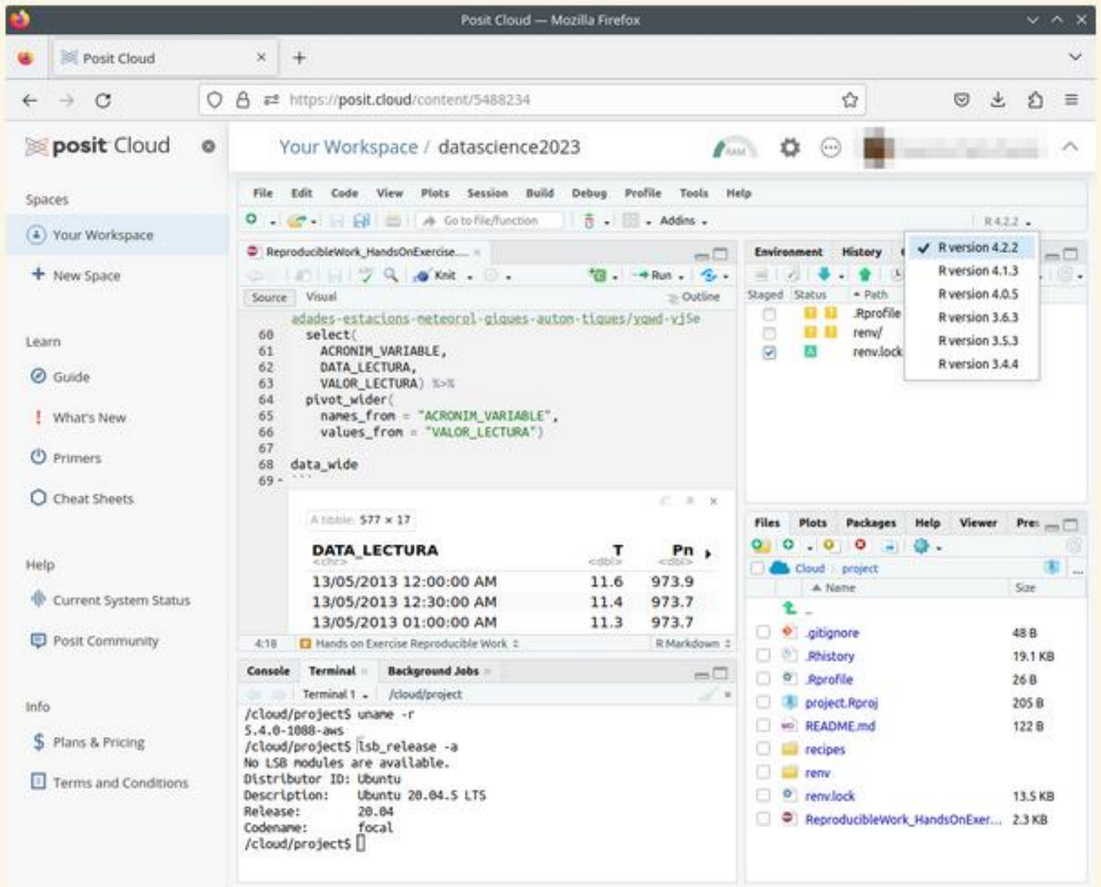
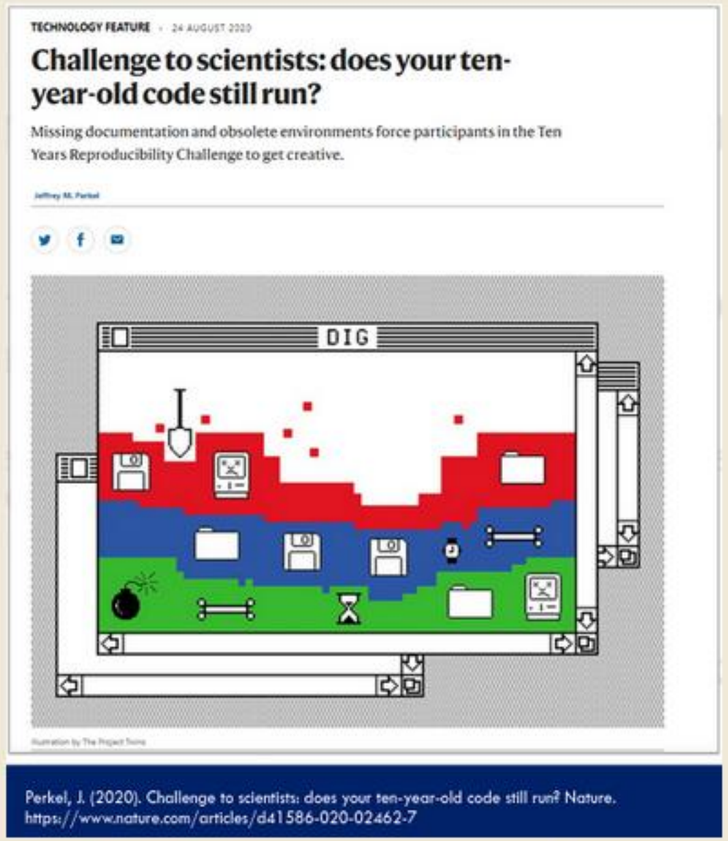
2023 Reproducible work in Data Science (X. de Pedro)

"Data Science. Applications to Biology and Medicine with Python and R", at IL3 - University of Barcelona. Feb 27th, 2023 (16-19:15h).

Content at https://seeds4c.org/reproduciblework2023



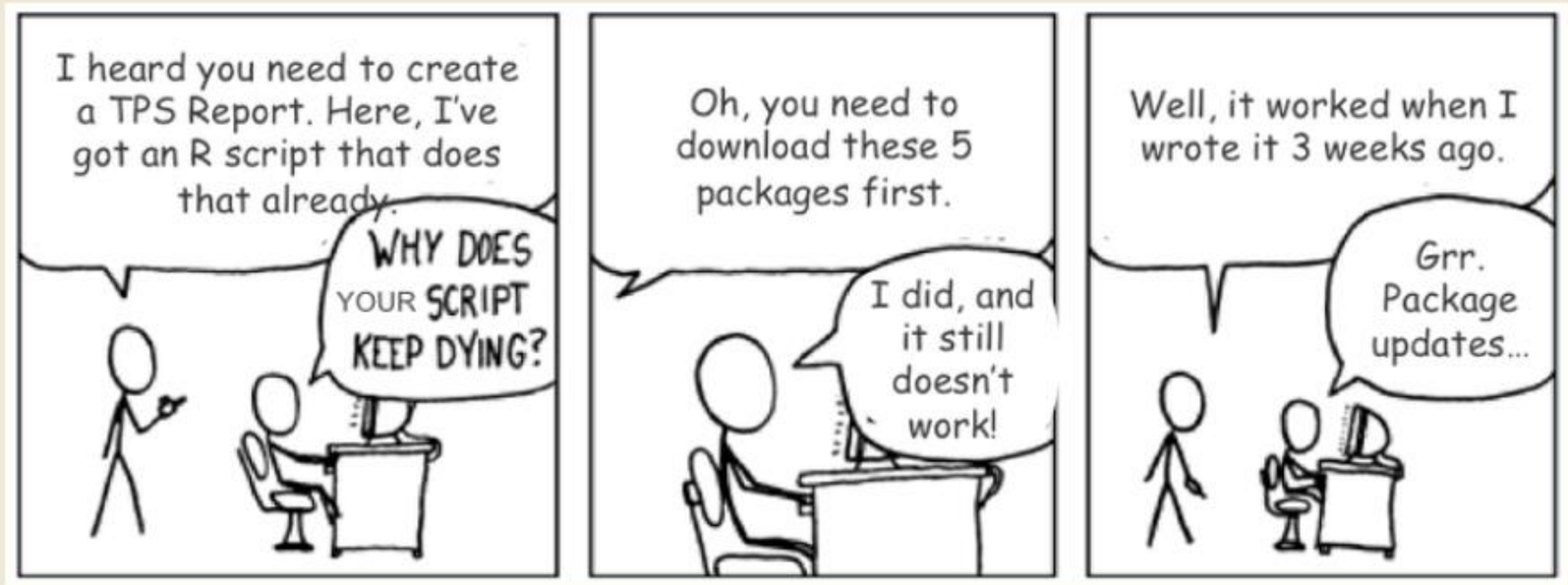
1. Introduction - the problems (i)



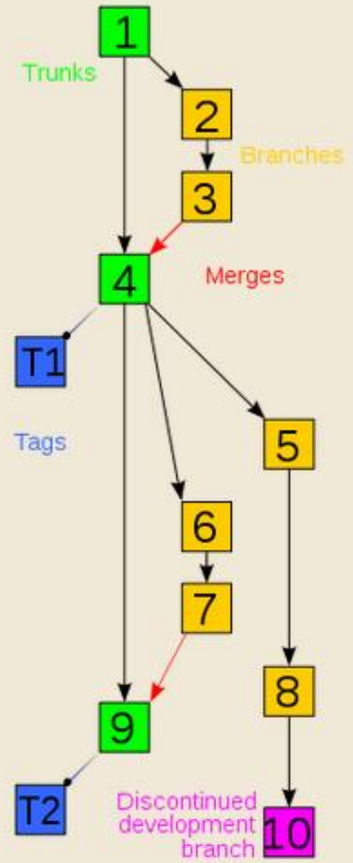
-->

Obsolete Devices storing code & data --> Ease copying to new devices (legally also: copyleft, ...) + online repositories

1.1. The problems (ii)



-->



Software obsolescence and incompatible dependency versions

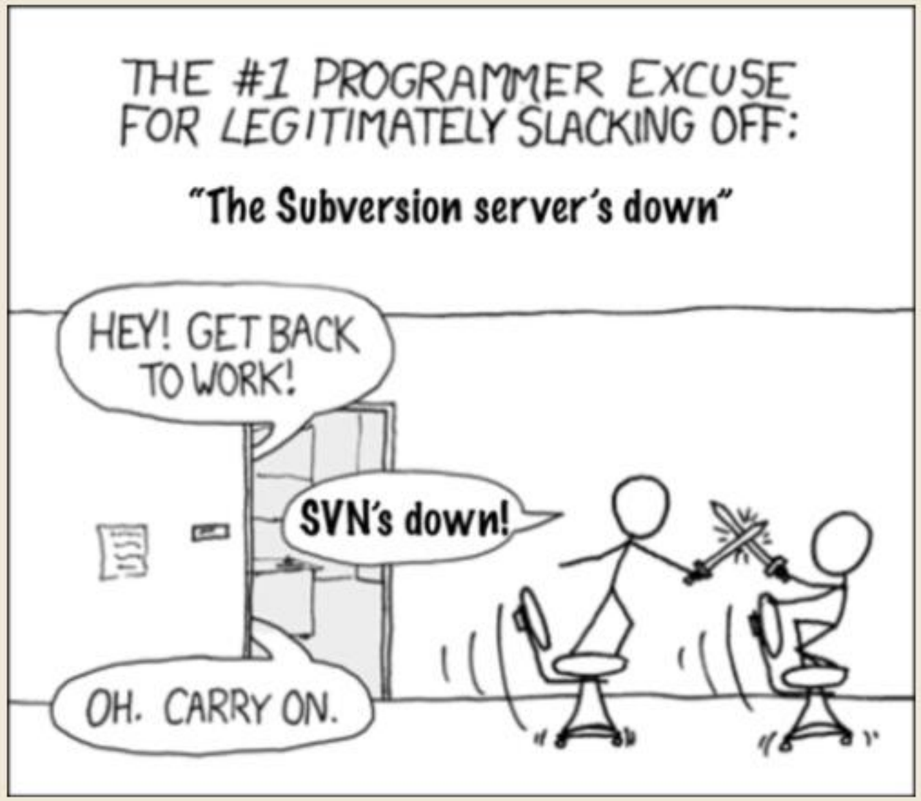
-->

Adapt to code evolution:

- Controlling Package Versions (`renv`)
- VCS (**git**, bazaar, svn...)

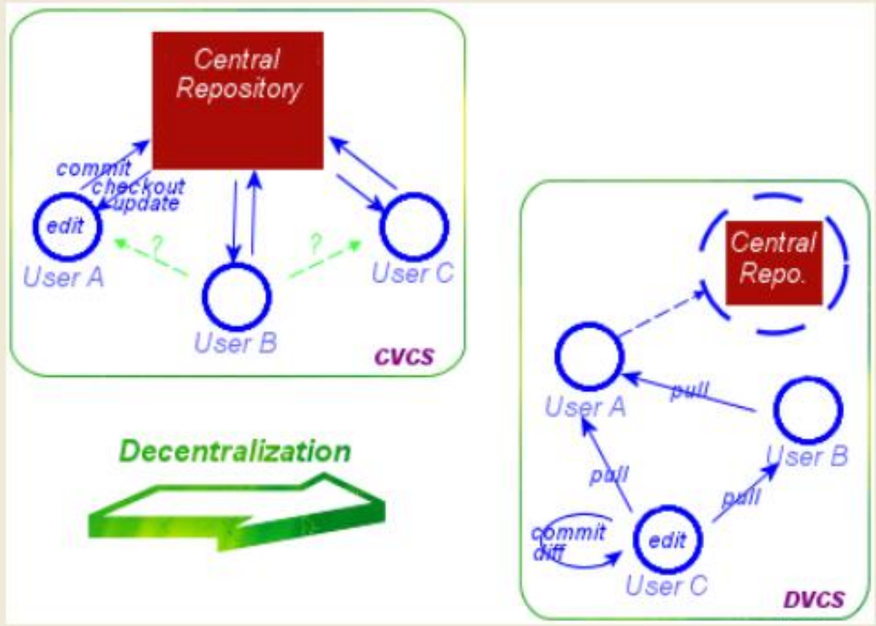
VCS = Version Control Systems

1.2. The problems (iii)



Centralization (such as **Subversion** VCS (**svn**) may increase efficiency but it also **decreases Resilience** ("shit happens")

