + c_ma
                                          return(ifelse(p > 0, p, 0))}

# Ages to natural mortalities in aquaculture.
fun_a_aqmort = function(a, b1, b2, mmin){m = b1 * exp(b2 * a * 12) + mmin} # Turn that 1 into a paramet

# Weights to optimal stocking densities in numbers.
fun_ns = function(cage_size_aq, dens_aq, w){ns = (cage_size_aq * dens_aq) / w}
```

3. Set-Up

```r
knitr::read_chunk('set.R')
```

```r
# Market
#  Estimate inverse demand.
nlm = nls(p ~ q * a + (g ^ b) + c, data = ma_dat, start = c(a = -5.00, b = 2.00, c = 20))
#  Clean results.
nlm_tidy = tidy(nlm)

# Aquaculture
#  Estimate incremental mortalities.
#  Regression:
am_reg = nls(m_months ~ b1 * exp(b2 * a_months), aq_mort_dat, start = list(b1 = 8.00, b2 = - 1.00))
#  Clean results.
am_reg_tidy = tidy(am_reg)


# Pull initial and intermediate parameters together.
pars_full = pars %>%
  # Add market outputs to parameter table.
  add_row(name_long = "Quantity Elasticity", name_short = "a_ma", "function" = "Demand", default = nlm_t
  add_row(name_long = "Size Premium", name_short = "b_ma", "function" = "Demand", default = nlm_tidy$est
  add_row(name_long = "Choke Price", name_short = "c_ma", "function" = "Demand", default = nlm_tidy$esti
  # Add aquaculture outputs to parameter table.
  add_row(name_long = "Aq. Mortality Coefficient", name_short = "b1_mort_aq", "function" = "Aquaculture
  add_row(name_long = "Aq. Mortality Coefficient", name_short = "b2_mort_aq", "function" = "Aquaculture


# Turn parameters into a matrix for multiple model runs.
pars = pars_full %>%
  select(2, 4:6) %>%
  column_to_rownames(var = "name_short")

# Build out a matrix of parameters for sensitivity analysis.
pars[4:6] = pars[1:3]
pars[]
```

```
##                    default   pessimistic    optimistic     default.1
## linf_al_aq    2.000000e+02  2.000000e+02  2.000000e+02  2.000000e+02
## k_al_aq       1.550000e-01  1.550000e-01  3.162000e-01  1.550000e-01
## t0_al_aq     -6.500000e-01 -6.500000e-01 -6.500000e-01 -6.500000e-01
## a_lw          4.128000e-06  4.128000e-06  4.128000e-06  4.128000e-06
## b_lw          3.246740e+00  3.246740e+00  3.246740e+00  3.246740e+00
## dens_aq       2.000000e+01  1.000000e+01  3.000000e+01  2.000000e+01
## cage_size_aq  8.000000e+03  8.000000e+03  1.100000e+04  8.000000e+03
## sale_size_aq  2.000000e+00  2.000000e+00  2.000000e+00  2.000000e+00
## by1           1.610000e-02  1.370000e-02  1.610000e-02  1.610000e-02
## by2           3.100000e-01  3.100000e-01  3.700000e-01  3.100000e-01
## f_z           7.000000e-01  7.000000e-01  7.000000e-01  7.000000e-01
## g_z           6.500000e+00  6.500000e+00  6.500000e+00  6.500000e+00
## h_z           4.000000e-02  8.000000e-02  1.000000e-02  4.000000e-02
## j_z           2.000000e+00  2.000000e+00  1.545000e+00  2.000000e+00
```

```
## k_z           1.700000e+05  1.700000e+05  1.700000e+05  1.700000e+05
## l_z           2.000000e+00  2.000000e+00  2.000000e+00  2.000000e+00
## mmin_aq       1.000000e+00  1.000000e+01  0.000000e+00  1.000000e+00
## disc_aq       9.000000e-01  1.000000e+00  7.500000e-01  9.000000e-01
## loss          1.000000e-01  5.000000e-02  1.500000e-01  1.000000e-01
## switch_aq     1.000000e+00  0.000000e+00  1.000000e+00  1.000000e+00
## b_2017        3.642900e+04  2.570900e+04  4.714900e+04  3.642900e+04
## bprop_2017    9.259457e-01  6.534667e-01  1.198425e+00  9.259457e-01
## f_2017        1.400000e+03  1.400000e+03  1.400000e+03  1.400000e+03
## fn_2017       8.640000e+04  8.640000e+05  2.100000e+04  8.640000e+04
## e_2017        1.200000e+02  1.000000e+02  2.000000e+02  1.200000e+02
## b_b           8.000000e+04  8.000000e+04  8.000000e+04  8.000000e+04
## c_2017        9.865421e+04  1.900746e+04  1.783010e+05  9.865421e+04
## eta_limit     1.000000e-01  1.000000e+00  5.000000e-02  1.000000e-01
## linf_al       1.805200e+02  1.805200e+02  1.805200e+02  1.805200e+02
## k_al          1.790600e-01  1.790600e-01  1.790600e-01  1.790600e-01
## t0_al        -6.516000e-01 -6.516000e-01 -6.516000e-01 -6.516000e-01
## a_mat_am      3.500000e+00  3.500000e+00  3.500000e+00  3.500000e+00
## a_old_am      2.050000e+01  2.050000e+01  2.050000e+01  2.050000e+01
## m_juv_am      5.490000e-01  5.490000e-01  5.490000e-01  5.490000e-01
## m_mat_am      6.900000e-02  6.900000e-02  6.900000e-02  6.900000e-02
## m_old_am      4.110000e-01  4.110000e-01  4.110000e-01  4.110000e-01
## a_ls          9.500000e-01  9.500000e-01  9.500000e-01  9.500000e-01
## b_ls          1.988000e+01  1.988000e+01  1.988000e+01  1.988000e+01
## m_ls          1.300000e-01  1.300000e-01  1.300000e-01  1.300000e-01
## a_r           1.313500e+02  1.313500e+02  1.313500e+02  1.313500e+02
## b_r           3.070000e+04  3.070000e+04  3.070000e+04  3.070000e+04
## d_r           1.639951e+00  1.639951e+00  1.639951e+00  1.639951e+00
## t_0           0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## t_i           9.000000e+00  9.000000e+00  9.000000e+00  9.000000e+00
## a_0           5.000000e-01  5.000000e-01  5.000000e-01  5.000000e-01
## a_i           2.650000e+01  2.650000e+01  2.650000e+01  2.650000e+01
## y_arb         0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## a_ma         -5.283123e+00 -8.178585e+00 -2.387660e+00 -5.283123e+00
## b_ma          5.253984e-01  4.544604e-01  5.963363e-01  5.253984e-01
## c_ma          8.299177e+01  4.599855e+01  1.199850e+02  8.299177e+01
## b1_mort_aq    5.936535e+00  6.370672e+00  5.502397e+00  5.936535e+00
## b2_mort_aq   -1.987843e-01 -1.802000e-01 -2.173687e-01 -1.987843e-01
##              pessimistic.1  optimistic.1
## linf_al_aq    2.000000e+02  2.000000e+02
## k_al_aq       1.550000e-01  3.162000e-01
## t0_al_aq     -6.500000e-01 -6.500000e-01
## a_lw          4.128000e-06  4.128000e-06
## b_lw          3.246740e+00  3.246740e+00
## dens_aq       1.000000e+01  3.000000e+01
## cage_size_aq  8.000000e+03  1.100000e+04
## sale_size_aq  2.000000e+00  2.000000e+00
## by1           1.370000e-02  1.610000e-02
## by2           3.100000e-01  3.700000e-01
## f_z           7.000000e-01  7.000000e-01
## g_z           6.500000e+00  6.500000e+00
## h_z           8.000000e-02  1.000000e-02
## j_z           2.000000e+00  1.545000e+00
## k_z           1.700000e+05  1.700000e+05
```

