



Applied Predictive Analytics

# Infrastructure

S. Lessmann, N. Kozodoi, A. Zharova



Conda

Environment and package management for Python

# Overview

## Packages and environments

### ■ Packages

- Python heavily relies on external packages (e.g., numpy, pandas, scikit-learn, ...)
- many packages are under active development and are regularly updated

### ■ Dependencies

- most packages rely on dependencies (other packages) to implement their functionality
- different package versions may require different dependency versions

### ■ Environments

- environment is a (virtual) machine that has a set of specific packages installed
- single computer can have multiple environments

# Overview

## Why we need environment/package management?

### ■ Common issues during code development

- projects require different Python and/or package versions
- packages require dependencies with conflicting versions
- code that was working yesterday fails today
- code produces different results on different machines

### ■ Environment and package managers helps with these issues

- ability to set up environments with custom Python / package versions for different projects
- resolving conflicts between dependencies
- making projects self-contained and reproducible
- all dependencies can be written in a single requirements file for sharing



# Overview

## What is Conda?

- Free and open source package and environment management system
- Easy to use with command line on Windows / macOS / Linux
- Supports multiple programming languages



# Background

## Integrated development environments (IDEs)

### ■ **Web-based IDEs:** notebook environments connected to a local/cloud machine

- e.g., Google Colab, Kaggle Notebooks, Amazon Sagemaker
- externally curated set of pre-installed packages
  - Python and package versions are managed externally
  - still need to install missing packages manually
  - using Conda is difficult (but possible)



### ■ **Software-based IDEs:** programming software installed on a local/cloud machine

- e.g., jupyterlab, Visual Studio Code, PyCharm
- using Conda enables custom environments with desired packages
  - ability to specify package versions yourself
  - different environments for different projects



# Background

## Integrated development environments (IDEs)

### ■ **Web-based IDEs:** notebook environments connected to a local/cloud machine

- e.g., Google Colab, Kaggle Notebooks, Amazon Sagemaker
- externally curated set of pre-installed packages
  - Python and package versions are managed externally
  - still need to install missing packages manually
  - using Conda is difficult (but possible)



- **good fit for short-term one-person projects in a single notebook**

### ■ **Software-based IDEs:** programming software installed on a local/cloud machine

- e.g., jupyterlab, Visual Studio Code, PyCharm
- using Conda enables custom environments with desired packages
  - ability to specify package versions yourself
  - different environments for different projects



- **good fit for collaborative development with multiple notebook/script files**

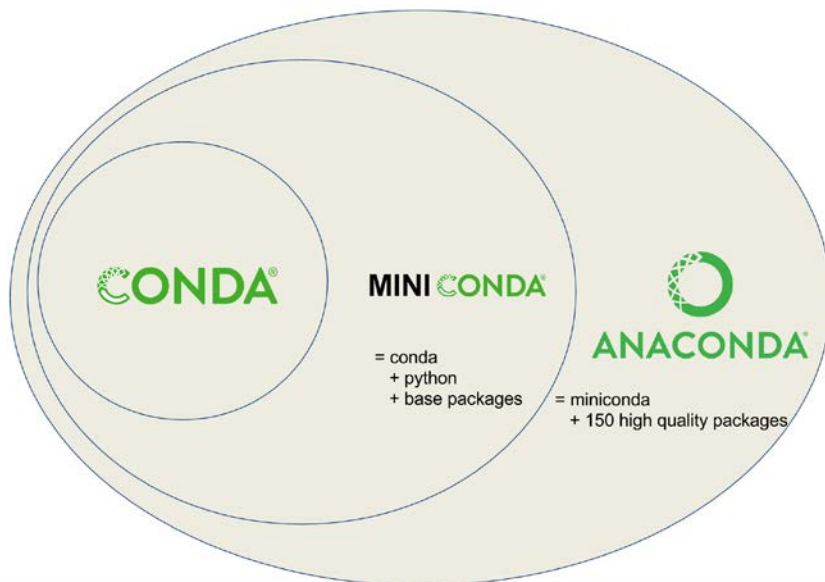
# Installing Conda

## ■ Download & install the latest version

- Miniconda: <https://docs.conda.io/en/latest/miniconda.html>
- Anaconda: <https://www.anaconda.com/download/>

