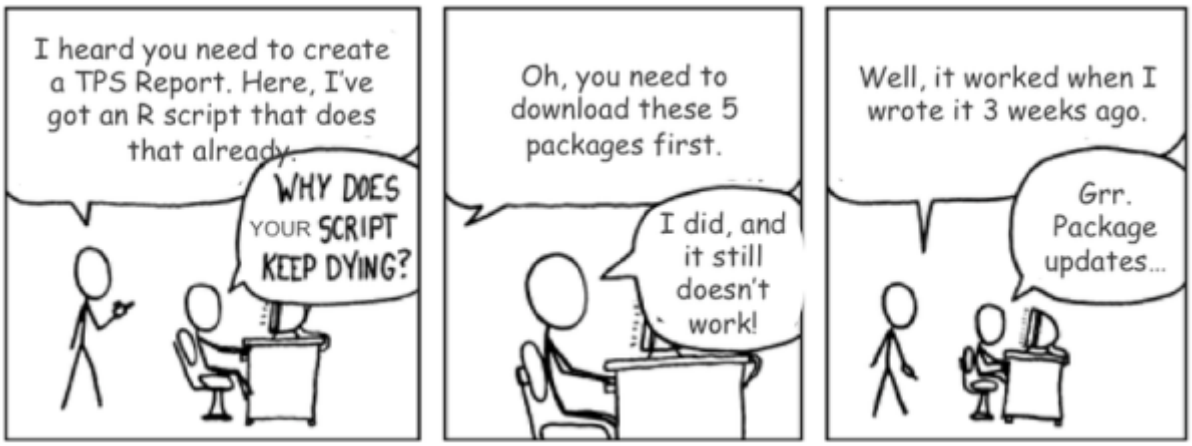
Com emprar renv (i no morir en l'intent)

Aquesta pàgina "renv" vol documentar l'ús del paquet `renv` d'R dins de projectes de desenvolupament de codi R.

> Es pot accedir a aquesta documentació a la url:
> https://seeds4c.org/renv[1]

`renv` és als paquets de R d'un projecte de RStudio el que `git` és a un projecte de codi: permet controlar quines versions estàs emprant de cada paquet/arxiu.



- Com emprar renv (i no morir en l'intent)
- 1. Introducció
  - 1.1. Inici
  - 1.2. Controlem arxius nous via git
  - 1.3. En re-obrir el projecte d'RStudio...
- 2. Afegir paquets nous
- 3. Restaurar paquets antics
- 4. Moure app a un servidor web com els shiny de CityOS
- 5. Desactivar renv si cal
- 6. Retocs a una màquina per deixar l'aplicació R operativa inicialment amb renv i git
  - 6.1. instal.lar versions de paquets de R des de repositoris
  - 6.2. Posada a punt dels entorns de treball amb renv
    - 6.2.1. Canviar a renv.lock una versió d'un paquet d'R concret
    - 6.2.2. Restaurar al projecte un paquet concret a la versió indicada a l'renv.lock

    - 6.3. Afegir una versió concreta d'un sol paquet paquet concret a renv.lock
    - 6.4. R en Windows dins de projectes amb renv actiu darrera firewall proxy o VPN
- 7. Més informació

1. Introducció

Derivat de: https://blog.rstudio.com/2019/11/06/renv-project-environments-for-r/[2]

Potser No sempre cal?

A pensar en cada cas, potser...

Instal.lar si cal

Primer de tot, cal instal.lar el paquet si no el tenim encara:

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

Obrir Projecte en RStudio

Llavors podem obrir ja un projecte de codi R en RStudio, en el que encara no haguem demanat a `renv` que controli les versions dels paquets de R que s'han d'emprar amb el projecte concret.

Cal que ens assegurem que estem treballant amb un projecte de RStudio i no pas dins l'entorn global obrint arxius a ma, per tal que funcioni tant `renv` com **git** adequadment.

Si no tenim el codi dins un projecte de RStudio, cal crear un projecte nou a partir de la carpeta de l'ordinador on tenim el codi en qüestió.