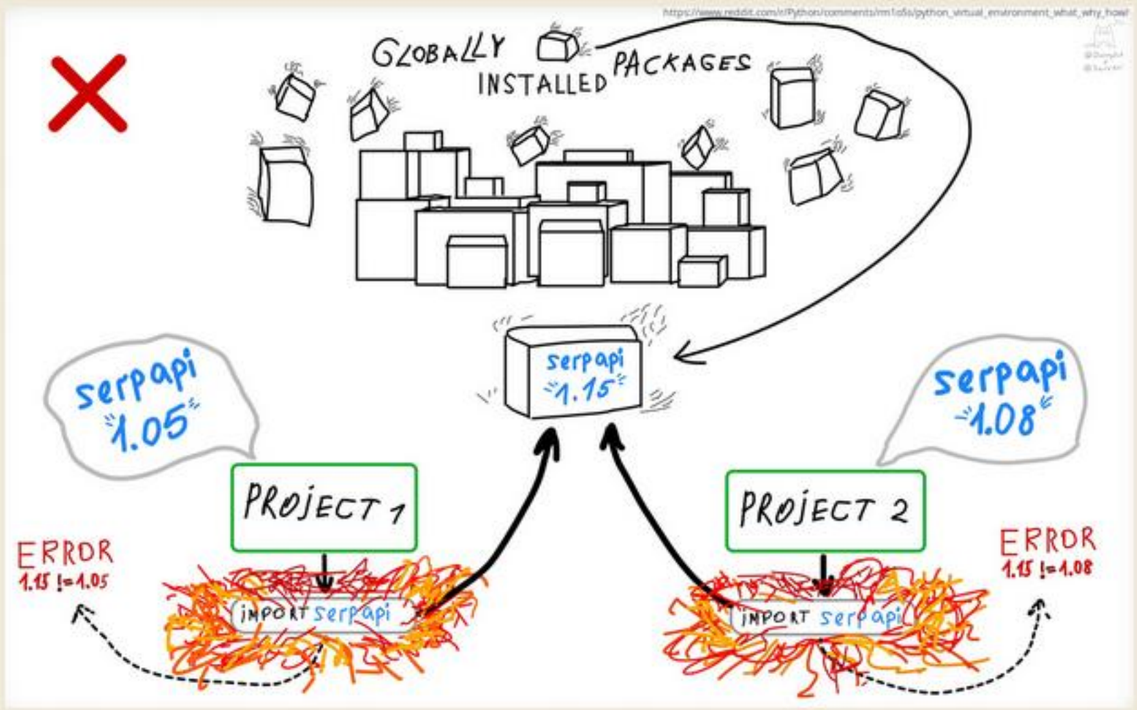
-->



--> From *Centralized* VCS (such as **svn**) to *Decentralized* VCS (such as **git**)

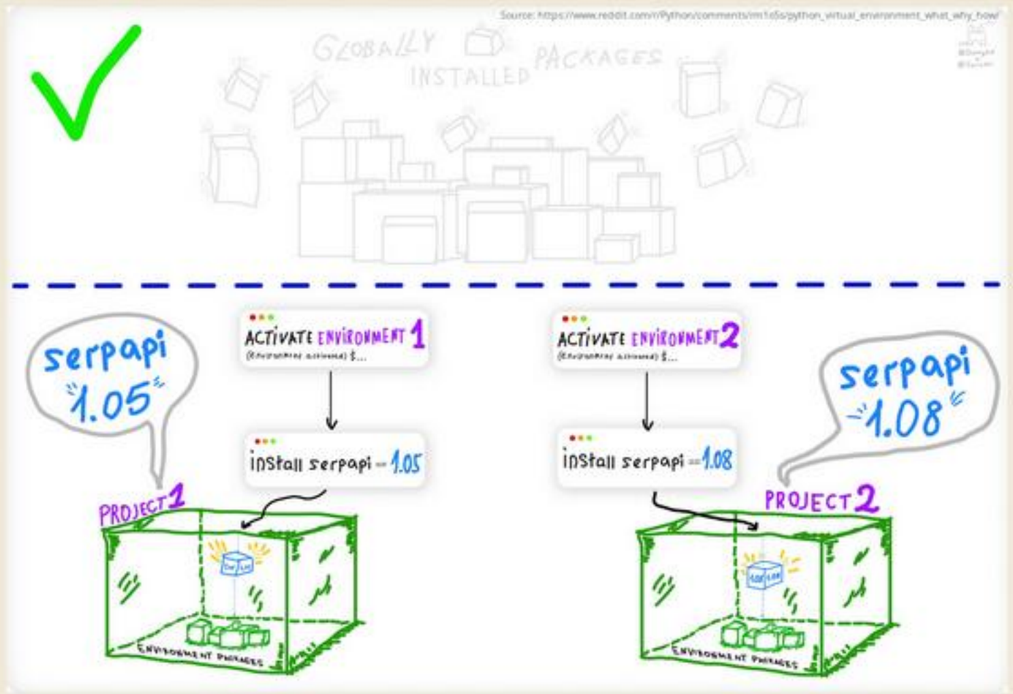
VCS = Version Control Systems

1.3. The problems (iv)



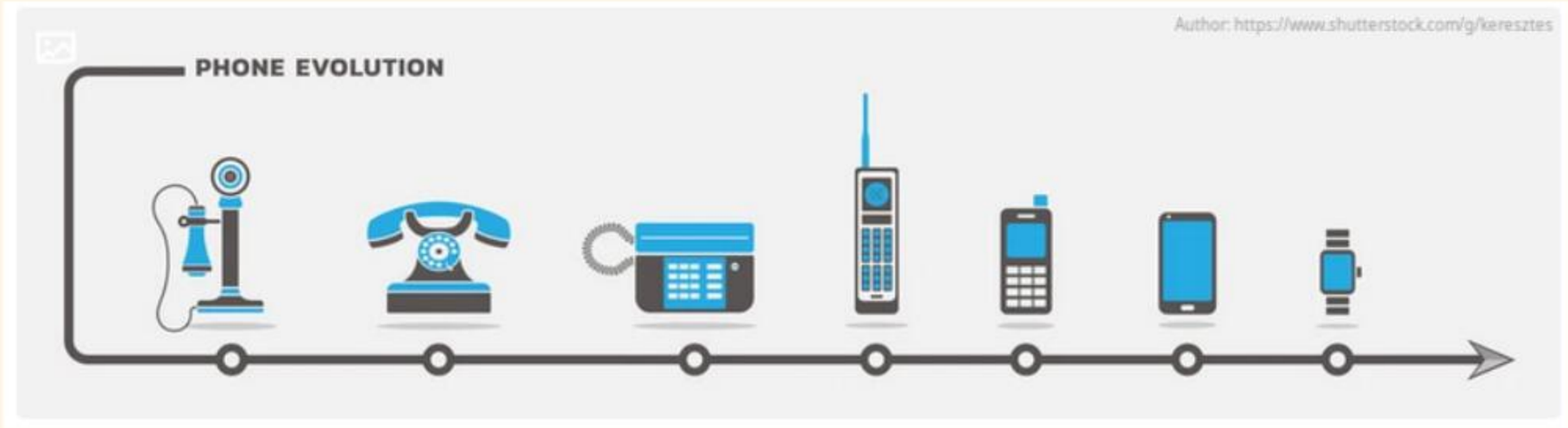
Conflicting package versions at system level with package versions at project levels

-->



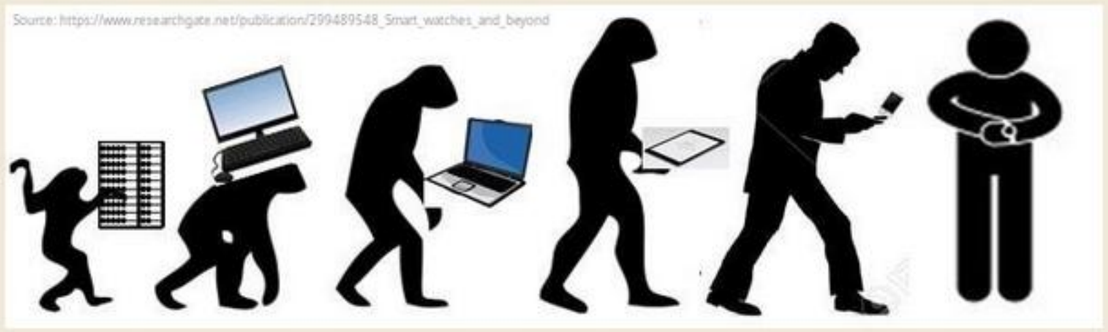
Package versions per project Environment (`renv`)

1.4. The problem (v)

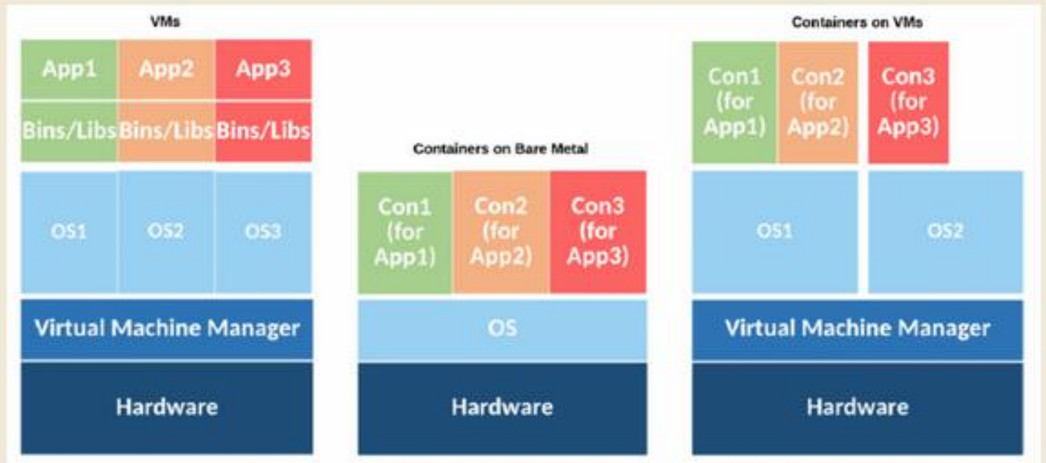


Sometimes a project was developed with a major version of a programing language (R 3.x, Python 2.x), while another project in the same server requires a different major version (R 4.x, Python 3.x)	-->	**R case**: from RStudio Server to Posit Workbench (former *RStudio Server Pro*) You can choose R version per project **Python**: Several approaches (conda, PyCharm, ...): see this as an example.

1.5. The problem (vi)



-->



Changes per Operating System itself (32 bit systems unsuported anymore, discontinued linux distros, ...)	-->	**Virtual Machines or Containers** (OVA, KVM, LXC, Docker, Pod...): you can choose OS version per container within the same server

2. Enemies of reproducibility & adaptability

Enemies of reproducibility and adaptability (in levels): Changes / Evolution / Versions!

1. **Operating system** and its **dependencies** (and their versions)
2. **Programming language** (and its version)
3. **Specific Packages** (and versions) as dependencies for your Work Project
4. **Versions** of your **own code** (algorythm and param variations, etc): lacking versioning system
5. **Readability and tidyness** of your own code / routines / scripts
6. Lack of **documentation/help resources** + steep learning curve to use it or adapt it to your context or infrastructure

3. Reproducibility & Adaptability

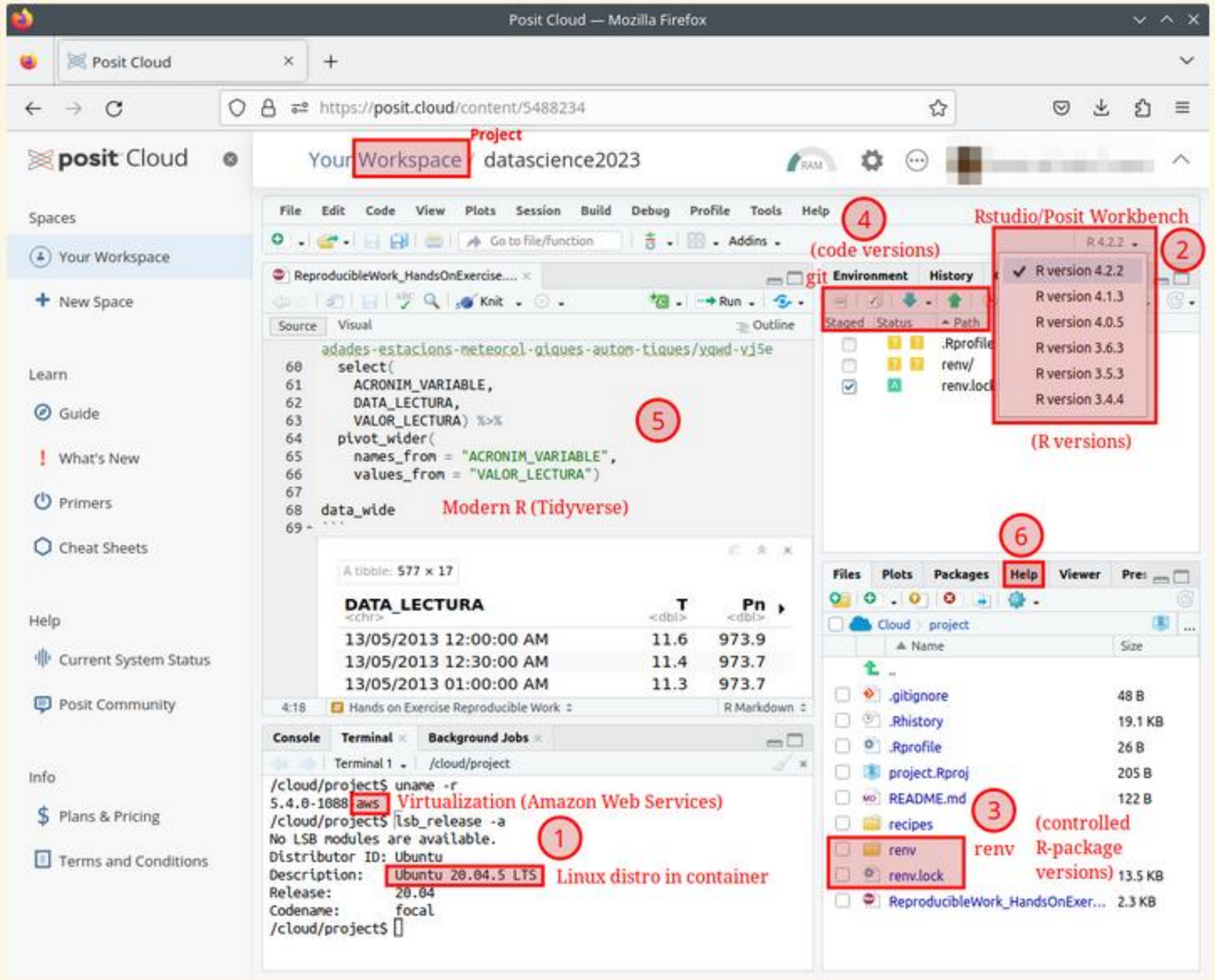
How to avoid reproducibility & adaptability enemies (in R & Python for Data Science):

ISSUES	SOLUTIONS / WORKAROUNDS
(Level 1) **Versions in OS repos & critical dependencies**: curl, ssl, GDAL, Java, cpp, V8...	Virtual Machines or Containers (VBox, KVM, LXC, Docker, Pod...)
(Level 2) **Versions in Programming language**: Python 2.x vs 3.x, R 3.x vs 4.x, ...	Python: Conda, ~~Google Colab~~, ... R: RStudio/Posit Workbench General (in Linux clusters): *software modules*.
(Level 3) **Versions in Specific packages**	=== Py: .env, poetry R: ~~Packrat~~, Renv (by versions), MRAN (by date)
(Level 4) **Versions in Your own scripts**	Decentralized VCS: Git (Gitlab, Github, ...), Bazaar (Launchpad), ... Centralized VCS: CVS, SVN (Sourceforge, ...), ... *VCS = Version Control system*
(Level 5) **Tidy script content and organization**	Literate Coding (Scripting & Coding) / Analysis - **R**: **Rstudio Notebooks** with modern R (*Tidyverse*). VS Notebooks, G-Colab, ... - **Python**: **Jupyter Notebooks**, Rstudio Notebooks, VS Notebooks, G-Colab, ... (`Quarto` Mardown and rendering for both and more)
(Level 6) **Help** to lower the learning curve	Documentation, Code Vignettes, Examples, Tutorials, Learning material (`learnr`), Books (`bookdown`)...

