Package 'Rcompadre'

October 17, 2024

Type Package

Title Utilities for using the 'COM(P)ADRE' Matrix Model Database

Version 1.4.0

Description Utility functions for interacting with the 'COMPADRE' and 'COMADRE' databases of matrix population models. Described in Jones et al. (2021) <doi:10.1101/2021.04.26.441330>.

License GPL-3

URL `https://github.com/jonesor/Rcompadre`

BugReports `https://github.com/jonesor/Rcompadre/issues`

Depends R (>= 2.10)

Imports methods, popdemo, stats, tibble, utils

Suggests dplyr, ggplot2, knitr, maps, rcrossref, rmarkdown, testthat

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel false

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.2.3

Collate 'CompadreDB.R' 'Rcompadre-package.R' 'CompadreDB-Methods.R' 'CompadreDB-Subsetting.R' 'CompadreDB-Tidyverse.R' 'CompadreMat.R' 'ClassUnionMethods.R' 'cdb_check_species.R' 'cdb_build_cdb.R' 'cdb_collapse.R' 'cdb_compare.R' 'cdb_fetch.R' 'cdb_flag.R' 'cdb_flatten.R' 'cdb_id.R' 'cdb_id_stages.R' 'cdb_id_studies.R' 'cdb_mean_matF.R' 'cdb_metadata.R' 'cdb_rbind.R' 'cdb_unflatten.R' 'cdb_unnest.R' 'data.R' 'mpm_elementwise_apply.R' 'mpm_mean.R' 'mpm_methods.R' 'mpm_sd.R' 'mpm_median.R' 'string_representation.R' 'zzz.R'

NeedsCompilation no

Author Patrick Barks [aut] (<https://orcid.org/0000-0002-5947-8151>),
Danny Buss [aut],
Roberto Salguero-Gomez [aut] (<https://orcid.org/0000-0002-6085-4433>),
Iain Stott [aut] (<https://orcid.org/0000-0003-2724-7436>),
William K. Petry [aut] (<https://orcid.org/0000-0002-5230-5987>),
Tamora James [aut] (<https://orcid.org/0000-0003-1363-4742>),
Owen Jones [aut, cre] (<https://orcid.org/0000-0001-5720-4686>),
Julia Jones [aut] (<https://orcid.org/0000-0001-9223-1778>),
Gesa Römer [aut] (<https://orcid.org/0000-0002-4859-5870>),
Sam Levin [aut] (<https://orcid.org/0000-0002-3289-9925>)

Maintainer Owen Jones <jones@biology.sdu.dk>

Repository CRAN

Date/Publication 2024-10-16 22:30:02 UTC

Contents

`as_cdb` *Convert legacy COM(P)ADRE database object to CompadreDB*

Description

Convert a legacy COM(P)ADRE database object (of class 'list') to a CompadreDB object.

Usage

```
as_cdb(from)
```

Arguments

`from`	A legacy COM(P)ADRE database

Value

A CompadreDB object

Author(s)

Iain M. Stott

Examples

```
Compadre <- as_cdb(CompadreLegacy)
```

`cdb_build_cdb` *Create a CompadreDB object from user-specified data*

Description

Creates a CompadreDB object from data provided by the user in the form of matrices and metadata. Users can provide either a list of A matrices (i.e. the whole matrix population model) or lists of process-based submatrices `matU`, `matF`and `matC`. In this latter case, we assume that `matA` = `matU` + `matF` + `matC`. If only one type of the submatrices are provided, the others are assumed to be 0. If only the A matrices are provided, the submatrices are recorded as ‘NA‘.

Usage

```
cdb_build_cdb(
  mat_a = NULL,
  mat_u = NULL,
  mat_f = NULL,
  mat_c = NULL,
  stages = NULL,
  version = NULL,
  metadata = NULL
)
```

Arguments

`mat_a`	A ‘list‘ of A matrices
`mat_u`	A ‘list‘ of U matrices (representing survival and growth)
`mat_f`	A ‘list‘ of F matrices (representing sexual reproduction)
`mat_c`	A ‘list‘ of C matrices (representing clonal reproduction)
`stages`	A ‘list‘ of stage definitions provided as ‘data.frame‘s that include two columns: ‘MatrixClassOrganized‘ and ‘MatrixClassAuthor‘. If this argument is not provided, numeric stage names are generated automatically
`version`	An optional string allowing users to add version information to their output object. If this argument is not provided the current date and time is used.
`metadata`	A ‘data.frame‘ of metadata associated with each matrix. Metadata should be provided by row in the same order as the matrices are placed in the lists.

Value

A valid CompadreDB object

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data acquisition: `cdb_fetch()`, `cdb_metadata()`

Examples

```
# If you only have A matrices

mat_a1 <- rbind(
  c(0.1, 1.9),
  c(0.5, 0.7)
)

mat_a2 <- rbind(
  c(0.2, 1.4, 2.3),
  c(0.6, 0.3, 1.1),
  c(0.2, 0.2, 1.5)
)

mat_a3 <- rbind(
  c(0.1, 2.1),
  c(0.3, 0.4)
)

# Place the matrices into a list
mat_a_list <- mget(ls(pattern = "mat_a[0-9]"))

my_compadre <- cdb_build_cdb(mat_a = mat_a_list, version = "testrun")
my_compadre


mat_u1 <- rbind(
  c(0.1, 0.0),
  c(0.5, 0.7)
)

mat_u2 <- rbind(
  c(0.2, 0.0, 0.0),
  c(0.6, 0.3, 1.1),
  c(0.2, 0.2, 1.5)
)

mat_f1 <- rbind(
  c(0.0, 1.9),
  c(0.0, 0.0)
)

mat_f2 <- rbind(
  c(0.0, 1.4, 2.3),
  c(0.0, 0.0, 0.0),
  c(0.0, 0.0, 0.0)
)

mat_u_list <- mget(ls(pattern = "mat_u[0-9]"))
mat_f_list <- mget(ls(pattern = "mat_f[0-9]"))

meta <- data.frame(idNum = 1:2, SpeciesAccepted = c("A", "B"), x = 4:5)

stageInfo <- list(
  data.frame(
    MatrixClassOrganized = rep("active", 2),
    MatrixClassAuthor = c("small", "large")
  ),
  data.frame(
    MatrixClassOrganized = rep("active", 3),
    MatrixClassAuthor = c("small", "medium", "large")
  )
)
```

```
my_compadre <- cdb_build_cdb(
  mat_u = mat_u_list, mat_f = mat_f_list,
  metadata = meta, stages = stageInfo
)
my_compadre

my_compadre <- cdb_build_cdb(
  mat_u = mat_u_list, mat_f = mat_f_list,
  metadata = meta
)
my_compadre
```

cdb_check_species *Check whether a COM(P)ADRE database contains one or more species of interest*

Description

Takes a vector of species names and checks whether those species are represented within a CompadreDB object. It outputs either a data frame depicting the species of interest and whether they occur in the database (TRUE/FALSE), or, if `return_db == TRUE`, a CompadreDB object subset to the species of interest.

Usage

```
cdb_check_species(cdb, species, return_db = FALSE)
```

Arguments

`cdb`	A CompadreDB object
`species`	Character vector of binomial species names, with the genus and specific epithet separated by either an underscore or a space (e.g. `c("Acipenser_fulvescens", "Borrelia_burgdorferi")`)
`return_db`	Logical argument indicating whether a database should be returned

Value

If `return_db == FALSE`, returns a data frame with a column of species names and a column indicating whether a species occurs in the database. If `return_db == TRUE`, returns a subset of `cdb` containing only those species within argument `species`.

