Project 3: Group 3–Notebook Reproduction

Teams attempt to reproduce [the quickstart demo notebook](https://github.com/leap-stc/ClimSim/blob/main/demo_notebooks/quickstart_example.ipynb).

Jason Cho
Noreen Mayat
Yihan Zhang
Miao Zhang
Stephanie Shu

Choosing a Computing Environment:

Each of our local computers had a memory limit of 8GB; each time we tried to load the files in a Jupyter Notebook, for example, we began experiencing "out of memory" errors. To address this issue, we turned to Google Colab, which has up to 13GB of RAM. The data files are 12GB, so this worked to initially load our data. Here's the workflow and steps taken to reproduce the notebook:

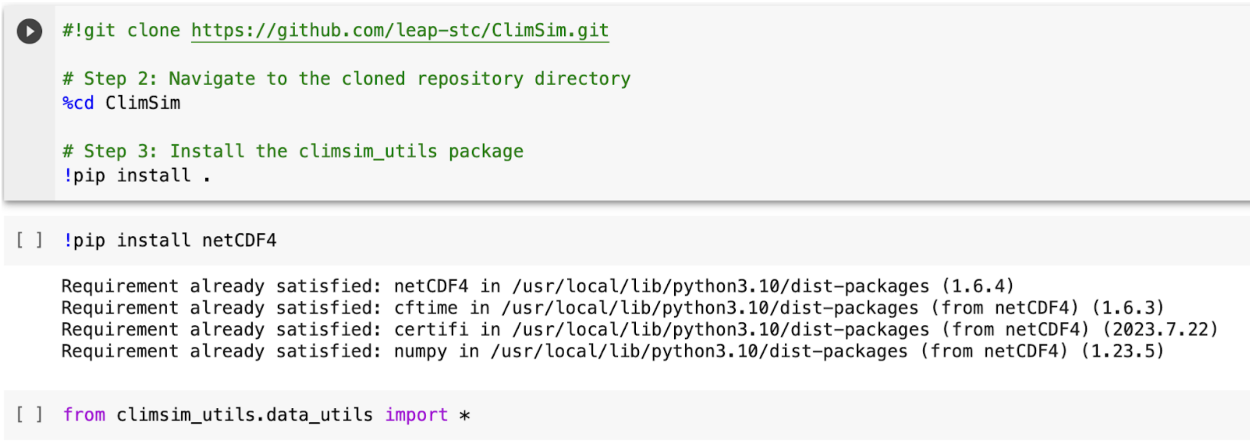
Steps taken for the successful reproduction:

Step1

- Followed the link provided and downloaded the datasets to the local.
- Uploaded the data to Google drive for accessing to Colab.

Step2

- Forked the ClimSim repo and installed the `climsim_utils` python tools for our project. Installed the dependencies command given in the *quickstart_example.ipynb*.



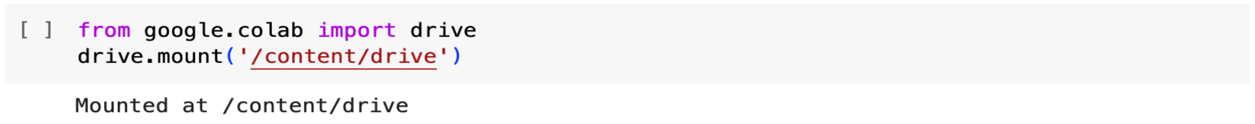
```
#!git clone https://github.com/leap-stc/ClimSim.git

# Step 2: Navigate to the cloned repository directory
%cd ClimSim

# Step 3: Install the climsim_utils package
!pip install .
```

```
!pip install netCDF4
```

```
Requirement already satisfied: netCDF4 in /usr/local/lib/python3.10/dist-packages (1.6.4)
Requirement already satisfied: cftime in /usr/local/lib/python3.10/dist-packages (from netCDF4) (1.6.3)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from netCDF4) (2023.7.22)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from netCDF4) (1.23.5)
```

```
from climsim_utils.data_utils import *
```

- Mounted the Drive to access the uploaded dataset in our Google Drive. Then, we were able to read, process, and write back data without any hitches.

