Case Study 2: Ellner, Childs & Rees 2016

A more complicated, age and size structured model

Many life cycles cannot be described by a single, continuous state variable. For example, some plants are best modeled using height or diameter at breast height (DBH) as a state variable, and may also form a seed bank. Seeds in the seed bank can't have a value for height or DBH, but may lose their viability as they age. Thus, we require a discrete state to capture the dynamics of the seed bank. Models that include discrete states and/or multiple continuous state variables are *general IPMs*.

Many species experience some form of senesence, where survival increases and fecundity decreases after a certain age is attained. Age may interact with a measure of size, for example body mass, resulting in neither single variable reliably predicting demography on its own. Data sets where individuals are cross-classified by age and size represent an interesting opportunity for demographic research. Unfortunately, they can be a bit more complicated to model. This case study will use an age and size structured population to illustrate how to implement general IPMs in `ipmr`.

The model itself can be written on paper as:

1. $n_0(z', t+1) = \sum_{a=0}^{M} \int_L^U F_a(z', z) n_a(z, t) dz$
2. $n_a(z', t+1) = \int_L^U P_{a-1}(z', z) n_{a-1}(z, t) dz$ a = 1,2,. . . , M-1

In this case, there is also a maximum age class M defined as $a > 20$. We need to define one more equation to indicate the number of individuals in that age group.

3. $n_M(z', t+1) = \int_L^U [P_M(z', z) n_M(z, t) + P_{M-1}(z', z) n_{M-1}(z, t)] dz$

The sub-kernel $P_a(z', z)$ is comprised of the following functions:

4. $P_a(z', z) = S(z, a) * G(z', z, a)$
5. $Logit^{-1}(S(z, a)) = \alpha_s + \beta_s^z * z + \beta_s^a * a$
6. $G(z', z, a) \sim Norm(\mu_g(z, a), \sigma_g)$
7. $\mu_g(z) = \alpha_g + \beta_g^z * z + \beta_g^a * a$

and the sub-kernel $F_a(z',z) is comprised of the following functions:

8. $F_0 = 0$
9. $F_a = S(z, a) * p_b(z, a) * p_r(a) * f_d(z', z) * 0.5$

The F_a kernel is multiplied by 0.5 because this model only models the population dynamics of females.

10. $Logit^{-1}(p_b(z, a)) = \alpha_{p_b} + \beta_{p_b}^z * z + \beta_{p_b}^a * a$
11. $Logit^{-1}(p_r(a)) = \alpha_{p_r} + \beta_{p_r}^a * a$
12. $f_d(z', z) \sim Norm(\mu_{f_d}(z), \sigma_{f_d})$
13. $\mu_{f_d}(z) = \alpha_{f_d} + \beta_{f_d}^z * z$

Equations 5, 7, 10, 11, and 13 are parameterized from regression models, and equations 6 and 12 are constant parameters derived from observed data. This example will skip the step of model parameterization, and go straight into the IPM.

First, we will define the model parameters and function for the F_a kernels.

```
library(ipmr)

# Set parameter values and names

param_list <- list(
  ## Survival
  surv_int  = -1.70e+1,
  surv_z    =  6.68e+0,
  surv_a    = -3.34e-1,
  ## growth
  grow_int  =  1.27e+0,
  grow_z    =  6.12e-1,
  grow_a    = -7.24e-3,
  grow_sd   =  7.87e-2,
  ## reproduce or not
  repr_int  = -7.88e+0,
  repr_z    =  3.11e+0,
  repr_a    = -7.80e-2,
  ## recruit or not
  recr_int  =  1.11e+0,
  recr_a    =  1.84e-1,
  ## recruit size
  rcsz_int  =  3.62e-1,
  rcsz_z    =  7.09e-1,
  rcsz_sd   =  1.59e-1
)

# define a custom function to handle the F kernels. We could write a rather
# verbose if(age == 0) {0} else {other_math} in the define_kernel(), but that
# might look ugly. Note that we CANNOT use ifelse(), as its output is the same
# same length as its input (in this case, it would return 1 value, not 10000 values).

f_fun <- function(age, s_age, pb_age, pr_age, recr) {

  if(age == 0) return(0)

  s_age * pb_age * pr_age * recr * 0.5

}
```

Next, we set up the P_a kernels (Equations 4-7 above). Because this is a general, deterministic IPM, we set the model class to `"general_di_det"`, and set `has_age = TRUE` to indicate that our model has age structure as well as size structure. There are 3 k ey things to note:

1. the use of the suffix `_age` appended to the names of the `"P"` kernel and the `mu_g_age` variable
2. the value `age` used in the vital rate expressions
3. the list in the `levels_ages` argument

The values in the `levels_ages` list will automatically get substituted in for `"age"` each time it appears in the vital rate expressions and kernels. This single call to `define_kernel()` will result in 22 actual kernels, one for each value of `age` from 0-21.