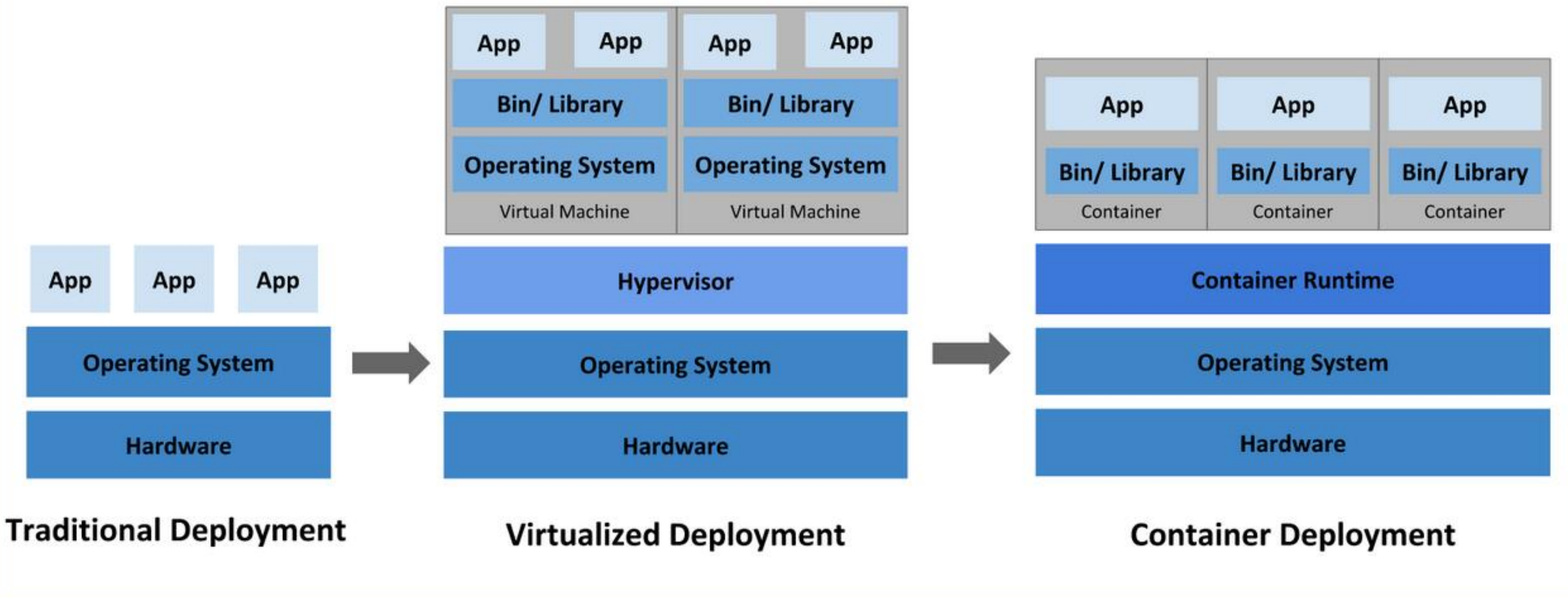
4. Reproducibility & Adaptability - Example in Posit Cloud

Example in https://posit.cloud (former *RStudio Server Pro*) :

- Level 1: A **Container** with a specific linux distro (e.g. Ubuntu Linux 20.04 Focal LTS) per project.
- Level 2: RStudio/**Posit Workbench** (which allows choosing R version per project)
- Level 3: `renv` for your R package collection (and specific versions) in your project
- Level 4: **git** or svn for your scripts in your project
- Level 5: YOU (*Tidyverse* is your friend)
- Level 6: YOU (+ helpers: `roxygen2`, `blogdown`, `learnr`, `bookdown`, ...)

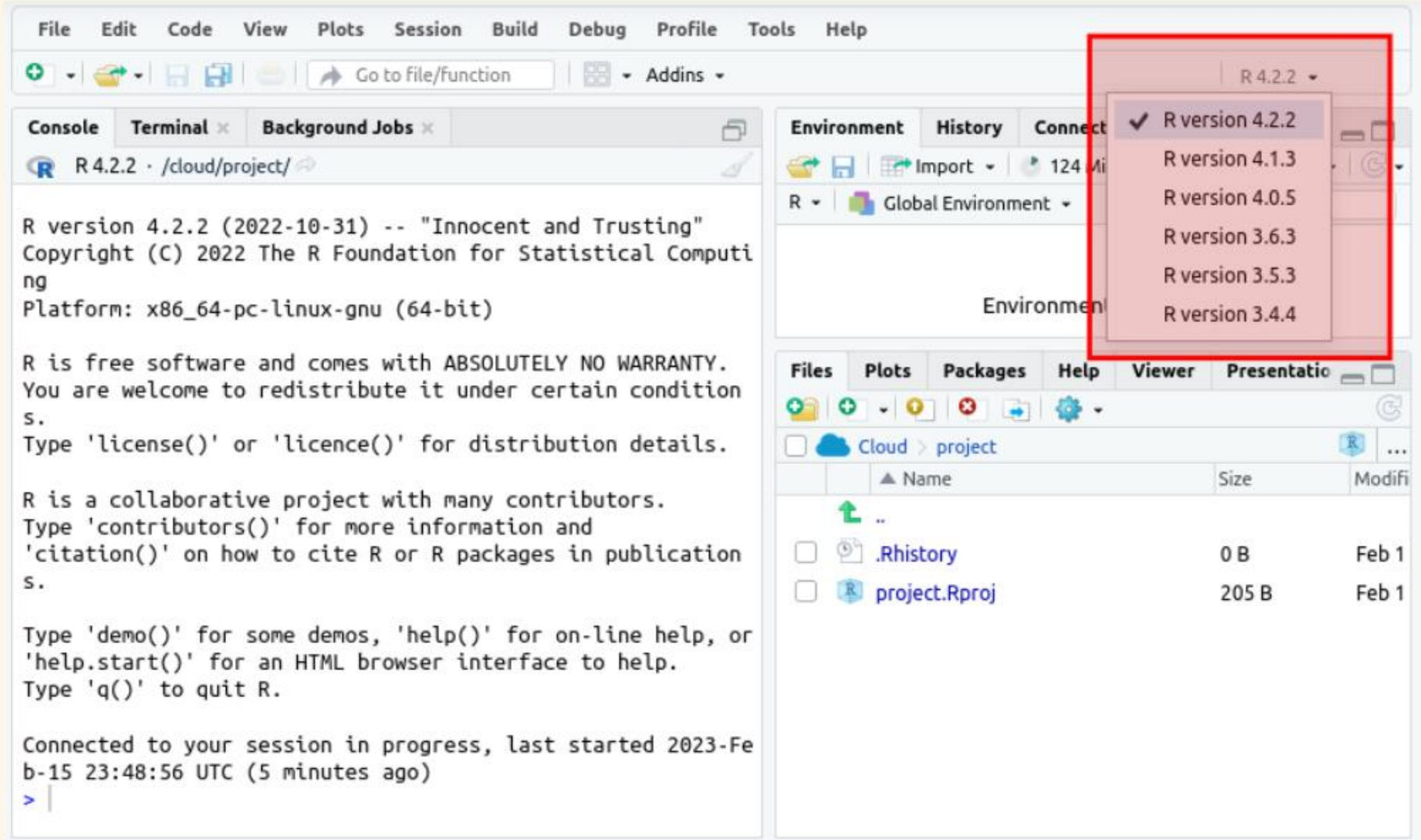


4.1. Level 1: Virtual Machines or Containers



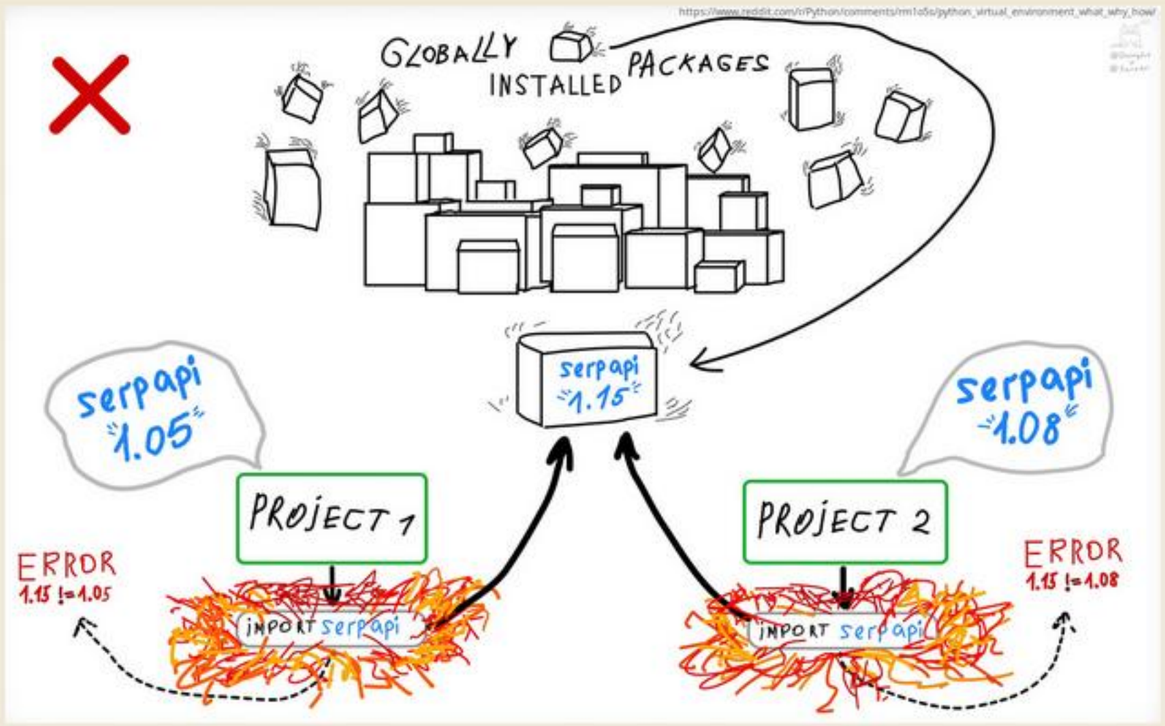
From:
https://kubernetes.io/docs/concepts/overview/

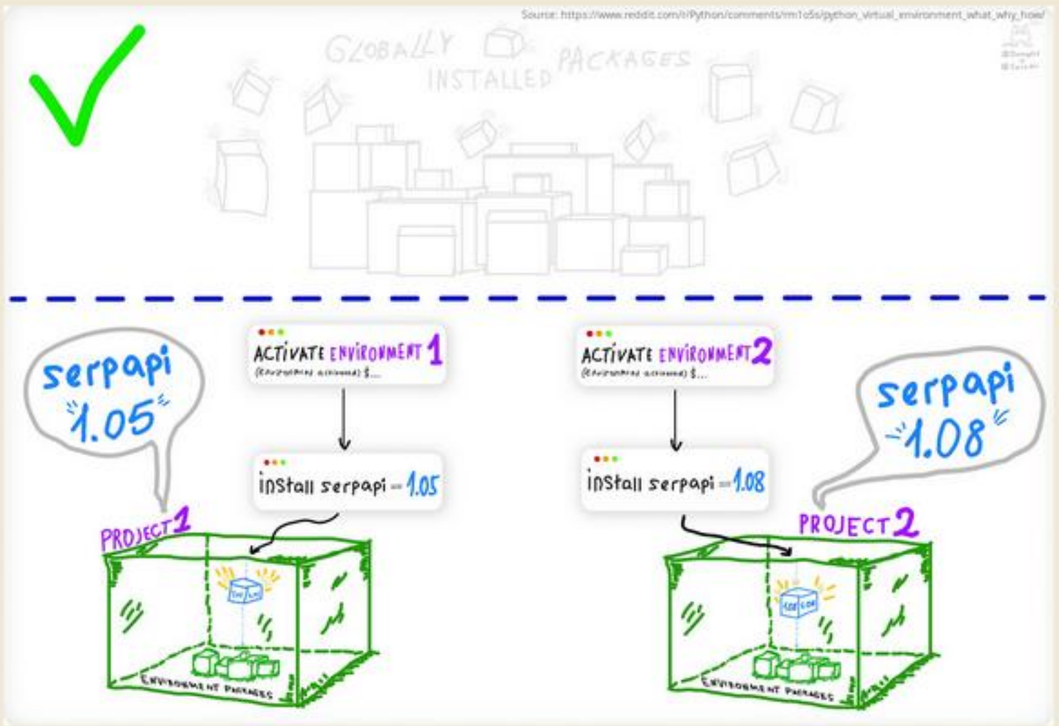
4.2. Level 2: RStudio-Posit Workbench



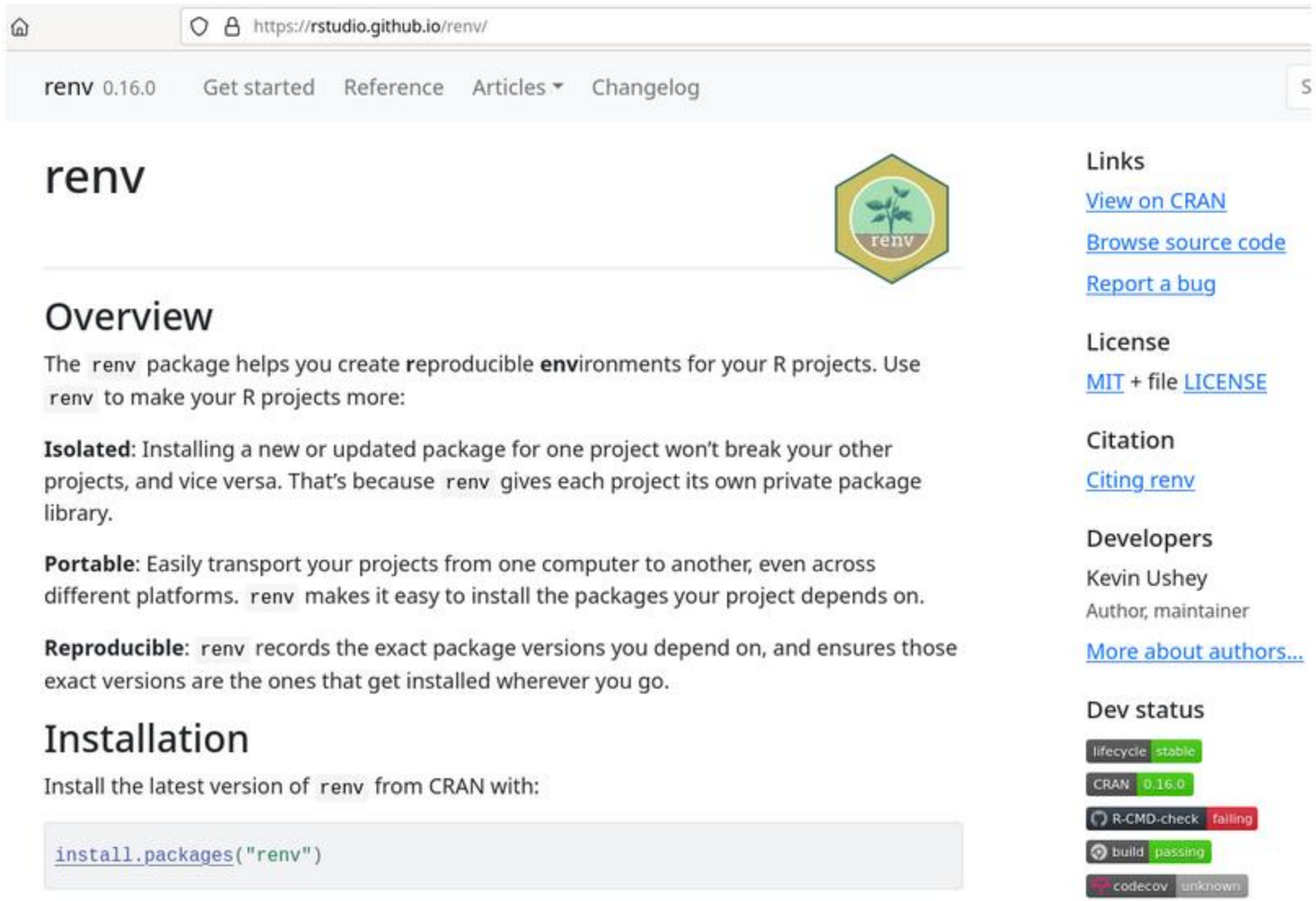
4.3. Level 3: renv - for packages

Version control in work "environments"





4.3.1. Virtual environments in R with renv



https://rstudio.github.io/renv/

renv 0.16.0 Get started Reference Articles Changelog

renv

Overview

The `renv` package helps you create **r**eproducible **env**ironments for your R projects. Use `renv` to make your R projects more:

Isolated: Installing a new or updated package for one project won't break your other projects, and vice versa. That's because `renv` gives each project its own private package library.

