Model expressions in PADRINO

Sam Levin

Last updated: 2021-04-05

Contents

Overview

PADRINO aims to store the mathematical forms of IPMs as text strings. This means that we need to be able to write out the IPM on paper/$LaTeX$/somewhere before we can really know how to enter it. It is important to have a basic understanding of how to write out functional forms for the mean response of a GLM. If you are not already familiar with these, please let speak to either Rob or Sam ASAP so we can get you up to speed!

Models stored in PADRINO are rebuilt using ipmr. It will also be helpful to read that project's homepage, as well as the first two articles on simple and general IPMs. These readings will give you a much better understanding of how this package handles different vital rate expressions and kernel formulae. Since PADRINO depends on this package to rebuild models, the syntax it uses will look very similar to what should go into PADRINO (with a couple exceptions, namely when digitizing probability density functions).

Finally, `ipmr` is designed to generate all model outputs via iteration, rather than generating large kernels and then using linear algebra methods to derive the relevant metrics. Therefore, we don't actually need to digitize the $K(z', z)$ iteration kernel, only the sub-kernels that comprise the model (e.g. $P(z', z), F(z', z), C(z', z)$). However, the notation in many publications reports the iteration kernel, rather than the model iteration. Therefore, it will sometimes be necessary to re-write the published model in terms of the population state at time t and $t + 1$, rather than simply $K(z', z) = ...$ There will be an example of this below as well.

We'll start with simple models, and build complexity as we go through. Note that for the sake of realism, we're going to use examples where the notation used in the papers is not consistent with how we'd write them in PADRINO (or perhaps even with how the rest of the "foundational literature" deals with these models). This will be prominent in the first three examples used here.

Simple models

A simple IPM is an IPM that makes use of 1, and only 1, continuous state variable. It does not include any additional discrete states. These were more common in the early literature, though now more demanding analyses have required authors to create more complicated models. Models that use more than 1 continuous state, and/or include discrete states, are general models, and are detailed below.

An example model - Summary

Below is a shortened description of a simple IPM for *Lonicera maackii* from Levin et al. 2019.[1] x, y correspond to plant size at time t and $t + 1$, respectively. We will use `size_1` and `size_2` in place of x, y when writing out the model in PADRINO format (see below).

1. $s(x)$ (survival of plants) was estimated using a Bayesian logistic regression with both linear and quadratic fixed effects.
2. $g(y, x)$ (growth of surviving plants) was estimated using a linear regression of `y ~ x`. We also fit GAMs to our data to see if they produced better predictions. GAMs produced lower AIC scores, but visual inspection of the data for the competitor removal treatment showed that the GAM overfit the data. Thus, to ensure our control and competitor removal treatment IPMs were comparable, we modeled growth using the the linear model. We correct for eviction by adding evicted individuals into the smallest or largest bins after generating the growth kernel.
3. $f_p(x)$ (probability of becoming reproductive) was estimated using a logistic regression of reproductive status on size. We used a logit link and a quasi-binomial error structure to account for overdispersion in our data.

[1] We will only look at the control treatment for now

4. $f_s(x)$ (fruits per reproductive plant) was estimated using a generalized linear model with a log link and a Poisson error structure.
5. f_f (seeds per fruit) is a constant based on sampling fruits from 12 individuals.
6. E_p (seedling establishment probability) was taken from a separate experiment in our study that measured new recruits per estimated seed production per plot (Guthrie et al. 2016).
7. $f_d(y)$ (recruit size distribution) was estimated as a Gaussian distribution with mean μ and standard deviation σ from the 17 new recruits that we found in July of 2013.
8. L, U are constants corresponding to the lower and upper bounds of integration. L was defined as the smallest observed size $\times$ 0.9, and U was defined as the largest observed size $\times$ 1.1. We used 500 meshpoints for integration.
9. $n(x,t)$ is the relative frequency of individuals of size x at time t.

