|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| tariffCalc | | Utils |
| It is used to give an annual file of hourly electricity price from the grid in €/kWh, in 3 possible ways: flat tariff, hourly tariff (2.0TD) based on user values or PVPC (based on historic values) + surplus electricity price in €/kWh. | | |
| **Class** | **Inputs** | **Outputs** |
| typeTariff | index\_y: day, month, year,hour time index | - |
| tariffSingle | index\_y: day, month, year,hour time index  tariff\_single: value in €/kWh for grid electricity tariff  tariff\_surplus: value in €/kWh for electricity surplus | df\_tariff : dataframe with index=index\_y, column “grid” with value for grid tariff and column “surplus” with value for surplus injected to the grid all in €/kWh |
| tariffHourly | index\_y: day, month, year,hour time index  tariff\_peak: value in €/kWh for grid electricity tariff for peak period  tariff\_flat: value in €/kWh for grid electricity tariff for flat period  tariff\_valley: value in €/kWh for grid electricity tariff for valley period  tariff\_surplus: value in €/kWh for electricity surplus |
| tariffPVPC | index\_y: day, month, year,hour time index |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| compSimplificada | | Utils |
| It is used to do several operations of the economic calculation of the energy community. | | |
| **Class** | **Inputs** | **Outputs** |
| balEcoYear | df\_balance:results from energy balance  area\_cub: total installed PV area  parameters: json with economic parameters (resources\data\dataeconomic.json) | - |
| **methodes** | **Inputs** | **Outputs** |
| monthlyBalance | tariff: hourly tariff values for 1 year (output of tariffCalc) | balance\_mensual: series of 12 values with the total energy term to pay each month of the year with the simplified compensation of surpluses (taxes not included) |
| monthlyBalanceBS | tariff: hourly tariff values for 1 year (output of tariffCalc) | balance\_mensual: series of 12 values with the total energy term to pay each month of the year with the simplified compensation of surpluses (taxes not included) for a user with bono social |
| annualSavings | tariff: hourly tariff values for 1 year (output of tariffCalc) | annual\_savings: sum of the total savings during one year in € (considering also the taxes saved on self-consumed energy) |
| annualSavingsBS | tariff: hourly tariff values for 1 year (output of tariffCalc) | p\_cost: yearly electricity cost without self-consumption in € for a user with bono social (considering taxes)  annual\_savings: sum of the total savings during one year in € for a user with bono social(considering also the taxes saved on self-consumed energy) |
| savingsLife | tariff: hourly tariff values for 1 year (output of tariffCalc) | savings\_y\_life: savings in € during the whole lifetime of the system, considering variation of electricity cost |
| savingsLifeBS | tariff: hourly tariff values for 1 year (output of tariffCalc) | savings\_y\_life: savings in € during the whole lifetime of the system, for a user with bono social, considering variation of electricity cost |
| simplePayback | annual\_savings: yearly savings (output of savingsLife or savingsLifeBS)  cost: cost per unit of area | payback\_s\_round: simple payback rounded  total\_cost: total investment cost |
| dataCost | area\_cubiertas: list with PV area of each installation | cost\_ref\_inv: total investment cost according to the reference costs  cost\_mant\_y: maintainance yearly cost according to the reference costs (including inverter replacement once during the system’s lifetime) |
| calcNPV | total\_cost: total investment cost  tariff: hourly tariff values for 1 year (output of tariffCalc)  cost\_mant: yearly maintainance cost | NPV: list of NPV values for each year during the system’s lifetime  payback\_c: payback time of the investment considering the discount rate |
| calcNPVBS | total\_cost: total investment cost  tariff: hourly tariff values for 1 year (output of tariffCalc)  cost\_mant: yearly maintainance cost | NPV: list of NPV values for each year during the system’s lifetime for a user with bono social  payback\_c: payback time of the investment considering the discount rate for a user with bono social |
| graphNPV | NPV  payback\_c | Draws a graph for NPV across the years with the payback printed on it |
| calcLCOE | total\_cost | LCOE: value of levelized cost of energy for the whole system’s lifetime. |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| co2Balance | | Utils |
| It is used to calculate yearly CO2 savings and primary energy savings according to factors of emissions and coefficients from RITE (2016). | | |
| **Class** | **Inputs** | **Outputs** |
| balCO2Year | df\_balance\_y:results from energy balance | - |
| **methodes** | **Inputs** | **Outputs** |
| prim\_before |  |  |
| CO2\_before |  |  |
| prim\_after |  |  |
| CO2\_after |  |  |
| prim\_savings |  | Prim\_estalvi: yearly primary energy savings |
| CO2\_savings |  | CO2\_estalvi : yearly CO2 savings |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| repartoSomCom | | Utils |