Author(s)

Danny Buss <dlb50@cam.ac.uk>

Owen R. Jones <jones@biology.sdu.dk>

Rob Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

See Also

Other data checking: `cdb_collapse()`, `cdb_compare()`, `cdb_flag()`, `mpm_methods`

Examples

```
species <- c("Primula vulgaris", "Trillium ovatum", "Homo sapiens")
cdb_check_species(Compadre, species)
CompadreSubset <- cdb_check_species(Compadre, species, return_db = TRUE)
```

`cdb_collapse` *Collapse a COM(P)ADRE database by averaging matrices over levels of one or more grouping variables*

Description

Collapses a CompadreDB object by averaging matrices over levels of one or more grouping variables (e.g. SpeciesAuthor, MatrixPopulation).

For a given study and species, a COM(P)ADRE database may contain multiple matrices, reflecting different combinations of population, year, and/or treatment. Collapsing allows a user to obtain a single 'grand mean matrix' for each group of interest (e.g. MatrixPopulation), and therefore limit pseudoreplication.

All members of a group *must* have the same matrix dimension (consider adding MatrixDimension as a grouping variable). All members of a group *should* have the same ProjectionInterval and matrix stage class definitions (see `cdb_id_stages`). Note that Seasonal matrices should not be collapsed using this method (they should be matrix-multiplied rather than averaged).

Usage

```
cdb_collapse(cdb, columns)
```

Arguments

`cdb`	A CompadreDB object
`columns`	Vector of grouping variables to collapse over (corresponding to columns within `cdb`)

Details

Will give a warning if members of any group do not all share the same ProjectionInterval or stage class definitions, or if `cdb` contains any rows with a MatrixComposite value of "Seasonal".

Prior to collapsing, columns of class 'factor' will be coerced to 'character', and any list-column apart from `mat` will be removed.

Within a group, rows of a given column are collapsed as follows:

- `mat`: matrices are averaged using mpm_mean, and stage class definitions are taken from the first group member
- `MatrixComposite`: return original value if N = 1, else return "Collapsed"
- `Lat`: re-calculated by averaging Lat column (if available)
- `Lon`: re-calculated by averaging Lon column (if available)
- `SurvivalIssue`: re-calculated from the collapsed `mat` (`max(colSums(matU))`)
- `others`: if all elements equal return that unique value, else paste together all unique values separated by "; "

Value

A CompadreDB object

Author(s)

Patrick M. Barks <patrick.barks@gmail.com>

Owen R. Jones <jones@biology.sdu.dk>

See Also

cdb_id_stages

Other data checking: `cdb_check_species()`, `cdb_compare()`, `cdb_flag()`, `mpm_methods`

Examples

```
# filter out Seasonal matrices
CompSub <- subset(Compadre, MatrixComposite != "Seasonal")

# add column identifying unique stage class definitions
CompSub$id_stage <- cdb_id_stages(CompSub, "MatrixClassOrganized")

# collapse
CompCollapse <- cdb_collapse(CompSub, columns = c("id_stage"))
```

`cdb_compare` *Compare two versions or subsets of a COM(P)ADRE database*

Description

Prints a summary of the differences between two CompadreDB objects, including the number of species, studies, and matrices in each. If argument `verbose == TRUE`, additionally prints a list of the species and studies that are present in one database but not the other.

Usage

```
cdb_compare(cdb1, cdb2, verbose = FALSE)
```

Arguments

`cdb1, cdb2`	CompadreDB objects to compare
`verbose`	Logical argument indicating whether or not to return lots of detail

Value

NULL. Output is printed rather than returned.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data checking: `cdb_check_species()`, `cdb_collapse()`, `cdb_flag()`, `mpm_methods`

Examples

```
Compadre1 <- subset(Compadre, Continent == "Asia")
Compadre2 <- subset(Compadre, Continent == "Africa")

cdb_compare(Compadre1, Compadre2)
```

`cdb_fetch` *Fetch the COM(P)ADRE database from compadre-db.org or a local file*

Description

Fetches the current version of a COM(P)ADRE database from `https://compadre-db.org`, or load any version stored in a local .RData file.

Usage

```
cdb_fetch(cdb, version = NULL, flag = FALSE, userComment = NULL, quiet = FALSE)
```

Arguments

`cdb`	Either "comadre" or "compadre" (case insensitive) to fetch the most recent database from `https://compadre-db.org`, or a path to an existing COMPADRE database (i.e. .RData file) stored on the local machine.
`version`	Optional. The version number of a particular database to be downloaded e.g. "3.0.0". If this is not included (as default) the latest versions are downloaded.
`flag`	Logical argument where 'TRUE' will automatically run cdb_flag to add logical columns to the metadata to flag potential problems in the matrix population models. Default is 'FALSE'.

`userComment`	An optional string to enable users to add a comment as an attribute to the returned data frame. This could be useful for keeping track of multiple versions of the database. Accessed by ‘attributes(db)$comment‘.
`quiet`	A logical argument. If ‘TRUE‘ the download message that includes version information and link to the user agreement is suppressed. Default is ‘FALSE‘.

Value

A CompadreDB object

Note

The downloaded databases include a set of attributes accessible with ‘attributes‘. These include version information and date and time of creation.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Patrick M. Barks <patrick.barks@gmail.com>

See Also

Other data acquisition: `cdb_build_cdb()`, `cdb_metadata()`

Examples

```
## Not run:
# Download latest version of COMPADRE direct from the website
compadre <- cdb_fetch("compadre")

# Examine the attributes of the downloaded database
attributes(compadre)

# Download COMPADRE version 3.0.0 direct from the website
compadre <- cdb_fetch("compadre", version = "3.0.0")

# using file path to downloaded data
compadre <- cdb_fetch("data/COMPADRE_v.5.0.1.RData")

## End(Not run)
```

`cdb_flag` *Flag potential issues in matrices of a COM(P)ADRE database*

Description

Adds columns to the data slot of a ‘CompadreDB‘ object that flag potential problems in the matrix population models. These columns can subsequently be used to subset the database by logical argument.

Optional checks include:

