

Updating your setup

NSDF-Data student guide 7 — Windows edition

Version 6 · 13 October 2026

Routine maintenance

Do this regularly, not once. The project keeps changing: new releases bring new code and guides, and sometimes a new version of Python or of the packages the notebooks use. If your setup is behind, things can fail in confusing ways. Follow this guide **whenever the project lead announces a new release, whenever the environment check warns you, and when you start a new piece of work after a long break.** It takes about 10 minutes, or 20 to 30 minutes if Python itself changed (mostly waiting).

Tip

Who this is for. You have finished the setup guide and the first analysis guide for Windows. Nothing here deletes your notebooks or your work. If you are not sure about a step, stop and ask the project lead.

Before you start

- **Time:** about 10 minutes (20 to 30 if Python changed; most of that is waiting for deleting and installing).
- **You need:** the setup guide finished, and your own work saved in Git (see Step 7 of the first analysis guide). Step 2 below checks.
- **Close Jupyter first:** in JupyterLab choose **File**, then **Shut Down**, and close the terminal that was running it. Updating files while Jupyter is open can fail, especially on Windows.
- **Where to find out what changed:** the [release notes](#). The project lead also writes to you when a release needs something from you: look for the words “Students, please”.

Words you will meet

Word	What it means here
Release	A numbered version of the whole project, such as 0.6.0. The number is in the file VERSION.
Environment	The sealed box of programs with fixed versions that you made in the setup guide, called <code>darkmatter_cli_env</code> .

Word	What it means here
Pinned	A package listed in <code>requirements.txt</code> with an exact version, such as <code>numpy==2.2.6</code> .
Merge	Bring the newest shared changes of the project (the branch <code>develop</code>) into your own branch.

Step 1. Check your setup

Open a **Miniforge Prompt** (click **Start**, type `Miniforge Prompt`, press **Enter**), and type these lines, one at a time, pressing **Enter** after each:

```
conda activate darkmatter_cli_env
cd %USERPROFILE%\Research\NSDF-Data\python
python check_environment.py
```

What these lines do: the first switches your environment on, the second moves you into the project's `python` folder, and the third runs the project's environment check.

Checkpoint

If it says `Environment OK`, your environment is current: go to Step 2 anyway if a new release was announced, otherwise you are done. If it says `WARNING`, it lists what differs. A line about **Python** means Step 4. A line about a **package** means Step 3. Do Step 2 first in both cases.

Step 2. Bring the newest project changes into your branch

Your branch is your own copy of the project. The newest shared changes are on the branch `develop`. Type:

```
cd %USERPROFILE%\Research\NSDF-Data
git branch --show-current
git status
```

Checkpoint

`git branch --show-current` prints `student-yourname` (your own branch), and `git status` says `nothing to commit` (or lists only files you have not added, such as your own notebook folder). If it lists changed files you want to keep, **save them first**: commit them as in Step 7 of the first analysis guide, then come back. If the branch is `develop` or `master`, stop and ask the project lead.

Now bring the changes in:

```
git fetch origin
git merge --no-edit origin/develop
```

What these lines do: `git fetch` downloads the newest changes without touching your files, and `git merge` combines them with your branch (`--no-edit` keeps `git` from opening a text editor to ask for a message). It prints `Already up to date.` or a list of the files that changed.

Careful

If git says CONFLICT, stop. Do not edit anything and do not try to fix it: copy the message and send it to the project lead.

Step 3. If packages changed: install the project's list again

Do this when the check named a package, or when the release notes or the project lead say the NSDF software changed. This step is how the NSDF software (nsdf-cli and the nsdf_dark_matter library, from the Nexus-DM project; see *Good to know* below) is updated: requirements.txt names the exact version of each, and the line below installs them. Type:

```
conda activate darkmatter_cli_env
cd %USERPROFILE%\Research\NSDF-Data
python -m pip install -r requirements.txt
```

What these lines do: the last one reads requirements.txt and installs every package on the list at the exact version given, replacing any that differ. It prints a lot of text and can take a few minutes.