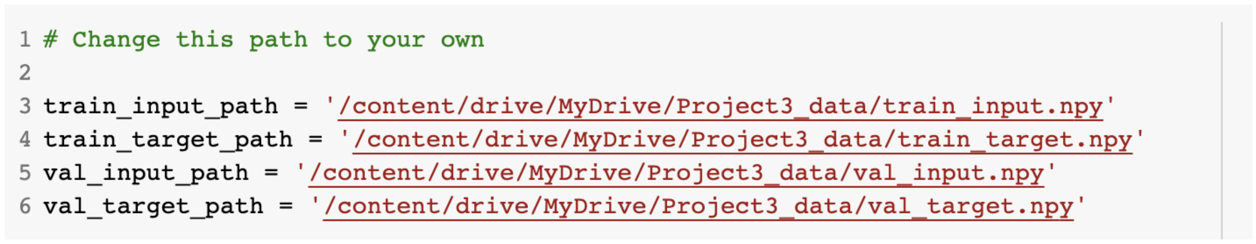


```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Step3

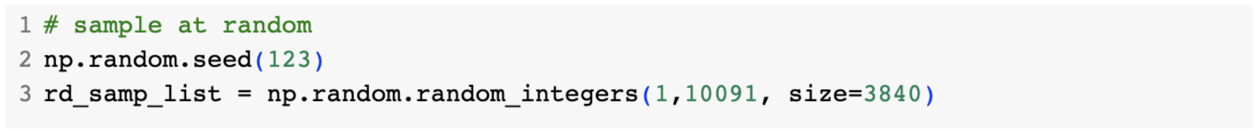
- Defined paths that point to our datasets stored on Google Drive and used these paths to access the datasets.



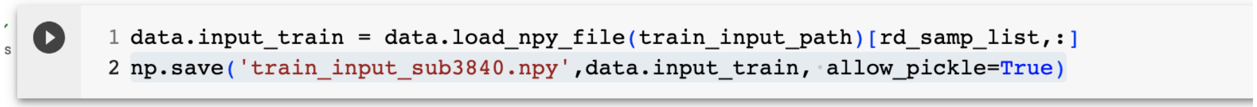
```
# Change this path to your own

train_input_path = '/content/drive/MyDrive/Project3_data/train_input.npy'
train_target_path = '/content/drive/MyDrive/Project3_data/train_target.npy'
val_input_path = '/content/drive/MyDrive/Project3_data/val_input.npy'
val_target_path = '/content/drive/MyDrive/Project3_data/val_target.npy'
```

Step4

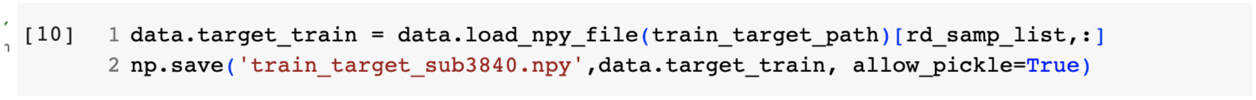
- Generated 3849 random integers and used them as indices to select a subset of datasets.
- Saved the subsetted datasets in the current working directory as new Numpy binary files. Setted *allow_pickle* = *True* to allow loading pickled object arrays stored in our .npy files.



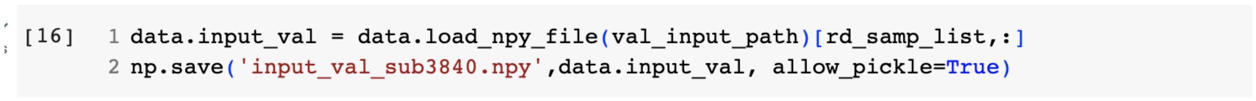
```
# sample at random
np.random.seed(123)
rd_samp_list = np.random.random_integers(1,10091, size=3840)
```



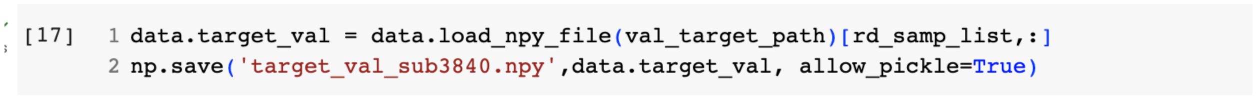
```
data.input_train = data.load_npy_file(train_input_path)[rd_samp_list,:]
np.save('train_input_sub3840.npy',data.input_train, allow_pickle=True)
```



