# NEON instructions for developing or updating Data Skills Tutorials

Donal O'Leary, Claire Lunch

24 June 2021

## Table of Contents:

### 1. Workflow Overview

This brief overview will serve as an outline of the steps described in detail below.

1. Identify need for update or development of a tutorial
2. Fork the NEON-Data-Skills repository
   - Create or update your tutorial on your fork, based on the template
   - Update YAML header information
   - Refer to the template and style guide for examples of preferred method for links, graphics, etc.
   - New tutorials can be written initially in the tutorials-in-development directory, which does not sync to the website. This is especially useful if you will need to push other materials while the tutorial is in progress.
   - Confer with NEON Subject Matter Experts (SMEs) on the materials (as needed)
3. Once the tutorial document is ready:
   - Update your fork to bring in any changes from main
   - Assign the tutorial title to the next syncID in TutorialUniqueIDRecord.csv
   - Copy the syncID into the YAML header
   - Run the appropriate 'knitting' script in the 'processing_code' directory
   - Check that output *.html,* .md, *.r,* .py, figures all look good
   - Submit pull request from your fork to the main NEON-Data-Skills on GitHub
4. Conduct review process. Admins are permitted to approve their own pull requests; others should tag an admin in the PR for review.

5. When the 'main' tutorials repo is updated on GitHub, it calls a 'hook' to initiate a CircleCI process which will take in the updated .md file and convert it into a 'tutorial' content type page on neonscience.org
    - All of the YAML header information will be used to generate the syncID, title, description, author list, link to download code, etc.in the tutorial webpage
    - Verify that the CircleCI process results in a green check mark on GitHub. If not, investigate for formatting errors, and/or contact an admin for help.
    - Check the resulting page to ensure that links and figures work, that code chunk formatting makes sense.
6. If this is a new tutorial, the sync to the website will create the page but leave it unpublished. Verify that the page looks as expected, then publish it. If you are not an admin, ask an admin to take these steps.
7. Add tutorial to a tutorial series if needed

## 2. Background

The NEON Data Skills program is a central component of NEON education and outreach efforts. By some measures, the Data Skills Tutorials are the most visited parts of the neonscience.org website, with ~70% of all page views being tutorials, and ~50% of all site visits including a visit to at least one tutorial. Indeed, many tutorial users are unfamiliar with NEON and have just found these resources by searching for how to solve a particular data skills challenge. In brief, NEON's data skills tutorials are important and highly visible resources with a great potential for introducing scientists to the NEON program.

These resources have had many authors since the inception of the NEON program. The tutorial publication system, which syncs documents from the public NEON-Data-Skills repo to the NEON website, allows for contributions from both internal NEON staff and external collaborators.

## 3. Accounts

You will need the following accounts:

### Contributors:

- GitHub account with membership in the NEONScience organization

### Admins:

- GitHub account with Admin access to NEON-Data-Skills
- neonscience.org/user account with edit access to the DS content types (tutorial, workshops, series, teaching module)

## 4. Tutorial Template

The tutorial template is a good place to start when making a new tutorial from scratch. Rmarkdown tutorial template is found in NEON-Data-Skills/tutorials-in-development/0_templates_style_guide/tutorial-template-Rmarkdown.rmd There is currently no template for Python-based tutorials. In practice, much of your tutorial may be copy/paste from another existing tutorial, especially the introductory information. Therefore, it may be faster to modify an existing tutorial document, making sure to keep them separate and update the syncID for the new tutorial so as to not overwrite the copied tutorial on neonscience.org! You should also review the style guide at NEON-Data-Skills/tutorials-in-development/0_templates_style_guide/

