DEPMWt Estimation

Jonathan Smart

2020-01-20

Load the DEPM and tidyverse libraries.

```r
library(DEPM)
library(tidyverse)
```

Introduction

A recent advancement in Daily Egg Production Methods (DEPM) is the development of the DEPMWt approach (McGarvey et al. in review). This approach is similar to a traditional DEPM performed using the methods of Parker (1980) except that rather than using the mean body weight and mean fecundity of female fish as DEPM parameters, these quantities are instead broken down into weight bins. The benefits of these weight bins are that weight and fecundity are much better described when the weight structure of a population is not normally distributed and therefore more precise estimates of biomass are obtained.

The `DEPM` package includes functions that perform the DEPMWt approach as well as a traditional DEPM approach, with several functions used in both approaches. Here, an example of the DEPMWt approach is given using South Australian Snapper (*Pagrus auratus*).

Necessary data

Five datasets are required to use the DEPMWt approach:

- Egg density data to estmate daily egg production (P_0)
- Adult sex ratios to estimate R
- Female spawning fractions to estimate S
- Female total weights to estimate the proportion of females in each weight bin (PN_{wt})
- Female total weight and batch fecundities to determine a fecundity-at-weight relationship (F_{wt})

Spawning Area (A) is not calculated as part of this package but its estimates will be included as one of the final steps.

DEPMWt parameter estimation

Daily egg production (P_0)

Daily egg production is estimated using a stage based estimator (McGarvey et al. 2018) via the function `Estimate_P0()` which requires a dataset that has been formatted as such:

```
# egg density structure
str(egg_data)
#> 'data.frame':    2961 obs. of  6 variables:
#>  $ Region   : Factor w/ 4 levels "NGSV","NSG","SGSV",..: 1 1 1 1 1 1 1 1 1 1 ...
#>  $ Site     : Factor w/ 329 levels "GV18_005","GV18_006",..: 1 2 3 4 5 6 7 8 9 10 ...
#>  $ Stage_num: num  1 1 1 1 1 1 1 1 1 1 ...
#>  $ Density  : num  0 0 0 0 0 0 0 0 0 0 ...
#>  $ Age      : num  0.178 0.173 0.17 0.173 0.188 ...
#>  $ Hatch    : num  1.3 1.26 1.24 1.26 1.36 ...

# egg density head()
head(egg_data)
#>   Region     Site Stage_num Density       Age    Hatch
#> 1   NGSV GV18_005         1       0 0.1782736 1.295414
#> 2   NGSV GV18_006         1       0 0.1730048 1.257129
#> 3   NGSV GV18_007         1       0 0.1701226 1.236185
#> 4   NGSV GV18_008         1       0 0.1727973 1.255621
#> 5   NGSV GV18_013         1       0 0.1877140 1.364012
#> 6   NGSV GV18_014         1       0 0.1839234 1.336468
```

This dataset is in a long format where every row is a density estimate of the eggs in each stage of each sample. This dataset contains several columns that must include:

- Site - which represents each sample in the data (i.e. one plankton tow a location). It's variable name must be specified as an argument using `site =`
- Density - the number of eggs in a single development stage in a metre cubed of seawater. The function will automatically determine the density column based on similar names (i.e. dens, density, DENSITY or other similar names). However, there cannot be multiple density columns.
- Age - the age in days of each egg stage in each sample. The function will automatically determine the Age column based on similar names in a similar manner to the `Density` variable
- Hatching time - The age in days where the eggs in each sample are estimated to hatch (usually differs based on temperature). The function will automatically determine the `Hatch` column based on similar names in a similar manner to the `Density` variable
- Z - a pre-specified level of egg mortality used to determine P_0. Multiple values can be provided to simultaneously determine different values of P_0 based on this parameter. Instruction on how to do this and how to perform sensitivity tests are provided in the final section of this vignette.

Additional columns can be included in the dataset, so the user does not need to remove variables that could be useful to them later on. For example, in the `egg_data` dataset, the `Stage_num` variable represents the development stage of each density estimate in each sample. However as each row of this dataset is a different observation, this variable is not needed by the function.

The function also has the ability to break the data down spatially and temporaly. If the `Region` and `Time` arguments are not used then P_0 will be estimated using all of the data. If you provide either `Region` or `Time` then a P_0 will be returned for each Region/Time combination. Therefore, a time series of data can analysed in a single function call. Here is an example using 4 DEPM surveys conducted in different areas but in the same year. The results are returned as a dataframe where each row is a survey and the estimate, standard error and specified mortality are returned.

```
# run P0 function
P0_results <- Estimate_P0(data = egg_data,
                          site = "Site",
                          Region = "Region",
                          Z = .4)

P0_results
```

```
#> # A tibble: 4 x 4
#>   Region    P0  P0_se     Z
#>   <fct>  <dbl>  <dbl> <dbl>
#> 1 NGSV   2.03  0.391    0.4
#> 2 NSG    0.947 0.218    0.4
#> 3 SGSV   0.527 0.129    0.4
#> 4 SSG    0.725 0.0849   0.4
```

Spawning fraction (S)

Spawning fraction is estimated using the function `Estimate_Spawning_fraction()` which requires a dataset that has been formatted as two columns: 1) the number of females in spawning condition and 2)the total number of females. Each row represents a sample and the spawning fraction is estimated using a ratio estimator.

The correct columns in the dataset will be detected by the function. The column with the total number of females should be called "Total", "Tot" or something similar. The column with the number of spawning females can include the term "spawn" or "yes" depending on how your dataset is setup. Ours is named "yes" as we had a "yes" or "no" designation for spawning fish.

```
str(S_data)
#> 'data.frame':    20 obs. of  3 variables:
#>  $ yes  : int  1 16 9 5 3 2 1 2 4 5 ...
#>  $ no   : int  6 2 1 4 2 1 2 4 2 1 ...
#>  $ Total: int  7 18 10 9 5 3 3 6 6 6 ...

head(S_data)
#>   yes no Total
#> 1   1  6     7
#> 2  16  2    18
#> 3   9  1    10
#> 4   5  4     9
#> 5   3  2     5
#> 6   2  1     3
```

Similar to other functions, specifying the `Region` or `Time` argument with relevant column names will group the estimates according to specific surveys. However, here an example is given without a region or time grouping. The results are returned as a dataframe with the estimate, its variance, the standard error and its CV. Each row represents a survey.

```
Spawn_results <- Estimate_Spawning_fraction(S_data)

Spawn_results
#>   Ratio estimate    Variance         SE         CV
#> 1       0.722488 0.002977749 0.05456875 0.07552894
```

Sex Ratio (R)

