Package 'Rpadrino'

April 30, 2022

Title Interact with the 'PADRINO' IPM Database

Version 0.0.4

Description 'PADRINO' houses textual representations of
Integral Projection Models which can be converted from their
table format into full kernels to reproduce or extend an
already published analysis. 'Rpadrino' is an R interface to this database. For
more information on Integral Projection Models, see Easterling et al. (2000)
<doi:10.1890/0012-9658(2000)081[0694:SSSAAN]2.0.CO;2>, Merow et al. (2013)
<doi:10.1111/2041-210X.12146>, Rees et al. (2014) <doi:10.1111/1365-2656.12178>,
and Metcalf et al. (2015) <doi:10.1111/2041-210X.12405>. See Levin et al. (2021)
for more information on 'ipmr', the engine that powers model reconstruction
<doi:10.1111/2041-210X.13683>.

Depends R (>= 3.4.0), ipmr (>= 0.0.5)

Imports ggplot2, magrittr, mvtnorm, purrr, rlang (>= 0.3.0),
rmarkdown, tools, truncdist, utils

License GPL-3

Encoding UTF-8

LazyData true

BugReports https://github.com/padrinoDB/Rpadrino/issues

URL https://github.com/padrinoDB/Rpadrino,
https://padrinoDB.github.io/Rpadrino/

RoxygenNote 7.1.2

Suggests covr, knitr, maps, roxygen2, testthat

VignetteBuilder knitr

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation no

Author Sam Levin [aut, cre] (<https://orcid.org/0000-0002-3289-9925>),
Aldo Compagnoni [aut],
Dylan Childs [aut],

Sanne Evers [aut],
Tomos Potter [aut],
Roberto Salguero-Gomez [aut],
Tiffany Knight [aut]

Maintainer Sam Levin <levisc8@gmail.com>

Repository CRAN

Date/Publication 2022-04-29 23:20:14 UTC

R topics documented:

---

pdb *Selected models from the Padrino Database*

Description

Selected models from the Padrino Database

Usage

```
pdb
```

Format

A list of data frames, each corresponding to a different table in the Padrino Database.

pdb_citations *Access pieces of metadata from a* pdb *object*

Description

These functions access pieces of specific pieces metadata from the `Metadata` table of a pdb object. The exception is `pdb_report`, which automatically generates a report with summary statistics and citation information for the pdb object.

Usage

```
pdb_citations(pdb, ipm_id = NULL)

pdb_species_accepted(pdb, ipm_id = NULL)

pdb_species_author(pdb, ipm_id = NULL)

pdb_genus(pdb, ipm_id = NULL)

pdb_family(pdb, ipm_id = NULL)

pdb_order(pdb, ipm_id = NULL)

pdb_class(pdb, ipm_id = NULL)

pdb_phylum(pdb, ipm_id = NULL)

pdb_kingdom(pdb, ipm_id = NULL)

pdb_org_type(pdb, ipm_id = NULL)

pdb_dicot_monocot(pdb, ipm_id = NULL)

pdb_angio_gymon(pdb, ipm_id = NULL)

pdb_authors(pdb, ipm_id = NULL)

pdb_journal(pdb, ipm_id = NULL)

pdb_pub_year(pdb, ipm_id = NULL)

pdb_doi(pdb, ipm_id = NULL)

pdb_comments(pdb, ipm_id = NULL)

pdb_appendix_link(pdb, ipm_id = NULL)
```

```
pdb_duration(pdb, ipm_id = NULL)

pdb_start_year(pdb, ipm_id = NULL)

pdb_start_month(pdb, ipm_id = NULL)

pdb_end_year(pdb, ipm_id = NULL)

pdb_end_month(pdb, ipm_id = NULL)

pdb_periodicity(pdb, ipm_id = NULL)

pdb_population_name(pdb, ipm_id = NULL)

pdb_number_populations(pdb, ipm_id = NULL)

pdb_lat(pdb, ipm_id = NULL)

pdb_lon(pdb, ipm_id = NULL)

pdb_altitude(pdb, ipm_id = NULL)

pdb_country(pdb, ipm_id = NULL)

pdb_continent(pdb, ipm_id = NULL)

pdb_ecoregion(pdb, ipm_id = NULL)

pdb_studied_sex(pdb, ipm_id = NULL)

pdb_eviction_used(pdb, ipm_id = NULL)

pdb_evict_type(pdb, ipm_id = NULL)

pdb_treatment(pdb, ipm_id = NULL)

pdb_has_time_lag(pdb, ipm_id = NULL)

pdb_has_age(pdb, ipm_id = NULL)

pdb_report(
  pdb,
  title = "",
  keep_rmd = TRUE,
  rmd_dest = getwd(),
  output_format = "html",
  render_output = TRUE,
  map = TRUE,
```

```
  translate_eqs = FALSE,
  block_eqs = FALSE,
  long_eq_length = 65
)
```

