Econometrics Pedagogy and Cloud Computing: Training the Next Generation of Economists and Data Scientists

Danielle V. Handel* Anson T. Y. Ho† Kim P. Huynh‡ David T. Jacho-Chávez§
Carson H. Rea¶

October 6, 2020

Abstract

This paper describes how cloud computing tools widely used in the instruction of data scientists can be introduced and taught to economics students as part of their curriculum. The demonstration centers around a workflow where the instructor creates a virtual server and the students only need internet access and a web browser to complete in-class tutorials, assignments, or exams. Given how prevalent cloud computing platforms are becoming for data science, introducing these techniques into students' econometrics training would prepare them to be more competitive when job hunting, while making instructors and administrators re-think what a computer laboratory means on campus.

Keywords: Virtual Online Learning, Jupyter, GitHub, Python, R, Stata

JEL classification: A11, A22, A23, C87, C88

*Department of Economics, Emory University, Rich Building 306, 1602 Fishburne Dr., Atlanta, GA 30322-2240, USA. E-mail: dvhande@emory.edu

†Ted Rogers School of Management, Ryerson University, 55 Dundas Street West, Toronto ON, M5G 2C3, Canada. E-mail: atyho@ryerson.ca

‡Corresponding Author: Bank of Canada, 234 Wellington Ave., Ottawa ON, K1A 0G9, Canada. E-mail: kim@huynh.tv

§Department of Economics, Emory University, Rich Building 306, 1602 Fishburne Dr., Atlanta,GA 30322-2240, USA. E-mail: djachocha@emory.edu

¶Department of Economics, Emory University, Rich Building 306, 1602 Fishburne Dr., Atlanta, GA 30322-2240, USA. E-mail: chrea@emory.edu

> "*For what greater or better service can I render to the commonwealth than to instruct and train the youth.*"
>
> -Marcus Tullius Cicero, De Divinatione Book 2, page 375 translated by W.A. Falconer and transcribed by Bill Thayer.

1 Introduction

As cloud computing services are becoming mainstream and the tool of choice of many private companies, universities, and government agencies, there is an industry need for students in economics to acquire the associated skills earlier on in their undergraduate training in higher education institutions. This paper describes how industry standards such as version control, cloud computing services, as well as open-source web applications for data science can be taught in most undergraduate programs in economics. This framework re-defines the 'computer lab' component in econometric instruction from the traditional idea of a physical space on campus, where students meet synchronously, to a 'virtual lab' with servers hosted on the internet (also known as *the cloud*) that students and instructors can connect from anywhere. Using their own personal computing equipment, students and instructors can simulate a lab setting during a synchronous lecture on a video conference platform, or connect to the cloud service asynchronously across different time zones to complete and submit assignments.

This paper demonstrates a specific workflow that instructors, currently teaching in the traditional computer lab setting, can adopt to teach the same material remotely either in a synchronous or asynchronous setting. The suggested workflow requires the instructor to set up a server on the cloud using one of the many service vendors available, install the necessary software on this server that would otherwise need to be installed in one and each of the computers in the traditional computer lab, and provide access to this resource to the entire class. It is shown how the instructor can distribute, collect, and grade assignments on the cloud without ever having a physical copy of them.

The remainder of this article is organized as follows: Section 2 familiarizes the casual reader with both the software and instructional tools commonly used in undergraduate and graduate training in economics and data science. Section 3 discusses three potential workflows for instructors and outlines the benefits of implementing a centralized cloud computing platform. Section 4 walks the reader through the steps of setting up an assignment using version control, have it completed on the cloud, and submitted by the students for grading. Section 5 concludes.

2 Software and the Cloud

In this section we describe the most common software used for econometrics and data science as well as the most common cloud environments currently used by practitioners in higher education, government, and industry. By introducing these industry standards in the empirical training of students in economics, they are better prepared to join the workforce upon graduation and face better job prospects than other majors, see Athey and Luca (2019). Table 1 provides a glossary of terms used throughout this and subsequent sections.

Git, GitHub, and GitHub Classroom

Version control systems give users the opportunity to track changes along iterations of code (or other documents) and provide a working revision history. Git, a widely used distributed version control system, see, e.g., Bryan (2018), is compatible for use with data science applications because it correctly compiles Python notebooks (see below) without need for a third party. Git also recognizes markdown,

a language that formats plain text. Markdown text can be used in Jupyter notebooks to create notes or assignments that include text, code, and mathematical notation. Git is most useful when paired with GitHub, a centralized cloud service that hosts Git repositories (folders to store code and other project files). Users can clone and download repositories for work on their local systems. Similarly, users may create "branches" of their repository for editing code without changing the original files. They may later push their changes to the master branch, allowing for non-linear workflows and collaboration. For classroom application, it is important to note that GitHub users can create and share private repositories.

GitHub Classroom provides a tool for streamlining the organization of private student repositories and has been found to be very successful in teaching Statistics, see, i.e., Fiksel et al. (2019). The materials for a lesson, an assignment, or computer lab can be put into a repository for distribution among students. Once the student has finished the work, the repository can then be 'pushed' for the instructor to 'pull' for grading. Feedback can be disseminated through comments directly in the student's Jupyter notebook or in the 'feedback' section of the student assignment repository on GitHub if enabled. The core GitHub functions, including unlimited public and private repositories, are free for all users, while GitHub Classroom is only free for instructors. Instructors, graduate students, and undergraduate students working on research projects can apply for an educator or researcher discount to get a free GitHub pro account to increase storage space and active time.

Python, Jupyter Notebook/Lab/Hub

Python has distinguished itself as the lingua franca of data science and is the fastest-growing programming language in the world, see e.g., Stackoverflow (2019). Python has several libraries that facilitate the pedagogy of econometrics: Numpy, Pandas, and statsmodels. Anaconda is a distribution of Python that contains most of the commonly used packages for data science and machine learning. We recommend instructors to adopt this distribution for the ease of use. Along with its convenience, Anaconda includes the open-source package management system, Conda.[1]

Anaconda includes an interactive notebook interface called Jupyter Notebook, with its namesake being the combination of Julia, Python, and R. A notebook allows the creation and sharing of documents that contain live code, equations, and explanatory text, making it one of the most popular coding learning tool, see, e.g., Koehler and Kim (2020). Specifically, Jupyter Notebook combines Markdown with LaTeX, and it can run Python, R, and Stata programs through the IPython, IRkernel, and stata_kernel kernels.[2] It provides an interactive environment where students can learn how to write/execute code step by step, mimicking an iterative discussion between them and the instructor. This environment is also useful for sharing work with others and for ensuring reproducible work. It has become the working tool for data scientists while streamlining workflows and a expected learned skill for entry-level jobs in data science, see, e.g., Perkel (2018).

