



An Overview of Data Visualizations with PyVista and RAPIDS

Jhamieka Greenwood

Mentor: Thomas Martin

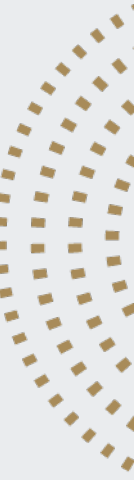
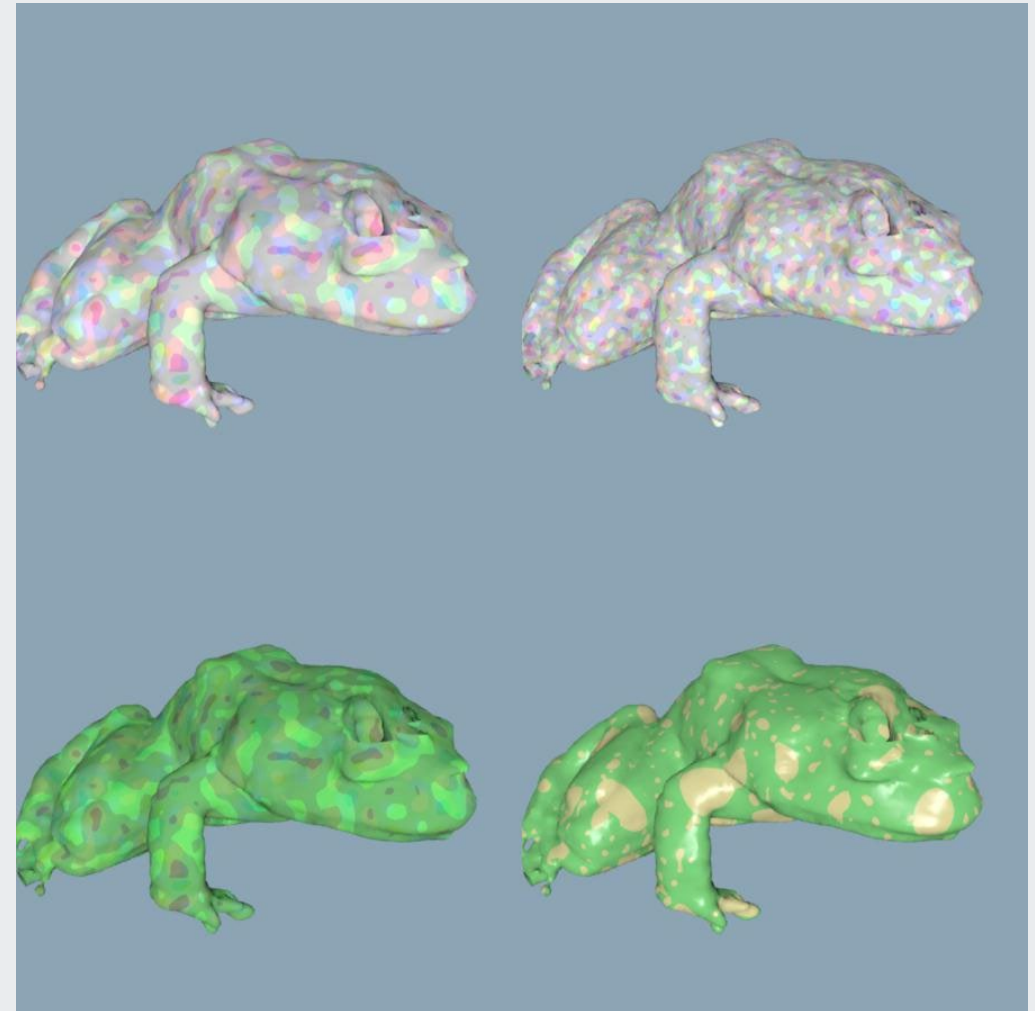


PyVista

- A Python package that provides a concise, well-documented interface to the Visualization Toolkit (VTK)
- PyVista is an easier framework for interactive visualizations than Matplotlib
- It enables researchers to rapidly explore large datasets, communicate their spatial findings, and facilitate reproducibility.
- PyVista further seeks to simplify standard mesh creation and plotting routines without compromising on the speed of the C++ VTK backend.



