sozioe2s-test Documentation

Release 0.0.1

test user

Aug 06, 2020

Contents:

CHAPTER 1

Overview

CHAPTER 2

The Inve2st Model

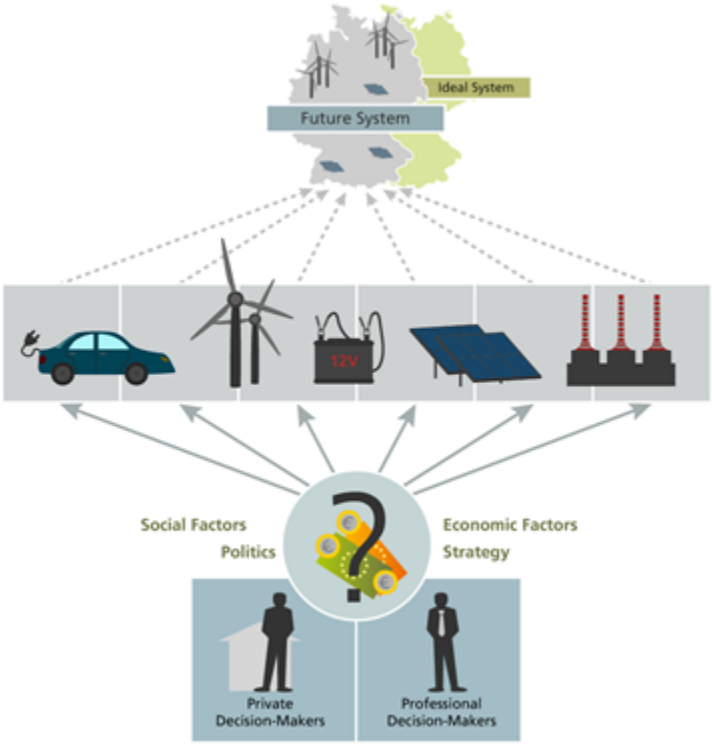
2.1 1. Explanation of the model

The framework is designed to calculate the market diffusion of energy technologies until 2050 in dependence on investment-decisions made by respective actors. The target of the model is to evaluate if future target systems can be achieved considering the investment behaviour and its influencing factors. The understanding of a target system is hereby a quantified energy scenario, which is based on the CO2-emission targets of the German government. All sectors and components within the system (generation, storage, grid) are integrated within a mathematical model, which minimizes the total system costs. Examples for those models are, among others, REMod or TIMES. A meta-study of target systems can be found in (Jülch, Senkpiel, Kost, Hartmann, & Schlegl, 2018). However, there is a gap between the cost-optimal solutions and the real technology diffusion as the latter on depends on actual adoption or investment decisions. Those in turn are affected by the given framework conditions. The Inve2st framework aims to evaluate the impact of changing framework conditions on technology adoption and therewith the technology diffusion. Further it prodvides insights into factors which are important to achieve certain technology shares. The investment decisions differ between private and corporate decisions and also between the technologies.

2.2 2. The exemplary cases

Within this framework, three different technologies are modelled.

- alternative drive systems for passenger cars
- PV-battery systems of private house owners
- Power-to-Gas systems



The PV-battery systems and passenger cars focus on private investment decisions, whereas the diffusion of Power-to-Gas represents corporate investment decisions.

The three technologies serve as exemplary cases on the following question:

1. Which methodology is applicable to model the diffusion of the specific technology?
2. What kind of data (qualitative and/or quantitative) is necessary for modelling and how can this data be empirically collected?
3. How can the diffusion be modelled?
4. How strongly do changing framework conditions influence the diffusion process?

CHAPTER 3

Use Cases In Detail

This chapter describes the modelling of the adoption of

- alternative drive systems of passenger cars
- PV-battery systems of private house owners
- Power-to-Gas systems

3.1 1. Passenger cars

Introduction

The aim of the model is to simulate the annual adoption of alternative drive systems of passenger cars by private persons. With the model results it is possible to compare these results with the targets that originate from energy system scenarios to fulfill the necessary CO2-emission reduction targets.