The kernels take the following form:

1. $n(y,t+1) = \int_L^U [P(y,x) + F(y,x)]n(x,t)dx$
2. $P(y,x) = s(x) * g(y,x)$
3. $F(y,x) = E_p * f_p(x) * f_s(x) * f_d(y)$

The vital rates take the following form:[2]

1. $Logit(s(x)) = \alpha_s + \beta_{s,1} * x + \beta_{s,2} * x^2$
   - α_s = -2.831, $\beta_{s,1} = 0.4035$, $\beta_{s,2}$ = -0.000421
2. $g(y,x) = \alpha_g + \beta_g * x$
   - $\alpha_g = 16.884$, $\beta_g = 0.9972$, $sd(g(y,x)) = 32.74774$
3. $Logit(f_p(x)) = \alpha_{f_p} + \beta_{f_p} * x$
   - α_{f_p} = -10.4478, $\beta_{f_p} = 0.0485$
4. $Log(f_s(x)) = \alpha_{f_s} + \beta_{f_s} * x$
   - $\alpha_{f_s} = 3.391$, $\beta_{f_s} = 0.0105$
5. $f_d(y) = Norm(\mu_{f_d} = 3.118, \sigma_{f_d} = 1.215)$

How to digitize this?

Now that we've written out the model based on the Appendix's description of how it works, we need to convert this to something PADRINO and `ipmr` can work with. I've found that it's usually easiest to start with the sub-kernels and work "top-down", though you are free to start with parameters and work "bottom-up". It's up to you!

Sub-kernels

We'll start with the IPM kernels. PADRINO basically uses the mathematical notation from above, but removes the (x,y) portion. So the $P(x,y) = s(x) * g(y,x)$ becomes `P = s * g`. Similarly, $F(y,x) = ...$ becomes `F = e_p * f_p * f_s * f_d`. We do not need to write out the model iteration from Equation 1. `ipmr` infers the correct form of that, provided we use the right states in `domain_start` and `domain_end` columns for each kernel formula.

[2]The notation used here is from the paper. We will need to re-write some of these to work in PADRINO.

Table 1: Names and forms for some common link functions and their inverses. In PADRINO, we want to digitize expressions using inverse link functions, because those give the expected value of the response on the correct scale. In this table, μ_i is the expected value of the response, η is the linear predictor. Log functions here are always natural logs (i.e. Log_e/Ln).

Link Name	Link Function	Inverse Link Function
Identity	μ_i	η_i
Log	$Log(\mu_i)$	$exp(\eta_i)$
Logit	$Log(\frac{\mu_i}{1-\mu_i})$	$\frac{1}{1+exp(-\eta_i)}$
Log-Log	$-Log(-Log(\mu_i))$	$exp(-exp(-\eta_i))$
Square Root	$\sqrt{\mu_i}$	η_i^2

Another important thing to note is that is that it really doesn't matter what you call the kernels, vital rates, and parameters. What matters is that their names match across the various tables in the database. Thus, if an author gives some vital rate a ludicrously long name, feel free to shorten it.[3] The examples below use the same names as the notation above for kernels, vital rates, and parameters, but this is not required!

Finally, because of some design decisions early in the development process, **you must always separate the left and right hand side of an = with a space.** For example, the `P = s * g` from above will not work if it is entered as `P=s*g`. On the bright side, it will make the formulae that you enter easier to read![4]

Vital rate expressions

PADRINO uses a similar notation as above, except now there are two new elements:

1. We will move all the link functions on the left hand side of each equation to the right hand side, and use the inverse link function instead. Table 1 below contains translations of some common link functions and their inverses.[5]
2. We only remove the x, y from the left hand side of the equation.

Survival The survival function $Logit(s(x)) = \alpha_s + \beta_{s,1} * x + \beta_{s,2} * x^2$ becomes:

- `s = 1 / (1 + exp(-(alpha_s + beta_s_1 * size_1 + beta_s_2 * size_1 ^ 2)))`

The `1 / (1 + exp(- (...)))` is the inverse logit transformation. This generates probabilities of survival, which is what we want for the IPM. Additionally, notice how the x's were replaced by `size_1`'s on the right hand side of the equation. Next, we'll deal with the growth kernel.