Then run the check again (Step 1). It should say Environment OK. Step 5 also shows you the version of nsdf-cli you now have.

Step 4. If Python changed: make a new environment

Do this when the check says your Python is not the version the guides use. Conda can update Python inside an existing environment, but most of this project's packages were installed with pip, which does not rebuild them for the new Python, and that can leave a broken mix. It is more reliable to remove the old environment and make a new one. Nothing you need is lost: the environment is rebuilt from requirements.txt, and the first lines below save a list of what the old one had, in case you installed something extra. Type each line and press Enter, and be patient: **removing the old environment takes a few minutes, and so does the final install.**

```
cd %USERPROFILE%
conda activate darkmatter_cli_env
python -m pip freeze > %USERPROFILE%\old-environment.txt
conda deactivate
conda env remove -n darkmatter_cli_env -y
conda create -n darkmatter_cli_env python=3.12 pip -y
conda activate darkmatter_cli_env
cd %USERPROFILE%\Research\NSDF-Data
python -m pip install -r requirements.txt
jupyter kernelspec remove darkmatter_cli_env -y
python -m ipykernel install --user --name darkmatter_cli_env
```

What these lines do

Line	What it does
cd (the first line)	Moves you to your home folder, where the list of old packages is saved.
conda activate	Switches your old environment on, so the next line can read it.

Line	What it does
<code>pip freeze > ...old-environment.txt</code>	Writes the names and versions of everything in the old environment into the file <code>old-environment.txt</code> in your home folder.
<code>conda deactivate</code>	Switches the old environment off.
<code>conda env remove</code>	Deletes the old environment. This takes a few minutes.
<code>conda create</code>	Makes a new, empty environment called <code>darkmatter_cli_env</code> , with Python 3.12 and <code>pip</code> .
<code>conda activate</code>	Switches the new environment on.
<code>cd (the second one)</code>	Moves you into the NSDF-Data folder, where <code>requirements.txt</code> is.
<code>pip install -r requirements.txt</code>	Installs the project's list of packages. It takes about five minutes.
<code>jupyter kernelspec remove</code>	Removes Jupyter's old entry for the environment. If it says there is no such kernel, that is fine.
<code>ipykernel install</code>	Tells Jupyter about the new environment.

Careful

If you installed anything yourself into the old environment, its name is in `old-environment.txt` in your home folder. Install it again into the new environment with `python -m pip install` and the name, if you still need it. If one of the lines above fails, do not skip it: see *If something goes wrong* below.

Step 5. Check that everything works

```
conda activate darkmatter_cli_env
cd %USERPROFILE%\Research\NSDF-Data\python
python check_environment.py
python -m pytest -q
nsdf-cli version
```

Checkpoint

The check says Environment OK, `pytest` ends with 101 passed, and the last line prints NSDF Dark Matter CLI: 0.6.0 (or the newer version the release notes name). If not, do not guess: copy the output and send it to the project lead.

Step 6. Start Jupyter again

```
conda activate darkmatter_cli_env
cd %USERPROFILE%\Research\NSDF-Data\data_analysis\R76\student
cd yourname
jupyter lab
```

Use the same yourname as in your branch name. If Jupyter asks for a kernel, choose **darkmatter_cli_env**. Open your notebook and run the first cell: it should print Environment OK and then (1517, 4096).