```
data.target_train = data.load_npy_file(train_target_path)[rd_samp_list,:]
np.save('train_target_sub3840.npy',data.target_train, allow_pickle=True)
```



```
data.input_val = data.load_npy_file(val_input_path)[rd_samp_list,:]
np.save('input_val_sub3840.npy',data.input_val, allow_pickle=True)
```



```
data.target_val = data.load_npy_file(val_target_path)[rd_samp_list,:]
np.save('target_val_sub3840.npy',data.target_val, allow_pickle=True)
```

Step5

- Trained with const model and multiple linear regression model in the *quickstart_example.ipynb* on our data.

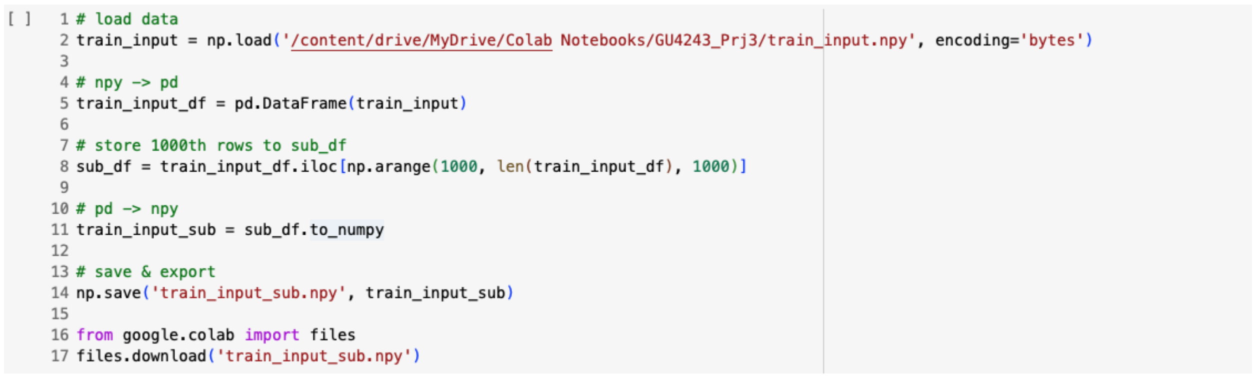
Step6

- Evaluated on both validation and scoring data without any hyperparameter tuning.
- Created plots to better understand each of our metrics: `'MAE'`, `'RMSE'`, `'R2'`, `'bias'`.

Main Issues Encountered:

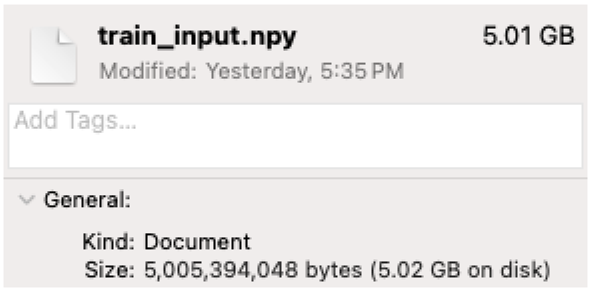
1. Challenge Identification and Initial Strategy:

To address the challenges we faced with large data files, we turned to Google Colab. This allows everyone to process them on their local machines. However, even Google Colab has its limitations, and only supports files up to 13 GB. Although Colab may have initially allowed us to load all our data, we anticipated experiencing space issues again upon trying to run our model. Thus, we had to again devise another strategy for reduction. Instead of simply selecting the initial 'n' rows, which might lead to a loss of variability, we chose every 1000th row during the subsetting process. Below is the code used to achieve this subsetting:



```
[ ]
# load data
train_input = np.load('/content/drive/MyDrive/Colab Notebooks/GU4243_Prj3/train_input.npy', encoding='bytes')

# npy -> pd
train_input_df = pd.DataFrame(train_input)

# store 1000th rows to sub_df
sub_df = train_input_df.iloc[np.arange(1000, len(train_input_df), 1000)]

# pd -> npy
train_input_sub = sub_df.to_numpy

# save & export
np.save('train_input_sub.npy', train_input_sub)

from google.colab import files
files.download('train_input_sub.npy')
```

2. Data Reduction and Storage: Using this approach, we reduced the size of the original dataset, 'train_input', from 5.01 GB to 5.1 MB and renamed each file by attaching '_sub' at the end.

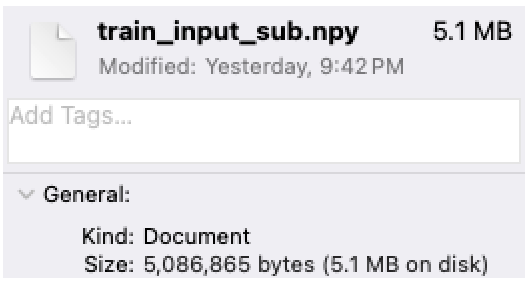


train_input.npy 5.01 GB
Modified: Yesterday, 5:35 PM

Add Tags...

General:

Kind: Document
Size: 5,005,394,048 bytes (5.02 GB on disk)



train_input_sub.npy 5.1 MB
Modified: Yesterday, 9:42 PM

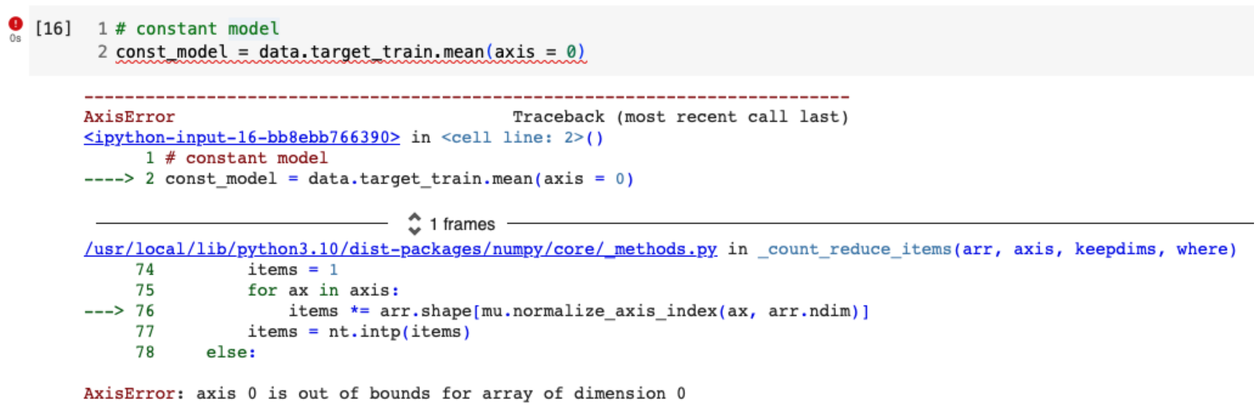
Add Tags...

General:

Kind: Document
Size: 5,086,865 bytes (5.1 MB on disk)

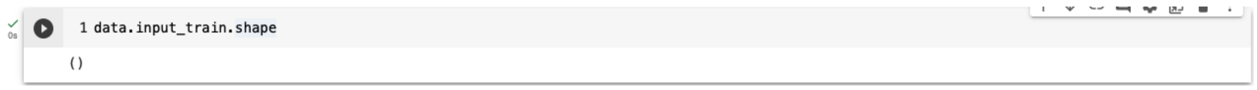
After subsetting our data, we uploaded the subsetted files to our joint Colab on Google Drive.

3. Model Error: However, when attempting to run even our newly subsetted files through our consistent model, we encountered a dimensionality issue with the subsetted array. Below is the error we encountered:



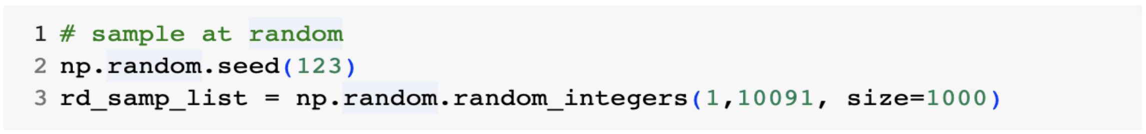
```
[16]  # constant model
      const_model = data.target_train.mean(axis = 0)