Portable: Easily transport your projects from one computer to another, even across different platforms. `renv` makes it easy to install the packages your project depends on.

Reproducible: `renv` records the exact package versions you depend on, and ensures those exact versions are the ones that get installed wherever you go.

Installation

Install the latest version of `renv` from CRAN with:

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

Links

View on CRAN

Browse source code

Report a bug

License

MIT + file LICENSE

Citation

Citing renv

Developers

Kevin Ushey

Author, maintainer

More about authors...

Dev status

lifecycle stable

CRAN 0.16.0

R-CMD-check failing

build passing

codecov unknown

4.3.2. From utils: : sessionInfo() to renv: : snapshot() + renv.lockalso fails

utils::sessionInfo()

```
> sessionInfo()

R version 4.1.2 (2021-11-01)

Platform: x86_64-pc-linux-gnu (64-bit)

Running under: Ubuntu 22.04.1 LTS

Matrix products: default

BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0

LAPACK: /usr/lib/x86_64-linux-gnu/lapack
/liblapack.so.3.10.0

locale:

 [1] LC_CTYPE=ca_ES.UTF-8       LC_NUMERIC=C
LC_TIME=ca_ES.UTF-8

 [4] LC_COLLATE=ca_ES.UTF-8
LC_MONETARY=ca_ES.UTF-8
LC_MESSAGES=ca_ES.UTF-8

 [7] LC_PAPER=ca_ES.UTF-8       LC_NAME=C
LC_ADDRESS=C

[10] LC_TELEPHONE=C
LC_MEASUREMENT=ca_ES.UTF-8 LC_IDENTIFICATION=C

attached base packages:

[1] stats     graphics  grDevices datasets  utils     methods
base

other attached packages:

 [1] kableExtra_1.3.4 fs_1.5.2       tictoc_1.1
lubridate_1.9.0  timechange_0.1.1

 [6] janitor_2.1.0    knitr_1.40      markdown_1.3
RODBC_1.3-19     fst_0.9.8

[11] forcats_0.5.2    stringr_1.4.1   dplyr_1.0.10
purrr_0.3.5      readr_2.1.3

[16] tidyr_1.2.1      tibble_3.1.8    ggplot2_3.4.0
tidyverse_1.3.1

loaded via a namespace (and not attached):

 [1] httr_1.4.4       jsonlite_1.8.3   viridisLite_0.4.1
modelr_0.1.10    assertthat_0.2.1

 [6] renv_0.16.0      cellranger_1.1.0 yaml_2.3.6
pillar_1.8.1     backports_1.4.1

[11] glue_1.6.2       digest_0.6.30    rvest_1.0.3
snakecase_0.11.0 colorspace_2.0-3

[16] htmltools_0.5.3  pkgconfig_2.0.3  broom_1.0.1
haven_2.5.1      scales_1.2.1

[21] webshot_0.5.4    svglite_2.1.0    openxlsx_4.2.5.1
rio_0.5.29       tzdb_0.3.0

[26] generics_0.1.3   ellipsis_0.3.2   withr_2.5.0      cli_3.4.1
magrittr_2.0.3
```

renv::snapshot() i ./renv.lock


```
{
 "R": {
  "Version": "4.1.2",
  "Repositories": [
   {
    "Name": "CRAN",
    "URL": "https://cloud.r-project.org"
   }
  ]
 },
 "Packages": {
  "DBI": {
   "Package": "DBI",
   "Version": "1.1.3",
   "Source": "Repository",
   "Repository": "CRAN",
   "Hash": "b2866e62bab9378c3cc9476a1954226b",
   "Requirements": []
  },
  "tinytex": {
   "Package": "tinytex",
   "Version": "0.42",
   "Source": "Repository",
   "Repository": "CRAN",
   "Hash": "7629c6c1540835d5248e6e7df265fa74",
   "Requirements": [
    "xfun"
   ]
  },
  "tzdb": {
   "Package": "tzdb",
   "Version": "0.3.0",
   "Source": "Repository",
   "Repository": "CRAN",
   "Hash": "b2e1cbce7c903eaf23ec05c58e59fb5e",
   "Requirements": [
    "cpp11"
   ]
  },
  "zip": {
```

4.3.3. "Happy path"

For a reproducible environment

Commands in terminal - Computer 1

```
cd project_folder
git init
R
[obrir projecte de RStudio]
renv::init() # to initialize renv in your code project
renv::snapshot() # to make a snapshot "picture" of the list of R packages used within the whole R project and their respective package versions
q()
git commit ...
git push
```

Commands in terminal - Computer 2

```
cd project_folder
git clone/git pull ...
R
[open same RStudio project]
renv::status() # for a report on which steps are suggested for you to follow
renv::restore() # to restore the package library (with the required pacckage versions) for this project
[continue working in/developing your code]
renv::snapshot() # to make a new snapshot "picture" (in case there are new packages and/or versions or R packages neweer or older in use in your project ;-) )
q()
git commit ...
git push
```

4.3.4. Infraestructure

Projects with `renv` write and use these files in order to work:

File	Use
.Rprofile	Used to activate renv for new R sessions launched in the project.
renv.lock	The lockfile, describing the state of your project's library at some point in time.
renv/activate.R	The activation script run by the project **.Rprofile**.
renv/library	The private project library.
renv/settings.dcf	Project settings – see `?settings` for more details.

By default, `renv` uses a package memory-cache here:

Platform	Location
Linux	`~/.local/share/renv`
macOS	`~/Library/Application Support/renv`
Windows	`%LOCALAPPDATA%/renv`

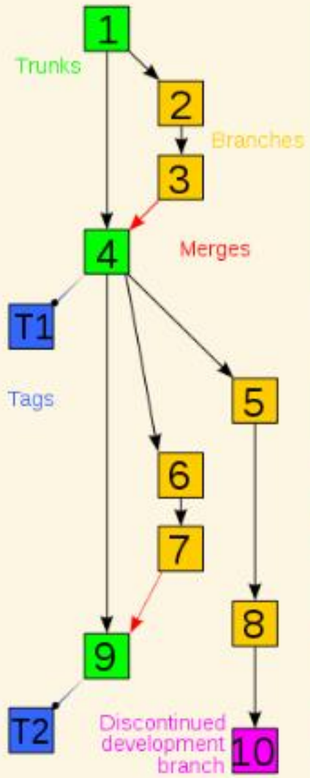
4.3.5. Advanced use

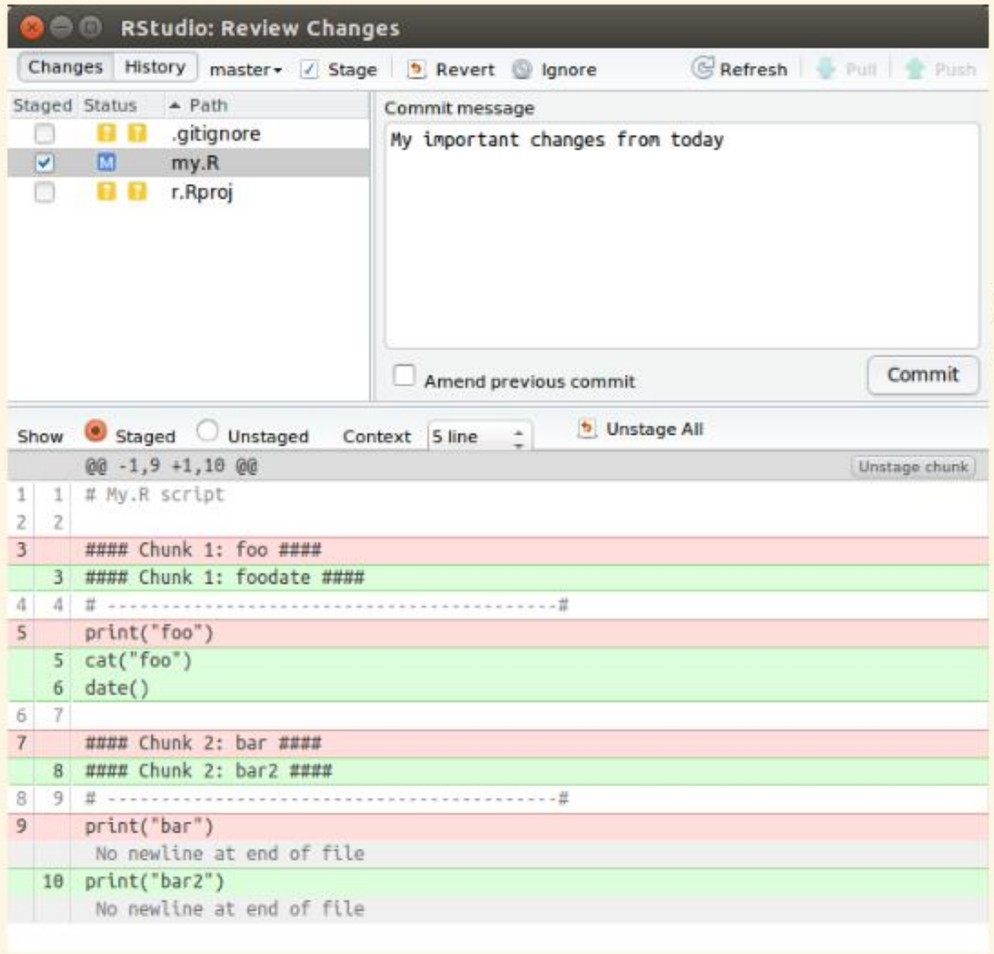
```r
renv::install("packagename", version="0.1") # to install old versions from a package (useful also for discontinued packages in CRAN!). See possible package-version numbers at https://cran.r-project.org/src/contrib/Archive/yourpackage/
renv::record("packagename", version="0.1") # to save at renv.lock the specific version you need for this package
renv::deactivate() # to temporarily deactivate renv in your project
renv::activate() # to reactivate renv in your project
renv::equip() # for special installations in MS Windows
vignette("docker", package = "renv") # for a commbined use with Docker
vignette("collaborating", package = "renv") # to improve collaborative use in work teams
```

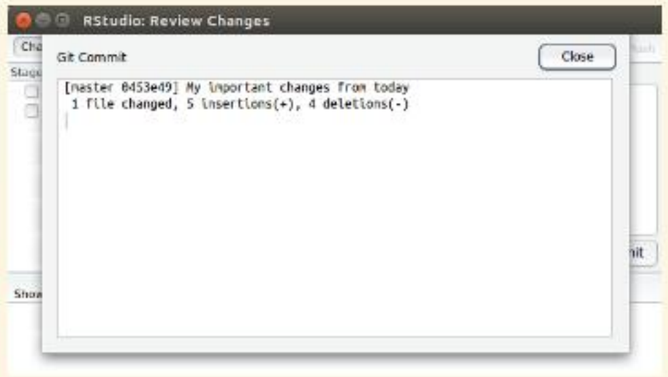
And much more. See:

- https://rstudio.github.io/renv/articles/renv.html
- https://solutions.posit.co/envs-pkgs/environments/

4.4. Level 4: git - for code







See: https://gitlab.com/radup/curs-r-introduccio/ > Folder "codi" > **10.compartir.via.git.Rmd** (or .pdf)

See also my own git recipes over some years, github cheatsheet, ...: https://seeds4c.org/git

5. More information

Work Environments in R

- https://solutions.posit.co/envs-pkgs/environments/

Videos

- An Introduction to Reproducible Research Practices. 29 d'abr. 2022. John Little. Duke University. Video
- Designing a Reproducible Workflow with R and GitHub. John Little. 22 de nov. 2021 Video | Tutorial
- The workflowr R package: a framework for reproducible and collaborative data science. 13 de jul. 2018. R Consortium. Video
- Kevin Ushey | renv: Project Environments for R | RStudio (2020). Posit PBC.. 20 de des. 2020. Video

R Packages

renv | workflowr | learnr | roxygen2 | Tidyverse

Free Work environments for Collaborative Data Science with R & Python

- https://posit.cloud/plans/free

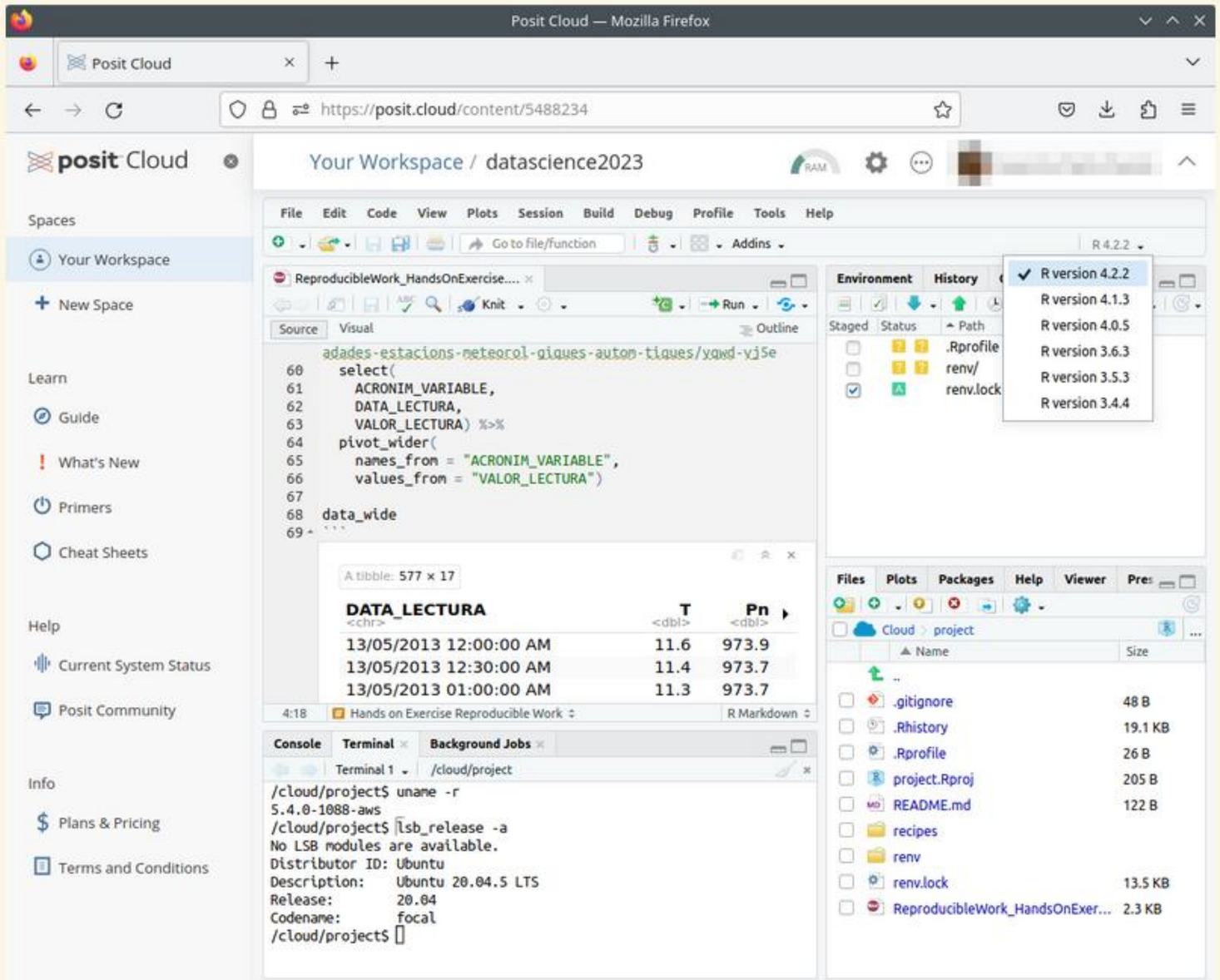
Additional tutorial with big data to follow on site (R Cloud)

- Danielle Navarro. 2022. "Using Amazon S3 with R" March 17, 2022.