## ■ Miniconda vs anaconda

- Anaconda has popular packages pre-installed (3 Gb)
- Miniconda starts from scratch and lets you install everything manually



## Windows installers

Python version	Name	Size
Python 3.9	Miniconda3 Windows 64-bit	57.7 MiB
	Miniconda3 Windows 32-bit	54.9 MiB
Python 3.8	Miniconda3 Windows 64-bit	57.0 MiB
	Miniconda3 Windows 32-bit	54.2 MiB
Python 2.7	Miniconda2 Windows 64-bit	54.1 MiB
	Miniconda2 Windows 32-bit	47.7 MiB

## MacOSX installers

Python version	Name	Size
Python 3.9	Miniconda3 MacOSX 64-bit bash	42.2 MiB
	Miniconda3 MacOSX 64-bit pkg	49.7 MiB
Python 3.8	Miniconda3 MacOSX 64-bit bash	54.5 MiB
	Miniconda3 MacOSX 64-bit pkg	62.0 MiB
Python 2.7	Miniconda2 MacOSX 64-bit bash	40.3 MiB
	Miniconda2 MacOSX 64-bit pkg	48.4 MiB



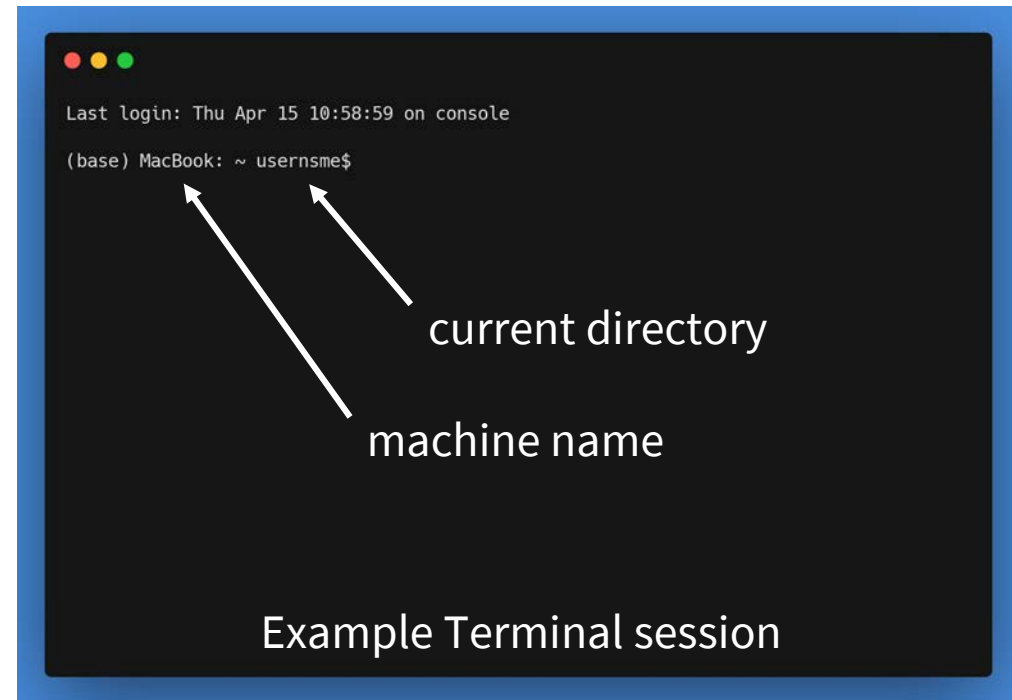
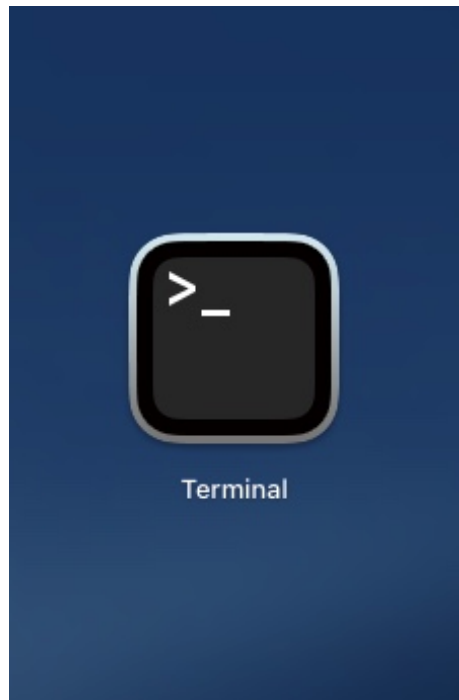
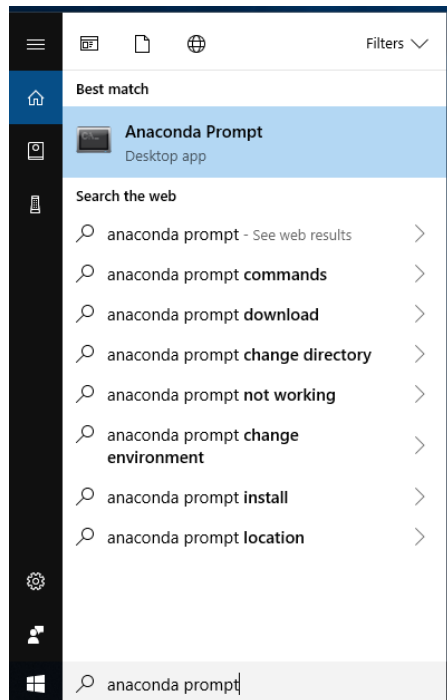
# Working in Conda

