# **USER MANUAL: Secondary Programs**

The following manual is for the programs RenameFiles, RasterOrganization and Temperature Sensitivity Analysis. These programs were programmed specifically for the Dirt-X input rasters, but can be modified for future projects and different raster.

# RenameFiles

**Program location:** Y:\Abt1\hiwi\Oreamuno\GIS\03\_R\_Factor\Python\Python Programs\Raster Organization

**Objective**: Its main purpose is to change the name of the Dirt-X raster files, in order for the name to only correspond to the date of the precipitation or temperature measurement, and transform the file to a .txt file, so the program “RasterManipulation\_2” can read the files.

## Input data:

* *Input\_path*= folder location where the precipitation or temperature rasters are found
  + Files don’t need to be in .txt form
* *savepath*= folder location where the .txt files with the new names will be saved

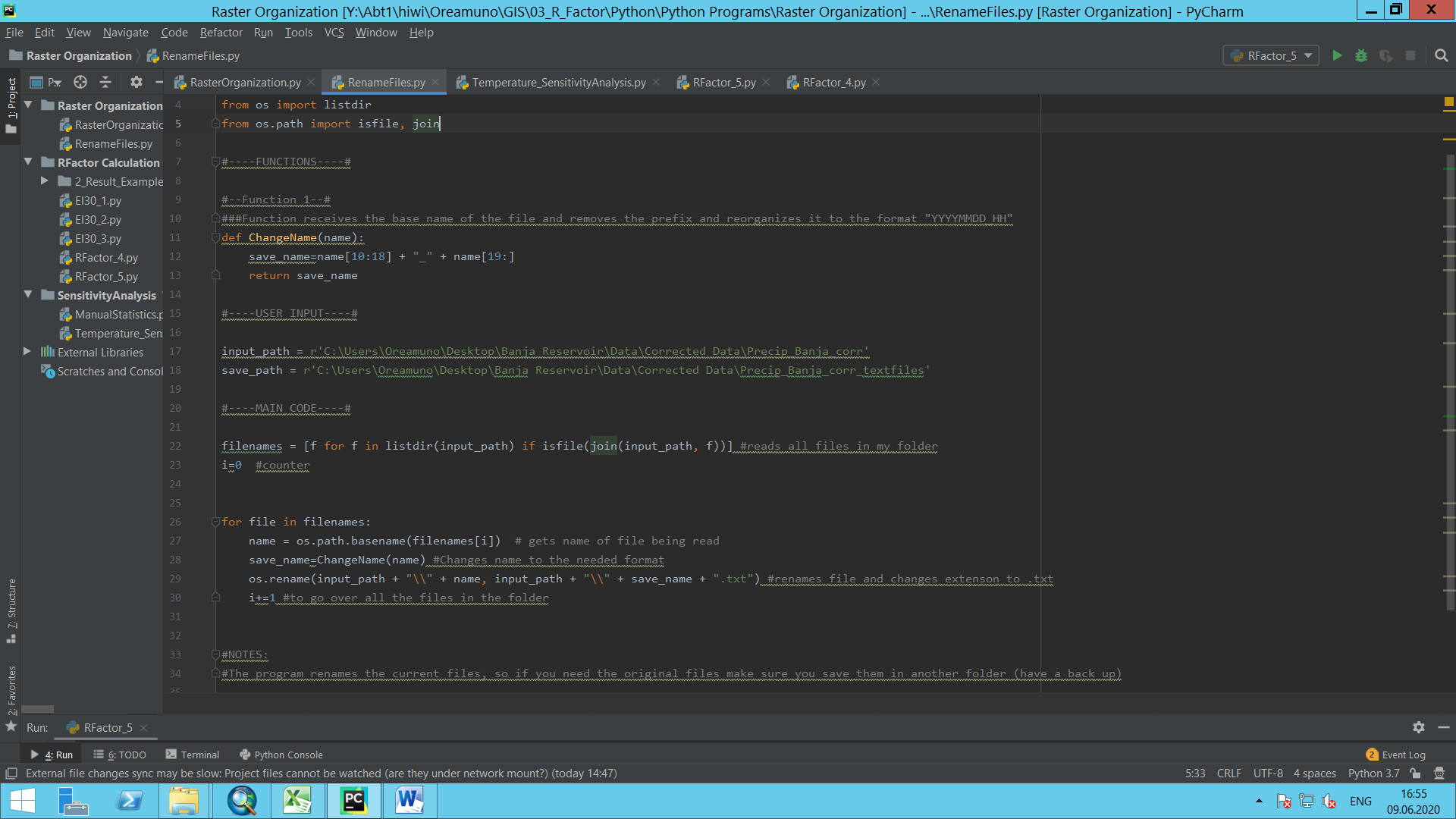


Figure 1. RenameFiles program

## Output data:

“.txt” files with the name corresponding to the date of the precipitation record with the following format: “YYYYMMDD\_HH” (year,month,date\_hour) (see Figure 2).