NB: Many more recent papers will have more complicated forms for the linear predictor which include terms for things like climate or competition. If the linear predictor gets really long, we can actually split it out into its own line in PADRINO like so:

```
-s_lin_p = alpha_s + beta_s_1 * size_1 + beta_s_2 * size_1 ^ 2

-s = 1 / (1 + exp(- s_lin_p))
```

[3] Caveat: in the course of debugging, it may be harder to track very short names. Additionally, if you have to wait a while for an author response to a question, you may forget what everything stand for by the time you return to a model!

[4] I hope to update this to take names from a separate column at some point, rather than using the left hand side of the `=`. However, since it currently works and only requires a bit of care when digitizing, this change isn't particularly high on my priority list.

[5] Table 15.1 on the second page of this book

Growth

In this case, the growth kernel uses a notation that will not work with PADRINO. However, the written description of the model indicates this is a linear regression with a Gaussian error structure. We can re-write this model using a more conventional notation. We can do this like so:

1. $g(y, x) = f_g(y, \mu_g, \sigma_g)$ Where f_g denotes a Gaussian probability density function.
2. $\mu_g = \alpha_g + \beta_g * x$

This format will correspond more closely to how we enter it PADRINO. Note that PADRINO does not use the *R* functions like `dnorm`, `dgamma`, etc. Rather, it uses its own shorthand that roughly follows the accepted statistical abbreviations for each probability function family (with some exceptions to avoid name conflicts in *R*/other languages). A table that contains the PADRINO format is here. PADRINO uses the "Abbreviation" column of this table. The PADRINO format for (1) and (2) become:

- `g = Norm(mu_g, sigma_g)`
- `mu_g = alpha_g + beta_g * size_1`

Notice that we omit the y from the right hand side of (1). PADRINO will infer the correct state variable based on the kernel it appears in. Additionally, in this particular case, their is no link function (or, rather, it's an identity link function which we can ignore), so we don't need to wrap the right hand side of (2) in any thing.

Seed production

The next vital rate we'll go to is seed production: $Log(f_s(x)) = \alpha_{f_s} + \beta_{f_s} * x$. Much like the survival model, we'll need to use an inverse link function on the right hand side. Unlike the survival model, we don't need the inverse logit, but rather, the exponential function (the inverse of the *Log*).

- `f_s = exp(alpha_f_s + beta_f_s * size_1)`

Probability of flowering and recruit size distributions.

These are similar to survival probability and the growth distributions. In the spirit of learning, try writing these down on your own and then checking them against the answers given down here.

General models

Above, we dealt with a simple model. It was a simple model because it used only 1 state variable, and had no discrete states. The vast majority of newly published IPMs are not simple models. Complex lifecycles often cannot be described by a single trait value. General IPMs allow for multiple continuous traits, and/or multiple discrete states. In PADRINO, little changes with respect to vital rate expressions and parameter values. However, the IPM kernels table will have more entries per model now, as we need more sub-kernels to describe the extra transitions.

Examples of discrete states are things like a seed bank for a plant, or the number of eggs produced by an insect that do not hatch immediately. These are discrete because they do not have any continuous trait that describes their population dynamics, only a single number indicating how many of them there are.

Some IPMs may have multiple continuous state variables. For example, a tree species may use height to describe dynamics of a seedlings below a certain size threshold, and then DBH (diameter at breast height) to describe the dynamics of larger trees.

Below, we'll go through another example from Baer & Maron (2018). We'll look at a single transition year from a single population first, and then use the rest of the data in the next example to demonstrate the suffix syntax that PADRINO uses. You can find the main text here and the appendix containing the necessary information here.

Where to begin

Assuming we've already entered most of the Metadata table, the first step is figuring out where to get the rest of it. Browsing through the paper, it looks like section 2.4 is likely to contain the information on the actual IPM. The end of the first paragraph contains key information on the number of kernels:

> "Our IPMs contained a continuous stage describing size based vital rates of juvenile and reproductive adult plants and two discrete stages in which constants described the vital rates of seedlings (whose size was not recorded) and seeds within the seed bank. The continuous stage of our IPMs was structured according to logtransformed basal rosette area (hereafter, size) and integrated across a range of sizes from the minimum size to 1.2 times the maximum size of any individual recorded in a population throughout all years of the study."