Sex ratio is estimated using the function `Estimate_sex_ratio()` which requires a dataset that has been formatted to include the weight of males and females along with `Region`/`Time` variables. Each row represents a sample and the sex ratio is estimated using a ratio estimator in the same manner as spawning fraction. The same Region/Time break down can be applied and Region is available in this dataset. The results returned are the same format as `Estimate_Spawning_fraction()`.

```
head(R_data)
#>           Sample.code Year Region        F        M    Total
#> 1 ST12/1801B18-Dec-18 2018   NGSV  2.45000  6.42500  8.87500
#> 2 ZA12/1801B19-Dec-18 2018   NGSV 11.54800 10.83200 22.38000
#> 3 DC12/1801B11-Dec-18 2018    NSG  8.95600 16.22400 25.18000
#> 4 SA12/1801B12-Dec-18 2018    NSG  2.94000  7.18500 10.12500
#> 5     GLLOYD16-Dec-18 2018   SGSV 43.33143 51.11838 94.44980
#> 6     GLLOYD17-Dec-18 2018   SGSV 20.28760 16.20903 36.49663

sex_ratio_results <- Estimate_sex_ratio(R_data, Region = "Region")

sex_ratio_results
#>   Region Ratio estimate     Variance         SE         CV
#> 1   NGSV      0.4478643 0.0095201654 0.09757133 0.21785913
#> 2    NSG      0.3369494 0.0007137716 0.02671650 0.07928936
#> 3   SGSV      0.3939694 0.0011661437 0.03414885 0.08667892
#> 4    SSG      0.5741385 0.0018484630 0.04299376 0.07488394
```

Female weight data

The key difference between teh DEPMWt approach and the traditional DEPM is how female weight and fecundity are handled. In the DEPMWt approach, female weight is broken into discrete weight bins and their variance is described using a multinomial distribution. This is handled by the `Estimate_proportion_female()` function which requires a dataframe of total female weight. This data.frame only needs one column but this does not need to be subset from a larger dataset as the column is specified by the `Weight` argument. Other columns are ignored unless (like other parameter functions), a `Region` or `Time` argument is specified.

To break the female weights into bins, an upper bound and a bin width is required. Here 27 weight bins are applied for Snapper begining at zero and ending at 13500g in 500g bins.

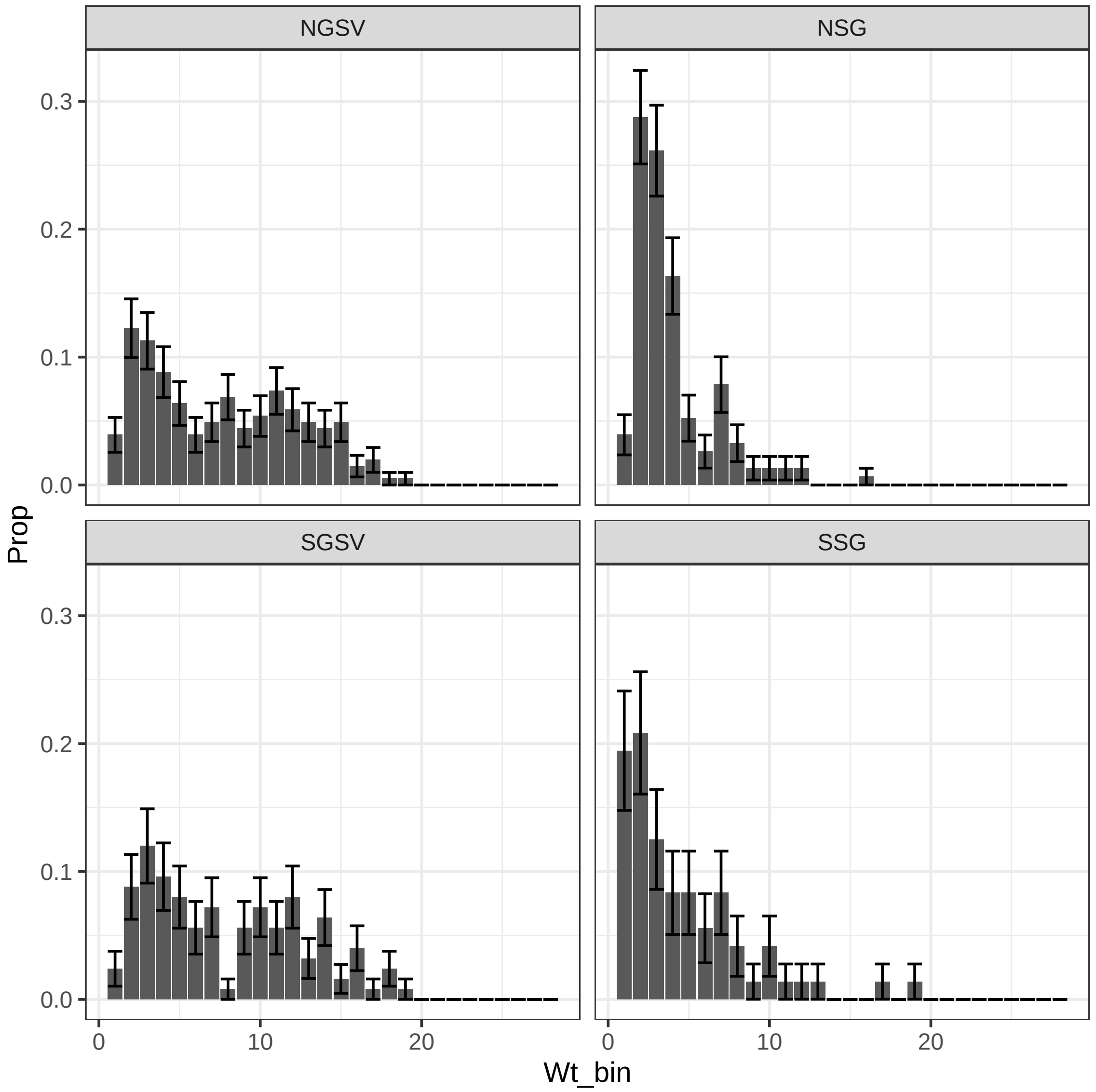
NOTE: Weight data must be Total Weight in grams

```
head(Wt_data)
#>   Region CFL  Tot.Wt
#> 1    NSG 627 4002.11
#> 2    NSG 657 4551.52
#> 3    NSG 610 3710.49
#> 4    NSG 693 5271.38
#> 5    NSG 593 3432.78
#> 6    NSG 557 2889.23

Prop_fem_results <- Estimate_proportion_female(Wt_data,
                                               Weight =  "Tot.Wt",
                                               max.weight = 13500,
                                               bin.width = 500,
                                               Region = "Region")
```

The returned results are the weight bin number (not the weight of the bin), sample size, proportion of females in each bin and the multinomial variance of the bins. Here `Region` is specified so these values are grouped according.