- `check_NA_A`: missing values in ‘matA‘? Missing (‘NA‘) values in matrices prevent most calculations using those matrices.
- `check_NA_U`: missing values in ‘matU‘? Missing (‘NA‘) values in matrices prevent most calculations using those matrices.
- `check_NA_F`: missing values in ‘matF‘? Missing (‘NA‘) values in matrices prevent most calculations using those matrices.
- `check_NA_C`: missing values in ‘matC‘? Missing (‘NA‘) values in matrices prevent most calculations using those matrices.
- `check_zero_U`: ‘matU‘ all zeros (including ‘NA‘)? Submatrices composed entirely of zero values can be problematic. There may be good biological reasons for this phenomenon. For example, in the particular focal population in the particular focal year, there was truly zero survival recorded. Nevertheless, zero-value submatrices can cause some calculations to fail and it may be necessary to exclude them.
- `check_zero_F`: ‘matF‘ all zeros (including ‘NA‘)? Submatrices composed entirely of zero values can be problematic. There may be good biological reasons for this phenomenon. For example, in the particular focal population in the particular focal year, there was truly zero reproduction recorded. Nevertheless, zero-value submatrices can cause some calculations to fail and it may be necessary to exclude them.
- `check_zero_U_colsum`: Columns of ‘matU‘ that sum to zero imply that there is is no survival from that particular stage. This may be a perfectly valid parameterisation for a particular year/place but is biologically unreasonable in the longer term and users may wish to exclude problematic matrices from their analysis.
- `check_singular_U`: ‘matU‘ singular? Matrices are said to be singular if they cannot be inverted. Inversion is required for many matrix calculations and, therefore, singularity can cause some calculations to fail.
- `check_component_sum`: do ‘matU‘/‘matF‘/‘matC‘ submatrices sum to ‘matA‘ (see *Details*)? A complete MPM (‘matA‘) can be split into its component submatrices (i.e., ‘matU‘, ‘matF‘ and ‘matC‘). The sum of these submatrices should equal the complete MPM (i.e., ‘matA‘ = ‘matU‘ + ‘matF‘ + ‘matC‘). Sometimes, however, errors occur so that the submatrices do NOT sum to ‘matA‘. Normally, this is caused by rounding errors, but more significant errors are possible.
- `check_ergodic`: is ‘matA‘ ergodic (see `isErgodic`)? Some matrix calculations require that the MPM (‘matA‘) be ergodic. Ergodic MPMs are those where there is a single asymptotic stable state that does not depend on initial stage structure. Conversely, non-ergodic MPMs are those where there are multiple asymptotic stable states, which depend on initial stage structure. MPMs that are non-ergodic are usually biologically unreasonable, both in terms of their life cycle description and their projected dynamics. They cause some calculations to fail.
- `check_irreducible`: is ‘matA‘ irreducible (see `isIrreducible`)? Some matrix calculations require that the MPM (‘matA‘) be irreducible. Irreducible MPMs are those where parameterised transition rates facilitate pathways from all stages to all other stages. Conversely,

reducible MPMs depict incomplete life cycles where pathways from all stages to every other stage are not possible. MPMs that are reducible are usually biologically unreasonable, both in terms of their life cycle description and their projected dynamics. They cause some calculations to fail. Irreducibility is necessary but not sufficient for ergodicity.

- `check_primitive`: is ‘matA‘ primitive (see `isPrimitive`)? A primitive matrix is non-negative matrix that is irreducible and has only a single eigenvalue of maximum modulus. This check is therefore redundant due to the overlap with ‘check_irreducible‘ and ‘checkErdogic‘.
- `check_surv_gte_1`: does ‘matU‘ contains values that are equal to or greater than 1? Survival is bounded between 0 and 1. Values in excess of 1 are biologically unreasonable.

Usage

```
cdb_flag(
  cdb,
 checks = c("check_NA_A", "check_NA_U", "check_NA_F", "check_NA_C", "check_zero_U",
    "check_zero_F", "check_zero_C", "check_zero_U_colsum", "check_singular_U",
   "check_component_sum", "check_ergodic", "check_irreducible", "check_primitive",
    "check_surv_gte_1")
)
```

Arguments

`cdb`	A CompadreDB object
`checks`	Character vector specifying which checks to run. Defaults to all, i.e. `c("check_NA_A", "check_NA_U", "check_NA_F", "check_NA_C", "check_zero_U", "check_singular_U", "check_component_sum", "check_ergodic", "check_irreducible", "check_primitive", "check_surv_gte_1")`

Details

For the flag `check_component_sum`, a value of `NA` will be returned if the matrix sum of matU, matF, and matC consists only of zeros and/or `NA`, indicating that the matrix has not been split.

Value

Returns `cdb` with extra columns appended to the data slot (columns have the same names as the corresponding elements of `checks`) to indicate (TRUE/FALSE) whether there are potential problems with the matrices corresponding to a given row of the data.

Author(s)

Owen Jones <jones@biology.sdu.dk>

Julia Jones <juliajones@biology.sdu.dk>

Roberto Salguero-Gomez <rob.salguero@zoo.ox.ac.uk>

Danny Buss <dlb50@cam.ac.uk>

Patrick Barks <patrick.barks@gmail.com>

References

Stott, I., Townley, S., & Carslake, D. 2010. On reducibility and ergodicity of population projection matrix models. Methods in Ecology and Evolution. 1 (3), 242-252

See Also

Other data checking: `cdb_check_species()`, `cdb_collapse()`, `cdb_compare()`, `mpm_methods`

Examples

```
CompadreFlag <- cdb_flag(Compadre)

# only check whether matA has missing values, and whether matA is ergodic
CompadreFlag <- cdb_flag(Compadre, checks = c("check_NA_A", "check_ergodic"))
```

`cdb_flatten` *Convert a COM(P)ADRE database to a flat data frame with matrices and vectors stored in string representation*

Description

Converts a CompadreDB object to a flat data frame by extracting the data slot, and splitting the `mat` column into separate columns for each component (matrices `matA`, `matU`, `matF`, `matC`, and vectors `MatrixClassAuthor`, and `MatrixClassOrganized`). The component matrices and vectors within the six new columns are stored in string format so that the database can be written to a flat file format such as csv (see string_representation).

Usage

```
cdb_flatten(cdb)
```

Arguments

`cdb`	A CompadreDB object

Value

A data frame based on the data slot of `cdb`, but with the column `mat` replaced by six separate columns (for matrices `matA`, `matU`, `matF`, `matC`, and vectors `MatrixClassAuthor`, and `MatrixClassOrganized`), whose elements are matrices or vectors in string representation.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Patrick M. Barks <patrick.barks@gmail.com>

See Also

cdb_unflatten string_representation

Other data management: `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
CompadreFlat <- cdb_flatten(Compadre)
```

---

cdb_id — *Create integer identifiers for a COM(P)ADRE database corresponding to unique combinations of a given set of columns*

Description

Creates a vector of integer identifiers corresponding to the rows of a CompadreDB object, based on unique combinations of the elements in a given set of columns.

Usage

```
cdb_id(cdb, columns)
```

Arguments

cdb	A CompadreDB object
columns	Vector of column names from which unique combinations should be identified

Details

Identifiers are assigned by pasting together the relevant columns, assigning factor levels based on alphabetical order, and then converting the factor levels to integers.

Value

Vector of integer identifiers corresponding to the rows of cdb, based on unique combinations of the elements in `columns`.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
cdb_id(Compadre, columns = c("SpeciesAuthor", "MatrixTreatment"))
```

`cdb_id_stages` *Create integer identifiers for a COM(P)ADRE database corresponding to unique combinations of species and matrix stage class definitions*

Description

Creates a vector of integer identifiers corresponding to the rows of a CompadreDB object, based on unique combinations of the column 'SpeciesAccepted' and a list of matrix stage class definitions (either 'MatrixClassAuthor' or 'MatrixClassOrganized').

Usage

```
cdb_id_stages(cdb, stage_def = "MatrixClassAuthor")
```

Arguments

`cdb`	A CompadreDB object
`stage_def`	Whether to define matrix stage class based on "MatrixClassAuthor" or "MatrixClassOrganized" (see *Details*). Defaults to "MatrixClassAuthor".

Details

The vector 'MatrixClassOrganized' reflects standardized stage classes ('prop', 'active', or 'dorm'), whereas 'MatrixClassAuthor' reflects a description of the stage classes as defined by the original author (e.g. `c('Seedling', 'Medium rosette', 'Large (2 rosettes)', 'Flowering')`).