For instructional purposes, Jupyter Notebook can be used as the front end of R and Stata, which assists the teaching of data science (see below). Jupyter Notebook supports several user-written extensions suitable for work in a classroom setting, see, e.g., Jupyter et al. (2019). Anaconda also offers the newly established interface JupyterLab, which is meant to be the next generation of Jupyter Notebook.[3] Project Jupyter's products are the preferred tools of data scientists when interactively working with big data. Furthermore, many cloud services, such as Microsoft's Azure Notebooks and Amazon

[1] Anaconda offers a more basic version of this distribution known as Miniconda. Miniconda includes Conda; however, it leaves the installation of packages to the user which can be cumbersome to the inexperienced student.

[2] Several other community maintained kernels are also available.

[3] In this paper, we use the term Jupyter Notebook and JupyterLab interchangeably. Although the user interfaces look differently, Jupyter Notebook and JupyterLab have identical underlying functionality.

Sagemaker incorporate these platforms as their interface with Python or R.

Finally, JupyterHub is another Jupyter product that instructors can install on remote servers to give students access to a shared workspace where they will be able to launch their own Jupyter notebook, execute their programs, and collaborate without setting up their own computing platforms and manage their package dependencies.

R and Stata

R continues to be the language of choice for instruction and academic research, with its extensive suite of statistics and econometrics packages, see, e.g., Kaplan (2018). R is most commonly paired with RStudio which is a highly optimized integrated development environment (IDE) hosting an interactive interface with multiple windows allowing for object viewing. As mentioned earlier, using R as a Jupyter Notebook's kernel allows the student to interact not only with code, printed output, and plots, but with equations and text in a unified framework. On the other hand, package management is integrated in RStudio but not in Jupyter Notebook. Both RStudio and Jupyter Notebook allow for the separation of code into "chunks" and support Markdown text and multiple distributions of R. One such distribution, Microsoft R Open, builds upon the core distribution of R to increase speed and reproducibility. This is the distribution of R that is automatically deployed in Microsoft's Azure Notebooks for example (see below).

As with R, Econometricians can use Stata as a software to aid in statistical analysis and instruction. With the launch of version 16, Stata now integrates with Python, see, e.g., Ho et al. (forthcoming). The Jupyter Stata kernel requires a currently-licensed version (13 or above) of Stata already installed, and the type of license would vary between the workflows explained in the next section. For example, workflow 2 in Section 3.2 would require students to purchase a license to be run on their virtual machines (VMs). At time of writing, individual 6-month student licenses can be obtained for 48 U.S. dollars. These can be installed in any machine owned by the student running any operating system, including the students cloud-hosted virtual machines (VM). On the other hand, workflow 3 in Section 3.3 would require a Student Labs license, with the size of the lab reflecting the number of student installations. Instructors should contact Stata to discuss the best option because many economics departments may already have licenses that could be used for this purpose.

Cloud Environments

The Cloud allows the storage of files, data, and programs by a third party provider that can be accessed from the internet; consequently, cloud computing is the act of accessing this storage. Many vendors provide a cloud computing platform free of cost; however, network delay, known as latency, can hinder larger data cleaning and preprocessing tasks, see, e.g., Popescu, Zilberman, and Moore (2017). The proposed lab environment will not require this magnitude of processing and should avoid any issues of this kind.

There are numerous cloud services vendors, but this paper will focus on Amazon Web Services (AWS) because both Emory University and Ryerson University use AWS as their provider. Other vendors that higher education institutions are likely to contract services from are Google Cloud, IBM Cloud, and Microsoft Azure. Although the full scope of their products must be paid for, all vendors listed offer free options to instructors and students. Students are able to run either VMs running Windows 10 OS or utilize pre-constructed environments (images) with interfaces designed for data science.

Both AWS and IBM Cloud offer data science 'workbenches' that offer a wide variety of pre-constructed tools geared toward machine learning, including cloud-based Jupyter Notebook editors.

However, instructors and students might find Google's Colaboratory and Microsoft's Azure Notebooks to be simpler, fully cloud-based alternatives that only require students to have access to a computer with a browser and an internet connection. Depending on the type of workflow the instructor chooses, either the instructor and/or the students will need to create accounts and may need to input credit card information. All of the aforementioned services offer free credit upon registration to both instructors and students to provide incentives to use the cloud.

As noted above, each vendor offers a cloud data science environment that gives students the opportunity to edit Jupyter Notebooks without downloading any software locally. AWS SageMaker, which may be accessed through an AWS Educate account, offers an extensive suite of machine learning tools which may be employed through the use of the provided educational credits. Using SageMaker, students may create projects, edit Jupyter notebooks using Python and R without need for any further installation, and manage and create Git repositories.

IBM Watson Studio operates in a similar manner. Students gain access through the IBM Cloud website and may also create projects that include IBM DataPlatform Notebooks, integrated Jupyter Notebook instances using R or Python. A free version available to students which limits the amount of storage. The interface is easier to navigate than that of SageMaker, though it is also designed for machine learning tasks and includes an array of tools that will not be useful to introductory students. Watson Studio may be paired with a local program, Watson Studio Desktop, which is also free for students.

Alternatively, students may access Microsoft's Azure Notebooks directly via a browser. Signup with an existing or newly created Outlook email is free. They can clone repositories directly from GitHub or create their own projects with Jupyter Notebooks in R or Python. Like SageMaker and Watson Studio, Azure Notebooks uses the native Jupyter editor; however, it does not provide the capacity for collaboration within its platform.

Google Colaboratory can also be accessed directly via a web browser using a Google account. Notebooks are stored individually and not as a part of larger projects, and the editor differs slightly from the native Jupyter editor with different menu items and keyboard shortcuts. Collaboration is quite simple: notebooks are saved into the user's Google Drive and they may share notebooks with other Google users to edit or view, though there is latency of up to 30 seconds for updating edits when working simultaneously. It is important to note Colaboratory sacrifices some useful features for simplicity: notebooks cannot be downloaded into HTML, datasets are deleted at the end of each session, and larger datasets may take longer to load. Table 2 summarizes the above discussion.

