Package 'atime'

July 22, 2025

Type Package

Title Asymptotic Timing

Version 2025.5.24

Description Computing and visualizing comparative
asymptotic timings of different algorithms and code versions.
Also includes functionality for comparing empirical timings with
expected references such as linear or quadratic,
<https://en.wikipedia.org/wiki/Asymptotic_computational_complexity>
Also includes functionality for measuring asymptotic memory and other
quantities.

License GPL-3

URL https://github.com/tdhock/atime

BugReports https://github.com/tdhock/atime/issues

Imports data.table, bench, lattice, git2r, utils, stats, grDevices

Suggests directlabels, ggplot2, testthat, knitr, markdown, stringi,
re2, binsegRcpp, grates, fastymd, wbs, fpop, changepoint,
LOPART, cumstats, PeakSegDisk, callr, readr, dplyr, tidyr, nc,
RColorBrewer, tibble, Matrix

VignetteBuilder knitr

NeedsCompilation no

Author Toby Hocking [aut, cre]

Maintainer Toby Hocking <toby.hocking@r-project.org>

Repository CRAN

Date/Publication 2025-05-27 07:20:02 UTC

Contents

`atime` *Asymptotic timing*

Description

Computation time and memory for several R expressions of several different data sizes.

Usage

```
atime(
 N, setup, expr.list=NULL, times=10, seconds.limit=0.01, verbose=FALSE,
 result=FALSE, N.env.parent=NULL, ...)
```

Arguments

`N`	numeric vector of at least two unique data sizes, default is `2^seq(2,20)`.
`setup`	expression to evaluate for every data size, before timings.
`expr.list`	named list of expressions to time.
`times`	number of times to evaluate each timed expression.
`seconds.limit`	if the median timing of any expression exceeds this many seconds, then no timings for larger N are computed.
`verbose`	logical, print messages after every data size?
`result`	logical: save the result of evaluating each expression? Or a function to compute a result, given the value obtained after evaluating each expression. If each result is a data frame with one row, then the numeric column names will be saved as more units to analyze (in addition to kilobytes and seconds).
`N.env.parent`	environment to use as parent of environment created for each data size N, or NULL to use default parent env.
`...`	named expressions to time.

Details

Each iteration involves first computing the setup expression, and then computing several times the `...` expressions. For convenience, expressions may be specified either via code (`...`) or data (`expr.list` arg).

Value

list of class atime with elements `unit.col.vec` (character vector of column names to analyze), `seconds.limit` (numeric input param), `measurements` (data table of results).

Author(s)

Toby Dylan Hocking

See Also

`atime_grid` for avoiding repetition when measuring asymptotic properties of several similar expressions.

Examples

```
data.table::setDTthreads(1) # for CRAN.

## Polynomial and exponential time string functions.
atime_result_string <- atime::atime(
  seconds.limit=0.001,
  N=unique(as.integer(10^seq(0,3,l=100))),
  setup={
    subject <- paste(rep("a", N), collapse="")
    pattern <- paste(rep(c("a?", "a"), each=N), collapse="")
    linear_size_replacement <- paste(rep("REPLACEMENT", N), collapse="")
  },
  PCRE.match=regexpr(pattern, subject, perl=TRUE),
  TRE.match=regexpr(pattern, subject, perl=FALSE),
  constant.replacement=gsub("a","constant size replacement",subject),
  linear.replacement=gsub("a",linear_size_replacement,subject))
plot(atime_result_string)
```

---

atime_grid *Asymptotic timing grid*

Description

Create expressions for asymptotic timing by substituting values into expressions.

Usage

```
atime_grid(
  param.list = list(),
  ...,
  name.value.sep="=",
  expr.param.sep=" ",
  collapse = ",",
  symbol.params=character())
```

Arguments

param.list	Named list of items to replace in ... expressions, default empty list means nothing to replace.
...	Named expressions which each must contain each name of param.list.
name.value.sep	string: separator between names and values from param.list, default "=".
expr.param.sep	string: separator between expressions and parameters, default " ".
collapse	string: separator between parameters, default ",".
symbol.params	character vector: these elements of param.list will be converted to symbols before substitution.

Value

Named list of expressions which can be used as expr.list argument of atime.