The proportions of females in each weight bin and their standard deviations (calculated from the variances) demonstrate how this data is not normally distributed. Therefore, this is now explicitly included in the biomass estimates along with much more precise estimates of variance.

```
head(Prop_fem_results)
#> # A tibble: 6 x 5
#> # Groups:   Region [4]
#>   Region Wt_bin     n   Prop Prop_var
#>   <fct>   <int> <dbl>  <dbl>    <dbl>
#> 1 NGSV        1     8 0.0392 0.000185
#> 2 NSG         1     6 0.0392 0.000246
#> 3 SGSV        1     3 0.024  0.000187
#> 4 SSG         1    14 0.194  0.00218
#> 5 NGSV        2    25 0.123  0.000527
#> 6 NSG         2    44 0.288  0.00134

ggplot(Prop_fem_results,
       aes(Wt_bin, Prop, ymin = Prop-sqrt(Prop_var), ymax = Prop + sqrt(Prop_var)))+
  geom_col()+
  geom_errorbar()+
  facet_wrap(~Region)+
  theme_bw()
```



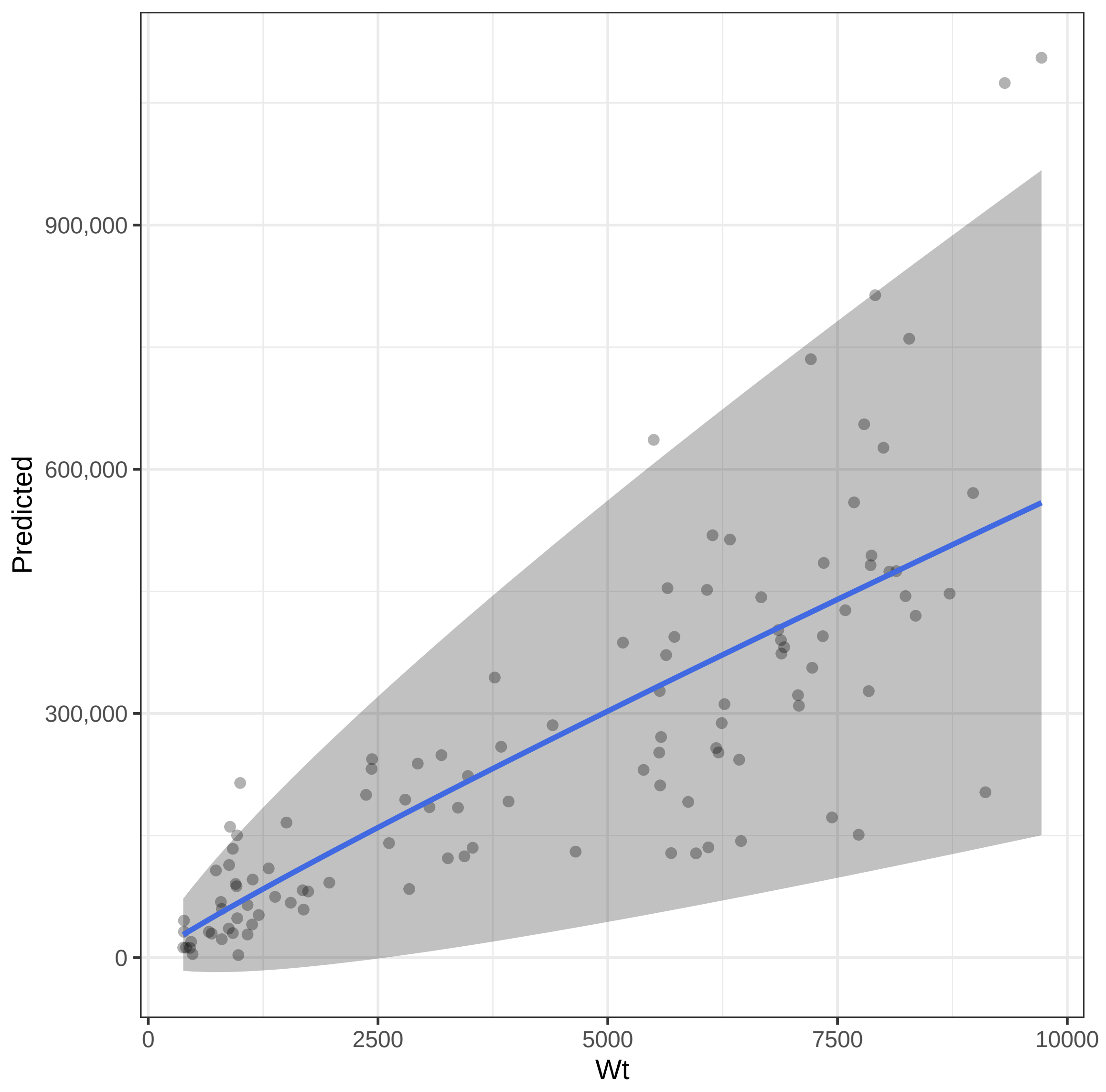
Fecundity-at-weight

As weight is no longer expressed as a mean and variance, fecundity must follow suit. Therefore, the fecundity at each weight bin must now be calculated using the the `Estimate_Batch_Fecundity()` function. This function is quite useful as it can return three sets of outputs:

- A data.frame of parameters and their variances for the relationship (these can also be printed to the screen) by setting `return.parameters = TRUE`.
- A dataframe with the predicted fecundity for each weight along with SD and 95% confidence intervals
- A dataframe with the predicted fecundity at specified weights and their variance. By providing the mid-points of weight bins, this will return the fecundity-at-weight for each bin and becomes the new inputs for estimating biomass.

The batch fecundity relationship estimator uses an allometric relationship which allows for wider variance with larger weights. Therefore, there are four parameters returned, alpha, beta and two sigma parameters that determine how variance changes with weight. A dataframe with two columns must be provided that includes **Total weight in grams** and the number of eggs for that fish. The function will determine which is which based on their scales (number of eggs > Wt in grams). A set of starting parameters are required and are provided to the `start_pars` argument as a list. The estimated parameters are printed to the screen if `verbose = TRUE`.

```
head(fecundity_data_TotWT)
#>   Total_Wt batch_fecund
#> 1     5725     394035.7
#> 2     7080     309483.9
#> 3     7870     494000.0
#> 4     5635     371710.5
#> 5     8000     626406.2
#> 6     8975     570789.5

# List of starting parameters for the batch fecundity estimator
parameters <- list( alpha= 110, beta = 0.85 , Sigma0 = 10.5, Sigma1 = 0.7)

# Return parameters
Estimate_Batch_Fecundity(fecundity_data_TotWT, start_pars = parameters, return.parameters = TRUE)
#>   Parameter         Val         var
#> 1     alpha 118.1105923 57.97564446
#> 2      beta   0.9215988  0.05831767
#> 3    Sigma0  11.0284006 12.06341963
#> 4    Sigma1   0.7440693  0.08895222

# Return batch fecundity estimates
Batch_fecundity_relationship <- Estimate_Batch_Fecundity(fecundity_data_TotWT, start_pars = parameters,

#plot of relationship
ggplot(Batch_fecundity_relationship, aes(Wt, y = Predicted, ymin = low, ymax  = upp))+
  geom_ribbon( alpha = .3)+
  scale_y_continuous(labels = scales::comma)+
  geom_point(aes(y = Fecundity), alpha = .3)+
  geom_line(aes(y = Predicted), col = "royalblue", size = 1) +
  theme_bw()
```