3 Workflow Environments

This section describes the three most common workflows that instructors use to set up in-class exercises, assignments, or on-demand tests. Figure 1 displays them and should be read clockwise starting from the top-left corner. These workflows share a common theme that begins with the instructor creating an assignment using Jupyter Notebook. The assignment is then distributed electronically to each student in the class using GitHub Classroom. Once the students complete their assignment, they submit by uploading (also known as pushing) the files to their respective GitHub repository. The instructor access the submitted files via GitHub Classroom to download each student repository for grading, alternatively the instructor can grade them directly on GitHub using its feedback function. Note that all workflows use a version control system, GitHub specifically, for assignment distribution and collection. Introducing students to version control in these workflows teaches them industry standards used in most data analyst entry level jobs.

The crucial aspect of how these workflows differ is the environment in which students work on

their assignments. All workflows function identically in terms of teaching students how to write and execute code. We believe that providing a centralized cloud computing environment (workflow 3 in Section 3.3 below) will allows students to better focus on developing their programming skills. Detailed explanation for our rationale is provided in Section 3.4.

3.1 Workflow 1 – Traditional Way

In this workflow (Figure 2a), students find their own ways to complete their assignments. This is the way that most instructors might be familiar with. Students may use a computer lab where all the necessary software are pre-installed to complete the assignment. If a computer lab is not accessible, students may have to use their own computing equipment, like laptops or tablets, to do the work. In this scenario, students need to install all the necessary software and packages on their own equipment, but unlike workflows 2 and 3 access to the internet is not needed. Although installation and management of software packages are valuable skills on their own right, it can be a frustrating experience that might overwhelm many beginners, given the diversity of OS and hardware specifications. These tasks are otherwise internalized by the instructors in workflows 2 and 3 explained below.

3.2 Workflow 2 – DIY Cloud Computing Service

The instructor may use a virtual lab (Figure 2b) to avoid requiring students to install and maintain the necessary software. There are two possible scenarios. In one scenario, the instructor request students to use their own equipment to connect a fully-managed cloud computing service with most/all the required software installed. For example, the instructor might ask students to open an account on AWS Sagemaker or Azure Notebooks with Jupyter Notebook pre-installed. However, some required packages may need to be installed each time the student connects and a fully-managed service does not allow the installation of any other statistical software like Stata for example.

The other scenario is for the instructor to create a computing environment with all the needed software, such as lab-licensed Stata, set up and ready for use. This computing environment is registered as a pre-built image with a cloud service vendor. Students sign up for their own cloud service and set up their virtual machines by choosing their hardware specifications and loading the registered image. Although an instructor-customized environment is more flexible than a fully-managed cloud computing service, students still need to follow a convoluted set of steps to set up their virtual machines. While most cloud service vendors provide certain free tier of their cloud services, the downside is that each student in the class must register and provide a credit card number upon registration to cover any expense above the free computing time allowance.

3.3 Workflow 3 – Centralized Cloud Computing Environment

In this workflow, the instructor creates a VM on the cloud and sets it up as a server that students can access via a web browser. (Figure 1c). The instructor is in charge of all the hardware specifications and setting up the necessary software applications for the students. Each student connects to this server via a web browser, completes the assignment using the server resources, and save their work (either on their own computers or on the server) to be push to their GitHub repositories at a later time. Access to the server can be granted by various methods, and resource available for each student can also be specified by the instructor.

All major cloud service vendors provide *auto-scaling* capabilities, which allows the cloud to monitor resource usages and automatically adjusts its capacity to provide stable performance at a low cost. For example, when usage is low over weekends and early mornings, the cloud can automatically scale down

to a smaller size to reduce cost, and it automatically scales up to avoid congestion when many students simultaneously log in for lectures or before their homework is due.[4]

3.4 Our Chosen Workflow

The choice of workflow should depend on students' knowledge in computing. Workflow 3 has the *lowest* entry barrier for students because a web browser is essentially all it takes for students to start their computation exercise. Students working in identical computing environment means that they can focus on the tasks at hand, without worrying about course-irrelevant issues such as installing new libraries and having the correct version for their applications. Thus, workflow 3 is most suitable for first-semester courses in which students have limited knowledge about programming and scientific computing. While workflow 3 imposes greater responsibilities on instructors to set up their VM, the information technology support at their higher education institutions should have sufficient knowledge to assist them efficiently taking into consideration the class size.

Workflow 2 is suitable for more experienced students enrolled in advanced undergraduate courses or postgraduate studies, because they may more flexibility in cloud resources or software needs. For example, students may conduct term projects using different data sets or implementing different methods, some of which may involve more intensive computation exercise such as parallel computing or machine learning models. Learning how to do cloud computing is an intrinsic element of workflow 2, which per se is a valuable skill set.

Since students with limited programming knowledge requires more assistance, and because workflow 2 and 3 can indeed be created with similar procedures, the rest of the paper is focused on demonstrating workflow 3. The process of setting up a virtual server, security, and deployment is quite involved to be discussed here. Instead, a thorough guide on how to create a virtual server on AWS can be found at the following static link for future reference:

`https://docs-jupyter.davidjachochavez.org/`

4 Workflow Demonstration

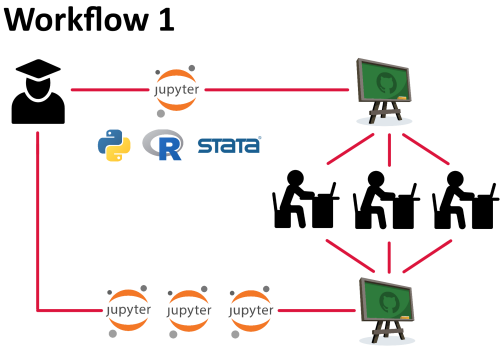
An example based on replicating some results and figures in Hansen (2020) is used to demonstrate how an instructor could go about implementing workflow 3 described in Section 3.3. It explains the process from setting up an empirical assignment, its distribution, its collection, and eventual grading.