Ok, so we have 3 different states that we'll need to track: size of non-seedling plants, the number of seedlings, and the number of seeds in the seedbank. Next, we need to find out how they move around within and between those states. It becomes apparent from reading on that we won't find that information in the main text. However, the first sentence in the second column on page 5 indicates that we might be able to find it in Appendix S1. Let's open that up!

Digitizing the kernel formulae

Let's start with the seed bank terms. They've written it out as:

$$B_{t+1} = B_t * p_{sbanksurv} * (1 - p_{sbankgerm}) + \int_L^U p_{flower}(x) * f_{seeds}(x) * (1 - p_{germ}) n_t(x) dx$$

This expression is comprised of two sub-kernels, which we can recognize by the `+` that occurs about halfway through, and the fact that one part is multiplied by a discrete state (B_t) and another part is multiplied by a continuous state ($n_t(x)$). Thus, we need to split this single expression into two lines in the `IpmKernels` table. Bearing in mind that we do not include the actual state in these expressions (i.e. B_t and $n_t(x)$), these will take the form (column names now separated by commas):

1. formula = `SbStay = sb_surv * (1 - sb_germ)`, model_family = `DD`, domain_start = `seedbank`, domain_end = `seedbank`
2. `SbGo = p_fl * n_seeds * (1 - p_germ) * d_size`, model_family = `CD`, domain_start = `size`, domain_end = `seedbank`

One critical aspect to note is that in this case, we've added the `d_size` variable to expression 2. `ipmr` requires us to specify the integration for general IPMs, whereas it will automatically infer it for simple ones. In all cases, if you encounter an integral, this `d_size` will correspond to the dx in the integral. We just need to make sure we choose the state variable.

Additionally, we now have different values for model_family, domain_start, and domain_end.

We can now move onto the second expression in the IPM iteration:

$$S_{t+1} = p_{sbankgerm} * B_t + \int_L^U p_{flower}(x) * f_{seeds}(x) * p_{germ} n_t(x) dx$$

Again, we have two sub-kernels defined in this portion of the iteration. The first moves seeds out of the seedbank to create seedlings, and the other creates seedlings from adult plants that are reproductive. We write them as follows:

1. formula = `SbSdl = p_sb_germ`, model_family = `DD`, domain_start = `seedbank`, domain_end = `seedling`
2. formula = `goSdl = p_fl * n_seeds * p_germ * d_size`, model_family = `CD`, domain_start = `size`, domain_end = `seedling`

The final part of the IPM iteration generates plants with some size distribution. It has the following form:

$$K = p_{seedlingsurv} * f_{recrsize}(x') * S_t + \int_L^U s(x) * g(x'|x) n_t(x) dx$$

Try writing out the formula, model_family, domain_start, and domain_end on your own. You can check your answers here. This one will be tricky because the notation in this particular equation tells you what the iteration kernel K looks like for the continuously distributed variable, size. We need to re-write it so the left hand side is the continuous portion of the population state at $t+1$ (i.e. $n(x', t+1) = ...$)?. Hint: the function for new recruit sizes ($f_{recrsize}(x')$) is a probability density function.

Digitizing the Vital rate expressions

Most of the vital rate expressions are only described qualitatively, rather than explicitly written out. Additionally, they are mostly written out in the main text, specifically in the second and third paragraphs of section 2.4. We'll start by parsing the following section:

> "Within the continuous portion of the IPMs, logistic regressions with a logit link described size-based binomial survival probability. Linear models described size-based growth rates and variance around this relationship. Log-transformation of plant size eliminated the need for a size-based estimate of the variance around the size– growth relationship for most models, as verified by a Breusch–Pagan test performed on each model for this relationship (bptest, lmtest package, Zeileis & Hothorn, 2002). In these cases, variance was described according to the residual standard deviation. For size-growth models that remained heteroscedastic following log-transformation of plant size, we described variance in growth as an exponential function of log-transformed plant size when constructing growth models: $\sigma(size) = \sqrt{\sigma^{2[2*constant*size]}}$, where the constant and σ^2 were calculated within IPMpack... We compared the fit of models containing linear and quadratic terms for each vital rate using their corrected Aikake Index Criterion scores (AICc; quasi-AICc—QAICc—for models of overdispersed per capita seed production). If $\Delta(Q)AICc > 2$, we selected the model with the lower score unless the model was graphically determined to be biologically unrealistic (Supporting Information Appendix S1). In cases where $\Delta(Q)AICc \leq 2$, we constructed models with averaged parameters weighted by (Q)AICc score (`model.avg()`; MuMIn package; Barton, 2016)."