Arguments

pdb	A Padrino Database object.
ipm_id	The ID of the model. The default (NULL) returns all values in the pdb object.
title	The title for the created report.
keep_rmd	Keep the un-rendered Rmd file? Useful for manual editing.
rmd_dest	The folder to save the Rmd file at if keep_rmd = TRUE. The default is getwd().
output_format	The output format to create. Options are "html", "pdf", "word", "odt", "rtf", or "md".
render_output	A logical - should the document be rendered for inspection?
map	Create a map of studies included in the pdb object?
translate_eqs	A logical - should the mathematical equations of the IPM(s) also be included in the report? These are translated from R to Latex by make_ipm_report_body. Currently, this is only available for IPMs that do not have parameter set indexed terms.
block_eqs	If report_eqs = TRUE, should equations be reported in block format or as inline equations? This main difference for "pdf" formats is that equation numbering is done with tag{}. For non-"pdf" formats, the difference is that equations are centered. Numbering may yield strange results for non-"pdf" formats.
long_eq_length	For longer equations, make_ipm_report tries to wrap these into multiple lines using \\. This parameter controls the number of characters per line. Default is 65. Ignored when block_eqs = FALSE.

Value

A named vector of the metadata. The names correspond to ipm_idss. For pdb_report, the file path to the rendered output, or to the .rmd file when render_output = FALSE.

---

pdb_download *Download PADRINO* pdb *objects*

Description

Download PADRINO from Github.

Usage

```
pdb_download(save = TRUE, destination = NULL)

pdb_save(pdb, destination = NULL)

pdb_load(path)
```

Arguments

save	Write the PDB object to a folder of text files?
destination	Where to write the pdb object to.
pdb	A pdb object.
path	The directory where the PADRINO tables are stored

Details

This does not currently support versioning because there is only one version. destination should be a folder name. When save = TRUE, a set of 12 text files will be saved in the destination folder. The files are tab-delimited.

Value

pdb_download and pdb_load return pdb objects. pdb_save returns a pdb object invisibly.

---

pdb_make_ipm *Generate IPMs from Padrino objects*

Description

This function generates complete IPMs from objects created with pdb_make_proto_ipm.

Usage

```
pdb_make_ipm(proto_ipm_list, addl_args = list())
```

Arguments

proto_ipm_list	Output from pdb_make_proto_ipm.
addl_args	A named list of additional arguments to pass to make_ipm.

Details

The format of addl_args should be a nested list. The names of the outermost level should correspond to the ipm_id that the arguments apply to. Each entry of the outermost level should itself then be a named list where the names correspond to arguments to make_ipm, and the values are the values for each argument. See examples.

Value

A list of IPMs.

Examples

```
## Not run:

data("pdb_ex")

proto <- pdb_make_proto_ipm(pdb_ex, ipm_id = "aaa341", det_stoch = "det")

ipm   <- pdb_make_ipm(proto)

proto <- pdb_make_proto_ipm(pdb_ex,
                            ipm_id     = "aaaa55",
                            det_stoch  = "stoch",
                            kern_param = "kern")

args  <-list(

 # The names in the outermost list should be ipm_id's

 aaaa55 = list(

   # The names in the inner list should be arguments to make_ipm()

   report_progress = TRUE,
   iterate         = TRUE,
   iterations      = 100,
   kernel_seq      = sample(2004:2014, 100, replace = TRUE)
 )
)

ipm   <- pdb_make_ipm(proto, addl_args = args)

## End(Not run)
```

`pdb_make_proto_ipm` *Generate proto_ipms from Padrino objects*

Description

This function generates `proto_ipm` objects from Padrino Database tables.

Usage

```
pdb_make_proto_ipm(pdb, ipm_id = NULL, det_stoch = "det", kern_param = "kern")
```

Arguments

pdb	A pdb object.

ipm_id	Optionally, one or more ipm_id's to build. If empty, all models contained in the pdb object will be processed into proto_ipm's.
det_stoch	A vector containing either "det" or "stoch". This determines whether we want to construct a deterministic or stochastic model. Default is "det". See details
kern_param	If det_stoch = "stoch", then whether or not to construct a kernel resampled model, or a parameter resampled model. See details.