Author(s)

Toby Dylan Hocking

Examples

```
data.table::setDTthreads(1) # for CRAN.

## Example 0: with no param.list, same as quote inside named list.
atime::atime_grid(m=mean(data), s=sum(data))
list(m=quote(mean(data)), s=quote(sum(data)))

## Example 1: polynomial vs exponential time regex.
(expr.list <- atime::atime_grid(
  list(PERL=c(TRUE, FALSE)),
  expr.param.sep="\n",
  regexpr=regexpr(pattern, subject, perl=PERL)))
atime.list <- atime::atime(
  seconds.limit=0.001,
  N=unique(as.integer(10^seq(0,2,l=30))),
  setup={
    subject <- paste(rep("a", N), collapse="")
    pattern <- paste(rep(c("a?", "a"), each=N), collapse="")
  },
  expr.list=expr.list)
plot(atime.list)

## Example 2: symbol.params arg.
sub.param.list <- list(FUN=c("sub","gsub"), PERL=c(TRUE,FALSE))
## with base R we can use as.symbol and substitute:
sapply(sub.param.list$FUN,function(name)substitute(fun("a","",subject), list(fun=as.symbol(name))))
## with atime_grid the analog is to use symbol.params argument:
(sub.expr.list <- atime::atime_grid(
  sub.param.list,
  replace=FUN("a","",subject,perl=PERL),
  symbol.params="FUN"))
```

```
sub.atime.list <- atime::atime(
  seconds.limit=0.001,
  setup={
    subject <- paste(rep("a",N),collapse="")
  },
  expr.list=sub.expr.list)
plot(sub.atime.list)

## Customize ggplot, which shows expr.name column as direct labels.
sub.atime.edited <- sub.atime.list
library(data.table)
sub.atime.edited$measurements <- data.table(sub.atime.list$measurements)[
, expr.name := paste0("PERL=",PERL)]
if(require(ggplot2)){
  plot(sub.atime.edited)+
    facet_grid(unit ~ FUN, labeller=label_both)
}
```

`atime_pkg` *Asymptotic timing package tests*

Description

R package performance testing, by computing time/memory usage of several R expressions of several different data sizes, for several package versions (base, HEAD, CRAN, merge-base, other historical references specified by user). `atime_pkg_test_info` returns an environment containing test code/calls (useful for running a single test), whereas `atime_pkg` runs all tests and saves results/plots to disk.

Usage

```
atime_pkg(pkg.path=".", tests.dir=NULL)
atime_pkg_test_info(pkg.path=".", tests.dir=NULL)
```

Arguments

`pkg.path`	path to git repository containing R package.
`tests.dir`	path to directory which contains `atime/tests.R`, relative to `pkg.path` (default `NULL` means first existing directory of `"inst"` or `".ci"`).

Details

There should be a `tests.R` code file which defines `test.list`, a list with names corresponding to different tests. Each element should be a list with at least three named elements: `N`, `setup`, `expr`, (possibly others such as `pkg.edit.fun` and `version_name="1234567890abcdef"`) to be passed as named arguments to `atime_versions`, along with the following versions which are passed using the `sha.vec` argument: `base` ref comes from `GITHUB_BASE_REF` environment variable (default

master), HEAD ref is the branch that you want to merge into base, CRAN is current published version (sha value ""), merge-base is most recent common ancestor commit between base and HEAD. For visualization, default colors are provided for versions with names: HEAD, base, CRAN, merge-base, Before, Regression, Slow, Fixed, Fast; other version names will be gray using the default colors. If tests.R defines a variable named version.colors, then it should be a character vector of colors to be used instead of the default (names for versions, values for colors). If tests.R defines a variable named pval.thresh, then it should be a positive numeric value between 0 and 1 (any smaller p-values will generate messages about slowdowns in HEAD_issues.md).

Value

atime_pkg_test_info returns an environment in which the code of tests.R was evaluated, including a variable test.call which is a list of un-evaluated atime_versions calls, one for each test (use with eval to run a single test). atime_pkg returns a named list of test results, names come from names of test.list, and values come from results of atime_versions. Side effect is that data/plot files are saved in atime directory, including tests.RData (test results which can be read into R if you want to make your own alternative plots/analyses), tests_all_facet.png (plot summarizing all test results), tests_preview_facet.png (plot summarizing only most significant results), and install_seconds.txt (total number of seconds used to install different package versions). Plots emphasize differences between HEAD and a comparison version, which is defined as the first of these versions which is available: merge-base, base, CRAN. Plots show test cases, first sorted by N.factor, which is the ratio of N values at the time limit, HEAD versus comparison version. If N.factor<1 then HEAD is much slower than comparison version (the smaller N value for HEAD went over the time limit). N.factor=1 means HEAD and comparison version had same value for largest N (so have nearly the same speed); ties are broken using P.value, un-paired two-sample T-test for HEAD time greater than comparison time at largest N.

Author(s)

Toby Dylan Hocking

See Also

atime_test for defining each test, atime_test_list for defining common arguments in each element of the test list.

