Description of the scripts for looping over measures & neighbourhood types

First of all, I assume that you’ve installed the model using Python. If not, then this should be done first.

Tip: start with the run ‘Run model simple.py’, as it contains a simple run, showing the modelled variables and the basics on how the scripts work.

If you are using Spyder, make sure to set the working directory to the folder ‘Model run’.

**Run model.py**

This script uses a .csv as input for both the measures and the neighbourhoods. Within the ‘Parameters measures.csv’ are the parameters defined for each measure. Currently, these parameters are mainly based on Dutch conditions, obtained from the NKWK Excel file (I believe Laura Kleerekoper made it). The ‘Parameters

Input

*Parameters measures.csv* : Excel worksheet containing parameters defined for each measure. Currently, these parameters are mainly based on Dutch conditions, obtained from the NKWK Excel file (I believe Laura Kleerekoper made it).

*Parameters neighbourhoods.csv* : Excel worksheet containing parameters defined for each neighbourhood type. These parameters are set up for eight different types in the Netherlands. There are less parameters in the .csv, compared to the .ini file. I have only added the parameters that differ in between the neighbourhood types. If you feel like more should be added, then you can just create a new column with the parameter name and add the parameters. The model will read it and adjust the parameters accordingly.

*ep\_ts.csv* : Timeseries of the climate data (rainfall, evaporation and PET). The format in this file should always be used as input for the model.

*ep\_neighbourhood.ini & ep\_measure.ini* : These files are used for the list of parameters, which are then changed according to the measure and neighbourhood type in the model.

Output

An Excel workbook for each neighbourhood type containing the following measure efficiency values:

* Runoff reduction factor (Fmeas)
* Runoff reduction factor over the entire area (Ftot)
* Groundwater recharge, relative to the base recharge
* Evaporation, relative to the base evaporation

Exceptions

There are currently three measures in the script (id = 3, 25, 26), which are implemented in a different way. Whereas the normal measures are implemented as an extra modelling ‘part/area’, these measures are implemented by changing the catchment properties. E.g. implementing an urban forest is done by converting the open paved area into unpaved area. Therefore, these require a separate script and function.

**Run model simple.py**

This is a simple model run to see how the model is run. Basically, the climate input is read, followed by the input of the catchment and measure (which is currently set on not implemented: Apply\_measure = false). Next, the model runs using these inputs, with the ‘base\_run’ as output. This dataframe contains all modelled variables.

Input

*ep\_ts.csv* : Timeseries of the climate data (rainfall, evaporation and PET). The format in this file should always be used as input for the model.

*ep\_neighbourhood.ini & ep\_measure.ini* : These files are used for the list of parameters, which are then changed according to the measure and neighbourhood type in the model.

**SDF Curve.py**

This script creates a SDF (Storage-Discharge-Frequency) curve, based on a list of discharge capacities of the outflow of the model, given in ‘q\_list’. I tried to explain it on here: <https://publicwiki.deltares.nl/display/AST/Urban+Water+balance+model#UrbanWaterbalancemodel-5.3.2batch_run_sdf>. However, it may be a bit unclear as I find it hard to explain.

The SDF curve can be used to determine whether it is best to invest into storage capacity, or to increase the drainage capacity of the system.

The input is the same as for the simple model.