# Package 'mlmpower'

May 8, 2026

**Type** Package

**Title** Power Analysis and Data Simulation for Multilevel Models

**Version** 1.0.11

**Description** A declarative language for specifying multilevel models, solving for population parameters based on specified variance-explained effect size measures, generating data, and conducting power analyses to determine sample size recommendations. The specification allows for any number of within-cluster effects, between-cluster effects, covariate effects at either level, and random coefficients. Moreover, the models do not assume orthogonal effects, and predictors can correlate at either level and accommodate models with multiple interaction effects.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**URL** https://github.com/bkeller2/mlmpower

**BugReports** https://github.com/bkeller2/mlmpower/issues

**Depends** R (>= 4.2.0)

**Imports** cli, lme4, lmerTest, varTestnlme

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2026-02-16 17:10:02 UTC

# Contents



---

`analyze` *Analyzes a single* `mp_data` *using* `lme4::lmer`

---

## Description

Analyzes a single `mp_data` based on the data generating model.

## Usage

```
analyze(data, alpha = 0.05, no_lrt = FALSE, ...)
```

### Arguments

| | |
|---|---|
| data | a mp_data. |
| alpha | the significance level to determine if an effect is statistically significant. If NULL then no nested model testing is conducted. |
| no_lrt | do not perform additional likelihood ratio tests. Setting to TRUE will speed up the analysis because the model is only fit once. |
| ... | other arguments passed to lme4::lmer(). |

### Value

A list that with the following named elements:

- estimates: The estimates from fitting the model.
- sig_test: The logical if the estimates were statistically siginificant based on alpha.
- parameters: The mp_parameters extracted from data.

### Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(198723)
# Create data set and analyze
model |> generate(5, 50) |> analyze() -> results
```

---

## as.data.frame.mp_power

*Coerce a mp_power to a Data Frame*

---

### Description

Outputs mp_power as a data frame.

### Usage

```
## S3 method for class 'mp_power'
as.data.frame(x, row.names = NULL, optional = FALSE, power = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | a mp_power. |
| row.names | passed to base::as.data.frame |
| optional | passed to base::as.data.frame |
| power | logical: do you want the power or the estimates |
| ... | other arguments not used by this method. |

### Value

returns a data frame

### Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(19723)
# Create data set and analyze
# Note: Generally Use more than 50 replications
model |> power_analysis(50, 5, 50) -> powersim
# Obtain Results as a data frame
as.data.frame(powersim)
```

---

## as.list.mp_parameters *Convert* mp_parameters *to a* list

---

### Description

A wrapper to coerce mp_parameters to a list.

### Usage

```
## S3 method for class 'mp_parameters'
as.list(x, ...)
```

### Arguments

| | |
|---|---|
| x | the mp_parameters to be coered. |
| ... | additional arguments passed to as.list |

### Value

a list

## Examples

```
# Specify model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = c(0.1, 0.2),
        within = 0.3
    )
)
# Obtain parameters and convert to a list
model |> summary() |> as.list() -> param_list
```

---

`center` *Center a data set based on a* `mp_data`

---

## Description

Provides multilevel centering of a `mp_data` data set.

## Usage

```
center(data, all = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `data` | a `mp_data` or a `list` of `mp_data`. |
| `all` | a logical value to center all variables based on model defaults |
| `...` | see details below |

## Details

The `...` needs to be the variable's name followed by equals and the centering strategy requested. There are three different strategies available:

- `cwc` = centering within cluster
- `cgm` = centering with group mean
- `none` = no centering

If `all` is set to `TRUE` then the default centering will be used unless overwritten by specifying a specific centering strategy.

## Value

For `ndata = 1` a single `data.frame` is returned. If a list of data sets are included then they will be contained in a `list`. Each `data.frame` has an additional `center` attribute which denotes the centering strategy used.

