Package: mcmcdata (via r-universe)

September 12, 2024

Title Manipulate MCMC Samples and Data Frames

Version 0.0.1.9000

Description Manipulates Monte Carlo Markov Chain samples and associated data frames.

License MIT + file LICENSE

Depends R (>= 4.0)

Imports abind, chk, coda, dplyr, mcmcderive, mcmcr, plyr, stats, stringr, term, tidyr

Suggests covr, doParallel, testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Repository https://poissonconsulting.r-universe.dev

RemoteUrl https://github.com/poissonconsulting/mcmcdata

RemoteRef HEAD

RemoteSha 0861524f2cb60aa5904586c3daa9c4f65cb417cf

Contents

`anti_join.mcmc_data` *Anti join mcmc data*

Description

Anti join mcmc data

Usage

```
## S3 method for class 'mcmc_data'
anti_join(x, y, by = NULL, copy = FALSE, ...)
```

Arguments

x, y	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
by	A join specification created with `join_by()`, or a character vector of variables to join by. If `NULL`, the default, `*_join()` will perform a natural join, using all variables in common across `x` and `y`. A message lists the variables so that you can check they're correct; suppress the message by supplying `by` explicitly. To join on different variables between `x` and `y`, use a `join_by()` specification. For example, `join_by(a == b)` will match `x$a` to `y$b`. To join by multiple variables, use a `join_by()` specification with multiple expressions. For example, `join_by(a == b, c == d)` will match `x$a` to `y$b` and `x$c` to `y$d`. If the column names are the same between `x` and `y`, you can shorten this by listing only the variable names, like `join_by(a, c)`. `join_by()` can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins.

For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, `by = c("a", "b")` joins `x$a` to `y$a` and `x$b` to `y$b`. If variable names differ between `x` and `y`, use a named character vector like `by = c("x_a" = "y_a", "x_b" = "y_b")`.

To perform a cross-join, generating all combinations of `x` and `y`, see `cross_join()`.

`copy` If `x` and `y` are not from the same data source, and `copy` is `TRUE`, then `y` will be copied into the same src as `x`. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.

`...` Other parameters passed onto methods.

`arrange.mcmc_data` *Arrange mcmcr data*

Description

Arrange mcmcr data

Usage

```
## S3 method for class 'mcmc_data'
arrange(.data, ...)
```

Arguments

`.data` An mcmc_data object

`...` <data-masking> Variables, or functions of variables. Use `desc()` to sort a variable in descending order.

Examples

```
arrange(mcmc_data_example, species, homeworld)
```

`as_tibble.mcmc_data` *As Tibble*

Description

As Tibble

Usage

```
## S3 method for class 'mcmc_data'
as_tibble(x, samples = FALSE, ...)
```

Arguments

x	An mcmc_data object to coerce to a tibble.
samples	A flag specifying whether to include samples
...	Unused, for extensibility.

---

bind_rows2 *Bind Rows*

Description

Bind Rows

Usage

```
bind_rows2(x, y)
```

Arguments

x	A mcmc_data
y	A second mcmc_data

Examples

```
bind_rows2(mcmc_data_example, mcmc_data_example)
```

---

check_mcmc_data *Check mcmc data*

Description

Checks an object is a mcmc_data.

Usage

```
check_mcmc_data(x, x_name = NULL)
```

Arguments

x	The object to check.
x_name	A string of the name of object x or NULL.

Value

An invisible copy of x (if it doesn't throw an error).

Examples

```
check_mcmc_data(mcmcdata::mcmc_data_example)
```

coef.mcmc_data *Coefficients*

Description

Gets the coefficient table as a tibble.

Usage

```
## S3 method for class 'mcmc_data'
coef(object, conf_level = 0.95, estimate = stats::median, ...)
```

Arguments

object	The mcmc_data object to get the coefficent table for.
conf_level	A probability specifying the confidence level.
estimate	A function to calculate the point estimate.
...	Unused

Examples

```
coef(mcmc_data_example)
```

fill_na.mcmc_data *Fill Missing Values*

Description

Fills all of an object's missing values while preserving the object's dimensionality and class.

Usage

```
## S3 method for class 'mcmc_data'
fill_na(x, value = 0, ...)
```

Arguments

x	An object.
value	A scalar of the value to replace values with.
...	Other arguments passed to methods.

Details

It should only be defined for objects with values of consistent class ie not standard data.frames.

Value

The modified object.

