EASI-Fish vignette

Introduction

Here, we provide example code and application of the Ecological Assessment of the Sustainable Impacts of Fisheries (EASI-Fish) ecological risk assessment approach as detailed in Hill et al. (in prep). We apply EASI-Fish to two example species, Alopias vulpinus (ALV) and Carcharhinus longimanus (OCS) and two fisheries to show how this approach can be applied to multiple species and fisheries simultaneously.

Preparing life history data

We read in a csv document listing the various life history parameters for two species and explore these to make sure they look reasonable. This csv has the structure of a param column that names all the parameters you can input, and then each of the following columns represents a species information. For parameters that can be varied, there are rows to input standard deviation, standard error, and distribution type (e.g. 'fixed', 'normal').

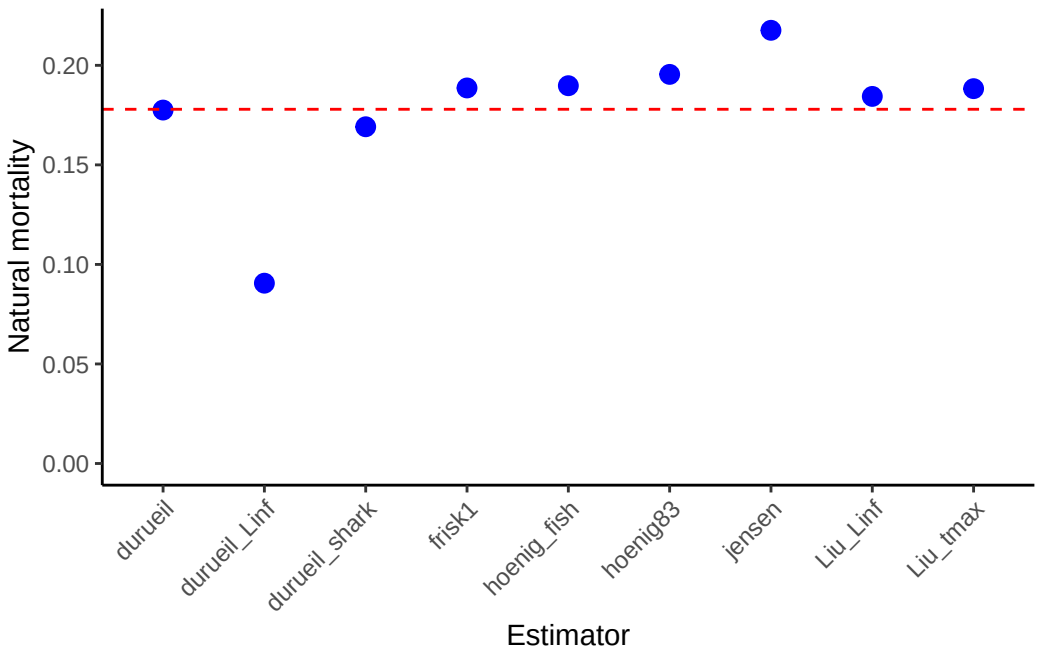
Note that you need the 'caret' and 'lattice' packages as well as the 'metadata234.Rdata' file to run the Liu natural mortality estimators. If not, you do not need these features.

```
# load in raw life history csv
lh <- read.csv('data/lh_inputs.csv')
head(lh)
```

```
         param     ALV  OCS
1         nsim      10   10
2      len_int       4    4
3  length_type      UF   TL
4 growth_model Schnute VBL0
5   max_length     292  342
6      max_age      22   25
```

Now, we rearrange this data and estimate natural mortality taking the mean of three different estimators. A function is provided herein to explore a range of commonly available natural mortality functions.

```
library(caret)
library(lattice)
lh_df <- lh |>
  pivot_longer(cols = -param, names_to = "spp",values_to = "value") |>
  pivot_wider(names_from = param ,values_from = value) |>
  type.convert(as.is=T) |>
  rowwise() |>
  mutate(M_normal = easiM_estimate(funcs = c('hoenig_fish', 'durueil_shark', 'Liu_tmax'),
         nsim = nsims)

# Explore all M estimates for ALV and plot
easiM_estimator_all(funcs = c("hoenig83", "Liu_tmax", "Liu_Linf",
                              "jensen", "pauly_sst","pauly_kt", "durueil",
                              "durueil_shark", "hoenig_fish", "durueil_Linf",
                              "frisk1","Then_tmax", "Then_Linf"),
                    CV = 0.1, tmax = 22, K = 0.136, L0 = 81, Linf = 245.8,
                    plot = T)
```



```
       func_name          M
1       hoenig83 0.19545455
2       Liu_tmax 0.18829228
3       Liu_Linf 0.18439000
4         jensen 0.21760000
5        durueil 0.17753348
6  durueil_shark 0.16915105
7    hoenig_fish 0.18976802
8   durueil_Linf 0.09056963
9         frisk1 0.18863485
10          mean 0.17793265
```