## 5. Tutorial YAML Header Info

At the top of every tutorial is a YAML header designated by three dash marks at top and bottom (—) and containing key pairs of information. YAML stands for 'yet another markup language' and is a flexible markup paradigm. The various key pair attributes that are listed populate into tutorial webpage attributes/text. Here is an example of a YAML header pulled from a real tutorial:

```
---
syncID: db9715ca243944fabbe81031f2ed5cec
title: "Select pixels and compare spectral signatures in R"
code1: https://raw.githubusercontent.com/NEONScience/NEON-Data-Skills/main/tutorials/R/Hyperspectral/In
contributors: Megan Jones, Felipe Sanchez
dateCreated: 2020-02-18
description: Plot and compare the spectral signatures of multiple different land cover types using an i
estimatedTime: 0.5 Hours
languagesTool: R
dataProudct: DP3.30006.001
packagesLibraries: rhdf5, raster, plyr, reshape2, ggplot2
authors: Donal O'Leary
topics: hyperspectral, HDF5, remote-sensing
tutorialSeries: null
urlTitle: select-pixels-compare-spectral-signatures-r
---
```

Many are fairly straightforward and intuitive ('title', 'authors', 'date'. etc.) but many deserve further explanation here where I have inserted the description instead of the value. These are described in the 'style guide' document. Note that sometimes the knitting process will arrange this YAML header alphabetically which does not seem to affect the performance of the process and therefore may be ignored.

```
---
syncID: A unique random key used to differentiate between tutorials
title: Title on the tutorial page, shown in the left navigation bar when viewing tutorial series
description: Narrative description of the tutorial, shown on tutorial overview page
dateCreated: Date the tutorial was first published
authors: Comma-separated list of authors
contributors: Authors of minor contributions
estimatedTime: Estimated amount of time for an average learner to complete the tutorial
packagesLibraries: Comma-separated list of packages required to complete this tutorial
topics: List of topics, matching options in tutorial overview page filters
languagesTool: Coding language used, matching options in tutorial overview page filters
dataProduct: DPIDs for the data products used in the tutorial
code1: The absolute URL to the 'raw code' file on GitHub
tutorialSeries: Deprecated in favor of the manual series definition process within Drupal
urlTitle: Suffix of the URl appended to https://www.neonscience.org/resources/learning-hub/tutorials/)
---
```

### 6. syncID: Tutorial Unique ID Record

The syncID is used by Drupal to identify unique tutorials in the system (these are not differentiated by 'title'). The syncID for every tutorial is listed in NEON-Data-Skills/processing_code/TutorialUniqueIDRecord.csv

When you make a new tutorial, you must take a new syncID from the list in the CSV, and add that ID to the YAML header. Then, take the new tutorial's title and add it to TutorialUniqueIDRecord.csv to ensure that nobody in the future used that syncID (which would overwrite your tutorial without any warning!).

Only work with this file in a text editor; opening it in Excel can cause re-formatting and other problems.

### 7. Setting Working Directories and filepath methods

There are a few different ways to do this. Donal prefers to save working directories as strings, and use the paste0() function to combine the working directory and output filename to make filepaths. There are several advantages to this method: 1) setting the full, explicit working directory as a string seems to work flawlessly on many different platforms (mac, windows, linux cloud servers, etc.), 2) it is easy to combine the working directory with other paths/filenames using the paste0() function, 3) this method works best when executing Rmd files one code chunk at a time, and 4) this method circumvents some issues that arise when batch processing multiple R-based tutorials. Others prefer to set working directory manually, or make a new 'project' that sets the working directory automatically, though these can be difficult to explain to others via static web pages, and may increase the risk of human error in the code. Therefore, it is suggested that you define the working directory as a string in your first code chunk an use that variable to set the working directory, and combine into filepaths, as needed per this tutorial: https://www.neonscience.org/resources/learning-hub/tutorials/select-pixels-compare-spectral-signatures-r

## 8. How to knit/process tutorial files into Markdown

### 8.1 Rmarkdown-based Tutorials:

The process to convert tutorials from the Rmarkdown (.rmd) file to a regular markdown (.md) file, and generate the associated files (.r, .html, and figures) is complex. Fortunately, we have a script to automate this process, which usually runs well once you set things up correctly.

