Package: rblimp (via r-universe)

June 2, 2026

Type Package

Title Integration of 'Blimp' Software into R

Version 1.2.0

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

Roxygen list(markdown = TRUE)

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

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

Config/roxygen2/version 8.0.0

Repository https://blimp-stats.r-universe.dev

Date/Publication 2026-06-02 16:39:44 UTC

RemoteUrl https://github.com/blimp-stats/rblimp

RemoteRef HEAD

RemoteSha fbc03bbf269510b4ad7079f0b1c4c61513bfc452

Contents

`algorithm` *Extract algorithmic options from Blimp output*

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`
Convert blimp_obj to a 3-D MCMC array

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

`x`	A `blimp_obj` object.
`warmup`	One of `"exclude"`, `"include"`, or `"only"`. Default `"exclude"`.
`...`	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)
```

---

at *Marker for fixing a moderator at specific SIMPLE evaluated values*

Description

Used inside `simple_plot` and `jn_plot` formulas to restrict the SIMPLE rows to those whose moderator was evaluated at the given label(s).

Usage

```
at(...)
```

Arguments

`...`	named arguments where each name is a moderator in the SIMPLE command and each value is a single label or a vector of labels (e.g., `"Q25"`, `"+1 SD"`, `"0"`, or a parameter name) to keep.

Value

Calling `at()` outside a formula raises an error; the function only has meaning when it appears in a `simple_plot()` or `jn_plot()` formula.

See Also

simple_plot, jn_plot, jn_map, join

Examples

```
## Not run:
# Pin `m2` at the SIMPLE label "0", leaving `m1` to color the lines
simple_plot(y ~ x | m1 + at(m2 = "0"), model)

# Pin to a subset (vector of labels) -- `m2` still faces over those values
simple_plot(y ~ x | m1 + at(m2 = c("Q25", "Q75")), model)

# Use inside `jn_plot()` the same way
jn_plot(y ~ x | m1 + at(m2 = "+1 SD"), model)

# In a 4-way fit, pin one moderator and map the slope surface over the other two
jn_map(y ~ x | m1 + m2 + at(m3 = "0"), four_way_model)

## 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 *S4 class for Blimp model results*

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 *Wrapper function to run multiple instances of* `rblimp` *based on grouping variable*

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 `rblimp` or `rblimp_fcs`.
`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

`model0`	A `blimp_obj`. The baseline or simpler model used as the reference point.
`model`	A `blimp_obj`. The comparison model (typically more complex) to evaluate.
`use`	Summary statistic to use as the cutpoint from `model0`. Options: • 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 `model0` iterations • List: Multiple summary statistics
`greaterThan`	Logical. If `TRUE` (default), calculates the proportion of `model` iterations greater than the `model0` cutpoint. If `FALSE`, calculates proportion less than.
`suffixes`	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 *Extract data information from Blimp output*

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 *Describe an object*

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 *The 'describe' class for object summaries*

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 *Produce Blimp location Conditional on Operating System*

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(prompt = TRUE)
```

Arguments

`prompt`	Logical; when `TRUE` (default) and no Blimp is found in an interactive session, ask whether to install Blimp via `install_blimp`. Set to `FALSE` to suppress the prompt and fail with an error instead (this is what `has_blimp` does internally).

Details

rblimp resolves Blimp's executable location in this order:

1. Any location previously set via set_blimp.
2. The R_BLIMP environment variable.
3. A managed install created by install_blimp.
4. The default operating-system install location of the Blimp system installer.

If none are found and the R session is interactive, detect_blimp offers to download and install Blimp via install_blimp (unless prompt = FALSE).

Value

A character string of blimp's executable location.

See Also

set_blimp to set blimp location, install_blimp to install Blimp into a managed directory.

Examples

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

---

estimates *Extract model estimates from Blimp output*

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 *Does the system have blimp installed?*

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()
```

install_blimp *Install the Blimp computational engine*

Description

Downloads and installs Blimp into a user-writable directory managed by rblimp. This is an alternative to installing Blimp via the system installer from https://www.appliedmissingdata.com/blimp.

Usage

```
install_blimp(dir = NULL, force = FALSE, quiet = FALSE, linux_variant = NULL)
```

Arguments

dir	Install directory. Defaults to a hidden, user-writable per-OS location (~/.blimp/ on macOS and Linux, %LOCALAPPDATA%/Blimp/ on Windows), overridable via the R_BLIMP_HOME environment variable.
force	Logical; overwrite an existing managed install, or proceed even when a system installation of Blimp is detected. Default FALSE. When both managed and system installations exist, the managed install takes precedence in detect_blimp.
quiet	Logical; suppress progress messages. Default FALSE.
linux_variant	Optional character; "ubuntu" or "rhel8". When NULL (default), the variant is auto-detected from /etc/os-release. Ignored on macOS and Windows.