Survival

It helps to split out the different vital rates when working out the correct functional form. We'll start with survival, which we'll denote $s(x)$ to follow their notation in the IPM kernels. They first specify that they used logistic regression with a logit link, so we know that the left hand side of the vital rate will look like this:

1. $Logit(s(x)) = ...$

Next, we need to work out what the linear predictor looks like. In the latter half of the quoted section above, the authors state that they compared models that included linear and quadratic fits, and then possibly averaged over the predictions if the two candidate models were not sufficiently different in their predictive

ability. However, it will be more general of an approach to always included the quadratic term in the vital rate expression, and then set the coefficient to 0 if it turns out that the model selection procedure chose the linear model over the quadratic/averaged model. Thus, we write:

1. $Logit(s(x)) = \alpha_s + \beta_{s,1} * x + \beta_{s,2} * x^2$

with the understanding that $\beta_{s,2}$ can be 0, which yields the linear model. We'll get into why this is helpful in the *Suffix syntax* section below. Now that we have our nice functional form for the regression model, we can re-write it with PADRINO's syntax (remember to use the inverse link function!):

- `s = 1/ (1 + exp( - (s_int + s_slope_1 * size_1 + s_slope_2 * size_1 ^ 2)))`

Growth

The growth model is similar to the survival model in terms of how the mean is parameterized, but the variance is something new for us. Fear not! We can actually accommodate this form with just one extra row in the VitalRateExpr table. Again, the first step is to write out what's going on. We have our linear and quadratic models that compute the mean size at $t+1$ given the size at t. Furthermore, it sounds like this is a linear regression with a Gaussian error distribution. However, they also mention that the variance in growth may be a function of size as well. Fortunately, they also provide the equation that tells us exactly how to compute that quantity (rather, they provide even more conveniently, the formula for the standard deviation, which is what we need for the growth kernel anyway). So, lets write down what's going on, first using the trick we used above for the linear/quadratic functional form.

- $\mu_g = \alpha_g + \beta_{g,1} * x + \beta_{g,2} * x^2$

Next, we also need to write out how to compute the standard deviation. The authors note that sometimes, they assume a constant variance around the growth curve, while other times, there is an exponential change in it as a function of initial size. We'll write it so that it's always an exponential function, and then see how to alter the way we enter the data so that we can use that functional form, no matter which type of model was actually used:

- $\sigma(x) = \sqrt{\sigma^2 * e^{2*\beta_{gv}*x]}}$

Ok, so what does this mean for us? For the exponential function case, we just need to find the value for the model's variance σ^2 and β_{gv}, the parameter that controls how rapidly the standard deviation changes with size. These are reported in the Appendix of this paper. For the model that uses constant variance, these authors report the standard deviation, rather than the variance. Thus, when we enter the parameter values for the constant variance models, we should enter the squared value of the standard deviation as the model's variance, and then set β_{gv} to 0, as anything raised to the zero-th power is 1 (i.e. it will become $\sqrt{\sigma^2 * 1}$). In PADRINO syntax, the complete set of expressions are:

- `` `g = Norm(mu_g, sigma_g)` ``

- `` `mu_g = g_int + g_slope_1 * size_1 + g_slope_2 * size_1^2` ``

- `` `sigma_g = sqrt(sigma_2_g * exp(2 * g_var_slope * size_1))` ``

Seed production

The final bit of new syntax we'll introduce here is for the seed production part of the model. The authors state that they used a GLM with a quasi-Poisson error and a log link. We are only interested in the link function for now.[6] The authors state that they compared linear and quadratic forms, so we'll use the same

[6]The error family is important in the estimation step, and when quantifying uncertainty in the model. However, implementing a model with observed parameter values is unaffected by this, so we'll ignore quasi-Poisson part for now, and return to it when we need to deal with the UncertaintyTable.

trick as before to enter a general formula for the vital rate expression and adjust the parameter values to work with it. Furthermore, they state that seed production was capped at the maximum observed seed number for each year × site combination. This indicates we'll need a second expression to implement to the capping. Fortunately, they tell us the cap's value at each year × site combination in the appendix, so we can add that as a parameter.

