

Package ‘rblimp’

April 15, 2026

Type Package

Title Integration of 'Blimp' Software into R

Version 1.1.0

Description Provides an interface to 'Blimp' software for Bayesian latent variable modeling, missing data analysis, and multiple imputation. The package generates 'Blimp' syntax, executes 'Blimp' models, and imports results back into 'R' as structured objects with methods for visualization and analysis. Requires 'Blimp' software (freely available at <https://www.appliedmissingdata.com/blimp>) to be installed separately.

License GPL-3

Encoding UTF-8

Depends R (>= 4.4.0)

SystemRequirements Blimp software
(<https://www.appliedmissingdata.com/blimp>)

Imports cli, methods, ggplot2

Suggests mitml, rstudioapi, posterior

RoxygenNote 7.3.3

URL <https://github.com/blimp-stats/rblimp>

BugReports <https://github.com/blimp-stats/rblimp/issues>

NeedsCompilation no

Author Brian T. Keller [aut, cre, cph]

Maintainer Brian T. Keller <btkeller@missouri.edu>

Repository CRAN

Date/Publication 2026-04-14 22:20:03 UTC

Contents

algorithm	3
as.array,blimp_obj-method	3
as.character.blimp_syntax	4

as.data.frame,blimp_obj-method	5
as.matrix,blimp_obj-method	6
as.mitml	6
as_draws.blimp_obj	7
blimp_cp-class	8
blimp_obj	9
by_group	9
compare	10
compute_condeff	12
datainfo	14
describe	14
describe-class	16
detect_blimp	16
estimates	17
has_blimp	17
jn_plot	18
jn_plot_func	19
modelfit	21
modelinfo	22
model_table	22
names,blimp_obj-method	24
output	24
posterior_plot	25
predict,blimp_obj-method	26
print.blimp_syntax	27
psr	27
rblimp	28
rblimp_getting_started	32
rblimp_sim	35
rblimp_source	37
resid,blimp_obj-method	38
residuals,blimp_obj-method	38
residual_plot	39
set_blimp	40
show,blimp_cp-method	41
show,blimp_obj-method	41
simple_plot	42
SIMULATE	43
standardized	44
summary,blimp_obj-method	45
trace_plot	45
update_blimp	47
with,blimp_bygroup-method	47
with,blimp_obj-method	48
write.blimp	49

algorithm	<i>Extract algorithmic options from Blimp output</i>
-----------	--

Description

Extracts the algorithmic options section from Blimp output.

Usage

```
algorithm(object)
```

Arguments

object	A blimp_obj or blimp_out object
--------	---------------------------------

Value

A blimp_out object containing algorithmic options

as.array,blimp_obj-method	<i>Convert blimp_obj to a 3-D MCMC array</i>
---------------------------	--

Description

Reshapes the MCMC draws in a blimp_obj into a 3-D numeric array with dimensions [iteration, chain, parameter]. This is the format consumed directly by bayesplot::mcmc_* functions and by posterior::as_draws_array().

Usage

```
## S4 method for signature 'blimp_obj'
as.array(x, warmup = c("exclude", "include", "only"), ...)
```

Arguments

x	A blimp_obj object.
warmup	One of "exclude", "include", or "only". Default "exclude".
...	Additional arguments (unused).

Details

The warmup argument controls whether burn-in draws are included:

"exclude" (default) returns only post-burn sampling draws.

"include" returns warmup draws stacked on top of sampling draws within each chain. The returned array carries an "n_warmup" attribute giving the number of warmup iterations, so it can be piped directly into bayesplot::mcmc_trace(arr, n_warmup = attr(arr, "n_warmup")).

"only" returns only warmup draws.

Value

A numeric 3-D array with dimensions [iteration, chain, parameter], named dimnames, and (when warmup = "include") an "n_warmup" attribute.

See Also

[as_draws_array.blimp_obj\(\)](#) for the `posterior::draws_array` equivalent.

Examples

```
# Generate Data with `rblimp_sim`
mydata <- rblimp_sim(
  c(
    'f ~ normal(0, 1)',
    'x1:x5 ~ normal(f, 1)',
    'y ~ normal(10 + 0.3*f, 1 - .3^2)'
  ),
  n = 500,
  seed = 19723,
  variables = c('y', 'x1:x5')
)

# Fit model
model <- rblimp(
  list(structure = 'y ~ f', measurement = 'f -> x1:x5'),
  mydata,
  chains = 4,
  seed = 3927,
  latent = ~ f
)

# Post-burn only (default)
arr <- as.array(model)
dim(arr)
dimnames(arr)$parameters

# Include warmup, feed to bayesplot with its n_warmup marker
## Not run:
if (requireNamespace("bayesplot", quietly = TRUE)) {
  arr_full <- as.array(model, warmup = "include")
  bayesplot::mcmc_trace(arr_full, n_warmup = attr(arr_full, "n_warmup"))
}

## End(Not run)
```

as.character.blimp_syntax

Convert blimp_syntax to character vector

Description

Convert blimp_syntax to character vector

Usage

```
## S3 method for class 'blimp_syntax'  
as.character(x, ...)
```

Arguments

x	A blimp_syntax object
...	Additional arguments (unused)

Value

A character string containing the formatted 'Blimp' input syntax with a header comment and all commands.

as.data.frame,blimp_obj-method
Convert blimp_obj to data.frame

Description

Convert blimp_obj to data.frame

Usage

```
## S4 method for signature 'blimp_obj'  
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A blimp_obj object
row.names	NULL or a character vector giving row names
optional	Logical. If TRUE, setting row names is optional
...	Additional arguments passed to as.data.frame

Value

A data.frame containing the MCMC iterations from the model.

as.matrix,blimp_obj-method
Convert blimp_obj to matrix

Description

Convert blimp_obj to matrix

Usage

```
## S4 method for signature 'blimp_obj'  
as.matrix(x, ...)
```

Arguments

x	A blimp_obj object
...	Additional arguments (unused)

Value

A numeric matrix containing the MCMC iterations from the model.

as.mitml *Coerces a [blimp_obj](#) or blimp_bygroup to a mitml.list*

Description

Coerces a [blimp_obj](#) or blimp_bygroup to a mitml.list

Usage

```
as.mitml(object)
```

Arguments

object	blimp_obj or blimp_bygroup object
--------	---

Value

A list of class "mitml.list" containing the imputed data sets, suitable for use with functions from the 'mitml' package.

as_draws.blimp_obj *Convert blimp_obj to a posterior draws object*

Description

S3 methods for the `posterior::as_draws()` family of generics, allowing `blimp_obj` to be used anywhere a `posterior::draws` object is expected. This is the gateway to the **posterior** and **bayesplot** ecosystems: once converted, the usual `summarise_draws()`, `mcmc_trace()`, `mcmc_areas()`, etc., all work.

The **posterior** package must be installed to use these methods. It is listed in `Suggests`, so install it explicitly with `install.packages("posterior")` if needed.

Usage

```
## S3 method for class 'blimp_obj'
as_draws(x, warmup = c("exclude", "include", "only"), ...)