Methods (by class)

- `fill_na(logical)`: Fill Missing Values for logical Objects
- `fill_na(integer)`: Fill Missing Values for integer Objects
- `fill_na(numeric)`: Fill Missing Values for numeric Objects
- `fill_na(character)`: Fill Missing Values for character Objects

See Also

Other fill: `fill_all()`

Examples

```
# logical
fill_na(c(TRUE, NA))

# integer
fill_na(c(1L, NA), 0)

# numeric
fill_na(c(1, NA), Inf)

# character
fill_na(c("text", NA))
fill_na(matrix(c("text", NA)), value = Inf)
```

`filter.mcmc_data` *Slice mcmcr data*

Description

Slice mcmcr data

Usage

```
## S3 method for class 'mcmc_data'
filter(.data, ...)
```

Arguments

`.data`	An mcmc_data object
`...`	For `slice()`: <`data-masking`> Integer row values. Provide either positive values to keep, or negative values to drop. The values provided must be either all positive or all negative. Indices beyond the number of rows in the input are silently ignored. For `slice_*()`, these arguments are passed on to methods.

Examples

```
filter(mcmc_data_example, species == "Droid")
```

---

group_by.mcmc_data *Group mcmc data*

Description

Group mcmc data

Usage

```
## S3 method for class 'mcmc_data'
group_by(.data, ..., .add = FALSE)
```

Arguments

.data	An mcmc_data object
...	In group_by(), variables or computations to group by. Computations are always done on the ungrouped data frame. To perform computations on the grouped data, you need to use a separate mutate() step before the group_by(). Computations are not allowed in nest_by(). In ungroup(), variables to remove from the grouping.
.add	When FALSE, the default, group_by() will override existing groups. To add to the existing groups, use .add = TRUE. This argument was previously called add, but that prevented creating a new grouping variable called add, and conflicts with our naming conventions.

Examples

```
group_by(mcmc_data_example, homeworld, species)
```

---

inner_join.mcmc_data *Inner join mcmc data*

Description

Inner join mcmc data

Usage

```
## S3 method for class 'mcmc_data'
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x, y	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
by	A join specification created with join_by(), or a character vector of variables to join by. If NULL, the default, *_join() will perform a natural join, using all variables in common across x and y. A message lists the variables so that you can check they're correct; suppress the message by supplying by explicitly. To join on different variables between x and y, use a join_by() specification. For example, join_by(a == b) will match x$a to y$b. To join by multiple variables, use a join_by() specification with multiple expressions. For example, join_by(a == b, c == d) will match x$a to y$b and x$c to y$d. If the column names are the same between x and y, you can shorten this by listing only the variable names, like join_by(a, c). join_by() can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins. For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, by = c("a", "b") joins x$a to y$a and x$b to y$b. If variable names differ between x and y, use a named character vector like by = c("x_a" = "y_a", "x_b" = "y_b"). To perform a cross-join, generating all combinations of x and y, see cross_join().
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.
...	Other parameters passed onto methods.

---

is.mcmc_data *Is an MCMC data object*

Description

Tests whether x is an object of class 'mcmc_data'

Usage

```
is.mcmc_data(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

`left_join.mcmc_data` *Left join mcmc data*

Description

Left join mcmc data

Usage

```
## S3 method for class 'mcmc_data'
left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x, y	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
by	A join specification created with `join_by()`, or a character vector of variables to join by. If `NULL`, the default, `*_join()` will perform a natural join, using all variables in common across `x` and `y`. A message lists the variables so that you can check they're correct; suppress the message by supplying `by` explicitly. To join on different variables between `x` and `y`, use a `join_by()` specification. For example, `join_by(a == b)` will match `x$a` to `y$b`. To join by multiple variables, use a `join_by()` specification with multiple expressions. For example, `join_by(a == b, c == d)` will match `x$a` to `y$b` and `x$c` to `y$d`. If the column names are the same between `x` and `y`, you can shorten this by listing only the variable names, like `join_by(a, c)`. `join_by()` can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins. For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, `by = c("a", "b")` joins `x$a` to `y$a` and `x$b` to `y$b`. If variable names differ between `x` and `y`, use a named character vector like `by = c("x_a" = "y_a", "x_b" = "y_b")`. To perform a cross-join, generating all combinations of `x` and `y`, see `cross_join()`.
copy	If `x` and `y` are not from the same data source, and `copy` is `TRUE`, then `y` will be copied into the same src as `x`. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.
suffix	If there are non-joined duplicate variables in `x` and `y`, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.
...	Other parameters passed onto methods.

mcmc_data *MCMC Data*

Description

Combines an mcmc object with a data frame. The mcmc object must be for a single parameter of the same length as the number of rows in the data frame. The resultant object can be filtered or summarized or the coef calculated.

Usage