In order to determine the fecundity for each weight bin, the mid points of those bins are provided to the `predition.int` argument as a vector. A dataframe is then returned with the weight bin mid point, the estimate of fecundity at that weight and its variance.

```
# The mid points of each weight bin need to be used to estimate fecundity.
mid_points <- seq(250, 13500, 500)


# Estimates for biomass calculation
fecundity_at_Wtbins <- Estimate_Batch_Fecundity(fecundity_data_TotWT, start_pars = parameters, predicti

head(fecundity_at_Wtbins)
#>     Wt Fecundity        Var
#> 1  250  19152.59  286796137
#> 2  750  52715.92 1293985026
#> 3 1250  84410.67 2607278612
#> 4 1750 115098.27 4136158659
#> 5 2250 145096.25 5838283824
#> 6 2750 174571.64 7687918584
```

Spawning Area (*A*)

Spawning area is the only DEPM parameter that is not estimated in this package as this is typically done using GIS methods. However, the Spawning area object can be created manually by taking another parameter object (*P0, S* or *R* are good choices) and using its `Time/Region` groupings to create new object. As this example has one survey from 4 regions in a single year, only the `Region` variable needs to be carried across. A new variable can then be added with the location in the correct order in that column. Spawning Area is a precise quantity and therefore does not require a variance. Note that spawning area must be in metres squared.

```
# Get Time/region combos from the P0 object
Area <- select(P0_results, Region)

Area$A <- c(2822201039, #NGSV
            1884895030, # NSG
            2285649237, # SGSV
            3239913469) # SSG

head(Area)
#> # A tibble: 4 x 2
#>   Region          A
#>   <fct>       <dbl>
#> 1 NGSV   2822201039
#> 2 NSG    1884895030
#> 3 SGSV   2285649237
#> 4 SSG    3239913469
```

Combining parameters as inputs for biomass estimation

With all of the parameters now available to calculate spawning stock biomass, these now need to be combined into a dataset that can be input into the `Estimate_DEPMWt_biomass()` function. For the DEPMWt approach, three objects are needed:

Combining non-weight bin parameter estimates

A dataframe of all of the parameter estimates can be assembled using `combine_estimates()` where each of the parameter objects are provided as arguments.

```
Adults_pars <- combine_estimates(P0 = P0_results, R = sex_ratio_results, S = Spawn_results, A = Area)
#> Joining, by = "Region"
#> Joining, by = "Region"
Adults_pars
#> # A tibble: 4 x 6
#>   Region    P0     Z     R          A     S
#>   <fct>  <dbl> <dbl> <dbl>      <dbl> <dbl>
#> 1 NGSV   2.03    0.4 0.448 2822201039 0.722
#> 2 NSG    0.947   0.4 0.337 1884895030 0.722
#> 3 SGSV   0.527   0.4 0.394 2285649237 0.722
#> 4 SSG    0.725   0.4 0.574 3239913469 0.722
```

Where a parameter is estimated as time invariant, it will be automatically allocated to all of the `Region` or `Time` groupings applied for the other parameters. Here this occurs for S. However, if a particular parameter is missing in one instance but is not time invariant, it will not be carried over and `NA` will be returned. In this situation, it will need to be inserted manually. For example, lets pretend that R is missing for "SSG":

```
Adults_pars[Adults_pars$Region == "SSG", "R"] <- 0.5741385
```

Combining non-weight bin parameter variances

Variance estimates are needed by `Estimate_DEPMWt_biomass()` in order to determine the precision of the biomass estimates. These are provided in the same fashion as the parameter estimates using `combine_variances()`.

```
Adults_vars <- combine_variances(P0_results, R = sex_ratio_results, S = Spawn_results)
#> Joining, by = "Region"
Adults_vars
#> # A tibble: 4 x 5
#>   Region     P0     Z        R       S
#>   <fct>   <dbl> <dbl>    <dbl>   <dbl>
#> 1 NGSV   0.391    0.4 0.00952  0.00298
#> 2 NSG    0.218    0.4 0.000714 0.00298
#> 3 SGSV   0.129    0.4 0.00117  0.00298
#> 4 SSG    0.0849   0.4 0.00185  0.00298
```

The same manual insertions can be performed on this object:

```
Adults_vars[Adults_vars$Region == "SSG", "R"] <- 0.0018484630
```

Combining proportion female and fecundity parameters

As the proportions of females and the fecundity at each weight bin have a different structure, they have their own function to combine them into a single object. Much like `combine_estimates()` and `combine_variances()`, this function is called `combine_wt_class_estimates()` and only requires these two dataframes which will be combined according to `Region`/`Time` groupings. Manually adjusting these results is not recomended.

```
weight_class_pars <- combine_wt_class_estimates(prop.fem.data = Prop_fem_results,
                                                fecundity.data = fecundity_at_Wtbins)
head(weight_class_pars)
#> # A tibble: 6 x 8
#> # Groups:   Region [4]
#>   Region Wt_bin     n   Prop Prop_var    Wt Fecundity     Fec_var
#>   <fct>   <int> <dbl>  <dbl>    <dbl> <dbl>     <dbl>       <dbl>
#> 1 NGSV        1     8 0.0392 0.000185   250    19153.  286796137.
#> 2 NSG         1     6 0.0392 0.000246   250    19153.  286796137.
#> 3 SGSV        1     3 0.024  0.000187   250    19153.  286796137.
#> 4 SSG         1    14 0.194  0.00218    250    19153.  286796137.
#> 5 NGSV        2    25 0.123  0.000527   750    52716. 1293985026.
#> 6 NSG         2    44 0.288  0.00134    750    52716. 1293985026.
```

Spawning Stock Biomass Estimation

All of the hard work is done once all of the parameters and their variances have been estimated and organised. The final step is to combine them using `Estimate_DEPMWt_biomass()`. This function requires each of these objects and will return a list of results which include the spawning stock biomass, total number of females and the number of females in each weight bin.