## Basics

### ■ Launching a command line

- Windows: Anaconda prompt (installed with Conda)
- macOS / Linux: Terminal (pre-installed with OS)

### ■ Using commands to interact with Conda



# Working in Conda

## Managing Conda

### ■ Check conda version

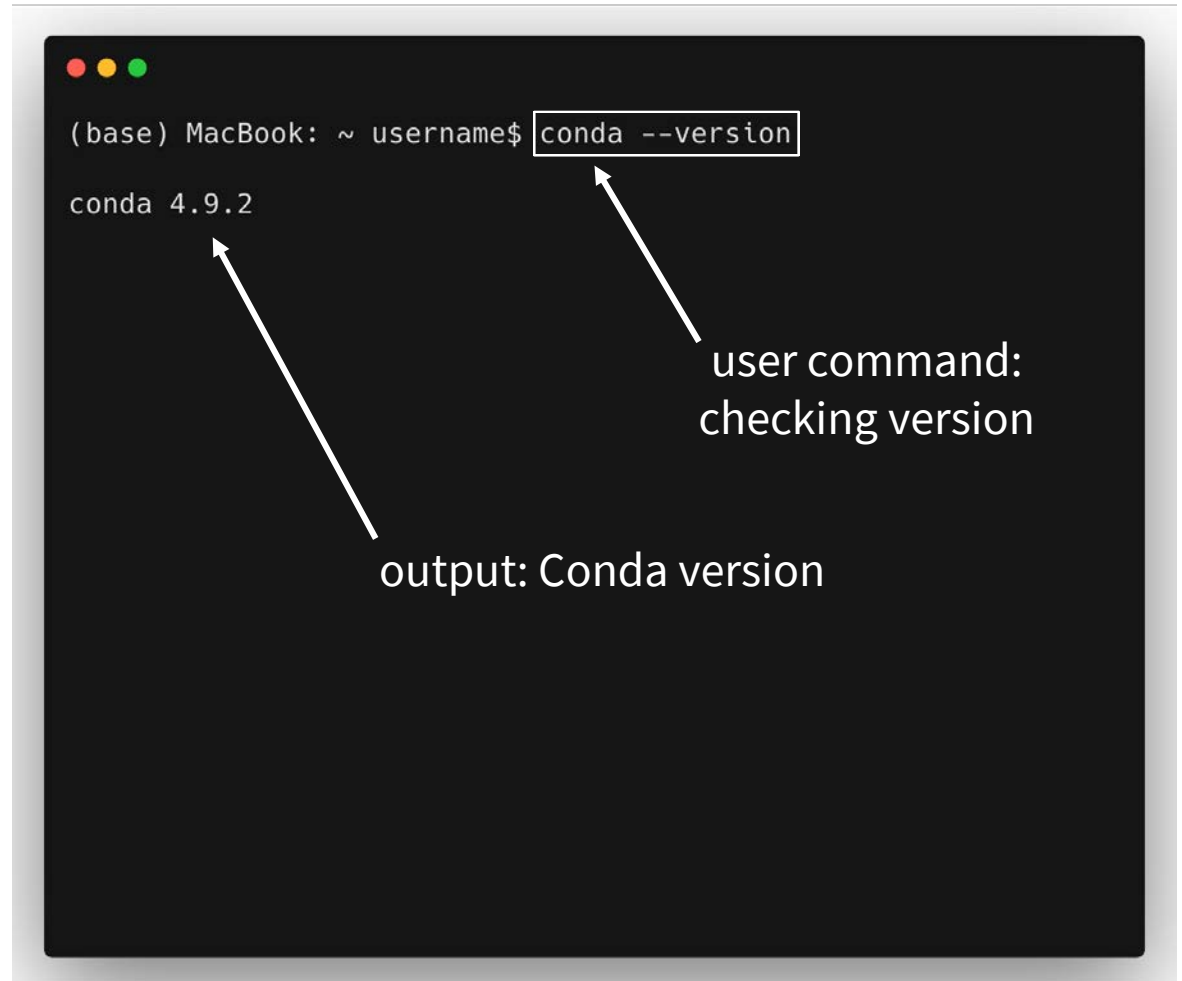
- ❑ `$ conda --version`

### ■ Update conda

- ❑ `$ conda update conda`
- ❑ `Proceed ([y]/n)? y`

### ■ Uninstall conda

- ❑ `$ conda install anaconda-clean`
- ❑ `$ anaconda-clean --yes`



```
(base) MacBook: ~ username$ conda --version
conda 4.9.2
```

user command: checking version

output: Conda version

# Working in Conda

## Managing environments

### ■ Create new environment

□ `$ conda create --name <env_name> python=3.9 pandas numpy`

□ Proceed ([y]/n)? y

Environment  
name

Python  
version

Space-separated  
list of packages

### ■ See list of environments

□ `$ conda info -envs`

### ■ Delete environment

□ `$ conda remove --name env_name --all`

□ Proceed ([y]/n)? y

# Working in Conda

## Switching environments

### ■ Environment selection


- base environment is active by default
