rix: Reproductibilité d'Environnements De Développement R Avec Nix

Bruno Rodrigues, Philipp Baumann

Introduction

- {rix} utilise Nix, un gestionnaire de paquets pensé pour la reproductibilité.
- Avec Nix, tu peux créer un environnement R sur mesure pour chaque projet.
- Tu choisis la version de R, les packages, et tout est installé automatiquement jusqu'aux dépendances système.
- L'environnement est entièrement figé, parfait pour la recherche ou les projets à long terme.
- Imagine {renv} et Docker en un seul outil.

Nix

- Nix est un gestionnaire de paquets que tu peux installer sur n'importe quel système.
- Nix fonctionne sous Linux, macOS et Windows (via WSL2).
- Son point fort : la reproductibilité des installations.
- Le dépôt de logiciels de Nix inclut CRAN et Bioconductor (et plus de 100'000 autres paquets).
- Avec Nix:
  - tous les collaborateurs d'un projet ont *exactement* le même environnement
  - on peut déployer une application Shiny avec *exactement* le même environnement en prod qu'en dev
  - on peut exécuter du code en intégration continue avec *exactement* le même environnement que localement
  - tout ça avec **une seule expression**!
- Mais Nix n'est pas facile à maîtriser… heureusement il y a {rix}!

rix

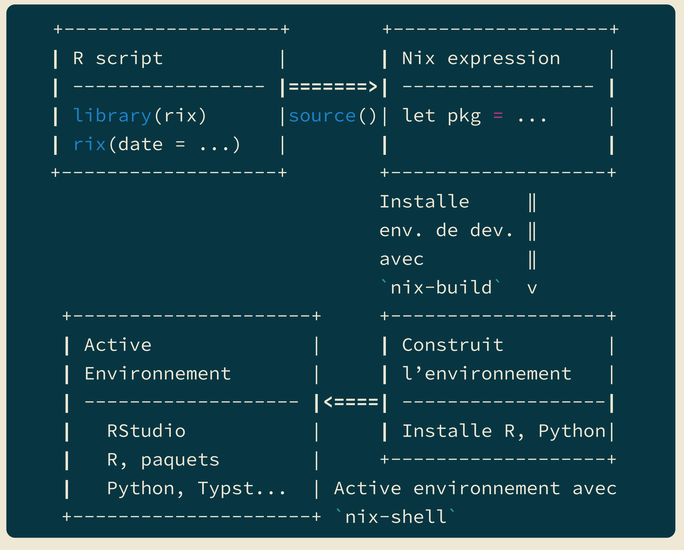
Nix est un outil complexe, mais {rix} :

- simplifie la création d'expressions Nix, y compris à partir de fichiers renv.lock ;
- {rix} est disponible depuis le CRAN et est un paquet rOpenSci!

```r
library(rix)

rix(
  date = "2025-05-05",
  r_pkgs = c(
    "chronicler",
    "dplyr",
    "igraph",
    "reticulate",
    "quarto"
  ),
  system_pkgs = "typst",
  git_pkgs = list(
    package_name = "cmdstanr",
    repo_url = "https://github.com/stan-dev/cmdstanr",
    commit = "79d377"
  ),
  py_conf = list(
    py_version = "3.12",
    py_pkgs = c(
      "pandas",
      "polars",
      "pyarrow")
  ),
  ide = "rstudio", # <- ou Positron, Vs Code...
  project_path = ".",
  overwrite = TRUE
)
```

Flux schématisé:



Mais… et Docker?!

- Nix n'est pas vraiment une alternative à Docker.
- Docker sert à faire des conteneurs ; Nix à gérer les paquets.
- Tu peux viser la reproductibilité avec Docker…
- …mais créer une image Docker n'est pas reproductible, seulement l'exécution d'un conteneur l'est.
- Si tu utilises déjà Docker, pas besoin de changer.
- Tu peux juste utiliser Nix dans tes images Docker pour installer ce qu'il te faut.

Et pour l'intégration continue?

```yaml
- name: Install Nix
  uses: cachix/install-nix-action@v31

- name: Setup Cachix
  uses: cachix/cachix-action@v15
  with:
    name: rstats-on-nix

- name: devtools::test() via nix-shell
  run: nix-shell --run "Rscript -e 'devtools::test()'"
```

Pour en savoir plus

docs.ropensci.org/rix/

Et si t'es fan de {targets}: jette un œil à rixpress![1]

1. https://github.com/b-rodrigues/rixpress