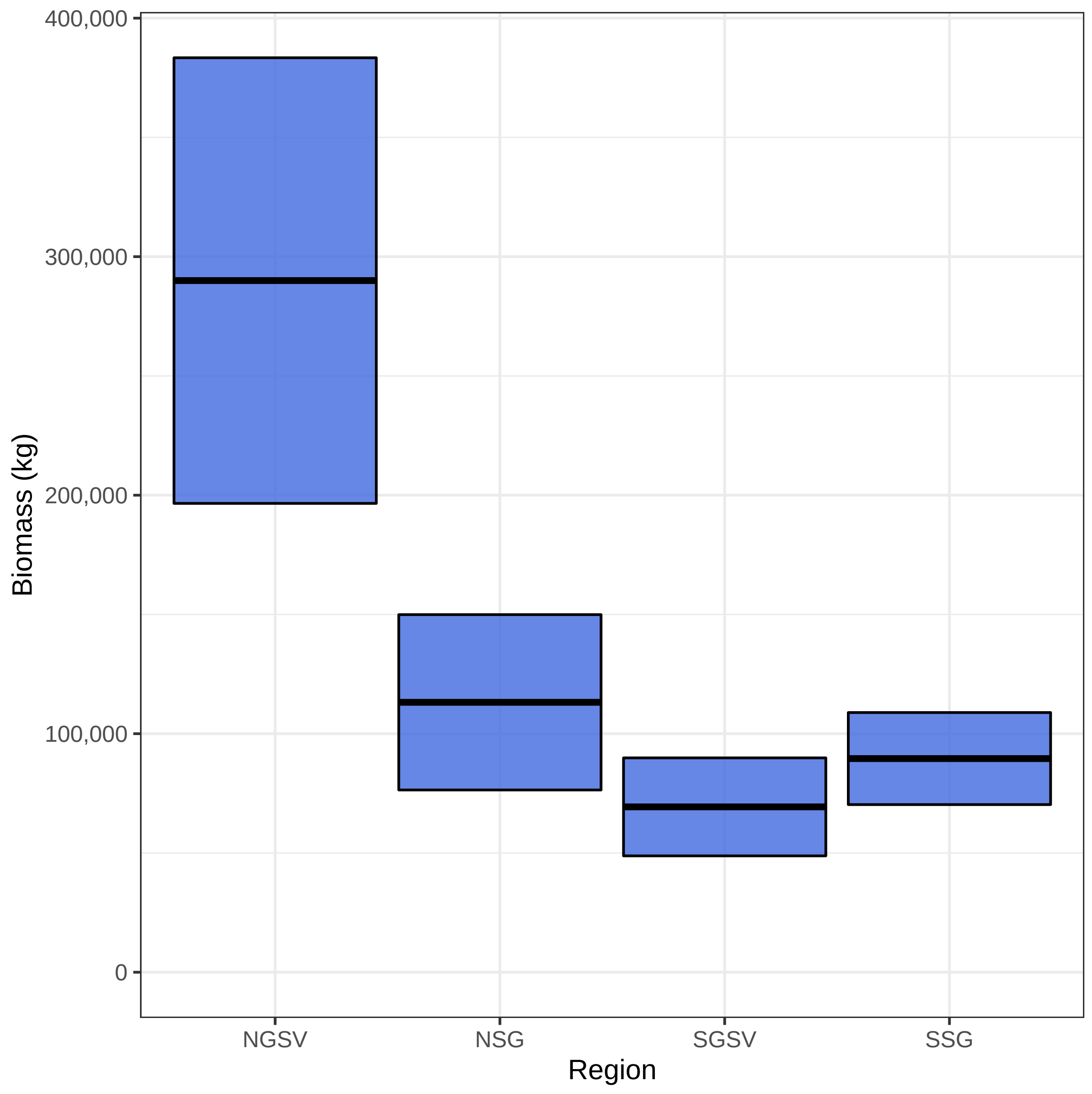
```
# Estimate Biomass
Snapper_biomass_results <- Estimate_DEPMWt_biomass(adult.pars = Adults_pars,
                                                   adult.vars = Adults_vars,
                                                   weight.pars.vars = weight_class_pars)
# a list of three outputs is returned
Snapper_biomass_results$Biomass # biomass in kgs
#>   Region   Z   Biomass       SD
#> 1   NGSV 0.4 289973.37 93410.85
#> 2    NSG 0.4 113168.58 36749.77
#> 3   SGSV 0.4  69327.79 20539.80
#> 4    SSG 0.4  89575.41 19295.47
Snapper_biomass_results$Nfem # number of females
#>   Region   Z      Nfem       SD
#> 1   NGSV 0.4 36517.189 9146.748
#> 2    NSG 0.4 22204.413 7290.200
#> 3   SGSV 0.4  7322.528 2202.955
#> 4    SSG 0.4 25537.006 6454.136
head(Snapper_biomass_results$NfemWt) # number of females in each weight bin
#>   Region   Z Nwt    NfemWt        SD
#> 1   NGSV 0.4   1 1432.0466  610.9450
#> 2    NSG 0.4   1  870.7613  448.8982
#> 3   SGSV 0.4   1  175.7407  113.1486
#> 4    SSG 0.4   1 4965.5289 1706.0851
#> 5   NGSV 0.4   2 4475.1458 1384.7827
#> 6    NSG 0.4   2 6385.5828 2208.1164
```

The list of dataframes returned from the function can then be saved and used for plotting

```
# Create new objects
Snapper_biomass <- Snapper_biomass_results$Biomass
Snapper_Nfem <- Snapper_biomass_results$Nfem
Snapper_NfemWt <- Snapper_biomass_results$NfemWt
```

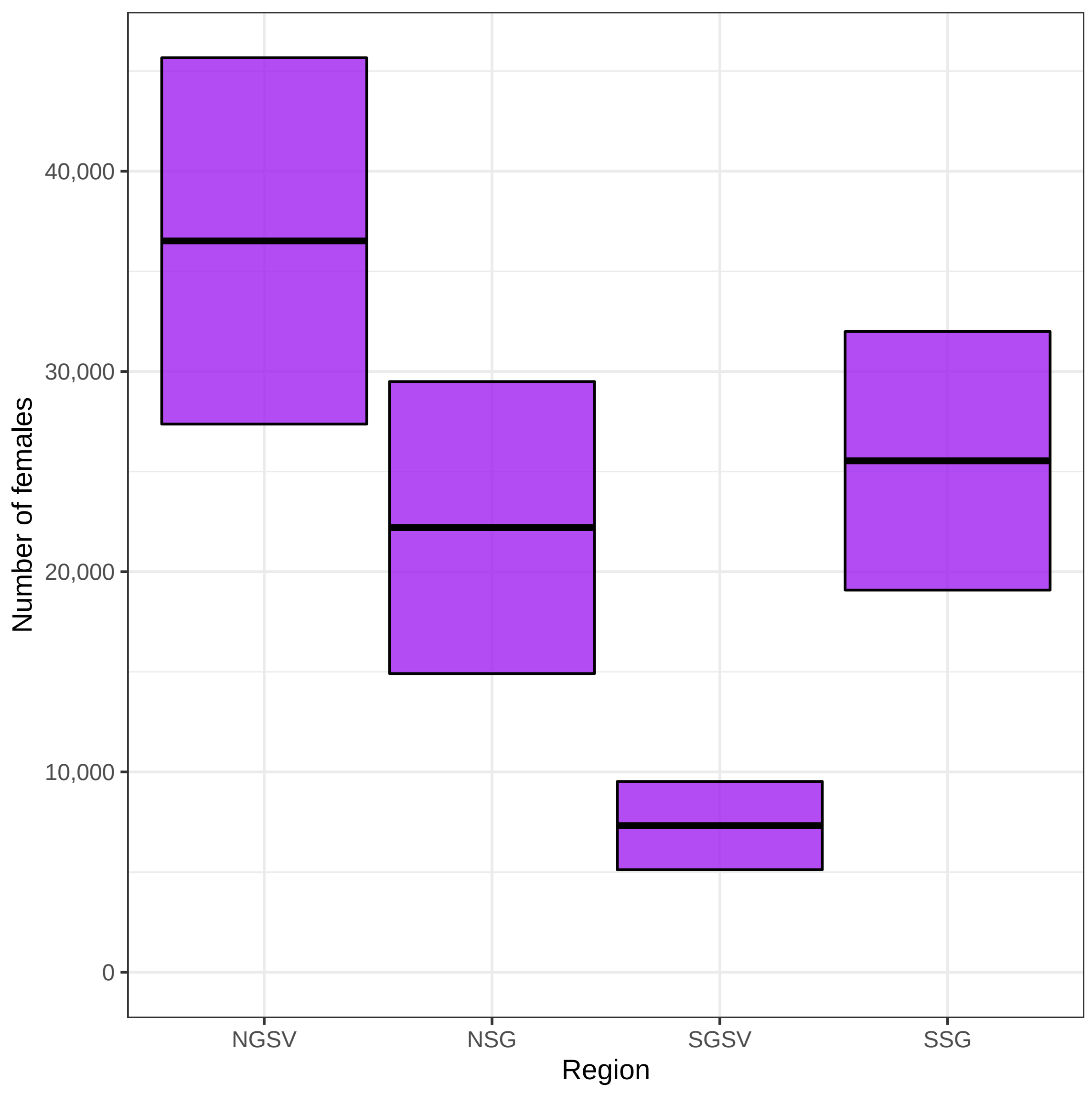
Plot biomass with 95% CIs

```
ggplot(Snapper_biomass, aes(Region,Biomass,
                            ymin = Biomass - SD,
                            ymax = Biomass + SD))+
  geom_crossbar(fill = "royalblue", alpha = .8)+
  expand_limits(y = 0)+
  scale_y_continuous(name = "Biomass (kg)", labels = scales::comma, breaks = scales::pretty_breaks(5))+
  theme_bw()
```