```
mcmc_data(mcmc, data)
```

Arguments

mcmc	An mcmc object with one parameter
data	A data frame

Value

An object of class `mcmc_data`

mcmc_data_example *Example mcmc data Object*

Description

An example mcmc_data object.

Usage

```
mcmc_data_example
```

Format

An object of class `mcmc_data` of length 2.

`mcmc_derive.mcmc_data` *mcmc_derive mcmc_data*

Description

Gets mcmcr object of one or more derived parameters. To return an mcmc_data object use mcmc_derive_data(). By default the current parameter is call par.

Usage

```
## S3 method for class 'mcmc_data'
mcmc_derive(
  object,
  expr = "new_par <- par",
  values = list(),
  parameter = "par",
  monitor = ".*",
  parallel = FALSE,
  silent = getOption("mcmcderive.silent", FALSE),
  ...
)
```

Arguments

`object`	An mcmc_data object
`expr`	A string of the R expression.
`values`	A named list of values for objects not in object$data
`parameter`	A string of the name of the current MCMC samples in expr.
`monitor`	A regular expression specifying the derived parameter(s) in expr to monitor.
`parallel`	A flag specifying whether to generate the derived parameters for each chain in parallel.
`silent`	A flag specifying whether to suppress messages and warnings.
`...`	Unused.

Value

An mcmcr object of the derived parameters.

mcmc_derive_data *Derive MCMC Data*

Description

Get mcmc_data object.

Usage

```
mcmc_derive_data(object, ...)
```

Arguments

object	The object.
...	Not used.

mcmc_derive_data.mcmc_data *mcmc_derive mcmc_data*

Description

Gets mcmc_data object of a derived parameters. To return an mcmcr object of one or more derived parameters use mcmc_derive(). By default the current parameter is call par and the derived one is called new_par.

Usage

```
## S3 method for class 'mcmc_data'
mcmc_derive_data(
  object,
  expr = "new_par <- par",
  values = list(),
  parameter = "par",
  monitor = "new_par",
  parallel = FALSE,
  silent = getOption("mcmcderive.silent", FALSE),
  ...
)
```

Arguments

object	An mcmc_data object
expr	A string of the R expression.
values	A named list of values for objects not in object$data
parameter	A string of the name of the current MCMC samples in expr.
monitor	A string of the name of the derived parameter to return.
parallel	A flag specifying whether to generate the derived parameters for each chain in parallel.
silent	A flag specifying whether to suppress messages and warnings.
...	Unused.

Value

An mcmc_data object.

---

mutate.mcmc_data *Mutate mcmc_data*

Description

Mutate mcmc_data

Usage

```
## S3 method for class 'mcmc_data'
mutate(.data, ...)
```

Arguments

.data	An mcmc_data object
...	For `rename()`: <tidy-select> Use `new_name = old_name` to rename selected variables. For `rename_with()`: additional arguments passed onto `.fn`.

Examples

```
rename(mcmc_data_example, Spp = species)
```

rename.mcmc_data *Rename mcmcr data*

Description

Rename mcmcr data

Usage

```
## S3 method for class 'mcmc_data'
rename(.data, ...)
```

Arguments

.data	An mcmc_data object
...	For rename(): <tidy-select> Use new_name = old_name to rename selected variables. For rename_with(): additional arguments passed onto .fn.

Examples

```
rename(mcmc_data_example, Spp = species)
```

replace_na.mcmc_data *Replace Missing Values mcmc data*

Description

Replace Missing Values mcmc data

Usage

```
## S3 method for class 'mcmc_data'
replace_na(data, replace, ...)
```

Arguments

data	A data frame or vector.
replace	If data is a data frame, replace takes a named list of values, with one value for each column that has missing values to be replaced. Each value in replace will be cast to the type of the column in data that it being used as a replacement in. If data is a vector, replace takes a single value. This single value replaces all of the missing values in the vector. replace will be cast to the type of data.
...	Additional arguments for methods. Currently unused.

`right_join.mcmc_data` *Right join mcmc data*

Description

Right join mcmc data

Usage