- user can switch between environments
- check `(env_name)` indication in Terminal

### ■ Activate existing environment

- `$ conda activate env_name`

### ■ Return to base environment

- `$ conda deactivate`



```
(base) MacBook: ~ username$  
  
activating "apa" environment  
  
(base) MacBook: ~ username$ conda activate apa  
(apa) MacBook: ~ username$
```

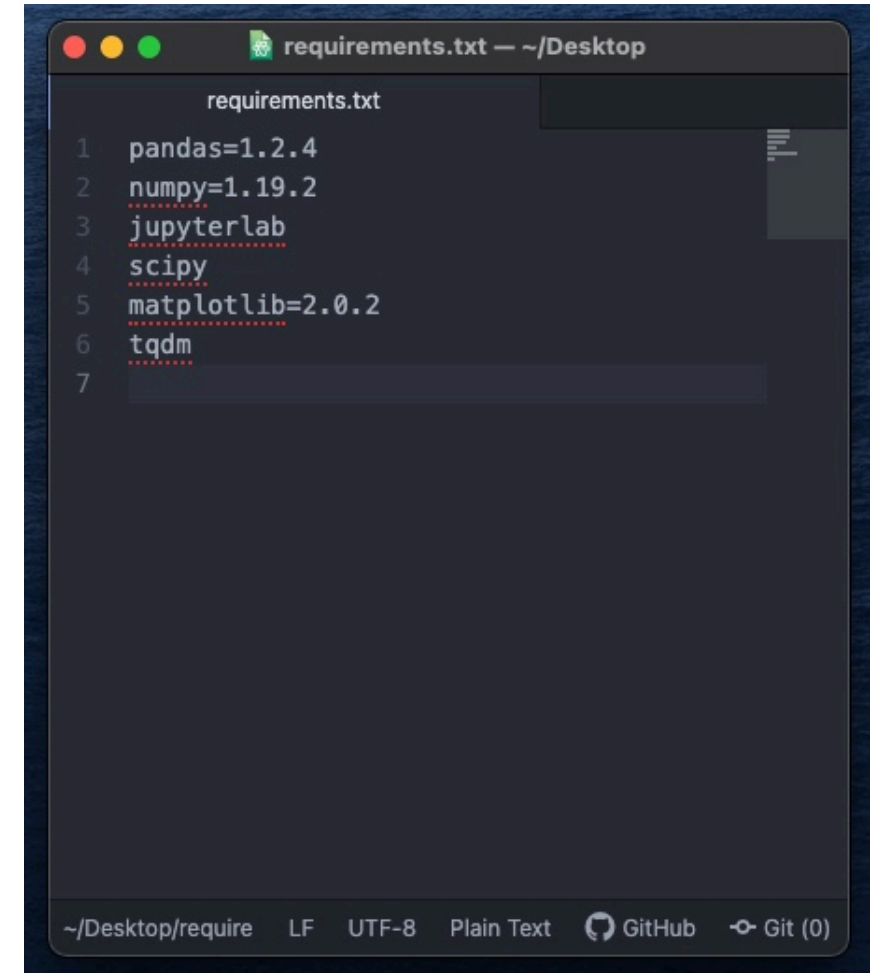
The terminal screenshot illustrates the process of switching Conda environments. The initial prompt is `(base) MacBook: ~ username$`, where `(base)` is the current environment indicator. An arrow points to this text with the label "current environment indicator". Below this, the command `conda activate apa` is entered. The subsequent prompt is `(apa) MacBook: ~ username$`, where `(apa)` is the updated environment indicator. An arrow points to this text with the label "updated environment indicator". The text "activating 'apa' environment" is placed between the two prompts, with an arrow pointing to the `conda activate apa` command.