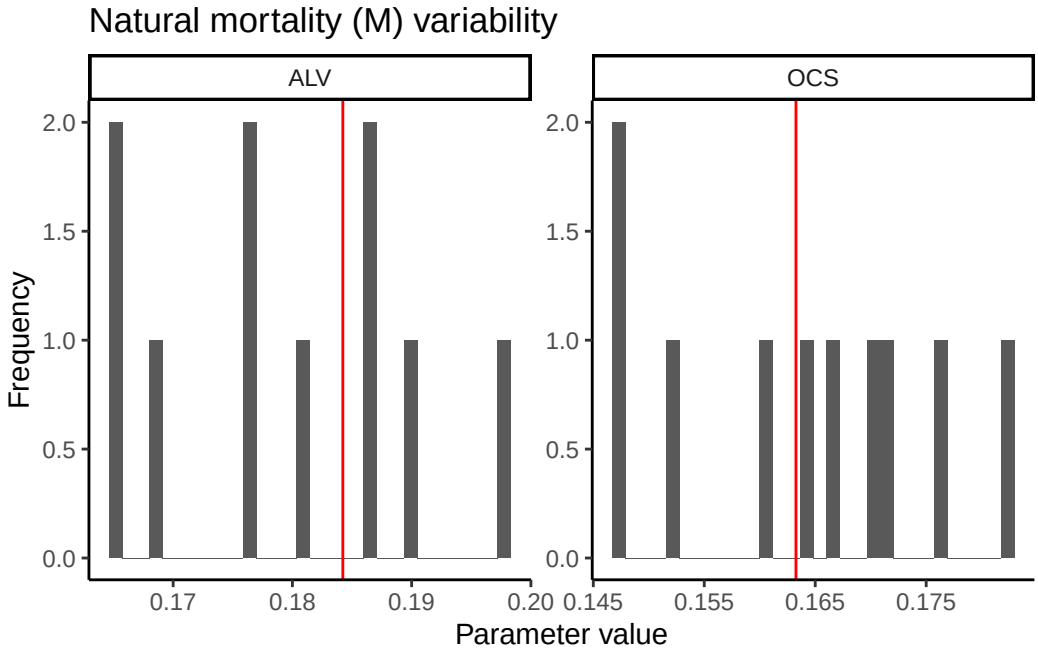
Here, we convert the data frame into a list by species, and vary parameters as in Hill et al. (in prep) by simulation. This also expands the dataframe from 0-Linf, and by sim. This structure makes it easy to apply EASI-Fish across many species.

```
lh_list <- prep_life_history(lh_df,
                      param_settings = list(K = list(dist = "lognormal", cv = 0.1),
                      schnute_a = list(dist = "lognormal", cv = 0.05),
                      lwa = list(dist = "lognormal", cv = 0.05),
                      M_normal = list(dist = "uniform",min = function(x) x - 0.02,
                      L50 = list(dist = "normal", sd = 2),
                      Lm = list(dist = "normal", sd = 2)))
head(lh_list[['ALV']])
```

```
# A tibble: 6 x 31
  spp     sim length mn_length max_age  nsim len_int length_type growth_model
  <chr> <int>  <dbl>     <dbl>   <int> <dbl>   <int> <chr>       <chr>
1 ALV       1      0         0      22    10       4 UF          Schnute
2 ALV       1      4         2      22    10       4 UF          Schnute
3 ALV       1      8         6      22    10       4 UF          Schnute
4 ALV       1     12        10      22    10       4 UF          Schnute
5 ALV       1     16        14      22    10       4 UF          Schnute
6 ALV       1     20        18      22    10       4 UF          Schnute
# i 22 more variables: max_length <int>, Linf <dbl>, K <dbl>, t0 <dbl>,
#   lwa <dbl>, lwb <dbl>, Lm <dbl>, L50 <dbl>, r <dbl>, L0 <dbl>, M <dbl>,
#   min_depth <int>, max_depth <int>, schnute_len1 <dbl>, schnute_len2 <lgl>,
#   schnute_age1 <dbl>, schnute_age2 <lgl>, schnute_a <dbl>, schnute_b <lgl>,
#   logistic_alpha <lgl>, logistic_g <dbl>, M_normal <dbl>
```

Here, we can inspect how natural mortality has been varied.

```r
bind_rows(lh_list) |>
  dplyr::select(spp, sim, M_normal) |>
  dplyr::distinct(sim, spp, .keep_all = TRUE) |>
  ggplot() +
  geom_histogram(aes(x = M_normal)) +
  geom_vline(data = lh_df, aes(xintercept = M_normal), col = 'red') +
  facet_wrap(~spp, scales = "free", ncol = 5) +
  theme_classic() +
  labs(x = 'Parameter value', y = 'Frequency',
       title = 'Natural mortality (M) variability')
```



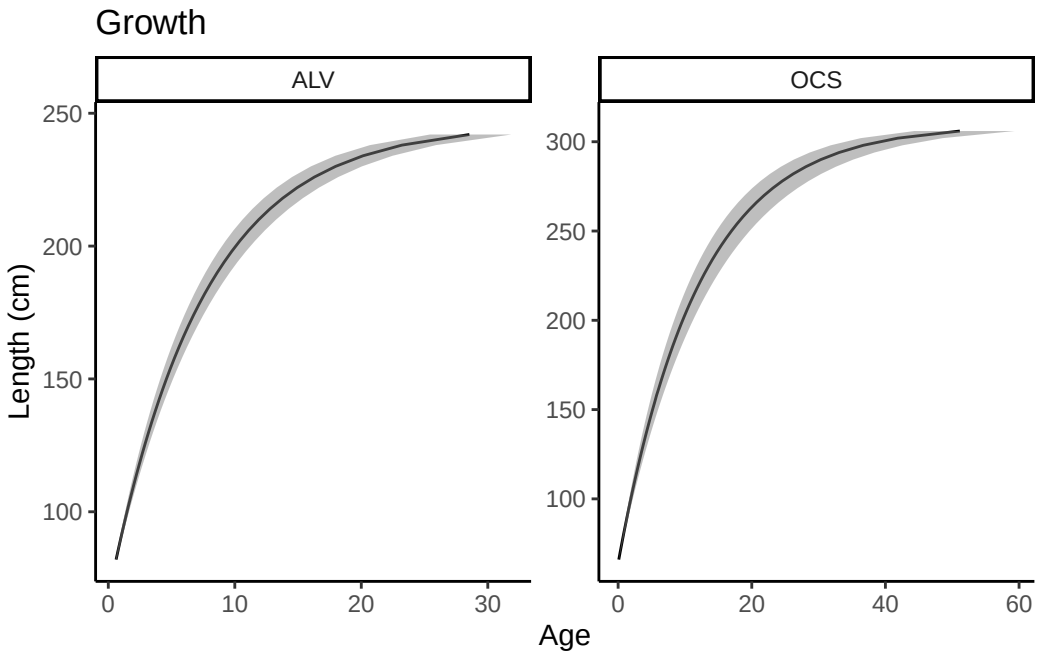
Lastly, now that we have the data in the correct format we can now estimate various processes including growth, maturity, length-weight, and delta-T. We also append selectivity data.

```
# load('data/easi_sels_wide3_gam.Rdata')
# sels <- easi_sels2 |>
#     dplyr::select(spp = sp, mn_length = len_bin, fshrys) |>
#   dplyr::filter(spp %in% spps)
# write.csv(sels, 'data/sels.csv')

sels <- read.csv('data/sels.csv')

lh_processes <- lh_list |>
  map(~ .x |>
        arrange(spp, sim, mn_length) |>
        mutate(age = easi_growth2(model = growth_model, len = mn_length,
                                  Linf = Linf, K = K, t0 = t0, L0 = L0,
                                  age1 = schnute_age1, age2 = schnute_age2,
                                  len1 = schnute_len1, len2 = schnute_len2,
                                  a = schnute_a, b = schnute_b,
                                  alpha = logistic_alpha, g = logistic_g),
        wt = lw(lwa, mn_length, lwb),
        mat = easi_mat2(len = mn_length, L50 = L50, r = r, Lm = Lm),
        dT = deltaT2(K = K, Linf = Linf, len = length, len_int = len_int))  |>
        dplyr::select(spp, sim, length, mn_length, age, wt, mat, dT,
          M_normal, L0, Linf, length_type, growth_model,
          min_depth, max_depth, max_age) |>
                left_join(sels, by = c('spp', 'mn_length')) |>
        pivot_longer(cols = fshrys,
    names_to = "fishery", values_to = "sel") |>
        dplyr::filter(mn_length >= 0 & length <= (Linf + 4)) |>
      dplyr::select(spp, sim, fishery, everything()))
```