---------------------------------------------------------------------------
AxisError                                 Traceback (most recent call last)
<ipython-input-16-bb8ebb766390> in <cell line: 2>()
      1 # constant model
----> 2 const_model = data.target_train.mean(axis = 0)

                         1 frames
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py in _count_reduce_items(arr, axis, keepdims, where)
     74         items = 1
     75         for ax in axis:
---> 76             items *= arr.shape[mu.normalize_axis_index(ax, arr.ndim)]
     77         items = nt.intp(items)
     78     else:

AxisError: axis 0 is out of bounds for array of dimension 0
```

We figured out that the shape of the subsetted array turned out to have nothing in it, instead of the number of rows from the subsetted amount.

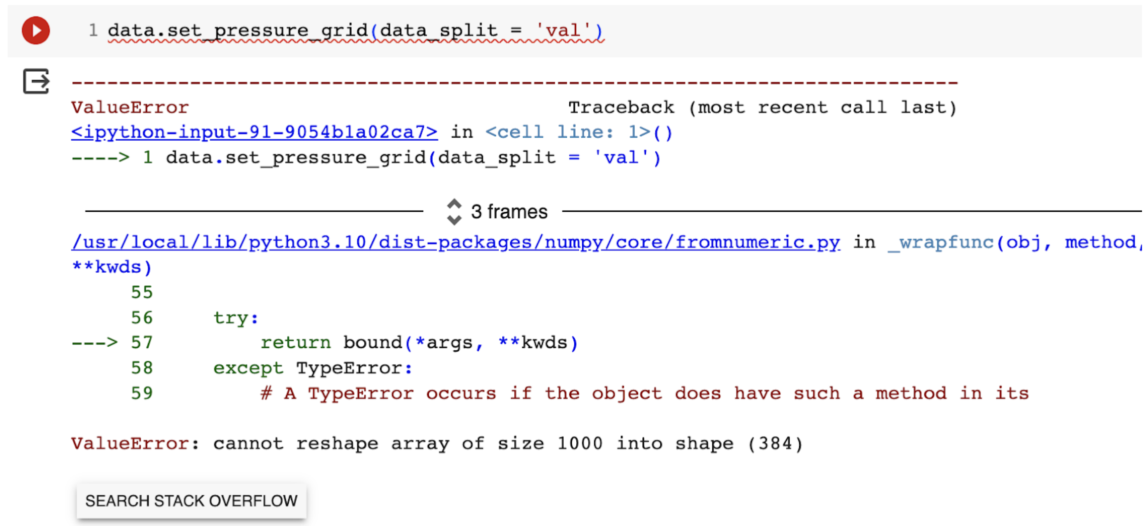


