Writing IPM Formulae

Sam Levin

January 12, 2019

Contents

General Overview

`Padrino` uses its own type of notation that is somewhat R-like, but still general enough that users of other languages could adapt it on their own. It uses text strings of the formula for each successive level of parameters in an IPM (constants, vital rates, kernels) to represent the underlying models and generate the correct iteration matrices. The notation requires some practice to get right and has a few quirks in it due to the nature of working with strings in R/Perl.

Briefly, this document uses the following format to distinguish between tables in the database, columns in a table, and entries in a column:

Table in the database

Column in a table

`Entry in a column`

This document generally only applies to the following tables in the database: **IpmKernels**, **VitalRateExpr**, **ParameterValues**, and **EnvironmentaVariables**. Occasionally, other tables will be referenced though. They will be highlighted using the same font scheme as above.

The Formulae

In general, The *formula* column of **IpmKernels** and **VitalRateExpr** will always have a Left Hand Side (abbreviated w/ LHS) and a Right Hand Side (RHS). Getting these correct is critical to the functioning of the database. The LHS and RHS have slightly different notation schemes at each level.

Explanations for how to enter each type of formula/expression are given in this section. Additionally, there are worked examples in the next section which will hopefully help to clarify any confusion in the definitions. If you do find yourself confused, please email me at levisc8@gmail.com or file an issue in the project's github repository (https://github.com/levisc8/Padrino/issues) so that I can work on improving this document! I'll also try to provide a more personalized introduction to this rather idiosyncratic topic.

Constants

In ParamateverValues

All entries in **ParameterValues** should be constants (e.g. there is no additional function stored away somewhere that calculates them). Thus, this is the exciting part where you get to choose names for all of the parameters in your model! Yippee!! Thus, the column `parameter_name` should just contain the name of the parameter; no LHS or RHS. All terms in this table should appear in the RHS of an expression in one of the other tables.

In VitalRateExpr and IpmKernels

All constants that **do not** appear in a **VitalRateExpr** and only appear in an **IpmKernel** need to be written as follows in **VitalRateExpr** *formula*:

`parameter_name = parameter_name` where parameter_name is the name of the parameter taken from the **ParameterValues** table.

This ensures that all kernels that use that value can access it when it comes time to build them.

In EnvironmetVariables

EnvironmentVariables is a catch-all table that describes the following types of IPMs: ones with hierarchical models (e.g. with random effects/discrete fixed effects beyond just the state variable), environmentally dependent IPMs (e.g. contains terms for temperature or precipitation in their vital rate expressions), and density dependent IPMs (where vital rates are at least partially a function of the number/state structure of individuals at each time step). Currently, only the first type are implemented, so this notation may change. Beware!

For hierarchical models, this table stores the different levels of the hierarchical effect and provides information on suffixes that these values are meant to replace. Temporal effects (e.g. years of sampling) and spatial effects (e.g. plot names) are the most common types of hierarchical effects, but others will appear as well.

env_variable provides a description of the effect. Using the two examples above, one could enter `year` or `plot` in this column.

vr_expr_name gives the suffix that is used to denote this effect in other parts of the database. This is appended to the name of the continuous state variable , so if the state variable in the model was `size`, then this could be entered as `size_yr` or `size_pl`. If a model contains multiple hierarchical effects, then each one gets its own row in the table.

Unforunately, *env_range* introduces a bit of complexity. Hierarchical effects are generally discrete, may or may not have a set order, and ordered effects may or may not be easily represented by a sequence of integers. Below we cover both cases.

For effects that have an ordered sequence to them (e.g. years of sampling), there are two different ways to enter them, depending on whether the ordering is consecutive or non-consecutive. If the order is consecutive and is easily represented with integers (e.g. years of sampling), they can be entered using the following notation: `year_1:year_n-1`) where year_1 is the first year of sampling and year_n-1 is the second to last year of sampling.

