Package 'checkpoint'

January 28, 2022

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 1.0.2

License GPL-2

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

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

Imports utils, tools, jsonlite, yaml, withr, pkgdepends

Depends R (>= 3.3.0)

Suggests pkgcache, knitr, rmarkdown, testthat, darts

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Author Folashade Daniel [cre],
Hong Ooi [aut],
Andrie de Vries [aut],
Gábor Csárdi [ctb] (Assistance with pkgdepends),
Microsoft [aut, cph]

Maintainer Folashade Daniel <fdaniel@microsoft.com>

Repository CRAN

Date/Publication 2022-01-28 19:10:09 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 `create_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 checkpoint 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 the following functions:

- `create_checkpoint`: Creates a checkpoint by scanning a project folder and downloading and installing any packages required from MRAN.
- `use_checkpoint`: Uses a previously created checkpoint, by setting the library search path to the checkpoint path, and the CRAN mirror to MRAN.
- `delete_checkpoint`: Deletes an existing checkpoint.
- `delete_all_checkpoints`: Deletes *all* existing checkpoints.
- `uncheckpoint`: Stops using a checkpoint, restoring the library search path and CRAN mirror to their original state.
- `scan_project_files`: Scans a project for any required packages.
- `list_mran_snapshots`: Returns all valid snapshot dates found on MRAN.

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 `create_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 `use_checkpoint` function can ensure the reproducibility of your scripts or projects at any time. The `checkpoint` function serves as a simple umbrella interface to these functions. It first tests if the checkpoint exists, creates it if necessary with `create_checkpoint`, and then calls `use_checkpoint`.

Usage

```
checkpoint(
  snapshot_date,
  r_version = getRversion(),
  checkpoint_location = "~",
  ...
)

create_checkpoint(
  snapshot_date,
  r_version = getRversion(),
  checkpoint_location = "~",
  project_dir = ".",
  mran_url = getOption("checkpoint.mranUrl", "https://mran.microsoft.com"),
  scan_now = TRUE,
  scan_r_only = FALSE,
  scan_rnw_with_knitr = TRUE,
  scan_rprofile = TRUE,
  force = FALSE,
  log = TRUE,
  num_workers = getOption("Ncpus", 1),
  config = list(),
  ...
)

use_checkpoint(
  snapshot_date,
  r_version = getRversion(),
  checkpoint_location = "~",
  mran_url = getOption("checkpoint.mranUrl", "https://mran.microsoft.com"),
```

```
  prepend = FALSE,
  ...
)

delete_checkpoint(
  snapshot_date,
  r_version = getRversion(),
  checkpoint_location = "~",
  confirm = TRUE
)

delete_all_checkpoints(checkpoint_location = "~", confirm = TRUE)

uncheckpoint()
```

Arguments

snapshot_date	Date of snapshot to use in YYYY-MM-DD format, e.g. "2020-01-01". Specify a date on or after "2014-09-17". MRAN takes one snapshot per day. To list all valid snapshot dates on MRAN, use list_mran_snapshots.
r_version	Optional character string, e.g. "3.6.2". If specified, this is compared to the current R.version, and if they differ, a warning is issued. The benefit of supplying this argument is that checkpoint can alert you when your R version changes while you are working on a project; this can just as easily lead to reproducibility issues as changes in third-party code. Consider supplying an explicit value for this argument, although checkpoint will still function without it.
checkpoint_location	File path where the checkpoint library is stored. Default is "~", i.e. your home directory. Use cases for changing this include creating a checkpoint library on a portable drive (e.g. USB drive), or creating per-project checkpoints. The actual checkpoints will be created under a .checkpoint directory at this location.
...	For checkpoint, further arguments to pass to create_checkpoint and use_checkpoint. Ignored for create_checkpoint and use_checkpoint.
project_dir	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 the current working directory.
mran_url	The base MRAN URL. The default is taken from the system option checkpoint.mranUrl, or if this is unset, https://mran.microsoft.com. Currently checkpoint 1.0 does not support local MRAN mirrors.
scan_now	If TRUE, scans for packages in the project folder (see 'Details'). If FALSE, skips the scanning process. Set this to FALSE if you only want to create the checkpoint subdirectory structure.
scan_r_only	If TRUE, limits the scanning of project files to R scripts only (those with the extension ".R").
scan_rnw_with_knitr	If TRUE, scans Sweave files (those with extension ".Rnw") with knitr::knitr, otherwise with utils::Stangle. Ignored if scan_r_only=TRUE.