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(198723)

# Create data set with default centering
model |> generate(5, 50) |> center(all = TRUE) -> mydata

# Create data centering X with cwc
model |> generate(5, 50) |> center(X = cwc) -> mydata

# See centering strategy
attr(mydata, 'center')
```

---

`correlations` *Specify the Correlation Structure for the Model*

---

## Description

Creates a list of correlations to be added to a `mp_model`.

## Usage

```
correlations(within, between, randeff)
```

## Arguments

| | |
|---|---|
| `within` | a single numeric value or `mp_corr_func` that specifies random correlations. Corresponds to the level-1 correlation among predictors. |
| `between` | a single numeric value or `mp_corr_func` that specifies random correlations. Corresponds to the level-2 correlation among predictors. |
| `randeff` | a single numeric value or `mp_corr_func` that specifies random correlations. Corresponds to the random effects correlation among predictors. |

## Details

The default values are `random(0.1, 0.3)`. Currently `randeff` are required to be zero if more than one random slope is in the model.

## Value

A list that corresponds to each correlation value.

## See Also

random() fixed()

## Examples

```
(
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = c(0.1, 0.2),
        within = 0.3
    )
    # Defaults
    + correlations(
        within  = random(0.1, 0.3),
        between = random(0.1, 0.3),
        randeff = random(0.1, 0.3)
    )
)
```

---

# effect_size *Specify the Effect Size for the Model*

---

## Description

Creates a list of effect sizes to be added to a mp_model.

Returns suggested ICC's for cross-sectional studies (0.05, 0.15, and 0.25).

Returns suggested ICC ranges for longitudinal studies (0.40, 0.50, and 0.60).

## Usage

```
effect_size(icc, within, between, random_slope, product)

cross_sectional()

longitudinal()
```

## Arguments

| | |
|---|---|
| icc | a numeric vector of global ICC values for mp_variable who are left unspecified. Values must be between 0 and 1. |
| within | a single numeric value that corresponds to the proportion of variance explained by the within variables. |
| between | a single numeric value that corresponds to the incremental proportion of variance explained by the between variables. |
| random_slope | a single numeric value that corresponds to the proportion of variance explained by the random slopes. |

product — a single numeric value that corresponds to the proportion of variance explained by the product terms.

## Value

A list that corresponds to each R2 value.

## Examples

```
# Set ICCs
(
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = c(0.1, 0.2),
        within = 0.3
    )
)

# With cross-sectional ICC
(
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = cross_sectional,
        within = 0.3
    )
)

# With longitudinal ICC
(
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = longitudinal,
        within = 0.3
    )
)
```

---

# generate *Generates Data Sets Based on a* mp_model

---

## Description

Generates data sets based on a mp_model. These data sets will be returned as a data.frame and include the solved parameters as an attribute to the data.frame.

## Usage

```
generate(model, n_within, n_between, ndata = 1, mechanism = NULL)
```

## Arguments

| | |
|---|---|
| model | a mp_model. |
| n_within | a single positive integer of the desired within cluster observations. |
| n_between | a single positive integer of the desired between cluster observations. |
| ndata | a single positive integer of the number of desired data sets. |
| mechanism | a function for inducing missing data to the data set. If NULL it is ignored. See details below. |

## Details

Note that there must only be one global ICC in mp_model.

Use the mechanism argument to specify missing data mechanisms. See mechanisms for predefined missing data mechanisms for the outcome and examples using them. When creating custom mechanisms care needs to be taken because it is considered for advanced usage. This a argument expects a function with the mp_data as the input, and the function should return the modified mp_data. Be careful when using this because it allows you to modify the population parameters, which will be incorrect. You should only induce missing data values on variables. Missing data on the predictors will cause listwise deletion to be used, but missing data on the outcome will be appropriate for MAR-based mechanisms. See examples below for an example that generates MCAR data on the outcome. See parameters to obtain the population parameters from each data set.

## Value

For ndata = 1 a single data.frame is returned. The first variable is the cluster identifier labeled _id. This object is also of class mp_data which means that it was generated based on a specified model. If multiple data sets are requested then they will be contained in a list.

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(198723)

# Create data set
model |> generate(5, 50) -> mydata

# Induce missing data with built-in function
model |> generate(50, 5, mechanism = MCAR(0.25)) -> mydata_mcar

# Induce missing data with custom function
model |> generate(50, 5, mechanism = \(data) {
    # `data` will be the `mp_data` used
    within(data, {
        # MCAR Process
```

```
        Y <- ifelse(runif(NROW(data)) < 0.5, Y, NA)
    })
}) -> mydata_mcar_custom
```