## S3 method for class 'blimp_obj'
as_draws_array(x, warmup = c("exclude", "include", "only"), ...)

## S3 method for class 'blimp_obj'
as_draws_df(x, warmup = c("exclude", "include", "only"), ...)
```

Arguments

<code>x</code>	A <code>blimp_obj</code> object.
<code>warmup</code>	One of "exclude", "include", or "only". Default "exclude".
<code>...</code>	Additional arguments (unused).

Details

All three methods accept a `warmup` argument with the same semantics as `as.array,blimp_obj-method`. Because `draws_array` objects do not carry a separate `warmup/sampling` distinction, the `"n_warmup"` attribute set by `as.array(x, warmup = "include")` is dropped during conversion; users who need the `warmup` marker should convert via `as.array()` and pass the result to `bayesplot::mcmc_trace()` directly.

Value

An object of class `posterior::draws_array` (for `as_draws` and `as_draws_array`) or `posterior::draws_df` (for `as_draws_df`).

Examples

```

mydata <- rblimp_sim(
  c(
    'f ~ normal(0, 1)',
    'x1:x5 ~ normal(f, 1)',
    'y ~ normal(10 + 0.3*f, 1 - .3^2)'
  ),
  n = 500,
  seed = 19723,
  variables = c('y', 'x1:x5')
)
model <- rblimp(
  list(structure = 'y ~ f', measurement = 'f -> x1:x5'),
  mydata,
  chains = 4,
  seed = 3927,
  latent = ~ f
)

draws <- posterior::as_draws_array(model)
posterior::summarise_draws(draws)

## Not run:
if (requireNamespace("bayesplot", quietly = TRUE)) {
  bayesplot::mcmc_trace(draws)
  bayesplot::mcmc_areas(draws)
}

## End(Not run)

```

blimp_cp-class

S4 class for Blimp model comparison results

Description

Result object from comparing two Blimp models.

Slots

table Matrix of comparison results
model0 The reference blimp_obj model
model The comparison blimp_obj model
greaterThan Logical indicating direction of comparison

blimp_obj	<i>S4 class for Blimp model results</i>
-----------	---

Description

The main result object containing Blimp model estimates, iterations, and output.

Slots

call The function call that created the object
 estimates Matrix of parameter estimates
 burn List of burn-in information for each chain
 iterations Data frame of MCMC iterations
 psr Data frame of Potential Scale Reduction values
 imputations List of imputed datasets
 average_imp Data frame of average imputations
 variance_imp Data frame of imputation variances
 waldtest Data frame of Wald test results
 simple Data frame for simple slopes analysis
 syntax The blimp_syntax object used to run the model
 output The blimp_out object containing raw output text

by_group	<i>Wrapper function to run multiple instances of <code>rblimp</code> based on grouping variable</i>
----------	---

Description

rblimp will generate the input, run the script, and load most the saved data into an R object. rblimp_fcs is used to specify the FCS command in Blimp. rblimp_syntax will generate the Blimp syntax file only.

Usage

```
by_group(expr, group)
```

Arguments

expr	a call to <code>rblimp</code> or <code>rblimp_fcs</code> .
group	a character vector or index pointing to grouping variable in data set

Details

Separates by grouping to run blimp command on each individual sub data set.

Value

a list of all blimp runs by group with class of blimp_bygroup

See Also

[as.mitml](#)

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'y ~ normal(0, 1)',
    'group = uniform(0, 1) < 0.5'
  ),
  n = 1000,
  seed = 10972
)

# Nonsensical model
mdl <- rblimp_fcs(
  c('x', 'y'),
  mydata,
  seed = 3927,
  nimps = 2,
) |> by_group('group')

# Analyze each imputation
results <- with(mdl, lm(y ~ x))
# use `mitml` package to pool results
```

compare

Compare two Blimp models

Description

Compares two Bayesian models by calculating the proportion of posterior samples where the comparison model's parameters exceed (or fall below) the reference model's summary statistic. This is useful for model comparison and assessing incremental variance explained (e.g., R-squared differences).

Usage

```
compare(
  model0,
  model,
  use = "mean",
  greaterThan = TRUE,
  suffixes = c("R2: Coefficients", "R2: Level-2 Random Intercepts",
    "R2: Level-2 Random Slopes", "R2: Level-3 Random Slopes",
    "R2: Level-3 Random Intercepts", "R2: Residual Variation",
    "R2: Level-1 Residual Variation")
)
```

Arguments

<code>model0</code>	A <code>blimp_obj</code> . The baseline or simpler model used as the reference point.
<code>model</code>	A <code>blimp_obj</code> . The comparison model (typically more complex) to evaluate.
<code>use</code>	Summary statistic to use as the cutpoint from <code>model0</code> . Options: <ul style="list-style-type: none"> • Character: "mean" or "median" • Numeric < 1: Quantile proportion (e.g., 0.5 for median) • Numeric >= 1: Percentile (e.g., 50 for median) • Function: Custom function applied to <code>model0</code> iterations • List: Multiple summary statistics
<code>greaterThan</code>	Logical. If TRUE (default), calculates the proportion of <code>model</code> iterations greater than the <code>model0</code> cutpoint. If FALSE, calculates proportion less than.
<code>suffixes</code>	Character vector of parameter name suffixes to compare. Defaults to all R-squared values (coefficients, random effects, residual variation).

Details

The comparison works by:

1. Computing a summary statistic (e.g., mean) from `model0`'s posterior samples
2. Calculating what proportion of `model`'s posterior samples exceed this value
3. Reporting this proportion for each parameter matching the specified suffixes

Value

A `blimp_cp` object containing a matrix of comparison proportions.

Note

Due to R restrictions, lists of functions will not give useful printed names.

Examples

```
# Generate data
mydata <- rblimp_sim(
  c(
    'x1 ~ normal(0, 1)',
    'x2 ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x1 + 0.3*x2, 1)'
  ),
  n = 200,
  seed = 123
)

# Fit baseline model (x1 only)
model0 <- rblimp(
  'y ~ x1',
  mydata,
  seed = 123,
  burn = 1000,
  iter = 1000
)

# Fit comparison model (x1 + x2)
model1 <- rblimp(
  'y ~ x1 x2',
  mydata,
  seed = 123,
  burn = 1000,
  iter = 1000
)

# Compare models - proportion of model1 R-squared > mean(model0 R-squared)
compare(model0, model1)
```

compute_condeff *Convenience Function for computing conditional effects for*
[jn_plot_func](#)

Description

Convenience Function for computing conditional effects for [jn_plot_func](#)

Usage

```
compute_condeff(value1, value2)
```

Arguments

value1 The base value
value2 The value to change as a function of moderator

Value

a function

See Also

[jn_plot_func\(\)](#)

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x1 ~ normal(0, 1)',
    'x2 ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x1 + 0.5*x2 + m + 0.2*x1*x2 + 0.3*x2*m + 0.1*x1*m + 0.7*x1*x2*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  'y ~ x1 x2 m x1*x2 x1*m x2*m x1*x2*m',
  mydata,
  center = ~ m,
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Get parameter values
params <- m1 |> as.matrix()