Papers

- Wallach JD, Boyack KW, Ioannidis JPA. (2018) Reproducible research practices, transparency, and open access data in the biomedical literature, 2015–2017. PLoS Biol 16 (11): e2006930. https://doi.org/10.1371/journal.pbio.2006930
- Leek JT, Peng RD. Opinion: Reproducible research can still be wrong: adopting a prevention approach. Proc Natl Acad Sci U S A. 2015 Feb 10;112(6):1645-6. doi: 10.1073/pnas.1421412111. PMID: 25670866; PMCID: PMC4330755

6. Hand on practical exercise



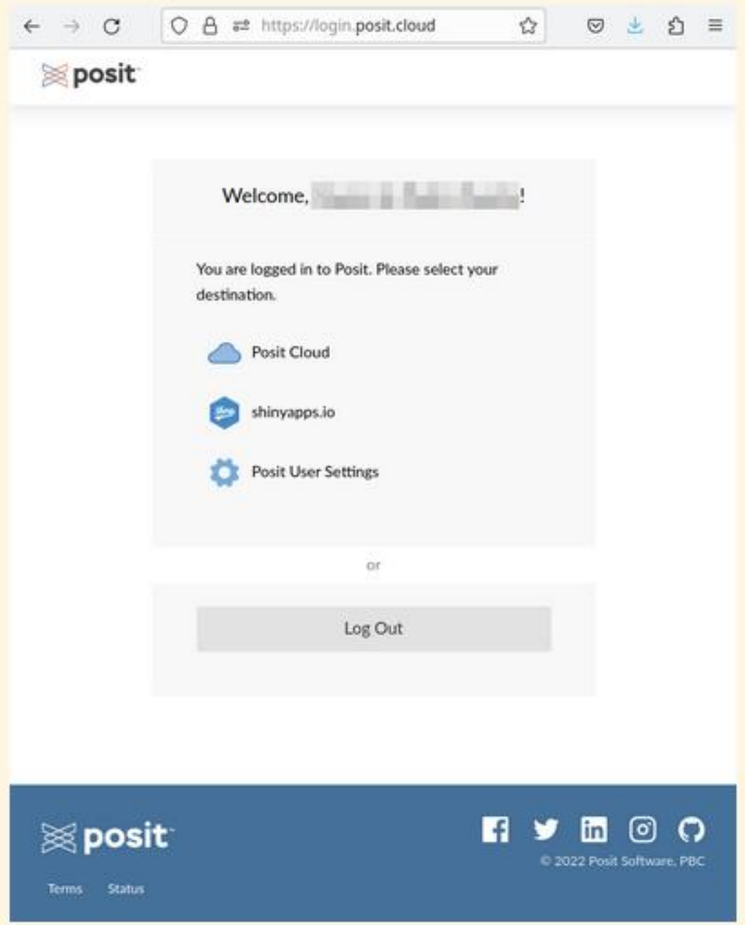
6.1. Register a free account a Posit Cloud

You can do so at:

- https://posit.cloud/plans/free

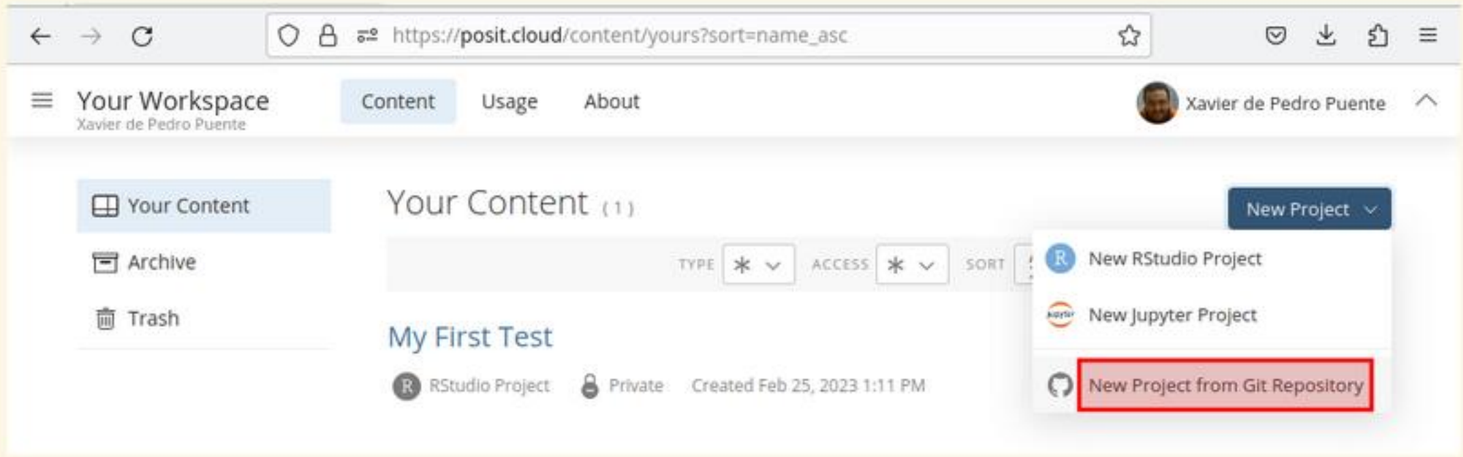
You will need to click on a link sent to your email inbox to validate your account.

Once done, you'll see something like:



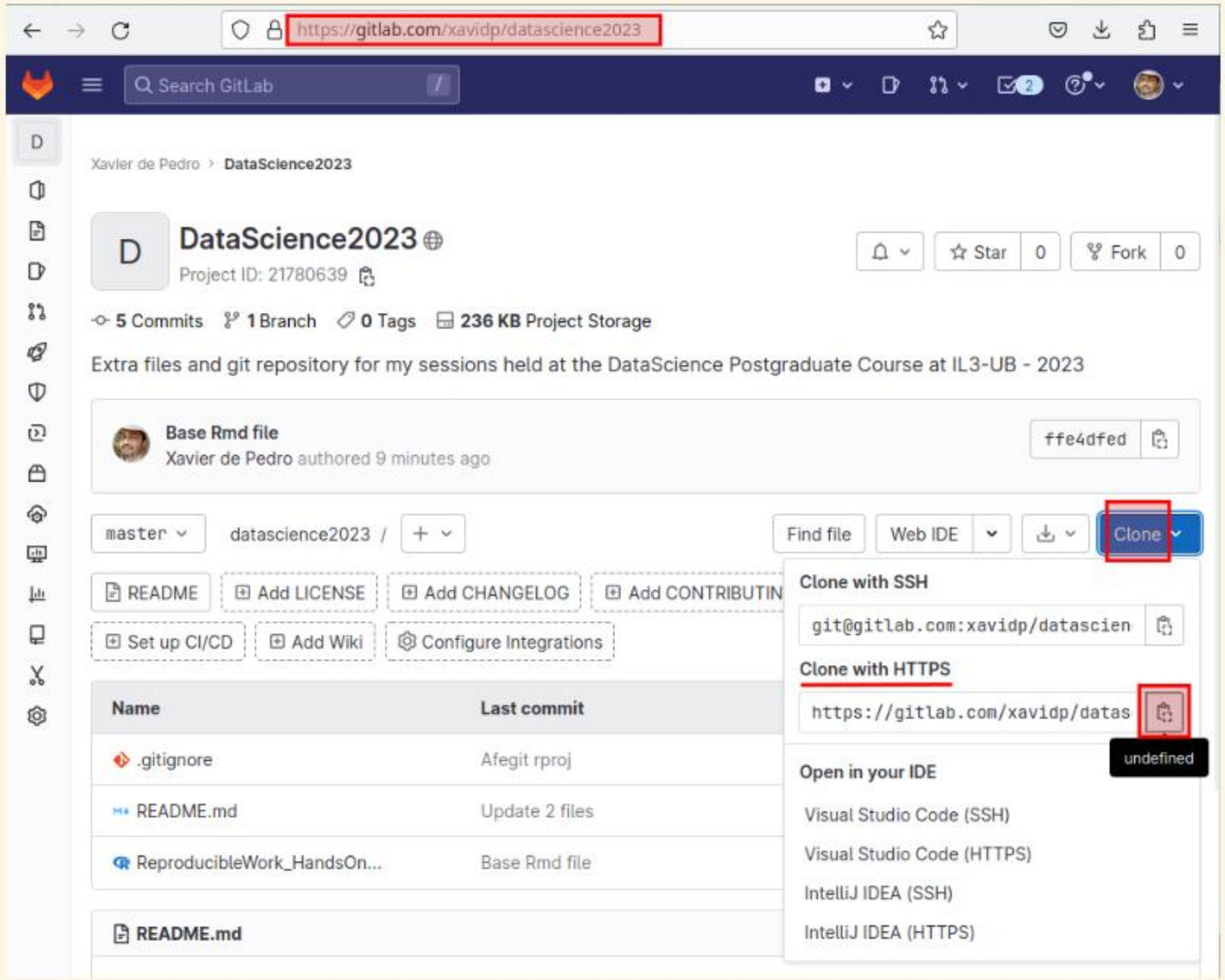
6.2. Create a Project from git repository

Enter Posit cloud and click at **New Project > New Project from Git Repository**



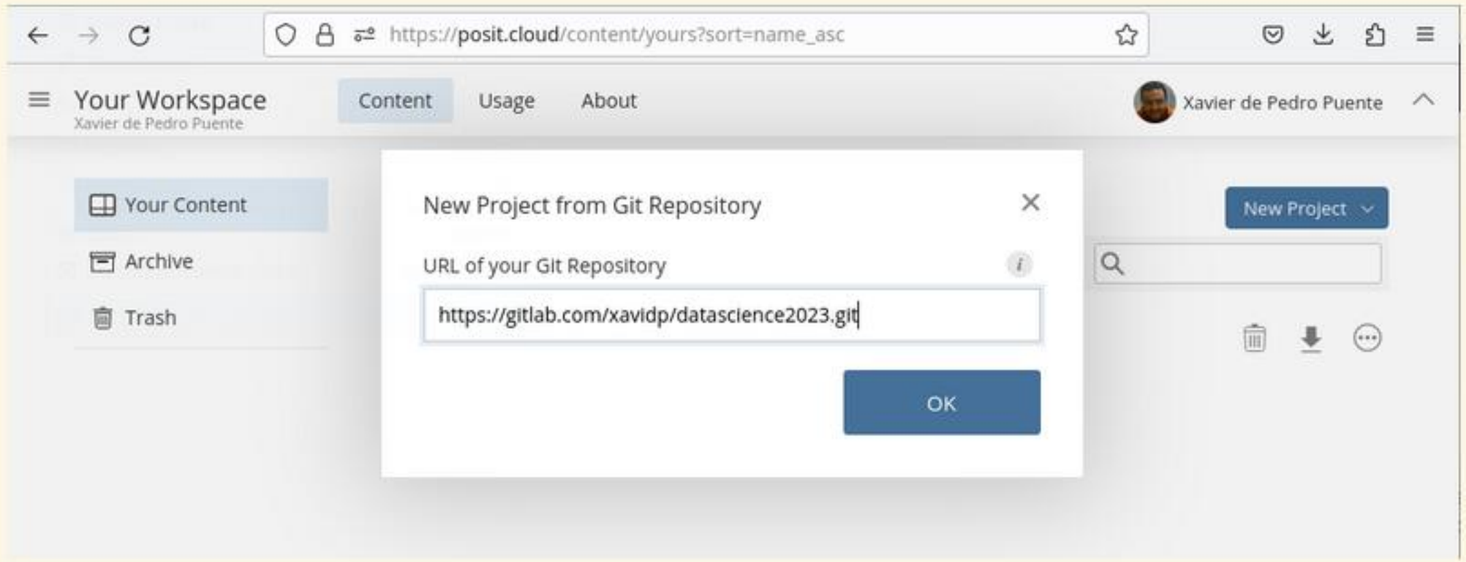
6.2.1. Visit gitlab to get clone url

Visit this code project in gitlab to get the project clone url:
https://gitlab.com/xavidp/datascience2023

