Quick Start Guide for Sharing Jupyter Notebooks

www.metabolomicsworkbench.org

Christine Kirkpatrick[1], Kevin Coakley[1], Manish Sud[1], Eoin Fahy[1],and Shankar Subramaniam[1,2]

NIH Grant: U2C- DK119886;https://commonfund.nih.gov/metabolomics/index

Developing Jupyter Notebooks

The Anaconda distribution is recommended for developing Jupyter Notebooks to be shared. Anaconda is a popular Open Source distribution for Python and R that is used by Researchers across many scientific domains. Anaconda improves reproducibility by including tools to export the working environment, which allows other researchers and services, like Binder, to easily replicate the Author's working environment. Anaconda can be installed on Windows, macOS and Linux workstations without needing privileged access. Anaconda can be downloaded for free at https://www.anaconda.com/download .

Preparing Jupyter Notebooks

Jupyter Notebooks should be clear, concise and have results that are reproducible. Below are guidelines for creating well written Jupyter Notebooks:

1. Document the workflow and code.
   a. Create Markdown cells to document whole cells or groups of cells.
   b. Use inline code comments to document complex algorithms and methods.
2. Do not include shell commands in cells unless the Jupyter Notebook will be distributed in a Docker container.
3. Keep the cell output if other researchers will use the included data and code to replicate the output.
4. Clear the cell output if other researchers will use their own data or modify the code to create their own original output.
5. Any information that isn't pertinent to the running of the Jupyter Notebook should be included in a README file and not in the Jupyter Notebook. The exception to this is Author and Grant information.

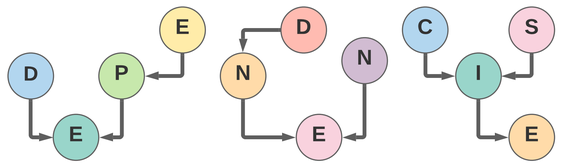
Jupyter Notebook Portability

Create a New Environment

Create a new sterile environment that isn't polluted with dependencies from prior projects when developing a new Jupyter Notebook project. Reusing the same environment can cause package version conflicts between projects, causing previously working Jupyter Notebooks to fail.

.

Managing Dependencies



The most important part of maintaining portability is managing dependencies. There are two parts to managing dependencies, identifying installed packages and identifying custom external libraries and datasets.

Package Dependencies

To ensure reproducibility include version numbers with all included packages. If version numbers are not specified then the latest version of the package will be installed. Algorithms can change and APIs can be deprecated when developers update their packages. Jupyter Notebooks can fail to run or return different results when package versions are different from when they were last tested.

Use the conda package manager to install packages in order to simplify package management when creating portable Jupyter Notebooks. The conda package manager has a large selection of packages for Python and R, however conda doesn't contain the complete selection of packages that are available. If the package or version isn't found then use the PIP or CRAN package repositories, try to avoid installing packages from their source code.

Custom External Library and Dataset Dependencies

Custom external libraries and datasets should be included in the root directory or a subdirectory. All references in the Jupyter Notebook to the custom external libraries and datasets should be done using paths relative to the root directory.

.

Large Datasets

Be mindful of large datasets. If the dataset is over 500 MB, include the smallest representative subset of the data with the Jupyter Notebook. Links to the full dataset can be added to a README file included with the Jupyter Notebook.

Sharing Jupyter Notebooks on GitHub

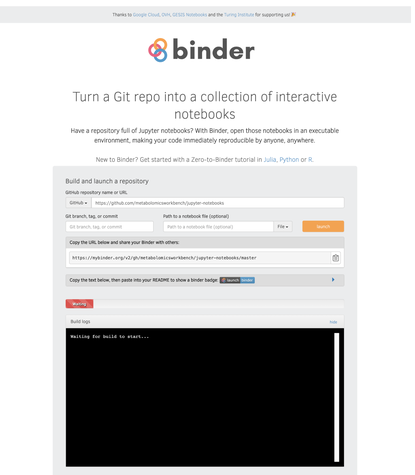
GitHub is a service for public software development and version control based around the git version control system. GitHub is the most popular service for sharing Jupyter Notebooks despite GitHub's focus on software developers. GitHub's web interface makes it easy for novice developers to share their software without having to learn the git version control system. Use the following steps to share your Jupyter Notebooks on GitHub.

1. Create a GitHub account by going to https://github.com/join.
2. Create new public GitHub repository.
   a. See instructions: https://tinyurl.com/create-repo
3. Drag and drop all of the Jupyter Notebook project to upload to GitHub.
   a. See Instructions: https://tinyurl.com/add-file-repo
   b. Ignore the recommendation to create a new branch and commit directly to the master branch.
4. View the publicly available Jupyter Notebook project at http://github.com/username/repo-name/.

Launching Jupyter Notebooks with MyBinder

The Binder service is a popular free service for sharing Jupyter Notebooks that allows anyone to run Jupyter Notebooks from their web browser without having to install any software. The Binder service works by creating a Docker container to run any Jupyter Notebooks that are available to publicly download, like on GitHub. Researchers who want to update these Jupyter Notebooks can either update them directly on the Binder service or they can run the Jupyter Notebook with Anaconda on their own workstation by cloning the GitHub repository.

http://mybinder.org is the most popular free Binder service. Binders can be created by entering the GitHub repository URL into the Repository URL textbox and pressing the Launch button. It can take a few minutes for the Jupyter Notebook environment to start since a new environment is built every time the Jupyter Notebook project is launched, the build progress can be viewed in the Build Logs window.



[1]San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92037 [2]Departments of Bioengineering, Computer Science and Engineering, Cellular and Molecular Medicine, and Chemistry and Biochemistry, University of California, San Diego, La Jolla, CA 92093