The RUSLE model on Google Earth Engine

First steps

- The RUSLE model calculates yearly average **soil erosion rates** using data on precipitation, landcover, soil, topography and management
- Objective: Implement RUSLE on GEE at a high spatial resolution, and analyze the **effects of terracing** on soil erosion reduction using the new high-resolution terracing map of China
- Output: Paper on terracing

Steps needed to make erosion model run on GEE

1. Modify model to take high-resolution data on ndvi, topography, landcover and precipitation (~ 30m) as input
2. Write code to extract data from satellite images
3. Modify source code of erosion model to make calculations on the GEE platform and display resulting erosion rates on google earth

**Steps 1 and 2 are already done by VN*

Model structure and files

1. The erosion model consists out of 3 scripts:
   a. Input_File.py : derives input data from GEE and exports them to tif files which are then read and output is transformed to arrays. The needed processing is done (reprojection, selection of test region etc.)
   b. Adj_RUSLE.py: This is the main model, where the RUSLE factors are calculated based on different methods from literature
   c. Output_File.py: The run script
2. In addition the erosion model has 2 functions that need to be imported to make the model run:
   a. erosivity.so and erosivity.f90
   b. scaling_func.py

Model structure and files (2)

- The model is coded in python
- Only the erosivity.f90 is written in fortran code and then transformed into a python module with: f2py -c -c erosivity erosivity.f90; this needs to be done on the local machine, not sure how this can work on GEE
- Also in the current setup the reprojection of the RUSLE factors (C, K and R) from degrees to meters (see Output_File.py) is done on the local machine. This needs to be modified when model runs on GEE
- All the model code files are available on github:
- The data to run the model has been uploaded to zenodo repository

Next steps:

1. Treat each RUSLe factor separately
2. Find a way to do array calculations directly on the extracted satellite images (writing and executing functions on GEE using Python API)
3. Find a way to use f2py and functions like reprojection on GEE
4. Display results as an image on GEE