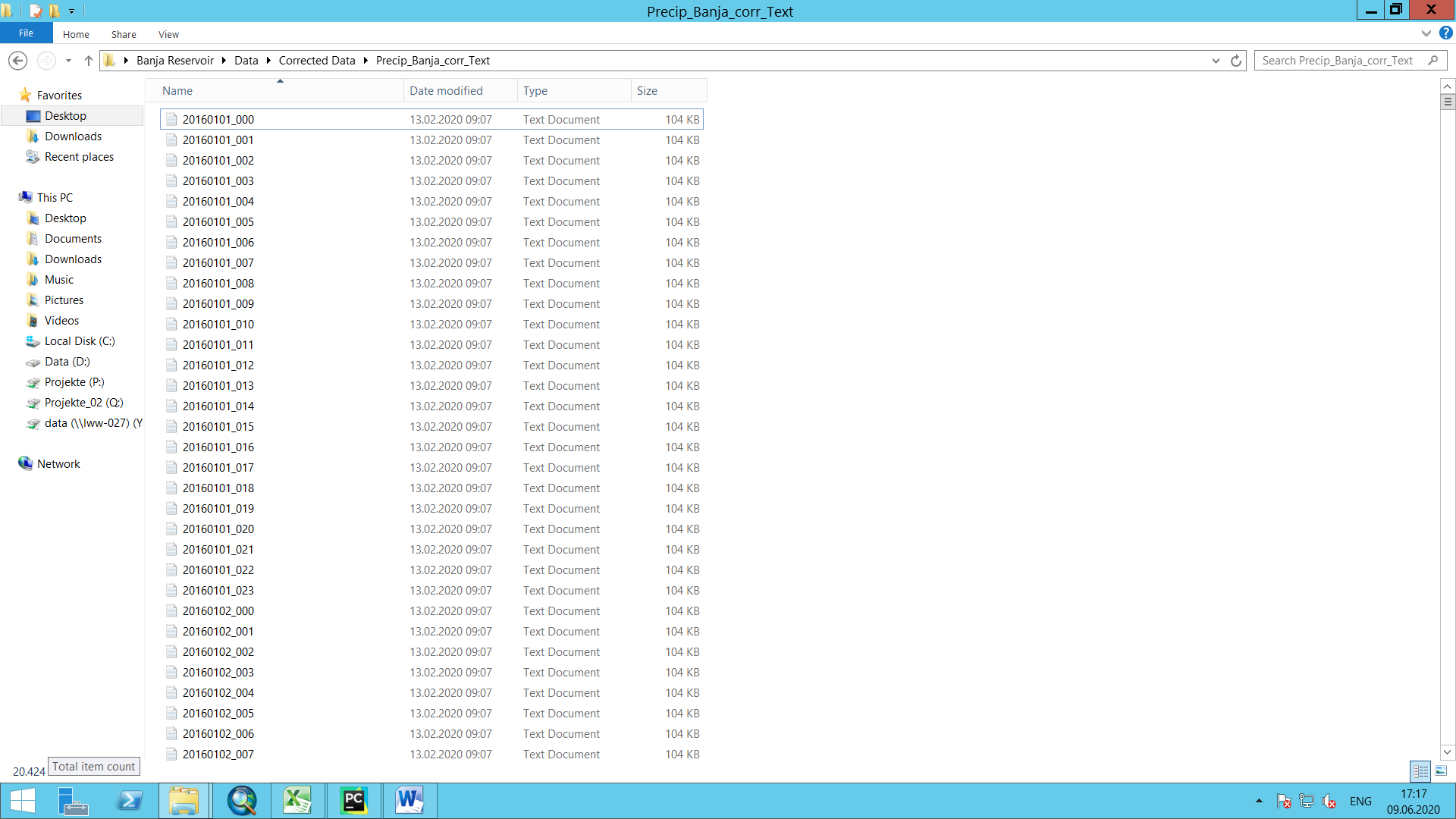


Figure 2. File name format

## Main functions in the program

### Main loop:

* Loop iterates through all the files in the input file folder path (*path*) and extracts the name of the file.
* Calls function **ChangeName,** which return the new file name, with the format yyyymmdd\_hh
* Assigns the name to the input file and saves it as a .txt file in the *savepath* folder.

### ChangeName Function

* Receives the name of the original file, which is in the format “precbanja\_ yyyymmdd.0hh” or “tempbanja\_yyyymmdd.0hh”.
* Extracts the positions 10 to 17 (corresponding to the year month and day (yyyymmdd), adds a subscript “\_” and then adds the positions 19 to the end (corresponding to the hours). This new name (without the prefix and eliminating the “.” In the name) is saved in a variable and returned as the new file name.

## Additional Information

Depending on the initial file name, the **ChangeName** function can be modified to fit any original file name and transform it to another. It must be taken into consideration that the program RasterManipulation\_2 only accepts file names with the format yyyymmdd\_hhh.

# Temperature\_SensitivityAnalysis

This program does the same calculations and receives the same input parameters as **RFactor\_5**, and does a sensitivity analysis for different temperature thresholds. The following descriptions include only additional inputs, outputs, variables and functions with regards to the **RFactor\_5** program.

**Location:** Y:\Abt1\hiwi\Oreamuno\GIS\03\_R\_Factor\Python\Python Programs\SensitivityAnalysis

**Objective:** Calculates the R factor on a per storm and per month basis for each cell in the original Raster, for each temperature threshold value within a user-specified range and generates an output raster for each month and each temperature threshold value with the corresponding R factors for each cell. The resulting Rasters can be opened in GIS program in order to be used in the RUSLE equation.