```
## l_z           2.000000e+00  2.000000e+00
## mmin_aq       1.000000e+01  0.000000e+00
## disc_aq       1.000000e+00  7.500000e-01
## loss          5.000000e-02  1.500000e-01
## switch_aq     0.000000e+00  1.000000e+00
## b_2017        2.570900e+04  4.714900e+04
## bprop_2017    6.534667e-01  1.198425e+00
## f_2017        1.400000e+03  1.400000e+03
## fn_2017       8.640000e+05  2.100000e+04
## e_2017        1.000000e+02  2.000000e+02
## b_b           8.000000e+04  8.000000e+04
## c_2017        1.900746e+04  1.783010e+05
## eta_limit     1.000000e+00  5.000000e-02
## linf_al       1.805200e+02  1.805200e+02
## k_al          1.790600e-01  1.790600e-01
## t0_al        -6.516000e-01 -6.516000e-01
## a_mat_am      3.500000e+00  3.500000e+00
## a_old_am      2.050000e+01  2.050000e+01
## m_juv_am      5.490000e-01  5.490000e-01
## m_mat_am      6.900000e-02  6.900000e-02
## m_old_am      4.110000e-01  4.110000e-01
## a_ls          9.500000e-01  9.500000e-01
## b_ls          1.988000e+01  1.988000e+01
## m_ls          1.300000e-01  1.300000e-01
## a_r           1.313500e+02  1.313500e+02
## b_r           3.070000e+04  3.070000e+04
## d_r           1.639951e+00  1.639951e+00
## t_0           0.000000e+00  0.000000e+00
## t_i           9.000000e+00  9.000000e+00
## a_0           5.000000e-01  5.000000e-01
## a_i           2.650000e+01  2.650000e+01
## y_arb         0.000000e+00  0.000000e+00
## a_ma         -8.178585e+00 -2.387660e+00
## b_ma          4.544604e-01  5.963363e-01
## c_ma          4.599855e+01  1.199850e+02
## b1_mort_aq    6.370672e+00  5.502397e+00
## b2_mort_aq   -1.802000e-01 -2.173687e-01
```

```r
# Change parameters in new columns for sensitivity analysis.
pars["switch_aq",1:3] = 1
pars["switch_aq",4:6] = 0

# Turn parameters into a different matrix for analysis of outcomes from different arbitrary annual aqua
pars_arb = pars_full %>%
  select(2, 4) %>%
  column_to_rownames(var = "name_short")

pars_arb["switch_aq", 1] = 0
pars_arb["eta_limit", 1] = 1
pars_arb[2:21] = pars_arb[1]
pars_arb["y_arb", 2:21] = seq(1, 20)
```

4. Hog Function

```
knitr::read_chunk('fun.R')
```

```
  fun = function(par){

    # Name inputs.
    for(i in 1:nrow(par)){assign(rownames(par)[i], par[i,])}
    # Run intermediate set-up.
    #  Fishery.
    #   Numbers in 2017.
    n0 = fi_biom_dat %>%
      mutate(n = 1000 * bprop_2017 * Biomasa / fun_l_w(a_lw, fun_a_l(Edad, linf_al, k_al, t0_al), b_lw
      select(n)
    n0 = n0[[1]]
    #   Catchability.
    # F = qENS > q = F / ENS; N is in numbers, F is in tonnes, and S is in proportions, so conversion
    q = 1000 * f_2017 / sum(n0 * fun_l_w(a_lw, fun_a_l(seq(a_0, a_i), linf_al, k_al, t0_al), b_lw) * 
    #  Aquaculture.
    #   Cohort count at first saleable size is the ratio of density in kgm^-3 to size in kg.
    nsale = fun_ns(cage_size_aq, dens_aq, sale_size_aq)
    #   Initial cohort count is density at first saleable size, plus cumulative mortality at first sa
    #   Casually, nstart = nsale + mort(a(l(wsale))).
    #   Cumulative mortality to first saleable age:
    a_sale = (t0_al_aq - 1 / k_al_aq * (log(1 - ((sale_size_aq / a_lw) ^ (1 / b_lw)) / linf_al_aq)))
    #   Initial stock to reach optimal density at first salable size:
    nstart = nsale * (100 / (100 - fun_a_aqmort(a_sale, b1_mort_aq, b2_mort_aq, mmin_aq))) *
      (100 / (100 - fun_a_aqmort(a_sale, b1_mort_aq, b2_mort_aq, mmin_aq))) *
      (100 / (100 - fun_a_aqmort(a_sale, b1_mort_aq, b2_mort_aq, mmin_aq)))

    # Build objects to fill.
    #  Fishery.
    n = matrix(nrow = t_i - t_0 + 1, ncol = a_i - a_0 + 1) # Build a matrix of numbers at age.
    m = matrix(nrow = t_i - t_0 + 1, ncol = a_i - a_0 + 1) # Build a matrix of natural mortalities at
    b = matrix(nrow = t_i - t_0 + 1, ncol = a_i - a_0 + 1) # Build a matrix of bycatch at age.
    y = matrix(nrow = t_i - t_0 + 1, ncol = a_i - a_0 + 1) # Build a matrix of catch at age.
    p_mat = matrix(nrow = t_i - t_0 + 1, ncol = a_i - a_0 + 1) # Build a matrix of prices at age.
    a_matrix = matrix(nrow = a_i - a_0 + 1, ncol = t_i - t_0 + 1) # Build a matrix of ages for refere
    rec = as.numeric(vector(length = a_i - a_0 + 1))  # Build a vector of recruitment at age.
    e = as.numeric(vector(length = t_i - t_0 + 1))  # Build a vector of effort. This is the variable 
    r_fi = as.numeric(vector(length = t_i - t_0 + 1))  # Build a vector of total revenues.
    c_fi = as.numeric(vector(length = t_i - t_0 + 1))  # Build a vector of total costs.
    #  Aquaculture.
    a0_aq = as.numeric(vector(length = t_i - t_0 + 1))
    a1_aq = as.numeric(vector(length = t_i - t_0 + 1))
    h_aq = as.numeric(vector(length = t_i - t_0 + 1))
    hinv_aq = as.numeric(vector(length = t_i - t_0 + 1))
    l0_aq = as.numeric(vector(length = t_i - t_0 + 1)) #x
    w0_aq = as.numeric(vector(length = t_i - t_0 + 1)) #x
    nm0_aq = as.numeric(vector(length = t_i - t_0 + 1))
    ns0_aq = as.numeric(vector(length = t_i - t_0 + 1))
    nt0_aq = as.numeric(vector(length = t_i - t_0 + 1))
    n0_aq = as.numeric(vector(length = t_i - t_0 + 1))
    rt0_aq = as.numeric(vector(length = t_i - t_0 + 1)) #x
    y0_aq = as.numeric(vector(length = t_i - t_0 + 1)) #x
```

```
p0_aq = as.numeric(vector(length = t_i - t_0 + 1))
rmaw0_aq = as.numeric(vector(length = t_i - t_0 + 1))
rround0_aq = as.numeric(vector(length = t_i - t_0 + 1)) #x
r0_aq = as.numeric(vector(length = t_i - t_0 + 1))
c0_aq = as.numeric(vector(length = t_i - t_0 + 1))
l1_aq = as.numeric(vector(length = t_i - t_0 + 1))
w1_aq = as.numeric(vector(length = t_i - t_0 + 1))
nm1_aq = as.numeric(vector(length = t_i - t_0 + 1))
ns1_aq = as.numeric(vector(length = t_i - t_0 + 1))
nt1_aq = as.numeric(vector(length = t_i - t_0 + 1))
n1_aq = as.numeric(vector(length = t_i - t_0 + 1))
rt1_aq = as.numeric(vector(length = t_i - t_0 + 1))
y1_aq = as.numeric(vector(length = t_i - t_0 + 1))
p1_aq = as.numeric(vector(length = t_i - t_0 + 1))
rmaw1_aq = as.numeric(vector(length = t_i - t_0 + 1))
rround1_aq = as.numeric(vector(length = t_i - t_0 + 1))
r1_aq = as.numeric(vector(length = t_i - t_0 + 1))
c1_aq = as.numeric(vector(length = t_i - t_0 + 1))
r_aq = as.numeric(vector(length = t_i - t_0 + 1))
c_aq = as.numeric(vector(length = t_i - t_0 + 1))