# Generate Plot
(
  jn_plot_func(
    compute_condeff(params[,6], params[,9]),
    xrange = c(-3, 3)
  )
  # Set custom colors
  + ggplot2::scale_fill_manual(
    values = c(`FALSE` = '#ca0020', `TRUE` = '#0571b0')
  )
  + ggplot2::labs(
    title = 'Johnson-Neyman Plot for `x1` * `x2` Moderated by `x2`',
    subtitle = 'Red area represents 0 within 95% interval',
    y = 'y ~ x1 * x2',
    x = 'm'
  )
  + ggplot2::theme_minimal()
)
```

datainfo	<i>Extract data information from Blimp output</i>
----------	---

Description

Extracts the data information section from Blimp output.

Usage

```
datainfo(object)
```

Arguments

object A blimp_obj or blimp_out object

Value

A blimp_out object containing data information

describe	<i>Describe an object</i>
----------	---------------------------

Description

describe provides a flexible summary of various R objects, such as data frames or Blimp model objects. It is an alternative to summary and is inspired by precis from the rethinking package.

Usage

```
describe(
  object,
  depth = 1,
  pars,
  prob = 0.95,
  digits = 2,
  sort = NULL,
  decreasing = FALSE,
  ...
)

## S4 method for signature 'data.frame'
describe(
  object,
  depth = 1,
```

```

    pars,
    prob = 0.5,
    digits = 2,
    sort = NULL,
    decreasing = FALSE,
    hist = TRUE,
    ...
)

## S4 method for signature 'blimp_obj'
describe(
  object,
  depth = 1,
  pars,
  prob = 0.95,
  digits = 2,
  sort = NULL,
  decreasing = FALSE,
  hist = TRUE,
  ...
)

```

Arguments

object	The object to describe.
depth	An integer (1, 2, or 3) that controls the display of vector and matrix parameters. depth=1 (default) shows only scalar parameters. depth=2 shows scalars and vectors. depth=3 shows all parameters.
pars	An optional character vector of parameter names to include in the summary.
prob	The probability mass for the credible interval (e.g., quantile interval).
digits	The number of decimal places to display in the output.
sort	An optional character string specifying a column name to sort the results by.
decreasing	Logical. If TRUE, sorting is in decreasing order.
...	Additional arguments passed to specific methods.
hist	Logical. If TRUE (and on a Unix-like OS), a unicode histogram (histospark) is included in the output. Defaults to TRUE.

Value

An object of class `describe`, which is a data frame containing summary statistics and attributes for printing.

Author(s)

The `data.frame` method is adapted from Richard McElreath's `precis` function in the `rethinking` package (<https://github.com/rmcelreath/rethinking>). The `histospark` function is by Hadley Wickham (GPL-3). The `blimp_obj` method was written for this package.

describe-class	<i>The 'describe' class for object summaries</i>
----------------	--

Description

An S4 class that holds a summary data frame and formatting information. It extends the `data.frame` class and is the return type for the `describe()` function.

Usage

```
## S4 method for signature 'describe'
show(object)
```

Arguments

`object` An object of class `describe`.

Slots

`digits` A numeric value indicating the number of digits to display.

detect_blimp	<i>Produce Blimp location Conditional on Operating System</i>
--------------	---

Description

This function is called when running Blimp and can be used to determine what location `rblimp` uses for the Blimp executable.

Usage

```
detect_blimp()
```

Details

`rblimp` will first use any location set with the `set_blimp` function. Next, it will check if `R_BLIMP` is set in the environment variables. Finally, it will fall back to the default Blimp install location based on the operating system.

Value

A character string of blimp's executable location.

See Also

`set_blimp` to set blimp location

Examples

```
# Obtain blimp location
detect_blimp()
```

estimates	<i>Extract model estimates from Blimp output</i>
-----------	--

Description

Extracts the model estimates section from Blimp output.

Usage

```
estimates(object)
```

Arguments

object A blimp_obj or blimp_out object

Value

A blimp_out object containing model estimates

has_blimp	<i>Does the system have blimp installed?</i>
-----------	--

Description

Returns a logical if blimp is detected by [detect_blimp](#)

Usage

```
has_blimp()
```

Details

This function uses [detect_blimp](#) to determine if blimp is installed.

Value

a logical

See Also

[detect_blimp](#)

Examples

```
# Detect if system has blimp
has_blimp()
```

jn_plot	<i>Function to generate a Johnson-Neyman Plot of Conditional Slope with rblimp</i>
---------	--

Description

Generates a Johnson-Neyman Plot based on the posterior summaries from the output of [rblimp](#).

Usage

```
jn_plot(formula, model, ci = 0.95, ...)
```

Arguments

formula	an object of class formula to specify simple effect to plot. The formula must have the following form: <code>outcome ~ focal moderator</code> . See Details below for nominal moderators.
model	an blimp_obj .
ci	a value between 0 and 1 specifying the credible interval size
...	passed bounds search algorithm. See jn_plot_func for details.

Details

To change colors use `ggplot2`'s scale system. Both fill and color are used. See [ggplot2::aes_colour_fill_alpha](#) for more information about setting a manual set of colors.

Value

a [ggplot2::ggplot](#) plot. The bounding values are saved in the attribute 'bounds'.

See Also

[jn_plot_func\(\)](#)

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x + m + 0.2*x*m, 1)'
  ),
```

```

      n = 100,
      seed = 981273
    )

# Run Rblimp
m1 <- rblimp(
  'y ~ x m x*m',
  mydata,
  center = ~ m,
  simple = 'x | m',
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Generate Plot
jn_plot(y ~ x | m, m1)

# Generate Plot with different colors
(
  jn_plot(y ~ x | m, m1)
  + ggplot2::scale_fill_manual(
    values = c(`FALSE` = '#ca0020', `TRUE` = '#0571b0')
  )
)
)

```

jn_plot_func

Function to generate a Johnson-Neyman Plot of Conditional Slope based on a function to produce the conditional effect.

Description

Generates a Johnson-Neyman Plot using a function to produce the conditional effect

Usage

```
jn_plot_func(func, xrange, ci = 0.95, ...)
```

Arguments

func	a function used to compute the conditional effect on moderator.
xrange	a numeric of length two with the min and max of the x-axis
ci	a value between 0 and 1 specifying the credible interval size
...	values passed to internal boundary search algorithm. See Details below.

Details

To change colors use ggplot2's scale system. Both fill and color are used. See [ggplot2::aes_colour_fill_alpha](#) for more information about setting a manual set of colors.

For . . . , the arguments are passed to the internal boundary search algorithm. This algorithm uses an initial grid search to locate boundaries based on the range and then a binary search to refine the estimates. The following arguments are available:

n_initial Number of points in the initial coarse grid search used to locate approximate boundary positions. Higher values improve detection of closely-spaced boundaries but increase computation time. Default is 1000.

refine_tol Tolerance for binary search refinement. The algorithm refines each boundary until the interval width is smaller than this value. Smaller values give higher precision but require more function evaluations. Default is 1e-12.

max_iter Maximum number of iterations for binary search refinement per boundary. Prevents infinite loops if tolerance cannot be achieved. Default is 100.

adaptive Logical indicating whether to perform additional refinement in regions where boundaries are detected to be closely spaced. When TRUE, uses a finer grid to resolve boundaries that may be missed by the initial coarse grid. Default is TRUE.

Value

a [ggplot2::ggplot](#) plot. The bounding values are saved in the attribute 'bounds'.