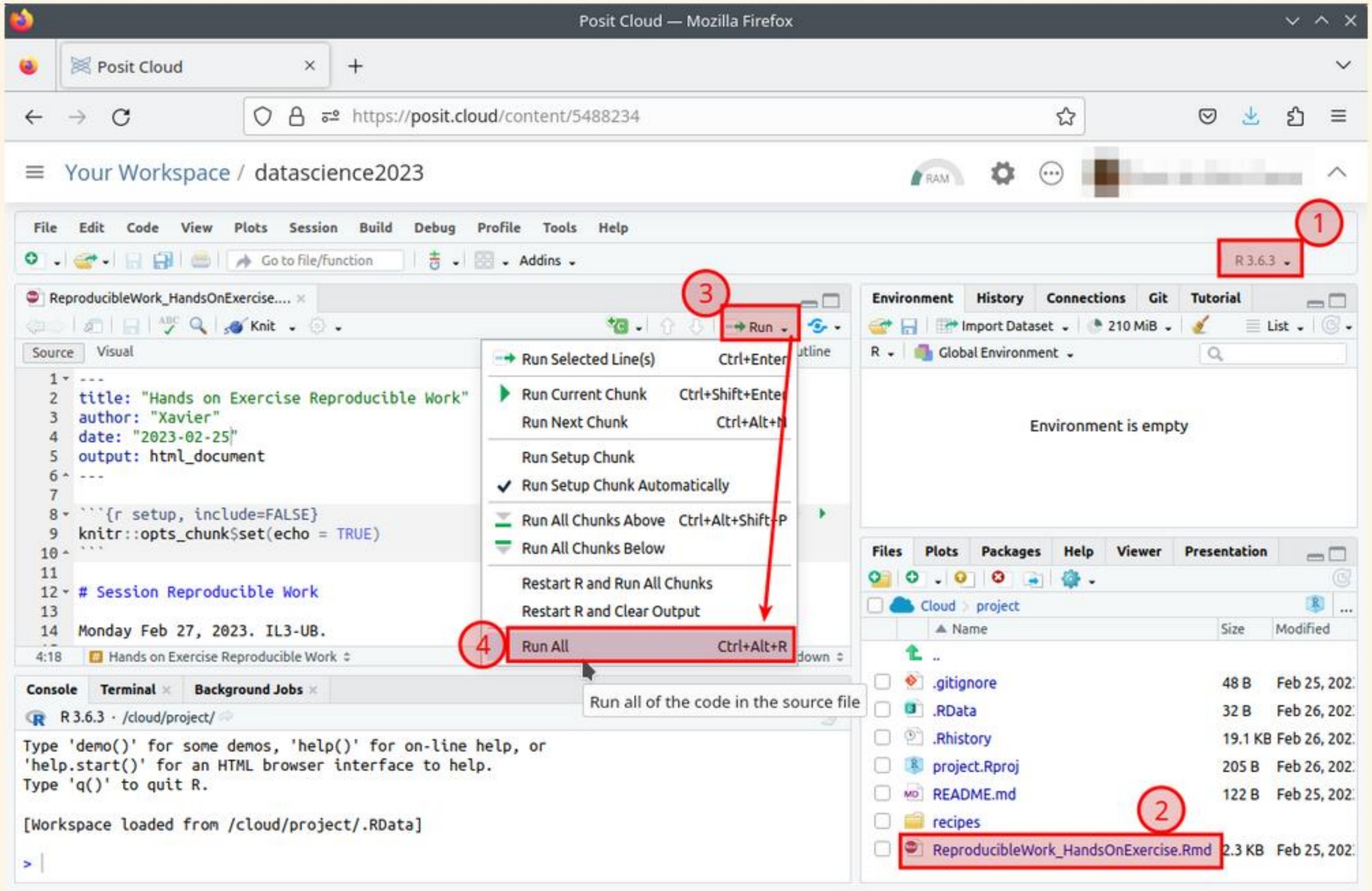


6.2.2. Create project from git repo

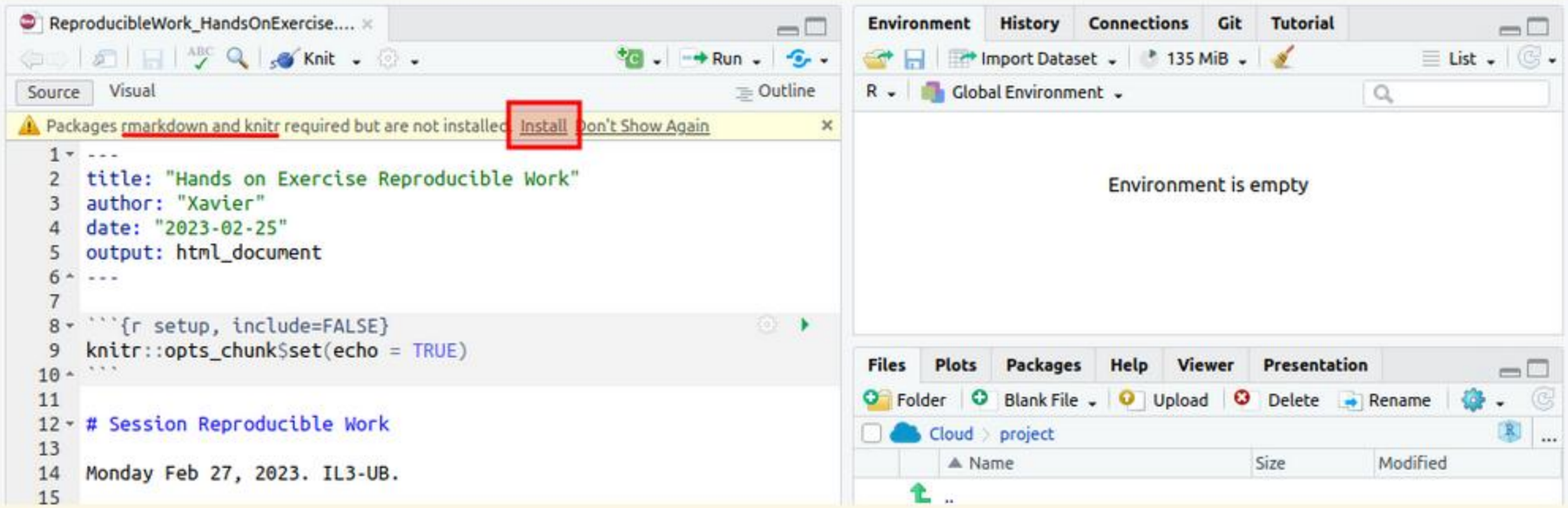
Paste it in the Posit cloud popup window and click at OK:

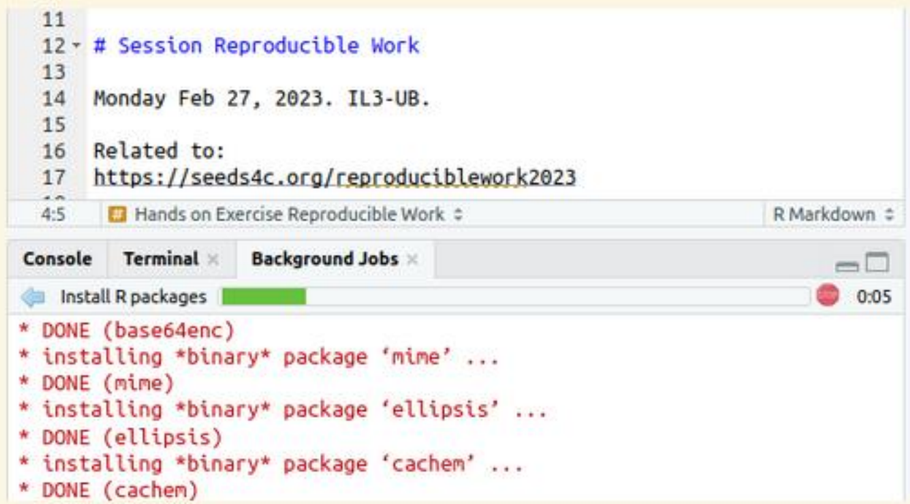


6.3. Choose R 3.6.x & Run Rmd



6.3.1. Install dependencies also



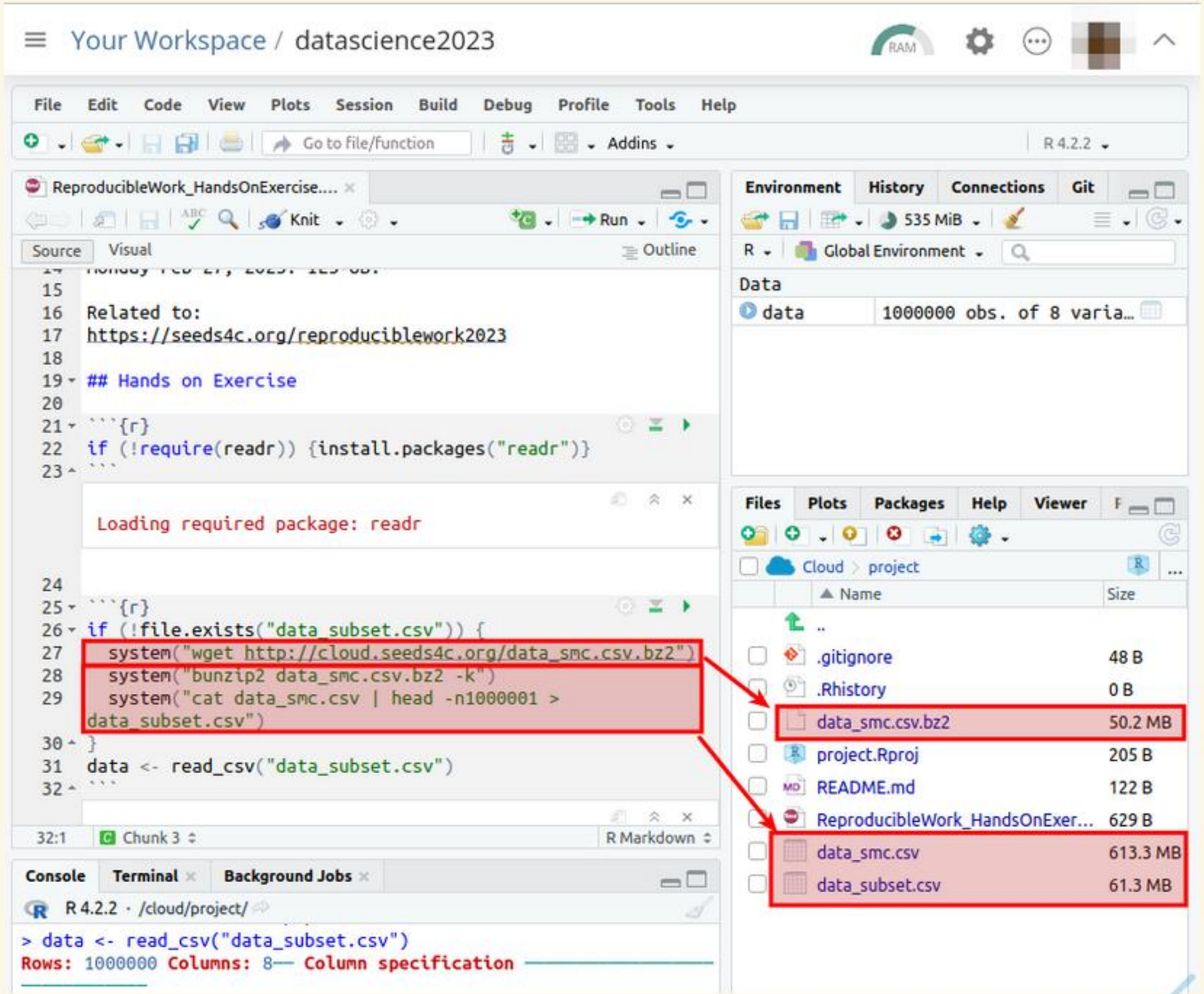


6.3.2. Running Rmd will perform GNU/Linux system commands also

GNU/Linux system commands will usually be much more efficient in memory & cpu

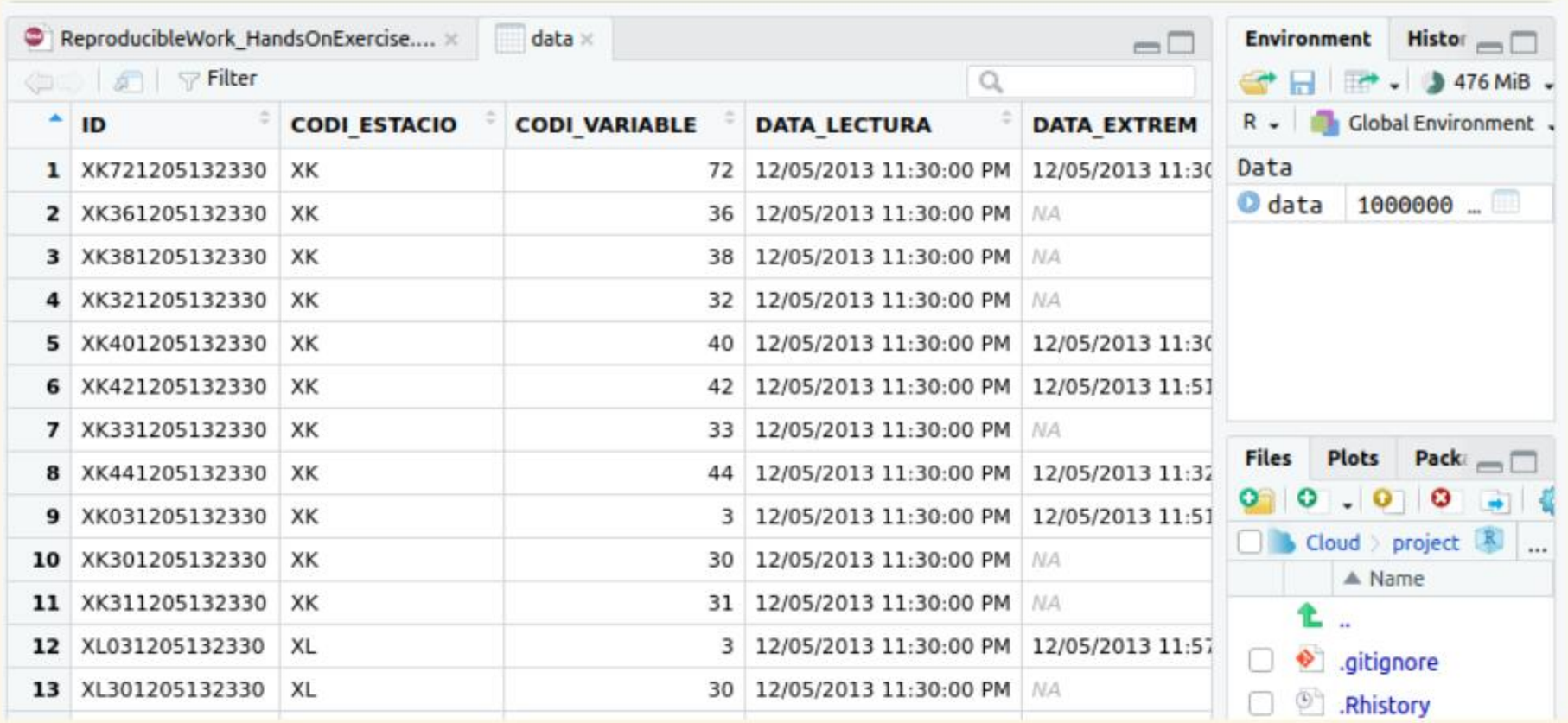
It helps to prevent RAM bottlenecks with just 1Gb RAM on Posit Cloud Free plan

(while csv file from reduced meteorological dataset is already 0.5 Gb).



6.3.3. Display raw data

Variables are in numeric codes (not easily readable by humans in a semantic way). We lack some varaible names (or acronyms at least) for readability.



