Environment management for scientific computing

A computer used for scientific analysis must be set up for the work, much as a bench is set up for an experiment. An environment manager creates an isolated, fully specified set of software dependencies for each project that can be shared with collaborators or restored years or more later without manual archaeology.
The one-time install takes about ten minutes; creating an environment for a new project adds roughly two minutes. This is the full overhead. Skipping it works until an analysis needs to be revisited after a software update, at which point reconstructing a working environment from scratch typically takes longer than the setup would have.
This protocol uses Conda, a fast, lightweight environment manager widely used in scientific computing, installed via Miniforge using Mamba as the solver. It handles Python and R packages as well as most command-line tools used in bioinformatics and data science in a unified manner.

Reviewed: May 10, 2026

This is a bench card. Full protocol available online.

Procedures

›› *One-time install on a new machine*

(1.) Download the miniforge installer for your OS and CPU architecture from Miniforge, then run it.

```
bash Miniforge3-$(uname)-$(uname -m).sh
```

Critical: Do not run the installer with `sudo`. Conda installs under your user account and never needs administrator rights.

(2.) Open a new terminal. Verify the install and set the channel configuration.

```
mamba init
mamba info
mamba config --add channels bioconda
mamba config --add channels conda-forge
mamba config --set channel_priority strict
mamba install -n base conda-lock
```

›› *Per-project environment workflow*

(1.) Create one environment per project repository, named after the repo. Create it from an existing environment specification file `environment.yml`, or from scratch. Specify the Python version explicitly to avoid silent upgrades when conda-forge advances.

```
mamba env create -n my-project -f environment.yml
mamba activate my-project
```

If no environment file exists yet:

```
mamba create -n my-project python=3.12
mamba activate my-project
```

Critical: Keep the `base` environment empty except for mamba itself. Installing project packages into `base` is the most common path to a fragile installation.

(2.) Install all packages the project needs (Python libraries, R packages, and command-line tools), then export and lock the environment immediately.

```
mamba install numpy pandas scipy matplotlib samtools bwa multiqc
mamba env export --from-history > environment.yml
conda-lock lock -f environment.yml
```

> ***Critical:*** Install command-line bioinformatics tools (`samtools`, `bwa`, `bcftools`, `nextflow`, etc.) via mamba from bioconda, not through Homebrew or apt. System-installed tools are invisible to `environment.yml` and will not travel with the project.

(3.) When you need a new package mid-project, install it, then re-export and re-lock in the same operation.

```
mamba install seaborn scikit-learn
mamba env export --from-history > environment.yml
conda-lock lock -f environment.yml
```

Finding packages

(1.) Search for a package by name from the command line:

```
mamba search numpy
mamba search r-ggplot2
mamba search samtools
```

(2.) If you prefer browsing, use the channel web interfaces.

›› *Sharing and managing environment files*

(1.) When working with Git to track changes SOP 0101, commit both `environment.yml` and the lockfile to the project repository alongside the source code. When you change one, regenerate and commit the other in the same commit.

> ***Critical:*** Updating `environment.yml` without regenerating the lockfile leaves the two artifacts inconsistent: the next collaborator who installs from the lockfile will not see your changes; the next who installs from `environment.yml` will get a different version set than you tested.

(2.) Pin top-level packages with major and minor versions in `environment.yml`; leave patches floating.

(3.) To recreate an environment exactly from a lockfile on a new machine, install from the lockfile rather than the environment file.

```
conda-lock install --name my-project conda-lock.yml
```

(4.) When working with an external repository, apply the same discipline. If forking to modify or extend: adopt the upstream `environment.yml` if one exists (create one if not), then immediately regenerate a lockfile in your fork. If cloning without modification: create the environment from the upstream file as-is.

```
mamba env create -f environment.yml
mamba activate my-project
```

Updating and pruning

(1.) Update mamba periodically. Update only in `base`, not in project environments.

```
mamba update -n base mamba
```

(2.) List existing environments and remove ones for projects no longer worked on.

```
mamba env list
mamba env remove -n old-project
```

(3.) Reclaim disk space from the package cache.

```
mamba clean --all
```

Resources (available online) Troubleshooting (available online) Notes (available online)