Package renv : présentation et retour d'expérience

Élise Maigné

Package renv : présentation et retour d'expérience

Figer ses versions de packages R

(et leurs dépendances et dépendances de dépendances)

Elise Maigné

2021-05-20

renv c'est quoi ?

Package renv

- Un package R créé par Kevin Ushey

> "new effort to bring project-local R dependency management to your projects".

- renv ~ packrat 2.0 (aussi par Kevin Ushey)

> "The goal is for renv to be a robust, stable replacement for the Packrat package, with fewer surprises and better default behaviors."

- Figer et/ou partager les versions des packages de R utilisés **pour un projet**.

Comment ça marche ?

Initialisation

Dans un dossier spécifique (la racine d'un projet) :

```
install.packages("renv")
renv::init()
```

OU

en mode projet Rstudio (.Rproj), sur un nouveau projet

File > New project > New directory > "Activate renv with this project".

(aussi dans les options du projet, onglet **Environments**)

Et voilà !

Fonctionnement

`renv::init()` crée :

- un fichier **.Rprofile**

- un fichier **renv.lock**
- un dossier **renv** contenant :
  - un fichier **.gitignore**
  - un fichier **activate.R**
  - un fichier **settings.dcf**
  - un dossier **stagging**
  - un dossier **library**

active renv au démarrage du projet/lancement de R depuis ce dossier

Fonctionnement

`renv::init()` crée :

- un fichier **.Rprofile**
- **un fichier renv.lock**
- un dossier **renv** contenant :
  - un fichier **.gitignore**
  - un fichier **activate.R**
  - un fichier **settings.dcf**
  - un dossier **stagging**
  - un dossier **library**

stocke la version de R utilisée et les versions des dépendances **dont on a besoin dans le code**

Fonctionnement - exemple renv.lock


```
{
  "R": {
    "Version": "4.0.3",
    "Repositories": [
      {
        "Name": "CRAN",
        "URL": "https://cran.rstudio.com"
      }
    ]
  },
  "Packages": {
    "BH": {
      "Package": "BH",
      "Version": "1.75.0-0",
      "Source": "Repository",
      "Repository": "CRAN",
      "Hash": "e4c04affc2cac20c8fec18385cd14691"
    },
    "R6": {
      "Package": "R6",
      "Version": "2.5.0",
      "Source": "Repository",
      "Repository": "CRAN",
      "Hash": "b203113193e70978a696b2809525649d"
    },
    ...
```

Fonctionnement

`renv::init()` crée :

- un fichier **.Rprofile**
- un fichier **renv.lock**
- un dossier **renv** contenant :
    - **un fichier .gitignore**
    - un fichier **activate.R**
    - un fichier **settings.dcf**
    - un dossier **stagging**
    - un dossier **library**

Si on travaille aussi avec git, les gros fichiers ne seront pas commités (les packages)

Fonctionnement

`renv::init()` crée :

- un fichier **.Rprofile**
- un fichier **renv.lock**
- un dossier **renv** contenant :
  - un fichier **.gitignore**
  - un fichier **activate.R**

    Script d'initialisation du renv
  - un fichier **settings.dcf**

    Les paramètres du renv pour le projet
  - un dossier **stagging**
  - un dossier **library**

Fonctionnement

`renv::init()` crée :

- un fichier **.Rprofile**
- un fichier **renv.lock**
- un dossier **renv** contenant :
  - un fichier **.gitignore**
  - un fichier **activate.R**
  - un fichier **settings.dcf**
  - un dossier **stagging**
  - un dossier **library**

Pour l'installation des "staged packages"

Pour l'installation des autres packages

Fonctionnement

Ce qui nous concerne c'est la gestion des libraries.

Quand on est dans un projet géré avec renv, on peut voir que le `.libPaths()` change :

Sans renv :

```
> .libPaths()
[1] "/home/emaigne/R/x86_64-pc-linux-gnu-library/4.0"
[2] "/opt/R/4.0.3/lib/R/library"
```

Avec renv :

```
> .libPaths()
[1] "/home/emaigne/Documents/mon_dossier/renv/library/R-4.0/x86_64-pc-linux-gnu"
[2] "/tmp/RtmpHmJq4n/renv-system-library"
```

Fonctionnement

Les packages sont installés dans un sous dossier de mon dossier de projet.

Et bien non !!