| It is used to give calculate coefficients for each user according to the distribution modality chosen. | | |
| **Class** | **Inputs** | **Outputs** |
| typeCoef | columns\_id: column name with the identification of each user | - |
| CoefUnicoViv | columns\_id: column name with the identification of each user  coef\_cp: coefficient of the “main” user | coef: Series with name column of each user and its corresponding coefficient (one coefficient per user) |
| CoefDifViv | columns\_id: column name with the identification of each user  coef\_cp: coefficient of the “main” user |
| CoefDifEdif | columns\_id: column name with the identification of each user  coef\_cp: coefficient of the “main” building  n\_viv: number of users in the main building |
| CoefVar | consumption\_df: series with column name of each user and total yearly consumption + column ‘Total’ with the total sum |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| dataEcon\_repository | | Repository |
| It is used to read economic data. | | |
| **Class** | **Inputs** | **Outputs** |
| ciclicaSimple | -  (there is a filepath for the json inside the class – dataeconomic.json) | get\_data: economic parameters for the basic profile of SCE calculation. |
| ciclicaAdv | -  (there is a filepath for the json inside the class – dataeconomic2.json) | get\_data: economic parameters for the advanced profiles of SCE calculation. |
| PVPCActive | -  (there is a filepath for the csv file inside the class – PVPCActiva2023.csv) | df\_out: dataframe with hourly values in €/kWh for the active energy of PVPC based on 2023 file |
| PVPCSurplus | -  (there is a filepath for the csv file inside the class – PVPCExcedente2023.csv) | df\_out: dataframe with hourly values in €/kWh for the surplus energy of PVPC based on 2023 file |
| defaultCosts | -  (there is a filepath for the csv file inside the class – precios\_ref.csv) | get\_data: reference prices of SCE and area superior limit of the intervals (investment cost and maintenance cost) |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| dataEconomic | | Interactors |
| It is used to call the repository and return parameters | | |
| **Class** | **Inputs** | **Outputs** |
| somCInputs | - | data\_params: economic parameters for the basic profile of SCE calculation. |
| somCInputsAdv | - | data\_params: economic parameters for the advanced profiles of SCE calculation. |
| PVPCData | - | active\_PVPC: dataframe with hourly values in €/kWh for the active energy of PVPC.  surplus\_PVPC: dataframe with hourly values in €/kWh for the surplus energy of PVPC. |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| dataProfiles\_repository | | Repository |
| It is used to read and treat data of stochastic profiles .out file, transforming it in an hourly consumption files (average of each 20 entries) | | |
| **Class** | **Inputs** | **Outputs** |
| fileProfile | filePath: name of the file to be analysed | cons\_df\_av : dataframe with hourly consumption values |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| dataProfiles | | Interactor |
| It is used to get hourly consumption data of several stochastic profiles inside a folder, and separate it into ConStove + ConOven + ConDHW + Sum (other equipment consumption) | | |
| **Class** | **Inputs** | **Outputs** |
| outputProfiles | -  (\resources\data\profiles - folder where all the files are) | folders: dictionary of dataframes for each file |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| repositoryConsumption | | Repository |
| It is used to read files of demand profiles | | |
| **Class** | **Inputs** | **Outputs** |
| SomProfilesSimple | -  (csv file with the hourly consumption for the average number of occupants) | data\_out: dataframe with hourly consumption references, columns: ConBase (base equipments consumption), ConStove (electric stove consumption), ConOven (electric oven consumption), ConDHW (electric water heater consumption), Qheat (heating demand), Qcool (cooling demand) |
| SomProfiles | n\_occ: number of occupants of the household |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| baseConsumption | | Interactors |
| It is used to create the consumption dataframe for a user according to the inputs of the electric loads. | | |
| **Class** | **Inputs** | **Outputs** |
| ConsSimple | numV: number of households in the building | profile\_total: hourly consumption profile for users in single-family or multi-family buildings (average values from SCE) |
| SomProfiles | oc\_viv: number of occupants  dhw\_viv: electric dhw (0/1)  coc\_viv: electric stove (0/1)  hor\_viv: electric oven (0/1)  heat\_viv: type of heating ("heat\_none","heat\_rad","heat\_hp","heat\_other")  acc\_viv: type of air conditioner( "acc\_none","acc\_split","acc\_hp")  area\_viv: surface area of the household  numV: number of households in the building  consum\_viv: yearly electric consumption of the household |  |
| **Method** | **Inputs** | **Outputs** |