Empirical data

The modelling approach follows the method of a discrete choice model using data from a representative discrete choice experiment incorporated in an online questionnaire study. The respondents had to choose among three alternative vehicle types (battery-electric (BEV), fuel cell (FCEV) and conventional (diesel/gasoline CV)), which were characterized by following attributes:

- CAPEX
- CO2-tax
- fuel costs
- infrastructure
- range
- well2well emissions

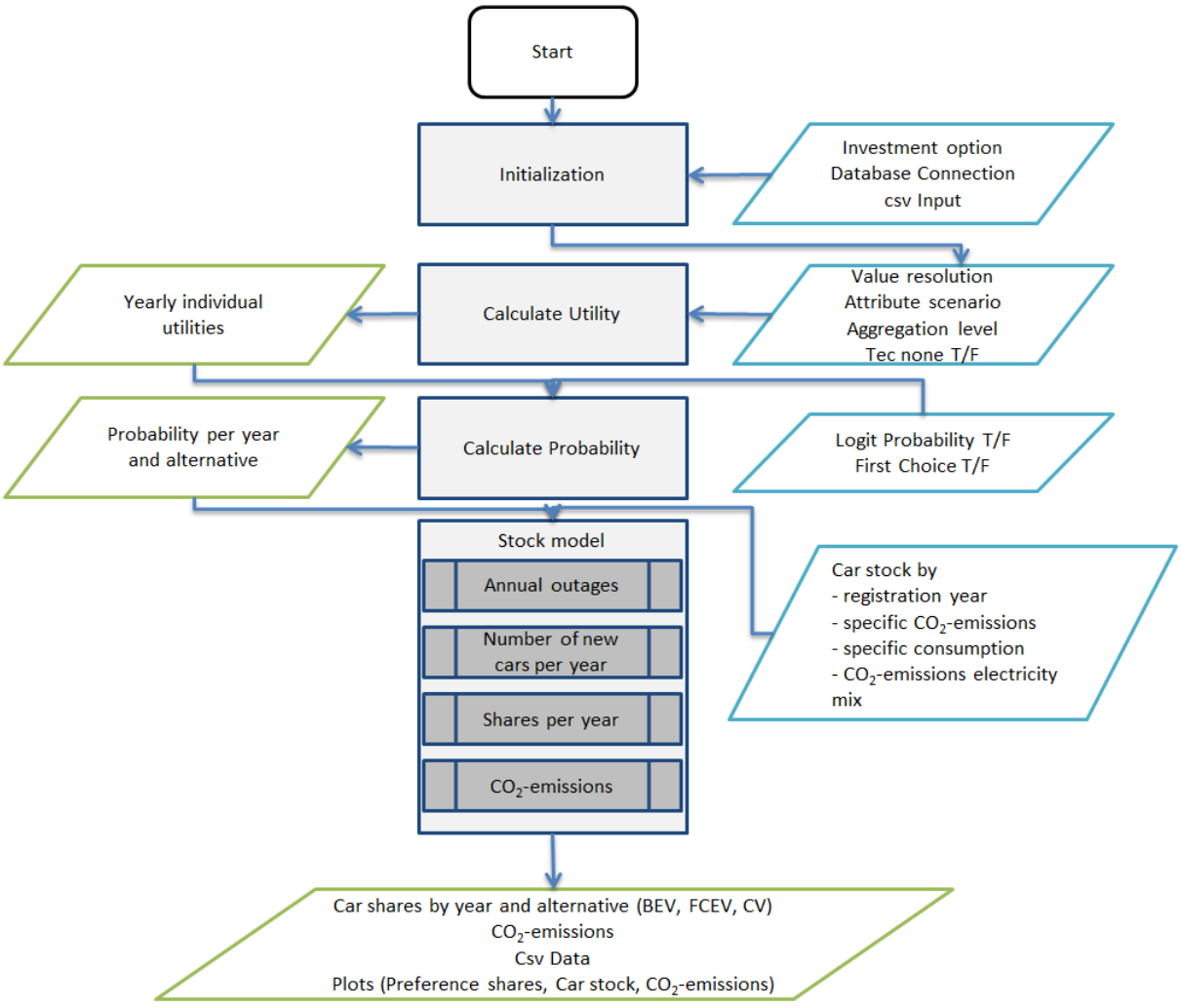
Each attribute had two to four attribute levels, which were chosen to represent the bandwidth of the development from today until 2050. The attribute levels rotated within the experiment, so that every participant responded to 10 choice tasks. As the attribute levels differ between different car classes a distinction between three classes was undertaken.

- Class1_small (minis, small cars and compact class)
- Class2_medium (mid-sized cars)
- Class3_upper (upper-sized cars and luxury cars)

Model process

For simulating the adoption

1. The model is initialized
2. The utilities are calculated
3. The probabilities are calculated
4. The change in the car stock is calculated



1. **Initialization def __init__() (*API*)**
    - the user chooses between different car classes (Class1_small, Class2_medium, Class3_upper)

- Data base is initialized (this is not necessary if csv data is read in (self.read_data = True))
- The starting year is defined according to the car stock data 2018

2. **Calculate Utility** calculate_utility() (*API*)

The target of this function is to calculate the total utility of possible alternatives for each respondent. All possible alternatives were created from a set of attribute levels. Via the discrete choice experiment, partial utilities of the respective attribute levels were calculated. The partial utilities were calculated with SAWTOOTH software on the basis of Hierarchical Bayes Estimation (https://www.sawtoothsoftware.com/support/technical-papers/hierarchical-bayes-estimation). The total utilties were calculated as the sum of partial utilities of each attribute level. Partial utilities were provided within the original dataset *Data and Database* of the discrete choice experiment.