Details

proto_ipm objects contain all of the information needed to implement an IPM, but stop short of actually generating kernels. These are intermediate building blocks that can be modified before creating a full IPM so that things like perturbation analysis are a bit more straightforward.

When requesting many models, the det_stoch and kern_param parameters can also be vectors. These are matched with ipm_id by position. If the lengths of det_stoch and kern_param do not match the length ipm_id, they will be recycled until they do.

For stochastic models, there is sometimes the option of building either a kernel-resampled or a parameter resampled model. A kernel resampled model uses some point estimate for time and/or space varying parameters to generate kernels for each year/site/grouping factor. Parameter resampled models sample parameters from distributions. Padrino stores this information for some models when it is available in the literature, and tries to fail informatively when these distributions aren't available in the database.

Value

A list containing one or more proto_ipms. Names of the list will correspond to ipm_ids.

See Also

For more info on kern_param definitions:

Metcalf *et al.* (2015). Statistial modeling of annual variation for inference on stochastic population dynamics using Integral Projection Models. *Methods in Ecology and Evolution*. DOI: 10.1111/2041-210X.12405

---

pdb_subset *Subset a Padrino database object*

Description

Subset a Padrino database object

Usage

```
pdb_subset(pdb, ipm_ids)
```

Arguments

pdb	A Padrino database object.
ipm_ids	The ipm_id's to subset the database to.

Details

Currently, the only variable to subset with is the ipm_id. Eventually, subsetting based on other variables will be possible with syntax similar to subset. At the moment, users will need to create a vector of ipm_ids based on searching and then pass that to subset. See Examples

Value

A new Padrino database object containing only the models specified in ipm_ids.

Examples

```
## Not run:

data(pdb)

poa_ind <- pdb$Metadata$ipm_id[pdb$Metadata$tax_family == "Poaceae"]

poa_db  <- pdb_subset(pdb, ipm_ids = poa_ind)


## End(Not run)
```

---

print.pdb *Print a* pdb *object.*

Description

Print a pdb object.

Usage

```
## S3 method for class 'pdb'
print(x, ...)

## S3 method for class 'pdb_proto_ipm_list'
print(x, ...)
```

Arguments

x	A pdb object.
...	Only used by pdb_new_fun_form, otherwise ignored. See details and examples for usage in pdb_new_fun_form.

Value

x invisibly.

`vital_rate_exprs.pdb_proto_ipm_list`
Padrino methods for 'ipmr' generic functions

Description

Provides wrappers around `ipmr` generic functions to extract some quantities of interest from `pdb_proto_ipm_lists` and `pdb_ipms`.

Usage

```
## S3 method for class 'pdb_proto_ipm_list'
vital_rate_exprs(object)

## S3 method for class 'pdb_ipm'
vital_rate_exprs(object)

## S3 method for class 'pdb_proto_ipm_list'
kernel_formulae(object)

## S3 method for class 'pdb_ipm'
kernel_formulae(object)

## S3 method for class 'pdb_proto_ipm_list'
domains(object)

## S3 method for class 'pdb_ipm'
domains(object)

## S3 method for class 'pdb_proto_ipm_list'
parameters(object)

## S3 method for class 'pdb_ipm'
parameters(object)

## S3 method for class 'pdb_proto_ipm_list'
pop_state(object)

## S3 method for class 'pdb_ipm'
pop_state(object)

## S3 method for class 'pdb_ipm'
vital_rate_funs(ipm)
```

```
## S3 method for class 'pdb_ipm'
int_mesh(ipm, full_mesh = TRUE)

## S3 method for class 'pdb_ipm'
lambda(ipm, ...)

## S3 method for class 'pdb_ipm'
right_ev(ipm, iterations = 100, tolerance = 1e-10, ...)

## S3 method for class 'pdb_ipm'
left_ev(ipm, iterations = 100, tolerance = 1e-10, ...)

## S3 method for class 'pdb_ipm'
is_conv_to_asymptotic(ipm, tolerance = 1e-10, burn_in = 0.1)

## S3 method for class 'pdb_ipm'
conv_plot(ipm, iterations = NULL, log = FALSE, show_stable = TRUE, ...)

## S3 method for class 'pdb_ipm'
make_iter_kernel(ipm, ..., name_ps = NULL, f_forms = NULL)

## S3 method for class 'pdb_ipm'
mean_kernel(ipm)

pdb_new_fun_form(...)

## S3 replacement method for class 'pdb_proto_ipm_list'
parameters(object, ...) <- value

## S3 replacement method for class 'pdb_proto_ipm_list'
vital_rate_exprs(object, kernel = NULL, vital_rate = NULL) <- value

## S3 replacement method for class 'pdb_proto_ipm_list'
kernel_formulae(object, kernel) <- value

## S3 method for class 'pdb_ipm'
x[i]
```