VTK is an open-source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and 2D plotting.





Installing PyVista



The only prerequisite for installing PyVista is Python itself.

conda

- easier installation using Anaconda to ensure you have the correct version of Python

```
conda install -c conda-forge pyvista
```

pip

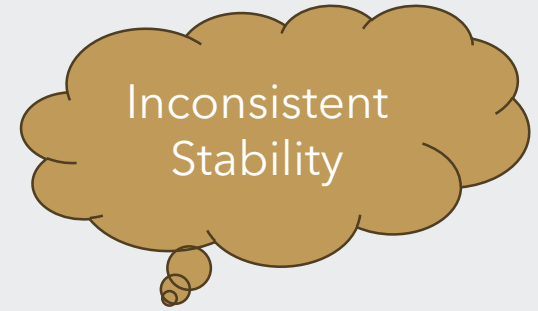
- quicker installation option
- `pip install pyvista[all]` ensures all additional packages needed are installed as well

```
pip install pyvista
```





Installing PyVista Xarray



PITFALLS

Prerequisite for installing PyVista Xarray is Python and PyVista.

conda

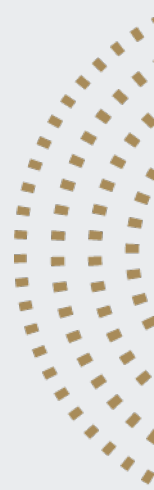
- easier installation using Anaconda to ensure you have the correct versions of Python and PyVista

```
conda install -c conda-forge pyvista-xarray
```

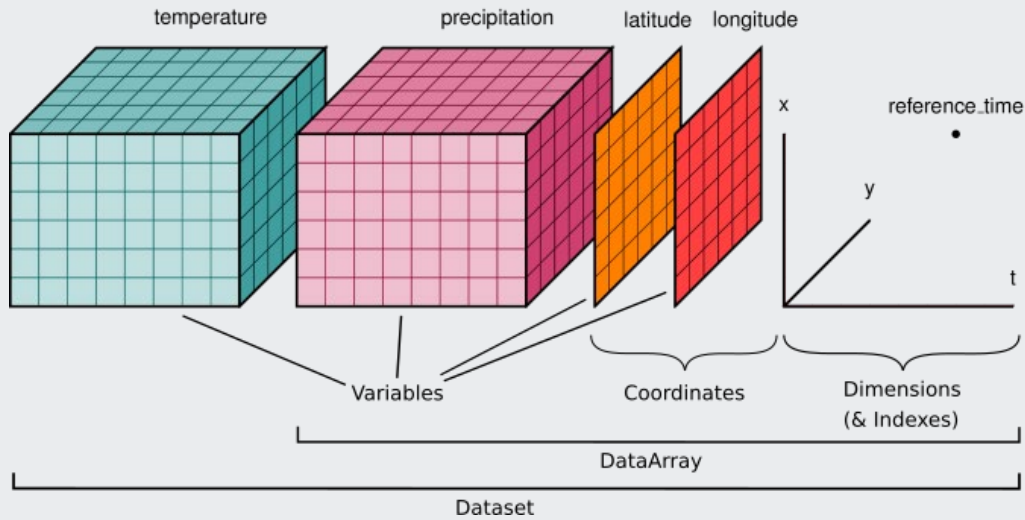
pip

- quicker installation option

```
pip install pyvista-xarray
```



Xarray Datasets



- Datasets are dictionary-like containers of data arrays mapping variable names to a data array.
- Inside a data array we have:
 - Dimensions which correspond to the axes of the data
 - Coordinate variables used for indexing and alignment
 - Arbitrary attributes which is a dictionary of Python objects (strings, integers, list, dictionaries)

```
xarray.Dataset
```

– Dimensions: (lat: 25, time: 2920, lon: 53)

▼ Coordinates:

lat	(lat)	float32	75.0 72.5 70.0 ... 20.0 17.5 15.0	📄 🗄
lon	(lon)	float32	200.0 202.5 205.0 ... 327.5 330.0	📄 🗄
time	(time)	datetime64[ns]	2013-01-01 ... 2014-12-31T18:00:00	📄 🗄