---

`is_valid` *Check if a Model is Properly Specified*

---

## Description

This function is used to validate if a `mp_model` is correct. If the model is incorrect an appopriate error message describing while will be supplied

## Usage

```
is_valid(x)
```

## Arguments

| | |
|---|---|
| x | a `mp_model` |

## Value

Invisibly returns the original model.

## Examples

```
# Create Model
model <- outcome('Y') + within_predictor('X')
# Throws error
tryCatch(
    is_valid(model),
    error = print
)
# Succeeds
is_valid(model + effect_size(icc = 0.1))
```

---

`levels.mp_variable` *Obtain Level of Observation for a Variable*

---

## Description

Returns which level a variable is observed at in the multilevel model.

## Usage

```
## S3 method for class 'mp_variable'
levels(x)
```

### Arguments

| | |
|---|---|
| x | a mp_variable. |

### Value

Returns a single integer of the level of observation

### Examples

```
# Returns 1
levels(
    within_predictor(
        'X',
        weight = 1,
        mean = 5,
        sd = 10,
        icc = 0.1
    )
)
```

---

## mechanisms *Helper functions for producing Missing Data Mechanisms*

---

### Description

Functions to generate data that always follows a specific mechanism in accordance to a single-level model.

### Usage

```
# Generate MCAR data on outcome
MCAR(mis.rate)

# Generate MAR data on outcome due to `cause`
MAR(mis.rate, cause, r2, lower = TRUE)
```

### Arguments

| | |
|---|---|
| mis.rate | A proportion for the missing data rate at population level |
| cause | A character for a variable name that is the cause of missingness |
| r2 | A proportion of variance explained by the cause in the missing data indicator's latent propensity |
| lower | A logical for the lower or upper tail being more likely to be missing |

### See Also

power_analysis

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)

# Induce MCAR data on outcome
set.seed(19723)
model |> power_analysis(50, 5, 50, mechanism = MCAR(0.25)) -> powersim_mcar

# Induce MAR data on outcome
set.seed(19723)
model |> power_analysis(
   50, 5, 50,
   mechanism = MAR(0.25, 'X', 0.6)
) -> powersim_mar
```

---

`mlmpower` `mlmpower` *Modeling Framework*

---

## Description

`mlmpower` constructs models by adding different features of the model using the plus sign (+).

Every model requires an `outcome` and an ICC specified in `effect_size` to be valid.

```
model <- outcome('y') + effect_size(icc = 0.1)
```

Once a model is constructed, we can add additional features to build the model out more. For example, we may want to include a level-1 predictor that is centered within cluster.

```
model <- model + within_predictor('x', icc = 0.0)
```

The additions can be chained together to produce the entire model object. For example, the previous two code blocks can be combined into one.

```
model <- (
    outcome('y')
    + effect_size(icc = 0.1)
    + within_predictor('x', icc = 0.0)
)
```

Finally, we can also wrap multiple variables into a list and add that. This feature can be useful when programmatically generating a model.