## Input data:

### Storm Event Input:

* *time\_interval*: User must input the time discretization of the precipitation records. This input will be checked with **one** of the input files. If the input value doesn’t coincide with input files, then an error is produced.
* *min\_6Hours*: minimum precipitation value (in mm) in a 6 hour time step that must be met in order to either start a storm event or decide when the event ends. If this value is set to 0, a storm will start when a precipitation record is greater than 0, and will end when, in a 6 hour window, the total precipitation is 0. Panagos et al, (2015) sets this value at 1.27 mm.
* *min\_P*: minimum total storm precipitation (in mm) in order to consider the given storm as erosive. If the value is set to 0, all storms will be considered, regardless of the total precipitation value. Panagos et. Al (2015) sets this value as 12.7 mm.
* *CF*: If the user input time interval is equal to 60 min (hourly discretization of data), the user can choose whether or not to consider an R Factor conversion factor, to take into consideration the error induced by an hourly discretization. **True** indicates the program to use the conversion, **False** to ignore the conversion factor.

### Temperature Input

* *min\_temp*: minimum temperature threshold (in °C) within the temperature range the user wants to test
* *Max\_temp*: minimum temperature threshold (in °C) within the temperature range the user wants to test

### Original Raster Data

This data must be modified if rasters different from the dirt-X rasters are being used. The data correspond to the coordinates (location) and cell size of the rasters being used. This information is the same for all original input rasters (precipitation and temperature) being used.

