datacamp

Conda Cheat Sheet

Learn data science online at www.DataCamp.com

Definitions

Conda is an application for data science package management and environment management. It is primarily used for managing Python packages, though it also supports R, Ruby, Lua, Scala, Java, JavaScript, C, C++, and Fortran packages.

A **package** is a pre-built software package, including code, libraries needed to run the code, and metadata. Conda packages are .conda files or .tar.bz2 files. Popular packages include pandas, seaborn, and r-dplyr.

A **channel** is a location that hosts packages. Popular channels include conda-forge, bioconda, and intel.

An **environment** is a directory containing installed packages. Having multiple environments lets you use different versions of packages for different projects.

anaconda is a meta-package that allows you to install hundreds of data science packages with a single conda command.

Mamba is an open source reimplementation of Conda with the same syntax and some performance improvements. The majority of the code in this cheat sheet will also work with Mamba.

Conda vs. Pip

Feature	Pip	Conda
Package management	✅	✅
Environment management	❌	✅
Language support	Python only	Many data languages
Installation	Included with Python	Separate install
License	MIT	BSD

Getting help

```
# Display Conda version with conda --version
conda --version

# Display Conda system info with conda info
conda info

# Get help on Conda with conda --help
docker --help

# Get help on Conda command usage with conda {command} --help
docker build --help
```

Listing packages

```
# List all installed packages with conda list
conda list

# List installed packages matching a regular expression with conda list {regex}
conda list ^z # lists packages with names starting with z

# List all versions of all packages in all conda channels with conda search
conda search

# List all versions of a package (all channels) with conda search {pkg}
conda search scikit-learn

# List versions of a package (all channels) with conda search '{pkg}{version}'
conda search 'scikit-learn>=1'

# List package versions for a conda channel with conda search {channel}::{pkg}
conda search conda-forge::scikit-learn
```

Installing & managing packages

```
# Install packages with conda install {pkg1} {pkg2} ...
conda install numpy pandas

# Install specific version of package with conda install {pkg}={version}
conda install scipy=1.10.1

# Update all packages with conda update --all
conda update --all

# Uninstall packages with conda uninstall {pkg}
conda uninstall pycaret
```

Working with channels

```
# List channels with conda config --show channels
conda config --show channels

# Add a channel (highest priority) with conda config --prepend channels {channel}
conda config --prepend channels conda-forge

# Add a channel (lowest priority) with conda config --append channels {channel}
conda config --append channels bioconda
```

Working with environments

```
# List environments with conda env list
conda env list

# Restrict command scope to an environment by appending --name {envname}
conda list --name base
conda install scikit-learn --name myenv
```

Managing environments

```
# Create an environment for technology with conda create -n {env}
conda create --name my_python_env

# Clone existing environment with conda create --clone {old_env} --name {new_env}
conda create --clone template_env --name project_env

# Create environment, auto-accepting prompts with conda create --yes --name {env}
# For non-interactive usage
conda create --yes --name my_env

# Make environment the default environment with conda activate {env}
# This prepends the environment directory to the system PATH environment variable
conda activate my_env

# Make the base environment the default with conda deactivate {env}
# This removes the environment directory from system PATH environment variable
conda deactivate my_env
```

Sharing environments

```
# Export active environment to a YAML file with conda env export > environment.yml
# Export every package including dependencies (maximum reproducibility)
conda env export > environment.yml
# Export only packages explicitly asked for (increased portability)
conda env export --from-history > environment.yml

# Import environment from YAML with conda env create --name {env} --file {yaml_file}
conda env create --name my_env2 --file environment.yml

# Export list of packages to TEXT file with conda list --export > requirements.txt
# Usually requires manual editing; you can also use pip freeze
conda list --export > requirements.txt

# Import environment from TEXT file with conda create --name {env} --file {yaml_file}
conda create --name my_env --file  requirements.txt
```

Learn Data Science Online at
www.DataCamp.com

datacamp