Data files are in /data/ and follow the naming convention: ‘compiled\_run\_{flux}\_a\_b\_v6\_varz.pkl’ with flux:

‘CO2’: Gross CO2 captured (tCO2)

‘grem’: Mining/Grinding Emissions (tCO2)

‘grco’: Mining/Grinding Costs ($)

‘trem’: Transport Emissions (tCO2)

‘trco’: Transport Costs ($)

‘spem’: Spreading Emissions (tCO2)

‘spco’: Spreading Costs ($)

‘rock’: Rock applied (t rock)

‘pH’: Soil pH (top 10 cm)

‘P\_mass’: Phosphorus Release (kg/ha)

‘K\_mass’: Potassium Release (kg/ha)

‘feem\_P’: Offset Phosphorus Emissions (tCO2)

‘feem\_K’: Offset Potassium Emissions (tCO2)

‘feco\_P’: Offset Phosphorus Costs ($)

‘feco\_K’: Offset Potassium Costs ($)

Negative fluxes denote carbon capture/costs saved.

Each pickle file (\*.pkl) contains a Python dictionary. The dictionary’s keys are unique points id. The values of the dictionary is a dictionary with keys and values:

‘nam’: Crop Name on the point. (List[str])

‘nam\_l’: Name of the US state that the point is located at. (str)

‘lind’: Location of the point as a linear index of an array with dimensions [768, 1152]. This is the native global extend of CESM output and the linear index represents the gridcell of the array. See file ‘landmask\_0.23x0.31.nc’. Some points share the same linear index as they are on the same gridcell of the array but have a different crop type. (np.int64)

‘ind’: Location of the point with indices of an array with dimensions [768, 1152]. This is the native global extend of CESM output. See file ‘landmask\_0.23x0.31.nc’. Some points share the same indices as they are on the same gridcell of the array but have a different crop type. (np.array[np.int64, np.int64])

‘ar’: Area of the point in hectares. (np.float64)

‘pH0’: Initial soil pH at different depths

‘scen’: Indices of the two scenarios. (np.array[np.ind64, np.ind64])

‘lamb’: *Ignore*

‘lambi’: *Ignore*

‘p80’: The particle size p80 in um (np.array[np.int64])

‘bas’: The name of the state from which the point receives basalt for each scenario. ‘Nan’ value if the point does not receive basalt in the scenario. (List[str, str])

‘yy’: The year when basalt is first applied at the point for each scenario. 0 value if the point does not receive basalt in the scenario. (np.array[np.int64, np.int64])

‘{}\_a\_b’: Pandas Dataframe. ‘Date’ is the index of the dataframe with annual values (‘YYYY-01-01’). The dataframe has 6 value columns with naming convention: S{scenario\_id}\_U1\_N{uncertainty\_id}\_p100. There are 2 scenario ids ([1, 2]) and 3 uncertainty ids ([1=low, 2=mean, 3=high]). (pd.DataFrame)

A Python file is also provided (‘aa\_paper\_data.py’) in /src/ which can be used to generate netCDF files of the fluxes. It contains the class PaperData with instance variables: scen\_id (scenario id), unc\_id (uncertainty id) and flx\_name (flux name). The user can input different values of the instanced variables to produce netCDF files with the flux for the particular case. The netCDF file will have dimensions [768, 1152, 51] with 51 the years of the run 2020-2070. For more info, see ‘aa\_paper\_data.py’.