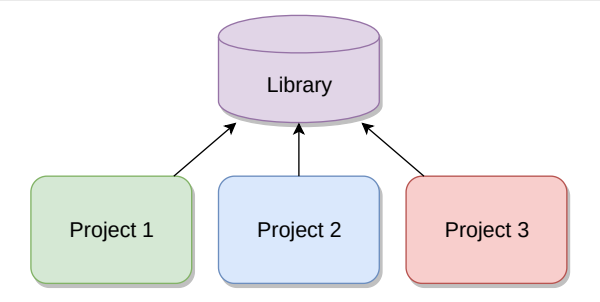
```
emaigne@camboue:~/Documents/FORMATIONS/renv_package/renv/library/R-4.0/x86_64-pc-linux-gnu$
emaigne@camboue:~/Documents/FORMATIONS/renv_package/renv/library/R-4.0/x86_64-pc-linux-gnu$ ll
total 500
drwxr-xr-x 15 emaigne miat 4096 févr.  4 15:46 ./
drwxr-xr-x  3 emaigne miat 4096 févr.  4 13:08 ../
drwxr-xr-x  7 emaigne miat 4096 févr.  4 13:58 addinexamples/
lrwxrwxrwx  1 emaigne miat  119 févr.  4 13:53 askpass -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/askpass/1.1/e8a22846fff485f0be3770c2da758713/askpass/
lrwxrwxrwx  1 emaigne miat  127 févr.  4 13:48 assertthat -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/assertthat/0.2.1/50c838a310445e954bc13f26f26a6ecf/assertthat/
lrwxrwxrwx  1 emaigne miat  125 févr.  4 13:09 base64enc -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/base64enc/0.1-3/543776ae6848fde2f48ff3816d0628bc/base64enc/
lrwxrwxrwx  1 emaigne miat  114 févr.  4 13:09 BH -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/BH/1.75.0-0/e4c04affc2cac20c8fec18385cd14691/BH/
lrwxrwxrwx  1 emaigne miat  115 févr.  4 13:53 brew -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/brew/1.0-6/92a5f887f9ae3035ac7afde22ba73ee9/brew/
lrwxrwxrwx  1 emaigne miat  115 févr.  4 13:48 brio -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/brio/1.1.1/36758510e65a457efeefa50e1e7f0576/brio/
drwxr-xr-x 14 emaigne miat 4096 févr.  4 13:58 bslib/
lrwxrwxrwx  1 emaigne miat  119 févr.  4 13:53 cachem -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/cachem/1.0.2/30f9523d900ca6d959ddddee7fc77006/cachem/
lrwxrwxrwx  1 emaigne miat  117 févr.  4 13:48 callr -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/callr/3.5.1/b7d7f1e926dfcd57c74ce93f5c048e80/callr/
```

Ce sont des liens symboliques vers un dossier de cache qui contient les différentes versions des packages.

```
brio -> /home/emaigne/.local/share/renv/cache/v5/R-4.0/x86_64-pc-linux-gnu/
        brio/1.1.1/36758510e65a457efeefa50e1e7f0576/brio/
```

Fonctionnement

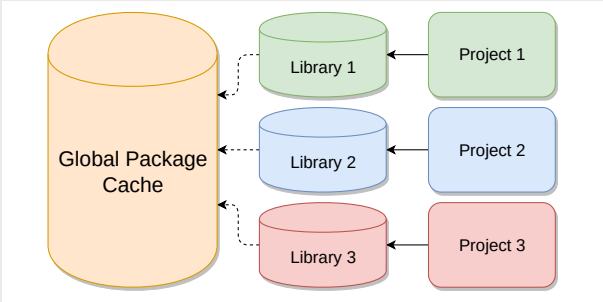
Sans renv :



© Kevin Ushley, https://kevinushey-2020-rstudio-conf.netlify.app/slides.html

Une seule version par package.

Avec renv :



© Kevin Ushley, https://kevinushey-2020-rstudio-conf.netlify.app/slides.html

Le cache permet d'avoir plusieurs versions du même package.

Le cache peut être désactivé pour un projet ► Les packages iront dans le `library` spécifique.

Schéma récapitulatif

1 dossier *library*
par projet

cache

Indépendance
des projets
sur une machine

renv.lock

Versions des packages
figées et
partagées

Comment l'utiliser ?

Commandes à connaitre

- **`renv::init()`**

- `renv::status()`
- `renv::dependencies()`
- `renv::snapshot()`
- `renv::restore()`
- `renv::install()`

Pour initialiser l'utilisation de renv dans le projet.

Commandes à connaitre

- `renv::init()`
- **`renv::status()`**
- `renv::dependencies()`
- `renv::snapshot()`
- `renv::restore()`
- `renv::install()`

Pour savoir où on en est par rapport au fichier renv.lock.

Est-ce qu'il y a des packages non présents dans renv.lock ?

Des packages dans renv.lock non utilisés dans le code R ?