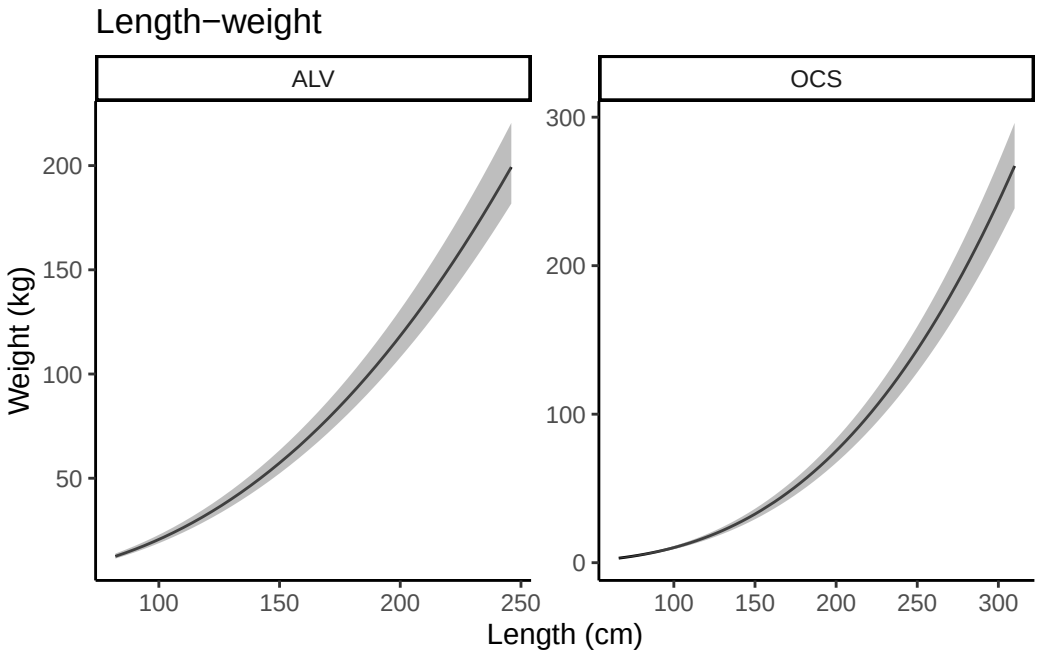
Now we can plot some of these life history processes to make sure they look reasonable.

```
bind_rows(lh_processes) |>
  dplyr::select(spp, sim, mn_length, age, L0) |>
  dplyr::filter(mn_length >= L0) |>
  group_by(spp, mn_length) |>
  summarise(mn_age = mean(age, na.rm = T), lwr_age = quantile(age, 0.0275, na.rm = T),
            upr_age = quantile(age, 0.975, na.rm = T)) |>
  ggplot() +
  aes(x = mn_age, y = mn_length) +
```

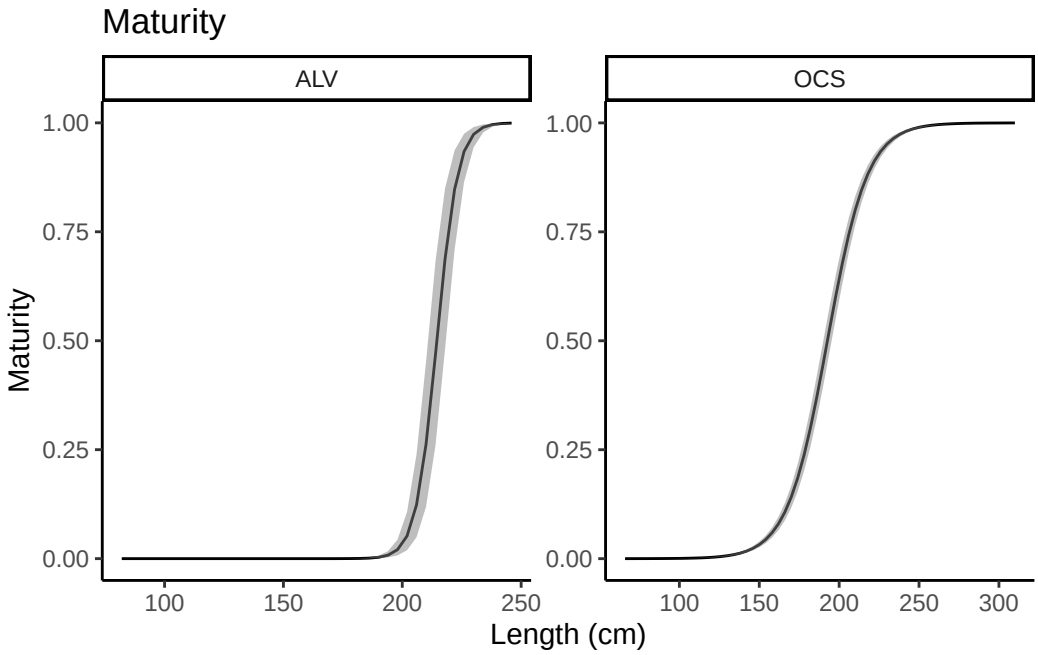
```
geom_line(col = 'black') +
geom_ribbon(aes(xmin = lwr_age, xmax = upr_age), fill = 'grey50', alpha = 0.5) +
facet_wrap(~spp, scales = 'free', ncol = 5) +
labs(x = 'Age', y = 'Length (cm)' , title = "Growth") +
theme_classic()
```