	ID	CODI_ESTACIO	CODI_VARIABLE	DATA_LECTURA	DATA_EXTREM
1	XK721205132330	XK	72	12/05/2013 11:30:00 PM	12/05/2013 11:30
2	XK361205132330	XK	36	12/05/2013 11:30:00 PM	NA
3	XK381205132330	XK	38	12/05/2013 11:30:00 PM	NA
4	XK321205132330	XK	32	12/05/2013 11:30:00 PM	NA
5	XK401205132330	XK	40	12/05/2013 11:30:00 PM	12/05/2013 11:30
6	XK421205132330	XK	42	12/05/2013 11:30:00 PM	12/05/2013 11:51
7	XK331205132330	XK	33	12/05/2013 11:30:00 PM	NA
8	XK441205132330	XK	44	12/05/2013 11:30:00 PM	12/05/2013 11:32
9	XK031205132330	XK	3	12/05/2013 11:30:00 PM	12/05/2013 11:51
10	XK301205132330	XK	30	12/05/2013 11:30:00 PM	NA
11	XK311205132330	XK	31	12/05/2013 11:30:00 PM	NA
12	XL031205132330	XL	3	12/05/2013 11:30:00 PM	12/05/2013 11:57
13	XL301205132330	XL	30	12/05/2013 11:30:00 PM	NA

6.3.4. Transform in tidy way (i)

```{r}
# Get the description of the variable codes
# From here: https://analisi.transparenciacatalunya.cat/Medi-Ambient/Metadades-variables-meteorol-giques/4fb2-n3yi/data
variables <- read_csv("https://analisi.transparenciacatalunya.cat/api/views/4fb2-n3yi/rows.csv?accessType=DOWNLOAD&sorting=true")
```

```
Rows: 26 Columns: 6── Column specification ────────────────────────────────────────
Delimiter: ","
chr (4): NOM_VARIABLE, UNITAT, ACRONIM, CODI_TIPUS_VAR
dbl (2): CODI_VARIABLE, DECIMALS
ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```{r}
# We prepare a small dataframe from the variable definition to join on the smc data frame
variables.to.join <- variables %>%
  select(CODI_VARIABLE, ACRONIM) %>%
  arrange(CODI_VARIABLE)

variables.to.join
```

A tibble: 26 × 2

CODI_VARIABLE <dbl>	ACRONIM <chr>
1	Px
2	Pn
3	HRx
30	VV10

6.3.5. Transform in tidy way (ii) - result

````
```{r}
# Let's join variable df on to the data df
data <- left_join(data, variables.to.join) %>%
  rename(ACRONIM_VARIABLE = ACRONIM)
```
````

```
Joining, by = "CODI_VARIABLE"
```

````
```{r}
# Let's convert the source data frame (which is long shape, as database) into a wide shape (table like, with meteorological variables as columns) while selecting just one meteorological station as an example
data_wide <- data %>%
  filter(CODI_ESTACIO =="D5") %>% # D5 corresponds to "Barcelona Observatori Fabra" Meteorological Observatory (at Collserola Mountain) https://analisi.transparenciacatalunya.cat/Medi-Ambient/Metadades-estacions-meteorol-giques-autom-tiques/yqwd-vj5e
  select(
    ACRONIM_VARIABLE,
    DATA_LECTURA,
    VALOR_LECTURA) %>%
  pivot_wider(
    names_from = "ACRONIM_VARIABLE",
    values_from = "VALOR_LECTURA")

data_wide
```
````

A tibble: 577 × 17

DATA_LECTURA <chr>	T <dbl>	Pn <dbl>	Tn <dbl>	HR <dbl>	HRn <dbl>	HRx <dbl>	VV10 <dbl>	DV10 <dbl>	VVx10 <dbl>
13/05/2013 12:00:00 AM	11.6	973.9	11.4	91	91	92	2.0	238	2.7
13/05/2013 12:30:00 AM	11.4	973.7	11.4	90	90	91	1.5	238	2.4
13/05/2013 01:00:00 AM	11.3	973.7	11.3	89	87	91	1.1	174	2.3
13/05/2013 01:30:00 AM	11.3	973.6	11.3	89	88	91	1.5	209	2.4

6.3.6. Last code chunks

````
70
71 ```{r}
72 # Save resulting dataset to disk
73 write_csv(data_wide, "data_subset_d5_wide.csv")
74 ```
75
76
77 ```{r}
78 # Produce a simple R version of this R Markdown document
79 knitr::purl("ReproducibleWork_HandsOnExercise.Rmd", documentation=2)
80 ```
````

```
[1] "ReproducibleWork_HandsOnExercise.R"
```

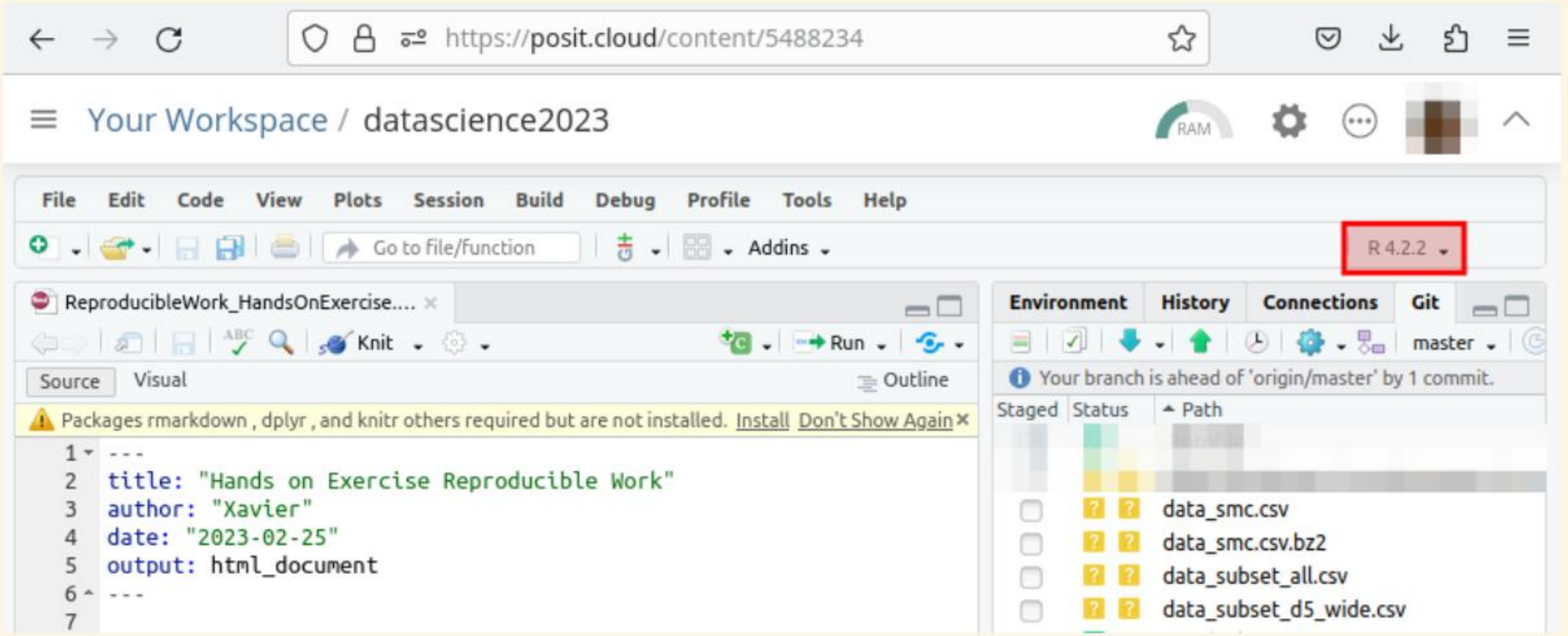
```
81
82
```

4:18 Hands on Exercise Reproducible Work

6.4. Choose R 4.2.x & Run Rmd again

Repeat the previous steps but in a R 4.2.x environment: install dependent R packages again... (new environment, but still installing from CRAN repos). renv not needed in this case still (lucky you!).

So far, so good.



6.5. Choose R 3.4.x & Run Rmd

Now let's touch some issues with R package versions in a R 3.4.x environment

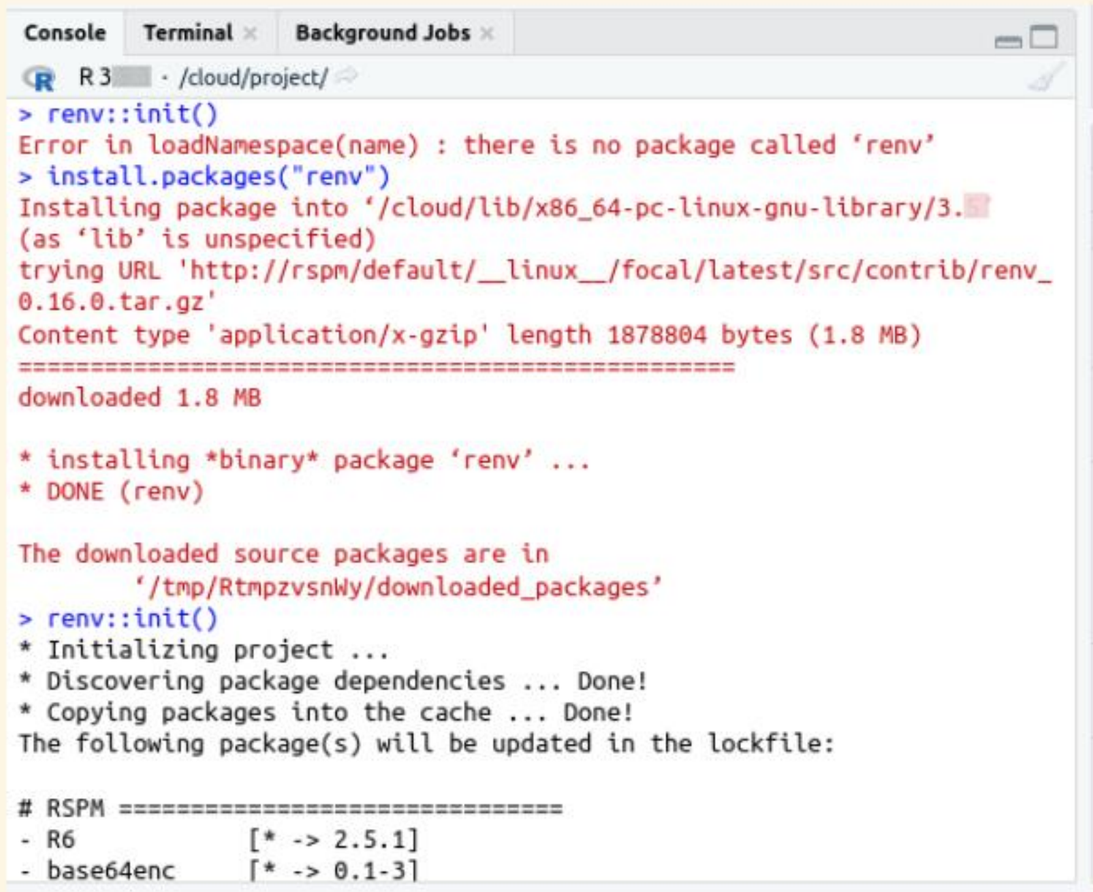
Running Rmd will fail at some package installations

- `dplyr` installation fails
- `readr` is reported as unavailable in R 3.4.4
- `tidyr` installation also fails (as well as `purrr`)

Solution

In this case, the solution involves finding some valid previous package version for each conflicting R package, and using this type of commands:

- **renv::init()**
- **renv::install("packagename@x.y.z")** # being x.y.z a valid package version number, as taken from https://cran.r-project.org/src/contrib/Archive/packagename/
- **renv::record("packagename@x.y.z")**
- **renv::snapshot()** # after all packages installed without any more issues

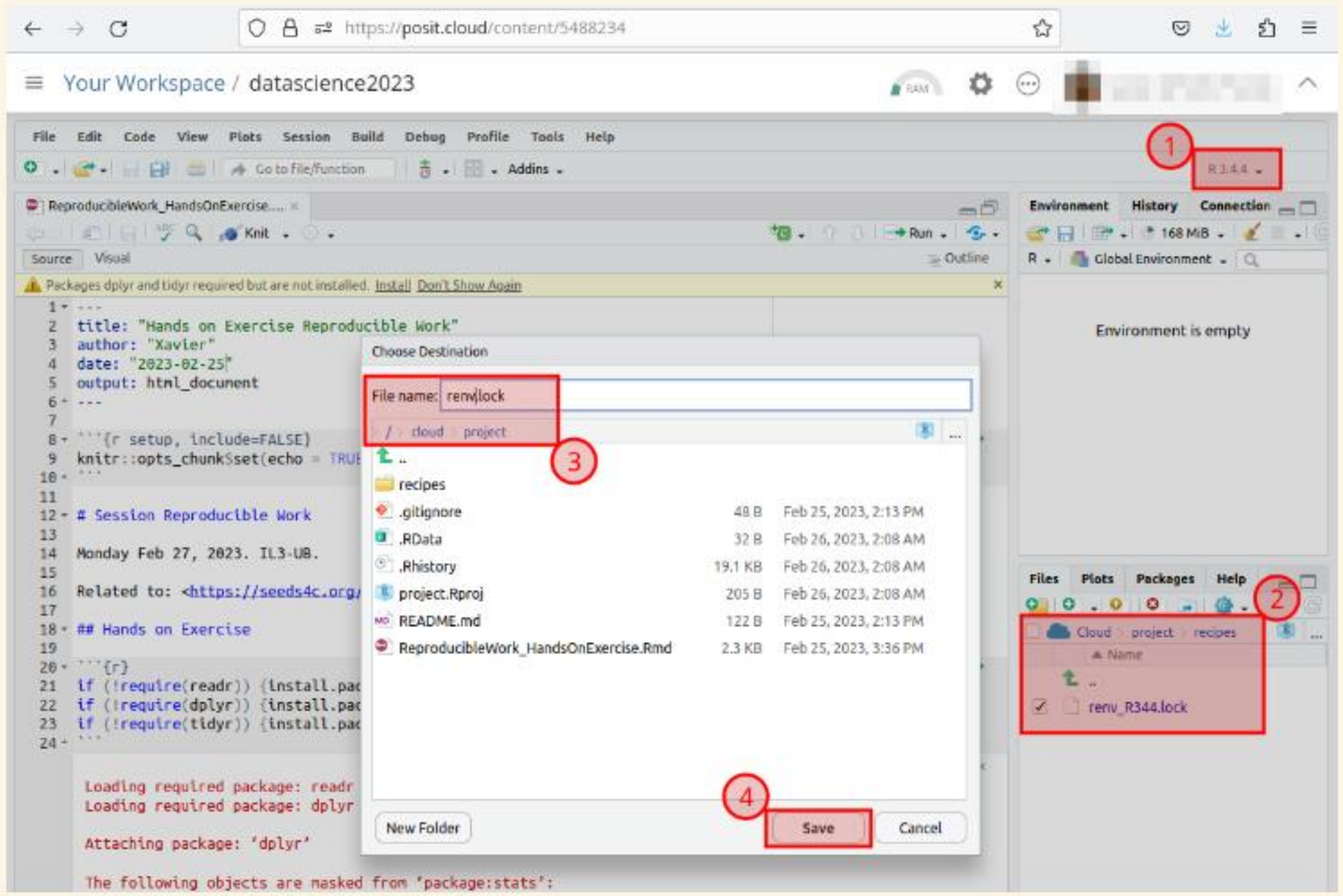


6.5.1. Use renv.lock recipe (i)

Let's get `renv` to the rescue. Once somebody solved these issues, and found a valid recipe of package versions for this environment, a file **./renv.lock** will have been produced in the project root folder after running the command `renv::snapshot()`

I did this already, and I uploaded the produced **renv.lock** file to the manually created **./recipes/** folder in this project as a backup for you (as **renv_R344.lock**).