# Add initial values.
#  Fishery.
a_matrix[, 1:(t_i - t_0 + 1)] = seq(a_0, a_i) # Matrix of ages.
a_matrix = t(a_matrix) # Transposing matrix of ages.
n[1,] = n0 # Age distribution for first year, e.g. 2017.
m[1,] = n[1,] * fun_a_nmort(a_matrix[1,], a_mat_am, a_old_am, m_juv_am, m_mat_am, m_old_am) # Nat
b[1,] = (n[1,] - m[1,]) * fun_a_bmort(a_matrix[1,], b_b, a_mat_am, n0) # Bycatch mortalities by c
e[1] = e_2017 # Effort in boats/season for 2017.
y[1,] = (n[1,] - m[1,] - b[1,]) * q * e[1] * fun_l_s(fun_a_l(a_matrix[1,], linf_al, k_al, t0_al),
p_mat[1,] = fun_p(sum(fun_l_w(a_lw, fun_a_l(a_matrix[1, ], linf_al, k_al, t0_al), b_lw) * y[1, ] *
                  fun_l_w(a_lw, fun_a_l(a_matrix[1, ], linf_al, k_al, t0_al), b_lw) * by1 * by2 *
                  a_ma, b_ma, c_ma) * loss
r_fi[1] = sum(p_mat[1,] * fun_l_w(a_lw, fun_a_l(a_matrix[1, ], linf_al, k_al, t0_al), b_lw) * y[1
c_fi[1] = e[1] * c_2017 # Costs for first year.
rec[1] = fun_rec(sum(n[1, 2:(t_i - t_0 + 1)]), a_r, b_r, d_r) # Recruitment for first year.
eta = (e[1] * eta_limit) / (r_fi[1] - c_fi[1]) # Parameter to restrict changes in effort.

#  Aquaculture.
#   Current.
a0_aq[1] = a_0
l0_aq[1] = fun_a_l(a0_aq[1], linf_al_aq, k_al_aq, t0_al_aq)
w0_aq[1] = fun_l_w(a_lw, l0_aq[1], b_lw)
nm0_aq[1] = nstart * (0.01 * fun_a_aqmort(a0_aq[1], b1_mort_aq, b2_mort_aq, mmin_aq))
ns0_aq[1] = nstart * (1 - 0.01 * fun_a_aqmort(a0_aq[1], b1_mort_aq, b2_mort_aq, mmin_aq)) # Note
nt0_aq[1] = ifelse(ns0_aq[1] - fun_ns(cage_size_aq, dens_aq, w0_aq[1]) > 0, ns0_aq[1] - fun_ns(ca
n0_aq[1] = nstart - nm0_aq[1] - nt0_aq[1]
rt0_aq[1] = nt0_aq[1] * w0_aq[1] * f_z * g_z # Fix placeholder variable names.
y0_aq[1] = w0_aq[1] * n0_aq[1] * by1 * by2
p0_aq[1] = p_mat[1, round(a0_aq[1] + 0.5)] * 1000 # Conversion for price in grams to revenue from
rmaw0_aq[1] = y0_aq[1] * n0_aq[1] * p0_aq[1]
rround0_aq[1] = w0_aq[1] * f_z * g_z # Fix placeholder variable names.
r0_aq[1] = (rmaw0_aq[1] + rround0_aq[1])
```

```
c0_aq[1] = n0_aq[1] * w0_aq[1] * h_z * j_z * 365 + k_z # Fix placeholder variable names.

#   Led.
a1_aq[1] = a0_aq[1] + 1
l1_aq[1] = fun_a_l(a1_aq[1], linf_al_aq, k_al_aq, t0_al_aq)
w1_aq[1] = fun_l_w(a_lw, l1_aq[1], b_lw)
# Since this implementation of mortality/survival and trimming require iteration to work, the cor
nm1_aq[1] = (nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_aq)) -
            ifelse(nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_a
                   nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_a
                   0)) * (0.01 * fun_a_aqmort(a1_aq[1], b1_mort_aq, b2_mort_aq, mmin_aq))
ns1_aq[1] = (nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_aq)) -
            ifelse(nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_a
                   nstart * (1 - 0.01 * fun_a_aqmort(a1_aq[1] - 1, b1_mort_aq, b2_mort_aq, mmin_a
                   0)) * (1 - 0.01 * fun_a_aqmort(a1_aq[1], b1_mort_aq, b2_mort_aq, mmin_aq))
nt1_aq[1] = ifelse(ns1_aq[1] - fun_ns(cage_size_aq, dens_aq, w1_aq[1]), ns1_aq[1] - fun_ns(cage_s
n1_aq[1] = fun_ns(cage_size_aq, dens_aq, w1_aq[1])
rt1_aq[1] = nt1_aq[1] * w1_aq[1] * f_z * g_z # Fix placeholder variable names.
y1_aq[1] = w1_aq[1] * n1_aq[1] * by1 * by2 # Fix placeholder variable names.
p1_aq[1] = p_mat[1, round(a1_aq[1] + 0.5)] * 1000 # Conversion for price in grams to revenue from
rmaw1_aq[1] = y1_aq[1] * p1_aq[1]
rround1_aq[1] = w1_aq[1] * f_z * g_z
r1_aq[1] = (rmaw1_aq[1] + rround1_aq[1])
c1_aq[1] = n1_aq[1] * w1_aq[1] * h_z * j_z * 365 + k_z # Fix placeholder variable names.

h_aq[1] = 0
hinv_aq[1] = 1
r_aq[1] = r0_aq[1] * h_aq[1]# + rt0_aq[1] * hinv_aq[1]
c_aq[1] = c0_aq[1] * hinv_aq[1]# + l_z * nstart * h_aq[1]

# Add iterations.
for(i in 2:(t_i - t_0 + 1)){
  for(j in 2:(a_i - a_0 + 1)){
    # Fishery.
    #  Numbers for time i and cohort j are numbers of the previous time and cohort less mortaliti
    n[i, j] = ifelse(n[i - 1, j - 1] - m[i - 1, j - 1] - b[i - 1, j - 1] - y[i - 1, j - 1] > 0,
                     n[i - 1, j - 1] - m[i - 1, j - 1] - b[i - 1, j - 1] - y[i - 1, j - 1],
                     0)

    #  Natural mortalities for time i and cohort j are numbers for the same multipled by a consta
    m[i, j] = n[i, j] * fun_a_nmort(a_matrix[i, j], a_mat_am, a_old_am, m_juv_am, m_mat_am, m_old

    #  Bycatch for time i and cohort j are numbers for the same after natural mortality multipled
    b[i, j] = (n[i, j] - m[i, j]) * fun_a_bmort(a_matrix[i, j], b_b, a_mat_am, n0)

  }

  #  Numbers for time i and first cohort.
  n[i, 1] = rec[i - 1]

  #  Natural mortalities for time i and first cohort.
  m[i, 1] = n[i, 1] * fun_a_nmort(a_matrix[i, 1], a_mat_am, a_old_am, m_juv_am, m_mat_am, m_old_a
```

```
#  Bycatch mortality for time i and first cohort.
b[i, 1] = (n[i, 1] - m[i, 1]) * fun_a_bmort(a_matrix[i, 1], b_b, a_mat_am, n0)

# Effort for time i and all cohorts from past effort, revenues, costs, and a stiffness paramete
e[i] = e[i - 1] + eta * (r_fi[i - 1] - c_fi[i - 1])