Good to know: the NSDF software changes too

The programs that download and read the data, the `nsdf-cli` command and the `nsdf_dark_matter` library, come from a separate project called **Nexus-DM** (formerly NSDF-SLAC). Its team improves it often, and our analysis depends heavily on it. So:

- **We pin its versions.** `requirements.txt` names the exact version of both packages. The project lead tests each new version here before it goes on the list, so a change upstream cannot quietly change your results. **Do not upgrade them yourself** (no `pip install --upgrade`): you could end up with a mix nobody has tested.
- **You get a new version the usual way.** When the list changes, the release notes say so, the project lead writes to you (“Students, please”), and you follow Step 3 of this guide. The environment check in Step 1 warns you when your versions differ from the list.
- **To see your versions,** type `nsdf-cli version`, and `python -m pip show nsdf-dark-matter nsdf-dark-matter-cli` (look at the Version lines).
- **New commands come with new versions.** Version 0.6.0 added `nsdf-cli config` (the folder that downloads go to), `nsdf-cli download ... --all` and `nsdf-cli prune`; the first analysis guide, Step 2, explains them.
- **If something behaves differently after an update,** send the project lead the output of `nsdf-cli version` and the message you saw. The lead reports problems in the NSDF software to its team; you do not need to.

What the Nexus-DM team changed in each version is in [their release notes](#).

You are done when

- ☐ `python check_environment.py` (in the `python` folder, with the environment on) says Environment OK.
- ☐ `git status` in the NSDF-Data folder shows your own branch and nothing you want to keep is unsaved.
- ☐ `python -m pytest -q` in the `python` folder ends with 101 passed.
- ☐ `jupyter kernelspec list` lists `darkmatter_cli_env`.
- ☐ Your notebook’s first cell prints Environment OK.

If something goes wrong

What you see	What to do
<code>python: can't open file 'check_environment.py' or No such file</code>	You are not in the <code>python</code> folder. Run the <code>cd</code> line of Step 1, and make sure Step 2 finished (the file arrives with the newest changes).
<code>EnvironmentLocationNotFound</code> or “not a conda environment” on <code>conda env remove</code>	There is no environment of that name any more. That is fine: carry on with the next line.
<code>CondaError: Prefix record '...' already exists on conda create</code>	Conda’s download cache has a bad entry. Type <code>conda clean --all -y</code> and then run the <code>conda create</code> line again.

What you see	What to do
conda create complains that the folder exists, or a leftover folder <code>darkmatter_cli_env</code> is in the <code>envs</code> folder	The removal left an empty shell behind. Type <code>conda info --envs</code> to see where the environments live, check that the leftover <code>darkmatter_cli_env</code> folder holds no python file, delete that folder, and run the <code>conda create</code> line again.
Python was not found; run without arguments to install from the Microsoft Store (Windows)	Your environment is not switched on, or it was not created. Type <code>conda env list</code> . If <code>darkmatter_cli_env</code> is not in the list, run the <code>conda create</code> line of Step 4 again. If it is, run <code>conda activate darkmatter_cli_env</code> .
<code>FileExistsError</code> on the <code>ipykernel install</code> line	Jupyter already has a kernel of that name. Run the <code>jupyter kernelspec remove</code> line of Step 4 (the one ending <code>-y</code>), then the <code>ipykernel install</code> line again.
git says CONFLICT	Stop and send the message to the project lead. Do not edit the files.
A text editor opens (a window full of ~ characters, or a message starting Merge branch) after <code>git merge</code>	git is asking for a message for the merge. In the editor (usually Vim): press Esc, type <code>:wq</code> and press Enter. The merge finishes with git's own message.
The kernel is missing in Jupyter	In the environment, run the last two lines of Step 4 again, then restart Jupyter.

Where to get help

Anthony Villano, project lead: anthony.villano@ucdenver.edu

When you write, include the step number, the command you typed, and the last few lines the computer printed. A screenshot helps. Nobody will be annoyed by a question about a step that did not work.

Session Info

Guide version 6, 13 October 2026, Windows edition. Written for beginners; please tell the project lead where a step was unclear, so the guide can be fixed.

Key links

- Release notes: <https://github.com/villano-lab/NSDF-Data/releases>
- Repository: <https://github.com/villano-lab/NSDF-Data>
- The setup guide and the first analysis guide: <https://villano-lab.github.io/NSDF-Data/>