```
model <- (
    outcome('y')
    + effect_size(icc = 0.1)
    + lapply(1:10, \(i) within_predictor(paste0('x', i), icc = 0.0))
)
```

For more detailed information see the help vignette by running the following:

```
vignette('mlmpower')
```

## See Also

Variables `effect_size()` `correlations()` `random_slope()` `product()`

---

`mp_corr_func` *Functions for Generating Correlations*

---

## Description

Specify a random correlation that is uniform between `lower` and `upper`

## Usage

```
random(lower, upper)

fixed(value)
```

## Arguments

| | |
|---|---|
| `lower` | the lower bound of the distribution. |
| `upper` | the upper bound of the distribution. |
| `value` | the fixed value for the correlation. |

## Value

A `mp_corr_func` that generates the desired correlation

## See Also

`correlations()`

## Examples

```
# Create Model with random and fixed correlations
(
    outcome('Y')
    + within_predictor('X')
    + between_predictor('Z')
    + effect_size(icc = 0.1)
    # Defaults
    + correlations(
        within  = random(0.1, 0.3),
        between = fixed(0.2)
    )
)
```

---

`mp_parameters` `mp_parameters` *Object for* `mlmpower`

---

## Description

An S3 class that contains an `base::environment` with the following objects:

- `r2`: The population proportion of variance explained
- `phi_b`: The population predictor covariance matrix for between
- `phi_p`: The population within cluster covariance matrix for products
- `phi_w`: The population predictor covariance matrix for within
- `mean_Z`: The population mean for level-2 predictors
- `mean_X`: The population mean for level-1 predictors
- `var_e`: The population within residual variance
- `tau`: The population level-2 covariance matrix
- `gammas`: The regression coefficients
- `mean_Y`: The population mean of the outcome

## See Also

parameters

parameters *Obtain* mp_parameters *from objects*

## Description

A generic function to obtain mp_parameters from defined models and data sets.

## Usage

```
parameters(object)

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

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

## S3 method for class 'mp_power'
parameters(object)
```

## Arguments

| | |
|---|---|
| object | an object which the mp_parameters are desired. |

## Details

Currently object can be:

- mp_model
- mp_data
- mp_power

If using on a mp_model and the model has random correlations then the average is used.

## Value

A mp_parameters object

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)

# Create data set and obtain population parameters
```

```
model |> parameters()
# Set seed
set.seed(198723)
# Create data set and obtain population parameters
model |> generate(5, 50) |> parameters()
# Set seed
set.seed(198723)
# Create data set and obtain population parameters
model |> power_analysis(50, 5, 50) |> parameters()
```

---

`power_analysis` *Conduct a Power Analysis Based on* `mp_model`

---

## Description

This function will construct a multilevel power analysis via a Monte Carlo Simulation based on a constructed `mp_model`.

## Usage

```
power_analysis(model, replications, n_within, n_between, ...)
```

## Arguments

| | |
|---|---|
| `model` | a `mp_model`. |
| `replications` | a single positive integer of the number of replications per condition. |
| `n_within` | an integer vector of the desired within cluster observations. |
| `n_between` | an integer vector of the desired between cluster observations. |
| `...` | other arguments passed to `analyze()`. |

## Details

Specifying multiple `n_within` and `n_between` will produce a full factorial simulation design.

Specify a `mechanism` argument to pass down to the `generate` function. See the details of `generate` for more information about specifying missing data mechanisms. See `mechanisms` for predefined missing data mechanisms.

Specify an `analyze` argument to use custom analysis functions. These functions should map onto `analyze`'s structure, but can allow for things like specifying multiple imputations etc. This is considered an advance usage that requires extreme care and caution.

## Value

A `mp_power` object that contains the results. See `print.mp_power` for more information. The object has the following slots:

- `sim`: The information about the simulation
- `power`: The power power results per condition.
- `estimates`: The simulation summaries of the parameter estimates per condition.
- `mean_parameters`: The average population parameter per condition.