▼ Data variables:

air	(time, lat, lon)	float32	241.2 242.5 243.5 ... 296.2 295.7	📄 🗄
-----	------------------	---------	-----------------------------------	-----

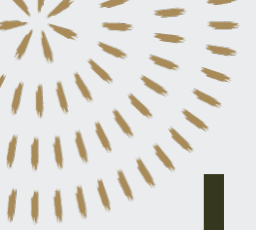
▼ Indexes:

lat	PandasIndex	🗄
lon	PandasIndex	🗄
time	PandasIndex	🗄

▼ Attributes:

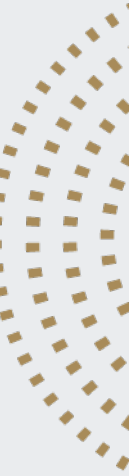
Conventions :	COARDS
title :	4x daily NMC reanalysis (1948)
description :	Data is from NMC initialized reanalysis (4x/day). These are the 0.9950 sigma level values.
platform :	Model
references :	http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.html

Example dataset from the National Center for Environmental Prediction



Let's Get Into Some Examples

- 3 Notebooks each with their own datasets
- Examples comparing Matplotlib and PyVista

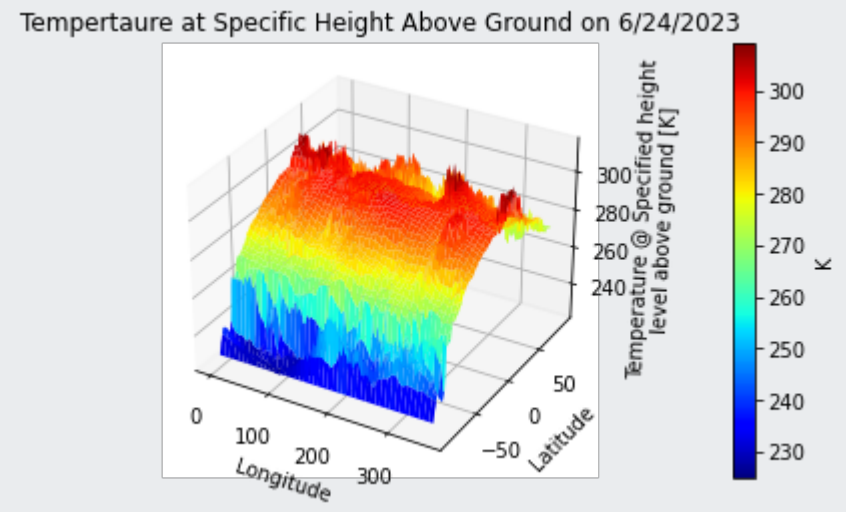
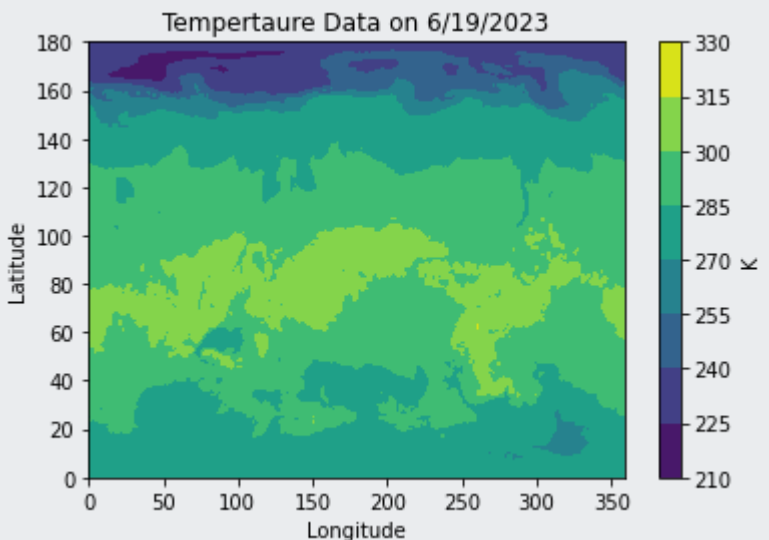