See Also

[compute_condeff\(\)](#)

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x1 ~ normal(0, 1)',
    'x2 ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x1 + 0.5*x2 + m + 0.2*x1*x2 + 0.3*x2*m + 0.1*x1*m + 0.7*x1*x2*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  'y ~ x1 x2 m x1*x2 x1*m x2*m x1*x2*m',
  mydata,
  center = ~ m,
  seed = 10972,
  burn = 1000,
  iter = 1000
)
```

```
)  
  
# Get parameter values  
params <- m1 |> as.matrix()  
  
# Generate Plot  
(  
  jn_plot_func(  
    compute_condeff(params[,6], params[,9]),  
    xrange = c(-3, 3)  
  )  
  + ggplot2::labs(  
    title = 'Johnson-Neyman Plot for `x1` * `x2` Moderated by `x2`',  
    subtitle = 'Red area represents 0 within 95% interval',  
    y = 'y ~ x1 * x2',  
    x = 'm'  
  )  
  + ggplot2::theme_minimal()  
)
```

modelfit

Extract model fit from Blimp output

Description

Extracts the model fit section from Blimp output.

Usage

```
modelfit(object)
```

Arguments

object A blimp_obj or blimp_out object

Value

A blimp_out object containing model fit information

modelinfo	<i>Extract model information from Blimp output</i>
-----------	--

Description

Extracts the model information section from Blimp output.

Usage

```
modelinfo(object)
```

Arguments

object A blimp_obj or blimp_out object

Value

A blimp_out object containing model information

model_table	<i>Create APA-formatted table of model results</i>
-------------	--

Description

Generates an APA-formatted HTML table of model estimates that opens in the RStudio Viewer. Provides publication-ready tables with proper formatting for Bayesian model results.

Usage

```
model_table(
  object,
  selector,
  digits = 3,
  caption = NULL,
  show_chains = TRUE,
  show_neff = FALSE
)
```

Arguments

object A [blimp_obj](#) containing model results.

selector Optional character string specifying variable name or block to display. If missing, creates separate tables for each variable.

digits Integer specifying number of decimal places. Default is 3.

caption Optional character string for table caption.

show_chains Logical indicating whether to show chain information. Default is TRUE.

show_neff Logical indicating whether to show effective sample size. Default is FALSE.

Details

The function creates APA-style tables with the following features:

- Proper coefficient formatting with confidence intervals
- Organized sections for different parameter types
- Publication-ready styling
- Responsive design for different screen sizes

Tables are displayed in the RStudio Viewer pane and can be exported or copied.

Value

Invisibly returns the HTML content. Primary purpose is displaying in Viewer.

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x, 1)'
  ),
  n = 100,
  seed = 10972
)

# Fit model
model <- rblimp(
  'y ~ x',
  mydata,
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Create APA table for all variables
model_table(model)

# Create table for specific variable
model_table(model, "y")

# Customize formatting
model_table(model, digits = 2, caption = "Bayesian Model Results")
```

names,blimp_obj-method

Return variable names from blimp_obj object

Description

Return variable names from blimp_obj object

Usage

```
## S4 method for signature 'blimp_obj'  
names(x)
```

Arguments

x A blimp_obj object

Value

A character vector of variable names from the average imputation data.

output

Extract output from blimp_obj

Description

Extracts the raw Blimp output text from a blimp_obj.

Usage

```
output(object)
```

Arguments

object A blimp_obj object

Value

A blimp_out object containing the raw Blimp output text

posterior_plot *Function to generate graph posterior density plots for parameters*

Description

Generates `ggplot2::ggplot` plots using `ggplot2::geom_density` based on the output from `rblimp`

Usage

```
posterior_plot(model, selector, ...)
```

Arguments

model	an <code>blimp_obj</code> .
selector	a name of a variable, a name of a parameter, a number of a parameter, or a combination of any of them. If left empty, a plot of all parameters will be returned. See Examples.
...	arguments passed to internally called <code>ggplot2::facet_wrap</code> when used.

Details

To change colors use `ggplot2`'s scale system. Both fill and color are used. See `ggplot2::aes_colour_fill_alpha` for more information about setting a manual set of colors.

Value

a `ggplot2::ggplot` plot

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x + m + 0.2*x*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  c(
    'y ~ x m',
    'x ~ m'
  ),
  mydata,
```

```

    seed = 10972,
    burn = 1000,
    iter = 1000
  )

  # Generate plot of all parameters with `y`
  posterior_plot(m1, 'y') + ggplot2::theme_minimal()

  # Generate plot of all parameters
  posterior_plot(m1) + ggplot2::theme_minimal()

  # Generate plot of all parameters for `y` and `x`
  posterior_plot(m1, c('x', 'y')) + ggplot2::theme_minimal()
  # Generate Plot of Parameter 5
  posterior_plot(m1, 5) + ggplot2::theme_minimal()

  # Generate plot of `x` residual variance`
  posterior_plot(m1, 'x residual variance') + ggplot2::theme_minimal()

  # Generate plot of Parameters 7 and 9
  posterior_plot(m1, c(7, 9)) + ggplot2::theme_minimal()

```

predict,blimp_obj-method

Predicted scores from blimp_obj

Description

Predicted scores from blimp_obj

Usage

```
## S4 method for signature 'blimp_obj'
predict(object, ...)
```

Arguments

object	A blimp_obj object
...	Additional arguments (unused)

Value

A list of data frames, one per imputation, each containing the predicted values and probability columns from the model.

print.blimp_syntax	<i>Print blimp_syntax</i>
--------------------	---------------------------

Description

Print blimp_syntax

Usage

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

Arguments

x	A blimp_syntax object
...	Additional arguments (unused)

Value

The blimp_syntax object (invisibly). Called for its side effect of printing the formatted syntax to the console.

psr	<i>Extract Potential Scale Reduction (PSR) output</i>
-----	---

Description

Extracts the PSR convergence diagnostic section from Blimp output.

Usage

```
psr(object)
```

Arguments

object	A blimp_obj or blimp_out object
--------	---------------------------------

Value

A blimp_out object containing PSR output

`rblimp`*Wrapper functions to provide Blimp functions in R*

Description

`rblimp` will generate the input, run the script, and load most the saved data into an R object. `rblimp_fcs` is used to specify the FCS command in Blimp. `rblimp_syntax` will generate the Blimp syntax file only.

Usage

```
rblimp(  
  model,  
  data,  
  burn = 10000,  
  iter = 10000,  
  seed,  
  thin,  
  nimps,  
  latent,  
  randomeffect,  
  parameters,  
  clusterid,  
  timeid,  
  weight,  
  ordinal,  
  nominal,  
  count,  
  center,  
  chains,  
  simple,  
  waldtest,  
  options,  
  transform,  
  dropout,  
  filter,  
  fixed,  
  output,  
  tmpfolder,  
  add_save = TRUE,  
  print_output = TRUE,  
  nopowershell = FALSE  
)  
  
rblimp_fcs(  
  variables,  
  data,
```

```
    burn = 10000,  
    iter = 10000,  
    seed,  
    thin,  
    nimps,  
    clusterid,  
    ordinal,  
    nominal,  
    chains,  
    options,  
    transform,  
    fixed,  
    output,  
    tmpfolder,  
    add_save = TRUE,  
    print_output = TRUE,  
    nopowershell = FALSE  
  )
```