This file is NEON-Data-Skills/processing_code/01knit-RMD-2-MDNDSRepo.R

You will need to update several lines including the variables: * gitRepoPath * wd_processing_doc * The 'dirs' and 'subDir' variables will need to point to the directory containing the tutorial(s) that you want to process. This script will search recursively in file.path(gitRepoPath, "tutorials", subDir) and process every tutorial that it finds. For example, if you wanted to re-process the entire set of R-based hyperspectral tutorials you would use "R/Hyperspectral" as your subDir Note that subDir is pasted together with other filepaths when defining rmd.files as per file.path(gitRepoPath, "tutorials", subDir), so be careful not to have a slash character at the beginning or end of subDir!

It is also recommended that you check the contents of 'rmd.files' variable to make sure that you have found full filepaths to the Rmd files that you want to process.

Also note that you will need to store all of your input datasets in a directory, currently listed as wd <- "~/Documents/data/" in most tutorials. For the tutorials that download data through the API, this should not be an issue, as those will load into the workspace automatically.

Once you are in the for (files in rmd.files){} loop everything should be automated to run smoothly (famous last words?). This will create some temporary files, convert the Rmd into .md, .R, .html files, and store all of the generated figures in the same directory as the original .Rmd file, then clear the temporary files. See the script itself for details.

A 'sanity check' is required at this point - I like to first open up the produced figures - does everything look as expected and all figures are present? Next, open the HTML file in a local browser just to see if things generally look ok (no 'errors' or other major problems). Note that you will not have the neonscience.org styling, and figures will not work until you push the files to GitHub, because the HTML file will be pointing to the NEON Data Skills git repo using full URLs, which will not be found if you're developing a new tutorial! It will also show the 'old' figures if you have changed the figures in the tutorial but haven't pushed those yet. You should also open up the .md file to see that things generally look ok and that there are no errors. It may be worth searching for 'error' or 'warning' in the .md file before pushing to GitHub.

If everything is looking good, it is time to push your results to GitHub and either go through the review process, or proceed directly to the CircleCI publication process.

### 8.2 Python-based Jupyter Notebook Tutorials:

The process to convert Python-based tutorials from the Jupyter Notebook (.ipynb) file to a regular markdown (.md) file, and generate the associated files (.py, .html, and figures) is complex. Fortunately, we have a script to automate this process, which usually runs well once you set things up correctly.

This file is NEON-Data-Skills/processing_code/02process-ipynb-2-MD_NDSRepo.R

Yes, you have read correctly. The script used to process the python files is written in R. Donal has done this for several reasons. First, it was easy to copy/paste code from the R-based processing script and some of the Redesign scripts to automate most of the processes. Second, it is very straightforward to make system calls (call command line functions directly to the terminal) from R, and it is easiest to process the Jupyter Notebook (.ipynb) files using command line functions (as shown in the script here). Third, all of the other scripts are written in R aside from the .ipynb files themselves. Finally, Rstudio has a really great integration for GitHub and Donal prefers to manage content and push changes to the NEON-Data-Skills repo using this GUI, so it is nice to work in Rstudio.

When using the 02process-ipynb-2-MD_NDSRepo.R script, you must first update the 'pattern' variable to match the filepath to the …/NEON-Data-Skills/tutorial/ directory on your local computer. This will only need to be set once on your machine.

Next, you will need to update the 'input.path' variable to point to the directory that will be searched recursively for .ipynb files. This will generate a list called 'ipynb.files' and will process all files on that list.

This script is quite a bit more straightforward than the Rmd processing script. Once inside the It will first reset the kernel and clear all code chunk outputs in the .ipynb file (to start with a blank script, essentially). It will then run the .ipynb file code chunks. Next, it will convert the completed notebook (with code chunk outputs) as .html, .py, and .md files. Finally, it clears out the .ipynb file so that the user will download a fresh, empty .ipynb file every time they choose the 'get lesson code' button.

The second half of this script will update the URLs for all of the generated figures in the .md file (so that the images link to the correct place on GitHub). This process was first developed in some of the 'Redesign Scripts' when updating the URLs for graphics en masse.