For vehicles, the attributes are specified as a (future attribute development) scenario, in which the car attributes (capex, co2-tax, fuel costs, infrastructure, range and well2well emissions) are specified by means of the predicted? attribute levels for each year from the start_year (2018) until 2050 *Class1_small_average__False_S1_moderate_afv_logit_* . By adapting the attribute levels the user can create a new scenario for simulating different technology deployments. As a result, the influence of changing parameters (for example good infrastructure for battery electric vehicles) on the investment decision of alternative vehicle types can be modelled. For a proper functioning of the model ?? it needs to be ensured, that the attribute levels are within the minimum and maximum value of the DCE values. Interpolations of continuous values are possible, extrapolations are not valid. For the discrete attributes (CO2-tax, infrastructure and wel2whell emissions) the partial utilities are directly used from the data set whereas for the continuous attributes (CAPEX, fuel cost, range) the values of the partial utilities are interpolated linearly between two data points.

One option that can be chosen in the car_simulation.py is the inclusion of the NONE option. This operation is only recommended for the calculation of the preference shares but not for the stock model, in which an assumption of the development of the cumulated car stock (as a percentage) can be set as an input parameter. The input data needed as csv or data base query are:

1. alternatives *Class1_small_average__False_S1_moderate_afv_logit_*
2. query_attribute_level_putility (partial utilities from original dataset)
3. query_utility_none_option (partial utility of none option if enabled)
4. query_attribute_level_per_year (scenario definition from today until 2050 (e.g.attribute_scenanrio ='S1_moderate_afv'))

As a result of the function a pandas.dataframe (utilities_alternatives) is generated. The probability is not calculated in this step (-1 serves as a placeholder). The usage of the average utilities is not recommended, as the results differ distinctly from the usage of the individual utilities. The value resolution and aggregation *average* is recommended to use for the understanding and further development of the model, as the simulation is much faster than for individual values.

Index	year	alternative	respondend	utility	probability
0	2018	tec_bev	resp_average	0.179992	-1
1	2018	tec_cv	resp_average	1.3132	-1
2	2018	tec_fcev	resp_average	-2.49958	-1
3	2019	tec_bev	resp_average	0.299005	-1
4	2019	tec_cv	resp_average	1.33014	-1
5	2019	tec_fcev	resp_average	-2.40204	-1

3. **Calculate Choice Probability** On the basis of the total utilities per alternative per respondent, the preference share for one alternative compared to the other alternatives is calculated. For this purpose, different logics can be applied. One of them needs to be chosen in the car_simulation.py *(probability_calc_type = 'logit').*

First choice calculate_first_choice() (*API*)

The assumption of this rule is that the respondent chooses the alternative with the highest utility.

Logit choice probability calculate_logit_probabilities() (*API*)

Within this rule a share of preference towards each alternative is calculated per respondent. Following the equation:

$$P_j = e^{U_{ij}} / \sum_j e^{U_{ij}}$$

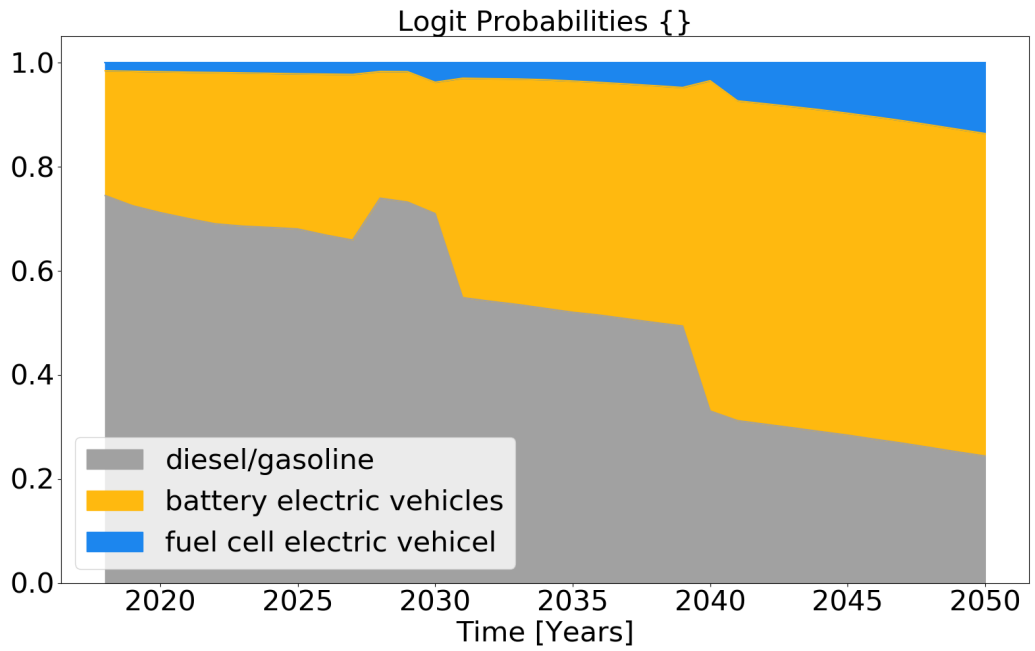
Where i is individual, j is alternative, U is utility and P is the probability