#  Catches from effort and the rest.
for(j in 2:(a_i - a_0 + 1)){
  y[i, j] = ifelse((n[i, j] - m[i, j] - b[i, j]) * q * e[i] * fun_l_s(fun_a_l(a_matrix[i, j], li
                   (n[i, j] - m[i, j] - b[i, j]) * q * e[i] * fun_l_s(fun_a_l(a_matrix[i, j], li
                   0)
  y[i, 1] = ifelse((n[i, 1] - m[i, 1] - b[i, 1]) * q * e[i] * fun_l_s(fun_a_l(a_matrix[i, 1], li
                   (n[i, 1] - m[i, 1] - b[i, 1]) * q * e[i] * fun_l_s(fun_a_l(a_matrix[i, 1], li
                   0)
}

#  Recruitment for time i.
rec[i] = fun_rec(sum(n[i, 2:(a_i - a_0 + 1)]), a_r, b_r, d_r)

# Aquaculture.
a0_aq[i] = a0_aq[i - 1] * hinv_aq[i - 1] + 1
a1_aq[i] = a0_aq[i] + 1

l0_aq[i] = fun_a_l(a0_aq[i], linf_al_aq, k_al_aq, t0_al_aq)
w0_aq[i] = fun_l_w(a_lw, l0_aq[i], b_lw)
nm0_aq[i] = (nstart * h_aq[i - 1] + n0_aq[i - 1] * hinv_aq[i - 1]) * (0.01 * fun_a_aqmort(a0_aq
ns0_aq[i] = (nstart * h_aq[i - 1] + n0_aq[i - 1] * hinv_aq[i - 1]) * (1 - 0.01 * fun_a_aqmort(a0
nt0_aq[i] = ifelse(ns0_aq[i] - fun_ns(cage_size_aq, dens_aq, w0_aq[i]) > 0, ns0_aq[i] - fun_ns(c
n0_aq[i] = (nstart * h_aq[i - 1] + n0_aq[i - 1] * hinv_aq[i - 1]) - nm0_aq[i] - nt0_aq[i]
rt0_aq[i] = nt0_aq[i] * w0_aq[i] * f_z * g_z # Fix placeholder variable names.
y0_aq[i] = w0_aq[i] * n0_aq[i] * by1 * by2
p0_aq[i] = p_mat[i - 1, a0_aq[i]] * 1000 # Conversion for price in grams to revenue from kilogr
rmaw0_aq[i] = y0_aq[i] * p0_aq[i]
rround0_aq[i] = w0_aq[i] * f_z * g_z # Fix placeholder variable names.
r0_aq[i] = (rmaw0_aq[i] + rround0_aq[i])
c0_aq[i] = n0_aq[1] * w0_aq[1] * h_z * j_z * 365 + k_z # Fix placeholder variable names.

l1_aq[i] = fun_a_l(a1_aq[i], linf_al_aq, k_al_aq, t0_al_aq)
w1_aq[i] = fun_l_w(a_lw, l1_aq[i], b_lw)
nm1_aq[i] = (nstart * h_aq[i - 1] + n1_aq[i - 1] * hinv_aq[i - 1]) * (0.01 * fun_a_aqmort(a1_aq
ns1_aq[i] = (nstart * h_aq[i - 1] + n1_aq[i - 1] * hinv_aq[i - 1]) * (1 - 0.01 * fun_a_aqmort(a1
nt1_aq[i] = ifelse(ns1_aq[i] - fun_ns(cage_size_aq, dens_aq, w1_aq[i]) > 0, ns1_aq[i] - fun_ns(c
n1_aq[i] = (nstart * h_aq[i - 1] + n1_aq[i - 1] * hinv_aq[i - 1]) - nm1_aq[i] - nt1_aq[i]
rt1_aq[i] = nt1_aq[i] * w1_aq[i] * f_z * g_z # Fix placeholder variable names.
y1_aq[i] = w1_aq[i] * n1_aq[i] * by1 * by2
p1_aq[i] = p_mat[i - 1, a1_aq[i]] * 1000 # Conversion for price in grams to revenue from kilogr
rmaw1_aq[i] = y1_aq[i] * p1_aq[i]
rround1_aq[i] = w1_aq[i] * f_z * g_z # Fix placeholder variable names.
r1_aq[i] = (rmaw1_aq[i] + rround1_aq[i])
c1_aq[i] = n1_aq[1] * w1_aq[1] * h_z * j_z * 365 + k_z # Fix placeholder variable names.

# Mind wrapper for hard-coding lower bound age at harvest.
h_aq[i] = ifelse(a0_aq[i] > 3, ifelse((r0_aq[i] - l_z * nstart) > (disc_aq * (r1_aq[i] - c0_aq[i
```

```
    hinv_aq[i] = (h_aq[i] - 1) ^ 2

    r_aq[i] = (r0_aq[i] + rt0_aq[i] * hinv_aq[i])
    c_aq[i] = c0_aq[i] * hinv_aq[i] + l_z * nstart * h_aq[i]

    # Prices in matrix. Use this one. Think harder about the lag problem.
    for(j in 1:(a_i - a_0 + 1)){
      p_mat[i, j] = fun_p(sum(
                            fun_l_w(a_lw, fun_a_l(a_matrix[i, ], linf_al, k_al, t0_al), b_lw) * y
                            switch_aq * (y0_aq[i] * by1 * by2 * h_aq[i] + nt0_aq[i] * w0_aq[i] * 
                            )
                            / 1000 + y_arb, # Conversion to tonnes and addition of arbitrary prod
                        fun_l_w(a_lw, fun_a_l(a_matrix[i, j], linf_al, k_al, t0_al), b_lw) * by1 
                        a_ma, b_ma, c_ma) * loss
    }

    # Revenues.
    r_fi[i] = sum(p_mat[i,] * fun_l_w(a_lw, fun_a_l(a_matrix[i, ], linf_al, k_al, t0_al), b_lw) * y

    # Costs.
    c_fi[i] = e[i] * c_2017
  }

  # Tidy results: numbers, recruitment, catches, effort, revenues, costs, profits.
  #  Numbers.
  tidyn = melt(n)
  tidyn$var = "Numbers"
  #  Catches.
  tidyy = melt(y)
  tidyy$var = "Catches"
  #  Poaching Effort.
  tidye = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, valu
  tidye$Var1 = seq(1, t_i - t_0 + 1)
  tidye$Var2 = NA
  tidye$value = e
  tidye$var = "Effort"
  #  Poaching Revenue.
  tidyr_fi = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, va
  tidyr_fi$Var1 = seq(1, t_i - t_0 + 1)
  tidyr_fi$Var2 = NA
  tidyr_fi$value = r_fi
  tidyr_fi$var = "Poaching Revenue"
  #  Poaching Cost.
  tidyc_fi = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, va
  tidyc_fi$Var1 = seq(1, t_i - t_0 + 1)
  tidyc_fi$Var2 = NA
  tidyc_fi$value = c_fi
  tidyc_fi$var = "Poaching Cost"
  #  Poaching Profit.
  tidypi_fi = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, 
  tidypi_fi$Var1 = seq(1, t_i - t_0 + 1)
  tidypi_fi$Var2 = NA
  tidypi_fi$value = r_fi - c_fi
```