Exemple :

```
renv::status()
The following package(s) are installed but not recorded in the lockfile:
  Rcpp              [1.0.6]
  whisker           [0.4]
  knitr             [1.31]

Use `renv::snapshot()` to add these packages to your lockfile.
```

Commandes à connaitre

- `renv::init()`
- `renv::status()`

- **`renv::dependencies()`**

- `renv::snapshot()`
- `renv::restore()`
- `renv::install()`

Quels sont les packages utilisés dans le code ?

Uniquement packages appelés dans les scripts R (.R ou .Rmd ou .Rnw ou DESCRIPTION) d'une des façons suivantes :

```
library(package)
require(package)
requireNamespace("package")
package::method()
```

Commandes à connaitre

- `renv::init()`
- `renv::status()`
- `renv::dependencies()`

- **`renv::snapshot()`**

- `renv::restore()`
- `renv::install()`

Pour sauver l'état des packages utilisés dans le fichier renv.lock

Exemple :

```
> renv::snapshot()
The following package(s) will be updated in the lockfile:
# CRAN ===============================
- BH                [* -> 1.75.0-0]
- R6                [* -> 2.5.0]
- Rcpp              [* -> 1.0.6]
- base64enc         [* -> 0.1-3]

Do you want to proceed? [y/N]:
```

Commandes à connaitre

- `renv::init()`
- `renv::status()`
- `renv::dependencies()`
- `renv::snapshot()`

- **`renv::restore()`**

- `renv::install()`

Pour revenir à l'état du renv.lock. i.e. désinstalle les packages en trop, réinstalle les packages manquants.

(ça ne désinstalle pas vraiment, ça supprime l'entrée dans `library`)

Commandes à connaitre

- `renv::init()`
- `renv::status()`
- `renv::dependencies()`
- `renv::snapshot()`
- `renv::restore()`

- **`renv::install()`**

Pour installer un package en utilisant renv (CRAN, version spécifique, github, commit spécifique, Bioconductor, ...).

Installation de manière intelligente.

Exemple :

```
renv::install("digest") # latest
renv::install("digest@0.6.18") # specific version
renv::install("eddelbuettel/digest") # github latest dev version
renv::install("bioc::Biobase") # (note: requires the BiocManager package)
```

Commandes à connaitre

- `renv::init()`
- `renv::status()`
- `renv::dependencies()`
- `renv::snapshot()`
- `renv::restore()`
- `renv::install()`

Pour une utilisation plus fine du package, voir :

https://kevinushey-2020-rstudio-conf.netlify.app/slides.html

Retour d'expérience

Retour d'expérience - Changement de version de R

renv ne gère pas la version de R (bien que ce soit dans renv.lock).

Sur un projet à plusieurs certains étaient en R3, d'autres en R4. En cours de route on a essayé d'uniformiser

1. tout le monde installe la version 4
2. utilisation de renv

La majorité des problèmes rencontrés venaient du fait du partage de libraries par défaut de R, une fois passé à la version 4.

Les packages étaient potentiellement déjà installés, mais compilés avec une mauvaise version de R ► renv ne savait pas les installer, installation manuelle de tous les packages et dépendences.

Retour d'expérience - Utilisation avec git

renv prend tout son sens !! On peut partager les memes versions de packages avec ses collaborateurs.

Pour partager en cours de développement (comportement par défaut) :

- le fichier renv.lock
- le fichier .Rprofile
- le dossier renv/ avec dedans:
  - .gitignore
  - settings.dcf
  - activate.R

A l'aide du .Rprofile, renv s'active tout seul chez les autres membres du projet.

Ils peuvent alors utiliser `renv::restore()` pour récupérer les bonnes versions des packages.

Partage à minima :

- le fichier renv.lock

Une autre personne peut, chez elle faire un `renv::init()` pour installer les packages correspondants.

