Package 'simpleCache'

August 22, 2017

Version 0.3.1

Date 2017-08-21

Title Simply Caching R Objects

Description Provides intuitive functions for caching R objects, encouraging reproducible, restartable, and distributed R analysis. The user selects a location to store caches, and then provides nothing more than a cache name and instructions (R code) for how to produce the R object. Also provides some advanced options like environment assignments, recreating or reloading caches, and cluster compute bindings (using the 'batchtools' package) making it flexible enough for use in large-scale data analysis projects.

Maintainer Nathan Sheffield <nathan@code.databio.org>

Suggests knitr, testthat

Enhances batchtools

VignetteBuilder knitr

License GPL-3

URL http://www.github.com/databio/simpleCache

BugReports http://www.github.com/databio/simpleCache

RoxygenNote 6.0.1

NeedsCompilation no

Author VP Nagraj [aut],
Nathan Sheffield [aut, cre]

Repository CRAN

Date/Publication 2017-08-22 17:07:10 UTC

R topics documented:

`addCacheSearchEnvironment` *Add a cache search environment*

Description

Append a new Environment name (a character string) to a global option which is a vector of such names. SimpleCache will search all of these environments to check if a cache is previously loaded, before reloading it.

Usage

```
addCacheSearchEnvironment(addEnv)
```

Arguments

`addEnv`	Environment to append to the shared cache search list

`deleteCaches` *Deletes caches*

Description

Given a cache name, this function will attempt to delete the cache of that name on disk.

Usage

```
deleteCaches(cacheNames, cacheDir = getOption("RCACHE.DIR"), force = FALSE)
```

Arguments

cacheNames	Name(s) of the cache to delete
cacheDir	Directory where caches are kept
force	Force deletion without user prompt

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")

# what's available?
listCaches()

# load a cache
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

---

listCaches *Show available caches.*

Description

Lists any cache files in the cache directory.

Usage

```
listCaches(cacheSubDir = "")
```

Arguments

cacheSubDir	Optional parameter to specify a subdirectory of the cache folder.

Value

character vector in which each element is the path to a file that represents an available cache (within getOption("RCACHE.DIR"))

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")

# what's available?
listCaches()

# load a cache
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

---

loadCaches *Loads pre-made caches*

Description

This function just takes a list of caches, and loads them. It's designed for stuff you already cached previously, so it won't build any caches.

Usage

```
loadCaches(cacheNames, ...)
```

Arguments

cacheNames	Vector of caches to load.
...	Additional parameters passed to simpleCache.

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
```

```
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")

# what's available?
listCaches()

# load a cache
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

`resetCacheSearchEnvironment` *Sets global option of cache search environments to NULL.*

Description

Sets global option of cache search environments to NULL.

Usage

```
resetCacheSearchEnvironment()
```

`secToTime` *This function takes a time in seconds and converts it to a more human-readable format, showing hours, minutes, or seconds, depending on how long the time is. Used by my implementation of tic()/toc().*

Description

This function takes a time in seconds and converts it to a more human-readable format, showing hours, minutes, or seconds, depending on how long the time is. Used by my implementation of tic()/toc().

Usage

```
secToTime(timeInSec)
```

Arguments

`timeInSec`	numeric value of time measured in seconds.

setCacheBuildDir *Sets local cache build directory with scripts for building files.*

Description

Sets local cache build directory with scripts for building files.

Usage

```
setCacheBuildDir(cacheBuildDir)
```

Arguments

cacheBuildDir	Directory where build scripts are stored.

setCacheDir *Sets a global variable specifying the default cache directory for simpleCache() calls.*

Description

Sets a global variable specifying the default cache directory for simpleCache() calls.

Usage

```
setCacheDir(cacheDir)
```

Arguments

cacheDir	Directory where caches should be stored

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")
```

```
# what's available?
listCaches()

# load a cache
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

`setSharedCacheDir` *Set shared cache directory*

Description

Sets global variable specifying the default cache directory for simpleCacheShared() calls; this function is simply a helper alias for caching results that will be used across projects.

Usage

```
setSharedCacheDir(sharedCacheDir)
```

Arguments

`sharedCacheDir` Directory where shared caches should be stored

`simpleCache` *Provides intuitive functions for caching R objects, encouraging faster reproducible and restartable R analysis*

Description

simpleCache provides a function (simpleCache())

Given a unique name for an R object, and instructions for how to make that object, use the simpleCache function to create and cache or load the object. This should be used for computations that take a long time and generate a table or something used repeatedly (in other scripts, for example). Because the cache is tied to the object name, there is some danger of causing troubles if you misuse the caching system. The object should be considered static.

Usage

```
simpleCache(cacheName, instruction = NULL, buildEnvir = NULL,
  reload = FALSE, recreate = FALSE, noload = FALSE,
  cacheDir = getOption("RCACHE.DIR"), cacheSubDir = NULL, timer = FALSE,
  buildDir = getOption("RBUILD.DIR"), assignToVariable = NULL,
  loadEnvir = parent.frame(), searchEnvir = getOption("SIMPLECACHE.ENV"),
  parse = NULL, nofail = FALSE, batchRegistry = NULL,
  batchResources = NULL, pepSettings = NULL, ignoreLock = FALSE)
```

