Package 'atime'

October 12, 2022

Type Package

Title Asymptotic Timing

Version 2022.9.16

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

Suggests directlabels, ggplot2, testthat, knitr, markdown, stringi, re2, binsegRcpp, wbs, fpop, changepoint, LOPART, cumstats, PeakSegDisk, callr

VignetteBuilder knitr

NeedsCompilation no

Author Toby Hocking [aut, cre]

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

Repository CRAN

Date/Publication 2022-09-19 22:56:18 UTC

R topics documented:

`atime` *Asymptotic timing*

Description

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

Usage

```
atime(
N, setup, expr.list, times=10, seconds.limit=0.01, verbose=FALSE,
results=TRUE, ...)
```

Arguments

`N`	numeric vector of data sizes to vary.
`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?
`results`	logical, save results?
`...`	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 `seconds.limit` (numeric input param), `timings` (data table of results).

Author(s)

Toby Dylan Hocking

Examples

```
## Example 1: polynomial vs exponential time regex.
atime.list <- atime::atime(
  PCRE=regexpr(pattern, subject, perl=TRUE),
  TRE=regexpr(pattern, subject, perl=FALSE),
  setup={
    subject <- paste(rep("a", N), collapse="")
    pattern <- paste(rep(c("a?", "a"), each=N), collapse="")
  },
  N=1:30)

if(require("ggplot2")){
  measurements <- atime.list[["measurements"]]
  sec.df <- data.frame(panel="seconds", measurements)
  mem.df <- data.frame(panel="kilobytes", measurements)
  hline.df <- with(atime.list, data.frame(seconds.limit, panel="seconds"))
  gg <- ggplot()+
    theme_bw()+
    facet_grid(panel ~ ., scales="free")+
    geom_hline(aes(
      yintercept=seconds.limit),
      color="grey",
      data=hline.df)+
    geom_ribbon(aes(
      N, ymin=min, ymax=max, fill=expr.name),
      data=sec.df,
      alpha=0.5)+
    geom_line(aes(
      N, median, color=expr.name),
      data=sec.df)+
    geom_line(aes(
      N, kilobytes, color=expr.name),
      data=mem.df)+
    scale_y_log10("")+
    scale_x_log10()
  if(require("directlabels")){
    directlabels::direct.label(gg, "last.polygons")+
      coord_cartesian(xlim=c(1,40))
  }else{
    gg
  }
}

## Example 2: split data table vs frame, constant factor difference.
library(data.table)
atime.list <- atime::atime(
  N=as.integer(10^seq(1, 7)),
  setup={
    set.seed(1)
    DT <- data.table(
      x1 = rep(c("c","d"), l=N),
```

```
      x2 = rep(c("x","y"), l=N),
      x3 = rep(c("a","b"), l=N),
      y = rnorm(N)
    )[sample(.N)]
    DF <- as.data.frame(DT)
  },
  frame=split(DF[names(DF) != "x1"], DF["x1"], drop = TRUE),
  table=split(DT, by = "x1", keep.by = FALSE, drop = TRUE)
)
best.list <- atime::references_best(atime.list)

if(require(ggplot2)){
  hline.df <- with(atime.list, data.frame(seconds.limit, unit="seconds"))
  gg <- ggplot()+
    theme_bw()+
    facet_grid(unit ~ ., scales="free")+
    geom_hline(aes(
      yintercept=seconds.limit),
      color="grey",
      data=hline.df)+
    geom_line(aes(
      N, empirical, color=expr.name),
      data=best.list$meas)+
    geom_ribbon(aes(
      N, ymin=min, ymax=max, fill=expr.name),
      data=best.list$meas[unit=="seconds"],
      alpha=0.5)+
    scale_x_log10()+
    scale_y_log10("median line, min/max band")
  if(require(directlabels)){
    gg+
      directlabels::geom_dl(aes(
        N, empirical, color=expr.name, label=expr.name),
        method="right.polygons",
        data=best.list$meas)+
      theme(legend.position="none")+
      coord_cartesian(xlim=c(1,2e7))
  }else{
    gg
  }
}
```