Example 1: Plotting with Temperature Data

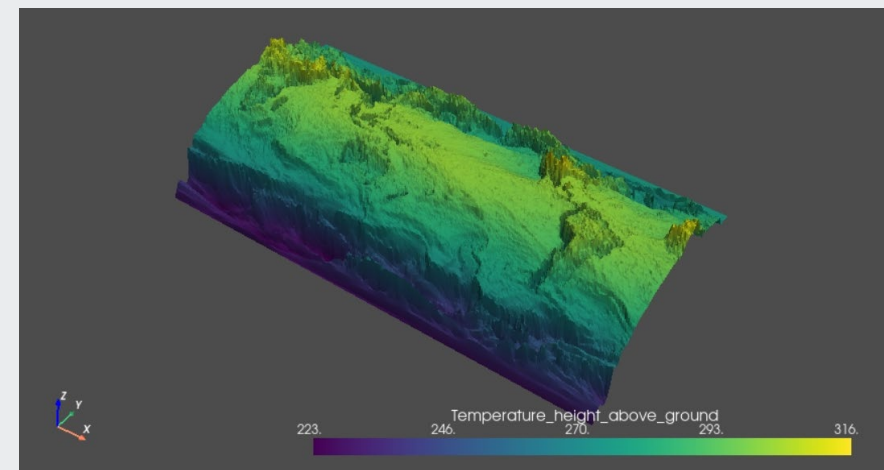
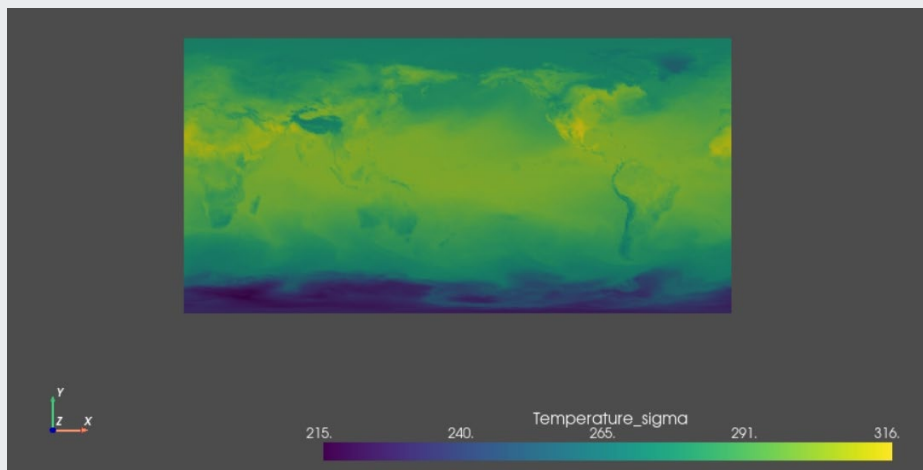
2D Plotting

