Package 'checkpoint'

February 23, 2020

Title Install Packages from Snapshots on the Checkpoint Server for Reproducibility

Description The goal of checkpoint is to solve the problem of package reproducibility in R. Specifically, checkpoint allows you to install packages as they existed on CRAN on a specific snapshot date as if you had a CRAN time machine. To achieve reproducibility, the checkpoint() function installs the packages required or called by your project and scripts to a local library exactly as they existed at the specified point in time. Only those packages are available to your project, thereby avoiding any package updates that came later and may have altered your results. In this way, anyone using checkpoint's checkpoint() can ensure the reproducibility of your scripts or projects at any time. To create the snapshot archives, once a day (at midnight UTC) Microsoft refreshes the Austria CRAN mirror on the ``Microsoft R Archived Network'' server (<https://mran.microsoft.com/>). Immediately after completion of the rsync mirror process, the process takes a snapshot, thus creating the archive. Snapshot archives exist starting from 2014-09-17.

Version 0.4.9

License GPL-2

URL https://github.com/RevolutionAnalytics/checkpoint

BugReports https://www.github.com/RevolutionAnalytics/checkpoint/issues

Imports utils

Depends R(>= 3.0.0)

Suggests knitr, rmarkdown, testthat(>= 0.9), MASS, darts, mockery, cli

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Hong Ooi [aut, cre],
Andrie de Vries [aut],
Microsoft [aut, cph]

Maintainer Hong Ooi <hongooi@microsoft.com>

Repository CRAN

Date/Publication 2020-02-23 05:10:02 UTC

R topics documented:

`checkpoint-package` *Install packages from snapshots on the checkpoint server for reproducibility*

Description

The goal of checkpoint is to solve the problem of package reproducibility in R. Specifically, checkpoint allows you to install packages as they existed on CRAN on a specific snapshot date as if you had a CRAN time machine.

Details

To achieve reproducibility, the `checkpoint()` function installs the packages required or called by your project and scripts to a local library exactly as they existed at the specified point in time. Only those packages are available tot your project, thereby avoiding any package updates that came later and may have altered your results. In this way, anyone using the checkpoint `checkpoint()` function can ensure the reproducibility of your scripts or projects at any time.

To create the snapshot archives, once a day (at midnight UTC) we refresh the Austria CRAN mirror, on the checkpoint server (https://mran.microsoft.com/). Immediately after completion of the 'rsync" mirror process, we take a snapshot, thus creating the archive. Snapshot archives exist starting from 2014-09-17.

`checkpoint` exposes functions for:

- Creating and using snapshots:
  - `checkpoint()`: Configures R session to use packages as they existed on CRAN at time of snapshot.
  - `setSnapshot()`: Set default CRAN repository to MRAN snapshot date.
  - `getValidSnapshots()`: Read list of available snapshot dates from MRAN.
- Managing local archives:
  - `checkpointArchives()`: List checkpoint archives on disk.
  - `checkpointRemove()`: Remove checkpoint archive from disk.

  - getAccessDate(): Returns the date the snapshot was last accessed.
- Other:
  - unCheckpoint(): (Experimental) Reset the .libPath to the user library

---

checkpoint *Configures R session to use packages as they existed on CRAN at time of snapshot.*

Description

Together, the checkpoint package and the checkpoint server act as a CRAN time machine. The checkpoint() function installs the packages referenced in the specified project to a local library exactly as they existed at the specified point in time. Only those packages are available to your session, thereby avoiding any package updates that came later and may have altered your results. In this way, anyone using the checkpoint checkpoint() function can ensure the reproducibility of your scripts or projects at any time.

Usage

```
checkpoint(snapshotDate, project = getwd(), R.version,
  scanForPackages = TRUE, checkpointLocation = "~/", verbose = TRUE,
  use.knitr, auto.install.knitr = TRUE, scan.rnw.with.knitr = FALSE,
  forceInstall = FALSE, forceProject = FALSE, use.lock = TRUE)
```

