Lesson 0: Preparing Your Computer for the Course

Aaron A. King   Edward L. Ionides   Kunyang He

Introduction

- This course uses state-of-the-art tools and some care may be needed to set up the computational environment.
- We use `pypomp`, a Python implementation of partially observed Markov process (POMP) models.
- `pypomp` relies on `jax` for vectorization, just-in-time compilation, GPU accessibility, and automatic differentiation.
- `jax` is a high-performance compting package for Python. It depends on
  - CUDA, the NVIDIA GPU programming platform
  - XLA, an accelerated linear algebra library for CUDA
- Additionally, the course emphasizes reproducible data analysis using Quarto

What you will need

To get started:

- Python 3.10 or higher.
- Basic familiarity with Python programming
- A text editor or IDE (e.g., VS Code, PyCharm, Jupyter, Positron). The notes are in Quarto (qmd) format. VS Code (with the Quarto extension) and Positron are recommended for working with qmd files.
- If you run Windows, the WSL2 Linux virtual machine is recommended for `jax`.

Python Installation

First, check if you have Python installed and verify the version:

```
python --version
```

or

```
python3 --version
```

You should have Python 3.10 or higher. If not, make a fresh installation. Various Python distributions exist, but we recommend using the official Python distribution:

1. Download Python from https://www.python.org/downloads/
2. Follow the installation instructions for your operating system
3. Make sure to check "Add Python to PATH" during installation (Windows)

Setting Up a Virtual Environment

It is highly recommended to use a virtual environment to avoid conflicts with other Python projects. We use the built-in `venv`

```
# Create a virtual environment in the
# working directory for your project
python -m venv .venv

# Activate the environment
source .venv/bin/activate
```

Installing Required Packages

The main package we will use is `pypomp`. Install it using `pip`:

```
pip install pypomp
```

This will also install various dependencies, including `jax`, `numpy`, `pandas`, `scipy`, `matplotlib`.

The following additional packages are commonly used in the course:

```
pip install jupyter
```

Optional: GPU acceleration with JAX

For faster computation, especially for large models, you can install JAX for GPU/TPU acceleration:

```
# For CPU-only (default)
pip install jax jaxlib

# For GPU support (CUDA)
# See https://github.com/google/jax#installation for detai
pip install jax[cuda12]
```

Optional: Additional plotting libraries

For enhanced visualizations:

```
pip install seaborn plotly
```

Setting Up Jupyter

Jupyter provides an alternative to Quarto for running Python code.

Installing Jupyter

If you haven't already installed Jupyter:

```
pip install jupyter notebook jupyterlab
```

Launching Jupyter

To start Jupyter Notebook:

```
jupyter notebook
```

To start JupyterLab (more modern interface):

```
jupyter lab
```

Registering your virtual environment as a Jupyter kernel

If you created a virtual environment, register it with Jupyter:

```
# Make sure your virtual environment is activated
pip install ipykernel
python -m ipykernel install --user --name=sbied-env --displ
```

Verifying Your Installation

Test basic Python functionality

Create a test Python script or run in a Python interpreter:

```python
import sys
print(f"Python version: {sys.version}")

import numpy as np
import scipy
import matplotlib.pyplot as plt
import pandas as pd

print("All basic packages imported successfully!")
```

Test pypomp installation

Create a simple test to verify pypomp is working:

```python
import pypomp

print(f"pypomp version: {pypomp.__version__}")
print("pypomp installed successfully!")
```

Test with a minimal example

Run this minimal POMP model to ensure everything is working:

```
import numpy as np
import pypomp as pp

# Create a simple random walk model
def step_fn(x, t, params, **kwargs):
    return {'X': x['X'] + np.random.normal(0, params['sigma

def rinit_fn(params, **kwargs):
    return {'X': 0.0}

def rmeas_fn(x, t, params, **kwargs):
    return {'Y': x['X'] + np.random.normal(0, params['tau']

def dmeas_fn(y, x, t, params, log=True, **kwargs):
    ll = -0.5 * np.log(2 * np.pi * params['tau']**2) - \
         0.5 * (y['Y'] - x['X'])**2 / params['tau']**2
    return ll if log else np.exp(ll)
```

Downloading Course Materials

Using Git (recommended)

If you have Git installed:

```
git clone https://github.com/[repository-url]/sbied-pypomp
cd sbied-pypomp
```

Direct download

Alternatively, download the course materials as a ZIP file from the course website and extract them to your working directory.

Troubleshooting: Common issues and solutions

"pip: command not found"

- Make sure Python is installed and added to your PATH
- Try using `python -m pip` instead of `pip`

"Permission denied" errors

- On macOS/Linux, try using `pip install --user package_name`
- Or use a virtual environment (recommended)

Import errors

- Make sure your virtual environment is activated
- Verify package installation: `pip list | grep package_name`
- Try reinstalling: `pip uninstall package_name && pip install package_name`

Jupyter kernel issues

- Make sure ipykernel is installed in your environment
- Re-register the kernel (see "Registering your virtual

Getting help

If you encounter issues:

1. Check the pypomp documentation: https://github.com/pypomp/pypomp
2. Search for error messages online
3. Ask for help during the course sessions
4. Check the course discussion forum

Summary

By now, you should have:

- Python 3.8+ installed
- A virtual environment created and activated
- pypomp and required packages installed
- Quarto and/or Jupyter set up
- Verified your installation with test code

You are now ready to begin the course!

Additional Resources

Additional Resources

- **Python Tutorial**: https://docs.python.org/3/tutorial/
- **NumPy Documentation**: https://numpy.org/doc/
- **Matplotlib Tutorial**: https://matplotlib.org/stable/tutorials/index.html
- **Jupyter Documentation**: https://jupyter.org/documentation
- **pypomp Repository**: https://github.com/pypomp/pypomp
- **JAX Documentation**: https://docs.jax.dev