---

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, results=TRUE,
...)
```

Arguments

pkg.path	Path to git repo containing R package.
N	numeric vector of data sizes to vary.
setup	expression to evaluate for every data size, before timings.
expr	code with package double-colon prefix, for example Package::fun(argA, argB) which will be evaluated for each different package version.
sha.vec	named character vector / list of SHA commit IDs.
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.
results	logical, save results?
...	named SHA/commit IDs to time. Values passed as branch arg to `git2r::checkout`, names used to identify/interpret this version of the code in the output.

Details

First each version specified by ... is checked out and installed (to whatever R library is first on `.libPaths()`), using the package name Package.SHA. Then the atime function is called with arguments defined by the different SHA arguments, `atime(name1=Package.SHA1::fun(argA, argB), name2=Package.SHA2::fun(argA,argB))`.

Value

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

Author(s)

Toby Dylan Hocking

Examples

```
if(FALSE){

  tdir <- tempfile()
  dir.create(tdir)
  git2r::clone("https://github.com/tdhock/binsegRcpp", tdir)
  atime.list <- atime::atime_versions(
    pkg.path=tdir,
    N=2^seq(2, 20),
    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")
  refs.best <- atime::references_best(atime.list)
  plot(refs.best)

  atime::atime_versions_remove("binsegRcpp")

}
```

---

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 Package::fun(argA, argB) which will be evaluated for each different package version.
`sha.vec`	named character vector / list of SHA commit IDs.

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.
...	named SHA/commit IDs to time. Values passed as branch arg to git2r::checkout, names used to identify/interpret this version of the code in the output.

Details

First each version is checked out and installed (to whatever R library is first on .libPaths()), using the package name Package.SHA. Then an expression is created for each version, by replacing the PKG name in colon-prefix with PKG.SHA, atime(name1=Package.SHA1::fun(argA, argB), name2=Package.SHA2::fun(argA,argB)). For convenience, versions can be specified either as code (...) or data (sha.vec).

Value

list of expressions.

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))
    refs.best <- atime::references_best(atime.list)
    plot(refs.best)
  }
```

```
  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.

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.

Usage

```
references_best(L, unit.col.vec=NULL, more.units=NULL, fun.list=NULL)
```

Arguments

L	List output from atime.
unit.col.vec	Named character vector of units, default NULL means standard units (kilobytes and seconds).
more.units	Named character vector of units to add to unit.col.vec, default NULL means nothing.
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 references), measurements (data table of measurements).

Author(s)

Toby Dylan Hocking

Examples

```
atime.list <- atime::atime(
  PCRE=regexpr(pattern, subject, perl=TRUE),
  TRE=regexpr(pattern, subject, perl=FALSE),
  setup={
    subject <- paste(rep("a", N), collapse="")
    pattern <- paste(rep(c("a?", "a"), each=N), collapse="")
  },
  N=1:30)
best.list <- atime::references_best(atime.list)
plot(best.list)

if(require("ggplot2")){
  hline.df <- with(atime.list, data.frame(seconds.limit, unit="seconds"))
  ref.dt <- best.list$ref[overall.rank==1]
  gg <- ggplot()+
    theme_bw()+
    facet_grid(unit ~ ., scales="free")+
    geom_line(aes(
      N, reference, group=paste(expr.name, fun.name)),
      color="grey",
      data=ref.dt)+
    geom_hline(aes(
      yintercept=seconds.limit),
      color="grey",
      data=hline.df)+
    geom_ribbon(aes(
      N, ymin=min, ymax=max, fill=expr.name),
      data=best.list$meas[unit=="seconds"],
      alpha=0.5)+
    geom_line(aes(
      N, empirical, color=expr.name),
      data=best.list$meas)+
    scale_y_log10("")+
    scale_x_log10()
  if(require("directlabels")){
    gg+
      theme(legend.position="none")+
      directlabels::geom_dl(aes(
        N, empirical, color=expr.name, label=expr.name),
        data=best.list$meas,
        method="last.polygons")+
      directlabels::geom_dl(aes(
        N, reference, label=fun.name),
        data=ref.dt,
        color="grey",
        method="bottom.polygons")+
```

```
        coord_cartesian(xlim=c(1,40))
    }else{
      gg
    }
  }
```

Index