Because the 'MatrixClassAuthor' definitions are less standardized, they are more prone to typos that could lead to slight differences between stage descriptions of matrices that really do have the same stage classes (e.g. a set of matrices from a single study/species/population). Therefore, using 'MatrixClassAuthor' to define stage classes is potentially prone to mistakenly 'splitting' identifiers that should really be the same.

'MatrixClassOrganized' has the opposite problem. It's possible for two matrices from a given study to have the same stage definitions based on 'MatrixClassOrganized', but legitimately differ in stage definitions as defined by the author. Therefore, using 'MatrixClassAuthor' to define stage classes is potentially prone to mistakenly 'lumping' identifiers that should actually differ.

Because the majority of studies in COM(P)ADRE use a single set of stage definitions for all matrices, and typos are rare, results for the different stage definitions will usually be similar. Note, however, that the actual integers returned for the different stage definitions are likely to be very different (because they are based on alphabetical order).

Value

Vector of integer identifiers corresponding to the rows of `cdb`.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

cdb_id

Other data management: `cdb_flatten()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
cdb_id_stages(Compadre, stage_def = "MatrixClassOrganized")
```

---

cdb_id_studies *Create a vector of unique study identifiers from a COM(P)ADRE database*

Description

Creates a vector of integer study identifiers corresponding to the rows of a CompadreDB object, based on unique combinations of the columns 'Authors', 'Journal', 'YearPublication', and 'DOI_ISBN' (or optionally, a different set of columns supplied by the user).

Usage

```
cdb_id_studies(
  cdb,
  columns = c("Authors", "Journal", "YearPublication", "DOI_ISBN")
)
```

Arguments

`cdb`	A CompadreDB object
`columns`	Vector of column names from which unique combinations should be identified. Defaults to `c("Authors", "Journal", "YearPublication","DOI_ISBN")`.

Details

Identifiers are assigned by pasting together the relevant columns, assigning factor levels based on alphabetical order, and then converting the factor levels to integers.

Value

Vector of integer study identifiers corresponding to the rows of cdb, based on unique combinations of the elements in `columns`.

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

`cdb_id`

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
Compadre$StudyID <- cdb_id_studies(Compadre)
```

`cdb_mean_matF` *Calculate a population-specific mean fecundity matrix for each set of matrices in a COM(P)ADRE database*

Description

Takes a CompadreDB object and calculates a grand mean fecundity matrix for each unique population (a mean of all population-specific fecundity matrices, including fecundity matrices for which `MatrixComposite == 'Mean'`).

Populations are defined based on unique combinations of the columns 'SpeciesAuthor', 'MatrixPopulation', and 'MatrixDimension', (or optionally, a different set of columns supplied by the user).

The main purpose of this function is to identify stage classes that are *potentially* reproductive (i.e. the absence of fecundity in a given stage class and year does not necessarily indicate that the stage in question is non-reproductive).

Usage

```
cdb_mean_matF(
  cdb,
  columns = c("SpeciesAuthor", "MatrixPopulation", "MatrixDimension")
)
```

Arguments

`cdb`	A CompadreDB object
`columns`	Vector of column names from which unique populations should be identified. Defaults to `c("SpeciesAuthor", "MatrixPopulation","MatrixDimension")`.

Value

Returns a list of matrices, representing the mean fecundity matrix associated with each row of the database.

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Danny Buss <dlb50@cam.ac.uk>

Julia Jones <juliajones@biology.sdu.dk>

Iain Stott <stott@biology.sdu.dk>

Patrick Barks <patrick.barks@gmail.com>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
# print matF associated with row 16 of database
Compadre$mat[[16]]

# create list of meanMatFs
meanF <- cdb_mean_matF(Compadre)

# print meanMatF associated with row 16 of database
meanF[[16]]
```

---

cdb_metadata *Extract metadata from a COM(P)ADRE database*

Description

Extract a tibble with only metadata information from a CompadreDB object, by dropping the matrix column "mat".

Usage

```
cdb_metadata(cdb)
```

Arguments

cdb	A CompadreDB object

Details

Transforms the large CompadreDB object into a tibble and drops the matrix column ("mat").

Value

Tibble with all metadata columns of cdb

Author(s)

Gesa Romer <gesa.roemer@gmail.com>

See Also

Other data acquisition: `cdb_build_cdb()`, `cdb_fetch()`

Examples

```
Compadre_metadata <- cdb_metadata(Compadre)
```

`cdb_rbind` *Merge two COM(P)ADRE databases via row-bind*

Description

Merges two CompadreDB objects via a row-bind of the data slots.

Usage

```
cdb_rbind(cdb1, cdb2)
```

Arguments

`cdb1, cdb2`	CompadreDB objects

Value

A CompadreDB object created by binding the rows of `cdb1` and `cdb2`

Author(s)

Sam Levin <levisc8@gmail.com>

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
Compadre1 <- subset(Compadre, Continent == "Asia")
Compadre2 <- subset(Compadre, Continent == "Africa")

cdb_rbind(Compadre1, Compadre2)
```

cdb_unflatten — *Reconstitute a flattened COM(P)ADRE database (created by cdb_flatten) as a CompadreDB object*

Description

Converts a flattened COM(P)ADRE database (created by cdb_flatten) back to the CompadreDB class

Usage

```
cdb_unflatten(db)
```

Arguments

db	A data frame created with cdb_flatten, with columns for matrices `matA`, `matU`, `matF`, `matC`, and vectors `MatrixClassAuthor`, and `MatrixClassOrganized` in string representation.

Value

A CompadreDB object. Because version details are lost when the database is flattened, the `Version` and `DateCreated` elements of the returned CompadreDB object will be `NA`.

Author(s)

Patrick M. Barks <patrick.barks@gmail.com>

See Also

cdb_flatten string_representation

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
CompadreFlat <- cdb_flatten(Compadre) # flatten
Compadre2 <- cdb_unflatten(CompadreFlat) # reconstitute
```

cdb_unnest — *Unnest a COM(P)ADRE database by spreading the components of CompadreMat into separate list-columns*

Description

Unnests a CompadreDB object by spreading the components of CompadreMat into separate list-columns. Components that may be extracted include:

- `matA` (matrix)
- `matU` (matrix)
- `matF` (matrix)
- `matC` (matrix)
- `MatrixClassAuthor` (character vector)
- `MatrixClassOrganized` (character vector)
- `MatrixClassNumber` (integer vector)

Usage

```
cdb_unnest(
  cdb,
  components = c("matA", "matU", "matF", "matC", "MatrixClassAuthor",
    "MatrixClassOrganized", "MatrixClassNumber")
)
```

Arguments

`cdb`	A CompadreDB object
`components`	Character vector specifying which components to extract. Defaults to all, i.e. `c("matA", "matU", "matF", "matC", "MatrixClassAuthor", "MatrixClassOrganized", "MatrixClassNumber")`

Value

cdb with additional list-columns for each element of argument `components`

Author(s)

Patrick M. Barks <patrick.barks@gmail.com>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
# unnest all components
CompadreUnnest <- cdb_unnest(Compadre)

# unnest select components (matU and MatrixClassAuthor)
CompadreUnnest <- cdb_unnest(Compadre, c("matU", "MatrixClassAuthor"))
```

Compadre *Subsamples of the COMPADRE Plant Matrix Database and COMADRE Animal Matrix Database for testing and examples*

Description

`Compadre` (plant matrices) and `Comadre` (animal matrices) are subsamples of the COMPADRE Plant Matrix Database and COMADRE Animal Matrix Database, respectively, that are used for testing and examples. Each object is of class 'CompadreDB' and therefore has the following two slots: `data` and `version`.