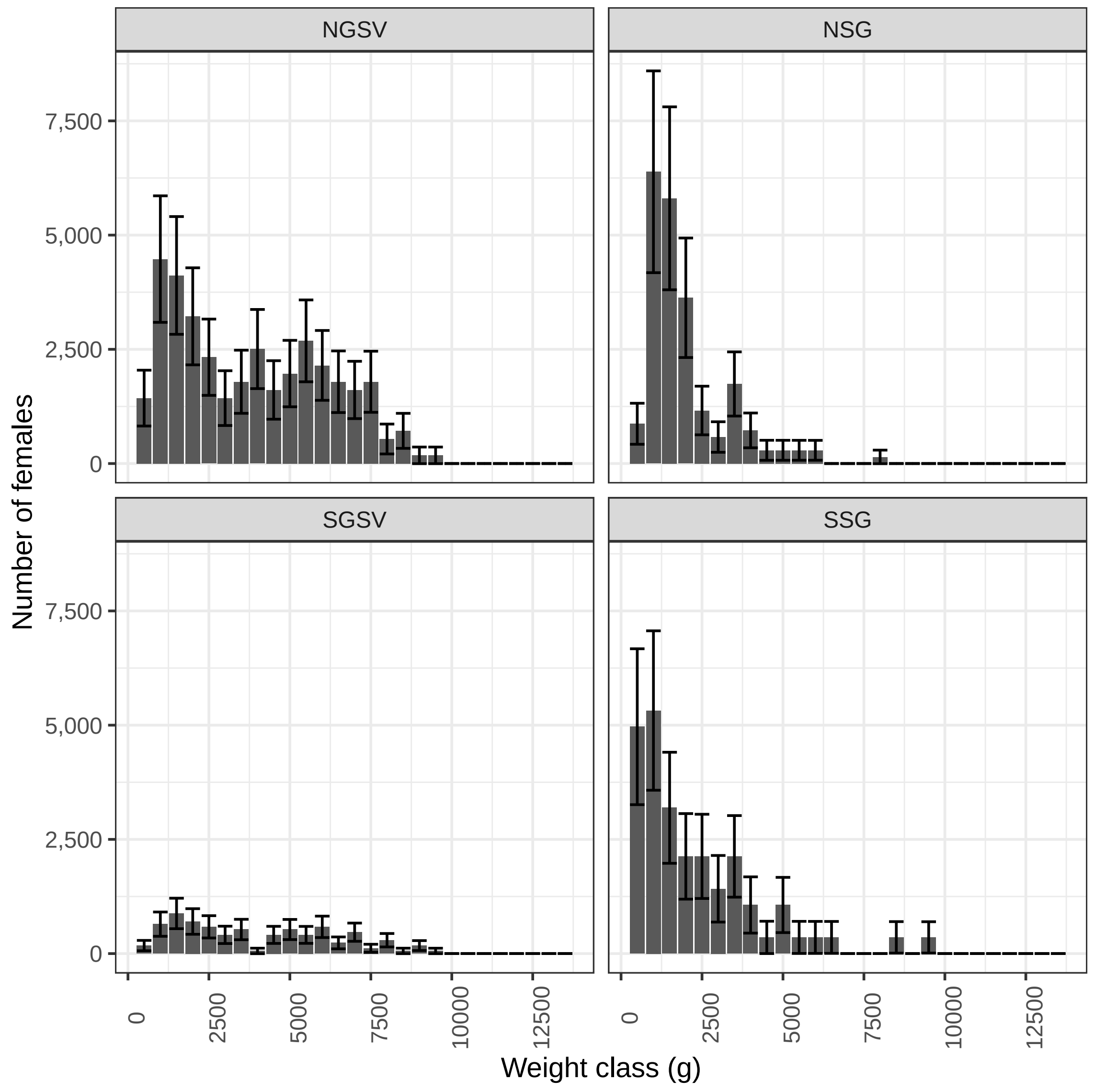
Plot total number of females

```
ggplot(Snapper_Nfem, aes(Region,Nfem,
                         ymin = Nfem - SD,
                         ymax = Nfem + SD))+
  geom_crossbar(fill = "purple", alpha = .8)+
  expand_limits(y = 0)+
  scale_y_continuous(name = "Number of females", labels = scales::comma, breaks = scales::pretty_breaks
  theme_bw()
```



Plot number of females in each weight bin

```
#plot female weights
ggplot(Snapper_NfemWt,
       aes(Nwt,NfemWt, ymin = NfemWt - SD,
                                                      ymax = NfemWt + SD))+
  facet_wrap(~Region, ncol = 2)+
  geom_col()+
  geom_errorbar()+
  expand_limits(y = 0)+
  scale_y_continuous(name = "Number of females", labels = scales::comma)+
  scale_x_continuous(name = "Weight class (g)",
                     breaks = scales::pretty_breaks(6),
                     labels = function(x){x * 500})+
  theme_bw()+
  theme(axis.text.x = element_text(angle = 90))
```