```
rblimp_syntax(  
  model,  
  data,  
  burn,  
  iter,  
  seed,  
  thin,  
  nimps,  
  latent,  
  randomeffect,  
  parameters,  
  clusterid,  
  timeid,  
  weight,  
  ordinal,  
  nominal,  
  count,  
  transform,  
  dropout,  
  filter,  
  fixed,  
  center,  
  chains,  
  simple,  
  waldtest,  
  options,  
  save,  
  output  
)
```

Arguments

model	a character string or vector/list of character strings. Specifies Blimp's MODEL command. See details.
data	a <code>data.frame</code> or a <code>SIMULATE</code> object. If a <code>data.frame</code> , the data set used by Blimp. If a <code>SIMULATE</code> object (created with <code>SIMULATE()</code>), Blimp will generate simulated data instead of using existing data
burn	an integer. The number of burn-in iterations to be run
iter	an integer. The number of post burn-in iterations to be run
seed	a positive integer. The seeding value for Blimp's pseudo random number generator
thin	an integer. The thinning interval for imputations only.
nimps	an integer. The number of imputations to save.
latent	a character string, formula, or vector/list of character strings. Specifies Blimp's LATENT command. See details.
randomeffect	a character string or vector/list of character strings. Specifies Blimp's RAN- DOMEFFECTS command.
parameters	a character string or vector/list of character strings. Specifies Blimp's MODEL command. See details.
clusterid	a character string, formula, or vector/list of character strings. Specifies Blimp's CLUSTERID command. See details.
timeid	a character string, formula, or vector/list of character strings. Specifies Blimp's TIMEID command. See details.
weight	a character string, formula, or vector/list of character strings. Specifies Blimp's WEIGHT command. See details.
ordinal	a character string, formula, or vector/list of character strings. Specifies Blimp's ORDINAL command. See details.
nominal	a character string, formula, or vector/list of character strings. Specifies Blimp's NOMINAL command. See details.
count	a character string, formula, or vector/list of character strings. Specifies Blimp's COUNT command. See details.
center	a character string, formula, or vector/list of character strings. Specifies Blimp's CENTER command. See details.
chains	an integer, character string, or vector/list of character strings. Specifies Blimp's CHAINS command.
simple	a character string or vector/list of character strings. Specifies Blimp's SIMPLE command. See details.
waldtest	a character string or vector/list of character strings. Specifies Blimp's WALDTEST command. See details.
options	a character string or vector/list of character strings. Specifies Blimp's OPTIONS command.
transform	a character string or vector/list of character strings. Specifies Blimp's TRANS- FORM command.

dropout	a character string, formula, or vector/list of character strings. Specifies Blimp's DROPOUT command. See details.
filter	a character string. Specifies Blimp's FILTER command.
fixed	a character string, formula, or vector/list of character strings. Specifies Blimp's FIXED command. See details.
output	a character string or vector/list of character strings. Specifies Blimp's OUTPUT command
tmpfolder	a character string. If specified rblimp will use the given file path as a temporary directory instead of creating one with <code>tempdir</code>
add_save	a single logical value or a list of logical values. If TRUE then <code>saveLatent</code> , <code>saveResidual</code> , and <code>savePredicted</code> will be included in OPTIONS command. Otherwise, it will be coerced to a list. The elements of the list should be labeled <code>latent</code> , <code>residual</code> , and <code>predicted</code> each containing a single logical value that can be used to toggle on or off them individually. Missing elements will be defaulted to TRUE.
print_output	The type of output printed to the console. 'iteration' or logical TRUE is only iteration history, 'none' or logical FALSE suppresses all output to console, and 'all' prints all output to console.
nopowershell	Windows only. Uses <code>cmd.exe</code> with some limited functions (instead of powershell).
variables	list of variables to be included in FCS procedure
save	Specify SAVE command for blimp syntax

Details

The above functions require knowledge of specifying Blimp commands. Blimp's syntax commands are documented in the [Blimp User Manual](#)

By default, these commands can be inputted as character strings that will be used to generate the syntax. For multiple lined commands, you can wrap multiple strings into a character [vector](#) or a [list](#). The appropriate semicolons will be specified, so they are not required in any character strings. If specifying a character [vector](#) or a [list](#) for the `model`, providing names to each element will be used as blocks in Blimp's model syntax. Similarly, specifying named elements can be used for the `center` command to specify if you would like centering within a cluster or grand mean centering. This also works for the `latent` input when requesting latent variables at a specific cluster identifier. See the [Blimp User Manual](#) for more details about types of centering and specifying latent variables.

In addition, R's formula syntax can be used to specify lists of variables in place of character strings. For example, specifying the `CLUSTERID` command with the variable `id` can be specified as `~ id` as opposed to a character string. Similarly, `+` can be used to specify multiple variables. For example, to specify two variables as ordinal the formula would be: `~ x1 + x2`. Finally, this can also be used to specify centering and latent variables. For example, to center `x1` and `x2` within cluster we can specify: `cwc ~ x1 + x2`.

Running `rblimp` will also check if `blimp` is up to date. See details in [rblimp_source](#) for more information.

Value

blimp_obj

See Also

- [SIMULATE\(\)](#) for creating simulated data to fit within [rblimp\(\)](#)
- [rblimp_sim\(\)](#) for generating simulated datasets

Examples

```
# Generate Data with `rblimp_sim`
mydata <- rblimp_sim(
  c(
    'f ~ normal(0, 1)',
    'x1:x5 ~ normal(f, 1)',
    'y ~ normal(10 + 0.3*f, 1 - .3^2)'
  ),
  n = 500,
  seed = 19723,
  variables = c('y', 'x1:x5')
)

# Fit SEM Model
model <- rblimp(
  list(
    structure = 'y ~ f',
    measurement = 'f -> x1:x5'
  ),
  mydata,
  seed = 3927,
  latent = ~ f
)

# View results
summary(model)
```

rblimp_getting_started

Getting Started with rblimp

Description

A guide to installing and configuring Blimp software for use with rblimp.

Installing Blimp Software

Before using rblimp, you must install the Blimp software (freely available):

1. Visit <https://www.appliedmissingdata.com/blimp>
2. Download the version for your operating system
3. Follow the installation instructions

After installation, configure Blimp. The package will look for Blimp in the following order:

1. The R_BLIMP environment variable
2. Common installation locations (automatic detection)

To configure Blimp:

```
# Automatic detection
detect_blimp()

# Or set path manually (for current session)
set_blimp("/path/to/blimp")

# Or set R_BLIMP environment variable (permanent)
# Add to .Renviron file:
# R_BLIMP="/path/to/blimp"

# Verify installation
has_blimp()
```

Automatic Update Checks

By default, rblimp automatically checks for Blimp updates when you run models. This ensures you're using the latest version with bug fixes and improvements.

To disable automatic update checks:

```
# Disable for current session
options(check_blimp_update = FALSE)

# Or add to .Rprofile for permanent setting:
# options(check_blimp_update = FALSE)
```

You can manually check for updates at any time:

```
update_blimp()
```

Function Reference

Model Fitting:

- [rblimp](#) - Fit Bayesian models
- [rblimp_fcs](#) - Fully Conditional Specification imputation

- `rblimp_syntax` - Generate Blimp syntax
- `rblimp_source` - Run existing Blimp syntax files

Simulation:

- `rblimp_sim` - Generate simulated datasets
- `SIMULATE` - Create simulation specifications

Output & Analysis:

- `summary` - Model summary
- `estimates` - Extract parameter estimates
- `describe` - Descriptive statistics
- `psr` - Potential Scale Reduction values
- `output` - Raw Blimp output
- `standardized` - Extract standardized parameters only
- `compare` - Compare models

Visualization:

- `trace_plot` - MCMC trace plots
- `posterior_plot` - Posterior density plots
- `residual_plot` - Residual diagnostics
- `jn_plot` - Johnson-Neyman plots
- `simple_plot` - Simple slopes plots
- `model_table` - Publication tables

Utilities:

- `by_group` - Grouped analysis
- `as.mitml` - Convert to mitml format
- `names` - Obtain imputation variable names
- `with` - Evaluate across imputations
- `write.blimp` - Write results to files

Setup & Configuration:

- `detect_blimp` - Detect Blimp installation
- `set_blimp` - Set path to Blimp executable
- `has_blimp` - Check if Blimp is available
- `update_blimp` - Update Blimp installation

Additional Resources

- GitHub repository: <https://github.com/blimp-stats/rblimp>
- Example files: <https://github.com/blimp-stats/rblimp-examples>
- Blimp User Guide: <https://docs.google.com/document/d/1D3MS79CakuX9mVVvGH13B5nRd9XLttp69oGsvrIRK64>
- Report issues: <https://github.com/blimp-stats/rblimp/issues>

rblimp_sim	<i>Simulate data using Blimp</i>
------------	----------------------------------

Description

Generate simulated datasets using Blimp's SIMULATE command. Supports both single-level and multilevel data generation.

Usage

```
rblimp_sim(
  model,
  n,
  seed,
  define = NULL,
  variables = NULL,
  tmpfolder,
  nopowershell = FALSE
)
```

Arguments

model	Character string or vector specifying data generation equations. Can be specified as a vector: <code>c("x = normal(0, 1)", "y = normal(10, 1)")</code> OR as a single string with newlines/semicolons: <code>"x = normal(0, 1); y = normal(10, 1);"</code> For multilevel: use named list with arbitrary level names (e.g., <code>list(schools = ..., students = ...)</code>). The sample size determines the level hierarchy (largest = level-1, smallest = highest level).
n	Sample size. For single-level: integer (e.g., 1000). For multilevel: named list with sample sizes (e.g., <code>list(students = 1000, schools = 100)</code>). Names must match the names used in model list. Sample sizes must be unique.
seed	Random seed for reproducibility.
define	Character vector or string of parameter definitions (optional). Can be <code>c("b0 = 10", "b1 = 0.5")</code> OR <code>"b0 = 10; b1 = 0.5;"</code> .
variables	Formula or character string specifying which variables to save (optional). If not specified, all variables from the model are saved.
tmpfolder	Character string. Temporary folder path. If not specified, creates one with tempdir .
nopowershell	Windows only. Uses cmd.exe instead of powershell.

Value

A data.frame with simulated data. The returned object has two attributes:

syntax The Blimp syntax used for simulation (blimp_syntax object)

output The raw Blimp output (blimp_out object)

Access these with `attr(result, "syntax")` and `attr(result, "output")`.

See Also

[SIMULATE\(\)](#) for creating a simulation specification to use with `rblimp()`

Examples

```
# Simple regression:  $y = 10 + 0.5x + \text{error}$ 
dat <- rblimp_sim(
  model = c(
    "x = normal(0, 1)",
    "y = normal(10 + x*0.5, 1)"
  ),
  n = 1000,
  seed = 10972
)

# Same thing with full quoted syntax
dat <- rblimp_sim(
  model = "x = normal(0, 1);
          y = normal(10 + x*0.5, 1);",
  n = 1000,
  seed = 10972
)

# With parameter definitions
dat <- rblimp_sim(
  model = c(
    "x = normal(0, 1)",
    "y = normal(b0 + b1*x, s2e)"
  ),
  n = 1000,
  define = c("b0 = 10", "b1 = 0.5", "s2e = 1"),
  seed = 10972
)

# Multilevel model
dat <- rblimp_sim(
  model = list(
    schools = c(
      "z = normal(0, 1)",
      "u0 = normal(10 + z*-0.5, 1.0)"
    ),
    students = c(
      "x = normal(0, 1)",
      "y = normal(u0 + x*0.5, 1)"
    )
  ),
  n = list(students = 1000, schools = 100),
  seed = 198723
)

# Access syntax and output
syntax <- attr(dat, "syntax")
```

```
print(syntax)

output <- attr(dat, "output")
print(output)
```

rblimp_source	<i>Runs an 'imp' file with Blimp</i>
---------------	--------------------------------------

Description

This function runs 'imp' file with Blimp and captures the output only.

Usage

```
rblimp_source(file, plots = FALSE, output = TRUE, nopowershell = FALSE)
```

Arguments

file	a character string to the 'imp' file's location
plots	a logical. Setting to TRUE will generate the data required for plotting by Blimp
output	The type of output printed to the console. 'iteration' or logical TRUE is only iteration history, 'none' or logical FALSE suppresses all output to console, and 'all' prints all output to console.
nopowershell	Windows only. Uses cmd.exe with some limited functions (instead of powershell).

Details

Running rblimp_source will also run a check to see if Blimp is up to date. If Blimp is not up to date, it will prompt the user if it would like to update or not. This check will only be performed on the first run in a session and then every ten hours. This behavior can be disabled by setting the check_blimp_update option to FALSE using [options](#). This check is not performed if R is not being run with an interactive session. See [interactive](#) for more information.

Value

a blimp_out object

Examples

```
# Run blimp script
## Not run: output <- rblimp_source("filepath/to/syntax")
```

```
resid,blimp_obj-method
```

Residuals scores from blimp_obj

Description

Residuals scores from blimp_obj

Usage

```
## S4 method for signature 'blimp_obj'  
resid(object, ...)
```

Arguments

object	A blimp_obj object
...	Additional arguments passed to residuals

Value

A list of data frames, one per imputation, each containing the residual columns from the model.

```
residuals,blimp_obj-method
```

Residuals scores from blimp_obj

Description

Residuals scores from blimp_obj

Usage

```
## S4 method for signature 'blimp_obj'  
residuals(object, ...)
```

Arguments

object	A blimp_obj object
...	Additional arguments (unused)

Value

A list of data frames, one per imputation, each containing the residual columns from the model.

residual_plot	<i>Function to generate conditional regression equation plots (i.e., simple effects) with rblimp and SIMPLE command</i>
---------------	---

Description

Generates a conditional effect plots based on the posterior summaries from the output of [rblimp](#).

Usage

```
residual_plot(  
  model,  
  variable,  
  nsigma = 1,  
  point_col = "black",  
  horz_line = "black",  
  col1 = "#0571b0",  
  col2 = "#ca0020",  
  linewidth = 1.1,  
  ...  
)
```

Arguments

model	an blimp_obj . The object must have a SIMPLE command output saved.
variable	the name of the outcome for which to create a plot
nsigma	the number of standard deviations to produce credible bounds
point_col	the color of the points in the plot
horz_line	the color of the horizontal zero line
col1	the color of the loess mean line
col2	the color of the loess credible bound lines
linewidth	the linewidth value for the loess lines and its bounds.
...	arguments passed to loess call used to loess lines.

Details

All colors are passed into `ggplot2`. See [ggplot2::aes_colour_fill_alpha](#) for details on changing colors.

Value

a `ggplot2::ggplot` plot

Examples