The instructor is assumed to have a work computer, access to the internet, and has successfully set up an organization on GitHub, created a GitHub classroom within this GitHub organization, and successfully invited students to join. Instructors are encourage to consult Fiksel et al.'s (2019) very detailed guide on how to set these up at `https://github.com/jfiksel/github-classroom-for-teachers`. The instructor is also assumed to have a working local installation of Git and GitHub Desktop in said work computer.

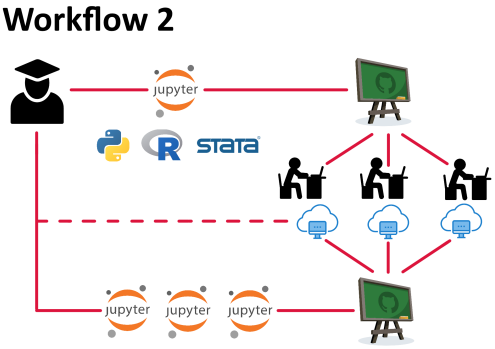
Similarly, students are assumed to have created a free basic GitHub account. The instructor can direct them to Fiksel et al.'s (2019) similar detailed guide on how to set these up and join the instructor's GitHub classroom at `https://github.com/jfiksel/github-classroom-for-students`. Students are also assumed to have gained familiarity with Jupyter Notebook/Lab through in-class demonstrations or through video tutorials provided by the instructor.

[4] The minimum and maximum size of the cloud can be specified by the instructor.

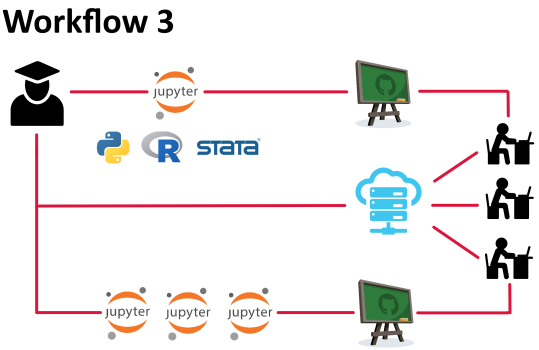
Figure 1: Potential Workflows for Instructors



(a) Workflow 1: Most decentralized workflow. Students complete assignments on their own computing hardware.



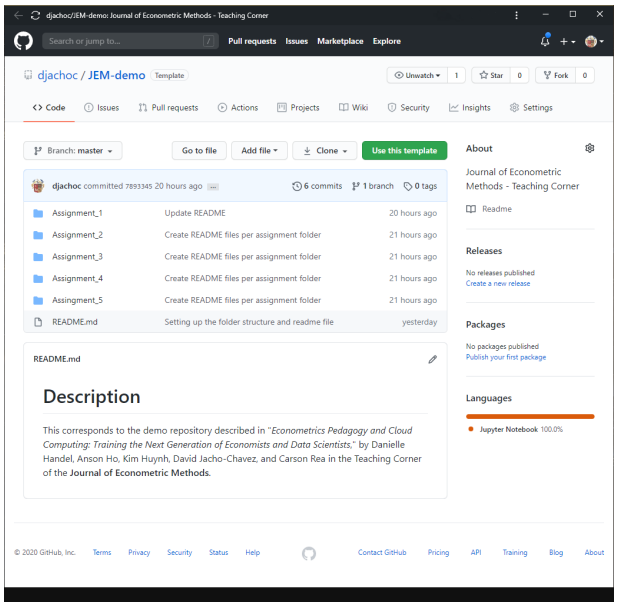
(b) Workflow 2: Partly decentralized workflow. Students complete assignments on individually managed free VMs or instructor's provided virtual lab.



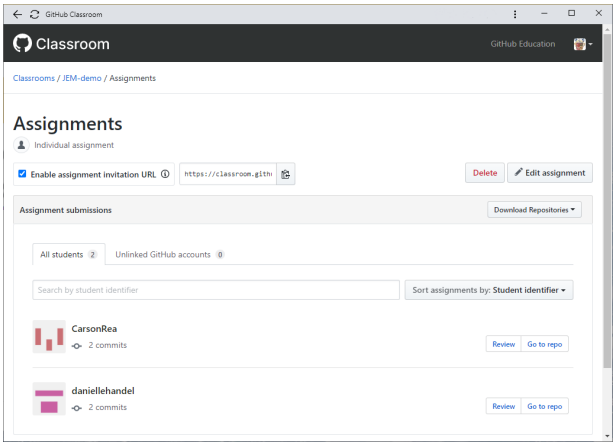
(c) Workflow 3: Most centralized workflow. Students complete assignments on a virtual server built and mantained by the instructor.

Note: The instructor ( ) creates a Jupyter notebook ( ) with a Python ( ), R ( ), or Stata ( ) kernel, and distributes it to (or collects it from) students ( ) using GitHub Classroom ( ). The students then use their own hardware (workflow 1), or build an individual VM ( ); the VMs could also be provided by the instructor, as denoted by the dashed line (---) in workflow 2. Alternatively, the instructor provides and maintains a virtual server ( ) in workflow 3.

Figure 2: GitHub Workflow



(a) Repository template for distribution to students through GitHub Classroom.



(b) Instructor's view from GitHub Classroom of students' individual assignment repositories.

4.1 The GitHub Repository Template

The instructor should start by creating a template repository. Here it is called `JEM-demo`, see, i.e., Figure 2a. Best practices suggest that instructors should create a folder per assignment or lesson, along with a `README.md` (markdown) file describing the repository content.

> The GitHub Desktop, a desktop application for GitHub, can easily aid instructors in creating this repository locally on their work computers. Once created, the instructor can easily create folders and text files using their operating system capabilities. Once completed, GitHub Desktop can be used again to stage and then push the changes to the master on GitHub.

By placing each Jupyter notebook assignment/lesson in its designated folder, this structure allows reusing the same student repository (see Sections 4.3 and 4.5) for the entire course duration. This effectively reduces cluttering in the instructor's organizational GitHub interface and it is highly recommended for large class sections.