Examples

```
if(FALSE){

  ## Clone binsegRcpp, and checkout a branch which has performance tests.
  tdir <- tempfile()
  dir.create(tdir)
  git2r::clone("https://github.com/tdhock/binsegRcpp", tdir)
  repo <- git2r::repository(tdir)
  git2r::checkout(repo, "another-branch")

  ## Run one test.
  info.env <- atime::atime_pkg_test_info(tdir)
  test_name <- "binseg(1:N,maxSegs=N/2) DIST=l1"
```

```
  (one.call <- info.env$test.call[[test_name]])
  one.result <- eval(one.call)
  plot(one.result) # constant difference for small N should be visible.

  ## Run all tests.
  result.list <- atime::atime_pkg(tdir)
  inst.atime <- file.path(tdir, "inst", "atime")
  dir(inst.atime)
  tests.RData <- file.path(inst.atime, "tests.RData")
  (objs <- load(tests.RData))
  plot(pkg.results[[test_name]]) # should look similar.

  atime::atime_versions_remove("binsegRcpp")

}

## https://github.com/tdhock/binsegRcpp/blob/atime-test-funs/.ci/atime/tests.R
## has another real example, see how to run it in tests/testthat/test-versions.R
```

`atime_test` *Define an atime performance test.*

Description

Use this to define an element of your `test.list` in atime/tests.R, prior to running `atime_pkg`.

Usage

```
atime_test(
  N, setup, expr, times, seconds.limit, verbose,
  pkg.edit.fun, result, ...)
```

Arguments

`N`	numeric vector of data sizes to vary.
`setup`	expression to evaluate for every data size, before timings. In contrast to `expr`, no replacement of `Package:` is performed.
`expr`	code with package double-colon prefix, for example `PKG::fun(argA, argB)`, where PKG is the name of the package specified by `pkg.path`. This code will be evaluated for each different package version, by replacing `PKG:` by `PKG.SHA:`. To run different versions of implicitly-called functions like `DT[i,j]`, you need to call them explicitly, as in `data.table:::`[.data.table`(DT,i,j)`.
`times`	number of times to evaluate each timed expression.
`seconds.limit`	if the median timing of any expression exceeds this many seconds, then no timings for larger N are computed.
`verbose`	logical, print messages after every data size?

pkg.edit.fun	function called to edit package before installation, should typically replace instances of PKG with PKG.SHA, default works with Rcpp packages.
result	logical, save results? (default FALSE)
...	named versions.

Value

List of expressions.

Author(s)

Toby Dylan Hocking

See Also

atime_test_list for defining common arguments in each element of the test list, atime_pkg for running tests.

Examples

```
atime::atime_test(
  N=c(1,10),
  setup=data.vec <- rnorm(N),
  expr=binsegRcpp::binseg("mean_norm",data.vec))

## https://github.com/tdhock/binsegRcpp/blob/atime-test-funs/.ci/atime/tests.R
## has a real example, see how to run it in tests/testthat/test-versions.R
```

---

atime_test_list *Define an atime performance test list.*

Description

Use this to define test.list in your atime/tests.R file, prior to running atime_pkg. Arguments in ... should all be named; if name is an argument of atime_versions, it will be copied to each test; otherwise it should be the name of a test.

Usage

```
atime_test_list(
  N, setup, expr, times, seconds.limit,
  verbose, pkg.edit.fun, result,
  tests = NULL, ...)
```

Arguments

...	names for tests, values are lists of arguments to pass to atime_versions (combined with tests).
tests	list of tests, with names for tests, values are lists of arguments to pass to atime_versions (combined with ...).
N	integer vector of data sizes.
setup	expression that depends on N, to run before timings. Not evaluated before copying to each test.
expr	expression to time. Not evaluated before copying to each test.
times	number of times to run expr.
seconds.limit	number of seconds after which we stop trying larger N.
verbose	logical: print output?
pkg.edit.fun	function for editing package prior to testing.
result	logical: save results?

Value

List representing performance tests, from ... and tests; each element is a list of arguments to pass to atime_versions.

Author(s)

Toby Dylan Hocking

See Also

atime_test for defining each test, atime_pkg for running tests.

Examples

```
(test.list.named <- atime::atime_test_list(
  N=as.integer(10^seq(1,3,by=0.5)),
  setup={
    set.seed(1)
    data.vec <- rnorm(N)
  },
  mean_norm=atime::atime_test(expr=binsegRcpp::binseg("mean_norm",data.vec)),
  poisson=atime::atime_test(expr=binsegRcpp::binseg("poisson",data.vec)),
  NULL))

## https://github.com/tdhock/binsegRcpp/blob/atime-test-funs/.ci/atime/tests.R
## has a real example, see how to run it in tests/testthat/test-versions.R
```

atime_versions *Asymptotic timing of git versions*

Description

Computation time and memory for a single R expression evaluated using several different git versions.