Details

The install directory location can be overridden globally via the R_BLIMP_HOME environment variable.

Set R_BLIMP_PREVENT_INSTALL=true to disable install_blimp (intended for locked-down environments).

Non-interactive R sessions require force = TRUE.

Value

Path to the installed blimp executable, invisibly.

Privacy

Calling install_blimp() contacts updates.blimpstats.com over HTTPS to fetch the Blimp engine. Information may be collected as described in the privacy policy.

See privacy policy: https://www.blimpstats.com/privacy.

See Also

uninstall_blimp, update_blimp

Examples

```
## Not run:
# Install Blimp to the default managed location
install_blimp()

# Install to a custom directory
install_blimp(dir = "~/blimp-managed")

# Force install on a Linux RHEL 8 system without auto-detection
install_blimp(linux_variant = "rhel8")

## End(Not run)
```

jn_map *Two-dimensional Johnson-Neyman map of a conditional slope*

Description

A 2-D analogue of jn_plot(). For a model with an outcome ~ focal x m1 x m2 interaction, the conditional slope of outcome on focal is a linear surface over (m1, m2):

$$\frac{\partial y}{\partial x} = a + b\,m_1 + c\,m_2 + d\,m_1 m_2.$$

jn_map() recovers those four coefficients per MCMC draw from the SIMPLE-evaluated points, evaluates them on a dense grid, and renders the posterior median as a heatmap. A contour line marks the boundary of the region where the ci posterior credible interval excludes zero – the 2-D "region of significance".

Usage

```
jn_map(formula, model, ci = 0.95, n_grid = 100, ...)
```

Arguments

formula	an object of class formula of the form outcome ~ focal | m1 + m2. Exactly two bare moderators are required; additional moderators present in the SIMPLE statement must be pinned via at().
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 for the significance contour. Default 0.95.
n_grid	integer length-1 or length-2. Number of grid points along m1 and m2. Default 100 (i.e. 100 x 100 cells).
...	currently unused.

Value

a ggplot2::ggplot plot. Attribute "grid" carries the per-cell data frame (with lower, median, upper, sig columns).

See Also

jn_plot, simple_plot, at, join

Examples

```
## Not run:
# Generate data with a three-way interaction
mydata <- rblimp_sim(
    c(
        'x  ~ normal(0, 1)',
```

```
        'm1 ~ normal(0, 1)',
        'm2 ~ normal(0, 1)',
        'y  ~ normal(10 + 0.4*x + 0.3*m1 - 0.2*m2 +
                       0.5*x*m1 + 0.3*x*m2 + 0.2*m1*m2 +
                       0.6*x*m1*m2, 1)'
    ),
    n = 500,
    seed = 981273
)

# Fit the model -- SIMPLE must evaluate both moderators
fit <- rblimp(
    'y ~ x m1 m2 x*m1 x*m2 m1*m2 x*m1*m2',
    mydata,
    center = ~ x + m1 + m2,
    simple = 'x | m1 @ quantile and m2 @ sd',
    seed   = 1071,
    burn   = 1000,
    iter   = 1000
)

# Default 2-D heatmap of the conditional slope of y on x
jn_map(y ~ x | m1 + m2, fit)

# Swap axes
jn_map(y ~ x | m2 + m1, fit)

# Inspect the underlying per-cell data
head(attr(jn_map(y ~ x | m1 + m2, fit), 'grid'))

## End(Not run)
```

`jn_plot` *Function to generate a Johnson-Neyman Plot of Conditional Slope with* `rblimp`

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: `outcome ~ focal	moderator`. 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.

When the model contains a SIMPLE command that evaluates the focal predictor at multiple values of the formula moderator (e.g., 'x | m @ quantile'), the conditional slope is reconstructed from the SIMPLE slope draws via a per-iteration linear fit across the evaluated points. If the same SIMPLE command also holds one or two additional moderators at multiple values (e.g., 'x | m @ quantile and z @ sd'), the plot is faceted by those additional moderators. When SIMPLE has no relevant rows, jn_plot falls back to building the conditional slope directly from the model's fixed-effect interaction parameter (the original behavior).

Value

a ggplot2::ggplot plot. The bounding values are saved in the attribute 'bounds'.