# Working in Conda

## Managing packages [1/2]

- **Make sure the desired environment is active**
- **Check what packages are installed**
  - `$ conda list`
- **Search for available package versions**
  - `$ conda search package_name`
- **Install package(s)**
  - `$ conda install package_name`
  - `$ conda install package_name=version`
- **Install package(s) from file**
  - `$ conda install --file requirements.txt`

Example requirements.txt



```
requirements.txt
1 pandas=1.2.4
2 numpy=1.19.2
3 jupyterlab
4 scipy
5 matplotlib=2.0.2
6 tqdm
7
```

~/Desktop/require LF UTF-8 Plain Text GitHub Git (0)



# Working in Conda

## Managing packages [2/2]

### ■ What if a package is not available in Conda?

- i.e., `$ conda search package_name` does not display desired package/version
- use Google to find the package GitHub page or documentation
- look for “installation” to find instructions

### ■ Most common ways to install a package (from more to less preferred)

- install with Conda (official Conda repository):
  - `$ conda install package_name`
- install with Conda-forge (community-led Conda repositories)
  - `$ conda install -c conda-forge package_name`
- install with pip (standard Python package manager)
  - `$ pip install package_name`

# Working in Conda

## Typical Conda workflow with Jupyter

### ■ Launch Terminal / Anaconda Prompt

### ■ Create environment with relevant packages [first time only]

- `$ conda create --name apa python=3.9 jupyterlab pandas numpy`

### ■ Activate environment

- `$ conda activate apa`

### ■ Launch Jupyter from command line

- `$ jupyter lab`

### ■ Close Jupyter with a command line shortcut

- `ctrl + c`

- `Proceed ([y]/n)? y`

### ■ Deactivate environment and quit Terminal

# Working in Conda

Typical Conda workflow with Jupyter



# Demo

# Working in Conda

## Summary

### ■ Conda is a powerful environment / package manager

- set up environments with custom Python / package versions
- helps to make projects self-contained and reproducible
- all dependencies can be written in a single requirements file for sharing

### ■ Conda is simple to install and use

- supported on Windows / macOS / Linux
- knowing ~10 Conda commands is enough for most applications
- more advanced functionality is available (see documentation)

### ■ More materials

- official documentation: <https://conda.io/projects/conda/en/latest/index.html>
- PDF cheat sheet: <https://conda.io/projects/conda/en/latest/user-guide/cheatsheet.html>

# Thank you for your attention!

Stefan Lessmann

Chair of Information Systems  
School of Business and Economics  
Humboldt-University of Berlin, Germany

Tel. +49.30.2093.99540

Fax. +49.30.2093.99541

[stefan.lessmann@hu-berlin.de](mailto:stefan.lessmann@hu-berlin.de)

<http://bit.ly/hu-wi>

[www.hu-berlin.de](http://www.hu-berlin.de)