1.1. Inici

En general es comença amb alguna d'aquestes dues instruccions, per fer fa la màgia inicial i seguir els passos que s'indiquen llavors per la consola d'R

```
renv::status()
```

ó

```
renv::init() # si es vol forçar la inicialització
```

Se'ns afegirà una crida a `renv::activate()` dins de l'arxiu **.Rprofile** del projecte.

Ja podem treballar amb normalitat.

1.2. Controlem arxius nous via git

- L'arxiu `renv.lock` que s'ha creat a la carpeta arrel del projecte és un arxiu crític que hem de posar sota control de canvis amb **git**.
- Conté la llista dels paquets necessaris en el projecte de RStudio, amb les versions concretes que calen,
- Cal fer un `commit` de git (com a mínim) i un `push` contra el gitlab.ajuntament.bcn (o repositori de projectes amb git que estiguem emprant en cada moment), per més seguretat.
- Idem podem fer amb la carpeta **./renv/**sencera
    - Aquesta carpeta `./renv/` ja té un arxiu **.gitignore** a dins, que fa que només s'afegeixin a control via git els arxius de configuració petits, però en cap cas els arxius dels paquets de R, per que ja estan als repositoris dels quals provenen.

1.3. En re-obrir el projecte d'RStudio...

En obrir de nou quan calgui el projecte d'RStudio, hem de veure que a la consola una de les darreres línies que es mostren indica que s'ha carregat el projecte de `renv`: "Project '~/code/454_foo' loaded. [renv 0.13.0]"

Exemple de la consola d'R en obrir el projecte fictici "454_foo" dins la carpeta **code** de l'usuari:

```
R version 3.6.3 (2020-02-29) -- "Holding the Windsock"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
(...)
Type 'q()' to quit R.

Loading required package: drat
* Project '~/code/454_foo' loaded. [renv 0.13.0]
[Workspace loaded from ~/code/454_foo/.RData]

>
```

Ja podem treballar amb normalitat.

Cal tenir present que els paquets que instal.lem a partir de llavors, seran instal.lats dins la subcarpeta **./renv** del nostre projecte d'R.

2. Afegir paquets nous

Primer afegim el paquet `devtools` sempre, per tal de disposar de la funció **install_version()** que hi ha dins del projecte amb `renv` nou:

```
install.packages("devtools")
```

Després ja podem afegim el paquet nou:

```
install.packages("paquetnou") # Així per a que se'ns instal.li la darrera versió disponible en el repositori, o bé
renv::install("paquetnou@1.0.1")  # :: D'aquesta altra forma per a que s'instal.li una versió concreta
```

Llavors ja podem fer una "foto" (de les noves versions que volem que `renv` controli), amb:

```
renv::snapshot()
```

Si hi ha algun paquet que ja no és necessari, sencillament ja no el cridem en el codi amb `library()`, i el propi `snapshot()` ens indicarà que hi ha aquell/s paquet/s que ja no són necessaris i

3. Restaurar paquets antics

Ho podem fer amb:

```
renv::restore()
```

4. Moure app a un servidor web com els shiny de CityOS

Cal cridar a aquesta instrucció `renv::isolate()` en la sessió d'R del projecte de RStudio que estem desenvolupant, segons el propi desenvolupador principal de Shiny[3]:

```
> renv::isolate()
* Copying packages into the private library ... [103/103] Done!
* This project has been isolated from the cache.
>
```

I llavors ja es pot moure tot el codi de l'app shiny (o equivalent) a la ruta de les apps shiny (o equivalent) del servidor.

Més informació:

[+]

> Cal tenir en compte que, en el cas d'emprar "**RStudio connect**" per posar l'app a un servidor com shinyapps.io o equivalent, cal no posar l'arxiu `.Rprofile` de dins del projecte amb renv al servidor web, segons la documentació d'aquell cas d'ús concret amb Rstudio connect: https://rstudio.github.io/renv/articles/rsconnect.html[5]

5. Desactivar renv si cal

Podria passar (com hem vist amb l'app d'historic d'eleccions) que una aplicació temporalment no funciona bé (per alguna dependència extranya no resolta) quan volem emprar `renv` per controlar les versions dels paquets.
Si ho necessitem, podem desactivar `renv` temporalment en aquest projecte de RStudio amb:

```
renv::deactivate()
# renv::activate() # O això si el volem tornar a activar en un altre moment
```