The input data needed as csv or data base query are for both rules:

1. utility_alternatives (result from calculate_utility())
2. main_sub (main_sub = { } (no building of subgroups))

Explanation main_sub: represents a subgroup of the respondents; for example only selecting the respondents that stated to be female for analyzing the influence of person-related factors. Note: The specification only works with database connection. If no connection to the database exists a subgroup of respondents can be manually built in the csv file df_sub.

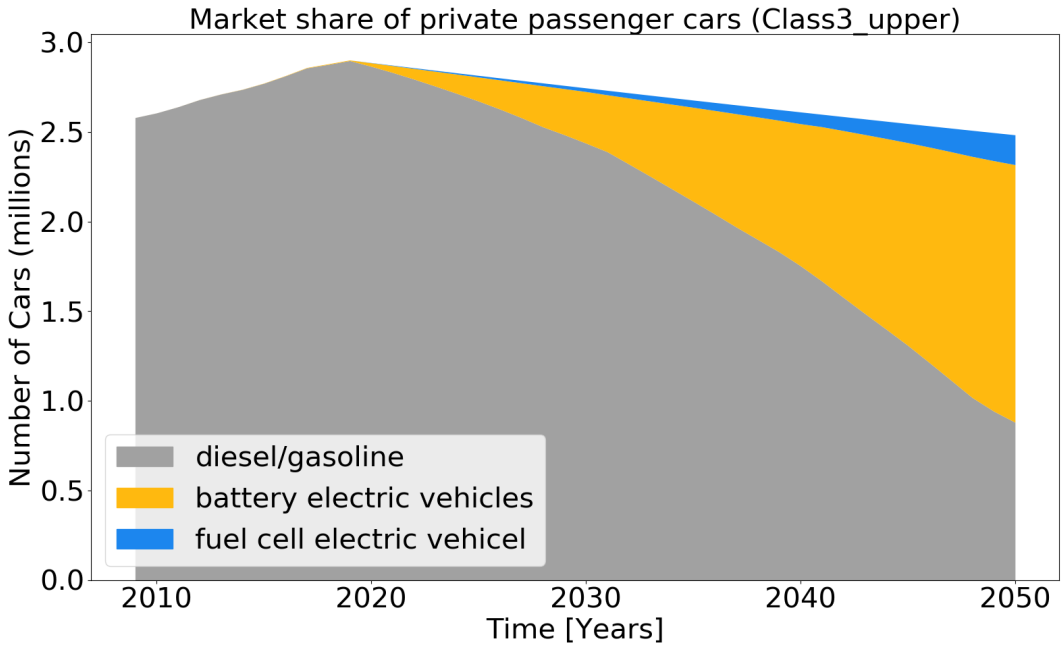
As a result of one of the decision rules a pandas data frame *tb_prob_alternatives* is passed and saved (results/tb_prob_alternatives.csv). In addition, a graph with the preference shares is saved (/results/preference_share.png).



4. **Stock model** stock_model()(*API*)

Aim of the stock model is the calculation of the **total car stock by vehicle type** (BEV, FCEV, CV) from 2018 until 2050, on the basis of the preference shares of the individuals. For this purpose, an assumption on the development of the total car stock as a yearly percentage (e.g. growth_scenario = 'S_constant') is made to calculate the total number of cars in the next year (stock_sum table), on an annual basis. Additionally, the number of cars that will be deregistered in the actual year is calculated dependent on the age of a car (car_stock table) by vehicle type. To calculate the outage

probabilities a Weibull Fit is used on the basis of the historic car stock development (tb_stock) from 2001 to 2018. Having the number of new total car stock for the next year and the outages in the current year, the total number of new cars is calculated. The distribution of the new cars per vehicle type is calculated using the preference shares, that are calculated in either calculate_first_choice() or calculate_logit_probabilities(). The process is repeated sequentially until 2050 on an annual basis. As a result, the csv File stock_sum is saved in the results folder. In addition, the plot stock.png is created and saved.