4.2 Creating an Assignment via Jupyter Notebook

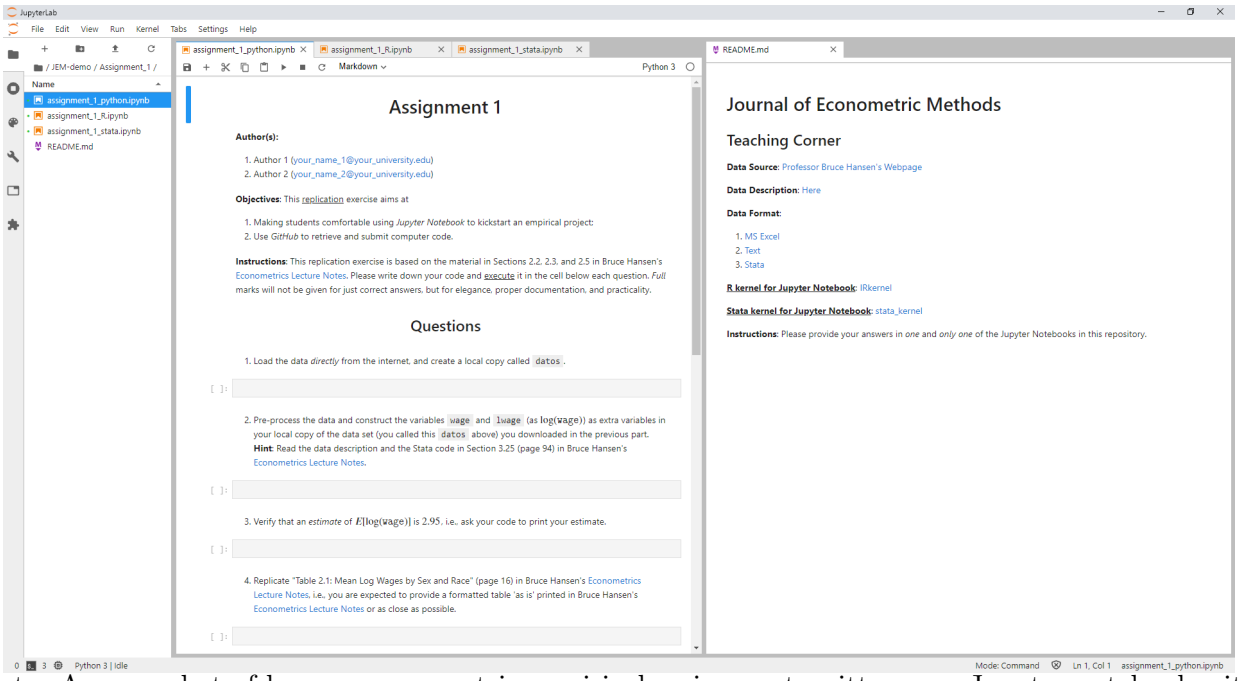
Using the work computer, the instructor creates a Jupyter notebook using the relevant Jupyter kernel chosen for the class inside the corresponding folder. Figure 3 displays how an assignment (`Assignment 1`) saved in the relevant folder (`Assignment_1`) may look like. If students are allowed to choose their programming language to complete the assignment, then adding the Jupyter kernel name to the Jupyter notebook would be a good idea, i.e., `asignment_1_python.ipynb`, `asignment_1_R.ipynb`, and `asignment_1_stata.ipynb` in Figure 3. It is also advisable the instructor creates a self-contained `README.md` file with detailed instructions per assignment.

As pointed out by Koehler and Kim (2020), it is recommended the Jupyter notebook is self-contained with all the necessary hyperlinks, graphs, and equations (knowledge of LaTeX syntax is needed) follow

the same structure as an assignment to be distributed on paper, but with coding cells instead of empty space.

> If the answers to an instructor's question requires the display of mathematical equations, the instructor might suggest students to use the free version of the Mathpix Snip desktop application or smartphone app. This service will convert any snippet of a hand-written or typeset formula into LaTeX code for inclusion into their Jupyter notebook.

Figure 3: Using JupyterLab to Access the Jupyter Notebook and Instructions



Note: A screenshot of how an econometric empirical assignment written as a Jupyter notebook with different kernels would look like when accessed using the JupyterLab GUI instead. Note that instructions (written in a `README.md` file) can be directly previewed from within JupyterLab.

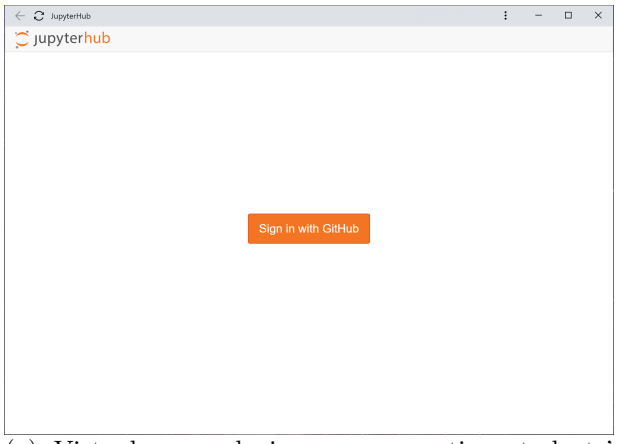
4.3 Distribution using GitHub Classroom

Once the instructor has finished the assignment in Jupyter notebook, the entire repository can be pushed to GitHub via the GitHub Desktop application. Then the instructor creates an assignment on GitHub Classroom using the previously created repository in Section 4.1 as a template. GitHub Classroom will then distribute the assignment repository to the list of students' GitHub handles the instructor used to populate the GitHub classroom. For example, Figure 2b displays the instructor's view of the distributed assignment. In this example, the `JEM-demo` classroom was created by the instructor and students with GitHub usernames `CarsonRea` and `daniellehandel` accepted the assignment sent to their e-mail addresses associated with their GitHub usernames.

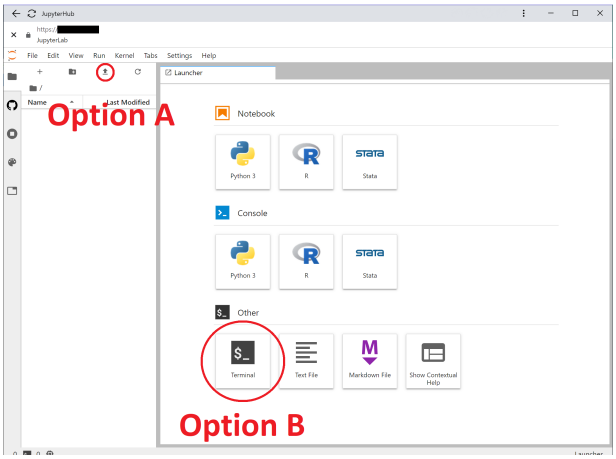
> Instructors should keep in mind that GitHub Classroom is a teacher-facing tool that simplifies the educational use of GitHub. Ultimately students use GitHub; they do not use GitHub Classroom.

