Padrino digitization guide

Sam Levin

3/11/2021

Contents

Padrino

Welcome to the project! Padrino is an open access data base that aims to store text representations of as many (ideally all) Integral Projection Models (IPMs) that have been published. The goal is to ensure these models can be rebuilt *as published*, though there are exceptions in some cases (see here). Beyond the functional forms and parameters, we also store a bunch of metadata to help users select models for synthesis.

It is important to note that Padrino does not store any raw data. That is a very different task from the one we've set out to accomplish, and, while it would be exceptionally useful to have all that raw data, it is not within the scope of this project. Fortunately, it is usually easier to get authors to send you a functional form

for a model than it is to get them to part ways with their hard-earned raw data. As a digitizer, this will hopefully make your life a lot easier.

Steps to digitizing

There a few steps in digitizing a paper. These are outlined briefly here, and each has a deeper dive (when applicable) somewhere below (with links to each).

Identify whether a publication contains an IPM

The first step of entering a paper is to find papers and store them somewhere. Below, we go through the steps this involves, in order.

1. Tracking our incoming literature channel on Slack, as well as checking other sources for papers we may have missed (e.g. Google Scholar, Web of Science, Twitter (seriously, it's a good source)).
2. If a paper has an IPM, then we first add it to our shared GoogleSheet containing citation information, species names, author contact info, and digitization progress. This can be accessed here. If you are denied access, then remind me to send you an invitation to edit.
    - NB: on the slack channel, papers that have been posted there and contain an IPM get a thumbs up reaction, all others get a thumbs down. You can "reply in thread" to open a discussion about whether or not to include a paper.
3. Download a PDF of the paper into the shared Dropbox literature folder. Additionally, if it contains an Appendix with further information, store that as well. The folder's relative path within the PADRINO dropbox folder is: `Dropbox/PADRINO/Literature/PDF/KINGDOM-NAME`.
4. If you are ready to begin digitizing the paper, then update the "Digitized" and "Digitizer" columns in the Google Sheet to reflect that you are working on this paper.

Determine the number of `ipm_id`'s in the paper

The `ipm_id` currently takes the place of an SQL key. It is the only column that appears in every table in Padrino. Some models may contain many rows in some tables while others contain none. This necessitates some piece of information to make sure we don't accidentally pull in information from a different model at build time. The `ipm_id` column fills this role. Because of the way that `ipmr` handles models with some grouping effects (e.g. plots, years, populations), it may be possible to re-construct many kernels using a slightly modified functional form (i.e. appending a suffix corresponding to the group effects). This saves us a lot of typing/copy+pasting (which can be error prone). The rules for this are as follows:

1. If all of the kernels in the paper have the same functional forms, then we can use a single IPM ID.
    - These are commonly the result of mixed effects models, or models where something like an experimental treatment is a fixed effect. More generally, if an underlying regression model contains some mixture of discrete and continuous predictors, then the result is going to be multiple kernels - 1 for each level of the discrete predictor.
2. If any of the underlying kernels have different functional forms across a discrete grouping variable, then we can't combine them, and they must be split into separate `ipm_id`'s.
    - This may happen when, for example, different sites or years generate such different demographic responses that the authors could not find a way to keep everything in one model.

Once we've decided how many entries the paper is going to get, then we can begin entering the model.

The tables

Next, we'll walk through individual tables and the columns within them. The first is `Metadata`.

Metadata

The metadata table contains information on species taxonomy, publication information, study duration, ecoregion, and any treatments applied to a population. The columns are:

1. `ipm_id`: This is a unique identifier for each model. It is 6 alphanumeric characters with no spaces.

Taxonomic information

2. `species_author`: This the latin species name that the author uses in the manuscript. It may no longer be the accepted name though. This has the format `genus_species`. Some authors may include additional information (subspecies, varieties, etc). These can be appended using underscores (e.g. `genus_species_subspp_var`).
3. `species_accepted`: The accepted name of the species (currently from The Plant List, but should switch to the Leipzig List soon). This follows the format `genus_species`, but does not contain any varietal or sub-species information.

The following taxonomic categories contain the `"tax_"` prefix to prevent naming ambiguity with some R functions.

4. `tax_genus`: The accepted genus.
5. `tax_family`: The accepted family.
6. `tax_order`: The accepted order.
7. `tax_class`: The accepted class.
8. `tax_phylum`: The accepted phylum.
9. `kingdom`: The kingdom.
10. `organism_type`: The type of organism. For plants, this is usually something like `"Herbaceous perennial"`, or `"Shrub"`. For animals, this could be, for example, `"mammal"` or `"reptile"`. See here for more details (but also don't hesitate to contact me if there instances that fall outside of the classification given there).
11. `dicot_monocot`: Whether the species is a dicotyledon or a monocotyledon (only applies to plants).
12. `angio_gymno`: Whether the species is a angiosperm or a gymosperm (only applies to plants).

Publication information

13. `authors`: The last names of each author on the manuscript, separated by a semicolon.
14. `journal`: The abbreviated name of the journal that the model appears in. This follows the BIOSIS format. Exceptions are when the source is not a journal (e.g. a PhD/MSc thesis, government report). In that case, we use something like `"PhD Thesis"` and then include a link in the `remark` column.
15. `pub_year`: The year the article was published.
16. `doi`: The DOI of the publication (NOT THE doi.org URL though!!).
17. `corresponding_author`: The last name of the corresponding author.

18. `email_year`: The corresponding author's email, along with the year of publication in parentheses to denote how old (and possibly inaccessible) it is. For example, this could `levisc8@gmail.com (2020)`
19. `remark`: Any qualitative comments you may have on the model. These can range from comments to accuracy of GPS coordinates to descriptions of the different levels of a treatment that was applied.
20. `apa_citation`: The full APA style citation for the paper.
21. `demog_appendix_link`: If there is one, a link to the Electronic Supplementary Material that contains further details/parameter values for the model.