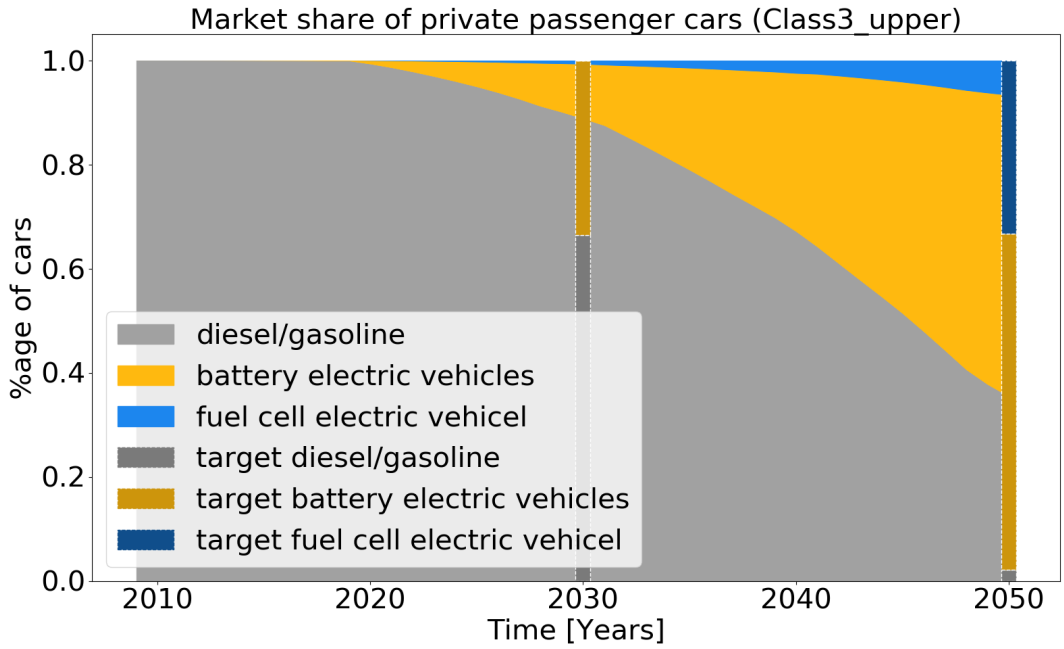


A comparison of the calculated diffusion of alternative driving concepts with shares from cost-minimizing, model-based quantified sector-coupled energy scenarios (e.g. REMod), which include a CO2-emission reduction target is realized on the basis of technology shares. It is plotted and saved in plot share.png and a statement is put in the command prompt :

"In 2030, the market share of battery electric vehicles (BV) is 11.46 %. The target of 33.51 % is not achieved. In 2030, the market share of fuel cell electric vehicles (FCEV) is 0.81 %. The target of 0.00 % is achieved. In 2050, the market share of battery electric vehicles (BV) is 57.78 %. The target of 64.52 % is not achieved. In 2050, the market share of fuel cell electric vehicles (FCEV) is 6.76 %. The target of 33.29 % is not achieved."

In addition, the **CO2-emissions** based on assummed passenger kilometers (which is specified in car_simulation.py - e.g average_passenger_kilometers = 20900) are estimated per passenger car and specific emission values. For the calculation of emissions of conventional vehicles, emissions (according to Agora and own assumptions) are calibrated based on the total Pkm in 2018 (source:" Destatis Verkehr in Zahlen") and 70% (according to Renewbility III) of the total emissions from road transport (UBA) for passenger cars, compared to freight transport. Historic and future specific emissions per construction year and vehicle class are taken as data basis. For calculating emissions of BEV and FCEV assumptions on the specific consumption (kWh/100km) as well as CO2-emissions of the electricity mix [gCO2/kWh]are used to calculate the CO2-emissions.

It has to be mentioned that the specific emissions from literature are much higher than the calibrated values, which shows that uncertainties arise from 1) specific emission values and 2) average driving performance. To adequately calculate the emissions a more detailed model (like TREMOD), which addresses relations between car classes, and driving performance, in terms of road usage, shares of innercity drives, highway drives, overland drives, and further factors would be needed. A plot of theCO2-emissions (CO2_emissions.png) which shows the total estimated CO2-emissions until 2050 is saved. A prompt *"The proportional CO2-emission reduction target of 40-42% in 2030*



compared to 1990 in the transport sector is not achieved, as a remaining share of 70% is estimated for 2030 and 40% for 2050" is printed in the console."