* *original\_rows*: number of rows in the original precipitation and temperature raster files
* *original\_columnss*: number of columns in the original precipitation and temperature raster files
* *xllcorner*: X coordinate of the origin of the original precipitation and temperature raster files
* *yllcorner*: Y coordinate of the origin of the original precipitation and temperature raster files
* *cellsize*: cell size of the original precipitation and temperature raster files

### Directory Paths

* *path*: folder directory where the .csv files for each raster cell is located. The files must be comma-delimited .csv files (generated by RasterManipulation\_2 program).
* *save\_path\_storm*: folder directory where the .csv files with the per-storm R factor results will be saved (no need for input if *SaveCSV(df\_storm, name, save\_path\_storm* command is commented out, see Figure 4)
* *save\_path\_month*: folder directory where the .csv files with the monthly R-factor results will be stored. (no need for input if *SaveCSV(df\_storm, name, save\_path\_month* command is commented out, see Figure 4)
* *Save\_path\_raster*: file directory where the monthly and yearly R Factor result rasters for the entire watershed will be saved.

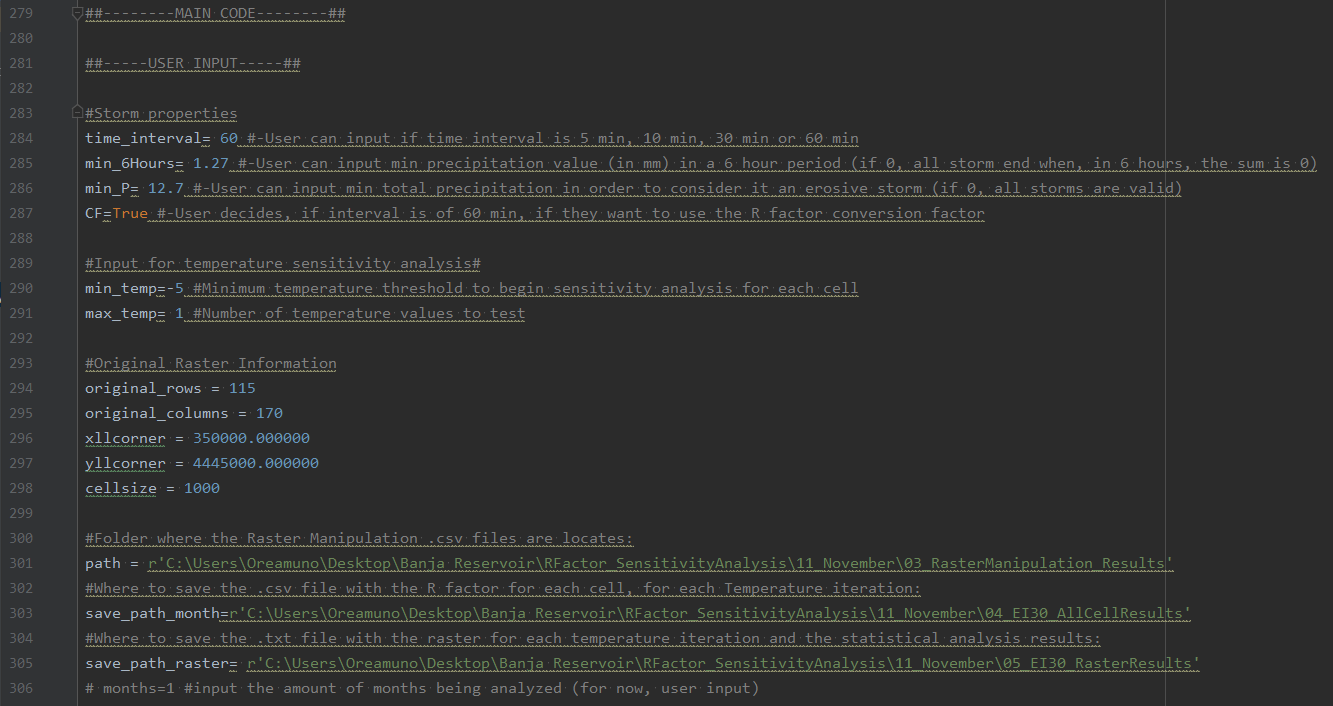


Figure 3. User input for program RFactor\_5

## Output data:

* **Per-Month and per Temperature R Factor results**: Comma-delimited .csv file with the R factor result for each month-year combination in the given time frame, for each cell and for each temperature within the given temperature range. Includes Month-Year and total R factor (seeFigure 4). The order of the cells is the same order in the folder where the input files are located.
  + Program generates one file that contains the results for all cells.
  + Each column represents the R factor for the given cell, the year-month combination and the temperature threshold. The last two colums correspond to the difference between the maximum and minimum R Factor for the row and the difference in percentage, respectively.
  + User can decide to not generate the files by commenting the code “*SaveCSV(df\_RasterR, “AllCells.csv, save\_path\_month*). The file can be used to manually check for other statistical properties.
* **Raster files**: tab delimited .txt files (ASCII Raster format), for each month-year-temperature combination, with R factor results. Each cell contains the corresponding R Factor value.
  + Generated by command “*SaveRaster(R\_3D\_months, R\_3D\_years, save\_path\_raster)*” (see Figure 4)
  + Program generates file for each month-year-temperature combination
  + File name corresponds to “year\_month\_temperature” combination
  + In order to open the Raster files in ArcGIS, the command “ASCII to Raster” tool must be used.