3D Plotting

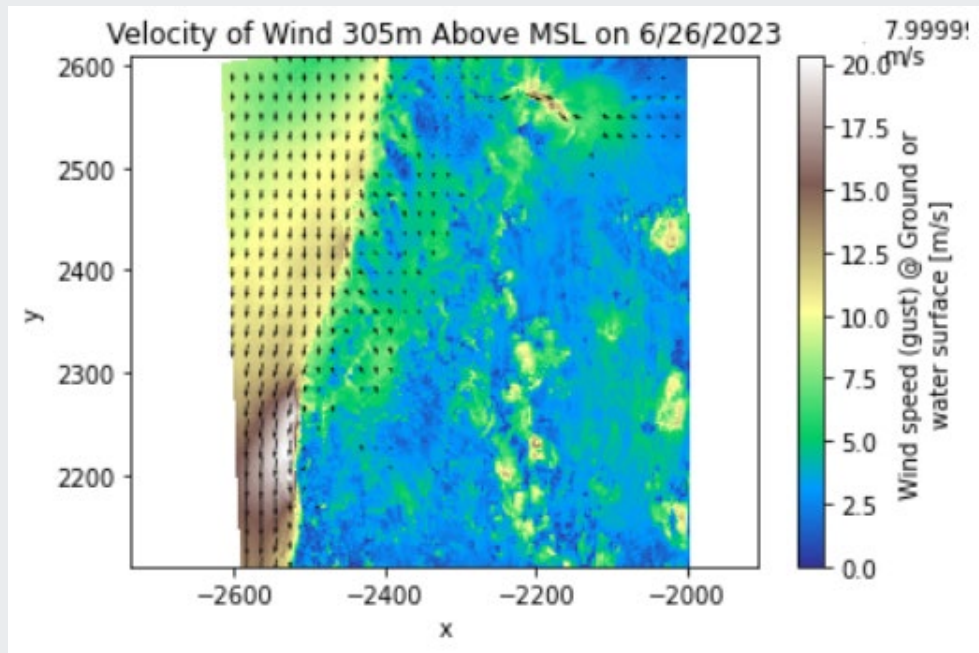
Matplotlib



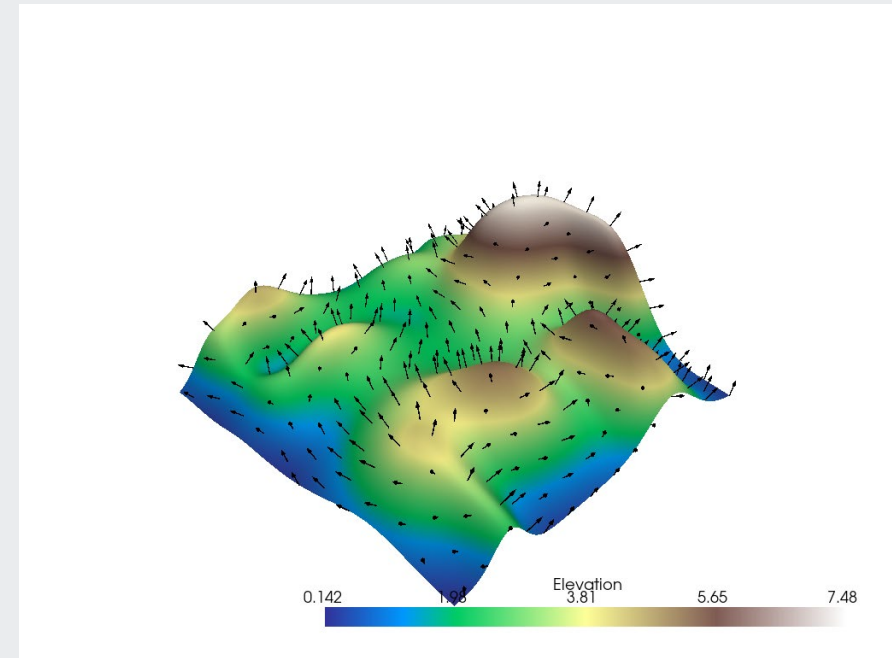
PyVista



Example 2: Vector Plotting with Wind Data