Arguments

snapshotDate	Date of snapshot to use in YYYY-MM-DD format, e.g. "2014-09-17". Specify a date on or after "2014-09-17". MRAN takes one snapshot per day. To list all valid snapshot dates on MRAN use getValidSnapshots()
project	A project path. This is the path to the root of the project that references the packages to be installed from the MRAN snapshot for the date specified for snapshotDate. Defaults to current working directory using getwd().
R.version	Optional character string, e.g. "3.1.2". If specified, compares the current R.version to the specified R.version. If these differ, stops processing with an error, making no changes to the system. Specifically, if the check fails, the library path is NOT modified. This argument allows the original script author to specify a specific version of R to obtain the desired results.
scanForPackages	If TRUE, scans for packages in project folder (see details). If FALSE, skips the scanning process. A use case for scanForPackages = FALSE is to skip the scanning and installation process, e.g. in production environments with a large number of R scripts in the project. Only set scanForPackages = FALSE if you are certain that all package dependencies are already in the checkpoint folder.
checkpointLocation	File path where the checkpoint library is stored. Default is "~/", i.e. the user's home directory. A use case for changing this is to create a checkpoint library on a portable drive (e.g. USB drive).

verbose — If TRUE, displays progress messages.

use.knitr — If TRUE, parses all Rmarkdown files using the knitr package.

auto.install.knitr — If TRUE and the project contains rmarkdown files, then automatically included the packages knitr in packages to install.

scan.rnw.with.knitr — If TRUE, uses knitr::knit() to parse .Rnw files, otherwise use utils::Sweave()

forceInstall — If TRUE, forces the re-installation of all discovered packages and their dependencies. This is useful if, for some reason, the checkpoint archive becomes corrupted.

forceProject — If TRUE, forces the checkpoint process, even if the provided project folder doesn't look like an R project. A commonly reported user problem is that they accidentally trigger the checkpoint process from their home folder, resulting in scanning many R files and downloading many packages. To prevent this, we use a heuristic to determine if the project folder looks like an R project. If the project folder is the home folder, and also contains no R files, then checkpoint() asks for confirmation to continue.

use.lock — if FALSE, sets the --no-lock argument to R CMD INSTALL.

Value

Checkpoint is called for its side-effects (see the details section), but invisibly returns a list with elements:

- files_not_scanned
- pkgs_found
- pkgs_not_on_mran
- pkgs_installed

Details

checkpoint() creates a local library into which it installs a copy of the packages required by your project as they existed on CRAN on the specified snapshot date. Your R session is updated to use only these packages.

To automatically determine all packages used in your project, the function scans all R code (.R, .Rmd, and .Rpres files) for library() and require() statements. In addition, scans for occurrences of code that accesses functions in namespaces using package[::]foo() and package[:::]foo(). Finally, any occurrences of the functions methods::setClass, methods::setRefClass, methods::setMethod or methods::setGeneric will also identify the methods package as a dependency.

Specifically, the function will:

- Create a new local snapshot library to install packages. By default this library folder is at ~/.checkpoint but you can modify the path using the checkpointLocation argument.
- Update the options for your CRAN mirror and point to an MRAN snapshot using options(repos)
- Scan your project folder for all required packages and install them from the snapshot using utils::install.packages()

Resetting the checkpoint

To reset the checkpoint, simply restart your R session.

You can also use the experimental function `unCheckpoint()`

Changing the default MRAN url

By default, `checkpoint()` uses https to download packages. The default MRAN snapshot defaults to `https://mran.microsoft.com/snapshot` in R versions 3.2.0 and later, if https support is enabled.

You can modify the default URL. To change the URL, use `options(checkpoint.mranUrl = ...)`.

Log file

As a side effect, the `checkpoint` function writes a log file with information about the downloaded files, in particular the package downloaded and the associated file size in bytes. The log is stored at the root of the `checkpointLocation`. For example, if `checkpointLocation` is the user home folder (the default) then the log file is at `~/.checkpoint/checkpoint_log.csv`. This file contains columns for:

- `timestamp`
- `snapshotDate`
- `pkg`
- `bytes`

Last accessed date

The `checkpoint()` function stores a marker in the snapshot folder every time the function gets called. This marker contains the system date, thus indicating the the last time the snapshot was accessed. See also `getAccessDate()`. To remove snapshots that have not been used since a given date, use `checkpointRemove()`

See Also

Other checkpoint functions: `checkpointArchives`, `checkpointRemove`, `getAccessDate`, `getValidSnapshots`, `mranUrl`, `setSnapshot`, `unCheckpoint`