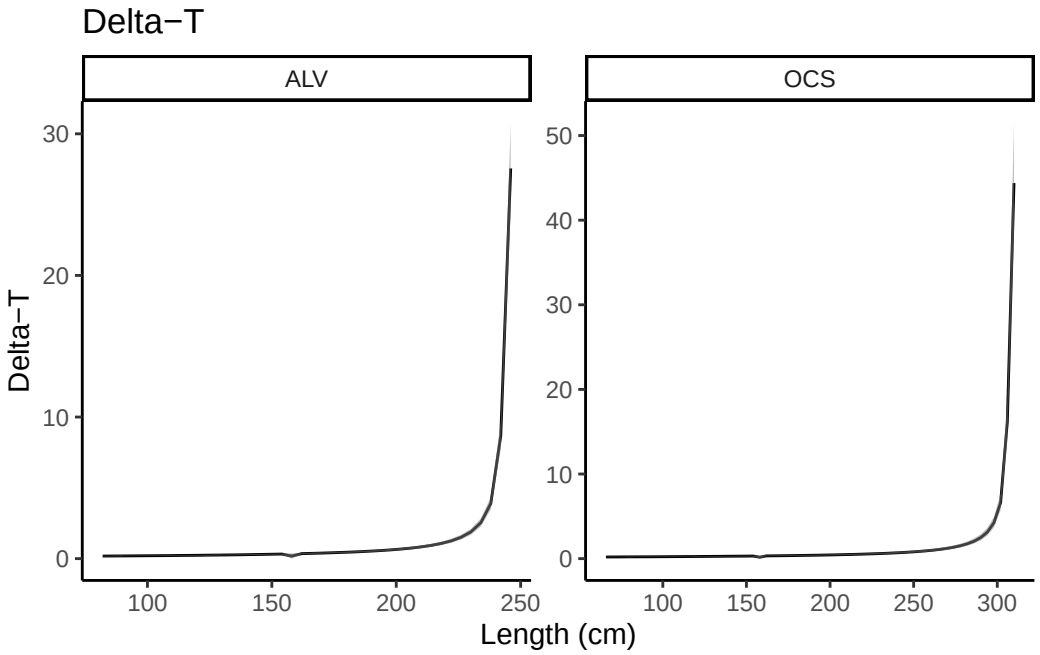
```r
bind_rows(lh_processes) |>
  dplyr::select(spp, sim, mn_length, wt, L0) |>
  dplyr::filter(mn_length >= L0) |>
  group_by(spp, mn_length) |>
  summarise(mn_wt = mean(wt, na.rm = T), lwr_wt = quantile(wt, 0.0275, na.rm = T),
            upr_wt = quantile(wt, 0.975, na.rm = T)) |>
  ggplot() +
  aes(x = mn_length, y = mn_wt) +
  geom_line(col = 'black') +
  geom_ribbon(aes(ymin = lwr_wt, ymax = upr_wt), fill = 'grey50', alpha = 0.5) +
  facet_wrap(~spp, scales = 'free', ncol = 5) +
  labs(x = 'Length (cm)', y = 'Weight (kg)' , title = "Length-weight") +
  theme_classic()
```



```r
bind_rows(lh_processes) |>
  dplyr::select(spp, sim, mn_length, mat, L0) |>
  dplyr::filter(mn_length >= L0) |>
  group_by(spp, mn_length) |>
  summarise(mn_mat = mean(mat, na.rm = T), lwr_mat = quantile(mat, 0.0275, na.rm = T),
            upr_mat = quantile(mat, 0.975, na.rm = T)) |>
  ggplot() +
  aes(x = mn_length, y = mn_mat) +
  geom_line(col = 'black') +
  geom_ribbon(aes(ymin = lwr_mat, ymax = upr_mat), fill = 'grey50', alpha = 0.5) +
  facet_wrap(~spp, scales = 'free', ncol = 5) +
  labs(x = 'Length (cm)', y = 'Maturity' , title = "Maturity") +
  theme_classic()
```

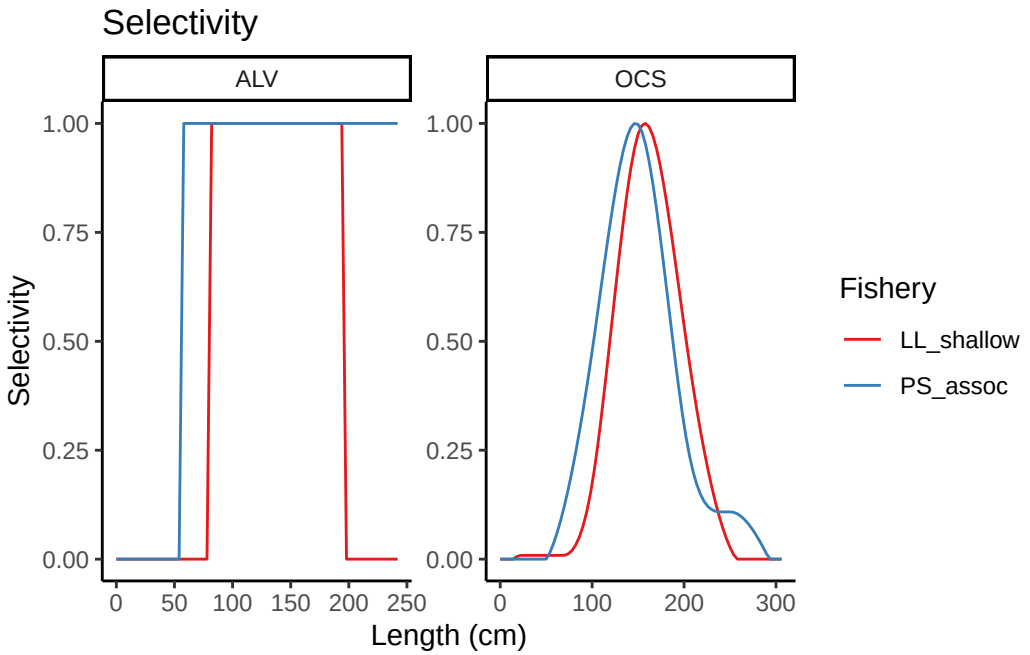


```
bind_rows(lh_processes) |>
  dplyr::select(spp, sim, mn_length, dT, L0) |>
  dplyr::filter(mn_length >= L0) |>
  group_by(spp, mn_length) |>
  summarise(mn_dT = mean(dT, na.rm = T), lwr_dT = quantile(dT, 0.0275, na.rm = T),
            upr_dT = quantile(dT, 0.975, na.rm = T)) |>
  ggplot() +
  aes(x = mn_length, y = mn_dT) +
  geom_line(col = 'black') +
  geom_ribbon(aes(ymin = lwr_dT, ymax = upr_dT), fill = 'grey50', alpha = 0.5) +
  facet_wrap(~spp, scales = 'free', ncol = 5) +
  labs(x = 'Length (cm)', y = 'Delta-T' , title = "Delta-T") +
  theme_classic()
```



Lastly, we plot selectivity data for the two fisheries here.

```
sels |>
  pivot_longer(-c(spp, mn_length), names_to = 'fishery', values_to = 'sel') |>
  ggplot() +
  aes(mn_length, sel, col = fishery) +
  geom_line() +
  theme_classic() +
  facet_wrap(~spp, scales = 'free') +
  scale_color_brewer(palette = 'Set1') +
  labs(x = 'Length (cm)', y = 'Selectivity', title = 'Selectivity',
       col = 'Fishery')
```