### 9. CircleCI / drupal integration

Routine tutorial creation and update don't require direct interaction with the CircleCI system. Just look for the green check on the GitHub commit. If something goes wrong, this is some background info:

CircleCI is a remote server platform that responds to periodic requests using pre-established pipelines. This is great for automating tasks such as the occasional tutorial update, or chron jobs that happen at intervals, basically any time that it doesn't make sense to host your own server because it will have 99% downtime. To access CircleCI you will need to set up a profile on the neonscience account (talk to Christine Laney or someone on CIT). The neonscience account hosts many processes called 'Projects' and this one is called NEON-Data-Skills and contains a workflow called 'build-deploy'. When you push an update to the NEON-Data-Skills GitHub repo, GitHub calls a 'webhook' to CircleCI, which initiates the CircleCI process. For details on this 'webhook' to CircleCI - see https://github.com/NEONScience/NEON-Data-Skills/settings/hooks … I don't know the details of how this works beause this was built by the ATEN Dev team. Essentially, CircleCI processes any .md files that it finds to be changed in the latest commit hash. This will do some backend magic to convert the .md file that you just pushed to GitHub into a tutorial web page, using the YAML header information to fill in various fields of the new Tutorial content type. This process usually takes less than two minutes to process. It seems to be fairly insensitive to the batch size (I think it is mostly time spent spinning up a mini server, processing the .md files happens very fast because they are relatively small text files). One this is complete, it usually takes 2-3 minutes for Drupal to make the updates to show the new or updated tutorial on the webpage. Note that new tutorials should begin as 'unpublished' pages so you will need to publish once you have reviewed the content.

Note there is also a mechanism in Drupal to re-process every tutorial in the Drupal menu 'Configuration > NEON Settings > NEON Tutorial Sync' here (https://www.neonscience.org/admin/config/neon/tutorial-sync). Read the instructions on that page for more details.

### 10. Validating new/updated Tutorials

At this point, you will of course want to check that your updates worked. If you open the content manager in neonscience.org/user you should see the updated tutorial(s) at the top of the content when sorting by 'updated.' It should also show if the tutorial is published or unpublished. Click on that tutorial to view the tutorial page. Some good things to check are:

- Links to example dataset downloads
- There are no errors in the code chunks
- All code formatting looks ok
- All links to internal and external pages go to the intended location
- ‘Download Lesson Code’ link works properly

### 11. Tutorial Series

In Repo version 1 tutorial series were defined in the YAML header, however, that method is now obsolete in favor of a new process for creating tutorial series from existing tutorials in the Drupal interface. There is now a Drupal content type called ‘Tutorial Series’ that imports existing tutorials and gives an order to those tutorials (drag and drop to re-arrange), as well as some metadata. This has made the creation and editing of tutorial series to be fairly straightforward and less bug-prone than when defining the series in the YAML header. These tutorial series should be labeled appropriately with ‘Series’ tag and the number of tutorials in that series in the Tutorial search page (see figure below).

### 12. Example Datasets on figshare

In most cases, data used in a tutorial should be freshly downloaded in the course of the tutorial, using loadByProduct() or one of the other download methods on the Data Portal or in the neonUtilities package, to ensure users understand the full process involved in working with NEON data. If this can’t be done, figshare can be used to store a static dataset.

NEON Data Skills stores some of the example datasets for these tutorials on a webpage called ‘figshare’ which is an open-access repository for scientific information. Part of the thinking behind this convention is that 1) these example datasets should be static for reproducibility with the tutorials, 2) if the NEON servers go down during a workshop the figshare links should still work, 3) many of the example datasets were developed prior to the finalization of the NEON data product that they represent, so there may be minor differences with the ongoing development of those products, and 4) there are many example datasets that are not exactly NEON data products, so they are not suitable to store on publicly-accessible NEON servers.

To use figshare you will need to login to the account. Prior to publishing anything on figshare, take a close look at the existing materials for the conventions re: citation, metadata, naming convention, etc. Warning: once something is published on figshare is it very difficult to edit or remove this content! Figshare is designed to be a permanent repository, so you will need their admin approval to edit your content. So be extra careful and triple-check your posts before publishing!