First, we need to write out what's going on. The model can generally be summarized as:

- $Log(f_s(x)) = \alpha_{f_s} + \beta_{f_s,1} * x + \beta_{f_s,2} * x^2$
- $f_{seeds}(x) = \begin{cases} f_s(x) & \text{if } f_s(x) < f_{max} \\ f_{max} & \text{if } f_s(x) \geq f_{max} \end{cases}$

The first equation should look familiar based on the first example we did. The second one is a piece-wise function with a condition that sets out when to use the estimated fecundity and when to use the maximum value. PADRINO does its best to remain "language-agnostic", that is provide a syntax that is not specific to *R*, even if that's what we're working with. However, for the second expression, we've decided to use *R*'s `ifelse()` in PADRINO.[7] These expressions then become:

- `f_s = exp(f_s_int + f_s_b_1 * size_1 + f_s_b_2 * size_1 ^2`
- `n_seeds = ifelse(f_s < seed_max, f_s, seed_max)`

Rather than attempt to write out the rest of the expressions, we'll now move on to the suffix syntax that `ipmr` and `PADRINO` use. We'll now work on a single site from Baer 2018, but we'll start writing expressions that represent all years sampled.

Suffix syntax

`ipmr` implements a suffix syntax that's meant to mirror the sub-/super-script notation of the math that represents IPMs. This allows us to create expressions that are indexed by some variable, usually time or space (e.g. sites, population, etc.), and can save a considerable amount of duplication in the database. We'll go through a quick introduction here, and then the worked example. If you want to read more about it, there is a vignette with a lengthier here.

We'll use the same paper as the general IPM example above, except this time, we're going to modfiy the time-varying expressions by appending a suffix to each parameter/vital rate/kernel. Since the interval of time-varying is year to year, we'll use the subscript x_{yr} in the mathematical formulae, and the suffix `_yr` in PADRINO.

Re-writing the model iteration

As before, we want to write out the expressions first. The catch here is that we need to make sure we notice which parameters are time-varying and modify those symbols accordingly. Glancing at the parameter tables in Appendix 1, we note that for each site, the only time-invariant parameters are $p_{sbanksurv}, p_{germ}, p_{sbankgerm}$. Therefore, we'll add a subscript for each year to every other vital rate function and parameter.

Beginning with the iteration for the seedbank, we get:

$$B_{t+1} = B_t * p_{sbanksurv} * (1 - p_{sbankgerm}) + \int_L^U p_{flower,yr}(x) * f_{seeds,yr}(x) * (1 - p_{germ}) n_t(x) dx$$

Next, we modify the iteration for the seedling stage:

[7] This should be translatable to, for example, *C++*'s ternary operator equivalent `x > y ? a : b`.

$$S_{t+1} = p_{sbankgerm} * B_t + \int_L^U p_{flower,yr}(x) * f_{seeds,yr}(x) * p_{germ} n_t(x) dx$$

And finally, we update the continuous distribution of plant sizes: [8]

$$n_{t+1}(x') = p_{seedlingsurv,yr} * S_t * \int_L^U f_{recrsize,yr}(x') dx + \int_L^U s_{yr}(x) * g_{yr}(x'|x) n_t(x) dx$$

Now that we've modified the model iteration, we can enter these into the IpmKernels table. For the seedling class (S_{t+1}), will have the following form:

1. formula = `SbSdl = p_sb_germ`, model_family = `DD`, domain_start = `seedbank`, domain_end = `seedling`
2. formula = `goSdl = p_fl_yr * n_seeds_yr * p_germ * d_size`, model_family = `CD`, domain_start = `size`, domain_end = `seedling`

Try writing out the formulae, model_family, and domain_start/end for the other two states in the model. You can check your answers here.