Retour d'expérience

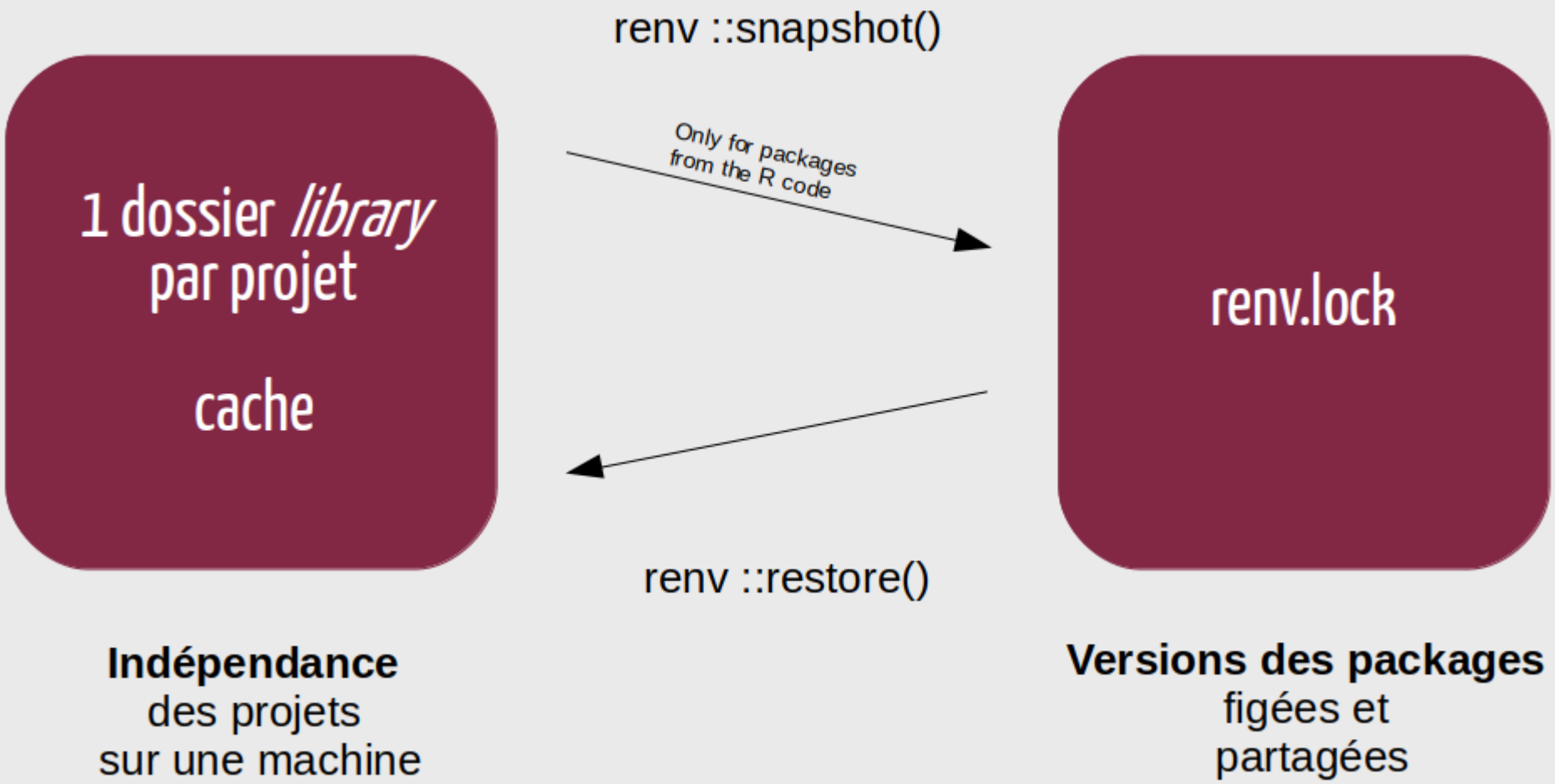
- **renv** c'est bien, notamment pour partager/archiver ses codes.
- initialement je pensais que c'était lourd (je pensais le réserver pour des projets type container) et puis j'ai découvert l'existence du cache.
- au démarrage, R nous dit si renv est utilisé (ou pas) :

```
Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

  Project '~/Documents/mon_dossier' loaded. [renv 0.12.3]
```

- `renv::status()` est très pratique, je vous conseille d'en abuser.
- Si vous n'êtes pas passé à R>=4, y passer (proprement) avant de se mettre à utiliser renv.
- renv marche même sans RStudio, et sans "projet" RStudio. C'est le .Rprofile et les fichiers de renv qui vont activer renv au niveau du dossier.

Schéma récapitulatif



des questions ?

Comparaison avec un environnement python

Tableau emprunté ici : https://6chaoran.wordpress.com/2020/07/20/introduction-of-renv-package/

task	R with renv	Python with conda	Python with pip
create the environment	renv::init()	conda create	virtualenv
save the environment	renv::snapshot()	conda env export > environment.yml	pip freeze > requirements.txt
load the environment	renv::restore()	conda env create -f environment.yml	pip install -r requirements.txt

Utilisation avec conda

Voir ici : http://russ-hyde.rbind.io/post/2021-02-23-renv-in-conda/

On peut installer r-base et r-renv avec conda puis c'est renv qui s'occupe de gérer l'environnement R, et non plus conda, ce qui évite des problèmes de non disponibilité des packages R dans conda.

Références

- Renv documentation : https://rstudio.github.io/renv/articles/renv.html
- Renv github : https://github.com/rstudio/renv/
- Présentation de Kevin Ushey : https://kevinushey-2020-rstudio-conf.netlify.app/slides.html
- http://russ-hyde.rbind.io/post/2021-02-23-renv-in-conda/
- https://6chaoran.wordpress.com/2020/07/20/introduction-of-renv-package/