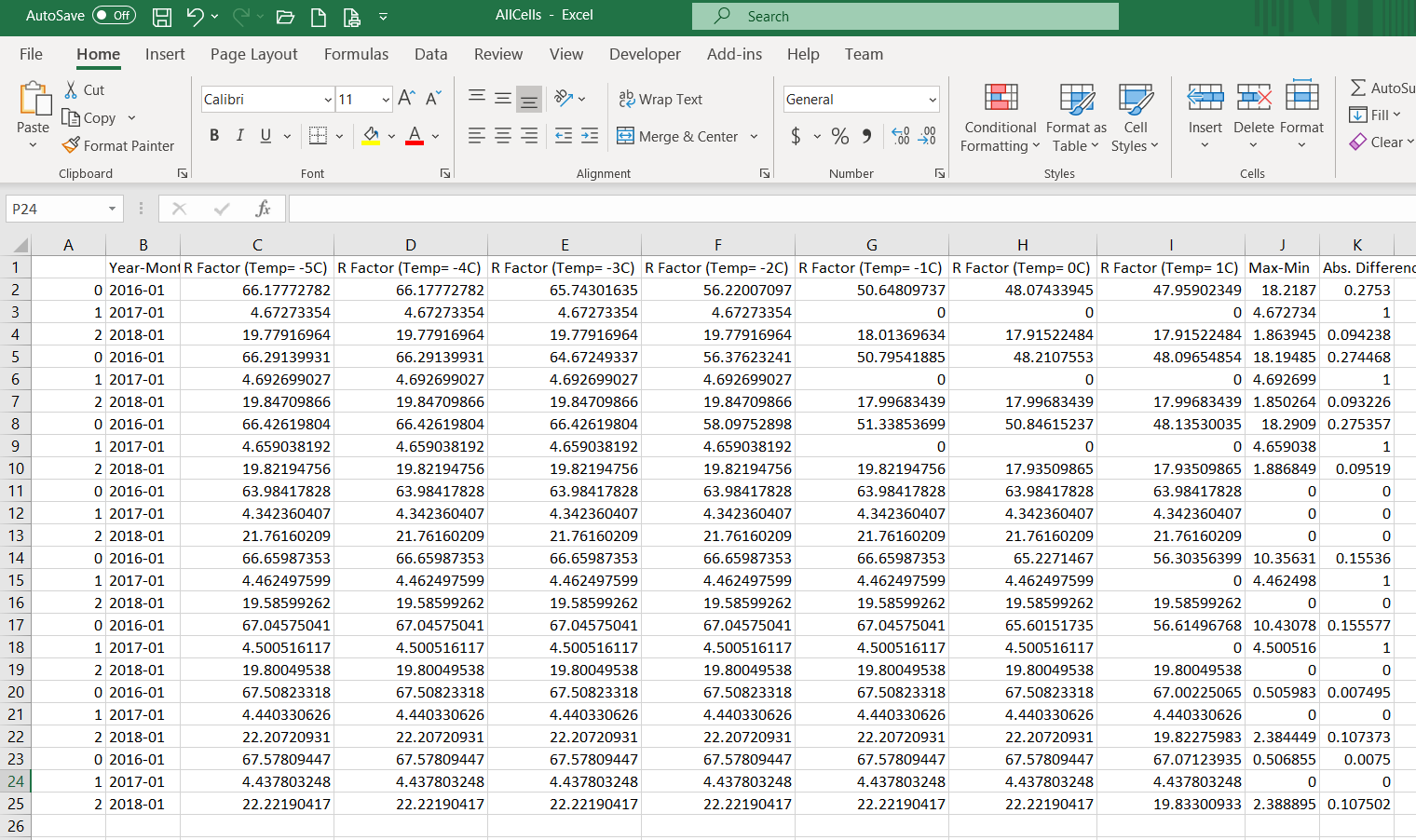


Figure 4. R Factor results for all cells

## Additional Information

* The program does the same calculations as the program “RFactor\_5”, with the difference that it runs a secondary loop within the main loop, and calculates the R factor for each cell for different temperature threshold values.
* The program calculates 2 statistical parameters for each cell in the raster, which are the difference between the minimum and maximum Rfactor (for each Month) and the percentage difference between these 2 values.
  + If more statistical values want to be added to the results, the following changes must be made:
    - The size of the 3D raster *R\_3D* must be changed to include the new values. The value multiplying the value *len(unique\_m\_y)* must be changed.
    - The new values must be calculated in the function RFactorStatistics (see 2.5.4)
    - The counters for the value “*k*”, which indicates the array in the 3D array to be filled must be changed to k+= n in the loops where the statistical values are being filled, where n is the number of statistical values being calculated (functions **ThreeDArray** and **SaveRaster**)

## Error Checks

### CheckInput:

* Checks if the user input *time\_interval* is equal to either 10 min 15 min, 30 min or 60 min.
* Checks if the *min\_6Hours* and *min\_P* user input correspond to a number
* Checks that *CF* is “True” only if the user input time\_interval is equal to 60 min.

### CheckTimeInterval:

* Checks, for the first file in the file directory, if the user input “*time\_interva*l” coincides with the time interval in the file. It assumes the same result for all files in the directory. If this were not the case, an error will come up during the running of the program.