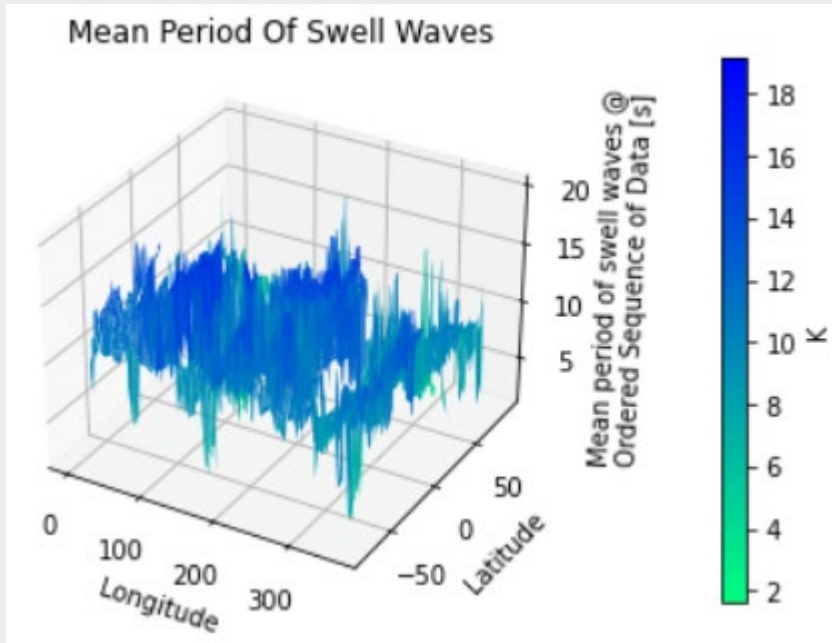


Matplotlib

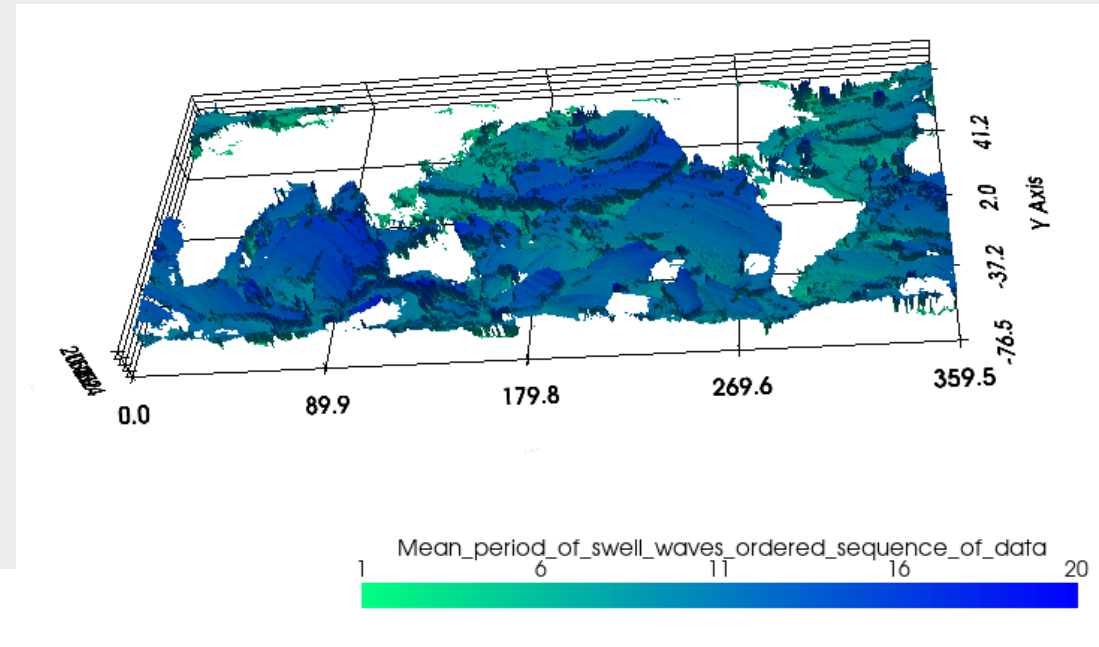


PyVista

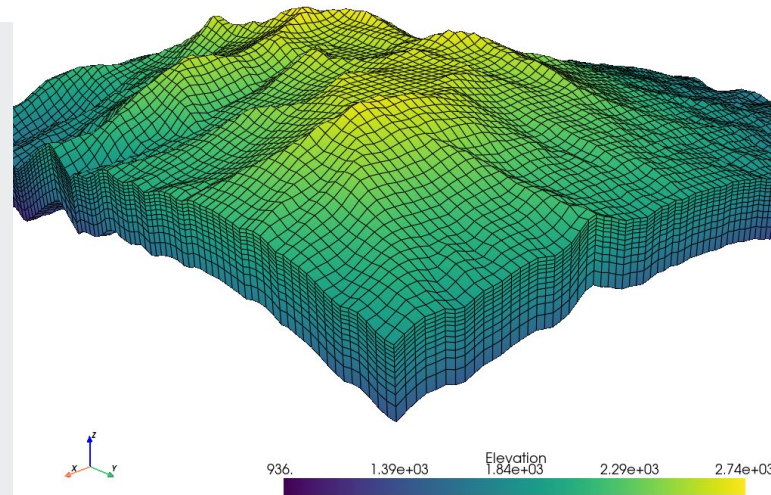
Example 3: 3D Representations of Topography