The instructor can also generate an assignment invitation URL (web address) and post it on a learning management system like Canvas for students to join directly from there.

Figure 4: Virtual Server Login and Assignment Completion



(a) Virtual server login page requesting students' GitHub credentials.



(b) Student's view of the virtual server upon login.

Instructors might suggest students to also use the GitHub Desktop application on their local machine to create a local copy of their private assignment repository. This free software from GitHub will ease their workflow by providing a GUI to execute Git commands.

4.4 Assignment Completion and Submission

Once the assignment has been distributed to students, each student can then use a web browser of their choice, type the web address of the JupyterHub created by the instructor, and sign in using the GitHub credentials, see, i.e., Figure 4a.

Once logged in, the students have two options to access their files, namely option A and B in Figure 4b. Option A is perhaps the simplest for students. They would simply upload the relevant Jupyter notebook from their cloned private assignment repository created by GitHub Classroom, see, i.e., 2b, to the virtual server. Once there, students can open it, complete the assignment, and download it back replacing the original Jupyter notebook in their locally cloned private assignment repository with their answers. They can then use the suggested GitHub Desktop application to *stage* and *push* it to the cloud for grading and feedback. However, this option does away with GitHub version control and collaboration (if allowed by the instructor) capabilities altogether.

Alternatively, option B in 4b provides a cleaner solution for advanced users who are comfortable with command-line operations. The students would simply need to open a 'Terminal' on the virtual server, and from the command line clone their private assignment repository to their server working directory, complete the assignment, and then stage and push it back to the cloud directly from the virtual server. Although students would not be able to use the GitHub Desktop application as in option A above, they can clone their private assignment repository (`repo_name`) with the following command

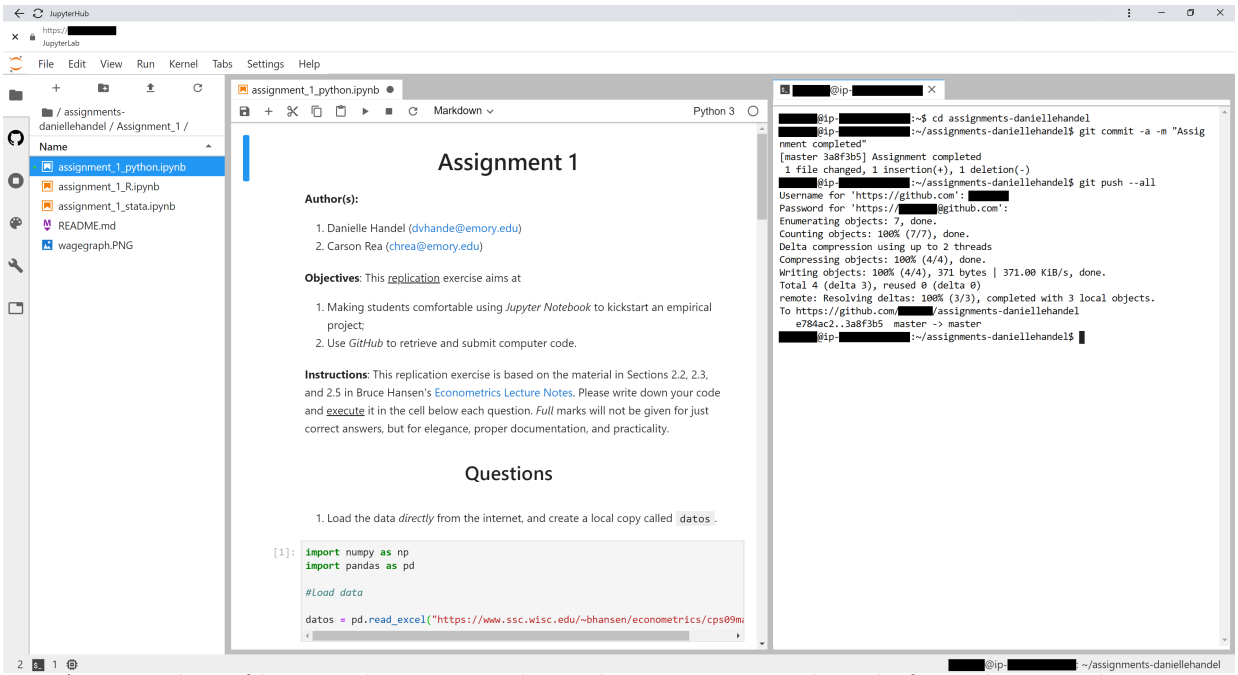
```
git clone https://<username>:<password>@github.com/<organization>/repo_name.git
```

where `<username>` and `<password>` refer to the student's GitHub username and password respectively, and `<organization>` refers to the name the instructor gave to his/her GitHub organization, see, i.e., https://github.com/jfiksel/github-classroom-for-teachers for details.

Upon assignment completion, the student again opens a terminal in the server, commit their changes, e.g., `git commit -a -m "A commit message"`, and submit it by pushing it to the online

GitHub repository, e.g., `git push --all`. Figure 5 provides a view of this process from the student's perspective.

Figure 5: Assignment Direct Submission from the Virtual Server



Note: A screenshot of how students may submit their assignment directly from the virtual server upon completion.

4.4.1 JupyterHub Administrator

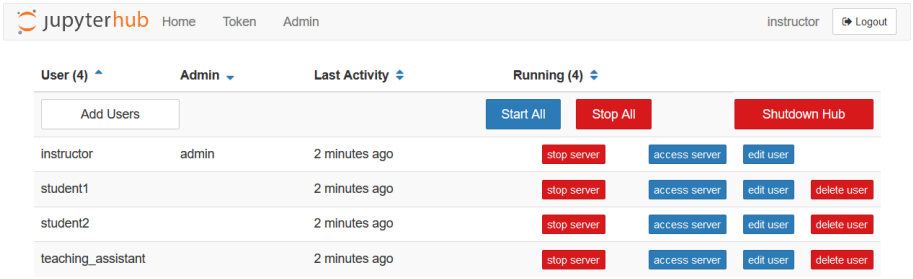
The instructor can control the JupyterHub server in real time through its administrator (admin) interface in the Hub control panel, see Figure 6. Available functions through the admin interface is shown in Figure 6a. Most importantly, the instructor can provide immediate help or troubleshooting for a specific student by accessing the student's server. Other administrative functions available include starting/stopping user servers and adding/deleting users. The instructor can also assign other users as JupyterHub administrators, for example a teaching assistant, by editing the target user, see Figure 6b.