3.2 2. PV-homestorage systems

Introduction

The aim of the model is to simulate the purchasing preferences of a PV home storage system (HSS). The alternatives are the purchase of the system or no purchase. The following cases are subdivided, for each of which different attribute levels were determined in the Disctrete Choice Experiment.

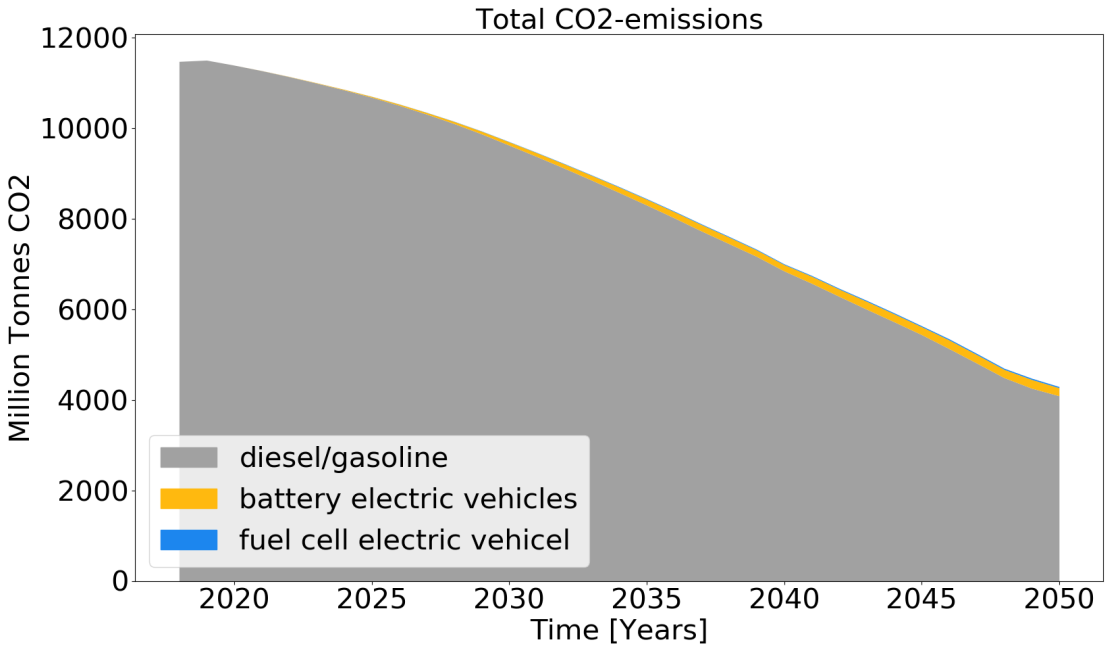
Empirical data

- House owner without PV or PV battery system
- House owner with PV system

The modelling approach follows the method of a discrete choice model using data from a representative discrete choice experiment, in which the respondents (house-owners) had to choose among three different types of HSS, which are characterized by following attributes:

- time of realization
- CAPEX
- IRR/Paybacktime
- Degree of autarky
- Environmental impact

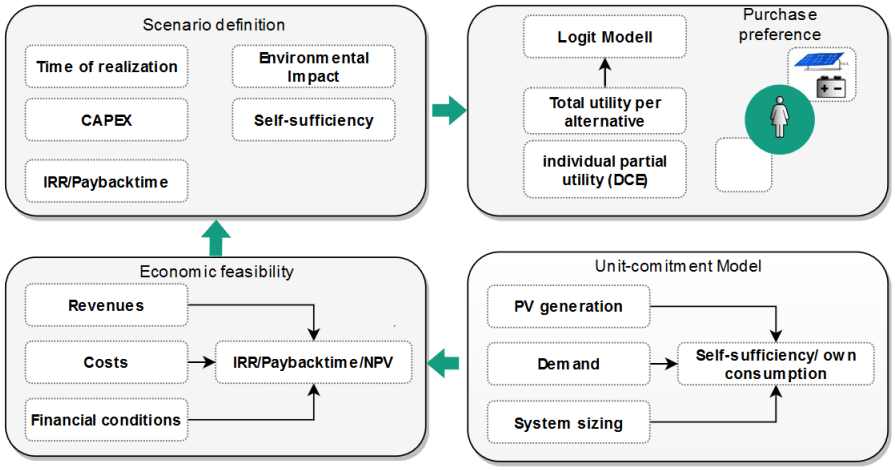
Each attribute has three to four attribute levels, which are chosen to represent the bandwidth of the development from today until 2050. The attribute levels are rotating within the experiment, meaning that the experiment was repeated multiple times. The empirical data, containing the individual partial utilities, can be found under the following link:



https://fordatis.fraunhofer.de/handle/fordatis/153

Calculation Steps

To be able to calculate the preferences to buy a HSS system the following calculation steps are undertaken:



The first two steps are performed in the script *calc_UCM_economics*, steps 3 and 4 in *Preferences_HSS*.