```
# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x + m + 0.2*x*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  'y ~ x m',
  mydata,
  nimps = 10,
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Generate Plot
residual_plot(m1, 'y') + ggplot2::theme_minimal()
```

set_blimp

Set Blimp Executable Location

Description

This function can set the Blimp executable location if it cannot be autodetected.

Usage

```
set_blimp(exec, beta = FALSE)
```

Arguments

exec a character string for the Blimp executable's location

beta a logical value. If true `rblimp` will use the beta version which must be installed

Details

By default `rblimp` tries to determine the location of Blimp's computational engine based on standard operating system installation locations. This function is useful for non standard installations or when wanting to use a beta version of blimp's computational engine (which must be installed via the updater).

Value

TRUE if the executable is successfully set; otherwise, it produces an error.

Examples

```
# example code
# Use blimp executable at `filepath` location
## Not run: set_blimp('filepath')

# Use default blimp location but beta build
set_blimp(beta = TRUE)
```

show,blimp_cp-method *Show method for blimp_cp*

Description

Show method for blimp_cp

Usage

```
## S4 method for signature 'blimp_cp'
show(object)
```

Arguments

object A blimp_cp object

Value

No return value, called for the side effect of printing the comparison table to the console.

show,blimp_obj-method *Show method for blimp_obj*

Description

Show method for blimp_obj

Usage

```
## S4 method for signature 'blimp_obj'
show(object)
```

Arguments

object A blimp_obj object

Value

No return value, called for the side effect of printing the model estimates to the console.

simple_plot	<i>Function to generate conditional regression equation plots (i.e., simple effects) with <code>rblimp</code> and <code>SIMPLE</code> command</i>
-------------	---

Description

Generates a conditional effect plots based on the posterior summaries from the output of `rblimp`.

Usage

```
simple_plot(formula, model, ci = 0.95, xvals, ...)
```

Arguments

formula an object of class `formula` to specify simple effect to plot. The formula must have the following form: `outcome ~ focal | moderator`. See Details below for nominal moderators.

model an `blimp_obj`. The model must have a `SIMPLE` command output saved.

ci a value between 0 and 1 specifying the credible interval size

xvals a list of values to evaluate for the focal variable. If empty, they will automatically be determined

... arguments passed to the internal `ggplot2::geom_line` call used to generate the median lines.

Details

To change colors use `ggplot2`'s scale system. Both fill and color are used. See `ggplot2::aes_colour_fill_alpha` for more information about setting a manual set of colors.

For nominal moderators, the variable must include the nominal code used in the dummy codes (e.g., `moderator.1`, `moderator.2`, etc). When there are multiple dummy codes, then all codes must be listed using a `+`. For example, after the `~` the following statement can be included:

```
focal | moderator.1 + moderator.2
```

Value

a `ggplot2::ggplot` plot

Examples

```

# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x + m + 0.2*x*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  'y ~ x m x*m',
  mydata,
  center = ~ m,
  simple = 'x | m',
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Generate Plot
simple_plot(y ~ x | m, m1)

```

SIMULATE

*Create a SIMULATE specification for use in rblimp()***Description**

Creates a simulation specification that can be passed to `rblimp()` as the data argument. Instead of reading existing data, `rblimp()` will use Blimp's SIMULATE command to generate data and then fit the specified model to it.

Usage

```
SIMULATE(model, n, define = NULL, variables = NULL)
```

Arguments

<code>model</code>	Character string or vector specifying data generation equations. Same format as <code>rblimp_sim()</code> .
<code>n</code>	Sample size. For single-level: integer. For multilevel: named list.
<code>define</code>	Character vector or string of parameter definitions (optional).
<code>variables</code>	Formula or character string specifying which variables to save (optional).

Value

A `blimp_simulate` object (subclass of `blimp_syntax`) that can be passed to `rblimp()` as the data argument.

See Also

[rblimp_sim\(\)](#) for directly generating simulated data without fitting a model

Examples

```
# Create simulation specification
sim_spec <- SIMULATE(
  model = c(
    "x = normal(0, 1)",
    "y = normal(10 + x*0.5, 1)"
  ),
  n = 1000
)

# View the specification
print(sim_spec)

# Use in rblimp to fit a model to simulated data
mdl <- rblimp(
  model = "y ~ x",
  data = sim_spec,
  seed = 123,
  burn = 5000,
  iter = 5000
)

summary(mdl)
```

standardized

Extract standardized solutions from Blimp

Description

Extracts the data information section from Blimp output.

Usage

```
standardized(object)
```

Arguments

`object` A `blimp_obj` or `blimp_out` object

Value

A `base::matrix` with standardized solutions

summary,blimp_obj-method

Summary method for blimp_obj

Description

Provides formatted summary output for `blimp_obj`, optionally allowing selection by variable name or block. Offers cleaner presentation than the raw estimates matrix.

Usage

```
## S4 method for signature 'blimp_obj'
summary(object, selector, digits = 3, ...)
```

Arguments

<code>object</code>	A <code>blimp_obj</code> object containing model results.
<code>selector</code>	Optional character string specifying the variable name or block to extract. If missing, shows all estimates. Can be a variable name (e.g., "y1") or a block name (e.g., "between").
<code>digits</code>	Integer specifying the number of decimal places for rounding. Default is 3.
<code>...</code>	Additional arguments (for S4 method compatibility).

Value

If `selector` is provided: invisibly returns a matrix or list of estimates for the selected variable/block.
If no `selector`: returns the full estimates matrix with improved formatting.

trace_plot

Function to generate trace plots of the burn-in iterations

Description

Generates `ggplot2::ggplot` plots using `ggplot2::geom_line` based on the output from `rblimp`

Usage

```
trace_plot(model, selector, ...)
```

Arguments

model an `blimp_obj`.
selector a name of a variable, a name of a parameter, a number of a parameter, or a combination of any of them. If left empty, a plot of all parameters will be returned. See Examples.
... arguments passed to internally called `ggplot2::facet_wrap` when used.

Details

To change colors use `ggplot2`'s scale system. Both fill and color are used. See `ggplot2::aes_colour_fill_alpha` for more information about setting a manual set of colors.

Value

a `ggplot2::ggplot` plot

Examples

```

# Generate Data
mydata <- rblimp_sim(
  c(
    'x ~ normal(0, 1)',
    'm ~ normal(0, 1)',
    'y ~ normal(10 + 0.5*x + m + 0.2*x*m, 1)'
  ),
  n = 100,
  seed = 981273
)

# Run Rblimp
m1 <- rblimp(
  c(
    'y ~ x m',
    'x ~ m'
  ),
  mydata,
  seed = 10972,
  burn = 1000,
  iter = 1000
)

# Generate plot of all parameters with `y`
trace_plot(m1, 'y') + ggplot2::theme_minimal()

# Generate plot of all parameters with `y`
# Add limits to only graph first 250 iterations
trace_plot(m1, 'y') + ggplot2::xlim(0, 250) + ggplot2::theme_minimal()