For full documentation see the COMPADRE User Guide.

Usage

```
Compadre

Comadre
```

Format

Slot **data** - A tibble-style data frame with the following 48 columns:

- mat - A list of 'CompadreMat' objects, each with the following slots:
  - matA - A matrix population model
  - matU - The survival- and growth-related component of matA
  - matF - The sexual reproduction component of matA
  - matC - The clonal reproduction component of matA
  - matrixClass - A data frame with the following columns:
    - MatrixClassOrganized - Standardized stage class of the matrix population model
    - MatrixClassAuthor - Stage description from the original publication
    - MatrixClassNumber - Integer stage number
- SpeciesAuthor - Binomial species name given by the paper's author
- SpeciesAccepted - Accepted binomial species name taken from *The Plant List* or *Catalogue of Life*
- CommonName - Common name for species used in the publication
- Genus - Taxonomic genus that the accepted species belongs to

- Family - Family that the species belongs to
- Order - Order that the species belongs to
- Class - Class that the species belongs to
- Phylum - Phylum that the species belongs to
- Kingdom - Kingdom that the species belongs to
- OrganismType - Organism type (see COMPADRE User Guide for documentation)
- DicotMonoc - Whether the species is a dicot or monocot. Non-angiosperms are `NA`.
- AngioGymno - Whether the species is an angiosperm or gymnosperm. Non-plants are `NA`.
- Authors - Last name of all authors (separated with ";")
- Journal - Abbreviated journal title, or type of source document (e.g. "PhD thesis")
- YearPublication - Year of publication
- DOI.ISBN - Digital Object Identifier or International Standard Book Number codes to identify each publication
- AdditionalSource - Additional source(s) used to reconstruct the matrix or obtain additional metadata for the matrix (if applicable)
- StudyDuration - Number of years of observation in study (`StudyEnd - StudyStart`)
- StudyStart - Study start year
- StudyEnd - Study end year
- AnnualPeriodicity - Inverse of the length of the projection interval (in years)
- NumberPopulations - The number of study populations as defined by the authors. Within site replication of permanent plots is not defined as separate populations
- MatrixCriteriaSize - Indicates whether the matrix contains stages based on size. If so, indicates what that measure of size is
- MatrixCriteriaOntogeny - Indicates whether the matrix contains stages based on ontogenic/developmental stages
- MatrixCriteriaAge - Indicates whether the matrix contains stages based on age
- MatrixPopulation - Population name or definition of where the matrix was recorded, usually given by the author. See COMPADRE User Guide.
- Lat - Latitude in decimal degrees
- Lon - Longitude in decimal degrees
- Altitude - Altitude of study site (m above sea level)
- Country - 3-letter ISO country code for the country in which the study took place (multiple countries separated with ";")
- Continent - Continent on which study took place
- Ecoregion - Ecoregion in which study took place. See COMPADRE User Guide.
- StudiedSex - Whether study included only males ("M"), only females ("F"), or both sexes ("M/F")
- MatrixComposite - Indicates the type of matrix population model. Possible values are `Individual`, `Mean`, `Pooled`, and `Seasonal`. See COMPADRE User Guide.

- MatrixTreatment - Describes if a treatment was applied to the population or not. If yes, includes a brief description of the treatment. If not, `Unmanipulated`
- MatrixCaptivity - Whether species was studied in the wild (`W`), captivity (`C`), or captured from a wild population (`CW`)
- MatrixStartYear - First year of matrix
- MatrixStartSeason - First season of matrix as described by author (hemisphere-specific)
- MatrixStartMonth - First month of matrix
- MatrixEndYear - Final year of matrix
- MatrixEndSeason - Final season of matrix as described by author (hemisphere-specific)
- MatrixEndMonth - Final month of matrix
- MatrixSplit - Whether the **A** matrix has been split into components **U**, **F**, and **C** ("Divided") or not ("Indivisible"). If not, elements of `matU`, `matF`, and `matC` are filled with `NA`
- MatrixFec - Whether fecundity was measured for the matrix model
- Observation - Additional observations recorded by database compilers
- MatrixDimension - Dimension of the **A** matrix
- SurvivalIssue - Denotes the maximum stage-specific survival value

Slot **version** - A list with the following elements:

- Version - The version number of the database
- DateCreated - The date that the `.RData` file was created
- Agreement - Link to the COMADRE license agreement

An object of class `CompadreDB` with 150 rows and 49 columns.

An object of class `CompadreDB` with 150 rows and 49 columns.

`CompadreDB` *CompadreDB Class*

Description

This page describes the CompadreDB class, including methods for accessing the slots (see functions `CompadreData` and `VersionData`), accessing (`$`) and replacing (`$<-`) columns within the `data` slot, accessing elements from the `version` slot (see functions `VersionData` and `DateCreated`), and converting legacy database objects to the CompadreDB class (see `as_cdb`).

Usage

```
CompadreData(object)

## S4 method for signature 'CompadreDB'
CompadreData(object)

## S4 method for signature 'CompadreDB'
x$name

## S4 replacement method for signature 'CompadreDB'
x$name <- value

## S4 method for signature 'CompadreDB,ANY,missing'
x[[i, j, ...]]

## S4 replacement method for signature 'CompadreDB,ANY,missing'
x[[i, j]] <- value

VersionData(object)

## S4 method for signature 'CompadreDB'
VersionData(object)

Version(object)

## S4 method for signature 'CompadreDB'
Version(object)

DateCreated(object)

## S4 method for signature 'CompadreDB'
DateCreated(object)
```

Arguments

`object`	A CompadreDB object
`x`	A CompadreDB object
`name`	The name of a column within x
`value`	Vector of values to assign to the column
`i, j`	elements to extract or replace (see [[.data.frame)
`...`	ignored

Slots

`data` A tibble-style data frame with a list-column of matrix population models (column `mat`) and a variety of other metadata columns.

version A list with elements Version (database version number), DateCreated (date of version release), and Agreement (a url link to the User Agreement)

Author(s)

Iain M. Stott

Tamora D. James

See Also

CompadreDB-Methods CompadreDB-Subsetting

Examples

```
# extract entire 'data' slot
dat <- CompadreData(Compadre)

# access the date of database creation
DateCreated(Compadre)

# extract column SpeciesAccepted
Compadre$SpeciesAccepted

# create new list-column with stage-specific survival
Compadre$stage_survival <- lapply(Compadre$mat, function(x) colSums(x@matU))
```

CompadreDB-Methods *Methods for CompadreDB objects*

Description

This page describes a variety of methods that can be used with CompadreDB objects, including common data frame operations (head, names, and merge), conversion methods (as.data.frame and as_tibble), and methods to calculate the number of species (NumberAcceptedSpecies), studies (NumberStudies), or matrices (NumberMatrices).