Usage

```
atime_versions(
 pkg.path, N, setup, expr, sha.vec=NULL,
 times=10, seconds.limit=0.01, verbose=FALSE,
 pkg.edit.fun=pkg.edit.default, result=FALSE,
 N.env.parent=NULL,
 ...)
```

Arguments

pkg.path	Path to R package in a git repo.
N	numeric vector of data sizes to vary.
setup	expression to evaluate for every data size, before timings. In contrast to expr, no replacement of Package:: is performed.
expr	code with package double-colon prefix, for example PKG::fun(argA, argB), where PKG is the name of the package specified by pkg.path. This code will be evaluated for each different package version, by replacing PKG:: by PKG.SHA::. To run different versions of implicitly-called functions like DT[i,j], you need to call them explicitly, as in data.table:::`[.data.table`(DT,i,j).
sha.vec	named character vector / list of versions.
times	number of times to evaluate each timed expression.
seconds.limit	if the median timing of any expression exceeds this many seconds, then no timings for larger N are computed.
verbose	logical, print messages after every data size?
pkg.edit.fun	function called to edit package before installation, should typically replace instances of PKG with PKG.SHA, default works with Rcpp packages.
result	logical or function, passed to atime.
N.env.parent	environment to use as parent of environment created for each data size N, or NULL to use default parent env.
...	named versions.

Details

For convenience, versions can be specified either as code (...), data (sha.vec), or both. Each version should be either "" (to use currently installed version of package, or if missing, install most recent version from CRAN) or a SHA1 hash, which is passed as branch arg to git2r::checkout; version names used to identify/interpret the output/plots.

Value

list of class atime with elements `seconds.limit` (numeric input param), `timings` (data table of results).

Author(s)

Toby Dylan Hocking

See Also

`atime_versions_exprs` converts `expr` into a list of expressions, one for each version, passed to `atime` as the `expr.list` argument.

Examples

```
if(FALSE){

  tdir <- tempfile()
  dir.create(tdir)
  git2r::clone("https://github.com/tdhock/binsegRcpp", tdir)
  atime.list <- atime::atime_versions(
    pkg.path=tdir,
    setup={
      max.segs <- as.integer(N/2)
      data.vec <- 1:N
    },
    expr=binsegRcpp::binseg_normal(data.vec, max.segs),
    cv="908b77c411bc7f4fcbcf53759245e738ae724c3e",
    "rm unord map"="dcd0808f52b0b9858352106cc7852e36d7f5b15d",
    mvl_construct="5942af606641428315b0e63c7da331c4cd44c091")
  plot(atime.list)

  atime::atime_versions_remove("binsegRcpp")

  gdir <- tempfile()
  dir.create(gdir)
  git2r::clone("https://github.com/tdhock/grates", gdir)
  glist <- atime::atime_versions(
    file.path(gdir,"pkg"),
    current = "1aae646888dcedb128c9076d9bd53fcb4075dcda",
    old     = "51056b9c4363797023da4572bde07e345ce57d9c",
    setup   = date_vec <- rep(Sys.Date(), N),
    expr    = grates::as_yearmonth(date_vec))
  plot(glist)

  atime::atime_versions_remove("grates")

}
```

`atime_versions_exprs` *Create expressions for different git versions*

Description

Install different git commit versions as different packages, then create a list of expressions, one for each version. For most use cases `atime_versions` is simpler, but `atime_versions_exprs` is more flexible for the case of comparing different versions of one expression to another expression.

Usage

```
atime_versions_exprs(
 pkg.path, expr, sha.vec=NULL,
 verbose=FALSE,
 pkg.edit.fun=pkg.edit.default, ...)
```

Arguments

`pkg.path`	Path to git repo containing R package.
`expr`	code with package double-colon prefix, for example `PKG::fun(argA, argB)`, where PKG is the name of the package specified by `pkg.path`. This code will be evaluated for each different package version, by replacing `PKG::` by `PKG.SHA::`. To run different versions of implicitly-called functions like `DT[i,j]`, you need to call them explicitly, as in `data.table:::`[.data.table`(DT,i,j)`.
`sha.vec`	named character vector / list of versions.
`verbose`	logical, print messages after every data size?
`pkg.edit.fun`	function called to edit package before installation, should typically replace instances of `PKG` with `PKG.SHA`, default works with Rcpp packages, but does not work with all packages. For an example of a custom package editing function, see the atime vignette about data.table.
`...`	named versions.