## Main functions in the program

### **MonthYear:**

* Extracts the unique months and years in the input data files and saves them in different vectors (*unique\_months* and *unique\_years* respectively). Then, it creates a vector with all the possible month-year combinations called *unique\_m\_y*. This last one will later be used to generate the resulting raster files for each month-year combination.
* Receives a Data Frame with the data from the first data file in the file directory. The program assumes all input files have the same time frame.
  + If files don’t have the same time frame, and the other files include dates not contemplated in the first file, these dates won’t be considered in the final raster file generation. (Doesn’t affect the Per-Storm or Per Month calculations.

### **Main Loop:**

* Reads through the data files in the “path” folder
* Saves the name (which contains the row\_column combination for the given cell)
* Saves the data file in a Numpy Array
  + Program prints the name of each data file being read. This can be commented out.
* Creates a blank Data Frame, (*df\_CellR*) where the monthly results for each temperature within the temperature range are stored.
* Starts **Secondary Loop**, where the data frame *df\_CellR* is filled with the R Results for the given cell for each temperature threshold value.
* Calls the function **RFactorStatisctics** and calculates, for each row in *df\_CellR* the difference between the maximum and minimum R factor, and the percentage difference between these 2 values.
* Calls the function **RFactor\_Raster** fills the Data Frame *df\_RasterR* which compiles all the R factor results for each cell (*df\_CellR*). The current cell’s results are added below the previous cell’s data. Each cell will add n rows, equivalent to the amount of month-year combinations. This will be constant for all cells.