```
data.input_train.shape

()
```

4. Sampling Error:
With the help of TA Claire, we used another method to subset the data. At first we subsetted the data set in size of 1000, but then we had an issue of 'cannot reshape array of size 1000 into shape (384)'.

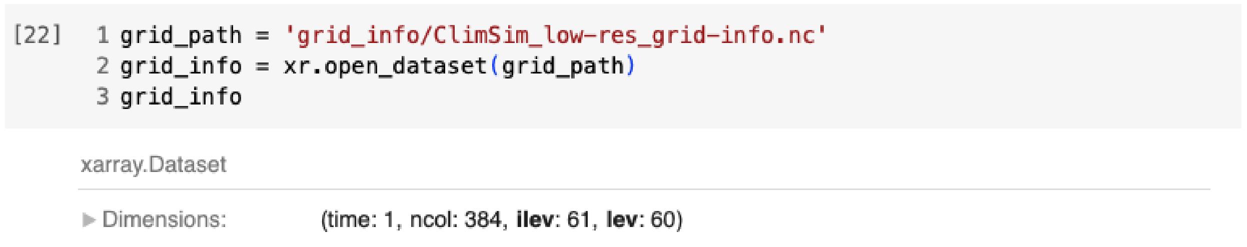


```
# sample at random
np.random.seed(123)
rd_samp_list = np.random.random_integers(1,10091, size=1000)
```



```
data.set_pressure_grid(data_split = 'val')

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-91-9054b1a02ca7> in <cell line: 1>()
----> 1 data.set_pressure_grid(data_split = 'val')

                         3 frames
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py in _wrapfunc(obj, method,
**kwds)
     55 
     56     try:
---> 57         return bound(*args, **kwds)
     58     except TypeError:
     59         # A TypeError occurs if the object does have such a method in its

ValueError: cannot reshape array of size 1000 into shape (384)
```

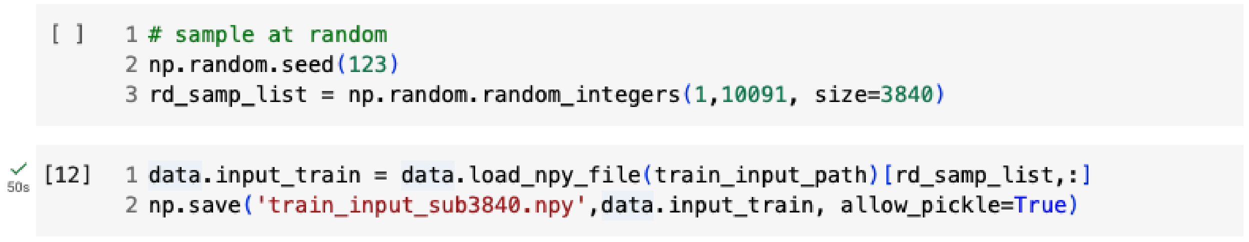
Resolving Issue 4: To align the subsetted data with each segment of the grid, Claire suggested that the subset size needed to be proportional to the grid size, which is 384. The code below indicates that the grid has 384 columns.



```
[22]  grid_path = 'grid_info/ClimSim_low-res_grid-info.nc'
      grid_info = xr.open_dataset(grid_path)
      grid_info
```

```
xarray.Dataset

▶ Dimensions:        (time: 1, ncol: 384, ilev: 61, lev: 60)
```

To address the dimensionality issue we encountered, we resized the subset to a size of 3840. This was achieved by first sampling random integers between 1 and 100091. By applying this approach to each dataset, we ensured that all original data was appropriately subsetted and could be executed on all team members' local computers. This method effectively resolved the problem we faced.



```
[ ]   # sample at random
      np.random.seed(123)
      rd_samp_list = np.random.random_integers(1,10091, size=3840)
```

```
50s [12]  data.input_train = data.load_npy_file(train_input_path)[rd_samp_list,:]
          np.save('train_input_sub3840.npy',data.input_train, allow_pickle=True)
```

Estimated time used for the full reproduction process:

Downloading Data: 20 minutes
Installing Dependencies and Instantiating Class: 5 minutes
Subsetting Datasets: 3 hours
Loading Datasets: 10 minutes
Training Model: 5 minutes
Evaluating on Validation Data: 2 minutes
Evaluating on Scoring Data: 2 minutes