Examples

```
## Not run:

# Create temporary project and set working directory

example_project <- paste0("~/checkpoint_example_project_", Sys.Date())

dir.create(example_project, recursive = TRUE)
oldwd <- setwd(example_project)


# Write dummy code file to project
```

```
cat("library(MASS)", "library(foreach)",
    sep="\n",
    file="checkpoint_example_code.R")


# Create a checkpoint by specifying a snapshot date

library(checkpoint)
checkpoint("2014-09-17")

# Check that CRAN mirror is set to MRAN snapshot
getOption("repos")

# Check that library path is set to ~/.checkpoint
.libPaths()

# Check which packages are installed in checkpoint library
installed.packages()

# cleanup
unlink(example_project, recursive = TRUE)
setwd(oldwd)


## End(Not run)
```

---

checkpointArchives *List checkpoint archives on disk.*

Description

List checkpoint archives on disk.

Usage

```
checkpointArchives(checkpointLocation = "~/", full.names = FALSE)
```

Arguments

checkpointLocation	File path where the checkpoint library is stored. Default is "~/", i.e. the user's home directory. A use case for changing this is to create a checkpoint library on a portable drive (e.g. USB drive).
full.names	passed to list.files()

See Also

Other checkpoint functions: checkpointRemove, checkpoint, getAccessDate, getValidSnapshots, mranUrl, setSnapshot, unCheckpoint

Examples

```
checkpointArchives()
## Not run:
checkpointRemove("2016-10-01")

## End(Not run)
```

---

checkpointRemove *Remove checkpoint archive from disk.*

Description

This function enables you to delete a snapshot archive folder from disk, thus releasing storage space. If you supply a single `snapshotDate`, then only this archive will be removed. You also have the option to remove a series of snapshots, including all snapshots before a given date, or all snapshots that have not been accessed since a given date.

Usage

```
checkpointRemove(snapshotDate, checkpointLocation = "~/",
  allSinceSnapshot = FALSE, allUntilSnapshot = FALSE,
  notUsedSince = FALSE)
```

Arguments

`snapshotDate`	Date of snapshot to use in `YYYY-MM-DD` format, e.g. `"2014-09-17"`. Specify a date on or after `"2014-09-17"`. MRAN takes one snapshot per day. To list all valid snapshot dates on MRAN use `getValidSnapshots()`
`checkpointLocation`	File path where the checkpoint library is stored. Default is `"~/"`, i.e. the user's home directory. A use case for changing this is to create a checkpoint library on a portable drive (e.g. USB drive).
`allSinceSnapshot`	If `TRUE`, removes all snapshot archives since the `snapshotDate`
`allUntilSnapshot`	If `TRUE`, removes all snapshot archives before the `snapshotDate`
`notUsedSince`	If `TRUE`, removes all snapshot archives that have not been accessed since the `snapshotDate`. See `getAccessDate()`

See Also

`getAccessDate()`

Other checkpoint functions: `checkpointArchives`, `checkpoint`, `getAccessDate`, `getValidSnapshots`, `mranUrl`, `setSnapshot`, `unCheckpoint`

Examples

```
checkpointArchives()
## Not run:
checkpointRemove("2016-10-01")

## End(Not run)
```

`getAccessDate` *Returns the date the snapshot was last accessed.*

Description

The `checkpoint()` function stores a marker in the snapshot folder every time the function gets called. This marker contains the system date, thus indicating the the last time the snapshot was accessed.

Usage

```
getAccessDate(checkpointLocation = "~/")
```

Arguments

`checkpointLocation`
: File path where the checkpoint library is stored. Default is `"~/"`, i.e. the user's home directory. A use case for changing this is to create a checkpoint library on a portable drive (e.g. USB drive).

Value

Named character with last access date

See Also

`checkpointRemove()`

Other checkpoint functions: `checkpointArchives`, `checkpointRemove`, `checkpoint`, `getValidSnapshots`, `mranUrl`, `setSnapshot`, `unCheckpoint`

getValidSnapshots *Read list of available snapshot dates from MRAN.*

Description

Returns vector of available dates from MRAN or local MRAN repository.