See Also

jn_plot_func, jn_map, simple_plot, at, join

Examples

```
## Not run:
# ---- Basic single-moderator example ----
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
)
m1 <- rblimp(
    'y ~ x m x*m',
    mydata, center = ~ m, simple = 'x | m',
    seed = 10972, burn = 1000, iter = 1000
)
jn_plot(y ~ x | m, m1)

# Custom significance-region fill
(
    jn_plot(y ~ x | m, m1)
    + ggplot2::scale_fill_manual(
        values = c(`FALSE` = '#ca0020', `TRUE` = '#0571b0')
    )
```

```
)

# ---- Two-moderator fit: first mod = x-axis, second auto-faceted ----
three_way <- rblimp_sim(
    c(
        'x  ~ normal(0, 1)',
        'm1 ~ normal(0, 1)',
        'm2 ~ normal(0, 1)',
        'y  ~ normal(10 + 0.5*x*m1 + 0.3*x*m2 + 0.6*x*m1*m2, 1)'
    ),
    n = 500, seed = 2024
)
fit <- rblimp(
    'y ~ x m1 m2 x*m1 x*m2 m1*m2 x*m1*m2',
    three_way, center = ~ x + m1 + m2,
    simple = 'x | m1 @ quantile and m2 @ sd',
    seed = 1071, burn = 1000, iter = 1000
)
jn_plot(y ~ x | m1 + m2, fit)

# Pin one moderator via `at()`
jn_plot(y ~ x | m1 + at(m2 = "0"), fit)

# Restrict to a subset of SIMPLE values
jn_plot(y ~ x | m1 + at(m2 = c("-1 SD", "+1 SD")), fit)

# Per-facet boundary x-values are also exposed as an attribute
attr(jn_plot(y ~ x | m1 + m2, fit), "bounds")

## End(Not run)
```

`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()
)
```

`join` *Marker for bundling multiple SIMPLE moderators into one compound moderator*

Description

Used inside `simple_plot` and `jn_plot` formulas to treat a set of SIMPLE moderators as a single conceptual moderator. Each row's color/legend label combines every component's `"name @ value"` so that, e.g., the dummy codes of a nominal predictor render as one set of colored lines instead of a sparse facet grid.

Usage

```
join(...)
```

Arguments

`...`	bare moderator names (at least two) to bundle together.

Details

Only the first bare term on the right-hand side of `|` may use `join(...)` (that is, the color / x-axis position). Facet positions must remain plain moderator names or `at(...)` calls.

Value

Calling `join()` outside a formula raises an error; it only has meaning inside `simple_plot()` / `jn_plot()` formulas.

See Also

simple_plot, jn_plot, jn_map, at

Examples

```
## Not run:
# Nominal predictor with dummy codes -- bundle them as one moderator
simple_plot(y ~ x | join(group.1, group.2), model)

# Combine with `at()` to pin an unrelated moderator
simple_plot(y ~ x | join(group.1, group.2) + at(z = "0"), model)

# In a 4-way fit, collapse the two facet moderators so the bound
#   annotation goes back into each strip of the JN plot
jn_plot(y ~ x | m1 + join(m2, m3), four_way_model)

## End(Not run)
```

---

model_table *Create APA-formatted table of model results*

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")
```

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 *Extract model information from Blimp output*

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

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 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`
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 *Print* blimp_syntax

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 *Extract Potential Scale Reduction (PSR) output*

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 data.frame or a SIMULATE object. If a data.frame, the data set used by Blimp. If a SIMULATE object (created with SIMULATE()), 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 RANDOMEFFECTS 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 TRANSFORM 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 tempdir
add_save	a single logical value or a list of logical values. If TRUE then saveLatent, saveResidual, and savePredicted will be included in OPTIONS command. Otherwise, it will be coerced to a list. The elements of the list should labeled latent, residual, and predicted 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 cmd.exe 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

rblimp requires the Blimp engine. The simplest way to install it is from R:

```
install_blimp()
```

This downloads the latest Blimp engine into a user-writable directory:

- macOS: `~/.blimp/`
- Windows: `%LOCALAPPDATA%/Blimp/`
- Linux: `~/.blimp/`

Override the location with the `R_BLIMP_HOME` environment variable. Use `uninstall_blimp` to remove it.

If you'd rather use Blimp's standalone system installer (downloaded from https://www.appliedmissingdata.com/blimp), rblimp will auto-detect it and use it instead.

If autodetection fails, set the path manually with `set_blimp` or the `R_BLIMP` environment variable.

Privacy: Downloads are recorded for usage statistics.

See privacy policy: https://www.blimpstats.com/privacy.

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)

# Or set environment variable (useful for CI / sysadmin contexts):
# R_BLIMP_NO_UPDATE_CHECK=true
```

You can manually update Blimp at any time:

```
update_blimp()
```

For managed installs, this re-downloads the latest Blimp engine. For system installs, this launches the Blimp Updater application.

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:

- install_blimp - Install Blimp into a managed directory
- uninstall_blimp - Remove the managed install
- 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 *Simulate data using Blimp*

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: `c("x = normal(0, 1)", "y = normal(10, 1)")` OR as a single string with newlines/semicolons: `"x = normal(0, 1); y = normal(10, 1);"` For multilevel: use named list with arbitrary level names (e.g., `list(schools = ..., students = ...)`). 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., `list(students = 1000, schools = 100)`). 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 `c("b0 = 10", "b1 = 0.5")` OR `"b0 = 10; b1 = 0.5;"`.
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 + 0.5*x + 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` *Runs an 'imp' file with Blimp*

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`, or by setting the `R_BLIMP_NO_UPDATE_CHECK` environment variable to `"true"`. 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.

`residual_plot` *Function to generate conditional regression equation plots (i.e., simple effects) with* `rblimp` *and SIMPLE command*

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 horiztonal 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()
```

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.

