Package 'comparer'

June 18, 2019

Type Package

Title Compare Output and Run Time

Version 0.2.0

Description Quickly run experiments to compare the run time and output of code blocks. The function mbc() can make fast comparisons of code, and will calculate statistics comparing the resulting outputs. It can be used to compare model fits to the same data or see which function runs faster. The function ffexp() runs a function using all possible combinations of selected inputs. This is useful for comparing the effect of different parameter values. It can also run in parallel and automatically save intermediate results, which is very useful for long computations.

License GPL-3

Encoding UTF-8

LazyData true

Imports R6

Suggests plyr, progress, testthat, covr, knitr, ggplot2, parallel, snow, rmarkdown, reshape, microbenchmark

RoxygenNote 6.1.1

URL https://github.com/CollinErickson/comparer

BugReports https://github.com/CollinErickson/comparer/issues

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Collin Erickson [aut, cre]

Maintainer Collin Erickson <collinberickson@gmail.com>

Repository CRAN

Date/Publication 2019-06-18 18:00:03 UTC

R topics documented:

`ffexp` *Full factorial experiment*

Description

A class for easily creating and evaluating full factorial experiments.

Usage

```
e1 <- ffexp$new(eval_func=, )

e1$run_all()

e1$plot_run_times()

e1$save_self()
```

Arguments

`eval_func` The function called to evaluate each design point.

`...` Factors and their levels to be evaluated at.

`save_output` Should the output be saved?

`parallel` If `TRUE`, function evaluations are done in parallel.

`parallel_cores` Number of cores to be used in parallel. If `"detect"`, `parallel::detectCores()` is used to determine number. `"detect-1"` may be used so that the computer isn't running at full capacity, which can slow down other tasks.

Methods

`$new()` Initialize an experiment. The preprocessing is done, but no function evaluations are run.

`$run_all()` Run all factor combinations.

`$run_one()` Run a single factor combination.

`$add_result_of_one()` Used to add result of evaluation to data set, don't manually call.

`$plot_run_times()` Plot the run times. Especially useful when they have been run in parallel.

`$save_self()` Save ffexp R6 object.

`$recover_parallel_temp_save()` If you ran the experiment using parallel with `parallel_temp_save=TRUE` and it crashes partway through, call this to recover the runs that were completed. Runs that were stopped mid-execution are not recoverable.

Examples

```
# Two factors, both with two levels.
#   The evaluation function simply prints out the combination
cc <- ffexp$new(a=1:2,b=c("A","B"),
                eval_func=function(...) {c(...)})
# View the factor settings it will run (each row).
cc$rungrid
# Evaluate all four settings
cc$run_all()


cc <- ffexp$new(a=1:3,b=2, cd=data.frame(c=3:4,d=5:6),
                eval_func=function(...) {list(...)})
```

`mbc` *Model benchmark compare*

Description

Compare the run time and output of various code chunks

Usage

```
mbc(..., times = 5, input, inputi, evaluator, post, target, targetin,
  metric = "rmse", paired, kfold)
```

Arguments

`...`	Functions to run
`times`	Number of times to run
`input`	Object to be passed as input to each function
`inputi`	Function to be called with the replicate number then passed to each function.
`evaluator`	An expression that the ... expressions will be passed as "." for evaluation.
`post`	Function or expression (using ".") to post-process results.
`target`	Values the functions are expected to (approximately) return.
`targetin`	Values that will be given to the result of the run to produce output.
`metric`	c("rmse", "t", "mis90", "sr27") Metric used to compare output values to target. mis90 is the mean interval score for 90% confidence, see Gneiting and Raftery (2007). sr27 is the scoring rule given in Equation 27 of Gneiting and Raftery (2007).
`paired`	Should the results be paired for comparison?

kfold — First element should be the number of elements that are being split into groups. If the number of folds is different from ‘times‘, then the second argument is the number of folds. Use ‘ki‘ in ‘inputi‘ and ‘targeti‘ to select elements in the current fold.

Value

Data frame of comparison results

References

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association, 102(477), 359-378.

Examples

```
# Compare distribution of mean for different sample sizes
mbc(mean(rnorm(1e2)),
    mean(rnorm(1e4)),
    times=20)

# Compare mean and median on same data
mbc(mean(x),
    median(x),
    inputi={x=rexp(1e2)})

# input given, no post
mbc({Sys.sleep(rexp(1, 30));mean(x)},
     {Sys.sleep(rexp(1, 5));median(x)},
    inputi={x=runif(100)})

# input given with post
mbc(mean={Sys.sleep(rexp(1, 30));mean(x)},
    med={Sys.sleep(rexp(1, 5));median(x)},
    inputi={x=runif(100)},
    post=function(x){c(x+1, x^2)})

# input given with post, 30 times
mbc(mean={Sys.sleep(rexp(1, 30));mean(x)+runif(1)},
    med={Sys.sleep(rexp(1, 50));median(x)+runif(1)},
    inputi={x=runif(100)},
    post=function(x){c(x+1, x^2)}, times=10)

# Name one function and post
mbc({mean(x)+runif(1)},
    a1={median(x)+runif(1)},
    inputi={x=runif(100)},
    post=function(x){c(rr=x+1, gg=x^2)}, times=10)

# No input
m1 <- mbc(mean={x <- runif(100);Sys.sleep(rexp(1, 30));mean(x)},
          med={x <- runif(100);Sys.sleep(rexp(1, 50));median(x)})
```

plot.mbc *Plot mbc class*

Description

Plot mbc class

Usage

```
## S3 method for class 'mbc'
plot(x, ...)
```

Arguments

x	Object of class mbc
...	Additional parameters

Value

None

Examples

```
m1 <- mbc(mn= {Sys.sleep(rexp(1, 30));mean(x)},
          med={Sys.sleep(rexp(1, 5));median(x)},
          input=runif(100))
plot(m1)
```

print.mbc *Print mbc class*

Description

Print mbc class

Usage

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

Arguments

x	Object of class mbc
...	Additional parameters

Value

None

Examples

```
m1 <- mbc({Sys.sleep(rexp(1, 30));mean(x)},
          {Sys.sleep(rexp(1, 5));median(x)},
          input=runif(100))
print(m1)
```

Index