Matplotlib



PyVista





Matplotlib vs. PyVista


Matplotlib

- Interactive but difficult to implement
- Easy to work with overall

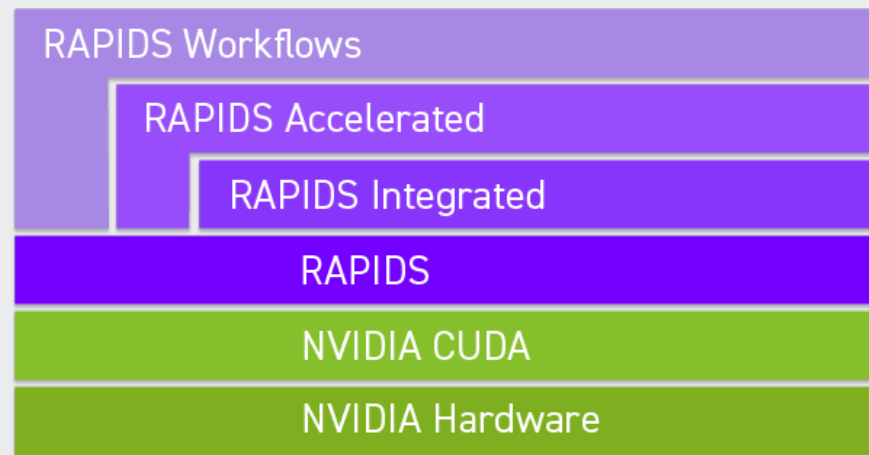
PyVista

- Interactivity built-in for every plot
- Fairly easy but some ideas maybe difficult to implement

	Example 1 (2D/3D)	Example 2	Example 3
Matplotlib	183 ms/507 ms	1.07 s	606 ms
PyVista	941 ms/213 ms	459 ms	815 ms/2.474 s



Rapids



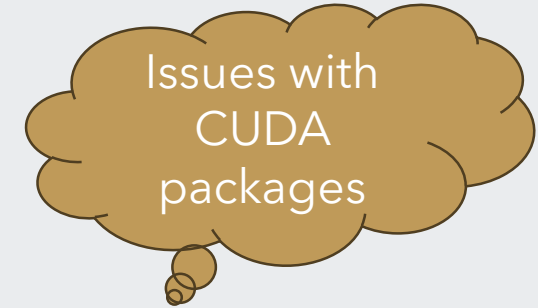
- a collection of open-source software libraries and APIs that gives you the ability to execute end-to-end data science and analytics pipelines entirely on NVIDIA GPUs using familiar PyData APIs and parallelized by CUDA
 - Faster data frame processing with cuDF (similar to pandas)
 - Faster machine learning with cuML (similar to scikit-learn)
 - Faster image processing with cuCIM (similar to scikit-image)
 - Faster spatial analytics with cuSpatial (similar to geoPandas)



Installing Rapids

Prerequisite for installing Rapids:

- NVIDIA Pascal or better GPU with compute capability 6.0 and above
- Ubuntu 20.04 or 22.04, CentOS 7, Rocky Linux 8, or **WSL2 on Windows 11**
- Recent CUDA version and NVIDIA driver pairs



PITFALLS

conda

```
> conda create -n rapids-23.06 -c rapidsai -c conda-forge -c nvidia  
rapids=23.06 python=3.10 cudatoolkit=11.8
```

pip

```
> pip install cudf-cu11 dask-cudf-cu11 -extra-index-  
url=https://pypi.nvidia.com  
> pip install cuml-cu11 -extra-index-url=https://pypi.nvidia.com  
> pip install cogrph-cu11 -extra-index-url=https://pypi.nvidia.com
```

Rapids Alternative Installation

- Set up script installs
 1. Updates gcc in Colab
 2. Installs Conda
 3. Install RAPIDS' current stable version of its libraries, as well as some external libraries including:
 1. cuDF
 2. cuML
 3. cuGraph
 4. cuSpatial
 5. cuSignal
 6. BlazingSQL
 7. xgboost
 4. Copy RAPIDS .so files into current working directory, a necessary workaround for RAPIDS+Colab integration.