## See Also

generate()

mechanisms

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(19723)
# Create data set and analyze
# Note: Generally Use more than 50 replications
model |> power_analysis(50, 5, 50)

# Induce missing data on outcome with built in mechanisms
set.seed(19723)
model |> power_analysis(50, 5, 50, mechanism = MCAR(0.25)) -> powersim_mcar
```

---

# print.mp_correlations *Prints a* mp_correlations

---

## Description

Prints a mp_correlations in a human readable format.

## Usage

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

## Arguments

| | |
|---|---|
| x | a mp_correlations. |
| ... | other arguments not used by this method. |

## Value

Invisibly returns the original variable.

## Examples

```
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = c(0.1, 0.2),
        within = 0.3
    )
)
# Print correlations only
print(model$corrs)
```

---

`print.mp_effsize` *Prints a* `mp_effsize`

---

## Description

Prints a `mp_effsize` in a human readable format.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a `mp_effsize`. |
| `...` | other arguments not used by this method. |

## Value

Invisibly returns the original variable.

## Examples

```
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = c(0.1, 0.2),
        within = 0.3
    )
)
# Print effect size only
print(model$effect_size)
```

## print.mp_model *Prints a* mp_model

### Description

Prints a mp_variable in a human readable format.

### Usage

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

### Arguments

| | |
|---|---|
| x | a mp_model. |
| ... | other arguments not used by this method. |

### Value

Invisibly returns the original variable.

### Examples

```
print(
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = cross_sectional)
)
```

## print.mp_parameters *Prints a* mp_parameters

### Description

Prints a mp_parameters in a human readable format.

### Usage

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

### Arguments

| | |
|---|---|
| x | a mp_parameters. |
| ... | arguments passed to print(). |

## Value

Invisibly returns the original variable.

## Examples

```
print(
    summary(
        outcome('Y')
        + within_predictor('X')
        + effect_size(icc = cross_sectional)
    )
)
```

---

`print.mp_power` *Prints a* `mp_power`

---

## Description

Prints a `mp_power` in a human readable format.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a `mp_power`. |
| `...` | other arguments not used by this method. |

## Value

Invisibly returns the original variable.

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(19723)
# Create data set and analyze
# Note: Generally Use more than 50 replications
model |> power_analysis(50, 5, 50) -> powersim
# Print results
print(powersim)
```

## print.mp_variable *Prints a* mp_variable

### Description

Prints a mp_variable in a human readable format.

### Usage

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

### Arguments

| | |
|---|---|
| x | a mp_variable. |
| ... | other arguments not used by this method. |

### Value

Invisibly returns the original variable.

### Examples

```
print(
    within_predictor(
        'X',
        weight = 1,
        mean = 5,
        sd = 10,
        icc = 0.1
    )
)
```

## product *Create a Product Term in a Model*

### Description

Creates a product term between two variables that can be added to a mp_model.

### Usage

```
product(name1, name2, weight = 1)
```

### Arguments

| | |
|---|---|
| name1 | a character string that references the first variable's name |
| name2 | a character string that references the second variable's name |
| weight | a single numeric value specifying the variable's contribution to the variance explained metric. Weights are normalized across all variables of the same level. |

### Details

Currently the product term is only limited to cross-level interactions between a level-1 centered within cluster variable (`icc = 0`) and level-2 variable.

### Value

A `mp_action` that can be added to a `mp_model`.

### Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X', icc = 0.0)
    + between_predictor('Z')
)
# Add random slope to the model
model + product('X', 'Z')
```

---

## random_slope *Create a Random Slope in a Model*

---

### Description

Creates a random slope that can be added to a `mp_model`.

### Usage

```
random_slope(name, weight = 1)
```

### Arguments

| | |
|---|---|
| name | a character string that references a variable's name |
| weight | a single numeric value specifying the variable's contribution to the variance explained metric. Weights are normalized across all variables of the same level. |

### Value

A `mp_action` that can be added to a `mp_model`.

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = 0.1,
        within = 0.1,
        random_slope = 0.03
    )
)
# Add random slope to the model
model + random_slope('X')
```

---

`subset.mp_model` *Subset a* `mp_model` *by Global ICC*

---

## Description

Subsets a `mp_model` with multiple ICC values specified in `effect_size` into a model with only the single ICC value.