1. Degree of autarky

For the calculation without database connection, databaseconnection=False must be set. In this case, Scenario_name = 'Default_Data' should be set. The degree of self-sufficiency is one variable for determining the utility of the HSS. Therefore, in a first step, it is determined as a function of various input parameters. The degree of autarky is defined as:

The share of own consumption is defined as:

$$Degree\ of\ autarky = {(PV generation - Feed\ in)}/{electricity\ demand}$$

$$Self\ consumption = {(PV generation - Feed\ in)}/{PV\ generation}$$

The user input for determining the degree of autarky and own consumption is the ratio of PV system size to demand, the ratio of PV system to battery system (both possible to list for iterative simulation) as well as the ratio of battery capacity to power and the roundtrip efficiency of the battery. The values can be specified as user input in the script.

```
iterables_on = True
#iterables_on = False

#demand_scenarios = ['Row_house_Family_4P','single_family_house_2_persons']    # available demand scenarios
demand_scenarios = ['Row_house_Family_4P','single_family_house_2_persons']

#regions =['DE731','DEF01','DE131']                                            # available regions for PV generation #DE731
regions =['DE131']

#PV_orientations= ['S1_pv_eastwest','S2_pv_renewable_ninja_south']             # available PV orientations
PV_orientations= ['S2_pv_renewable_ninja_south']

# Dimensioning of PV-battery system
#ratio PV-size to electricity demand
#ratio_PV_demands = [0.8, 1, 1.2, 1.4]                                         # suggestions of ratios Figgener.2018
ratio_PV_demands = [1.2]

# PV size to Battery capacity
#ratio_PV_Batterys = [0.8,1,1.5,2]                                             # suggestions of ratios (between 0.8 and 5 /
ratio_PV_Batterys = [0.8]

ratio_PV_energy_power = 4

roundtrip_efficiency_bat = 0.85                                               # Figgener.2018 (0.75 - 0.95)

#    GENERAL
```

Within the function calculate_PV-battery_use() (*API*), the use of the battery is calculated in order to calculate the degree of autarky as well as the own-consumption share for a specific application. The hourly load and the PV generation curve are used as input. The load can be specified for different applications and was determined with the load generator SynPro (https://www.elink.tools/elink-tools/synpro/). The generation time series were determined with renewables.ninja (https://www.renewables.ninja/) and scaled according to the assumed ratio of demand to load. The values are written to the data frame (df_PV_use).

First the operation without storage is calculated. The residual load (residual_load_wo_storage) results from the difference between load and generation (load_kW) - (gen_scaled). From this the grid feed can be calculated without storage (Feed_in_wo_stor). The maximum grid feed-in is limited to 70% of the nominal power according to EEG2014, §9. The curtailment (curtailment_wo_stor) is calculated as the amount of energy that is greater than 70% of the PV output.

The second step is to determine the battery usage. At the first hour the battery is assumed to be empty. The battery is charged if the grid feed is positive (PV surplus) and the state of charge of the previous hour is less than the battery capacity. Limits are the remaining storage level and the charging capacity.

The discharge of the battery always occurs when the residual load is positive (power shortage) and the battery state of charge (SOC) is greater than zero. Thus, the grid feed-in with storage as well as the curtailment can be calculated according to the equations for determining the degree of own consumption and self-sufficiency.