```
      tidypi_fi$var = "Poaching Profit"
      #  Aquaculture Revenue.
      tidyr_aq = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, va
      tidyr_aq$Var1 = seq(1, t_i - t_0 + 1)
      tidyr_aq$Var2 = NA
      tidyr_aq$value = r_aq
      tidyr_aq$var = "Aquaculture Revenue"
      #  Aquaculture Cost.
      tidyc_aq = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, va
      tidyc_aq$Var1 = seq(1, t_i - t_0 + 1)
      tidyc_aq$Var2 = NA
      tidyc_aq$value = c_fi
      tidyc_aq$var = "Aquaculture Cost"
      # Aquaculture Profit.
      tidypi_aq = rename(data.frame(matrix(NA, nrow = t_i - t_0 + 1, ncol = 4)), Var1 = X1, Var2 = X2, v
      tidypi_aq$Var1 = seq(1, t_i - t_0 + 1)
      tidypi_aq$Var2 = NA
      tidypi_aq$value = r_aq - c_aq
      tidypi_aq$var = "Aquaculture Profit"
      #  Everything!
      tidy = bind_rows(tidyn, tidyy, tidye, tidyr_fi, tidyc_fi, tidypi_fi, tidyr_aq, tidyc_aq, tidypi_aq
      tidy$group = ifelse(tidy$Var2 < a_mat_am, "Machorro", ifelse(tidy$Var2 < a_old_am, "Pre-Adulto",
      tidy = rename(tidy, Year = Var1, Age = Var2, Result = value, Variable = var, Group = group)

      # Get results.
      return(tidy)

    }
```

5. Hog Runs

```
# Build a home for results of runs.
results = list()

# Loop through parameter sets.
for(i in 1:6){par = select(pars, i)
              output = fun(par)
              output$Run = i
              output$Scenario = ifelse(output$Run < 4, "w/ Aquaculture", "w/o Aquaculture")
              output$Estimate = ifelse(output$Run == 1 | output$Run == 4, "Central", "Outer")
              results[[i]] = output}
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
# Go from list to dataframe for easier processing.
results = bind_rows(results)

# Build a home for results of runs with effort.
results_e = list()

# Loop through parameter sets for effort and price reduction.
for(i in 1:length(pars_arb)){par = select(pars_arb, i)
```

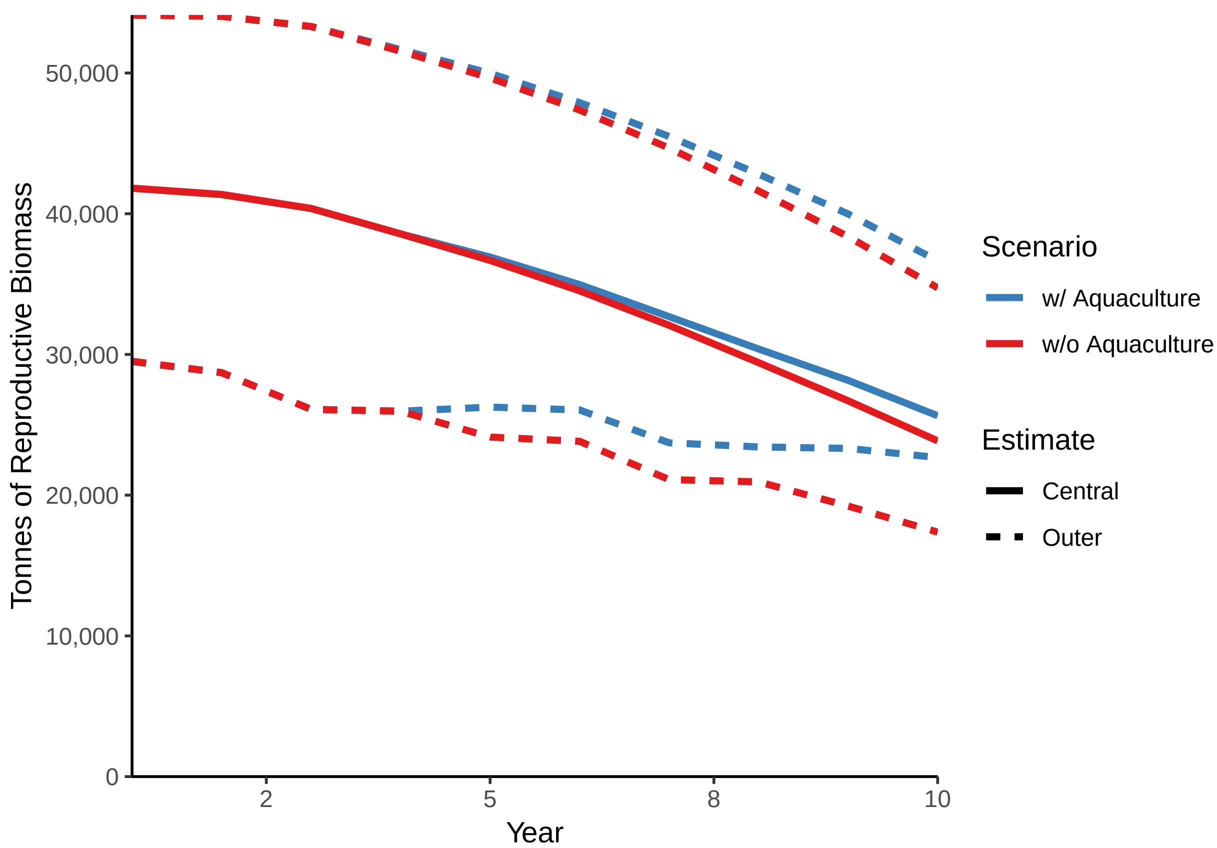
```
              output = fun(par)
              output$Run = i
              results_e[[i]] = output}

# Go from list to dataframe for easier processing.
results_e = bind_rows(results_e)
```

6. Hog Outputs

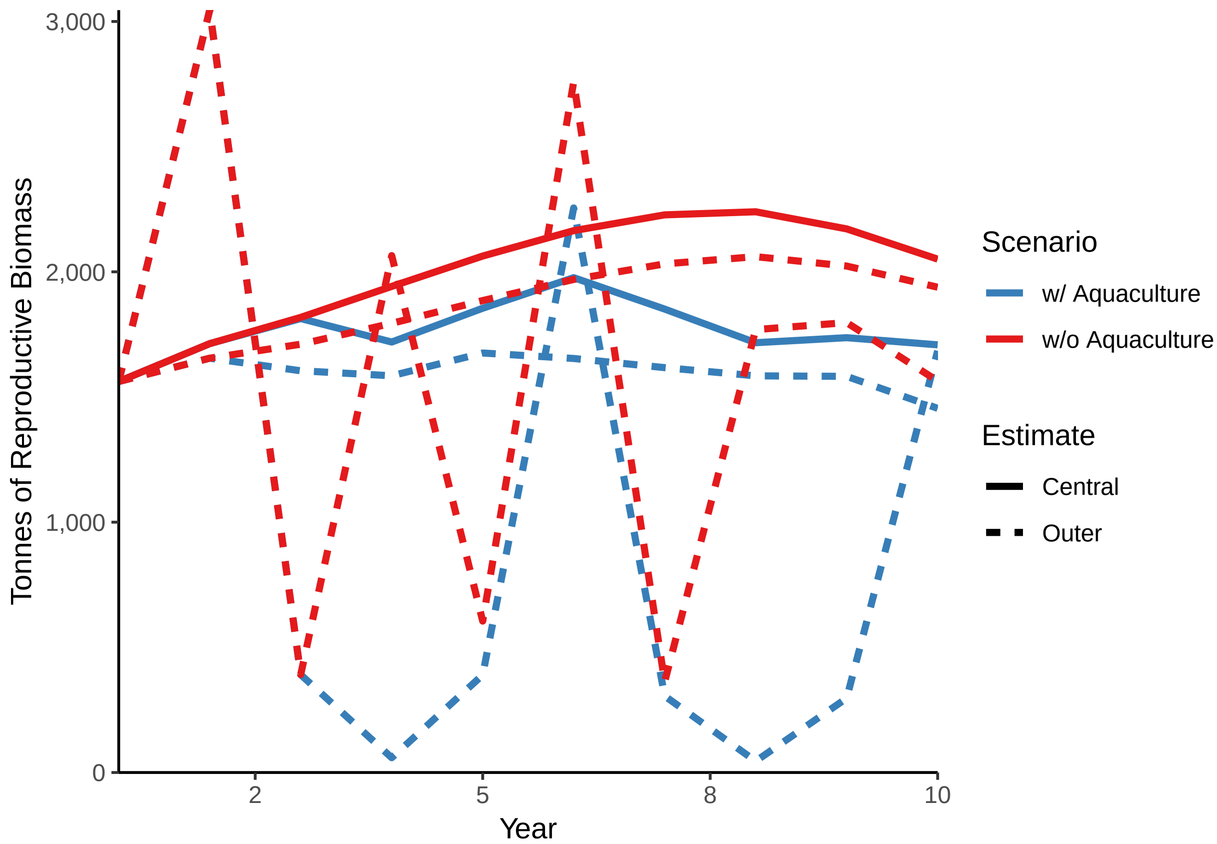
Visualization and tables.