Data collection information

21. `duration`: The duration of data collection used to implement the model. This is a crude measure, defined as `end_year - start_year + 1`, and does not account for years where data collection may have been skipped.
22. `start_year`: The year that demographic data collection began.
23. `start_month`: The month of the year that demographic data collection began. This is an integer between 1 and 12, where 1 corresponds to January, and 12 corresponds to December.
24. `end_year`: The final year of demographic data collection.
25. `end_month`: The month of the year that demographic data collection concluded.
26. `periodicity`: Indicates the time step (periodicity) for which the seasonal, annual, or multi-annual IPM was constructed. For example, 1 indicates that the IPM iteration period is 1 year; 0.5 indicates that the IPM iterates once every 0.5 years or 6 months; 2 indicates that the IPM iteration occurs every 2 years.
27. `population_name`: The name of the population given by the author. For example, `"Bear Creek"`, or `"Havatselet"`.
28. `number_populations`: Sometimes, a `population_name` may encompass multiple sub-populations that are located close by. This integer specifies the number of populations/sub-populations that are described by the model.
29. `lat`: The decimal latitude of the population.
30. `lon`: The decimal longitude of the population.
31. `altitude`: The altitude above/below sea level, in meters.
32. `country`: The ISO3 country code for the country in which the population is located.
33. `continent`: The continent that the population is located on. Options are `n_america`, `s_america`, `oceania`, `asia`, `europe` and `africa`. Others may be added as needed.
34. `ecoregion`: The ecoregion, as defined by the World Wildlife Fund (see here).

Model information

35. `studied_sex`: The sex of the population studied. Options are `M` (male only), `F` (female only), `H` (hermaphrodites), `M/F` (males and females modeled separately, but in the same IPM), and `A` (all sexes studied together).
36. `eviction_used`: Whether or not the authors account for unintentional eviction. This should either be `TRUE` or `FALSE`.

37. `evict_type`: If an eviction correction was used, then the name of the method to correct it. Options are `stretched_domain`, `rescale_kernel`, `truncated_distributions` and `discrete_extrema`. `rescale_kernel` is the same as `truncated_distributions` and a relic of the past. Feel free to update those entries to `truncated_distributions` as you go.
    - NB: `stretched_domain` does not indicate that any function was applied to the kernels to return individuals. Rather, it just means that the authors extended the size boundaries some amount beyond the observed size distributions and relied on vital rate functions to prevent individuals from ever getting to the bounds. This information is included in PADRINO so users can understand what approach was used to deal with the problem, and, theoretically, try implementing models with different eviction corrections.
38. `treatment`: A brief description of any experimental treatment that the authors applied to a given population.
39. `has_time_lag`: Indicates whether any vital rates have a time lag (i.e. are a function of $n(z^*, t-1)$ rather than $n(z,t)$).
40. `has_age`: Indicates whether the model has age structure.
41. `has_dd`: Indicates whether the model incorporates density dependence.

Database specific information

42 `.embargo`: Have the authors requested an embargo period for the model?

43. `.embargo_date`: If so, when have they agreed to allow us to release it?

You've made it through the metadata table! Next, we'll get into IPM specific details, starting with the state variables in use.

StateVariables

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.
2. `state_variable`: the name of the of state variable that the model uses. This is largely up to you to choose. It can be descriptive (e.g. `"dbh"`, `"leaf_area"`), or vague (`"size"`).
3. `discrete`: Whether or not the state variable is discretely or continuously distributed.

DiscreteStates

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.
2. `state_variable`: The name of the discrete state variable. This should match the entry from the StateVariables table.
3. `model_name`: This can be same as `state_variable`, or, if it's a long name, can be an abbreviated form of it. This is the name that will be used when building the model, so

ContinuousDomains

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

IntegrationRules

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

StateVectors

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

IpmKernels

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

VitalRateExpr

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

ParameterValues

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

EnvironmentalVariables

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

HierarchTable

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

UncertaintyTable

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

TestTargets

1. `ipm_id`: The 6 digit alphanumeric `ipm_id` from the Metadata table.

Common issues

There are a number of issues can (and probably will) arise when digitizing papers. Many authors do not include all the information needed to fully rebuild their models. Wrangling with $LaTeX$ when creating pretty functional forms can introduce accidental typos which do not reflect the code that was used to actually build and analyze the model. Below, there are some guideline for how to handle these situations.

Missing information

Tragically, the vast majority of papers published do not contain all the information we need to re-build these IPMs. Therefore, we also contact authors to request the missing details. The most common ones are things like the numerical integration rule, the number of meshpoints, and the upper/lower bounds for the state variables used. Functional forms for some vital rates are also pretty common. We have a template for requesting information available in `metadata/digitization/author_contact_pdb.Rmd` (at some point, I'll convert that to a function that automatically generates email text given an `ipm_id` from a `pdb_raw` object. For now, you'll need to edit it manually).

Exceptions to our digitization rules

There are only a few times when we might want to enter a model differently from how it appears in the publication. The main way is when there is a typo in a functional form in the manuscript/appendix that doesn't match what is happening in the code the authors provide. For example, consider a survival model with a logistic regression of `survival ~ size`. The correct form would be:

$$s_z = \frac{1}{1 + exp(-(\alpha_s + \beta_s * z))}$$

However, the paper may contain the following form (notice the missing $-$ in the denominator):

$$s_z = \frac{1}{1 + exp(\alpha_s + \beta_s * z)}$$

In this case, it is appropriate to contact the author to double check what the exact functional form used in the model is. If this turns out to be a typo, then we would update the form in the database.