| cons\_total |  | profile\_total: dataframe with hourly total consumption of the household, adjusted according to the yearly consumption informed |
| ConsBuilding | n\_part: number of households in the building participating in the self-consumption  oc\_viv: average number of occupants  dhw\_num: number of households with electric dhw  coc\_num: number of households with electric dhw hor\_num: number of households with electric stove heat\_none\_num: number of households with no heating  heat\_rad\_num: number of households with radiators for heating  heat\_hp\_num: number of households with HP for heating  heat\_gas\_num: number of households with gas heating  acc\_none\_num: number of households with no air conditioner  acc\_split\_num: number of households with split acc  acc\_hp\_num number of households with HP acc area\_viv: average area of household  consum\_viv: average yearly electric consumption of the household |  |
| **Method** | **Inputs** | **Outputs** |
| cons\_total |  | profile\_sum: dataframe with hourly total consumption of all households, adjusted according to the average yearly consumption informed  profile\_av: dataframe with hourly total average consumption of one household, adjusted according to the yearly consumption informed  f\_corr: correction factor used to adjust the consumption according to the yearly consumption informed |
| cons\_min | corr\_av: correction factor of total consumption | profile\_min: dataframe with hourly total consumption of the least consuming household, adjusted with the same correction factor as the total consumption |
| cons\_max | corr\_av: correction factor of total consumption | profile\_max: dataframe with hourly total consumption of the most consuming household, adjusted with the same correction factor as the total consumption |

New classes added to existing files:

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| energyBalance\_FV | | Utils |
| The new classes are adapted to SCE inputs and outputs. | | |
| **Class** | **Inputs** | **Outputs** |
| balancePropioSomCom | dataframe\_consumption: dataframe with user hourly consumption  dataframe\_PV: dataframe with PV available to the user | df\_balance: balance Dataframe (PV\_base,Med,Sc,Dt,Et,Net)  LoCov: % self consumption |
| balanceCombinadoCoefSomCom | dataframe\_consumption: dataframe with hourly consumption for each user (each column is a user)  dataframe\_PV\_total: dataframe with total PV available for the community  coef\_reparto: series with one coefficient for each user (each column is a user, with the same column name as the dataframe\_consumption) | df\_balance: dictionary with balance Dataframe for each user and Total  df\_LoCov\_all: % self consumption for each user  LoCov\_ag: % self consumption for the community |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| energyBalance\_FV | | Utils |
| The new classes are adapted to SCE inputs and outputs. | | |
| **Class** | **Inputs** | **Outputs** |
| balancePropioSomCom | dataframe\_consumption: dataframe with user hourly consumption  dataframe\_PV: dataframe with PV available to the user | df\_balance: balance Dataframe (PV\_base,Med,Sc,Dt,Et,Net)  LoCov: % self consumption |
| balanceCombinadoCoefSomCom | dataframe\_consumption: dataframe with hourly consumption for each user (each column is a user)  dataframe\_PV\_total: dataframe with total PV available for the community  coef\_reparto: series with one coefficient for each user (each column is a user, with the same column name as the dataframe\_consumption) | df\_balance: dictionary with balance Dataframe for each user and Total  df\_LoCov\_all: % self consumption for each user  LoCov\_ag: % self consumption for the community |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| DataFV\_repository | | Repository |
| The new classes are used for reading json files for each case. | | |
| **Class** | **Inputs** | **Outputs** |
| ciclicaSimple | -  (there is a filepath for the json inside the class – data3.json) | get\_data: parameters read from the files |
| ciclicaAdv | -  (there is a filepath for the json inside the class – datacol.json) |
| dataCIES | -  (there is a filepath for the json inside the class – dataCIES.json) |

|  |  |  |
| --- | --- | --- |
| **Name** | | **Type** |
| DataFV | | Interactors |
| The new classes are used to calculate with | | |
| **Class** | **Inputs** | **Outputs** |
| somCInputs | -  (there is a filepath for the json inside the class – data3.json) | data\_params: parameters of the building  df: dataframe with PV data |
| somCInputsAdv | -  (there is a filepath for the json inside the class – datacol.json) | data\_params: dictionary of parameters of each building  df: dataframe with PV data (based on the data of the first building of the set of data, supposedly there will be only one longitude, latitude, timezone for the whole EC, even though they are repeated in all sets) |
| CIESInputs | -  (there is a filepath for the json inside the class – dataCIES.json) | data\_params: parameters of the building  df: dataframe with PV data |