Usage

```
getValidSnapshots(mranRootUrl = mranUrl())
```

Arguments

mranRootUrl	MRAN root. This can be a URL, e.g. `https://mran.microsoft.com/snapshot/` or the path to a local MRAN repository, e.g.`file:///local/path`

Value

Character vector with dates of valid snapshots

See Also

Other checkpoint functions: checkpointArchives, checkpointRemove, checkpoint, getAccessDate, mranUrl, setSnapshot, unCheckpoint

mranUrl *Returns MRAN URL by querying options and defaults.*

Description

This function returns the current MRAN URL. The default for this is `http(s)://mran.microsoft.com/`, and is defined by setting the `checkpoint.mranUrl` option.

Usage

```
mranUrl()
```

Value

Character string with URL

Defining a new MRAN URL

To force checkpoint() to point to a differt URL, you can set the `checkpoint.mranUrl` option.

```
options(checkpoint.mranUrl = "new_url")
```

See Also

Other checkpoint functions: checkpointArchives, checkpointRemove, checkpoint, getAccessDate, getValidSnapshots, setSnapshot, unCheckpoint

Examples

```
## Not run:

mranUrl()

# Store the existing options
old_opts <- getOption("checkpoint.mranUrl")

# Set MRAN URL to different http address
options(checkpoint.mranUrl = "https://foobah")

# Set MRAN URL to local file address
options(checkpoint.mranUrl = "file:///~")

# Reset the original options
options(checkpoint.mranUrl = old_opts)

## End(Not run)
```

---

scanForPackages *Scans a project (or folder) for references to packages.*

Description

Scans a project (or folder) for references to packages.

Usage

```
scanForPackages(project = getwd(), verbose = TRUE, use.knitr = FALSE,
  auto.install.knitr = FALSE, scan.rnw.with.knitr = FALSE)
```

Arguments

project	A project path. This is the path to the root of the project that references the packages to be installed from the MRAN snapshot for the date specified for snapshotDate. Defaults to current working directory using getwd().
verbose	If TRUE, displays progress messages.
use.knitr	If TRUE, parses all Rmarkdown files using the knitr package.
auto.install.knitr	If TRUE and the project contains rmarkdown files, then automatically included the packages knitr in packages to install.
scan.rnw.with.knitr	If TRUE, uses knitr::knit() to parse .Rnw files, otherwise use utils::Sweave()

Value

A list with two elements:

- pkgs: a character vector with the names of identified packages
- error: a character vector with information about files that could not be parsed

---

`setSnapshot` *Set default CRAN repository to MRAN snapshot date.*

Description

Set default CRAN repository to MRAN snapshot date.

Usage

```
setSnapshot(snapshotDate, online = TRUE)
```

Arguments

`snapshotDate`	Date of snapshot to use in `YYYY-MM-DD` format, e.g. `"2014-09-17"`. Specify a date on or after `"2014-09-17"`. MRAN takes one snapshot per day. To list all valid snapshot dates on MRAN use `getValidSnapshots()`
`online`	If TRUE, performs online validation checks. This can be set to FALSE for programming purposes. Internally, `checkpoint()` sets this value to FALSE when not scanning for packages.

See Also

Other checkpoint functions: `checkpointArchives`, `checkpointRemove`, `checkpoint`, `getAccessDate`, `getValidSnapshots`, `mranUrl`, `unCheckpoint`

Examples

```
# Empty date field returns current repo

oldRepos <- getOption("repos")
setSnapshot()

# Valid snapshot date
# Connects to MRAN to check for valid URL, so skip on CRAN
## Not run:
setSnapshot("2014-11-16")

## End(Not run)

# Invalid snapshot date (in future), returns error
## Not run:
setSnapshot("2100-01-01")
```

```
## End(Not run)

options(repos = oldRepos)
```

`unCheckpoint` *Undo the effect of checkpoint by resetting .libPath to user library location.*

Description

This is an experimental solution to the situation where a user no longer wants to work in the checkpointed environment. The function resets .libPaths to its pre-checkpoint value.

Note that this does not undo any of the other side-effects of `checkpoint()`. Specifically, all loaded packages remain loaded, and the value of `getOption("repos")` remains unchanged.

Usage

```
unCheckpoint(new)
```

Arguments

`new`	Not used; for back-compatibility only.

See Also

Other checkpoint functions: `checkpointArchives`, `checkpointRemove`, `checkpoint`, `getAccessDate`, `getValidSnapshots`, `mranUrl`, `setSnapshot`

Index