Preparing fisheries data

Now we read in the fisheries data and prepare it for input into EASI-Fish with a similar pipeline to the life history data above. However, fisheries data is a little more complicated because some parameters are fishery related only, and some are fishery and species specific.

```
fshry <- read.csv('data/fishery_inputs.csv')

fshry_df <- expand_grid(spp = spps, nsim = nsims, fshry) |>
  pivot_longer(-c(spp, nsim, param), values_to = 'value', names_to = 'fishery') |>
  pivot_wider(names_from = 'param', values_from = 'value') |>
  dplyr::select(-matches("_(min|max|se|sd|dist|stdD)$")) |>
  dplyr::rename(min_depth_flt = min_depth, max_depth_flt = max_depth) |>
    mutate(across(.cols = -c(spp, fishery),~ as.numeric(.)))
```

Here, we append overlap, at-vessel mortality and post-release mortality values to the data frame by species and fishery. Two different estimates of overlap are included including the original estimation of overlap using presence/absence at an annual timestep, and the updated approach using the cumulative sum at a monthly timestep. See Hill et al. (in prep) for details on how these were calculated. At-vessel mortality and post-release mortality estimates were extracted from the literature for each species and gear type (i.e. purse seine, longline).

```
# load('data/overlaps_ensemble_df_combined_coastal2b_newxtract_ngb.Rdata')
#
# olaps <- olaps_df_all |>
#   dplyr::filter(spp %in% spps & fishery %in% fshrys) |>
#   ungroup() |>
#   dplyr::select(spp, fishery, temp_res, olap_pa, olap_cumsum) |>
#   pivot_wider(names_from = temp_res, values_from = c(olap_pa, olap_cumsum)) |>
#   dplyr::select(-c(olap_pa_month, olap_cumsum_annual))
#
# write.csv(olaps, ('data/olap_inputs.csv'))

tmp <- read.csv('data/spp_fishery_inputs.csv')

fshry_df <- left_join(fshry_df, tmp, by = c('spp', 'fishery'))
```

Here we append the life history df to the fishery df so that we can calculate encounterability which looks at the overlap of fishing gear and species distribution vertically in the water column. To calculate this you need the minimum and maximum depths of the gear, and the species.

```
spp_depth <- lh_df |>
  dplyr::select(spp, min_depth_spp = min_depth, max_depth_spp = max_depth)

fshry_df <- left_join(fshry_df, spp_depth, by = c('spp'))

fshry_df <- fshry_df |>
  rowwise() |>
  mutate(encount = encount(min_depth_spp, max_depth_spp,
                  as.numeric(min_depth_flt), as.numeric(max_depth_flt))) |>
  dplyr::select(-c(min_depth_spp, max_depth_spp,
                   min_depth_flt, max_depth_flt, Lc))
```

```
# load(file = 'data/easifish_AVM_PRM_upd2.Rdata')
#
# AVM_PRM <- easifish_AVM_PRM |>
#   dplyr::filter(sp %in% spps & fishery %in% fshrys) |>
#   dplyr::select(spp = sp, fishery, AVM = AVM_est2, PRM = PRM_est2)
#
# write.csv(AVM_PRM, ('data/AVM_PRM_inputs.csv'))

# AVM_PRM <- read.csv('data/AVM_PRM_inputs.csv')
#
# fshry_df <- left_join(fshry_df, AVM_PRM, by = c('spp', 'fishery'))
```

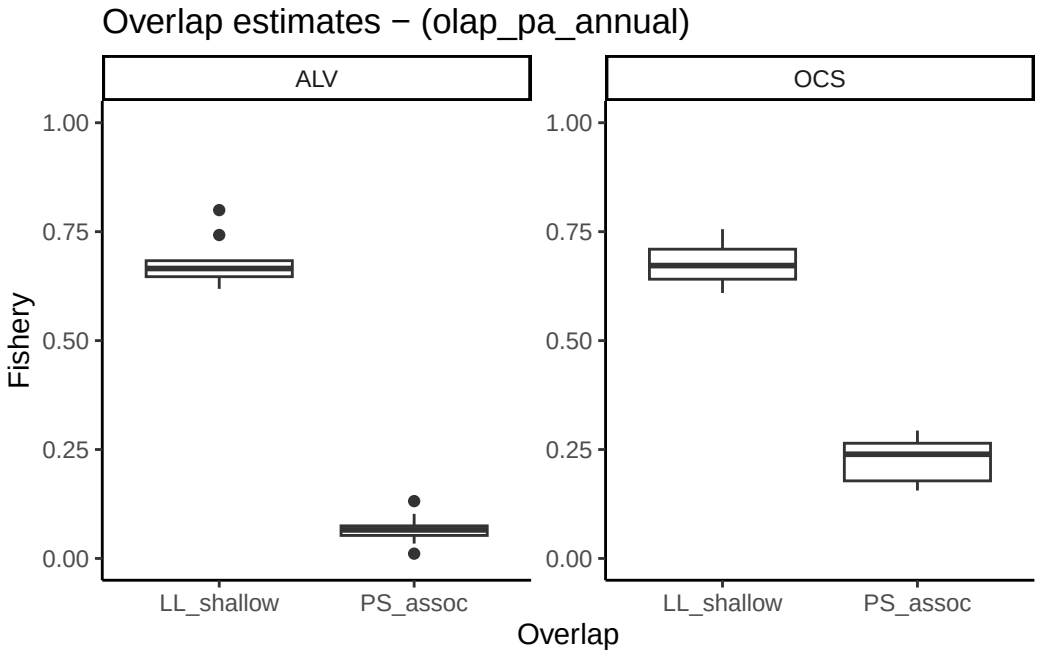
Now, we have compiled all the required fishery parameters, we can now rearrange this data so that it can be combined with the life history information, and vary any parameters we desire. As in Hill et al. (in prep), we add variability to the overlap parameter. Now the fishery data is in an equivalent list structure as the life history data for easy input into EASI-Fish.

```
fshry_list <- prep_fishery_data(fshry_df, param_settings = list(
        olap_pa_annual = list(dist = "truncnorm", sd = 0.05, min=0, max=1),
        olap_cumsum_month = list(dist = "truncnorm", sd = 0.05, min=0, max=1)))
```

Now we can plot some of the fishery data to see what it looks like.

```
bind_rows(fshry_list) |>
  dplyr::select(sim, spp, fishery, olap_pa_annual) |>
  dplyr::distinct(sim, spp, fishery, olap_pa_annual, .keep_all = TRUE) |>
  ggplot() +
  geom_boxplot(aes(x = fishery, y = olap_pa_annual)) +
  facet_wrap(~spp, scales = 'free', ncol = 5) +
```

```
  theme_classic() +
  labs(x = 'Overlap', y = 'Fishery', title = 'Overlap estimates - (olap_pa_annual)') +
  ylim(0,1)
```