```
knitr::read_chunk('vis.R')
```

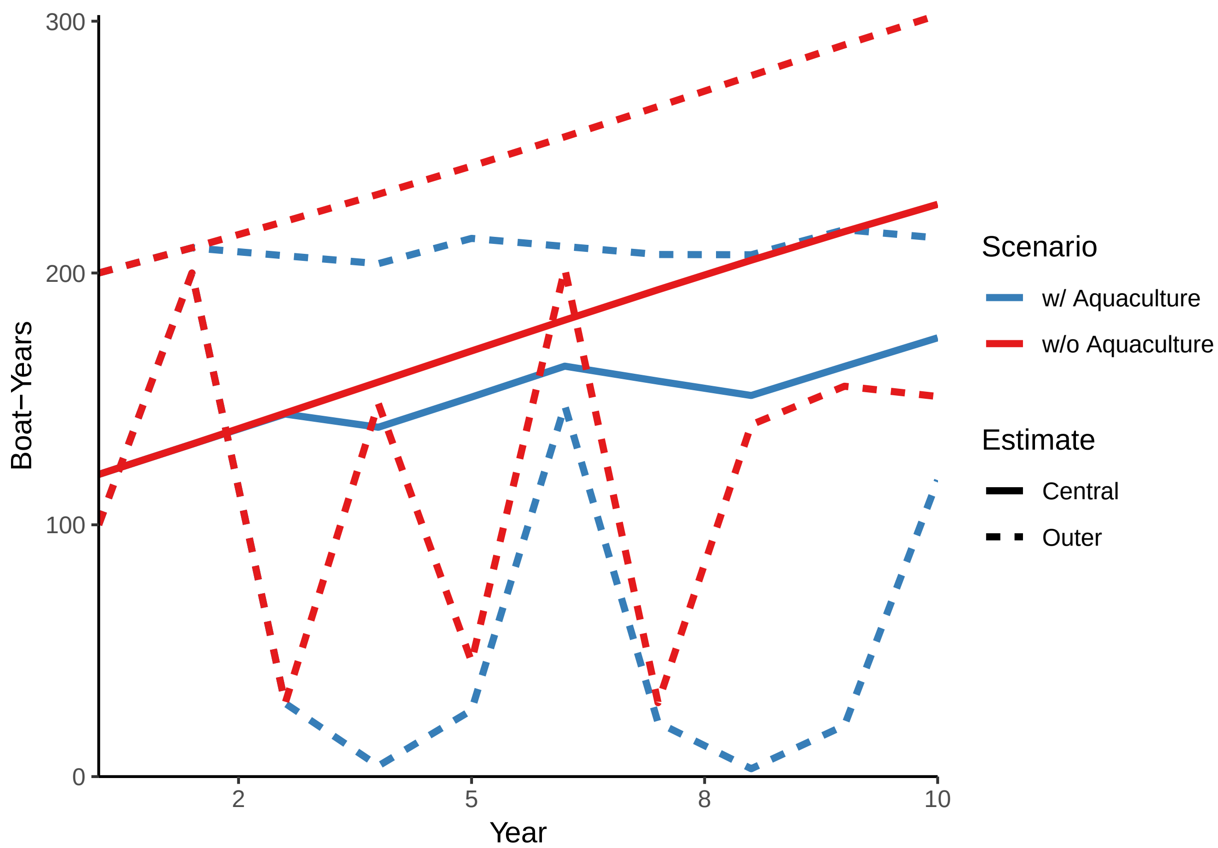
```
# Stock and catch of reproductive biomass in numbers and mass.
results_sum = filter(results, Age > 3) %>%
  mutate(Biomass = fun_l_w(pars$default[4], fun_a_l(Age - 0.5, pars$default[1], pars$default[2], pars$de
  group_by(Year, Variable, Run, Scenario, Estimate) %>% # Run,
  summarize(SumNum = sum(Result), SumBio = sum(Biomass)) %>%
  mutate(LogNum = log(SumNum + 1), LogBio = log(SumBio + 1)) %>%
  ungroup() %>%
  mutate(Run = as.factor(Run)) %>%
  unite("Estimate | Scenario", Estimate, Scenario, sep = " | ", remove = FALSE)

# Plot the summary numbers.
plot_nfig =
  ggplot(filter(results_sum, Variable == "Numbers")) +
  geom_line(aes(x = Year, y = SumBio, group = Run, color = Scenario, linetype = Estimate), size = 1.25)
  scale_color_brewer(palette = "Set1", direction = -1) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, NA), labels = scales::comma) +
  scale_x_continuous(expand = c(0, 0), labels = scales::comma) +
  labs(x = "Year", y = "Tonnes of Reproductive Biomass") +
  theme_classic()

print(plot_nfig)
```



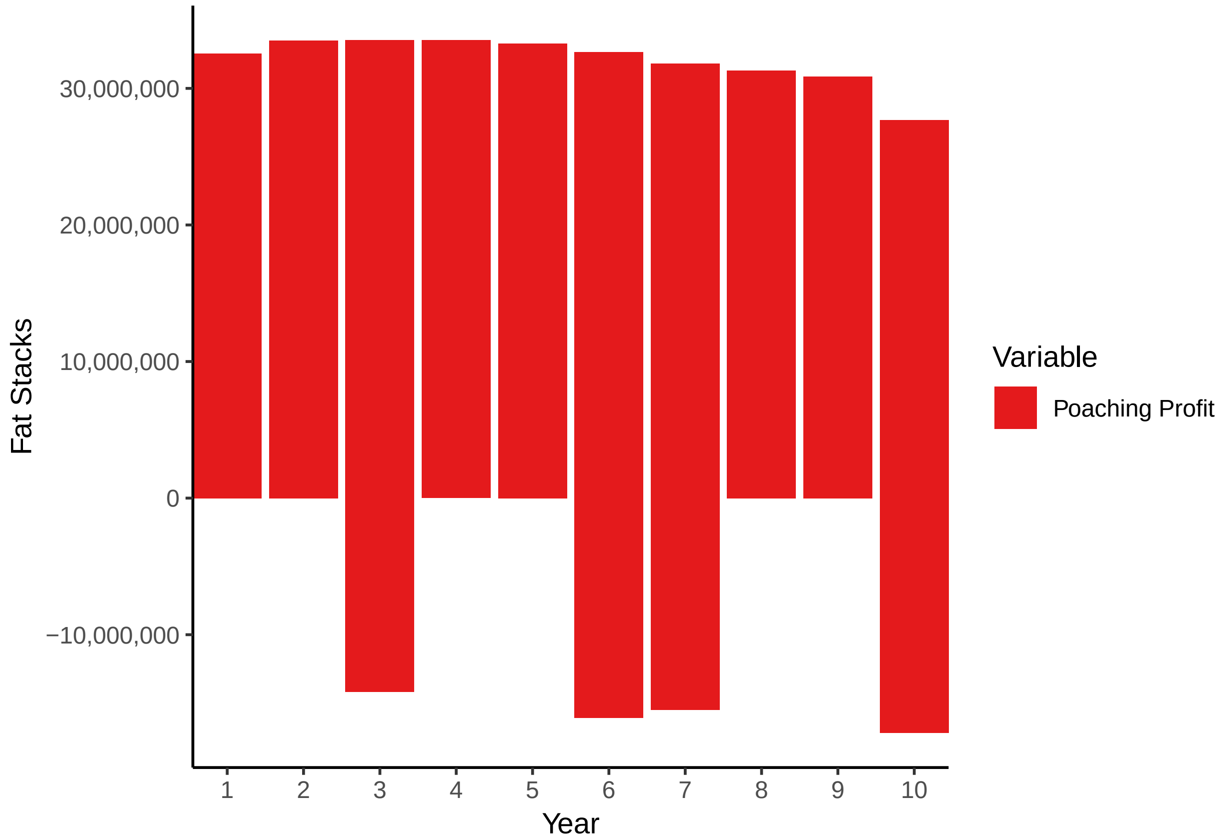

```r
#ggsave("plot_fig.png", plot_fig, dpi = 300, width = 6.5, height = 6.5)