4.5 Assignment Collection, Feedback, and Grading

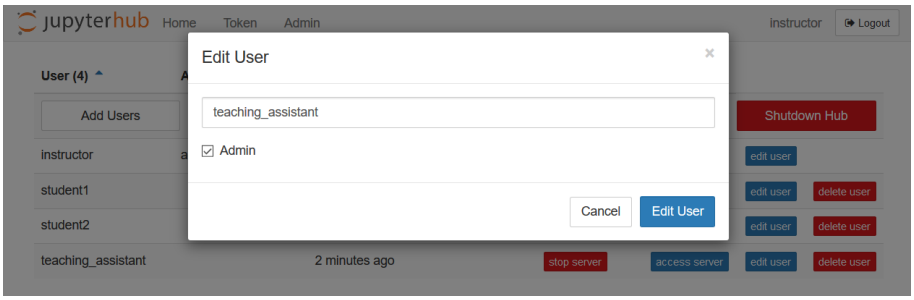
Once all students have submitted their assignments by the posted deadline, the instructor has two options. For smaller classes, feedback can be directly given on the GitHub online repository itself in the form of a feedback pull request. Figure 7 provides a visualization of this option. Alternatively, all students repositories can be forked and downloaded locally using the GitHub Classroom Assistant desktop application for grading, for example using the automated grading function with Jupyter et al.'s (2019) `nbgrader` extension.

- If the instructor chooses to 'enable feedback pull requests' as the assignment is set, a pull request automatically opens when a student accepts their assignment, so the instructor can start providing in-line feedback as they submit code.

Figure 6: JupyterHub Administration Interface

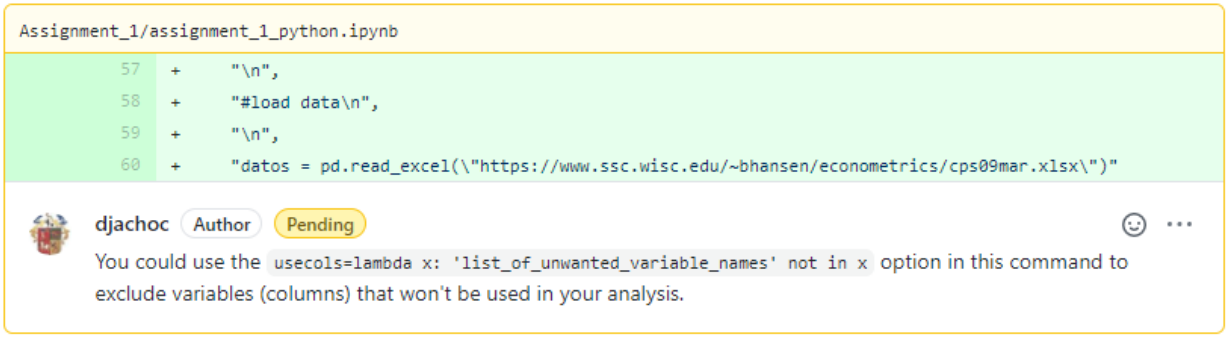


(a) JupyterHub admin interface.



(b) Assign new JupyterHub administrator.

Figure 7: Sample Feedback through GitHub



Note: A screenshot of how an instructor can provide feedback directly on a student GitHub online repository.

Finally, instructors should keep in mind that privacy policies adopted by most higher education institutions forbid publishing grades outside institutions' own sanctioned learning management system (LMS). Therefore, instructors should refrain from recording marks directly on the GitHub repositories, but use their LMS' rubric-based grading capabilities instead.

5 Conclusion

With the quick adoption of cloud computing services by higher education institutions, government, and industry along with the easy accessibility to free web-based applications, the instruction of 'how to' in the econometric curriculum can now be fully transformed to an online setting. Introducing undergraduate and graduate economics students to industry standards like version control, live coding,

and cloud services as part of their classroom training will not only prepare them better for the workforce, but because these tools are the same data scientists are trained with, they are going to be able to compete for similar jobs with the added benefit of understanding causality, and perform inference. The paper presents a demonstration on how an instructor can set up an assignment using Jupyter Notebook, distribute it to students using GitHub Classroom, setup a JupyterHub server for students to connect using almost any electronic device with an internet connection and a web browser, complete the assignment using Python, R, or Stata, and submit it back for grading and feedback. The workflow illustrated in this paper also allows an instructor to provide in-class demonstrations/exercises, without being restricted by the number of seats or even having access to a computer lab. A GitBook is provided with details instructions on how to setup said server on Amazon Web Services for future reference.

6 Acknowledgements

We thank the editor and an anonymous referee for helpful comments that improve the readability and exposition of the paper. We also thank Amazon Web Services (AWS) Educate and Stata Corporation for providing us with a cloud classroom with credits and temporary Stata lab licenses for testing, respectively. AWS is Emory University's preferred and recommended cloud service for faculty-led computational needs. Handel, Jacho-Chávez, and Rea acknowledge financial support from the Department of Economics at Emory University. The views expressed in this article are those of the authors. No responsibility for them should be attributed to the Bank of Canada. All remaining errors are the responsibility of the authors.

References

Athey, Susan and Michael Luca. 2019. "Economists (and Economics) in Tech Companies." *Journal of Economic Perspectives* 33 (1):209–30. URL https://www.aeaweb.org/articles?id=10.1257/jep.33.1.209.

Bryan, Jennifer. 2018. "Excuse Me, Do You Have a Moment to Talk About Version Control?" *The American Statistician* 72 (1):20–27.

Cicero, Marcus Tullius. 44 BCE. *De Divinatione, , II.(2).4.* English translation by William A. Falconer, Loeb Classical Library (1923), Cicero, Vol. 20, p375; as transcribed by Bill Thayer, https://penelope.uchicago.edu/Thayer/E/Roman/Texts/Cicero/de_Divinatione/2*.html#R2 (retrieved on September 23, 2020).