Arguments

object	An object produced by pdb_make_proto_ipm or pdb_make_ipm.
ipm	A pdb_ipm.
full_mesh	Logical. Return the complete set of meshpoints or only the unique ones.
...	Usage depends on the function - see Details and Examples.
iterations	The number of times to iterate the model to reach convergence. Default is 100.
tolerance	Tolerance to evaluate convergence to asymptotic dynamics.

burn_in	The proportion of iterations to discard as burn in when assessing convergence.
log	Log-transform lambdas for plotting?
show_stable	Show horizontal line denoting stable population growth?
name_ps	For pdb_ipm objects that contain age_x_size IPMs, a named list. The names of the list should be the ipm_ids that are age_x_size models, and the values in the list should be the the name of the survival/growth kernels.
f_forms	For pdb_ipm objects that contain age_x_size IPMs, a named list. The names of the list should be the ipm_ids that are age_x_size models, and the values in the list should be the the name of the fecundity kernels. If multiple sub-kernels contribute to fecundity, we can also supply a string specifying how they are combined (e.g. f_forms = "F + C").
value	The value to insert. See details and Examples.
kernel	Ignored, present for compatibility with ipmr.
vital_rate	Ignored, present for compatibility with ipmr.
x	A pdb_ipm object.
i	The index to extract

Details

There are number of uses for ... which depend on the function used for them. These are described below.

Value

Most of these return named lists where names correspond to ipm_ids. The exception is pdb_new_fun_form, which returns a list of expressions. It is only intended for setting new expressions with vital_rate_exprs<-.

pdb_new_fun_form

This must be used when setting new expressions for vital rates and kernel formulae. The ... argument should be a named list of named lists. The top most layer should be ipm_id's. The next layer should be a list where the names are vital rates you wish to modify, and the values are the expressions you want to insert. See examples.

make_iter_kernel

The ... here should be expressions representing the block kernel of the IPMs in question. The names of each expression should be the ipm_id, and the expressions should take the form of c(<upper_left>,<upper_right>, <lower_left>, <lower_right>) (i.e. a vector of symbols would create a matrix in row-major order). See examples.

conv_plot/lambda

The ... are used pass additional arguments to lambda and conv_plot.

Examples

```
data(pdb)
my_pdb <- pdb_make_proto_ipm(pdb, c("aaaa17", "aaa310"))

# These values will be appended to the parameter list for each IPM, as they
# aren't currently present in them.

parameters(my_pdb) <- list(
  aaa310 = list(
    g_slope_2 = 0.0001,
    establishment_prob = 0.02
  ),
  aaaa17 = list(
    g_var = 4.2,
    germ_prob = 0.3
  )
)

# We can overwrite a parameter value with a new one as well. Old values aren't
# saved anywhere except in the pdb object, so be careful!

parameters(my_pdb) <- list(
  aaa310 = list(
    s_s    = 0.93, # old value is 0.92
    gvar_i = 0.13 # old value is 0.127
  )
)

vital_rate_exprs(my_pdb) <- pdb_new_fun_form(
    list(
      aaa310 = list(mu_g = g_int + g_slope * size_1 + g_slope_2 * size_1^2),
      aaaa17 = list(sigmax2 = sqrt(g_var * exp(cfv1 + cfv2 * size_1))
     )
   )
 )

 kernel_formulae(my_pdb) <- pdb_new_fun_form(
   list(
     aaaa17 = list(Y = recr_size * yearling_s * germ_prob * d_size),
     aaa310 = list(F = f_n * f_d * establishment_prob)
   )
 )

 my_ipms    <- pdb_make_ipm(my_pdb)
 iter_kerns <- make_iter_kernel(my_ipms, aaaa17 = c(0, F_yr, Y, P_yr))
```

Index