6. Retocs a una màquina per deixar l'aplicació R operativa inicialment amb renv i git

Instal.lar paquets de sistema a ma

Per exemple, en CityOS Producció, vam haver d'instalar a mà els següents paquets de sistema: (per a centos)

```
sudo yum install libgit2-devel
sudo yum install libssh2 # libssh2 de sistema ha de ser la versió 1.70 o superior, i per omissió a PRE i PRO hi havia la versió 1.4.x. Actualitzem la llibreria libssh2 des dels repositoris per omissió de CentOS
```

6.1. instal.lar versions de paquets de R des de repositoris

Quan la versió més recent ja va bé:

```
install.packages("later")
```

Quan cal insta.lar les versions antigues (les que s'indiquen a `renv.lock`):

```
devtools::install_version("XML", version="3.98-1.20")
devtools::install_version("odbc", version="1.2.1")
devtools::install_version("pkgload", version="1.0.2") # A PRO havia fallat la instal.lació del pkgload, i hem hagut d'instal.lar la 1.0.2
# si falla per que no hi ha devtools instal.lat, es pot forçar a
install.packages("pkgload", version="1.0.2")
```

6.2. Posada a punt dels entorns de treball amb renv

En PRO, hem hagut de fer un `renv::restore()` per a que les versions dels paquets de R que empra l'aplicació coincideixin amb les versions amb les que es va desenvolupar l'aplicació (en molts casos són versions anteriors a les que hi havia instal.lades més noves de repositoris de R per omissió per a R 3.6.x)

6.2.1. Canviar a renv.lock una versió d'un paquet d'R concret

Un cop confirmat que ja et va el projecte, amb el paquet d'R concret que sigui en una versió diferent del que s'especificava a l' `renv.lock` , pots indicar que vols que es desi la nova versió del paquet d'R concret a dins de l' `renv.lock` del projecte. Per exemple, si volguessis pujar la versió del paquet ggplot2 a la versió 1.1.0 (numero de versió ficticia només amb fins de documentació ara), ho pots fer amb:

```
renv::record("ggplot2@1.1.0")
```

6.2.2. Restaurar al projecte un paquet concret a la versió indicada a l'renv.lock

Per restaurar un paquet concret només (per exemple, ggplot2) a la versió que hi està indicada al `renv.lock`, es pot fer amb una instrucció de l'estil:

```
renv::restore(packages = "ggplot2")
```

6.3. Afegir una versió concreta d'un sol paquet paquet concret a renv.lock

Ho pots fer amb comandes de l'estil:

```
renv::record("renv@0.13.1")
```

6.4. R en Windows dins de projectes amb renv actiu darrera firewall proxy o VPN

Sembla que en les màquines coporatives amb Windows 10 a l'oficina en que es treballa darrera un firewall (i.e. sense emprar Global Protect), o quan es treballa des de casa amb VPN (i firewall + proxy, etc), quan s'activa el paquet `renv` (per a la gestió fàcil de paquets de R dins de projectes de desenvolupament de codi/peticions), l'RStudio local en windows no es capaç de sortir a la xarxa correctament a descarregar i instal.lar paquets de R. En canvi, fora del projecte amb renv si que funciona (mateix R i mateix RStudio).

La sol.lució trobada en el context de teletreball i GlobalProtect (posant la informació del proxy a un arxiu `_curlrc`) no sembla funcionar a les màquines conectades per cable ethernet a les oficines corporatives.

Cal instal.lar primer unes eines genèriques a l'R (fora dels projectes amb renv):

```
renv::equip() # això t'hauria d'instal.lar curl entre altres paquets d'R i
llibreries externes
# install.packages("curl", type="binary") # si l'anterior falla
```

Després, encara s'ha d'actualitzar la versió del paquet **renv** que hi ha a dins del projecte d'R.

