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 as they age. 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. The parameter values are taken from Ellner, Childs & Rees, Chapter 6 (2016). These can be found here: https://github.com/levisc8/first-edition/blob/master/Rcode/c6/Ungulate%20Age%20Demog%20Funs.R#L13. In the code from the book, these paramters were used to simulate an

IBM to generate a data set. The data were then used to fit regression models and an age×size IPM. We are going to skip those steps and just use the "true" parameter estimates to generate the IPM.

First, we will define all the model parameters and a 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 number, not 10000
# numbers).

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 key things to note:

1. the use of the suffix `_age` appended to the names of the `"P_age"` 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. For general IPMs that are not age-structured, we would use `has_hier_effs` and `levels_hier_effs` in the same way we're using `age` below.

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

```
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 starting and ending states for each kernel. Age-size structured populations will look a little different from the previous ones, as we need to ensure that all fecundity kernels produce age-0 individuals, and all survival-growth kernels produce age+1 individuals. We define the implementation arguments using `define_impl()`, and set each kernel's `state_start` to `"z_age"`. Because the fecundity kernel produces age-0 individuals, regardless of the starting age, its `state_end` is `"z_0"`.

```
age_size_ipm <-  define_impl(
  proto_ipm = age_size_ipm,
  make_impl_args_list(
    kernel_names = c("P_age", "F_age"),
    int_rule     = rep("midpoint", 2),
    state_start    = c("z_age", "z_age"),
    state_end      = c("z_age", "z_0")
  )
```

```
)
```

The rest of the model definition pipeline should look similar to Case Study 1. 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 <- age_size_ipm %>%
  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)
lamb
```

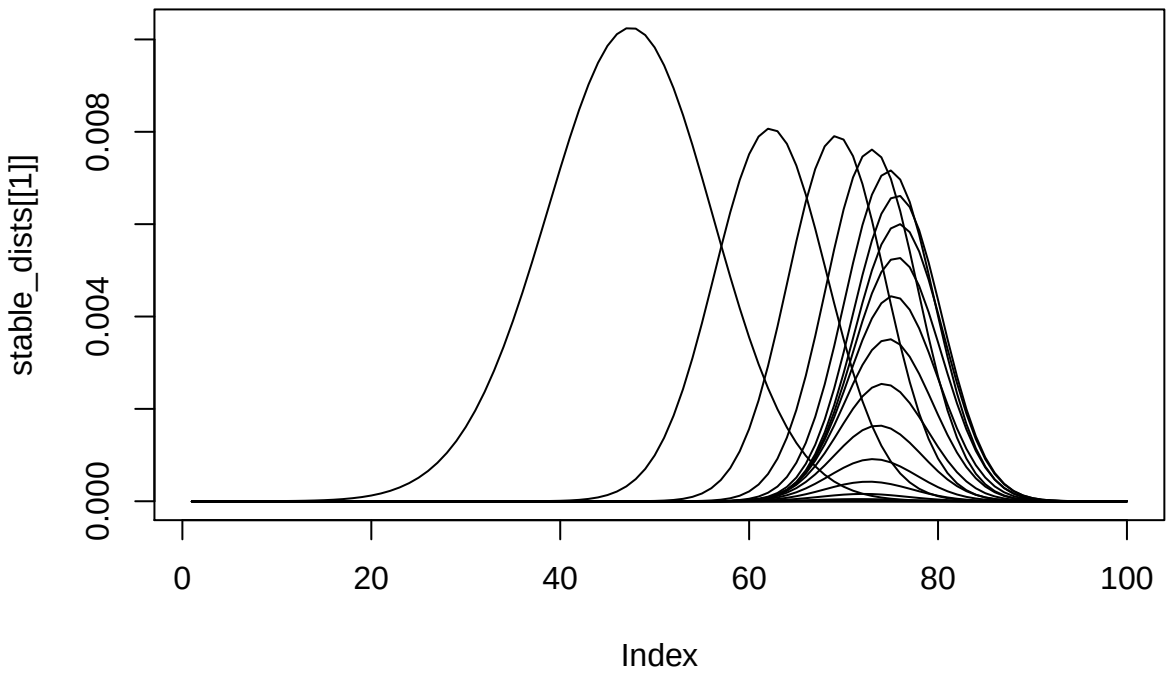
```
## [1] 1.014833
```

```
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')

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


stable_dists[[1]]
0.000
0.004
0.008
0
20
40
60
80
100
Index