### **Secondary Loop**

* Loops through each temperature within the given temperature range, starting with the minimum temperature (*min\_temp*) and ending with the maximum temperature (*max\_temp*).
* Calls “**StormID**” function, which returns two data frames, one with the R Factor results for each calculated storm (*df\_storm*) and one with the R factors for each month-year combination (*df\_month*).
* Calls **RFactor\_Cell** which adds (concatenates) the R factor results for the given temperature to the Main Loop’s *df\_CellR* data frame. Each new R factor is added in a new column, after the previous loop’s result.
* Calls **ThreeDArray** to fill each 2D array within the 3D array, for each temperature value in the temperature range.

### **RFactorStatisctics**

* Receives the Data Frame *df\_Cell*, where each row corresponds to a different month-year combination for the given cell and each column to the R Factor calculated with different temperature thresholds.
* Called in the primary loop, when the *df\_Cell* data frame is full.
* Function adds 2 new columns to each row:
  + Max-Min: The difference between the maximum and minimum Rfactor for each cell
  + Abs. Difference: the percentage difference between the maximum and minimum Rfactor for each cell (calculated as follows:

### **SaveCSV**

* Receives the *df\_RasterR* data frame, after the Statistics have been added.
* Saves the Data Frame a comma-delimited .csv file.

### **ThreeDArray**

* Receives
  + *name*: file name, which contains the original row\_column of the cell corresponding to the file being looped through in the **main loop**.
  + *df\_Month*: Data frame with R Factor for given temperature threshold value (**Second loop** iteration) for each month-year-temperature combination
  + *R\_3D*: 3D array, originally created before the main loop and filled with the value -9999, Has one array for each unique month-year-temperature, and each 2D array has the same number of rows (*original\_rows*) and columns (*original\_coumns*) as the original precipitation raster.
  + *K*: indicates which array is being filled. Only n number of arrays, corresponding to the number of month-year combinations, is filled in each loop in the **secondary loop** (for each temperature value in the temperature range)
* Function extracts the row and column location of the cell from the file name and saves it in a vector. This is the cell updated in each loop in the **main loop.**
* Function loops through each value in *unique\_m\_y* vector and finds the month-year combination in the *df\_month* data frame, and assigns the value to the cell in the corresponding array “*k*” in the 3D array R\_3D\_months.
  + These loops determine the order in which the data is saved in the 3D array, and is taken into consideration when naming each output raster.

### **RFactor\_Cell**

* Receives the Data Frame *df\_month* where the R factor, per month, is stored for each cell and each temperature value.
* Function extracts the column with the R factor values for the given iteration from *df\_Month* and adds (concatenates) it to the Data Frame *df\_CellR*. There is the same amount of rows as the month-year combinations.

### **StormID**

* Receives:
  + *Station*: Numpy Array with data from input file
  + *time\_interval*: User input time\_interval
  + *min\_6Hours*: User input
  + *min\_P*: User input
  + *CF*: User Input
  + *Min\_temp*: User Input
* Loops through each row (each date) in *Station* in search for either the start or the end of a storm
  + A storm starts if, for a 6 hour time period, the total precipitation is equal or greater than *min\_6Hours*. If a storm starts, the variable “*hours”* becomes non zero, indicating that a storm has started.
    - A storm won’t start if, for a precipitation record, the temperature is below the temperature threshold (see explanation in section 2.5.4)
    - The row number where the storm starts is saved in variable *sd*.
  + A storm ends if, for a 6 hour time period, the total precipitation is less than *min\_6Hours*. Additionally, for a storm to end the recorded precipitation value must be equal to 0. (Panagos et.al) After a storm ends, the variable “*hours*” becomes 0, indicating that a new storm start must be determined.
    - The row number where the storm ends is saved in variable *ed*.
  + In order to determine if a storm starts or ends, the function calls the function **SumSixHours,** which returns the sum of the precipitation values for a 6 hour period, starting from the time being looped through.
  + When a storm ends, using the variables *sd* and *ed*, the total precipitation for the storm is calculated. If the total precipitation is greater than *min\_P* then the storm is considered as erosive. If the value is less than *min\_P*, then the storm data is not calculated.
    - Even though the temperature threshold is considered when determining if a storm starts or ends, it is not taken into consideration when adding the total precipitation for the storm event. This means that precipitation records with temperatures below the threshold are considered in the total precipitation value. (Taken according to C Code attached to Meusburger et. Al (2012) report).
  + For erosive storms, the maximum 30 minute intensity (I30), Kinetic Energy (KE) and R factor (EI30) are calculated for each storm, and stored in a data frame (*df\_storm*), calling the function **FillMatrix**. For a detailed explanation of the equations used see the section 2.6 in this document.
  + The function **RFacto**r is called, which sums the R factor for each month, using the *df\_storm* data frame, and saves it in a data frame (*df\_month).*

