ECOG 315 / ECON 181, Summer 2025
Advanced Research Methods and Statistical Programming
Week 4 Lecture Slides

Matthew N. White

Howard University

June 20, 2025

Week 4 Administrivia

- If you did not successfully issue a PR for your survey, please do that
  - Ronald!
- Email me if you need help with GitHub (etc)
- Did the revised example Python code work for Mac users?
- Alan and I were pretty impressed with your presentations

Workflow Management: Branches on GitHub (1/2)

- Quite often, there is **a lot** going on with a project
- Some things might not work out, but want to experiment with them
- And/or multiple people are working on the project

Workflow Management: Branches on GitHub (1/2)

- Quite often, there is **a lot** going on with a project
- Some things might not work out, but want to experiment with them
- And/or multiple people are working on the project
- Can organize git workflow in **branches** of your repo
- Split off a mini-copy of the project, make commits, maybe merge into `main`

Workflow Management: Branches on GitHub (1/2)

- Quite often, there is **a lot** going on with a project
- Some things might not work out, but want to experiment with them
- And/or multiple people are working on the project
- Can organize git workflow in **branches** of your repo
- Split off a mini-copy of the project, make commits, maybe merge into `main`
- Merging happens via pull request, just like when making a PR from your fork
- Branches are cheap as free: if something doesn't work out, just toss it!

Workflow Management: Branches on GitHub (1/2)

- Quite often, there is **a lot** going on with a project
- Some things might not work out, but want to experiment with them
- And/or multiple people are working on the project
- Can organize git workflow in **branches** of your repo
- Split off a mini-copy of the project, make commits, maybe merge into `main`
- Merging happens via pull request, just like when making a PR from your fork
- Branches are cheap as free: if something doesn't work out, just toss it!
- Mature project could have a lot of branches (and PRs!)

Workflow Management: Branches on GitHub (2/2)

- On GitHub desktop, click the arrow on the “Current branch” button
- Will see a list of all branches; can click “New branch” and name it

Workflow Management: Branches on GitHub (2/2)

- On GitHub desktop, click the arrow on the "Current branch" button
- Will see a list of all branches; can click "New branch" and name it
- Commits are specific **to a branch**

Workflow Management: Branches on GitHub (2/2)

- On GitHub desktop, click the arrow on the "Current branch" button
- Will see a list of all branches; can click "New branch" and name it
- Commits are specific **to a branch**
- GitHub Desktop can "stash" one set of uncommitted changes
- But best practice is to commit (or discard) changes before changing branches

Workflow Management: Branches on GitHub (2/2)

- On GitHub desktop, click the arrow on the "Current branch" button
- Will see a list of all branches; can click "New branch" and name it
- Commits are specific **to a branch**
- GitHub Desktop can "stash" one set of uncommitted changes
- But best practice is to commit (or discard) changes before changing branches
- Don't forget to push back to the origin after committing!
- When issuing a PR, verify the branch you're merging and the target

Anaconda and Virtual Environments

- There are a lot of Python packages, and you will have multiple projects
- Project A requires `CoolPackage` v0.8 or higher; Project B requires v0.7 or lower

Anaconda and Virtual Environments

- There are a lot of Python packages, and you will have multiple projects
- Project A requires `CoolPackage` v0.8 or higher; Project B requires v0.7 or lower
- Not a problem! `conda` can manage virtual Python environments
- Can have multiple collections/configurations of packages on your computer
- Manage each separately, switch between them with a single command

Anaconda and Virtual Environments

- There are a lot of Python packages, and you will have multiple projects
- Project A requires `CoolPackage` v0.8 or higher; Project B requires v0.7 or lower
- Not a problem! `conda` can manage virtual Python environments
- Can have multiple collections/configurations of packages on your computer
- Manage each separately, switch between them with a single command
- We probably won't use this much, but will install more packages
- Open up Anaconda Prompt (from Windows start menu, e.g.); terminal will open

Installing New Packages

- Example: installing `HARK`, Econ-ARK's primary software package
- In Anaconda prompt, type `pip install econ-ark`, accept

Installing New Packages

- Example: installing `HARK`, Econ-ARK's primary software package
- In Anaconda prompt, type `pip install econ-ark`, accept
- `pip` is the most common **package manager**, draws on Python Package Index
- `conda` is also a package manager, but **slightly** fussier

Installing New Packages

- Example: installing `HARK`, Econ-ARK's primary software package
- In Anaconda prompt, type `pip install econ-ark`, accept
- `pip` is the most common **package manager**, draws on Python Package Index
- `conda` is also a package manager, but **slightly** fussier
- `HARK` is now available for use on your computer
- But we won't be working with it yet

Introduction to Jupyter Notebooks

- Communicating scientific ideas is sometimes aided by reader interaction
- **Jupyter notebooks** provide a way to integrate text, math, and code

Introduction to Jupyter Notebooks

- Communicating scientific ideas is sometimes aided by reader interaction
- **Jupyter notebooks** provide a way to integrate text, math, and code
- Sync your fork with the course repo, and pull down from remote

Introduction to Jupyter Notebooks

- Communicating scientific ideas is sometimes aided by reader interaction
- **Jupyter notebooks** provide a way to integrate text, math, and code
- Sync your fork with the course repo, and pull down from remote
- In Anaconda prompt, type `jupyter notebook`
- Navigate to your fork on your computer, then go to `/code/`
- Open `ExampleNotebook.ipynb`

Introduction to Jupyter Notebooks

- Communicating scientific ideas is sometimes aided by reader interaction
- **Jupyter notebooks** provide a way to integrate text, math, and code
- Sync your fork with the course repo, and pull down from remote
- In Anaconda prompt, type `jupyter notebook`
- Navigate to your fork on your computer, then go to `/code/`
- Open `ExampleNotebook.ipynb`
- Cells can be run individually with Shift+Enter

What Should You Use Jupyter Notebooks For?

- Notebooks are very good at conveying **mixed information**
- E.g. want to explain a model **and** its computational representation

What Should You Use Jupyter Notebooks For?

- Notebooks are very good at conveying **mixed information**
- E.g. want to explain a model **and** its computational representation
- Or document a code feature and explain it with examples
- Or have **interactive figures** to show model interactions (`ipywidgets`)

What Should You Use Jupyter Notebooks For?

- Notebooks are very good at conveying **mixed information**
- E.g. want to explain a model **and** its computational representation
- Or document a code feature and explain it with examples
- Or have **interactive figures** to show model interactions (`ipywidgets`)
- Avoid: "This notebook could have been a PDF"

What Should You Use Jupyter Notebooks For?

- Notebooks are very good at conveying **mixed information**
- E.g. want to explain a model **and** its computational representation
- Or document a code feature and explain it with examples
- Or have **interactive figures** to show model interactions (`ipywidgets`)
- Avoid: "This notebook could have been a PDF"
- sGood example: `TractableBufferStock-Interactive.ipynb`