Testing assumptions for pre-specified egg mortality (Z)

As the P_0 estimation methods in this package require a specified level of egg mortality (Z), it is sensible to test the sensitivities of the P_0 estimates, as well as the DEPMWt results to different Z values. Testing different values of Z can be done by passing a vector of values to the `z` argument of `Estimate_P0()`:

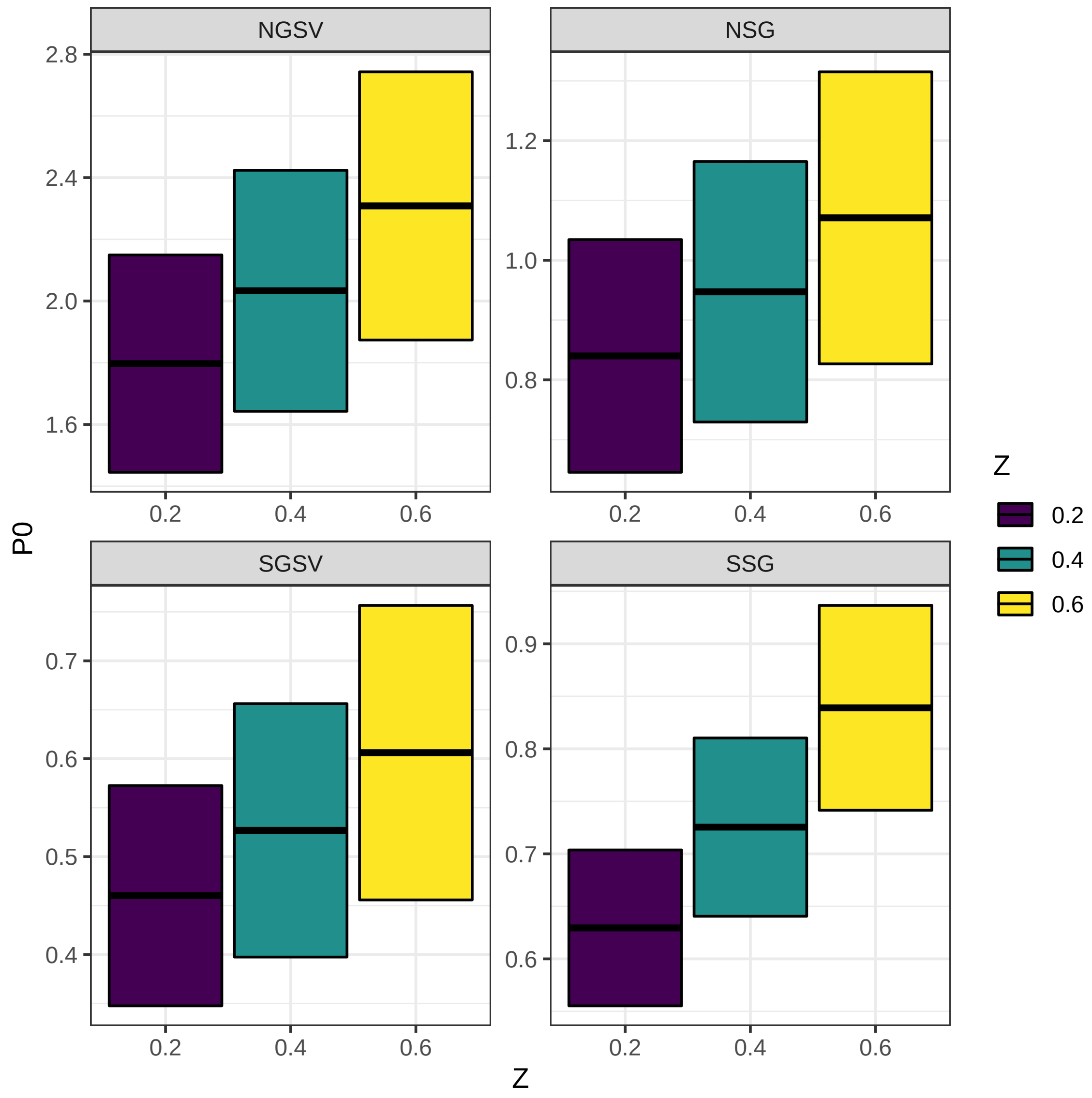
```
Multi_Z_P0_results <- Estimate_P0(data = egg_data,
                                  site = "Site",
                                  Region = "Region",
                                  Z = c(.2, .4, .6))
```

This will return a different P_0 estimate for each value of Z for each `Time`/`Region` grouping:

```
Multi_Z_P0_results
#>    Region        P0      P0_se   Z
#> 1    NGSV 1.7972317 0.35215150 0.2
#> 2     NSG 0.8400430 0.19454626 0.2
#> 3    SGSV 0.4600852 0.11248190 0.2
#> 4     SSG 0.6294788 0.07419926 0.2
#> 5    NGSV 2.0333120 0.39057148 0.4
#> 6     NSG 0.9472290 0.21775974 0.4
#> 7    SGSV 0.5268231 0.12938292 0.4
#> 8     SSG 0.7254297 0.08486910 0.4
#> 9    NGSV 2.3084189 0.43464390 0.6
#> 10    NSG 1.0709549 0.24423117 0.6
#> 11   SGSV 0.6061931 0.15043110 0.6
#> 12    SSG 0.8390163 0.09755602 0.6
```

For South Australian Snapper, the resulting P_0 estimates are very similar.

```
Multi_Z_P0_results$Z <- as.factor(Multi_Z_P0_results$Z) # convert Z to a factor for easy plotting

ggplot(Multi_Z_P0_results, aes(Z, P0, ymin = P0 - P0_se, ymax = P0 + P0_se, fill = Z))+
  geom_crossbar()+
  facet_wrap(~Region, ncol = 2, scales = "free")+
  scale_fill_viridis_d()+
  theme_bw()
```



An object from `Estimate_P0()` with multiple Zs can be included in the biomass estimation procedure in the same manner as before. When multiple Zs are provided to `combine_estimates()` and `combine_variances()` they are treated as groupings and all other parameters will be aligned with them. When creating an object for spawning area A, care must be taken to only have unique `Time`/`Region` groupings. Using an object from `Estimate_P0()` with multiple Zs can erroneously add duplicates if you're not careful. This will cause error messages.

```
# Make sure that Area has unique Time/Region breakdowns (Only Region used in this example)
Area <- data.frame(Region = unique(Multi_Z_P0_results$Region))

Area$A <- c(2822201039, #NGSV
            1884895030, # NSG
            2285649237, # SGSV
            3239913469) # SSG



Multi_Z_adult_pars <- combine_estimates(P0 = Multi_Z_P0_results, R = sex_ratio_results, S = Spawn_resul
#> Joining, by = "Region"
#> Joining, by = "Region"

Multi_Z_adult_vars <- combine_variances(P0 = Multi_Z_P0_results, R = sex_ratio_results, S = Spawn_resul
#> Joining, by = "Region"
```

A new object with Z as a grouping will now be produced which can be passed to the biomass estimation function.