plot_cfig =
  ggplot(filter(results_sum, Variable == "Catches")) +
  geom_line(aes(x = Year, y = SumBio, group = Run, color = Scenario, linetype = Estimate), size = 1.25)
  scale_color_brewer(palette = "Set1", direction = -1) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, NA), labels = scales::comma) +
  scale_x_continuous(expand = c(0, 0), labels = scales::comma) +
  labs(x = "Year", y = "Tonnes of Reproductive Biomass") +
  theme_classic()

print(plot_cfig)
```



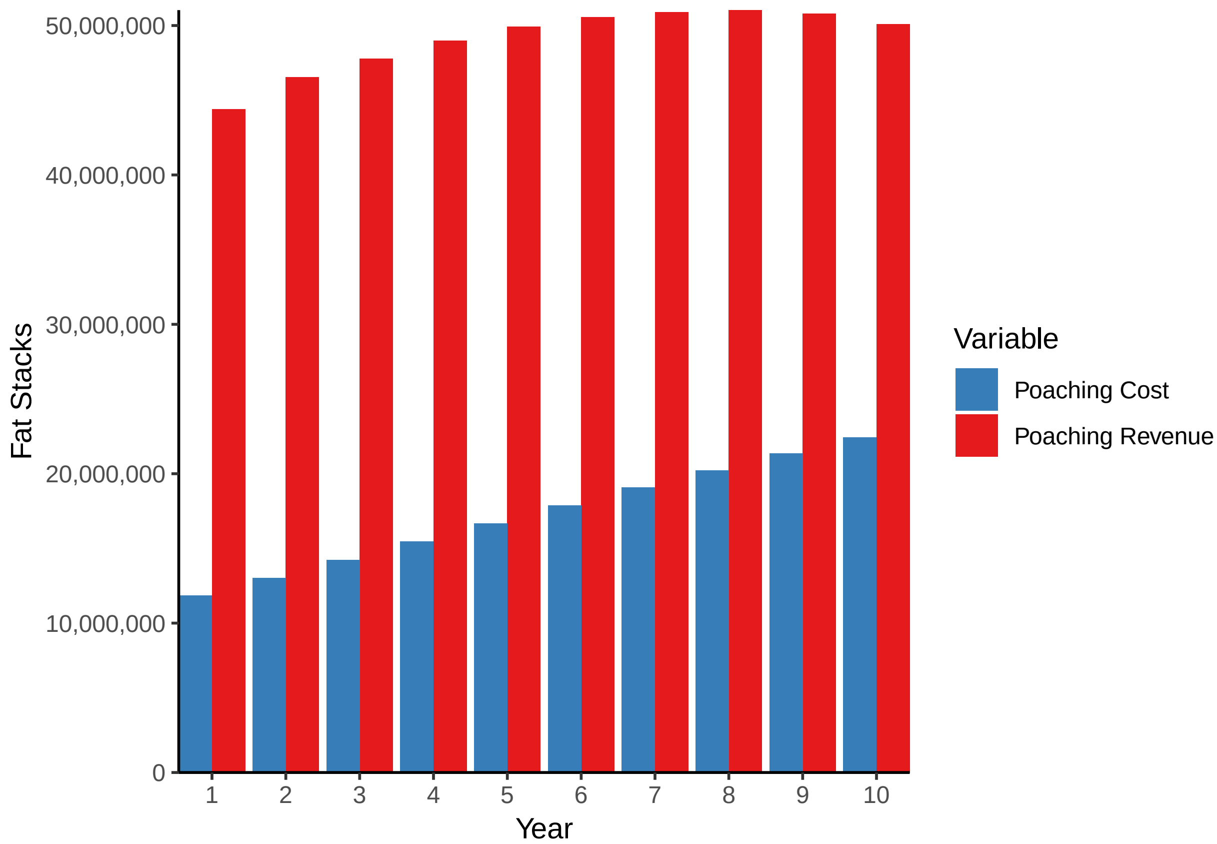

```
#ggsave("plot_cfig.png", plot_fig, dpi = 300, width = 6.5, height = 6.5)

# Plot recruitment.
#plot_recfig =
#  ggplot(filter(results, Variable == "Recruitment")) +
#  geom_line(aes(x = Year, y = Result, group = Run, color = Scenario, linetype = Estimate), size = 1.25
#  scale_color_brewer(palette = "Set1", direction = -1) +
#  scale_y_continuous(expand = c(0, 0), limits = c(0, NA), labels = scales::comma) +
#  scale_x_continuous(expand = c(0, 0), labels = scales::comma) +
#  labs(x = "Year", y = "Juveniles") +
#  theme_classic()

#print(plot_recfig)

# Plot effort.
plot_efig =
  ggplot(filter(results, Variable == "Effort")) +
  geom_line(aes(x = Year, y = Result, group = Run, color = Scenario, linetype = Estimate), size = 1.25)
  scale_color_brewer(palette = "Set1", direction = -1) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, NA), labels = scales::comma) +
  scale_x_continuous(expand = c(0, 0), labels = scales::comma) +
  labs(x = "Year", y = "Boat-Years") +
  theme_classic()

print(plot_efig)
```




```r
#ggsave("plot_cfig.png", plot_fig, dpi = 300, width = 6.5, height = 6.5)

# Data wrangling for a cleaner dataframe.
results_sum_pi = results %>%
  filter(Estimate == "Central" & Variable == "Poaching Revenue" |
           Estimate == "Central" & Variable == "Poaching Cost" |
           Estimate == "Central" & Variable == "Poaching Profit")

# Profits for in both scenarios, sort of.
plot_pi =
  ggplot(filter(results_sum_pi, Variable == "Poaching Profit")) +
  geom_col(aes(x = Year, y = Result, fill = Variable), position = "dodge") +
  scale_fill_brewer(palette = "Set1", direction = -1) +
  scale_y_continuous(labels = scales::comma) +
  scale_x_continuous(expand = c(0, 0), breaks = seq(1, 10), labels = scales::comma) +
  labs(x = "Year", y = "Fat Stacks") +
  theme_classic()