Usage

```
## S3 method for class 'CompadreDB'
as.data.frame(x, ...)

## S3 method for class 'CompadreDB'
as_tibble(
  x,
  .rows = NULL,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  rownames = NULL,
  ...
)

## S3 method for class 'CompadreDB'
head(x, n = 6L, ...)

## S3 method for class 'CompadreDB'
tail(x, n = 6L, ...)

## S3 method for class 'CompadreDB'
names(x)

## S3 method for class 'CompadreDB'
dim(x)

## S3 method for class 'CompadreDB'
merge(x, y, ...)

NumberAcceptedSpecies(object)

## S4 method for signature 'CompadreDB'
NumberAcceptedSpecies(object)

NumberStudies(object)

## S4 method for signature 'CompadreDB'
NumberStudies(object)

NumberMatrices(object)

## S4 method for signature 'CompadreDB'
NumberMatrices(object)
```

Arguments

`x, object`	A CompadreDB object
`...`	additional arguments
`.rows`	passed to [tibble::as_tibble()]
`.name_repair`	passed to [tibble::as_tibble()]
`rownames`	passed to [tibble::as_tibble()]
`n`	The number of rows to extract
`y`	A data.frame to merge with x

Value

No return value, called for side effects

CompadreDB-Subsetting *Subsetting CompadreDB objects*

Description

CompadreDB objects can be subset just like a regular data.frame, using either [or subset(). Note, however, that the mat column will always be retained during subsetting, even if it is not included in the user's column subset.

Usage

```
## S4 method for signature 'CompadreDB,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

## S3 method for class 'CompadreDB'
subset(x, subset, select, drop = FALSE, ...)
```

Arguments

x	A CompadreDB object
i	row indices (see [.data.frame)
j	column indices (see [.data.frame)
...	ignored
drop	ignored
subset	logical expression indicating which rows to keep
select	expression indicating which columns to keep

Value

No return value, called for side effects

Examples

```
# subset to the first 10 rows
Compadre[1:10, ]

# subset to the species 'Echinacea angustifolia'
subset(Compadre, SpeciesAccepted == "Echinacea angustifolia")

# remove the column SurvivalIssue
Compadre[, names(Compadre) != "SurvivalIssue"]

## Not run:
# column selection doesn't include mat, but mat will still be returned with a
#  along with a warning
subset(Compadre, select = c("SpeciesAccepted", "Authors"))

## End(Not run)
```

CompadreDB-Tidyverse *Tidyverse methods for CompadreDB objects*

Description

CompadreDB methods for functions in dplyr and ggplot2.

Usage

```
fortify.CompadreDB(model, data, ...)

filter.CompadreDB(.data, ...)

slice.CompadreDB(.data, ...)

arrange.CompadreDB(.data, ...)

mutate.CompadreDB(.data, ...)

group_by.CompadreDB(.data, ..., add = FALSE)

ungroup.CompadreDB(x, ...)

summarize.CompadreDB(.data, ...)

summarise.CompadreDB(.data, ...)

select.CompadreDB(.data, ...)

rename.CompadreDB(.data, ...)

left_join.CompadreDB(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...
)

right_join.CompadreDB(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...
)

inner_join.CompadreDB(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...
)

full_join.CompadreDB(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...
)
```

Arguments

`data`	see `fortify`
`...`	other arguments
`add`	Logical indicating whether to overwrite existing groups (`FALSE`) or add to any existing groups (`TRUE`)
`x, model, .data`	A CompadreDB object
`y`	see `join`
`by`	see `join`
`copy`	see `join`
`suffix`	see `join`

Value

No return value, called for side effects

`CompadreLegacy` *Subsample of a legacy version of the COMPADRE Plant Matrix Database for testing and examples*

Description

`CompadreLegacy` is a subsample of the COMPADRE Plant Matrix Database in the legacy format (class 'list'), for use in testing and examples. For full documentation see the COMPADRE User Guide.

Usage

```
CompadreLegacy
```

Format

A list with four elements:

metadata - A data frame with the following 47 columns:

- SpeciesAuthor - Binomial species name given by the paper's author
- SpeciesAccepted - Accepted binomial species name taken from *The Plant List* or *Catalogue of Life*
- CommonName - Common name for species used in the publication
- Genus - Taxonomic genus that the accepted species belongs to
- Family - Family that the species belongs to
- Order - Order that the species belongs to
- Class - Class that the species belongs to
- Phylum - Phylum that the species belongs to
- Kingdom - Kingdom that the species belongs to
- OrganismType - Organism type (see COMPADRE User Guide for documentation)
- DicotMonoc - Whether the species is a dicot or monocot. Non-angiosperms are `NA`.
- AngioGymno - Whether the species is an angiosperm or gymnosperm. Non-plants are `NA`.
- Authors - Last name of all authors (separated with ";")
- Journal - Abbreviated journal title, or type of source document (e.g. "PhD thesis")
- YearPublication - Year of publication
- DOI.ISBN - Digital Object Identifier or International Standard Book Number codes to identify each publication (note that the name has changed to DOI_ISBN in later database versions.)
- AdditionalSource - Additional source(s) used to reconstruct the matrix or obtain additional metadata for the matrix (if applicable)
- StudyDuration - Number of years of observation in study (`StudyEnd - StudyStart`)
- StudyStart - Study start year
- StudyEnd - Study end year
- AnnualPeriodicity - Inverse of the length of the projection interval (in years). Note that the name has changed to ProjectionInterval in later database versions.
- NumberPopulations - The number of study populations as defined by the authors. Within site replication of permanent plots is not defined as separate populations
- MatrixCriteriaSize - Indicates whether the matrix contains stages based on size. If so, indicates what that measure of size is
- MatrixCriteriaOntogeny - Indicates whether the matrix contains stages based on ontogenic/developmental stages
- MatrixCriteriaAge - Indicates whether the matrix contains stages based on age

- MatrixPopulation - Population name or definition of where the matrix was recorded, usually given by the author. See COMPADRE User Guide.
- Lat - Latitude in decimal degrees
- Lon - Longitude in decimal degrees
- Altitude - Altitude of study site (m above sea level)
- Country - 3-letter ISO country code for the country in which the study took place (multiple countries separated with ";")
- Continent - Continent on which study took place
- Ecoregion - Ecoregion in which study took place. See COMPADRE User Guide.
- StudiedSex - Whether study included only males ("M"), only females ("F"), or both sexes ("M/F")
- MatrixComposite - Indicates the type of matrix population model. Possible values are `Individual`, `Mean`, `Pooled`, and `Seasonal`. See COMPADRE User Guide.
- MatrixTreatment - Describes if a treatment was applied to the population or not. If yes, includes a brief description of the treatment. If not, `Unmanipulated`
- MatrixCaptivity - Whether species was studied in the wild (`W`), captivity (`C`), or captured from a wild population (`CW`)
- MatrixStartYear - First year of matrix
- MatrixStartSeason - First season of matrix as described by author (hemisphere-specific)
- MatrixStartMonth - First month of matrix
- MatrixEndYear - Final year of matrix
- MatrixEndSeason - Final season of matrix as described by author (hemisphere-specific)
- MatrixEndMonth - Final month of matrix
- MatrixSplit - Whether the **A** matrix has been split into components **U**, **F**, and **C** ("Divided") or not ("Indivisible"). If not, elements of `matU`, `matF`, and `matC` are filled with `NA`
- MatrixFec - Whether fecundity was measured for the matrix model
- Observation - Additional observations recorded by database compilers
- MatrixDimension - Dimension of the **A** matrix
- SurvivalIssue - Denotes the maximum stage-specific survival value

mat - A list of population projection models, which are also in list format. Each list element contains four matrices:

- matA - A matrix population model
- matU - The survival- and growth-related component of matA
- matF - The sexual reproduction component of matA
- matC - The clonal reproduction component of matA

matrixClass - A list of data frames, each with the following columns:

- MatrixClassOrganized - Standardized stage class of the matrix population model
- MatrixClassAuthor - Stage description from the original publication

- MatrixClassNumber - Integer stage number

version - A list with the following elements:

- Version - The version number of the database
- DateCreated - The date that the `.RData` file was created
- NumberAcceptedSpecies - The number of accepted species in the original version
- NumberStudies - The number of studies in the original version
- NumberMatrices - The number of matrices in the original version
- Agreement - Link to the COMADRE license agreement

`CompadreMatrixMethods` *Methods for working with matrices in com(p)adre*

Description

This page describes methods for accessing any matrix information from CompadreMat and CompadreDB objects.