For effects that are not ordered or are not easily represented with a sequence of consecutive integers, then the following notation is used: `c("level_A", "level_B", "level_C", "Level_...")`. Note that each level is now encapsulated with quotation marks, all levels are separated by commas, and everything is wrapped in `c()`. This is very important (though perhaps a bit lazy of me and the least “language-agnostic” aspect of the database).

The above is a bit vague and confusing, so see the Worked Examples in the next section for a more concrete demonstration of how this works.

Why all this confusion for hierarchical models?

The general goal of the framework we implemented for hierarchical models is to represent them as closely to how they are defined as possible. This has the additional benefit of greatly reducing the amount of typing required to enter them, consequently lowering the risk of typos. A separate PDF explains how this substitution works and can be found here.

Of course, there’s nothing quite like diving into the source code to understand the logic of it all! The code that deals with the split-duplicate-combining of hierarchical models can be found here (top level function) and here (internal helpers).

Vital rates

In VitalRateExpr

The following is relevant to *formula* and *kernel_id*.

formula should contain a textual representation of the vital rate model. The LHS of the model should include the name of the vital rate and have the state variable that is a function of in parentheses (e.g. `surv(size_1) = ...`). The RHS of the model should contain the coefficients and mathematical transformations needed to produce a single value on the LHS in the form of a mathematical expression (e.g. `1 / (1 + exp(-(surv_int + surv_slope * size_1)))`. The inverse logit transformation is performed on the RHS of the expression as opposed to the LHS (as it often appears in papers). Thus, the complete entry in *formula* becomes

```
surv(size_1) = 1 / (1 + exp(-(surv_int + surv_slope * size_1)))
```

Some vital rates may contain probability density functions (PDFs). Each density function has a specific abbreviation that we use and these expression wrap their arguments in parentheses. For example, a growth function may be entered as:

```
g(size_2, size_1) = Norm(mean_growth, sd_growth)
```

Vital rates that are modeled with hierarchical effects get the same suffixed notation that was described above. For example, if the survival model from above was to include a random intercept for year (abbreviated `yr` in **EnvironmentVariables), then it would be re-written as

```
s_yr(size_1) = 1 / (1 + exp(-(surv_int_yr + surv_slope * size_1)))
```

Notice that the only change we made was to insert the `_yr` suffix to the parameters that were modified by the hierarchical effect.

See the Common Transformations and Functions section for some examples of common mathematical transformations that appear in here. Additionally, see the PDF Abbreviations sections for a complete list of probability density functions that we support and their associated abbreviations.

For models without any sort of hierarchical effects, *kernel_id* is just the name of the kernel (e.g. `K_all; P; F`).

For models with hierarchical effects, the suffix from the **EnvironmentVariables** *vr_expr_name* is appended to the typical kernel id. For example, if the random year effect is abbreviated with `yr`, then the *kernel_id*s become `K_all_yr; P_yr; F_yr`, etc. For models with multiple hierarchical effects, for example multiple plots and many years, then both suffixes are appended to each *kernel_id* so the above becomes `K_all_yr_pl; P_yr_pl; F_yr_pl`.

Note that regardless of whether there are hierarchical effects or not, vital rates that appear in multiple kernels are only written once and are always separated by a semicolon (the ";").

In IpmKernels

Vital rates are denoted in **IpmKernels** using the LHS of the *formula* from **VitalRateExpr**. The key difference here is that rather than using parentheses to wrap the the state variable, square brackets are used. Below is an example of a P kernel that combines survival and growth.

```
P[size_2, size_1] = s[size_1] * g[size_2, size_1]
```

If the model contains hierarchical effects, then the suffix is again appended to each term that the hierarchical effect modifies. Consider a model with a random effect for year with the suffix `yr` from **EnvironmentVariables**. The P kernel from above then becomes

```
P_yr[size_2, size_1] = s_yr[size_1] * g_yr[size_2, size_1]
```

Note that the the state variables are not modified with the suffix, only the vital rates and kernels.

Worked Examples

Non-hierarchical models

Hierarchical models

Common Transformations and Functions

PDF Abbreviations