Package: paramtest (via r-universe)

September 2, 2024

Type Package

Title Run a Function Iteratively While Varying Parameters

Version 0.1.0

Description Run simulations or other functions while easily varying parameters from one iteration to the next. Some common use cases would be grid search for machine learning algorithms, running sets of simulations (e.g., estimating statistical power for complex models), or bootstrapping under various conditions. See the 'paramtest' documentation for more information and examples.

License GPL-3

Encoding UTF-8

LazyData true

Imports stats, boot

Suggests parallel, beepr, pwr, ggplot2, knitr, nlme, lavaan, dplyr, testthat, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

Repository https://jeff-hughes.r-universe.dev

RemoteUrl https://github.com/jeff-hughes/paramtest

RemoteRef HEAD

RemoteSha bee7be25e9bd6ef6c69c9d59291f0c451db3cfca

Contents

`gen_data` *Generate data through a factor matrix and effects matrix.*

Description

`gen_data` will generate sample data based on a factor structure and effects structure specified by the user.

Usage

```
gen_data(factor_struct, effects_struct, n_cases = 1000, true_scores = FALSE)
```

Arguments

`factor_struct`	A matrix describing the measurement model of latent factors (columns) as measured by observed variables (rows).
`effects_struct`	A matrix describing the variances and covariances of the latent variables in the model.
`n_cases`	Number of sample cases to generate.
`true_scores`	Whether or not to include the data for each variable as measured without error. If set to TRUE, the resulting data frame will include all the variables in the model twice: once with measurement error, and once without.

Value

Returns a data frame with `n_cases` rows and columns for each observed and latent variable. These variables will approximately accord with the factor structure and effects structure that was specified, within sampling error.

Examples

```
# two uncorrelated predictors, one criterion, with measurement error in all
# variables
beta1 <- .5
beta2 <- .6
y_resid_var <- sqrt(1 - (beta1^2 + beta2^2))
fmodel <- matrix(
    c(.8, 0, 0,   # x1
      0, .6, 0,   # x2
      0, 0, .5),  # y
```

```
    nrow=3, ncol=3, byrow=TRUE, dimnames=list(
    c('x1', 'x2', 'y'), c('x1', 'x2', 'y')))
    # in this case, observed and latent variables are the same
effects <- matrix(
    c(1, 0, beta1,
      0, 1, beta2,
      0, 0, y_resid_var),
    nrow=3, ncol=3, byrow=TRUE, dimnames=list(
    c('x1', 'x2', 'y'), c('x1', 'x2', 'y')))

sample_data <- gen_data(fmodel, effects, n_cases=1000)
round(var(sample_data), 2)
round(cor(sample_data), 2)
summary(lm(y ~ x1 + x2, data=sample_data))
    # note that beta coefficients are much smaller, due to measurement error
```

`grid_search` *Run a function iteratively using a grid search approach for parameter values, with options for parallel processing.*

Description

`grid_search` runs a user-defined function iteratively. Parameter values can be given to `grid_search`, which will fully cross all parameters so that each parameter value is tested at all other values of all parameters.

Usage

```
grid_search(func, params = NULL, n.iter = 1, output = c("list",
  "data.frame"), boot = FALSE, bootParams = NULL, parallel = c("no",
  "multicore", "snow"), ncpus = 1, cl = NULL, beep = NULL, ...)
```

Arguments

`func`	A user-defined function. The first argument to this function will be the iteration number.
`params`	A list of parameters to be passed to `func`. The parameters are fully crossed so that each parameter value is tested at all other values of all parameters. (For example, list(N=c(5, 10), x=c(1, 2)) will test four sets of parameters: N=5 and x=1, N=5 and x=2, N=10 and x=1, and N=10 and x=2.) Each set of parameters will then be passed to `func` in turn.
`n.iter`	Number of iterations (per set of params).
`output`	Specifies how `grid_search` provides the ultimate output from `func`: can return a "list" or a "data.frame". Note that if "data.frame" is specified, the supplied function must return a vector, matrix, or data frame, so it can be coerced into the data frame format. The "list" option will accept any type of output.