The `plogis` function is part of the `stats` package in *R*, and performs the inverse logit transformation.

```
# Implement the age x size model

age_size_ipm <- init_ipm("general_di_det", has_age = TRUE) %>%
  define_kernel(
    name          = "P_age",
    family        = "CC",
    formula       = s_age * g_age * d_z,
    s_age         = plogis(surv_int + surv_z * z_1 + surv_a * age),
    g_age         = dnorm(z_2, mu_g_age, grow_sd),
    mu_g_age      = grow_int + grow_z * z_1 + grow_a * age,
    data_list     = param_list,
    states        = list(c("z")),
    has_hier_effs = FALSE,
    levels_ages   = list(age = c(0:20), max_age = 21),
    evict_cor     = FALSE
  )
```

The F_a kernel (equations 8-13) will follow a similar pattern - we append a suffix to the `name` paramter, and then make sure that our functions also include `_age` suffixes and `age` values where they need to appear.

```
age_size_ipm <-   define_kernel(
  proto_ipm     = age_size_ipm,
  name          = "F_age",
  family        = "CC",
  formula       = f_fun(age, s_age, pb_age, pr_age, recr) * d_z,
  s_age         = plogis(surv_int + surv_z * z_1 + surv_a * age),
  pb_age        = plogis(repr_int + repr_z * z_1 + repr_a * age),
  pr_age        = plogis(recr_int + recr_a * age),
  recr          = dnorm(z_2, rcsz_mu, rcsz_sd),
  rcsz_mu       = rcsz_int + rcsz_z * z_1,
  data_list     = param_list,
  states        = list(c("z")),
  has_hier_effs = FALSE,
  levels_ages   = list(age = c(0:20), max_age = 21),
  evict_cor     = FALSE
)
```

Once we've defined the P_a and F_a kernels, we need to define how they interact with age-specific size distributions to produce size distributions at the next time step. This corresponds to equations 1-3 above. There a couple things to note here:

1. We do not add `"_age"` to the `name` here. This is because we do not want to expand this into 22 separate iteration procedures - we only want 1 that iterates the 22 separate P_a and F_a kernels together.
2. The `all_ages` function indicates that we want to expand this one expression to include all ages in our list in a *single* expression, which is then collapsed using the `fun` argument. Since we want the sum of the integrals, we use `"+"`. Note that we most likely never want to use the `sum()` function instead of the `+` function, as `sum()` returns a single number while `+` is vectorized over its arguments.
3. The insertion of `age_minus_1` in the `n_z_age_t_1` expression. `ipmr` automatically generates the correct indices using these markers, so there is no need to worry about indexing correctly. Tragically, we could not implement a literal `age - 1`. `_plus_` is also available for models with different formats. The amount to add/subtract is not limited to 1, though it is hard to think of examples where we would want to index by larger increments (perhaps with censuses every 2 or 3 years instead annual?).
4. the `max_age` in `n_z_max_age_t_1`. This corresponds to equation 3. We can adjust the maximum age value in the model by changing the `max_age` slot in the `levels_ages` list.

```
age_size_ipm <- define_k(
  proto_ipm       = age_size_ipm,
  name            = "K",
  family          = "IPM",
  n_z_0_t_1       = all_ages(expr = F_age %*% n_z_age_t, fun = "+"),
  n_z_age_t_1     = P_age_minus_1 %*% n_z_age_minus_1_t,
  n_z_max_age_t_1 = P_max_age %*% n_z_max_age_t +
                    P_max_age_minus_1 %*% n_z_max_age_minus_1_t,
  data_list       = param_list,
  states          = list (c("z")),
  has_hier_effs   = FALSE,
  levels_ages     = list(age = c(0:20), max_age = 21),
  evict_cor       = FALSE
  )
```

The rest of the model definition code looks like the first case study. We define the implementation arguments using `define_impl()`, the domains using `define_domains()`, and the initial population state using `define_pop_state()`. We can compute the populations asymptotic growth rate, and stable size distributions for each age using the `lambda()` and `right_ev()` functions.