Time-varying vital rate expressions

In order to express the time varying functions, we also need to add subscripts/suffixes to each of the symbols in the vital rate expressions. In the case of this model, we can see that, relative to the General IPM exercise, we need to add suffixes to the following vital rates:

- $p_{flower,yr}$
- $f_{seeds,yr}$
- $p_{seedlingsurv,yr}$
- $f_{recrsize,yr}$
- s_{yr}
- g_{yr}

Survival

Starting with survival at the Bountiful population, we see that 2 of the years have quadratic terms, and the first year does not. We can still write all of these using the quadratic functional form, and then just set `s_slope_2_2013` to 0 in the ParameterValues table. On paper, we can express the model like this:

- $Logit(s_{yr}(x)) = \alpha_{s,yr} + \beta_{s,1,yr} * x + \beta_{s,2,yr} * x^2$

In PADRINO, we append the suffix `_yr` to each parameter value it modifies:

- `s_yr = s_int_yr + s_slope_1_yr * size_1 + s_slope_2_yr * size_1^2`

In the ParameterValues table, we'll need to enter 9 rows for this vital rate:

- Intercepts: `s_int_2013`, `s_int_2014`, `s_int_2015`
- First slope: `s_slope_1_2013`, `s_slope_1_2014`, `s_slope_1_2015`
- Second slope: `s_slope_2_2013`, `s_slope_2_2014`, `s_slope_2_2015`

[8] NB: This has been re-written from the notation used in the appendix so that it is a model iteration rather than an iteration kernel. This is suffix-notation version of the exercise in the General IPMs section.

Next, in the HierarchTable, we need to enter the following:

- vr_expr_name = `yr`, range = `2013:2015`

Growth

Next, we'll do growth. Like survival, growth at Bountiful will have the same functional form as the General IPM example, except now with subscripts/suffixes. We write it out as follows:

1. $g_{yr}(x', x) = f_g(x', \mu_{g,yr}, \sigma_{g,yr})$ where f_g is a Normal probability density function.
2. $\mu_{g,yr} = \alpha_{g,yr} + \beta_{g,1,yr} * x + \beta_{g,2,yr} * x^2$
3. $\sigma_{g,yr} = \sqrt{\sigma^2_{g,yr} * e^{2*\beta_{gv,yr}*x}}$

We can write out the PADRINO expressions like so:

- `g_yr = Norm(mu_g_yr, sigma_g_yr)`
- `mu_g_yr = g_int_yr + g_slope_1_yr * size_1 + g_slope_2_yr * size_1 ^ 2`
- `sigma_g_yr = sqrt(sigma_2_g_yr * exp(2 * beta_gv_yr * size_1))`

The next step will be to enter the parameter values in the ParameterValues table. We don't need to enter anything more in the HierarchTable - those values can be used by the whole IPM.

The rest of the vital rates

Try writing these out on your own, and checking the answers here

Parameter resampled models

Parameter re-sampled models refer to stochastic models where some parameters are treated as time-varying random variables. The `EnvironmentalVariables` table is meant to hold the parameters associated with these random variables (e.g. the mean and variance of a temperature distribution), and the random number generators that produce values (e.g. the `Norm(temp_mu, temp_sd)`). These distribution functions use the same notation as the density functions, so you can still use the Probability Function Dictionary to translate these into PADRINO notation.

The way that we enter vital rate expressions and IPM sub-kernels doesn't change when working with these models. The only additional bit is that we now enter some of the parameters and expressions into the `EnvironmentalVariables` table. I will find a suitable example for this in the not so distant future.