## Usage

```
## S3 method for class 'mp_model'
subset(x, icc, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `mp_model` object |
| `icc` | a single numeric value to subset out of `x` |
| `...` | other arguments not used by this method. |

## Value

A new `mp_model` with only the subset ICC

## Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = cross_sectional)
)
# Obtain Model with only 0.15 ICC
model |> subset(icc = 0.15)
```

## summary.mp_model *Obtain the Parameter Summaries for A* mp_model

### Description

Provide the summarized parameter estimates for a mp_model, including the variance explained break downs.

### Usage

```
## S3 method for class 'mp_model'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | a mp_model |
| ... | other arguments not used by this method. |

### Value

A mp_parameters object that contains the population parameters based on the model. If random correlations are used the average correlation is used to compute the parameters. If multiple ICC's are specified then a named base::list is returned containing the parameter value for each ICC value.

### Examples

```
summary(
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = cross_sectional)
)
```

## summary.mp_power *Summarizes a* mp_power

### Description

Summarizes a mp_power in a human readable format. This is a simple wrapper for print.mp_power.

### Usage

```
## S3 method for class 'mp_power'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | a mp_power. |
| ... | other arguments not used by this method. |

### Value

Invisibly returns the original variable.

### Examples

```
# Create Model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(icc = 0.1)
)
# Set seed
set.seed(19723)
# Create data set and analyze
# Note: Generally Use more than 50 replications
model |> power_analysis(50, 5, 50) -> powersim
# Summarizes results
summary(powersim)
```

---

## to_formula *Convert* mp_data *to a* stats::formula *to be used for* lme4::lmer

---

### Description

Produces the formula including the centering functions based on a data set generated with generate.

### Usage

```
to_formula(data, nested = FALSE)
```

### Arguments

| | |
|---|---|
| data | the mp_data to be coerced. |
| nested | logical value, if true then produce the nested restricted model |

### Value

a stats::formula

## Examples

```
# Specify model
model <- (
    outcome('Y')
    + within_predictor('X')
    + effect_size(
        icc = 0.2,
        within = 0.3
    )
)
# Set seed
set.seed(198723)
# Create formula based on data set
model |> generate(5, 50) |> to_formula()
```

---

`Variables` *Functions for Creating Variables*

---

## Description

These functions are the building blocks used to create the multilevel model and are used to specify the names, properties, and variable types.

## Usage

```
outcome(name, mean = 10, sd = 5, icc = NULL)

within_predictor(name, weight = 1, mean = 0, sd = 1, icc = NULL)

within_time_predictor(name, values, weight = 1)

between_predictor(name, weight = 1, mean = 0, sd = 1)

between_binary_predictor(name, proportion = 0.5, weight = 1)
```

## Arguments

| | |
|---|---|
| `name` | a character string for the specific variable's name |
| `mean` | a single numeric value that specifies the variable's mean |
| `sd` | a single numeric value that specifies the variable's standard deviation |
| `icc` | a single numeric value between 0 and 1 that specifies the variable's intraclass correlation. If `NULL` then the global ICC specified in `effect_size()` is used instead. |
| `weight` | a single numeric value specifying the variable's contribution to the variance explained metric. Weights are normalized across all variables of the same level. |
| `values` | a numeric vector specifying the time scores that will be repeated within each cluster. |

`proportion` a single numeric value between 0 and 1 that specifies the proportion of 1's at the population.

## Details

Note that specifying an `icc = 0` in `within_predictor()` will result in a centered within cluster (CWC) predictor.

See vignettes for more details.

```
vignette(package = 'mlmpower')
```

## Value

Returns a `mp_variable` object based on the variable's type.

# Index