# Generate plot of all parameters
trace_plot(m1) + ggplot2::theme_minimal()

```

```

# Generate plot of all parameters for `y` and `x`
trace_plot(m1, c('x', 'y')) + ggplot2::theme_minimal()
# Generate Plot of Parameter 5
trace_plot(m1, 5) + ggplot2::theme_minimal()

# Generate plot of `x` residual variance`
trace_plot(m1, 'x residual variance') + ggplot2::theme_minimal()

# Generate plot of Parameters 7 and 9
trace_plot(m1, c(7, 9)) + ggplot2::theme_minimal()

```

update_blimp

Runs the Blimp Updater in order to Update Blimp

Description

Runs the Blimp Updater in order to Update Blimp

Usage

```
update_blimp()
```

Details

If the session is not an interactive session then it will not open the updater.

Value

Logical if the updater was opened or not

Examples

```

# Open Blimp Updater
update_blimp()

```

with,blimp_bygroup-method

Fit Model across imputations with mi tml package using [by_group](#)

Description

Fit Model across imputations with mi tml package using [by_group](#)

Usage

```
## S4 method for signature 'blimp_bygroup'
with(data, expr, ...)
```

Arguments

data	A <code>blimp_bygroup</code> object
expr	An expression to evaluate on each imputation
...	Additional arguments (unused)

Value

A list of class `"mitml.result"` containing the results of evaluating `expr` on each imputed data set.

`with,blimp_obj-method` *Fit Model across imputations with mitml package*

Description

Fit Model across imputations with `mitml` package

Usage

```
## S4 method for signature 'blimp_obj'
with(data, expr, ...)
```

Arguments

data	A <code>blimp_obj</code> object
expr	An expression to evaluate on each imputation
...	Additional arguments (unused)

Value

A list of class `"mitml.result"` containing the results of evaluating `expr` on each imputed data set.

`write.blimp`*A function to write out blimp input and output from a model*

Description

A function to write out blimp input and output from a model

Usage

```
write.blimp(object, folder = "")

## S4 method for signature 'blimp_syntax'
write.blimp(object, folder = "")

## S4 method for signature 'blimp_out'
write.blimp(object, folder = "")

## S4 method for signature 'blimp_obj'
write.blimp(object, folder = "")
```

Arguments

<code>object</code>	A blimp_obj .
<code>folder</code>	a location to a folder to write input and output

Value

No return value, called for its side effect of writing 'Blimp' input and output files to disk.

Methods (by class)

- `write.blimp(blimp_syntax)`: Write `blimp_syntax` to file
- `write.blimp(blimp_out)`: Write `blimp_out` to file
- `write.blimp(blimp_obj)`: Write `blimp_obj` files to folder

Examples

```
# Generate Data with `rblimp_sim`
mydata <- rblimp_sim(
  c(
    'f ~ normal(0, 1)',
    'x1:x5 ~ normal(f, 1)',
    'y ~ normal(10 + 0.3*f, 1 - .3^2)'
  ),
  n = 500,
  seed = 19723,
  variables = c('y', 'x1:x5')
```

```
)  
  
# Fit SEM Model  
model <- rblimp(  
  list(  
    structure = 'y ~ f',  
    measurement = 'f -> x1:x5'  
  ),  
  mydata,  
  seed = 3927,  
  latent = ~ f  
)  
  
# Write out input and output  
## Not run:  
write.blimp(model, "folder_location")  
  
## End(Not run)
```

Index

* documentation

- rblimp_getting_started, 32
- algorithm, 3
- as.array,blimp_obj-method, 3, 7
- as.character.blimp_syntax, 4
- as.data.frame,blimp_obj-method, 5
- as.matrix,blimp_obj-method, 6
- as.mitml, 6, 10, 34
- as_draws.blimp_obj, 7
- as_draws_array.blimp_obj
 - (as_draws.blimp_obj), 7
- as_draws_array.blimp_obj(), 4
- as_draws_df.blimp_obj
 - (as_draws.blimp_obj), 7
- base::matrix, 45
- blimp_cp-class, 8
- blimp_obj, 6, 9, 18, 22, 25, 32, 39, 42, 45, 46, 49
- blimp_obj-class (blimp_obj), 9
- by_group, 9, 34, 47
- compare, 10, 34
- compute_condeff, 12
- compute_condeff(), 20
- data.frame, 30
- datainfo, 14
- describe, 14, 15, 34
- describe(), 16
- describe,blimp_obj-method (describe), 14
- describe,data.frame-method (describe), 14
- describe-class, 16
- describe-methods (describe), 14
- detect_blimp, 16, 17, 34
- estimates, 17, 34
- formula, 18, 42
- function, 13, 19
- ggplot2::aes_colour_fill_alpha, 18, 20, 25, 39, 42, 46
- ggplot2::facet_wrap, 25, 46
- ggplot2::geom_density, 25
- ggplot2::geom_line, 42, 45
- ggplot2::ggplot, 18, 20, 25, 39, 42, 45, 46
- has_blimp, 17, 34
- interactive, 37
- jn_plot, 18, 34
- jn_plot_func, 12, 18, 19
- jn_plot_func(), 13, 18
- list, 31
- loess, 39
- model_table, 22, 34
- modelfit, 21
- modelinfo, 22
- names, 34
- names,blimp_obj-method, 24
- numeric, 19
- options, 37
- output, 24, 34
- posterior::as_draws(), 7
- posterior::as_draws_array(), 3
- posterior::draws, 7
- posterior::draws_array, 4, 7
- posterior::draws_df, 7
- posterior_plot, 25, 34
- predict,blimp_obj-method, 26
- print.blimp_syntax, 27
- psr, 27, 34
- rblimp, 9, 16, 18, 25, 28, 33, 39, 40, 42, 45

rblimp(), [32](#), [36](#)
rblimp_fcs, [9](#), [33](#)
rblimp_fcs (rblimp), [28](#)
rblimp_getting_started, [32](#)
rblimp_sim, [34](#), [35](#)
rblimp_sim(), [32](#), [43](#), [44](#)
rblimp_source, [31](#), [34](#), [37](#)
rblimp_syntax, [34](#)
rblimp_syntax (rblimp), [28](#)
resid, blimp_obj-method, [38](#)
residual_plot, [34](#), [39](#)
residuals, blimp_obj-method, [38](#)

set_blimp, [16](#), [34](#), [40](#)
show, blimp_cp-method, [41](#)
show, blimp_obj-method, [41](#)
show, describe-method (describe-class),
[16](#)
simple_plot, [34](#), [42](#)
SIMULATE, [30](#), [34](#), [43](#)
SIMULATE(), [30](#), [32](#), [36](#)
standardized, [34](#), [44](#)
summary, [34](#)
summary, blimp_obj-method, [45](#)

tempdir, [31](#), [35](#)
trace_plot, [34](#), [45](#)

update_blimp, [34](#), [47](#)

vector, [31](#)

with, [34](#)
with, blimp_bygroup-method, [47](#)
with, blimp_obj-method, [48](#)
write.blimp, [34](#), [49](#)
write.blimp, blimp_obj-method
(write.blimp), [49](#)
write.blimp, blimp_out-method
(write.blimp), [49](#)
write.blimp, blimp_syntax-method
(write.blimp), [49](#)