Model iterations

Vital rate expressions

EnvironmentalVariables expressions

Age × size models

Answers to exercises

Simple model Pr(flowering) and recruit size

1. $Logit(f_p(x)) = \alpha_{f_p} + \beta_{f_p} * x$
- `f_p = 1 / (1 + exp(-(alpha_f_p + beta_f_p * size_1)))`
2. $f_d(y) = Norm(\mu_{f_d}, \sigma_{f_d})$
- `f_d = Norm(mu_f_d, sigma_f_d)`

Resume from General models

General IPM kernel formulae

$K = p_{seedlingsurv} * f_{recrsize}(x') * S_t + \int_L^U s(x) * g(x'|x) n_t(x) dx$

1. formula = `SdlOut = p_sdl_surv * sdl_size * d_size`, model_family = `DC`, domain_start = `seedling`, domain_end = `size`
2. formula = `P = s * g * d_size`, model_family = `CC`, domain_start = `size`, domain_end = `size`
3. K re-write: $n(x', t+1) = p_{seedlingsurv} * S_t * \int_L^U f_{recrsize}(x')dx + \int_L^U s(x) * g(x'|x) n_t(x) dx$

Back to vital rate expressions in General IPMs

Suffix syntax kernel formulae

Seed bank

1. formula = `SbStay = sb_surv * (1 - sb_germ)`, model_family = `DD`, domain_start = `seedbank`, domain_end = `seedbank`
2. `SbGo_yr = p_fl_yr * n_seeds_yr * (1 - p_germ) * d_size`, model_family = `CD`, domain_start = `size`, domain_end = `seedbank`

Plants

1. formula = `SdlOut_yr = p_sdl_surv * sdl_size_yr * d_size`, model_family = `DC`, domain_start = `seedling`, domain_end = `size`
2. formula = `P_yr = s_yr * g_yr * d_size`, model_family = `CC`, domain_start = `size`, domain_end = `size`

Vital Rates

1. $Logit(p_{flower,yr}(x)) = \alpha_{p_f,yr} + \beta_{1,p_f,yr} * x + \beta_{2,p_f,yr} * x^2$

- `p_fl_yr = 1 / (1 + exp(-(p_fl_int_yr + p_fl_slope_1_yr * size_1 + p_fl_slope_2_yr * size_2 ^ 2)))`
- `ParameterValues` entries:
  - `p_fl_int_2013`, `p_fl_int_2014`, `p_fl_int_2015`
  - `p_fl_slope_1_2013`, `p_fl_slope_1_2014`, `p_fl_slope_1_2015`
  - `p_fl_slope_2_2013`, `p_fl_slope_2_2014`, `p_fl_slope_2_2015`

2. There are two expressions here. One for the predicted mean, and the other for the conditional cap on seed production

- $f_{s,yr} = \alpha_{f_s,yr} + \beta_{f_s,1,yr} * x + \beta_{f_s,2,yr} * x^2$
- $f_{seeds,yr}(x) = \begin{cases} f_{s,yr}(x) & \text{if } f_{s,yr}(x) < f_{max,yr} \\ f_{max,yr} & \text{if } f_{s,yr}(x) \geq f_{max,yr} \end{cases}$
- `f_s_yr = exp(f_s_int_yr + f_s_slope_1_yr * size_1 + f_s_slope_2_yr * size_1 ^ 2)`
- `f_seeds_yr = ifelse(f_s_yr < seed_max_yr, f_s_yr, seed_max_yr)`
  - `ParameterValues` entries:
    - `f_s_int_2013`, `f_s_int_2014`, `f_s_int_2015`
    - `f_s_slope_1_2013`, `f_s_slope_1_2014`, `f_s_slope_1_2015`
    - `f_s_slope_2_2013`, `f_s_slope_2_2014`, `f_s_slope_2_2015`
    - `seed_max_2013`, `seed_max_2014`, `seed_max_2015`

3. $p_{seedlingsurv,yr}$: This actually has no expression, it's just a set of 3 parameter values.

- `sdl_s_2013`, `sdl_s_2014`, `sdl_s_2015`

4. $f_{recrsize,yr} = Norm(\mu_{recrsize,yr}, \sigma_{recrsize,yr})$

- `ParameterValues` entries:
  - `mu_rcsz_2013`, `mu_rcsz_2014`, `mu_rcsz_2015`
  - `sigma_rcsz_2013`, `sigma_rcsz_2014`, `sigma_rcsz_2015`

Back to Suffix Notation