```
## S3 method for class 'mcmc_data'
right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x, y	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
by	A join specification created with `join_by()`, or a character vector of variables to join by. If `NULL`, the default, `*_join()` will perform a natural join, using all variables in common across `x` and `y`. A message lists the variables so that you can check they're correct; suppress the message by supplying `by` explicitly. To join on different variables between `x` and `y`, use a `join_by()` specification. For example, `join_by(a == b)` will match `x$a` to `y$b`. To join by multiple variables, use a `join_by()` specification with multiple expressions. For example, `join_by(a == b, c == d)` will match `x$a` to `y$b` and `x$c` to `y$d`. If the column names are the same between `x` and `y`, you can shorten this by listing only the variable names, like `join_by(a, c)`. `join_by()` can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins. For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, `by = c("a", "b")` joins `x$a` to `y$a` and `x$b` to `y$b`. If variable names differ between `x` and `y`, use a named character vector like `by = c("x_a" = "y_a", "x_b" = "y_b")`. To perform a cross-join, generating all combinations of `x` and `y`, see `cross_join()`.
copy	If `x` and `y` are not from the same data source, and `copy` is `TRUE`, then `y` will be copied into the same src as `x`. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.
suffix	If there are non-joined duplicate variables in `x` and `y`, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2.
...	Other parameters passed onto methods.

select.mcmc_data *Select mcmcr data*

Description

Select mcmcr data

Usage

```
## S3 method for class 'mcmc_data'
select(.data, ...)
```

Arguments

.data	An mcmc_data object
...	<tidy-select> One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables.

Examples

```
select(mcmc_data_example, species, homeworld)
```

semi_join.mcmc_data *Semi join mcmc data*

Description

Semi join mcmc data

Usage

```
## S3 method for class 'mcmc_data'
semi_join(x, y, by = NULL, copy = FALSE, ...)
```

Arguments

x, y	A pair of data frames, data frame extensions (e.g. a tibble), or lazy data frames (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
by	A join specification created with join_by(), or a character vector of variables to join by. If NULL, the default, *_join() will perform a natural join, using all variables in common across x and y. A message lists the variables so that you can check they're correct; suppress the message by supplying by explicitly.

To join on different variables between x and y, use a join_by() specification. For example, join_by(a == b) will match x$a to y$b.

To join by multiple variables, use a join_by() specification with multiple expressions. For example, join_by(a == b, c == d) will match x$a to y$b and x$c to y$d. If the column names are the same between x and y, you can shorten this by listing only the variable names, like join_by(a, c).

join_by() can also be used to perform inequality, rolling, and overlap joins. See the documentation at ?join_by for details on these types of joins.

For simple equality joins, you can alternatively specify a character vector of variable names to join by. For example, by = c("a", "b") joins x$a to y$a and x$b to y$b. If variable names differ between x and y, use a named character vector like by = c("x_a" = "y_a", "x_b" = "y_b").

To perform a cross-join, generating all combinations of x and y, see cross_join().

copy — If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.

... — Other parameters passed onto methods.

`slice.mcmc_data` *Slice mcmcr data*

Description

Slice mcmcr data

Usage

```
## S3 method for class 'mcmc_data'
slice(.data, ...)
```

Arguments

.data — An mcmc_data object

... — For slice(): <data-masking> Integer row values.

Provide either positive values to keep, or negative values to drop. The values provided must be either all positive or all negative. Indices beyond the number of rows in the input are silently ignored.

For slice_*(), these arguments are passed on to methods.

Examples

```
slice(mcmc_data_example, 1L)
```

summarise.mcmc_data *Summarise mcmc data*

Description

Summarise mcmc data

Usage

```
## S3 method for class 'mcmc_data'
summarise(.data, ..., .fun = sum)
```

Arguments

.data	An mcmc_data object
...	<data-masking> Name-value pairs of summary functions. The name will be the name of the variable in the result. The value can be: • A vector of length 1, e.g. min(x), n(), or sum(is.na(y)). • A data frame, to add multiple columns from a single expression. **[Deprecated]** Returning values with size 0 or >1 was deprecated as of 1.1.0. Please use reframe() for this instead.
.fun	The function to use to summarise the MCMC samples.

ungroup.mcmc_data *Ungroup mcmc data*

Description

Ungroup mcmc data

Usage

```
## S3 method for class 'mcmc_data'
ungroup(x, ...)
```

Arguments

x	An mcmc_data object
...	In group_by(), variables or computations to group by. Computations are always done on the ungrouped data frame. To perform computations on the grouped data, you need to use a separate mutate() step before the group_by(). Computations are not allowed in nest_by(). In ungroup(), variables to remove from the grouping.

Examples

```
ungroup(mcmc_data_example)
```

Index