Fiksel, Jacob, Leah R. Jager, Johannna S. Hardin, and Margaret A. Taub. 2019. "Using GitHub Classroom To Teach Statistics." *Journal of Statistics Education* 27 (2):110–119. URL https://doi.org/10.1080/10691898.2019.1617089.

Hansen, Bruce. 2020. "Econometrics." Retrieve from https://www.ssc.wisc.edu/~bhansen/econometrics/.

Ho, Anson T. Y., Kim P. Huynh, David T. Jacho-Chavez, and Diego Rojas. forthcoming. "Data Science in Stata 16: Frames, Lasso, and Python Integration." *Journal of Statistical Software* .

Jupyter, Project, Douglas Blank, David Bourgin, Alexander Brown, Matthias Bussonnier, Jonathan Frederic, Brian Granger, Thomas Griffiths, Jessica Hamrick, and Kyle et al. Kelley. 2019. "nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook." *Journal of Open Source Education* 2 (11):1:32.

Kaplan, Daniel. 2018. "Teaching Stats for Data Science." *The American Statistician* 72:89–96.

Koehler, Jacob Frias and Soomi Kim. 2020. "Interactive Classrooms with Jupyter and Python." *The Mathematics Teacher* 111:304. URL http://dx.doi.org/10.5951/mathteacher.111.4.0304.

Perkel, Jeffrey M. 2018. "Why Jupyter is data scientists' computational notebook of choice." *Nature* 563 (7729):145–146.

Popescu, Diana Andreea, Noa Zilberman, and Andrew W. Moore. 2017. "Characterizing the impact of network latency on cloud-based applications' performance." Tech. Rep. UCAM-CL-TR-914, University of Cambridge, Computer Laboratory. URL https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-914.pdf.

Stackoverflow. 2019. "Stack Overflow's Annual Developer Survey." Online. URL https://insights.stackoverflow.com/survey/2019.

Wikipedia. 2020a. "Distributed Version Control." *Wikipedia* URL https://en.wikipedia.org/wiki/Distributed_version_control.

———. 2020b. "Graphical User Interface." *Wikipedia* URL https://en.wikipedia.org/wiki/Graphical_user_interface.

———. 2020c. "HTML." *Wikipedia* URL https://en.wikipedia.org/wiki/HTML.

———. 2020d. "Integrated Development Environment." *Wikipedia* URL https://en.wikipedia.org/wiki/Integrated_development_environment.

———. 2020e. "Jupyter Kernels." *Wikipedia* URL https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_kernels.

———. 2020f. "Live Coding." *Wikipedia* URL https://en.wikipedia.org/wiki/Live_coding.

———. 2020g. "Machine Learning." *Wikipedia* URL https://en.wikipedia.org/wiki/Machine_learning.

———. 2020h. "Markdown." *Wikipedia* URL https://en.wikipedia.org/wiki/Markdown.

———. 2020i. "Open Source." *Wikipedia* URL https://en.wikipedia.org/wiki/Open_source.

———. 2020j. "Programming Language." *Wikipedia* URL https://en.wikipedia.org/wiki/Programming_language.

———. 2020k. "Software Repository." *Wikipedia* URL https://en.wikipedia.org/wiki/Software_repository.

———. 2020l. "Virtual Machine." *Wikipedia* URL https://en.wikipedia.org/wiki/Virtual_machine.

Table 1: Glossary of Terms

Term	Definition
Distributed Version Control System	Distributed version control systems, such as Git, enable the tracking of changes to code by several collaborators. This system allows for offline work and access to the full revision history. Wikipedia (2020a)
Graphical User Interface (GUI)	Graphical User Interfaces (GUIs) allow users to operate programs via an interactive interface including buttons and menus, rather than through a command line, which simply processes text. Source: Wikipedia (2020b)
Hypertext Markup Language (HTML)	Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. Source: Wikipedia (2020c)
IDE	An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. Spyder, Rstudio, and the inbuilt Stata do file editor are common IDEs for Python, R, and Stata respectively. Source: Wikipedia (2020d)
Jupyter Kernels	Jupyter kernels are programs that execute, complete and inspect code within Jupyter Notebooks. Choosing a kernel (R, Python, etc.) amounts to choosing a language in which to write the code within a Jupyter notebook. Source: Wikipedia (2020e)
Live Code	Live coding, sometimes referred to as on-the-fly programming, just in time programming and conversational programming, makes programming an integral part of the running program. Source: Wikipedia (2020f)
Markdown	Markdown is a lightweight markup language with plain-text-formatting syntax. It is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor. Source: Wikipedia (2020h)
Machine Learning	Machine Learning involves building computer algorithms based on sample or "training" data in order to make predictions. Source: Wikipedia (2020g)
Open Source	Open source refers to a computer program in which the source code is available to the general public for use for any (including commercial) purpose, or modification from its original design. Source: Wikipedia (2020i)
Programming Language	A programming language is a formal language comprising a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. Source: Wikipedia (2020j)
Repository	A software repository, or "repo" for short, is a storage location for software packages. Source: Wikipedia (2020k)
Virtual Machine (VM)	In computing, a virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination. Source: Wikipedia (2020l)

Note: This corresponds to the list of technical terms used in the main text along with their definitions taken from Wikipedia.

Table 2: Free Cloud Services and Available Software

Cloud Patform	Vendor	Software	Strengths	Weaknesses
AWS Sage-Maker	Amazon Web Services	Git, Jupyter, Jupyter-Lab, Python, R	Large selection of pre-installed machine learning tools.	• House in a complex environment. • Free USD credits are limited and expire.
Azure Note-books	Microsfot Azure	Git, Jupyter, Python, R	• Simple setup and interface. • Little to no learning curve.	• Assumes familiarity with Jupyter and Git. • Collaboration is not allowed. • Packages must be installed again once session ends.
Colaboratory	Google Cloud	Git, Jupyter, Python, R	• Simple setup and interface. • Simplifies collaboration. • Includes light version control.	• Does not allow downloading into HTML. • Deletes data sets at the end of the session.
IBM Watson Studios	IBM Cloud	Git, Jupyter, Python, R, Rstu-dio	Large selection of pre-installed machine learning tools.	• House in a complex environment. • It limits instances in its free version.

Note: This is a non-exhaustive list of popular cloud services available for free to learn or practice data science.