Estimating susceptibility

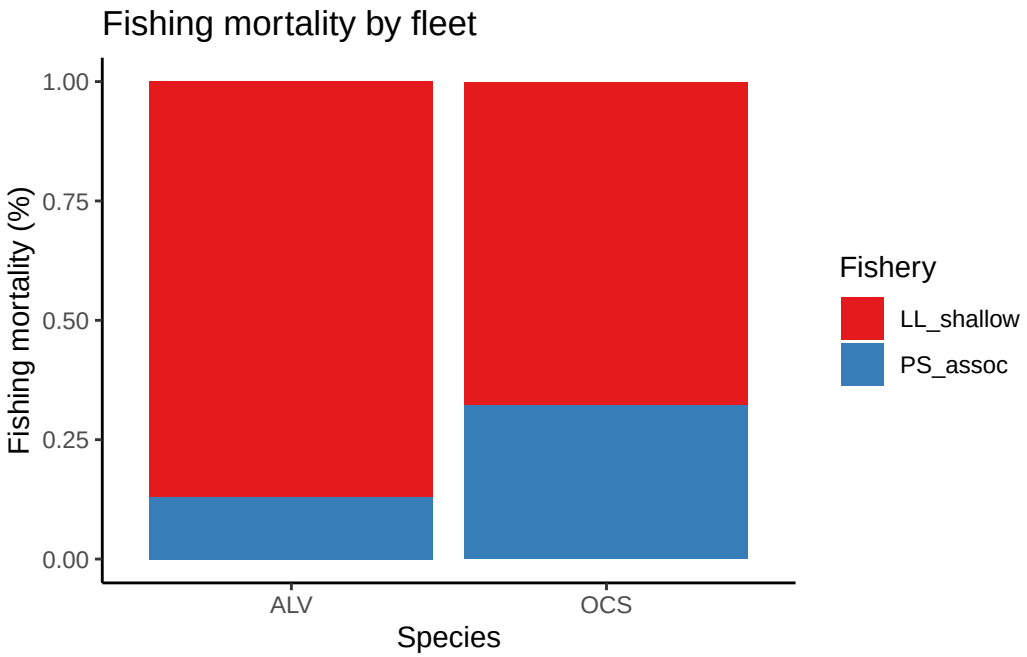
Now that we have prepped all the data we can now apply EASI-Fish. The first thing we do is estimate susceptibility. Here, we combine the life history and fisheries lists into a single dataframe, however these could be saved and read in separately within a loop if running a large number of species or scenarios.

```
# Define olap estimate of choice
olap_col <- 'olap_pa_annual'

# Combine data into a single object for ease
input_data <- map2(lh_processes, fshry_list,
  ~ left_join(.x, .y, by = c("spp", "sim", "fishery"))) |>
  bind_rows()

# Estimate susceptibility - check column names
Fsusc_df <-
```

```
  input_data |>
  mutate(Ffinite = get_Ffinite(olap_col, olap = .data[[olap_col]],
                               season = season, avail = avail, encount = encount,
                               sel = sel, avm = AVM, prm = PRM),
         Fadj = (Ffinite * Q) * effort,
         Fage = -log(1 - Fadj)) |>
  dplyr::select(spp, sim, fishery, mn_length, sel, Ffinite, Fadj, Fage)
```

Here, we collate susceptibility results by fishery and plot.

```
# Summarise susceptibility by fleet
Fsusc_fshry <-
  Fsusc_df |>
  dplyr::filter(!is.na(sel)) |>
  group_by(sim, spp, fishery) |>
  dplyr::summarise(nbins = n_distinct(mn_length),
                   Ffinite = sum(Ffinite)/nbins,
                   Fadj = sum(Fadj)/nbins,
                   Fage = sum(Fage)/nbins)

Fsusc_fshry |>
  group_by(spp, fishery) |>
  summarise(Fage_mn = mean(Fage, na.rm = T)) |>
  ggplot() +
  aes(spp, Fage_mn, fill = fishery) +
  geom_bar(stat = 'identity', position = 'fill') +
  scale_fill_brewer(palette = 'Set1') +
  theme_classic() +
  labs(x = 'Species', y = 'Fishing mortality (%)', fill = 'Fishery',
       title = 'Fishing mortality by fleet')
```



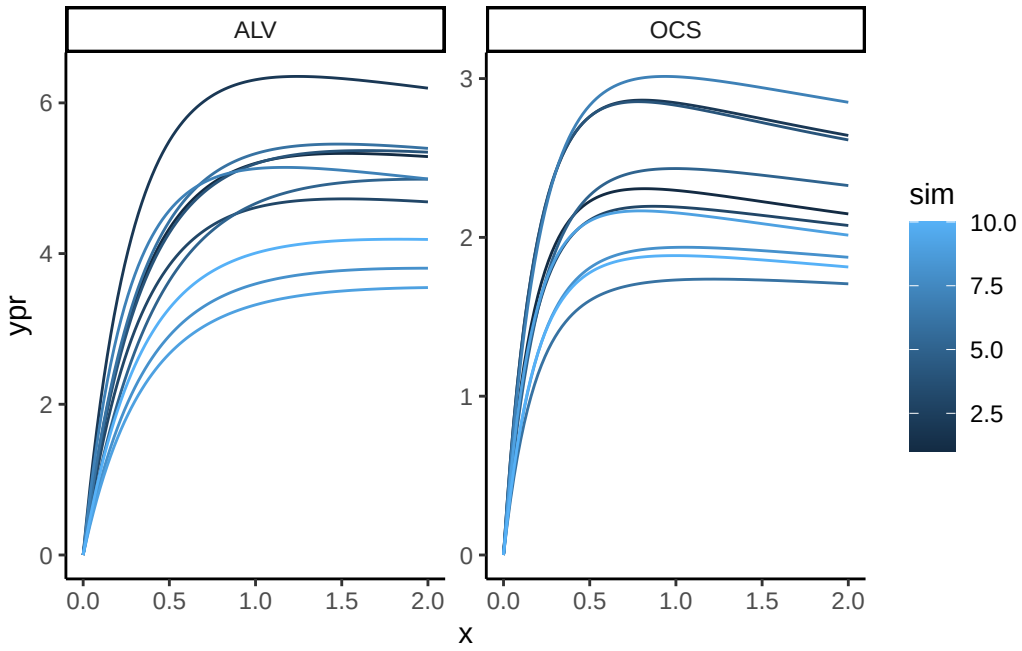
And by length.

```
# Summarise susceptibility by length
Fsusc_len <-
  Fsusc_df |>
  dplyr::select(spp, sim, fishery, mn_length, Ffinite, Fadj, Fage) |>
  group_by(spp, sim, mn_length) |>
  dplyr::summarise(Finst = sum(Fage)) |>
  mutate(Fsusc = 1 - exp(-Finst)) |>
  filter(!is.na(Fsusc)) |>
  group_by(spp) |>
  mutate(Fsusc_min = min(mn_length[Fsusc>0]))
#
# Fsusc_len |>
#   ggplot() +
#   aes(mn_length, Fsusc, col = sim, group = sim) +
#   geom_line() +
#   theme_classic() +
#   facet_wrap(~spp)
```