El problema ve associat amb que renv empar per omissió una forma diferent de sortir a internet que la forma que empra l'R. I la solució està en emprar una versió de `renv` prou nova (com a mínim la versió **renv v0.13.1**, de 2021-03-18) per a que tingui el mètode per canviar la forma de sortir a internet per un altre. Aquest és `renv_download_method()` , i la resposta si tenim la versió 0.13.1 de renv o més nova, serà **libcurl** per omissió:

```
> renv:::renv_download_method()
[1] "libcurl"
>
```

I en canvi, el mètode que empra R és un altre, que pots mirar amb `getOption("download.file.method")` :

```
> getOption("download.file.method")
[1] "wininet"
```

```
>
```

Tingues en compte que, si tenim alguna versió anterior a renv 0.13.1, ens respondrà que no coneix aquest objecte:

```
> renv:::renv_download_method()
Error in get(name, envir = asNamespace(pkg), inherits = FALSE) :
  object 'renv_download_method' not found
>
```

En aquest darrer cas, si dins d'un projecte amb " `renv` " s'està emprant una versió més antiga (per exemple. la v 0.9.2 com al projecte d'historic d'eleccions, o els d'informes d'eleccions), cal instal.lar l'renv més nou possible amb un `install.packages("renv")` a seques, des de fora del projecte de RStudio (i per tant sense emprar renv antic).

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

Llavors podràs instal.lar aquesta versió més nova (**0.15.2** o més nova, en funció de quan ho executis) des de dins del propi projecte de RStudio.

```
renv::install("renv@0.15.2")
```

Reiniciar ara sí el projecte de RStudio, per a que entri en funcionament aquesta nova versió de de `renv`, i llavors executar la instrucció per canviar la forma de sortir a internet, per tal de poder finalment instal.lar des del propi Renv més nou, la versió anterior de renv que volguem emprar, que com a mínim serà la versió 0.13.1:

```
renv::install("renv@0.13.1")
Sys.setenv(RENV_DOWNLOAD_FILE_METHOD = getOption("download.file.method"))
```

Ara ja pots revisar si pots restaurar sense problemes els paquets de R amb les versions indicades a renv.lock, que t'hauria de funcionar:

```
renv::restore()
```

Un cop confirmat que ara ja et va, quan acabis de restaurar el projecte, pots indicar que vols que l'renv.lock del projecte posi la nova versió del paquet renv mateix a dins. Ho pots fer amb:

```
renv::record("renv@0.13.1")
```

Per restaurar un paquet concret només, a la versió que hi està indicada al `renv.lock`, es pot fer amb una instrucció de l'estil:

```
renv::restore(packages = "renv")
```

I quan migrem un codi antic a un nou servidor amb versions més noves de R i paquets de sistema, poden haver-hi alguns paquets de R de la recepta a renv.lock que calgui actualitzar i tornar a desar la versió més nova a la recepta renv.lock que estem actualitzant.
Sovint ens trobem en aquestes situacions en intentar restaurar una recepta renv.lock antiga en un servidor nou ans després.
En aquests casos, aquest `1-liner` sol ajudar:

```
myp <- "XXX"; renv::install(myp, prompt=F); renv::record(myp);
renv::restore(prompt=F)
```

7. Més informació

Veure:

- 230116 Com emprar renv i no morir en l intent (xerrada llampec a l'ACA per a analistes de dades amb R a la Generalitat de Catalunya)
- https://rstudio.github.io/renv/articles/renv.html[6] (tutorial extès pels authors del paquet `renv`)
- https://6chaoran.wordpress.com/2020/07/20/introduction-of-renv-package/[7] (comparació amb python, captures de pantalla diverses i notes sobre docker)

[1] https://seeds4c.org/renv

[2] https://blog.rstudio.com/2019/11/06/renv-project-environments-for-r/

[3] https://community.rstudio.com/t/shiny-server-renv/71879/4

[4] https://community.rstudio.com/t/shiny-server-renv/71879/4

[5] https://rstudio.github.io/renv/articles/rsconnect.html

[6] https://rstudio.github.io/renv/articles/renv.html

[7] https://6chaoran.wordpress.com/2020/07/20/introduction-of-renv-package/