### **SumSixHours**

* Receives
  + *i*: row value being looped through
  + *Station*: Numpy array with data from file being looped through in **main loop**.
  + *min\_temp*: user input
  + *time\_interval*: user input
* Sums the precipitation values in a 6 hour period. Calls function **CalculateHour** to determine how many rows correspond to 6 hours, using user input *time\_interval*.
* If the temperature value of a given record is lower than *temp\_min*, the precipitation value is considered as 0, as below this threshold the precipitation is not considered to be erosive (snow).

### **SaveRaster**

* Receives:
  + *R\_3D*: 3D array, with one array for each month-year combination using *unique\_years* and *unique\_months* vectors, and filled in each loop in the **Secondaary Loop**, through the function **ThreeDArray**.
  + *Path*: *save\_path\_raster*, user input.
* Creates a data frame (*df\_head*) with information from the original raster, using the user input data for the original raster information. This is needed for the final raster to have an ASCII raster format and will be able to be read by a GIS program.
* Loops through each temperature value (in the same order as in the **Secondary Loop**), then loops through each year in *unique\_years* vector and each month in *unique\_months* vector, in the same order as in the **ThreeDArray** function, and:
  + Extracts each array “k” in the 3D array R\_3D\_month and converts it to a 2D array.
  + Concatenates df\_head with the 2D array into a single data frame and saves each data frame.
  + The name of the file corresponds to the combination of Year\_Month\_Temperature.

## Equations

### I30

* Corresponds to the maximum intensity in 30 minutes.
* Calculated in function **CalculateI30**, called from **StormID**.
* If *time\_interval* variable is less or equal to 30 min, calculates the rows corresponding to 30 minutes, sums the row precipitation values, and multiplies the sum by 2 to get intensity in mm/h.
* If *time\_interval* is 60 min, I30 is equal to the intensity in 60 min.

### Kinetic Energy (E)

* Corresponds to the total kinetic energy (in MJ/ha) for each storm.
* Calculatd in function **CalculateEnergy** called from **StormID**.
* First, it calculates the unit kinetic energy for each time step using the following equation (Panagos et.al, 2015)
* The total kinetic energy is equal to the sum of the individual unit kinetic energy value for each time step in the storm (Panagos et. Al, 2015):

### R Factor per storm

* Calculated in **CalculateEI** called from **StormID.**
* Calculates the R factor (in MJ\*mm/ha\*h) using the following equation (Panagos et.al, 2015)
* If *time\_interval* is equal to 60 min, and *CF* is True, the R factor is calculated using the following equation, applying a conversion factor to consider errors due to the larger time interval (Panagos et.al, 2015)

### R factor per month

* Calculated in function **RFactor**, called from **StormID**.
* Calculates the R Factor for each month

# References

Meusberger, K., Panagos, P., Montanarella, L., & Alewell, C. (2012). Spatial and temporal variability of rainfall erosivity factor for Switzerland. *Hydrology and Earth System Sciences*, 167-177.

Panagos, P., Ballabio, C., Borrelli, P., Meusburger, K., Klik, A., Rousseva, S., et al. (2015). Rainfall erosivity in Europe. *Science of the Total Environment*, 801-814.