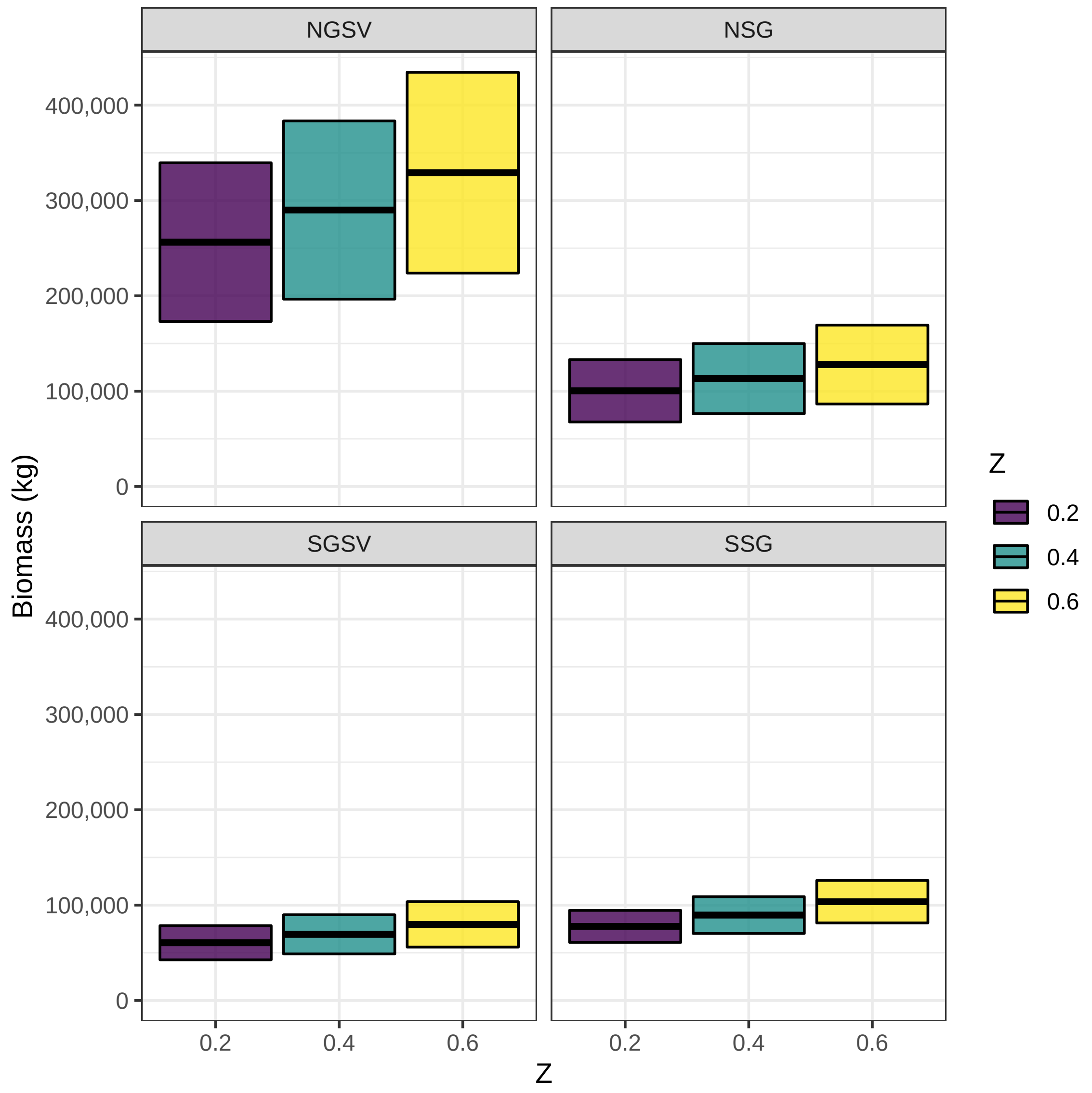
```
Multi_Z_adult_pars
#>    Region        P0   Z         R          A        S
#> 1    NGSV 1.7972317 0.2 0.4478643 2822201039 0.722488
#> 2     NSG 0.8400430 0.2 0.3369494 1884895030 0.722488
#> 3    SGSV 0.4600852 0.2 0.3939694 2285649237 0.722488
#> 4     SSG 0.6294788 0.2 0.5741385 3239913469 0.722488
#> 5    NGSV 2.0333120 0.4 0.4478643 2822201039 0.722488
#> 6     NSG 0.9472290 0.4 0.3369494 1884895030 0.722488
#> 7    SGSV 0.5268231 0.4 0.3939694 2285649237 0.722488
#> 8     SSG 0.7254297 0.4 0.5741385 3239913469 0.722488
#> 9    NGSV 2.3084189 0.6 0.4478643 2822201039 0.722488
#> 10    NSG 1.0709549 0.6 0.3369494 1884895030 0.722488
#> 11   SGSV 0.6061931 0.6 0.3939694 2285649237 0.722488
#> 12    SSG 0.8390163 0.6 0.5741385 3239913469 0.722488
```

The `weight_class_pars` object created earlier does not need to incorporate the new Zs and can be provided straight to the function.

```
# Estimate Biomass
Snapper_biomass_results <- Estimate_DEPMWt_biomass(adult.pars = Multi_Z_adult_pars,
                                                  adult.vars = Multi_Z_adult_vars,
                                                  weight.pars.vars = weight_class_pars)



# Create new objects
Snapper_biomass <- Snapper_biomass_results$Biomass
Snapper_Nfem <- Snapper_biomass_results$Nfem
Snapper_NfemWt <- Snapper_biomass_results$NfemWt
```

These results can then be examined to determine how biomass is influenced by the specification of Z.

```


Snapper_biomass$Z <- as.factor(Snapper_biomass$Z) # convert Z to a factor for easy plotting

ggplot(Snapper_biomass, aes(Z,Biomass, fill = Z,
                           ymin = Biomass - SD,
                           ymax = Biomass + SD))+
  geom_crossbar( alpha = .8)+
  facet_wrap(~Region, ncol =2)+
  scale_fill_viridis_d()+
  expand_limits(y = 0)+
  scale_y_continuous(name = "Biomass (kg)", labels = scales::comma, breaks = scales::pretty_breaks(5))+
  theme_bw()
```



References

McGarvey, R., Steer, M. A., Smart, J. J., Matthews, D. J. and Matthews, J. M.(in review). Generalizing the Parker model equation of DEPM: incorporating size dependence of population numbers and batch fecundity

McGarvey, R., Steer, M.A., Matthews, J.M., Ward, T.M. (2018). A stage-based estimator of daily egg production. ICES J. Mar. Sci. 75(5), 1638-1646.

Parker, K. 1980. A direct method for estimating northern anchovy, Engraulis mordax, spawning biomass. . FIsheries Bulletin 78:541-544.