Once your data are posted to figshare, you can link to these data in the ‘data download’ section of the tutorial of interest. See existing tutorials for examples of specific implementation.

### 13. Managing the GitHub account

This is information for admins of the NEON-Data-Skills repo and/or leads of the NEON Data Skills program. Tutorial contributors can skip this section.

NEON Data Skills stores all of our tutorial content on GitHub here: https://github.com/NEONScience/NEON-Data-Skills. Use of GitHub is outside of the scope of this document, though Donal generally recommends to use the Git interface in Rstudio for ease of use in this R-centric repository. Here is a link to a great resource for all things Git: https://happygitwithr.com/

Our repository contains a few sections worth noting.

- Use the ‘Main’ repo - ‘master’ as a term has been deprecated from Git
- The ‘readme’ doc becomes the text on the repo homepage below the file structure. Use this to inform your audience of organization and expectations.
- There may be a file called ‘NOT_VERIFIED.txt’ in some of the directories - this file was introduced early in the redesign process from repo Version 1 to Version 2. This is intended to be a reminder to check every aspect of each tutorial to ensure that all is well with the transition prior to removing this

file (i.e., once you have 'verified' that all aspects of the tutorial are good, you can remove the 'not verified' file).

- A hidden directory called .circleci contains a single file called config.yml that contains instructions for the CircleCI integration. Do not edit this file unless you really know what you're doing and have conferred with ATEN!
- A directory called 'scripts' also contains a single file called 'deploy.php' - again, do not edit this unless you really know what you are doing and have conferred with ATEN!
- The directory 'graphics' contains figures and screenshots to be integrated into tutorials that were not produced by the tutorial code (code-generated figures are contained within each tutorial/s directory). These graphics are organized thematically, and there is often a one-to-many relationship between graphics and tutorials (i.e., one graphic may be used in many tutorials).
- The directory 'processing_code' contains the 'knitting' scripts to convert R and Python documents (.Rmd and .ipynb files, respectively) into markdown (.md) documents along with the associated files and graphics.
- 'processing_code/redesign_scripts' contains many scripts that Donal used to update the repo from Version 1 to Version 2.0. This included a lot of automation of file management, editing/updating links to graphics, and other processes. These scripts are deprecated but retained here because they contain bits of code that may prove to be useful in the future.
- 'processing_code/_data' contains YML lists of the applicable 'languagesTools,' 'packagesLibraries,' and 'tags' to be used in the tutorial YAML headers.
- 'processing_code/_includes' is deprecated - this contains chunks of HTML code that were used as 'includes' files in repo Version 1. These were dropped into existing pages in the old Drupal. We have moved away from these to make every tutorial more stand-alone independent.
- The directory 'tutorials-in-development' (often abbreviated 'T-I-D' in commit messages) contains some loosely-organized tutorials that are in progress. To be honest though, most of these have been abandoned by their creators. These would make for good materials to 'mature' into real tutorials, given the time and resources to invest in such an effort.
- Finally, the directory 'tutorials' contains a rigorously organized collection of the tutorials that are published on neonscience.org.

Overall organization described here: https://docs.google.com/spreadsheets/d/1hgXckDx4OoJ5bJ0SuVgepXThCwgSD3jZWCUU-Fwtm8o/edit?usp=sharing in the 'filepath-map' tab

More information about edits to tutorials and organization can be found on the Teams work from home (WFH) channel 28. Create and Update Code-based Tutorials.' See the Wiki for beginner-level instructions of how to get started with editing these resources, and see the 'Files' tab for the sheet called 'Tutorials Progress Tracking' for more detailed notes about each tutorial and the process of updating for the new 2020 website. https://teams.microsoft.com/l/channel/19%3aaa1d4f77b12d47d7873913d1b53caf31%40thread.tacv2/28.%2520Outreach-%2520Create%2520and%2520Update%2520Code-based%2520Tutoria?groupId=e3922351-c262-45f6-992e-bb8474a51b07&tenantId=f44d2ab3-9099-4d85-9986-10165a8619f5