boot	Whether or not to use bootstrapped data to pass along to func. Using this option instead of bootstrapping within func is preferable to take advantage of parallelization.
bootParams	If boot=TRUE, then use bootParams to pass along a named list of arguments to the boot function. The statistic and R parameters will be filled automatically, but at minimum you will need to pass along data. Information about parallel processing will also be passed along automatically.
parallel	The type of parallel operation to be used (if any).
ncpus	Integer: the number of processes to be used in parallel operation.
cl	An optional parallel or snow cluster for use if parallel = 'snow'. If not supplied, a cluster on the local machine is created for the duration of the iterations.
beep	Include a numeric value or character vector indicating the sound you wish to play once the tests are done running. Requires the 'beepr' package, and information about supported values is available in the documentation for that package.
...	Additional arguments to be passed to func. If you do not need to vary certain parameters in your model, you can pass them to func here.

Value

Returns a list (by default) with one element per iteration. If output is specified as "data.frame", then func must return a (named) vector with the results you wish to capture.

See Also

boot

Examples

```
lm_test <- function(iter, N, b0, b1) {
    x <- rnorm(N, 0, 1)
    y <- rnorm(N, b0 + b1*x, sqrt(1 - b1^2))
    data <- data.frame(y, x)
    model <- lm(y ~ x, data)

    # capture output from model summary
    est <- coef(summary(model))['x', 'Estimate']
    se <- coef(summary(model))['x', 'Std. Error']
    p <- coef(summary(model))['x', 'Pr(>|t|)']

    return(c(xm=mean(x), xsd=sd(x), ym=mean(y), ysd=sd(y), est=est, se=se, p=p,
        sig=est > 0 & p <= .05))
}

# test power for sample size N=200 and N=300, with 500 iterations for each
power_sim <- grid_search(lm_test, params=list(N=c(200, 300)), n.iter=500, b0=0, b1=.15)
```

lm_error_var *Calculate error variance given model coefficients.*

Description

`lm_error_var` will calculate the required error variance for a linear model, given specified model coefficients, to create variance for your dependent variable of approximately 'var'.

Usage

```
lm_error_var(var = 1, ...)
```

Arguments

`var`	The variance you wish your dependent variable to be.
`...`	Pass along all model coefficients, excluding the intercept. These can be named or unnamed.

Details

Note: This function assumes that *all predictors are independent* (i.e., uncorrelated).

Value

Returns the required error variance so that the variance of your dependent variable is approximately 'var'.

Examples

```
lm_error_var(var=1, .15, .3)  # returns error variance of 0.8875
```

n.iter *Return the number of iterations performed by a parameter test.*

Description

`n.iter` extracts information about the number of iterations (per specific test) performed by a parameter test.

Usage

```
n.iter(test, ...)

## S3 method for class 'paramtest'
n.iter(test, ...)
```

Arguments

test	An object of type 'paramtest'.
...	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns the number of iterations done in each test.

Methods (by class)

- paramtest: Number of iterations for a parameter test.

---

print.paramtest_summary

Print summary of parameter tests.

Description

print.paramtest_summary prints a summary of the various combinations of parameter values tested in a given parameter test.

Usage

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

Arguments

x	An object of class 'paramtest_summary', from summary.paramtest.
...	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns a data frame with one row per set of unique tests.

See Also

summary.paramtest

random_search *Run a function iteratively using a random search approach for parameter values, with options for parallel processing.*

Description

random_search runs a user-defined function iteratively. Lower and upper bounds for parameter values can be given to random_search, which will then (uniformly) randomly select values within those bounds on each iteration.