Most methods for working with matrices are applicable to both CompadreMat and CompadreDB objects. These are described on this page (along with a couple) of methods applicable to only CompadreMat or CompadreDB objects).

Usage

```
matA(object)

## S4 method for signature 'CompadreMat'
matA(object)

## S4 method for signature 'CompadreDB'
matA(object)

## S4 method for signature 'list'
matA(object)

matU(object)

## S4 method for signature 'CompadreMat'
matU(object)

## S4 method for signature 'CompadreDB'
matU(object)

## S4 method for signature 'list'
matU(object)
```

```
matF(object)

## S4 method for signature 'CompadreMat'
matF(object)

## S4 method for signature 'CompadreDB'
matF(object)

## S4 method for signature 'list'
matF(object)

matC(object)

## S4 method for signature 'CompadreMat'
matC(object)

## S4 method for signature 'CompadreDB'
matC(object)

## S4 method for signature 'list'
matC(object)

matrixClass(object)

## S4 method for signature 'CompadreMat'
matrixClass(object)

## S4 method for signature 'CompadreDB'
matrixClass(object)

## S4 method for signature 'list'
matrixClass(object)

MatrixClassAuthor(object)

## S4 method for signature 'CompadreMat'
MatrixClassAuthor(object)

## S4 method for signature 'CompadreDB'
MatrixClassAuthor(object)

## S4 method for signature 'list'
MatrixClassAuthor(object)

MatrixClassOrganized(object)

## S4 method for signature 'CompadreMat'
MatrixClassOrganized(object)
```

```
## S4 method for signature 'CompadreDB'
MatrixClassOrganized(object)

## S4 method for signature 'list'
MatrixClassOrganized(object)

MatrixClassNumber(object)

## S4 method for signature 'CompadreMat'
MatrixClassNumber(object)

## S4 method for signature 'CompadreDB'
MatrixClassNumber(object)

## S4 method for signature 'list'
MatrixClassNumber(object)
```

Arguments

`object`	A CompadreDB object

Slots

- `matA` A matrix population model (i.e. a square projection matrix)
- `matU` The survival component of a matrix population model (i.e. a square projection matrix reflecting survival-related transitions; e.g. progression, stasis, and retrogression)
- `matF` The sexual component of a matrix population model (i.e. a square projection matrix reflecting transitions due to sexual reproduction)
- `matC` The clonal component of a matrix population model (i.e. a square projection matrix reflecting transitions due to clonal reproduction)
- `matrixClass` A data frame with columns `MatrixClassOrganized` (elements are "active", "prop", or "dorm") `MatrixClassAuthor` (the matrix author's stage description), and `MatrixClassNumber` (integer stage number)

`mpm_elementwise_apply` *Apply a function element-wise to a list of matrices*

Description

This function applies a specified function element-wise to the corresponding elements across a list of matrices.

Usage

```
mat_elementwise_apply(x, fun, na_handling = "stop", ...)

mpm_elementwise_apply(x, fun, na_handling = "stop", ...)
```

Arguments

`x`	A list of matrices.
`fun`	The function to apply to the elements.
`na_handling`	A character string specifying how to handle NA values. Possible values are "stop" (throw an error when NA values are encountered), "zero" (convert NA values to 0), and "ignore" (NA values are ignored and passed to ‘fun‘). Handling can then be processed appropriately by that function (e.g., with ‘na.rm‘).
`...`	Additional arguments passed to ‘fun‘.

Value

A matrix containing the result of applying the function element-wise to the corresponding elements across the matrices.

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
mpms <- Compadre$mat[Compadre$SpeciesAuthor == "Haplopappus_radiatus"]

#The object mpms is a list, containing compadre objects
class(mpms)
class(mpms[[1]])

# Get the mean, max and min for the matrices
mpm_elementwise_apply(mpms, mean)
mpm_elementwise_apply(mpms, max)
mpm_elementwise_apply(mpms, min)

# extract list of matA and take mean
mats <- matA(mpms)
mat_elementwise_apply(mats, mean)

# This should be the same as mat_mean()
mat_mean(mats)

# Mean values, with 25% trimmed from each end
mat_elementwise_apply(mats, mean, trim = 0.25)

# weighted mean, where the second matrix is weighted to 100% and the others to 0%
# do demonstrate usage. The result should be the same as mats[[2]]
mat_elementwise_apply(mats, weighted.mean, w = c(0,1,0,0))
mats[[2]]

#min and max values
mat_elementwise_apply(mats, min)
mat_elementwise_apply(mats, max)
```

```
#Demonstrating NA handling
#First adding some NA values to the matrices
mats[[2]][3,2] <- NA

#replace the NA with a 0
mat_elementwise_apply(mats, min, na_handling = "zero")

#ignore the NA
mat_elementwise_apply(mats, min, na_handling = "ignore")

#ignore the NA, but pass na.rm = TRUE to the function (min)
mat_elementwise_apply(mats, min, na_handling = "ignore", na.rm = TRUE)
```

`mpm_mean` *Calculate a mean over a list of matrices or CompadreMat objects*

Description

Calculates an element-wise mean over a list of matrices or CompadreMat objects of constant dimension.

The difference between function `mat_mean`) and (`mpm_mean` is that `mat_mean` takes input as a list of matrices (e.g., a list of **A** matrices) while `mat_mean` takes input as a list of ‘CompadreMat‘ objects and thus calculates the mean matrices for both the **A** matrix and its submatrices (**U**, **F**, **C**).

Usage

```
mat_mean(x, na.rm = FALSE)

mpm_mean(x, na.rm = FALSE)
```

Arguments

`x`	A list of matrices or, for `mpm_mean` a list of ‘CompadreMat‘ objects, all of the same dimension
`na.rm`	Logical indicating whether missing values should be excluded (see *Details*). Defaults to `FALSE`.

Details

If `na.rm` == `TRUE`, missing values are ignored in the calculation of the mean matrix. If `na.rm` == `TRUE` and a given element is `NA` in *every* matrix within `x`, the value returned for that element will be `0`.

Value

A matrix (`mat_mean`) or a CompadreMat object (`mpm_mean`).

Author(s)

Patrick Barks <patrick.barks@gmail.com>

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_median()`, `mpm_sd()`, `string_representation`

Examples

```
# there are four rows for species 'Haplopappus_radiatus' in Compadre
mpms <- Compadre$mat[Compadre$SpeciesAuthor == "Haplopappus_radiatus"]

#The object mpms is a list, containing compadre objects
class(mpms)
class(mpms[[1]])

mpm_mean(mpms)

# extract list of matA and take mean
mats <- matA(mpms)
mat_mean(mats)
```

`mpm_median` *Calculate a median over a list of matrices or CompadreMat objects*

Description

Calculates an element-wise median over a list of matrices or CompadreMat objects of constant dimension.

Usage