Estimating productivity

Now that we have susceptibility values, we can input these into the yield-per-recruit analysis and spawn-per-recruit to determine productivity values for EASI-Fish. Note that the biomass values output are not independent of the susceptibility values and so only susceptibility values (i.e. fishing mortality) were presented in Hill et al. (in prep). But, we still apply this portion of the analysis for completeness. It also compares the susceptibility results against various reference points which is needed.

```
# Combine susceptibility results with lh processes to run ypr and spr
Bprod_ins <- Fsusc_len |>
  left_join(bind_rows(lh_processes),
            by = c("sim", "spp", "mn_length")) |>
  dplyr::select(sim, spp, mn_length, wt, mat, dT, M = M_normal, Fsusc) |>
  group_by(sim, spp) |>
  nest()

# Apply get_Bprod function from easi_funs script
Bprod_df <- get_Bprod3(Bprod_ins, x_vals = seq(0, 10, 0.02))

# Example results output
Bprod_df |>
  ggplot() +
  aes(x, ypr, col = sim, group = sim) +
  geom_line() +
  facet_wrap(~spp, scales = 'free_y') +
  xlim(0,2) +
  theme_classic()
```



Estimating summary statistics

Finally, with these results extracted we can estimate fishing mortality and biomass values relative to various reference points. This function does this for each species and simulation and extracts an array of relevant values.

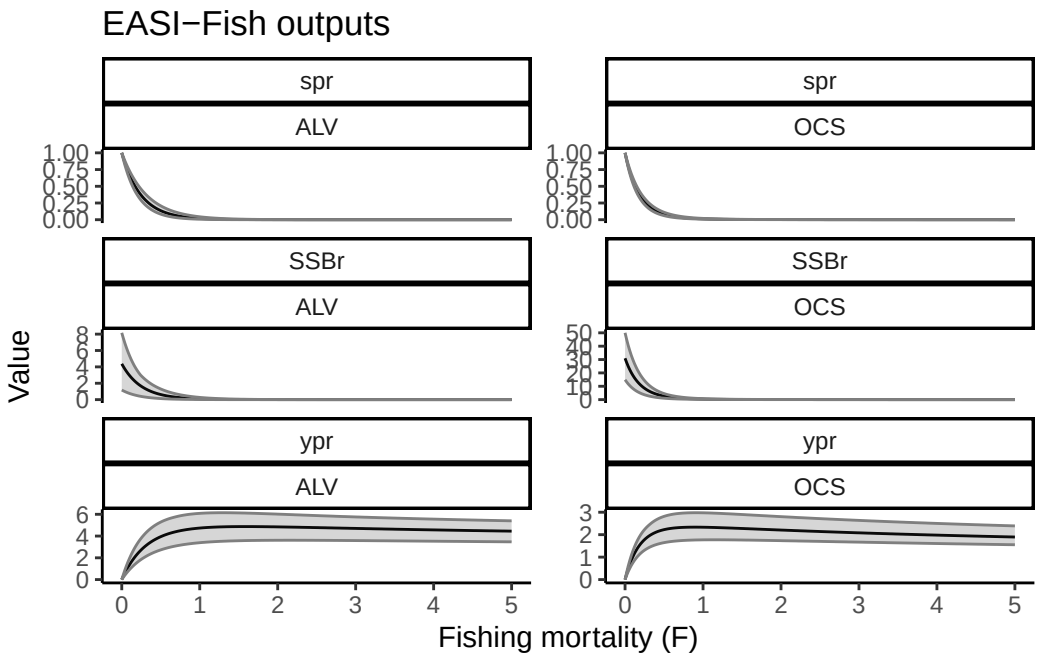
```
easi_stats <- get_easi_stats2(Bprod_df, Fsusc_fshry)

head(easi_stats)
```

```
# A tibble: 6 x 4
# Groups:   sim, spp [1]
    sim spp   param value
  <int> <chr> <chr> <dbl>
1     1 ALV   Fest  0.113
2     1 ALV   FMSY  1.52
3     1 ALV   F01   0.72
4     1 ALV   F20   0.42
5     1 ALV   F40   0.24
6     1 ALV   F60   0.14
```

Results

Now that we have run the EASI-Fish analysis, we can explore output and present the data in a number of different ways. First, we plot yield-per-recruit, spawn-per-recruit and spawning stock biomass by recruit to inspect that outputs are realistic.

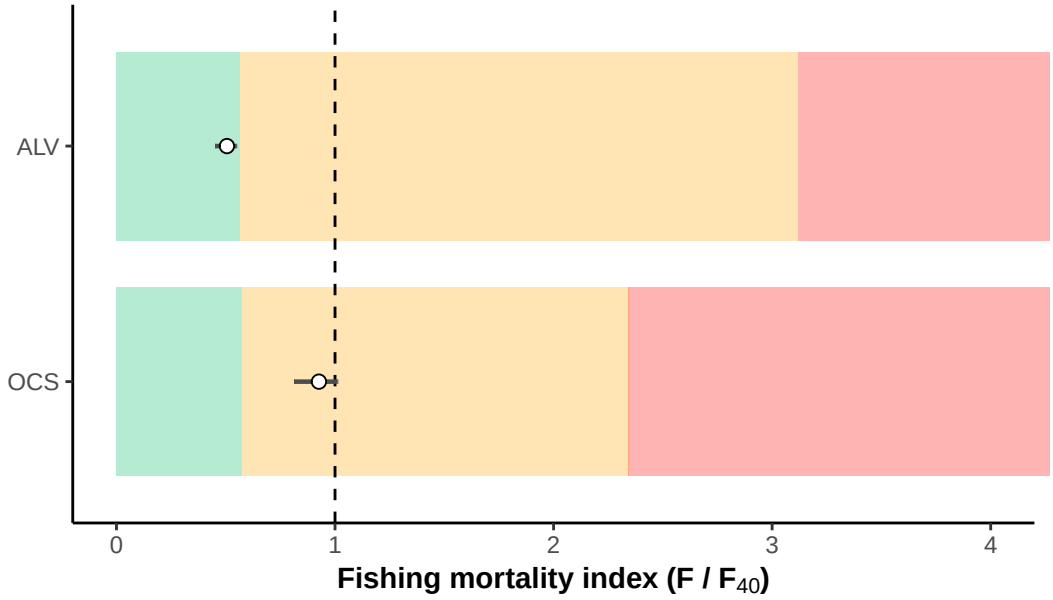
```
Bprod_df |>
  pivot_longer(-c(sim, spp, x), names_to = 'param', values_to = 'value') |>
  group_by(spp, x, param) |>
  summarise(val_mn = mean(value, na.rm = T),
            val_lwr = quantile(value, 0.0275, na.rm = T),
            val_upr = quantile(value, 0.975, na.rm = T)) |>
  dplyr::filter(param != 'slope') |>
  ggplot() +
  aes(x, val_mn) +
  geom_line(col = 'black') +
  geom_ribbon(aes(x = x, ymin = val_lwr, ymax = val_upr),
              col = 'grey50', alpha = 0.2) +
  facet_wrap(vars(param, spp), scales = 'free_y', ncol = 2) +
  xlim(0,5) +
  theme_classic() +
  labs(x = 'Fishing mortality (F)', y = 'Value',
       title = 'EASI-Fish outputs')
```