Arguments

cacheName	Unique name for the cache. Be careful.
instruction	Quoted R code to be evaluated. The returned value of this code is what will be cached under the cacheName.
buildEnvir	You may choose to provide additional variables necessary for evaluating the code in instruction.
reload	forces re-loading the cache, even if it exists in the env.
recreate	forces reconstruction of the cache
noload	noload is useful for: you want to create the caches, but not load them if they aren't there (like a cache creation loop).
cacheDir	The directory where caches are saved (and loaded from). Defaults to the global RCACHE.DIR variable
cacheSubDir	You can specify a subdirectory within the cacheDir variable. Defaults to NULL.
timer	Report how long it took to create the cache?
buildDir	Location of Build files (files with instructions for use If the instructions argument is not provided). Defaults to RBUILD.DIR global option.
assignToVariable	By default, simpleCache assigns the cache to a variable named cacheName; you can overrule that here.
loadEnvir	Into which environment would you like to load the variable? Defaults to parent.frame.
searchEnvir	a vector of environments to search for the already loaded cache.
parse	By default, simpleCache will guess whether you want to parse the instruction, based on whether it is quoted. You can overwrite the guess with this parameter; but this may disappear in the future. In general, you should note quote, but use around your instructions.
nofail	By default, simpleCache throws an error if the instructions fail. Use this option to convert this error into a warning. No cache will be created, but simpleCache will not then hard-stop your processing. This is useful, for example, if you are creating a bunch of caches and it's ok if some of them do not complete.
batchRegistry	A batchtools registry object (built with batchtools::makeRegistry()). If provided, this cache will be created on the cluster using your batchtools configuration
batchResources	A list of variables to provide to batchtools for cluster resource managers. Used as the 'res' argument to batchtools::batchMap()
pepSettings	Experimental untested feature.
ignoreLock	internal parameter used for batch job submission; don't touch.

Details

You should pass a bracketed R code snippet like ' rnorm(500) ' as the instruction, and simpleCache will create the object. Alternatively, if the code to create the cache is large, you can put an R script called object.R in the RBUILD.DIR (the name of the file *must* match the name of the object it creates *exactly*). If you don't provide an instruction, the function sources RBUILD.DIR/object.R

and caches the result as the object. This source file *must* create an object with the same name of the object. If you already have an object with the name of the object to load in your current environment, this function will not try to reload the object; instead, it returns the local object. In essence, it assumes that this is a static object, which you will not change. You can force it to load the cached version instead with "reload".

Because R uses lexical scope and not dynamic scope, you may need to pass some environment variables you use in your instruction code. You can use this using the parameter buildEnvir (just provide a list of named variables).

Author(s)

Nathan Sheffield

References

https://github.com/nsheff/

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")

# what's available?
listCaches()

# load a cache
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

`simpleCacheGlobal` *Helper alias for loading caches into the global environment. simpleCache normally loads variables into the calling environment; this ensures that the variables are loaded in the global environment.*

Description

Helper alias for loading caches into the global environment. simpleCache normally loads variables into the calling environment; this ensures that the variables are loaded in the global environment.

Usage

```
simpleCacheGlobal(...)
```

Arguments

...	Parameters passed to simpleCache().

`simpleCacheShared` *Alias to default to a shared cache folder.*

Description

Helper alias for caching across experiments/people. Just sets the cacheDir to the default SHARE directory (instead of the typical default PROJECT directory)

Usage

```
simpleCacheShared(...)
```

Arguments

...	Parameters passed to simpleCache().

`simpleCacheSharedGlobal` *Helper alias for loading shared caches into the global environment.*

Description

Helper alias for loading shared caches into the global environment.

Usage

```
simpleCacheSharedGlobal(...)
```

Arguments

...	Parameters passed to simpleCache().

storeCache *Stores as a cache an already-produced R object*

Description

Sometimes you use significant computational power to create an object, but you didn't cache it with simpleCache. Oops, maybe you wish you had, after the fact. This function lets you store an object in the environment so it could be loaded by future calls to simpleCache.

Usage

```
storeCache(cacheName, cacheDir = getOption("RCACHE.DIR"),
  cacheSubDir = NULL, recreate = FALSE)
```

Arguments

cacheName	Unique name for the cache (and R object to be cached).
cacheDir	The directory where caches are saved (and loaded from). Defaults to the global RCACHE.DIR variable
cacheSubDir	You can specify a subdirectory within the cacheDir variable. Defaults to NULL.
recreate	Forces reconstruction of the cache

Details

This can be used in interactive sessions, but could also be used for another use case: you have a complicated set of instructions (too much to pass as the instruction argument to simpleCache), so you could just stick a call to storeCache at the end.

Examples

```
# choose location to store caches
cacheDir = tempdir()
cacheDir
setCacheDir(cacheDir)

# build some caches
simpleCache("normSample", { rnorm(5e3, 0,1) }, recreate=TRUE, timer=TRUE)
simpleCache("normSample", { rnorm(5e3, 0,1) })
simpleCache("normSample", { rnorm(5e3, 0,1) }, reload=TRUE)

# storing a cache after-the-fact
normSample2 = rnorm(10, 0, 1)
storeCache("normSample2")

# what's available?
listCaches()

# load a cache
```

```
simpleCache("normSample")

# load multiples caches
loadCaches(c("normSample", "normSample2"), reload=TRUE)
```

`tic` *Start a timer*

Description

Start a timer

Usage

```
tic(gcFirst = TRUE, type = c("elapsed", "user.self", "sys.self"))
```

Arguments

`gcFirst`	Garbage Collect before starting the timer?
`type`	Type of time to return, can be 'elapsed', 'user.self', or 'sys.self'

`toc` *Check the time since the current timer was started with tic()*

Description

Check the time since the current timer was started with tic()

Usage

```
toc()
```

`viewCacheDirs` *View cache directories*

Description

Views cache directory global variables

Usage

```
viewCacheDirs()
```

Index