Usage

```
random_search(func, params = NULL, n.sample = 1, n.iter = 1,
  output = c("list", "data.frame"), boot = FALSE, bootParams = NULL,
  parallel = c("no", "multicore", "snow"), ncpus = 1, cl = NULL,
  beep = NULL, ...)
```

Arguments

func	A user-defined function. The first argument to this function will be the iteration number.
params	A named list of parameters to be passed to func. For continuous numeric values, a parameter must provide a two-element named vector with names "lower" and "upper" to specify the lower and upper bounds within which to sample. For parameters with integer values, provide a sequence, e.g., seq(5, 10). For parameters with non-numeric values, provide a vector with the values from which to sample. On each iteration, the random_search function will select a uniformly random value for each parameter and pass this set of parameter values to func.
n.sample	Number of times to sample from the parameter values.
n.iter	Number of iterations (per set of params).
output	Specifies how random_search provides the ultimate output from func: can return a "list" or a "data.frame". Note that if "data.frame" is specified, the supplied function must return a vector, matrix, or data frame, so it can be coerced into the data frame format. The "list" option will accept any type of output.
boot	Whether or not to use bootstrapped data to pass along to func. Using this option instead of bootstrapping within func is preferable to take advantage of parallelization.
bootParams	If boot=TRUE, then use bootParams to pass along a named list of arguments to the boot function. The statistic and R parameters will be filled automatically, but at minimum you will need to pass along data. Information about parallel processing will also be passed along automatically.
parallel	The type of parallel operation to be used (if any).
ncpus	Integer: the number of processes to be used in parallel operation.

cl	An optional parallel or snow cluster for use if parallel = 'snow'. If not supplied, a cluster on the local machine is created for the duration of the iterations.
beep	Include a numeric value or character vector indicating the sound you wish to play once the tests are done running. Requires the 'beepr' package, and information about supported values is available in the documentation for that package.
...	Additional arguments to be passed to func. If you do not need to vary certain parameters in your model, you can pass them to func here.

Value

Returns a list (by default) with one element per iteration. If output is specified as "data.frame", then func must return a (named) vector with the results you wish to capture.

See Also

boot

Examples

```
lm_test <- function(iter, N, b0, b1) {
    x <- rnorm(N, 0, 1)
    y <- rnorm(N, b0 + b1*x, sqrt(1 - b1^2))
    data <- data.frame(y, x)
    model <- lm(y ~ x, data)

    # capture output from model summary
    est <- coef(summary(model))['x', 'Estimate']
    se <- coef(summary(model))['x', 'Std. Error']
    p <- coef(summary(model))['x', 'Pr(>|t|)']

    return(c(xm=mean(x), xsd=sd(x), ym=mean(y), ysd=sd(y), est=est, se=se, p=p,
        sig=est > 0 & p <= .05))
}

# test power for sample sizes between N=200 and N=300, with 500 iterations total
power_sim <- random_search(lm_test, params=list(N=c(200, 300)), n.iter=500, b0=0, b1=.15)
```

---

results *Return results of a parameter test.*

Description

results returns the raw data from a parameter test.

Usage

```
results(test, ...)

## S3 method for class 'paramtest'
results(test, ...)
```

Arguments

test	An object of type 'paramtest'.
...	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns a data frame with all the data returned from each test.

Methods (by class)

- paramtest: Results for a parameter test.

---

run_test *Run a function iteratively, with options for parallel processing.*

Description

run_test runs a user-defined function iteratively. This function is intentionally kept general and flexible, to allow for a wide variety of applications. This function is the general-purpose function called by functions such as grid_search and random_search, which provide different methods for generating the parameters to be tested.

Usage

```
run_test(func, params = NULL, n.iter = 1, output = c("list",
  "data.frame"), boot = FALSE, bootParams = NULL, parallel = c("no",
  "multicore", "snow"), ncpus = 1, cl = NULL, beep = NULL, ...)
```

Arguments

func	A user-defined function. The first argument to this function will be the iteration number.
params	A list or data frame of parameters to be passed to func. Each set of parameters will be passed to func in turn.
n.iter	Number of iterations (per set of params).
output	Specifies how run_test provides the ultimate output from func: can return a "list" or a "data.frame". Note that if "data.frame" is specified, the supplied function must return a vector, matrix, or data frame, so it can be coerced into the data frame format. The "list" option will accept any type of output.