Next we can look at fishing mortality relative to various reference points.

```
easi_summ <- easi_stats |>
  dplyr::select(sim, spp, param, value) |>
  filter(param %in% c('Fest','FMSY','F40','F01','F20', 'F60')) |>
  pivot_wider(names_from = param, values_from = value) |>
  mutate(FF40 = Fest/F40, FMSY_F40 = FMSY/F40, F01_F40 = F01/F40,
         FF60 = Fest/F60, F60_F40 = F60/F40)

# Extract quantiles by spp and for olap_cumsum_month scen
easi_q <- easi_summ |>
  group_by(spp) |>
  summarise(q25 = quantile(FF40, 0.25, na.rm = TRUE),
            q50 = quantile(FF40, 0.50, na.rm = TRUE),
            q75 = quantile(FF40, 0.75, na.rm = TRUE))

# Create background colours species specific
tiles <- easi_summ |>
  group_by(spp) |>
  summarise(F60_F40 = mean(F60_F40, na.rm = T),
            F01_F40 = mean(F01_F40, na.rm = T)) |>
  rowwise() |>
```

```
  mutate(list_tiles = list(tibble(xmin = c(0, F60_F40, F01_F40),
       xmax = c(F60_F40, F01_F40, 15),fill = c("#00BE67", "orange", "red")))) |>
  tidyr::unnest(cols = list_tiles)

# Figure 5 error bar plot
p<-
ggplot() +
  geom_tile(data = tiles,aes(x = (xmin + xmax)/2,y = spp,
                             width = xmax - xmin,height = 0.8,fill = fill),alpha = 0.3) +
  geom_vline(xintercept = 1, linetype = 'dashed') +
  geom_errorbarh(data = easi_q,aes(y = spp, xmin = q25, xmax = q75),
                 height = 0,colour = "grey30",linewidth = 0.8) +
  geom_point(data = easi_q,aes(x = q50, y = spp),size = 2.2,
             colour = "black",fill = "white",shape = 21) +
  scale_fill_identity() +
  scale_y_discrete(limits = rev(unique(tiles$spp))) +
  coord_cartesian(xlim = c(0, 4), clip = "off") +
  #xlim(0, 10) +
  theme_classic() +
  labs(x = expression(bold("Fishing mortality index (F / F"[40]*")")),
       y = "Species") +
  theme(axis.title.y = element_blank())
#p

library(grid)
p2 <- p + annotation_custom(grob = textGrob(expression(bold(F[60])),
                                            gp = gpar(fontsize = 10, fontface = "bold")),
                            xmin = 0.65, xmax = 0.75, ymin = 25.8, ymax = 25.8) +
  annotation_custom(grob = textGrob(expression(bold(F[40])),
                                    gp = gpar(fontsize = 10, fontface = "bold")),
                    xmin = 0.9, xmax = 1.1, ymin = 25.8, ymax = 25.8) +
  annotation_custom(grob = textGrob(expression(bold(F[0.1])),
                                    gp = gpar(fontsize = 10, fontface = "bold")),
                    xmin = 1.9, xmax = 2, ymin = 25.8, ymax = 25.8) +
  ggtitle("")
p2
```



And finally a table with a variety of results.

```
tbl <-
easi_stats |>
  group_by(spp, param) |>
  summarise(mn_val = mean(value, na.rm = T),
            val_lwr = quantile(value, 0.0275, na.rm = T),
            val_upr = quantile(value, 0.975, na.rm = T)) |>
  mutate(across(where(is.numeric), round, 3)) |>
  mutate(output = paste0(mn_val, ' (', val_lwr, '-', val_upr, ')')) |>
  dplyr::select(spp, Parameter = param, output) |>
  pivot_wider(names_from = spp, values_from = output)

library(flextable)

flextable(tbl) |>
  bold(part = "header") |>
  bold(j = 1, part = "body") |>
  width(j = c(2, 3), width = 2.5) |>
  set_table_properties(layout = "fixed") |>
  autofit()
```

Parameter	ALV	OCS
F01	0.744 (0.575-0.898)	0.424 (0.385-0.476)
F20	0.41 (0.33-0.518)	0.322 (0.285-0.376)
F40	0.24 (0.19-0.306)	0.182 (0.16-0.216)
F60	0.136 (0.105-0.171)	0.104 (0.1-0.12)
F80	0.062 (0.06-0.076)	0.05 (0.04-0.06)
FF01	0.169 (0.117-0.26)	0.399 (0.33-0.49)
FFMSY	0.078 (0.052-0.126)	0.184 (0.14-0.231)
FMSY	1.656 (1.18-2.155)	0.93 (0.785-1.184)
Fest	0.122 (0.102-0.149)	0.168 (0.152-0.192)
SSB	2.814 (0.71-4.853)	13.937 (5.933-24.469)
SSB01	0.258 (0.033-0.553)	3.703 (1.401-6.871)
SSB20	0.85 (0.229-1.592)	5.767 (2.711-9.249)
SSB40	1.682 (0.445-3.175)	11.715 (5.745-18.902)
SSB60	2.558 (0.675-4.843)	17.762 (8.322-29.94)
SSB80	3.416 (0.9-6.336)	23.465 (11.397-36.71)
SSBMSY	0.013 (0-0.036)	0.62 (0.121-1.418)
SSBSSB40	1.669 (1.275-2.002)	1.145 (0.89-1.411)
SSBSSBMSY	1623.94 (129.244-8834.355)	28.147 (17.304-51.095)