Details

For convenience, versions can be specified either as code (`...`), data (`sha.vec`), or both. Each version should be either `""` (to use currently installed version of package, or if missing, install most recent version from CRAN) or a SHA1 hash, which is passed as branch arg to `git2r::checkout`; version names used to identify/interpret the output/plots. Each version is installed as a separate package (to whatever R library is first on `.libPaths()`), using the package name `PKG.SHA`.

Value

A list of expressions, one for each version, created by replacing `PKG::` in `expr` with `PKG.SHA::`. This list can be used as `expr.list` argument of `atime()`, instead of writing code like `atime(name1=Package.SHA1::fun(ar`
`argB), name2=Package.SHA2::fun(argA, argB))`.

Author(s)

Toby Dylan Hocking

Examples

```
if(FALSE){

  if(requireNamespace("changepoint")){
    tdir <- tempfile()
    dir.create(tdir)
    git2r::clone("https://github.com/tdhock/binsegRcpp", tdir)
    expr.list <- atime::atime_versions_exprs(
      pkg.path=tdir,
      expr=binsegRcpp::binseg_normal(data.vec, max.segs),
      cv="908b77c411bc7f4fcbcf53759245e738ae724c3e",
      "rm unord map"="dcd0808f52b0b9858352106cc7852e36d7f5b15d",
      "mvl_construct"="5942af606641428315b0e63c7da331c4cd44c091")
    atime.list <- atime::atime(
      N=2^seq(2, 20),
      setup={
        max.segs <- as.integer(N/2)
        data.vec <- 1:N
      },
      expr.list=expr.list,
      changepoint=changepoint::cpt.mean(
        data.vec, penalty="Manual", pen.value=0, method="BinSeg",
        Q=max.segs-1))
    plot(atime.list)
  }

  atime::atime_versions_remove("binsegRcpp")

}
```

`atime_versions_remove` *Remove packages installed by atime*

Description

`atime_versions_exprs` installs different git versions of a package, and this function removes them. This can be useful if you have changed compilers, which may have different performance characteristics, so you can then re-compile all package versions using the same compiler.

Usage

```
atime_versions_remove(Package)
```

Arguments

Package	Name of package without SHA.

Details

The library searched is the first on .libPaths().

Value

integer exit status code from unlink, non-zero if removal failed.

Author(s)

Toby Dylan Hocking

---

glob_find_replace *Find and replace within files*

Description

Find and replace for every file specified by glob.

Usage

```
glob_find_replace(glob, FIND, REPLACE)
```

Arguments

glob	character string: glob defining files.
FIND	character string: regex to find.
REPLACE	character string: regex to use for replacement.

Value

nothing.

Author(s)

Toby Dylan Hocking

Examples

```
## see vignette("data.table", package="atime")
```

`references_best` *Best references*

Description

Compute best asymptotic references, for all empirical measurements which are present (not missing) and increasing with data size.

Usage

```
references_best(L, fun.list=NULL)
```

Arguments

`L`	List output from atime.
`fun.list`	List of asymptotic complexity reference functions, default NULL means to use package default.

Value

list of class `"references_best"` with elements `references` (data table of all references), `plot.references` (data table of references to show using `plot` method, default is to show closest larger and smaller references), `measurements` (data table of measurements).

Author(s)

Toby Dylan Hocking

Examples

```
## Polynomial and exponential time string functions.
atime_result_string <- atime::atime(
  seconds.limit=0.001,
  N=unique(as.integer(10^seq(0,4,l=100))),
  setup={
    subject <- paste(rep("a", N), collapse="")
    pattern <- paste(rep(c("a?", "a"), each=N), collapse="")
    linear_size_replacement <- paste(rep("REPLACEMENT", N), collapse="")
  },
  PCRE.match=regexpr(pattern, subject, perl=TRUE),
  TRE.match=regexpr(pattern, subject, perl=FALSE),
  constant.replacement=gsub("a","constant size replacement",subject),
  linear.replacement=gsub("a",linear_size_replacement,subject))
(refs_best_string <- atime::references_best(atime_result_string))
## plot method shows each expr and unit in a separate panel.
## default is to show closest larger and smaller references.
plot(refs_best_string)
## modifying plot.references changes violet references shown by plot.
refs_best_string$plot.references <- refs_best_string$ref[c("N","N^2","N^3","2^N"),on="fun.name"]
```

```
plot(refs_best_string)
## predict method computes N for given units (default seconds limit).
(pred_string <- predict(refs_best_string))
plot(pred_string)
```

Index