scan_rprofile	if TRUE, includes the ~/.Rprofile startup file in the scan. See Startup.
force	If TRUE, suppresses the confirmation prompt if create_checkpoint is run with project directory set to the user home directory.
log	If TRUE, writes logging information (mostly the output from the methods of pkgdepends::pkg_installation_proposal) to the checkpoint directory.
num_workers	The number of parallel workers to use for installing packages. Defaults to the value of the system option Ncpus, or if this is unset, 1.
config	A named list of additional configuration options to pass to pkgdepends::new_pkg_installation_propo See 'Configuration' below.
prepend	If TRUE, adds the checkpoint directory to the beginning of the library search path. The default is FALSE, where the checkpoint directory replaces all but the system entries (the values of .Library and .Library.site) in the search path; this is to reduce the chances of accidentally calling non-checkpointed code. See .libPaths.
confirm	For delete_checkpoint and delete_all_checkpoints, whether to ask for confirmation first.

Details

create_checkpoint creates a local library (by default, located under your home directory) into which it installs copies of the packages required by your project as they existed on CRAN on the specified snapshot date. To determine the packages used in your project, the function scans all R code (.R, .Rmd, .Rnw, .Rhtml and .Rpres files) for library and require statements, as well as the namespacing operators :: and :::.

create_checkpoint will automatically add the rmarkdown package as a dependency if it finds any Rmarkdown-based files (those with extension .Rmd, .Rpres or .Rhtml) in your project. This allows you to continue working with such documents after checkpointing.

Checkpoint only installs packages that can be found on CRAN. This includes third-party packages, as well as those distributed as part of R that have the "Recommends" priority. Base-priority packages (the workhorse engine of R, including utils, graphics, methods and so forth) are not checkpointed (but see the r_version argument above).

The package installation is carried out via the pkgdepends package, which has many features including cached downloads, parallel installs, and comprehensive reporting of outcomes. It also solves many problems that previous versions of checkpoint struggled with, such as being able to install packages that are in use, and reliably detecting the outcome of the installation process.

use_checkpoint modifies your R session to use only the packages installed by create_checkpoint. Specifically, it changes your library search path via .libPaths() to point to the checkpointed library, and then calls use_mran_snapshot to set the CRAN mirror for the session.

checkpoint is a convenience function that calls create_checkpoint if the checkpoint directory does not exist, and then use_checkpoint.

delete_checkpoint deletes a checkpoint, after ensuring that it is no longer in use. delete_all_checkpoints deletes *all* checkpoints under the given checkpoint location.

uncheckpoint is the reverse of use_checkpoint. It restores your library search path and CRAN mirror option to their original values, as they were before checkpoint was loaded. Call this before calling delete_checkpoint and delete_all_checkpoints.

Value

These functions are run mostly for their side-effects; however `create_checkpoint` invisibly returns an object of class `pkgdepends::pkg_installation_proposal` if `scan_now=TRUE`, and `NULL` otherwise. `checkpoint` returns the result of `create_checkpoint` if the checkpoint had to be created, otherwise `NULL`.

Configuration

The pkgdepends package which powers checkpoint allows you to customise the installation process via a list of configuration options. When creating a checkpoint, you can pass these options to pkgdepends via the `config` argument. A full list of options can be found at `pkgdepends::pkg_config`; note that `create_checkpoint` automatically sets the values of `cran-mirror`, `library` and `r-version`.

One important use case for the `config` argument is when you are using Windows or MacOS, and the snapshot date does not include binary packages for your version of R. This can occur if either your version of R is too old, or the snapshot date is too far in the past. In this case, you should specify `config=list(platforms="source")` to get checkpoint to download the *source* packages instead (and then compile them locally). Note that if your packages include C, C++ or Fortran code, you will need to have the requisite compilers installed on your machine.

Last accessed date

The `create_checkpoint` and `use_checkpoint` functions store 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.

Examples

```
## Not run:

# Create temporary project and set working directory

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

dir.create(example_project, recursive = TRUE)


# Write dummy code file to project

cat("
library(MASS)
library(foreach)
", file="checkpoint_example_code.R")


# Create a checkpoint by specifying a snapshot date
# recommended practice is to specify the R version explicitly
rver <- getRversion()
create_checkpoint("2014-09-17", r_version=rver, project_dir=example_project)
use_checkpoint("2014-09-17", r_version=rver)
```

```
# more terse alternative is checkpoint(), which is equivalent to
# calling create_checkpoint() and then use_checkpoint() in sequence
checkpoint("2014-09-17", r_version=rver, project_dir=example_project)


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


# Check that (1st) library path is set to ~/.checkpoint
.libPaths()


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


# restore initial state
uncheckpoint()


# delete the checkpoint
delete_checkpoint("2014-09-17", r_version=rver)


## End(Not run)
```

`scan_project_files` *Scan R files for package dependencies*

Description

This function scans the R files in your project, including scripts, Sweave documents and Rmarkdown-based files, for references to packages.