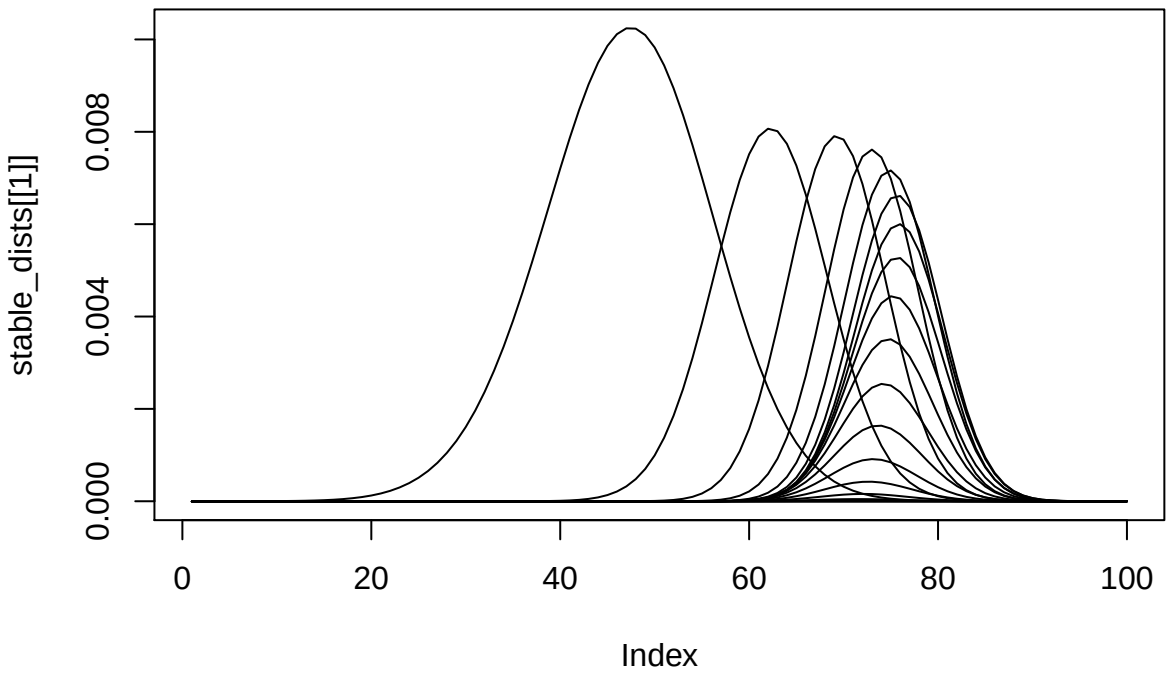
```
age_size_ipm <-  define_impl(
  proto_ipm = age_size_ipm,
  make_impl_args_list(
    kernel_names = c("P_age", "F_age", "K"),
    int_rule     = rep("midpoint", 3),
    dom_start    = rep("z", 3),
    dom_end      = rep("z", 3)
  )
) %>%
  define_domains(
    z = c(1.6, 3.7, 100)
  ) %>%
  define_pop_state(
    pop_vectors = list(
      n_z_age = runif(100))
  ) %>%
  make_ipm(
    usr_funs = list(f_fun = f_fun),
    iterate  = TRUE,
    iterations = 100
  )


lamb         <- lambda(age_size_ipm)
stable_dists <- right_ev(age_size_ipm)
```

We can plot the stable distributions using an `lapply()` on the `stable_dists` object.

```
plot(stable_dists[[1]], type = 'l')

lapply(stable_dists[2:22], function(x) lines(x))
```




```
## $z_1_w
## NULL
##
## $z_2_w
## NULL
##
## $z_3_w
## NULL
##
## $z_4_w
## NULL
##
## $z_5_w
## NULL
##
## $z_6_w
## NULL
##
## $z_7_w
## NULL
##
## $z_8_w
## NULL
##
## $z_9_w
## NULL
```

```
##
## $z_10_w
## NULL
##
## $z_11_w
## NULL
##
## $z_12_w
## NULL
##
## $z_13_w
## NULL
##
## $z_14_w
## NULL
##
## $z_15_w
## NULL
##
## $z_16_w
## NULL
##
## $z_17_w
## NULL
##
## $z_18_w
## NULL
##
## $z_19_w
## NULL
##
## $z_20_w
## NULL
##
## $z_21_w
## NULL
```