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 *Function to generate conditional regression equation plots (i.e., simple effects) with rblimp and SIMPLE command*

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 and for SIMPLE statements that include more than one moderator.
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, wrap the dummy codes in join(...) so they are treated as a single compound moderator (one set of colored lines instead of separate facets):

```
focal | join(moderator.1, moderator.2)
```

join() works for any user-controlled bundling, not just nominal dummies; however it is only valid in the first (color/legend) position of the formula.

When the SIMPLE command contains more than one moderator (e.g., mod1 @ values and mod2 @ value), the right-hand side of the formula may list the moderators that vary using + (e.g., focal | mod1 + mod2). Moderators that are omitted from the formula are treated as held-constant context: their value must be the same across the matched simple effects and is reported in the plot subtitle.

Value

a ggplot2::ggplot plot

See Also

jn_plot, jn_map, at, join

Examples

```
## Not run:
# ---- Basic single-moderator example ----
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
)
m1 <- rblimp(
    'y ~ x m x*m',
    mydata,
    center = ~ m,
    simple = 'x | m',
    seed = 10972,
    burn = 1000,
    iter = 1000
)
simple_plot(y ~ x | m, m1)

# ---- Two moderators: first colored, second auto-faceted ----
three_way <- rblimp_sim(
    c(
        'x  ~ normal(0, 1)',
        'm1 ~ normal(0, 1)',
        'm2 ~ normal(0, 1)',
        'y  ~ normal(10 + 0.5*x*m1 + 0.3*x*m2 + 0.6*x*m1*m2, 1)'
    ),
    n = 500, seed = 2024
)
fit <- rblimp(
    'y ~ x m1 m2 x*m1 x*m2 m1*m2 x*m1*m2',
    three_way, center = ~ x + m1 + m2,
    simple = 'x | m1 @ quantile and m2 @ sd',
    seed = 1071, burn = 1000, iter = 1000
)
simple_plot(y ~ x | m1 + m2, fit)

# Pin a moderator to one SIMPLE value via `at()`
simple_plot(y ~ x | m1 + at(m2 = "0"), fit)
```

```
# Restrict to a subset of values
simple_plot(y ~ x | m1 + at(m2 = c("-1 SD", "+1 SD")), fit)

# Bundle moderators (e.g. nominal dummy codes) via `join()`
simple_plot(y ~ x | join(m1, m2), fit)

## End(Not run)
```

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

`model`	Character string or vector specifying data generation equations. Same format as `rblimp_sim()`.
`n`	Sample size. For single-level: integer. For multilevel: named list.
`define`	Character vector or string of parameter definitions (optional).
`variables`	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

object	A blimp_obj object containing model results.
selector	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").
digits	Integer specifying the number of decimal places for rounding. Default is 3.
...	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()
```

`uninstall_blimp` *Remove the managed Blimp installation*

Description

Deletes the managed install directory created by `install_blimp`. Does not affect system installations of Blimp.

Usage

```
uninstall_blimp(quiet = FALSE)
```

Arguments

`quiet`	Logical; suppress messages and skip the interactive confirmation prompt. Default `FALSE`.

Value

`TRUE` if anything was removed, `FALSE` otherwise (invisibly).

See Also

`install_blimp`

Examples

```
## Not run: uninstall_blimp()
```

---

update_blimp *Update the installed Blimp engine*

Description

If Blimp was installed via `install_blimp`, the latest version is re-downloaded into the managed directory. If Blimp was installed via the system installer, the Blimp Updater application is launched (back-compat).

Usage

```
update_blimp()
```

Details

If the session is not interactive, the updater is not opened.

For managed installs, the manifest is checked first and the engine is re-downloaded only if a newer version exists.

Value

Logical indicating whether an update was started.

Privacy

Calling `update_blimp()` on a managed install contacts `updates.blimpstats.com` over HTTPS to check for a newer version of Blimp, and downloads from it if one is available. Information may be collected as described in the privacy policy.

See privacy policy: https://www.blimpstats.com/privacy.

See Also

`install_blimp`

Examples

```
# Update Blimp
## Not run: update_blimp()
```

`with,blimp_bygroup-method`
Fit Model across imputations with `mitml` *package using* `by_group`

Description

Fit Model across imputations with `mitml` package using `by_group`

Usage

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

Arguments

`data`	A `blimp_bygroup` 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 blimp_obj 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

object	A blimp_obj.
folder	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