boot	Whether or not to use bootstrapped data to pass along to func. Using this option instead of bootstrapping within func is preferable to take advantage of parallelization.
bootParams	If boot=TRUE, then use bootParams to pass along a named list of arguments to the boot function. The statistic and R parameters will be filled automatically, but at minimum you will need to pass along data. Information about parallel processing will also be passed along automatically.
parallel	The type of parallel operation to be used (if any).
ncpus	Integer: the number of processes to be used in parallel operation.
cl	An optional parallel or snow cluster for use if parallel = 'snow'. If not supplied, a cluster on the local machine is created for the duration of the iterations.
beep	Include a numeric value or character vector indicating the sound you wish to play once the tests are done running. If set to TRUE, a random sound will be played. Requires the 'beepr' package, and information about supported values is available in the documentation for that package.
...	Additional arguments to be passed to func. If you do not need to vary certain parameters in your model, you can pass them to func here.

Value

Returns a list (by default) with one element per iteration. If output is specified as "data.frame", then func must return a (named) vector with the results you wish to capture.

See Also

boot

Examples

```
lm_test <- function(iter, N, b0, b1) {
    x <- rnorm(N, 0, 1)
    y <- rnorm(N, b0 + b1*x, sqrt(1 - b1^2))
    data <- data.frame(y, x)
    model <- lm(y ~ x, data)

    # capture output from model summary
    est <- coef(summary(model))['x', 'Estimate']
    se <- coef(summary(model))['x', 'Std. Error']
    p <- coef(summary(model))['x', 'Pr(>|t|)']

    return(c(xm=mean(x), xsd=sd(x), ym=mean(y), ysd=sd(y), est=est, se=se, p=p,
        sig=est > 0 & p <= .05))
}

# test power for sample size N=200 and N=300, with 500 iterations for each
power_sim <- run_test(lm_test, params=data.frame(N=c(200, 300)),
    n.iter=500, b0=0, b1=.15)
```

summary.paramtest *Print summary of parameter tests.*

Description

summary.paramtest provides a summary of the various combinations of parameter values tested in a given parameter test.

Usage

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

Arguments

object	An object of class 'paramtest'.
...	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns a data frame with one row per set of unique tests.

See Also

run_test

Examples

```
lm_test <- function(iter, N, b0, b1) {
    x <- rnorm(N, 0, 1)
    y <- rnorm(N, b0 + b1*x, sqrt(1 - b1^2))
    data <- data.frame(y, x)
    model <- lm(y ~ x, data)

    # capture output from model summary
    est <- coef(summary(model))['x', 'Estimate']
    se <- coef(summary(model))['x', 'Std. Error']
    p <- coef(summary(model))['x', 'Pr(>|t|)']

    return(c(xm=mean(x), xsd=sd(x), ym=mean(y), ysd=sd(y), est=est, se=se, p=p,
        sig=est > 0 & p <= .05))
}

# test power for sample sizes between N=200 and N=300, with 500 iterations total
power_sim <- random_search(lm_test, params=list(N=c(200, 300)), n.iter=500, b0=0, b1=.15)
summary(power_sim)
```

tests *Return the parameter values that were tested by paramtest.*

Description

`tests` extracts information about the set of specific tests (parameter values) for a parameter test.

Usage

```
tests(test, ...)

## S3 method for class 'paramtest'
tests(test, ...)
```

Arguments

`test`	An object of type 'paramtest'.
`...`	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns a data frame with one row for each set of tests that was performed.

Methods (by class)

- `paramtest`: Parameter values for a parameter test.

timing *Return the timing information of a parameter test.*

Description

`timing` returns the information about how long a parameter test took.

Usage

```
timing(test, ...)

## S3 method for class 'paramtest'
timing(test, ...)
```

Arguments

`test`	An object of type 'paramtest'.
`...`	Not currently implemented; used to ensure consistency with S3 generic.

Value

Returns an object of class "proc_time" with information about how long the parameter test process took.

Methods (by class)

- `paramtest`: Timing information for a parameter test.

Index