Usage

```
scan_project_files(
  project_dir = ".",
  scan_r_only = FALSE,
  scan_rnw_with_knitr = TRUE,
  scan_rprofile = TRUE
)
```

Arguments

`project_dir`	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 the current working directory.

`scan_r_only` If `TRUE`, limits the scanning of project files to R scripts only (those with the extension `.R`).

`scan_rnw_with_knitr` If `TRUE`, scans Sweave files with `knitr::knitr`, otherwise with `utils::Stangle`. Ignored if `scan_r_only=TRUE`.

`scan_rprofile` if `TRUE`, includes the ~/.Rprofile startup file in the scan. See `Startup`.

Details

`scan_project_files` recursively builds a list of all the R files in your project. This includes regular R scripts, as well as Sweave files (those with extension `.Rnw`) and Rmarkdown-based files (those with extension `.Rmd`, `.Rpres` or `Rhtml`). It then parses the code in each file and looks for calls to `library` and `require`, as well as the namespacing operators `::` and `:::`. The detected packages are assumed to be available from CRAN/MRAN.

Value

A list with 2 components: `pkgs`, a vector of package names, and `errors`, a vector of files that could not be scanned. The package listing includes third-party packages, as well as those that are distributed with R and have "Recommended" priority. Base-priority packages (utils, graphics, methods and so forth) are not included.

In addition, if any Rmarkdown files are found, the package listing will include rmarkdown. This allows you to continue rendering them in a checkpointed session.

Manifest

As an **experimental feature**, you can specify additional packages to include or exclude via an optional `checkpoint.yml` manifest file located in your project directory. This should be a valid YAML file with 2 components:

- `refs`: An array of package references to include in the checkpoint, that can be parsed by `pkgdepends::new_pkg_installation_proposal`.
- `exclude`: An array of package names (without decorations) to exclude from the checkpoint, despite showing up in the scan.

A manifest file allows you to include packages from sources other than CRAN/MRAN in the checkpoint. You can include a Bioconductor package with a bioc:: reference: `bioc::BiocGenerics`. A GitHub reference begins with github::, for example `github::RevolutionAnalytics/checkpoint@v1.0`. A local:: reference can point to a package `.tar.gz` file, or to a directory containing the package source code.

You should use this feature with caution, as checkpoint does not check the versions of these packages. It's recommended that you include a version number, tag or commit hash in a reference, so that you always obtain the same version of the package. See `pkgdepends::pkg_refs` for a full description of the reference syntax; note that installed:: references are *not* currently supported by checkpoint.

A use case for exclusions is if your workflow loads packages that are not on CRAN or other public repositories. For example, Microsoft Machine Learning Server (MMLS) comes with a number of proprietary packages for big data and in-database analytics. You can exclude these packages

from checkpointing by listing them in the exclude entry in the manifest. In this case, you must ensure that your packages are still visible to the checkpointed session, for example by specifying prepend=TRUE in the use_checkpoint call. If you share your project with collaborators, they will also need to have these packages separately installed on their machines.

Examples

```
scan_project_files()
```

---

use_mran_snapshot *Utilities for working with MRAN snapshots*

Description

These functions are for working with the MRAN checkpoint server. use_mran_snapshot updates the CRAN mirror for your R session to point to an MRAN snapshot, using options(repos). list_mran_snapshots returns the dates for which an MRAN snapshot exists.

Usage

```
use_mran_snapshot(
  snapshot_date,
  mran_url = getOption("checkpoint.mranUrl", "https://mran.microsoft.com"),
  validate = FALSE
)

list_mran_snapshots(
  mran_url = getOption("checkpoint.mranUrl", "https://mran.microsoft.com")
)
```

Arguments

snapshot_date	Date of snapshot to use in YYYY-MM-DD format, e.g. "2020-01-01". Specify a date on or after "2014-09-17".
mran_url	The base MRAN URL. The default is taken from the system option checkpoint.mranUrl, or if this is unset, https://mran.microsoft.com.
validate	For use_mran_snapshot, whether to check if the snapshot date is valid (exists on the server).

Value

For use_mran_snapshot, the new value of getOption("repos"), invisibly. For list_mran_snapshots, a character vector of snapshot dates.

Examples

```
## Not run:

list_mran_snapshots()

use_mran_snapshot("2020-01-01")

# validate=TRUE will detect an invalid snapshot date
use_mran_snapshot("1970-01-01", validate=TRUE)


## End(Not run)
```

Index