You can then copy now the **./recipes/renv_R344.lock** file provided in the project as **./renv.lock** in the project root folder, for `renv` to be able use it.

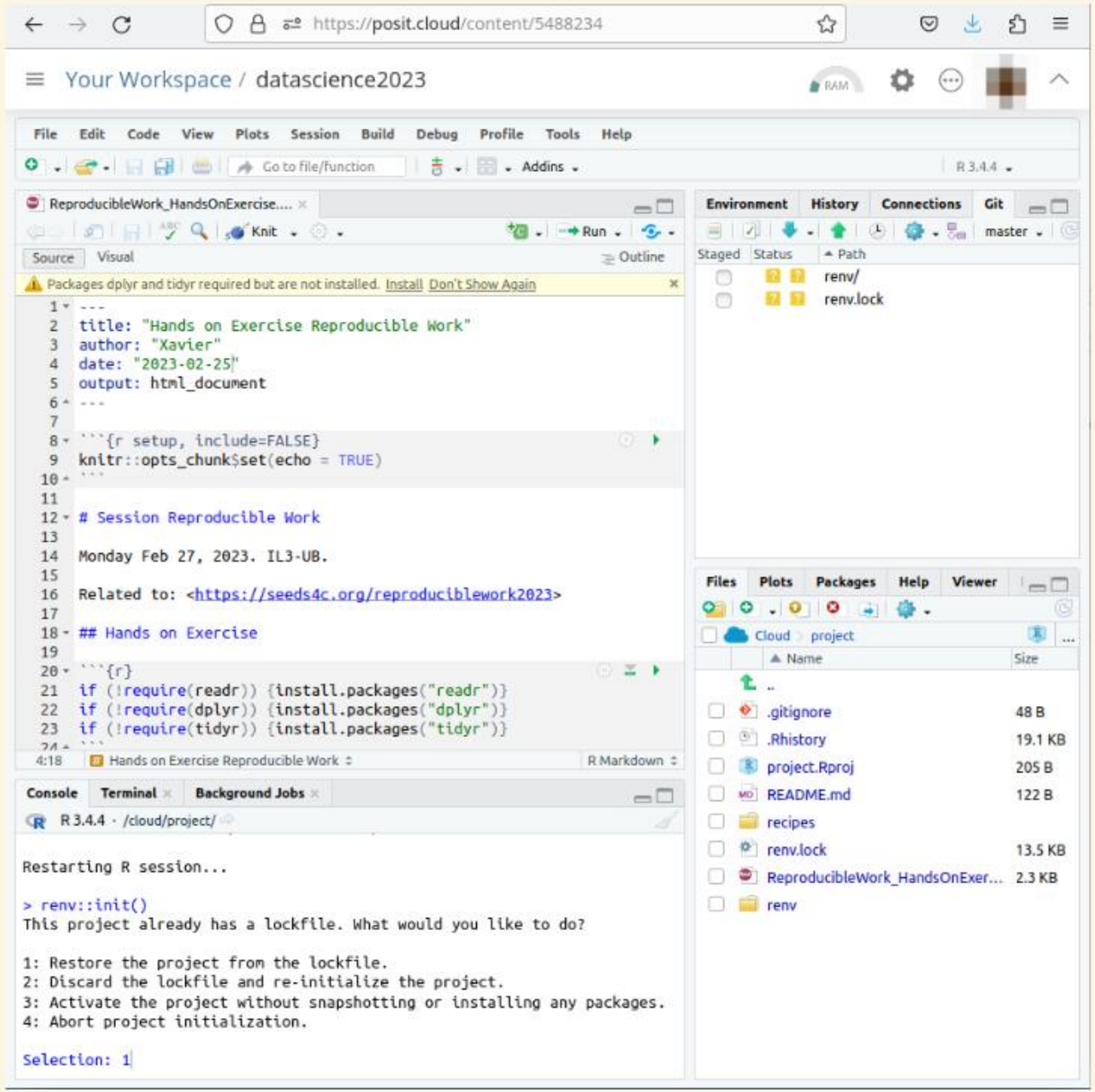


6.5.2. Use renv.lock recipe (ii)

Run `renv::init()` in the R console.

Choose restore the renv.lock package versions:

"1. Restore the project from the lockfile"

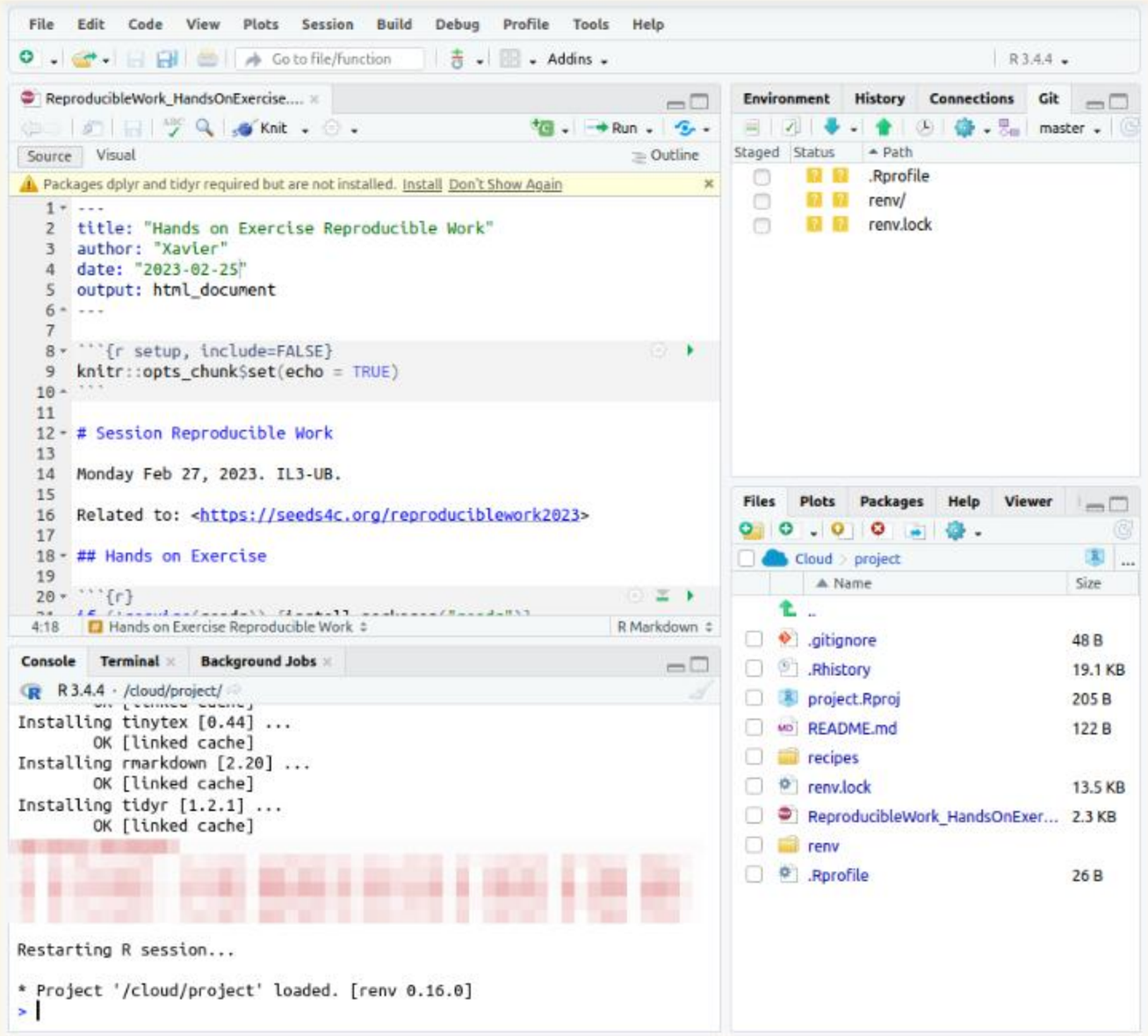


6.5.3. Use renv.lock recipe (iii)

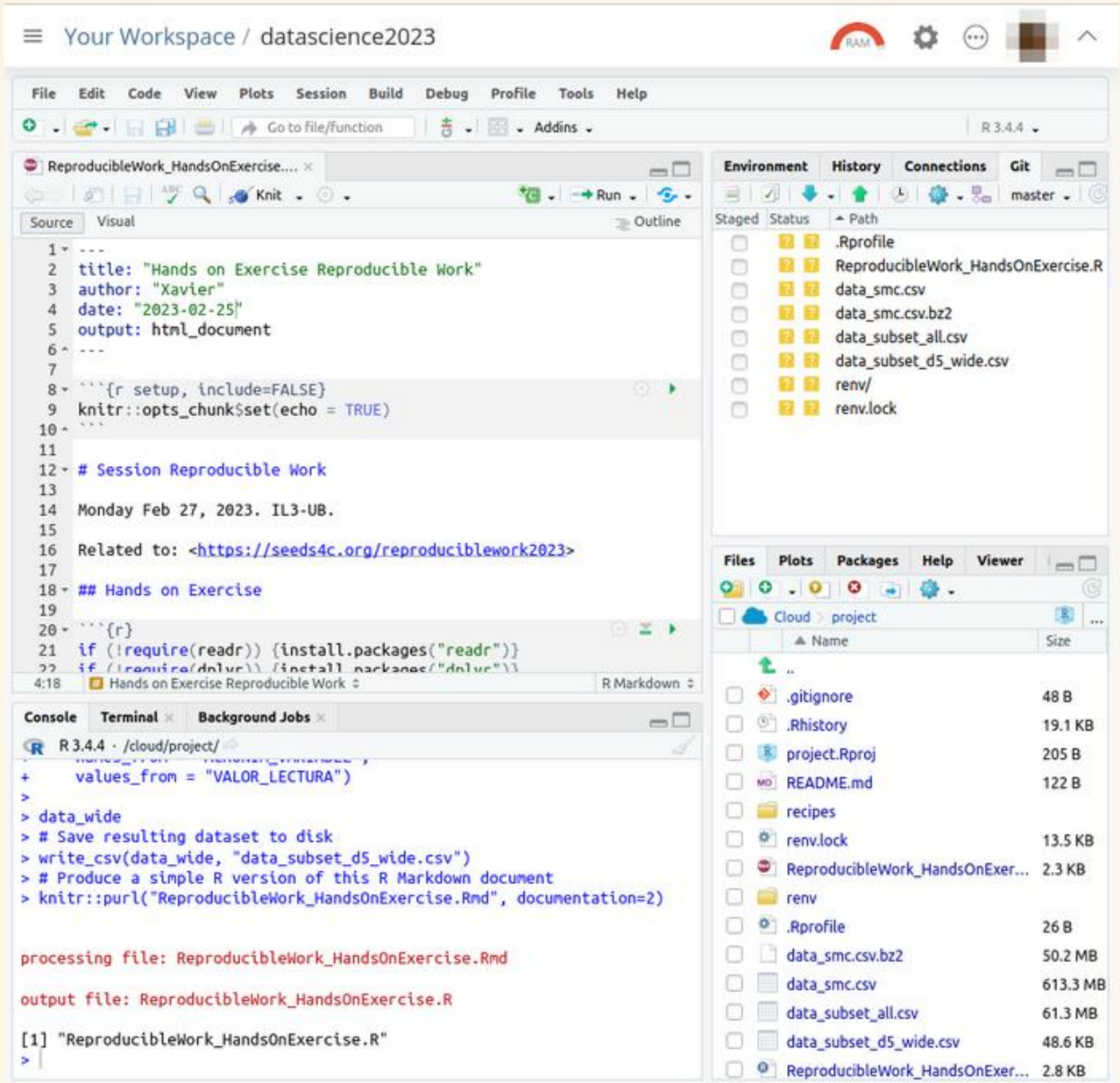
You will be ready to go with minimum human intervention.

All R packages will be installed in the backgound to their required package versions, following the recipe that someone created for R 3.4.4. already.

The key file is the **renv.lock** file.

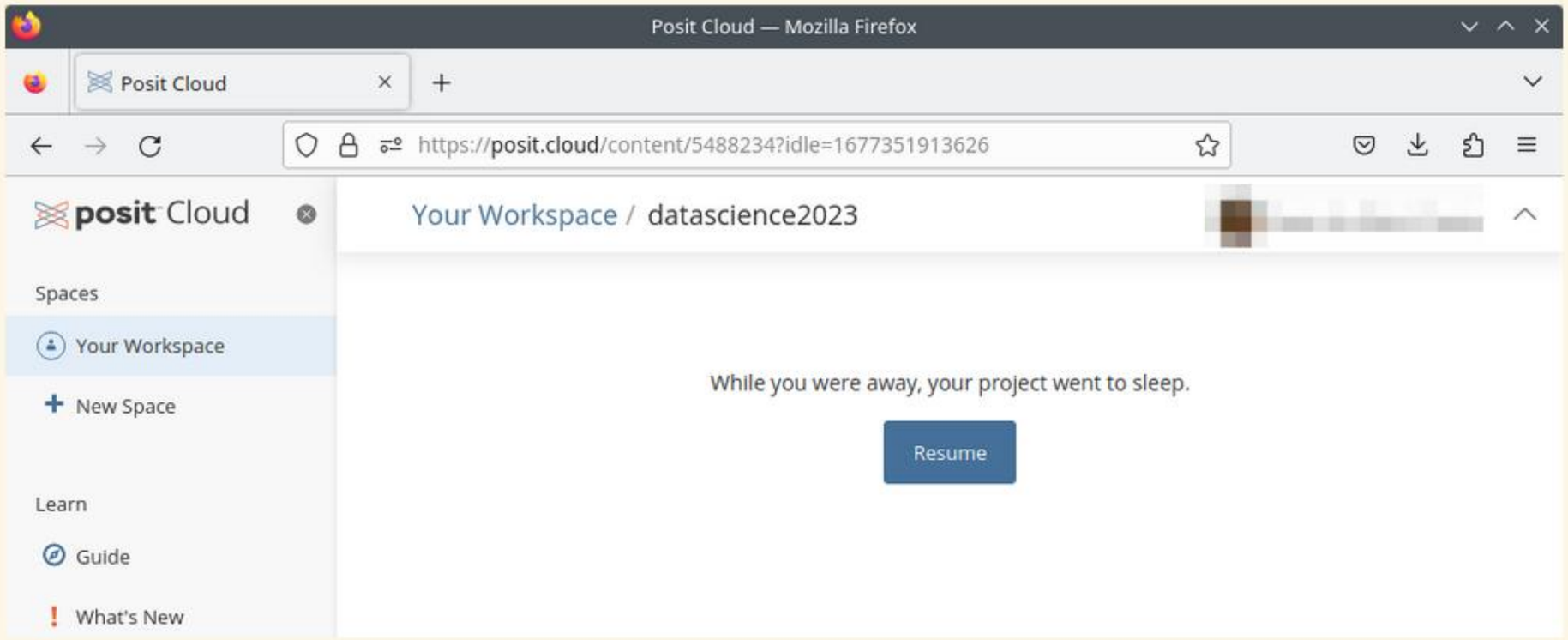


6.5.4. Use renv.lock recipe (iv) - finished



6.6. Additional info

Project (Container) goes to sleep on inactivity



Thanks

Xavier de Pedro Puente, Ph.D. -
xavier.depedro@seeds4c.org

Slides available at:
https://seeds4c.org
/reproduciblework2023