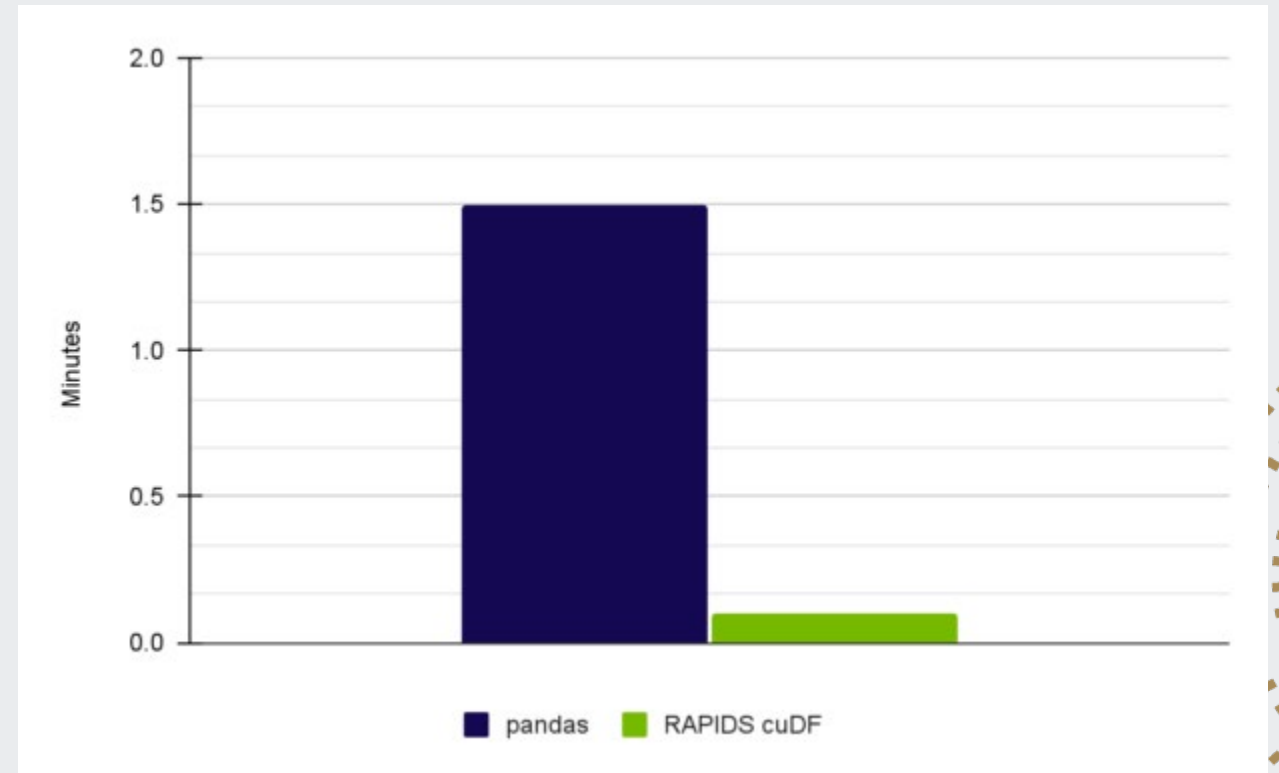


Good alternative
but installation
takes a while

**Google
Colaboratory**

Key Takeaways

- Realistic data can be large and complex. Having software libraries and API that can keep up with end-to-end data science is crucial.
- Compared to pandas, RAPIDS can provide a 15x speedup in a complex workload.
- However, pandas is a well-maintained library that works well and fast with smaller datasets.



<https://developer.nvidia.com/blog/accelerated-data-analytics-speed-up-data-exploration-with-rapids-cudf/>



References

- PyVista Installation <https://docs.pyvista.org/version/stable/getting-started/installation.html>
 - PyVista Xarray Installation <https://github.com/pyvista/pyvista-xarray>
 - Rapids Installation <https://rapids.ai/>
- 



Thank You

- Thomas Martin and Drew Camron
 - Julien Chastang, Ana Espinoza, & the Science Gateway Team for Jupyterhub Access
 - UCAR and Unidata Staff
- 