Utilisation de Conda en bio-informatique

Babacar Ndao
Ingénieur en Bio-informatique

PLAN

1. Introduction
2. Présentation de Conda
3. Installation
4. Gestion des environnements
5. Gestion des paquets
6. Astuces et bonnes pratiques
7. Conclusion et ressources

1. Introduction

Objectifs de la formation

•Comprendre ce qu’est Conda et pourquoi il est utile en bio-informatique.

•Apprendre à créer des environnements isolés pour différents projets.

•Installer et gérer facilement des outils bioinformatiques complexes.

1. Introduction

Pourquoi utiliser Conda

• installer des logiciels et bibliothèques facilement, même en ligne de commande

• gérer plusieurs versions de Python et d'outils en parallèle

• garantir des environnements reproductibles et partageables

1. Introduction

Cas d'usage en bio-informatique

- Installer des outils comme samtools, bwa, biopython, ou fastqc en quelques commandes
- Travailler sur plusieurs projets nécessitant des versions différentes de Python ou de bibliothèques
- Préparer des pipelines reproductibles pour des analyses RNA-Seq ou métagénomiques.

1. Introduction

Alternatives à Conda

- pip + virtualenv : bonne gestion de paquets Python, mais moins adaptée aux outils non Python (ex: bowtie2)

- Docker ou Singularity : conteneurs très puissants, mais plus lourds et nécessitent un apprentissage supplémentaire.

2. Présentation de Conda

Qu’est-ce que Conda ?

Conda est un gestionnaire d’environnements et de paquets, multiplateforme (Linux, macOS, Windows), qui permet :

- d’installer facilement des bibliothèques et outils (Python, R, C++, etc.),
- de créer des environnements isolés pour différents projets,
- de gérer les dépendances sans conflits.

Il est particulièrement adapté à la bio-informatique, car il peut gérer des outils complexes qui ne sont pas toujours disponibles via pip.

2. Présentation de Conda

Anaconda, Miniconda et Conda : quelle différence ?

Distribution	Description
conda	Le gestionnaire lui-même (inclus dans Anaconda et Miniconda)
Anaconda	Distribution complète avec +150 paquets préinstallés (Python, R, Jupyter, etc.). Lourde (~3 Go)
Miniconda	Version minimale avec juste Conda et Python. Léger (~50 Mo), recommandé pour la bio-informatique

💡 **Recommandation** : Utilisez **Miniconda** pour plus de contrôle et de légèreté

2. Présentation de Conda

Conda vs pip?

Fonction	conda	pip
Langages gérés	Multi-langage (Python, R…)	Python uniquement
Résolution des dépendances	Avancée (résout automatiquement les conflits)	Basique (pas toujours fiable)
Sources de paquets	Canaux (conda-forge, bioconda, etc.)	PyPI (Python Package Index)
Environnements isolés	Oui	Avec virtualenv/venv
Installation d'outils systèmes	Oui (ex: samtools, bwa, etc.)	Non (limité au Python)

3. Installation

3.1 Choix de la distribution : Miniconda (recommandé)

3.2 Téléchargement

🔗 Site officiel : https://docs.conda.io/en/latest/miniconda.html

Choisissez le bon installateur selon votre système :

Windows : Miniconda3 Windows 64-bit

macOS : Miniconda3 macOS 64-bit (ou Apple Silicon)

Linux : Miniconda3 Linux 64-bit

3. Installation

3.3 Installation (exemple Linux)

- créez un nouveau répertoire nommé « miniconda3 » dans votre répertoire personnel.
  mkdir -p ~/miniconda3

- téléchargez le script d'installation de Linux Miniconda et enregistrez le script sous miniconda.sh dans le répertoire miniconda3.
  wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh

- Exécutez le script d'installation miniconda.sh en mode silencieux à l'aide de bash.
  bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3

- supprimer le fichier du script d'installation miniconda.sh une fois l'installation terminée
rm ~/miniconda3/miniconda.sh

3. Installation

3.3 Installation (exemple Linux)

Après l'installation, fermez et rouvrez votre terminal ou actualisez-la en exécutant la commande suivante :

source ~/miniconda3/bin/activate

Ensuite, initialisez conda sur tous les shells disponibles en exécutant la commande suivante :

conda init –all

Afficher la version de conda

conda --version

3. Gestion des environnements

4.1 Créer un environnement

conda create -n bioinfo_env python=3.10

💡 Vous pouvez aussi ajouter des paquets à la création, ex. :
conda create -n bioinfo_env python=3.10 biopython pandas jupyterlab

Afficher la version de conda
conda --version

4.2 Activer / désactiver un environnement

conda activate bioinfo_env
 # Travailler dans l'environnement
conda deactivate

bash

3. Gestion des environnements

4.3 Lister les environnements existants

conda env list
ou
conda info --envs

4.4 Supprimer un environnement

conda remove -n bioinfo_env --all

4.5 Astuce : garder des environnements légers

- N’installez que ce dont vous avez besoin
- Nettoyez les caches Conda de temps en temps **:**

conda clean --all

3. Gestion des environnements

Exercice pratique 1

Créez un environnement python_env avec Python 3.10.

Installez-y biopython, pandas et jupyterlab.

Activez-le et lancez un notebook Jupyter

Exercice pratique 2

Créez un environnement R_env avec R.4.4.3

Installez-y rstudio, tidyverse et ggplot2.

Activez-le et lancez rstudio

bash

5. Gestion des paquets

5.1 Rechercher un paquet

conda search biopython

5.2 Installer un paquet spécifique depuis un canal (channel)

conda install -c bioconda fastqc

5.3 Mettre à jour un paquet

conda update pandas

Mettre à jour conda lui-même :

conda update conda

5.4 Supprimer un paquet

conda remove biopython

5.5 Afficher les paquets installés dans l'environnement

conda list

bash

5. Gestion des paquets

5.6 Gérer les canaux (channels)

```
conda config --add channels conda-forge
conda config --add channels bioconda
conda config --set channel_priority strict
```

5.7 Exporter / importer un environnement

- **Exporter** un environnement dans un fichier YAML :

```
conda env export > environment.yml
```

- **Recréer** un environnement à partir de ce fichier :

```
conda env create -f environment.yml
```

À retenir

channel_priority strict = environnements plus stables, reproductibles et cohérents.

Exemple de `environment.yml`

yaml

```yaml
name: bioinfo_env
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
  - python=3.10
  - biopython=1.81
  - pandas=1.5.3
  - fastqc=0.11.9
  - pip:
    - some-pip-package==1.0.0
```

5. Gestion des paquets

Exercice pratique

Créer un environnement nommé rnaseq_env

Dans l'environnement rnaseq_env, installez :

- **biopython** et **pandas** depuis **conda-forge**,
- **fastqc** et **multiqc** depuis **bioconda**.

Exportez l'environnement dans un fichier **rnaseq_env_env.yml**

7.3 Privilégiez conda-forge et bioconda
bash

6. Astuces et bonnes pratiques

7.1 Toujours utiliser des environnements isolés

7.2 Gardez vos environnements documentés

7.3 Privilégiez conda-forge et bioconda

7.4 Nettoyez régulièrement

7.5 Testez vos outils juste après installation

7.3 Privilegiez conda-forge et bioconda
bash

7. Conclusion

Ressources utiles

https://docs.conda.io

https://bioconda.github.io/

https://conda.io/projects/conda/en/latest/user-guide/cheatsheet.html