```
mat_median(x, na.rm = FALSE)

mpm_median(x, na.rm = FALSE)
```

Arguments

`x`	A list of matrices or, for `mpm_sd` a list of ‘CompadreMat‘ objects, all of the same dimension
`na.rm`	Logical indicating whether missing values should be excluded (see *Details*). Defaults to `FALSE`.

Details

The difference between function `mat_median`) and (`mpm_median` is that `mat_median` takes input as a list of matrices (e.g., a list of **A** matrices) while `mat_median` takes input as a list of ‘CompadreMat‘ objects and thus calculates the mean matrices for both the **A** matrix and its submatrices (**U**, **F**, **C**).

If `na.rm` == `TRUE`, missing values are ignored in the calculation of the mean matrix. If `na.rm` == `TRUE` and a given element is `NA` in *every* matrix within `x`, the value returned for that element will be `0`.

Value

A matrix containing the median of each element across all matrices in the list

Author(s)

Darren Norris

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_sd()`, `string_representation`

Examples

```
# set seed for repeatability
set.seed(42)

# create a function that generates a matrix with random values
create_matrix <- function() {
  matrix(runif(9, 0, 1), nrow = 3)
}

# use replicate() to call the create_matrix() function 20 times
mat_list <- replicate(20, create_matrix(), simplify = FALSE)

# get the median matrix
mat_median(mat_list)

# If the matrices are in an RCompadre object, extract them using `matA` before
# passing to `mat_median`
my_compadre <- cdb_build_cdb(mat_a = mat_list)
mat_median(matA(my_compadre))
```

mpm_methods *Extract stage-class information from CompadreMat or CompadreDB objects*

Description

Methods for extracting stage-class information from CompadreMat or CompadreDB objects, including whether the matrix population model includes one or more propagule stages (`mpm_has_prop`), dormant stages (`mpm_has_dorm`), or active stages (`mpm_has_active`), and the integer index of the first active stage class (`mpm_first_active`).

These methods will return a single value if passed a CompadreMat object, or a vector of values if passed a CompadreDB object (one value for every CompadreMat object within the column 'mat').

Usage

```
mpm_has_prop(object)

## S4 method for signature 'CompadreMat'
mpm_has_prop(object)

## S4 method for signature 'CompadreDB'
mpm_has_prop(object)

mpm_has_active(object)

## S4 method for signature 'CompadreMat'
mpm_has_active(object)

## S4 method for signature 'CompadreDB'
mpm_has_active(object)

mpm_has_dorm(object)

## S4 method for signature 'CompadreMat'
mpm_has_dorm(object)

## S4 method for signature 'CompadreDB'
mpm_has_dorm(object)

mpm_first_active(object)

## S4 method for signature 'CompadreMat'
mpm_first_active(object)

## S4 method for signature 'CompadreDB'
mpm_first_active(object)
```

Arguments

`object`	A CompadreMat or CompadreDB object

Value

No return value, called for side effects

Author(s)

Patrick Barks <patrick.barks@gmail.com>

See Also

Other data checking: `cdb_check_species()`, `cdb_collapse()`, `cdb_compare()`, `cdb_flag()`

Examples

```
# with CompadreMat object
mpm_has_prop(Compadre$mat[[1]])
mpm_has_active(Compadre$mat[[1]])
mpm_has_dorm(Compadre$mat[[1]])
mpm_first_active(Compadre$mat[[1]])

# with CompadreDB object
mpm_has_prop(Compadre)
mpm_has_active(Compadre)
mpm_has_dorm(Compadre)
mpm_first_active(Compadre)
```

`mpm_sd` *Calculate a standard deviation over a list of matrices or CompadreMat objects*

Description

Calculates an element-wise standard deviation over a list of matrices or CompadreMat objects of constant dimension.

Usage

```
mat_sd(x, na.rm = FALSE)

mpm_sd(x, na.rm = FALSE)
```

Arguments

`x`	A list of matrices or, for `mpm_sd` a list of ‘CompadreMat‘ objects, all of the same dimension
`na.rm`	Logical indicating whether missing values should be excluded (see *Details*). Defaults to `FALSE`.

Details

The difference between function `mat_sd`) and (`mpm_sd` is that `mat_sd` takes input as a list of matrices (e.g., a list of **A** matrices) while `mat_sd` takes input as a list of ‘CompadreMat‘ objects and thus calculates the mean matrices for both the **A** matrix and its submatrices (**U**, **F**, **C**).

If `na.rm` == `TRUE`, missing values are ignored in the calculation of the mean matrix. If `na.rm` == `TRUE` and a given element is `NA` in *every* matrix within `x`, the value returned for that element will be `0`.

Value

A matrix containing the standard deviation of each element across all matrices in the list

Author(s)

Darren Norris

Owen R. Jones <jones@biology.sdu.dk>

See Also

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `string_representation`

Examples

```
# set seed for repeatability
set.seed(42)

# create a function that generates a matrix with random values
create_matrix <- function() {
  matrix(runif(9, 0, 1), nrow = 3)
}

# use replicate() to call the create_matrix() function 20 times
mat_list <- replicate(20, create_matrix(), simplify = FALSE)

# get the sd matrix
mat_sd(mat_list)

# If the matrices are in an RCompadre object, extract them using `matA` before
# passing to `mat_sd`
```

```
my_compadre <- cdb_build_cdb(mat_a = mat_list)
mat_sd(matA(my_compadre))
```

`string_representation` *Convert vectors or square numeric matrices to and from string representation*

Description

Functions to convert vectors or square numeric matrices to and from string representation, which is primarily useful for writing data frames with list-columns containing vectors or matrices to a flat file format such as csv.

String representations of vectors and matrices begin with an open bracket ("[") and end with a closed bracket ("]"). Matrix elements are separated with a space ("[0.2 0.3 0.1 0]") whereas vector elements are separate with two vertical bars ("[Seedling||Juvenile||Reproductive]").

Usage

```
mat_to_string(mat)

vec_to_string(vec)

string_to_mat(mat_str)

string_to_vec(vec_str, numeric = FALSE)
```

Arguments

`mat`	A square numeric matrix
`vec`	A vector
`mat_str`	A square numeric matrix in string representation
`vec_str`	A vector in string representation
`numeric`	Logical value indicating whether a string representation of a vector should be coerced to numeric (if FALSE remains character)

Value

A square numeric matrix (`string_to_mat`), vector (`string_to_vec`), or string (`mat_to_string` or `vec_to_string`).

Author(s)

Owen R. Jones <jones@biology.sdu.dk>

Patrick M. Barks <patrick.barks@gmail.com>

See Also

cdb_flatten cdb_unflatten

Other data management: `cdb_flatten()`, `cdb_id_stages()`, `cdb_id_studies()`, `cdb_id()`, `cdb_mean_matF()`, `cdb_rbind()`, `cdb_unflatten()`, `cdb_unnest()`, `mpm_elementwise_apply()`, `mpm_mean()`, `mpm_median()`, `mpm_sd()`

Examples

```
mat_str <- "[3.3 5.2 6.1 0.1 NA 0.3 0.2 0.4 0.1]"
mat <- string_to_mat(mat_str)

vec1_str <- "[0.30||0.42||0.19||0.09]"
vec1 <- string_to_vec(vec1_str, numeric = TRUE)

vec2_str <- "[Seedling 1||Seedling 2||Juvenile||Reproductive]"
vec2 <- string_to_vec(vec2_str)

# convert back to string format
mat_to_string(mat)
vec_to_string(vec1)
vec_to_string(vec2)

## Not run:
# non-square matrix
mat_str <- "[0.42 0.52 0.15 0.23 0.14]"
string_to_mat(mat_str)

## End(Not run)
```

Index