print(plot_pi)
```

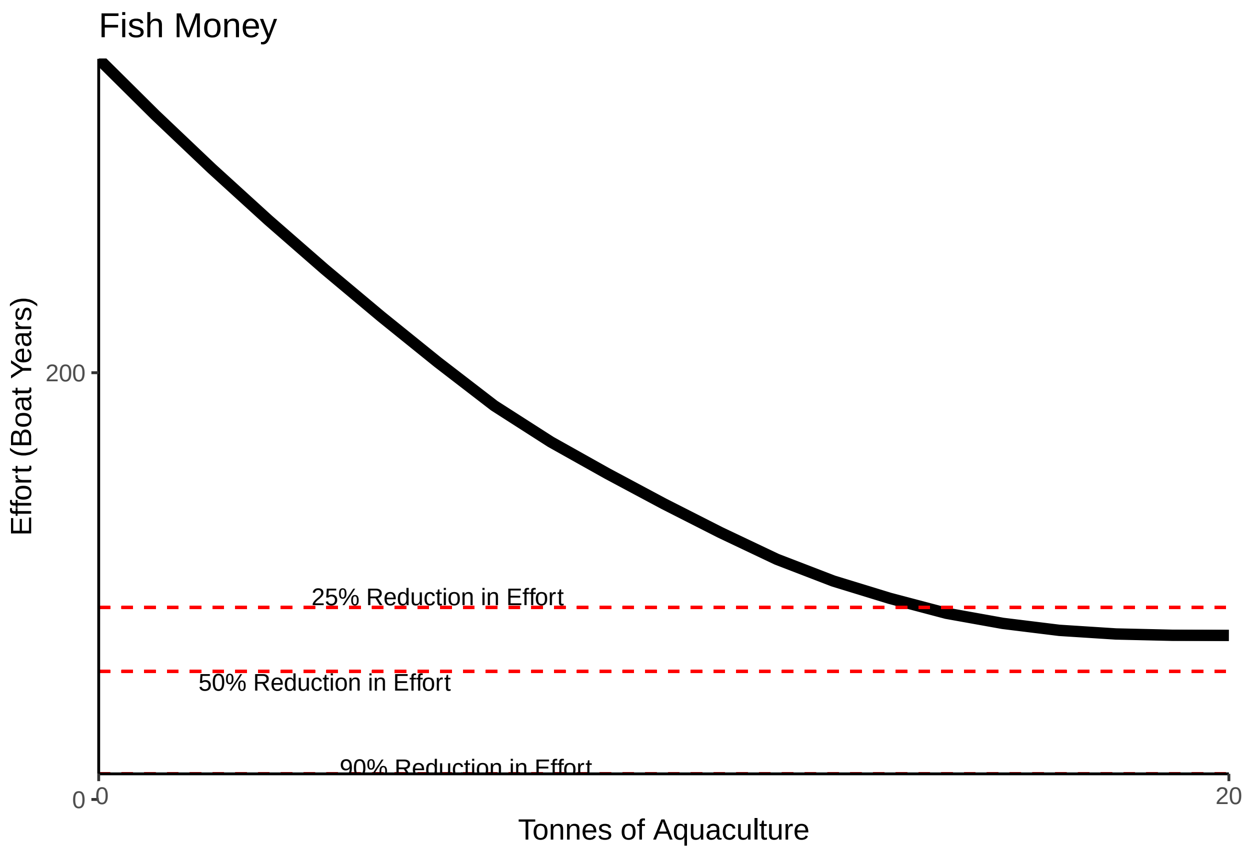



```r
#ggsave("plot_pi.png", plot_pi, dpi = 300, width = 6.5, height = 6.5)

# Revenues and costs for the central scenario. Stacked bar!
plot_rc =
  ggplot(filter(results_sum_pi, Variable == "Poaching Revenue" | Variable == "Poaching Cost")) +
  geom_col(aes(x = Year, y = Result, fill = Variable), position = "dodge") +
  scale_fill_brewer(palette = "Set1", direction = -1) +
  scale_y_continuous(expand = c(0, 0), labels = scales::comma) +
  scale_x_continuous(expand = c(0, 0), breaks = seq(1, 10), labels = scales::comma) +
  labs(x = "Year", y = "Fat Stacks") +
  theme_classic()

print(plot_rc)
```




```
#ggsave("plot_rc.png", plot_rc, dpi = 300, width = 6.5, height = 6.5)

# Wrangle effort.
results_e_sum = filter(results_e, Variable == "Effort") %>%
  group_by(Run) %>%
  summarize(mean = mean(Result)) %>%
  ungroup() %>%
  mutate("Annual Tonnes" = Run - 1) %>%
  rename("Mean Effort" = mean)

plot_earb =
ggplot(results_e_sum, aes(`Annual Tonnes`, `Mean Effort`)) +
  geom_line(color = "black", size = 2) +
  geom_segment(aes( x = 0, xend= Inf, y = 60, yend = 60), linetype = "dashed", color = "red")+
  geom_segment(aes( x = 0, xend= Inf, y = 12, yend = 12), linetype = "dashed", color = "red")+
  geom_segment(aes( x = 0, xend= Inf, y = 90, yend = 90), linetype = "dashed", color = "red")+
  annotate("text", x = 6, y = 95, label = "25% Reduction in Effort", size = 3) + #, family = "Century G
  annotate("text", x = 4, y = 55, label = "50% Reduction in Effort", size = 3) + #, family = "Century G
  annotate("text", x = 6.5, y = 15, label = "90% Reduction in Effort", size = 3) + #, family = "Century
  scale_x_discrete(expand = c(0,0), limits = c(0, 20)) +
  scale_y_discrete(expand = c(0,0), limits = c(0, 200)) +
  labs( x = "Tonnes of Aquaculture", y = "Effort (Boat Years)")+
  ggtitle("Fish Money")+
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))
```

```
plot_earb
```



Appendix: Equations and Variable, Parameter Definitions

Put these in the earlier sections!

Ages to Lengths | Edad y Tamaño

$L(t) = L_\infty(1 - e^{-K(t-t_0)})$

Variable	Definition	Value
L	Length (Centimeters)	
L_∞	Maximum Length (Centimeters)	180.0000
K_{fi}	Catabolic Constant (INAPESCA)	0.155000
K_{aq}	Catabolic Constant (EOF)	0.228560
t	Age (Years)	
t_0	Age, $L = 0$	0.000000

Lengths to Weights | Tamaño y Peso

$W = aL^b$

Variable	Definition	Value
W	Weight (Kilograms)	
L	Length (Centimeters)	
a	Parameter	0.000004128
b	Parameter	3.246740000

Ages to Marginal Natural Mortalities | Edad y Mortalidad Natural Marginal

Marginal Mortality at Age: $N_a = N_{a-1}e^{-M_a}$

Ages	$-M_a$
0 - 2.5	0.549
3.5 - 19.5	0.069
20.5 - 26.5	0.411

Ages to Bycatch Mortalities

$N_{2.5,t} = N_{1.5,t} * b$

$b = 0.20$

Ages to Poaching Mortalities (Selectivities)

$s = \frac{a}{e^{b-m*l}}$

Variable	Definition	Value
a	Shape Parameter	0.95
b	Shape Parameter	19.88
m	Shape Parameter	0.13
l	Length (Centimeters)	

Numbers at Age to Recruitment | Números por Edad y Reclutamiento

$n_{0,t} = \frac{\alpha N_{a,t-1}}{[1+(N_{a,t-1}/\beta)]^{\delta}}$

Variable	Definition	Value
$N_{0,t}$	Recruits	
$N_{a,t}$	Cohort of Age a	
α	Parameter	131.35
β	Parameter	3.07
δ	Parameter	1.639951478

Inverse Demand Specification

$p_{g,t} = \beta_0 + \beta_1 Y_t + g_a^{\beta_2}$

Variable	Definition
$p_{g,t}$	USD2018 / Gram of Dried Buche

Variable	Definition
Y_t	Tonnes of Dried Buche (Total Production)
g_a	Grams of Dried Buche (Individual Product)
β_0	Coefficient (Choke Price)
β_1	Coefficient (Quantity Elasticity)
β_2	Coefficient (Exponential Price Premium)

Effort

$E_t = E_{t-1} + \eta\pi_{t-1}$

Variable	Definition
E_t	Effort in Boat-Years
η	Resistance Parameter (i.e. limit to entry, exit)
π	Profit (USD2018)

Catch

$y_{a,t} = qE_t s_a n_{a,t}$

Variable	Definition
$y_{a,t}$	Numbers Caught by Cohort and Time
q	Catchability
E_t	Effort in Boat-Years
s_a	Proportional Selectivity
$n_{a,t}$	Numbers by Cohort and Time

Stock

$N_t = \sum(n_{a,t-1} - n_{a,t-1}m_{a,t-1} - n_{a,t-1}m_{a,t-1}b_{a,t-1} - y_{a,t-1}) + n_{0,t-1}$