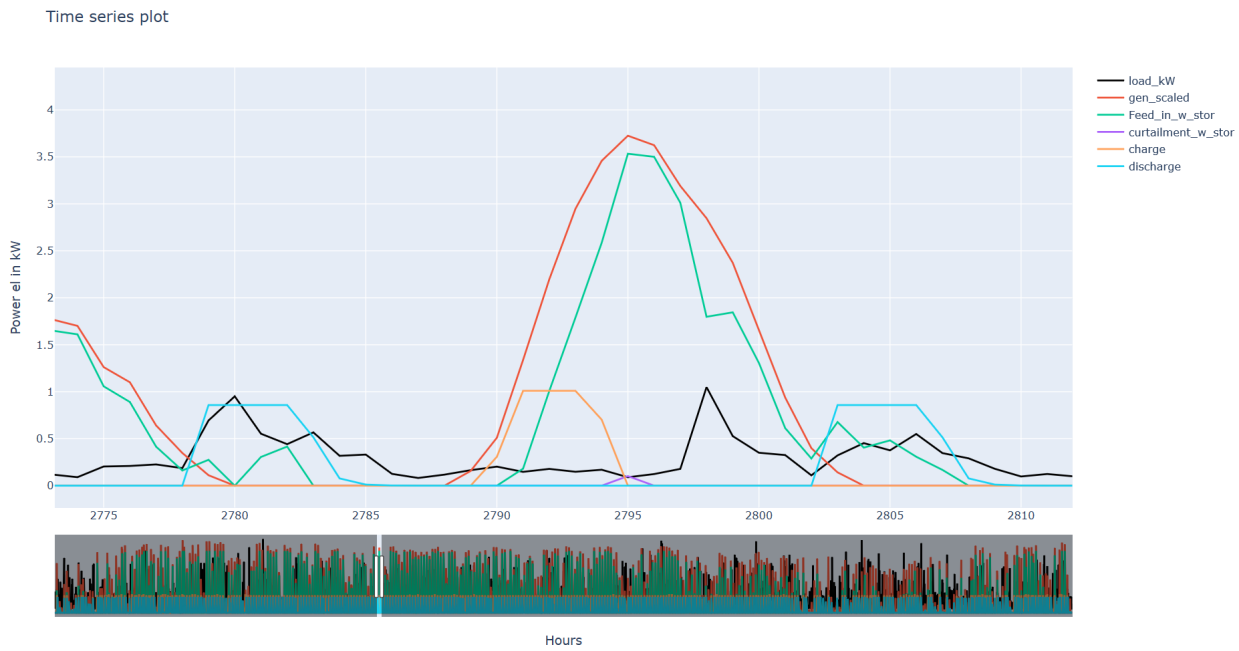
As a result, the hourly storage usage (df_PV_use), the load and generation sums as well as the autarky and own consumption values with and without storage are simulated and saved (result_df). In addition an interactive plot is generated when one system configuration (iterables_on = False) is calculated and opened in the browser.

2. Calculation of economic feasibility (IRR, payback, NPV)

```
df_PV_use = pd.merge(load, gen_norm, on='Datetime')
df_PV_use['gen_scaled'] = df_PV_use['generation_norm_kW'] * PV_cap
df_PV_use['residual_load_wo_stor'] = df_PV_use['load_kW'] - df_PV_use['gen_scaled']

# iteration over each hour without storage
for i in range(len(df_PV_use)):
    if df_PV_use.loc[i,'residual_load_wo_stor'] >= 0:
        df_PV_use.loc[i,'Feed_in_wo_stor'] = 0
        df_PV_use.loc[i,'curtailment_wo_stor'] = 0
    else:
        df_PV_use.loc[i,'Feed_in_wo_stor'] = df_PV_use.loc[i,'residual_load_wo_stor'] *(-1)
        df_PV_use.loc[i,'curtailment_wo_stor'] = 0
        if df_PV_use.loc[i,'Feed_in_wo_stor'] >= 0.7 * PV_cap:
            df_PV_use.loc[i,'curtailment_wo_stor'] = df_PV_use.loc[i,'Feed_in_wo_stor'] - 0.7 * PV_cap
            df_PV_use.loc[i,'Feed_in_wo_stor'] = 0.7 * PV_cap
```

```
206     # calculating battery charge
207     for i in range(2, len(df_PV_use)):
208         if df_PV_use.loc[i,'Feed_in_wo_stor'] > 0 and df_PV_use.loc[i-1,'SOC'] <= battery_cap:
209             if df_PV_use.loc[i,'residual_load_wo_stor'] *(-1) <= charge_cap:
210                 df_PV_use.loc[i,'charge'] = df_PV_use.loc[i,'Feed_in_wo_stor'] + df_PV_use.loc[i,'curtailment_wo_stor']
211             else: df_PV_use.loc[i,'charge'] = charge_cap
212             if (df_PV_use.loc[i-1,'SOC'] + df_PV_use.loc[i,'charge']) > battery_cap:
213                 df_PV_use.loc[i,'charge'] = battery_cap - df_PV_use.loc[i-1,'SOC']
214
215         # calculating battery discharge
216         if df_PV_use.loc[i,'residual_load_wo_stor'] > 0 and df_PV_use.loc[i-1,'SOC'] > 0:
217             if df_PV_use.loc[i-1,'SOC'] > discharge_cap:
218                 df_PV_use.loc[i,'discharge'] = discharge_cap * roundtrip_efficiency_bat
219             else: df_PV_use.loc[i,'discharge'] = df_PV_use.loc[i-1,'SOC'] * roundtrip_efficiency_bat
220
221         #Calculating SOC
222         df_PV_use.loc[i,'SOC'] = df_PV_use.loc[i-1,'SOC'] + df_PV_use.loc[i,'charge'] - df_PV_use.loc[i,'discharge']
223
224     #Calculate residual load with storage
```



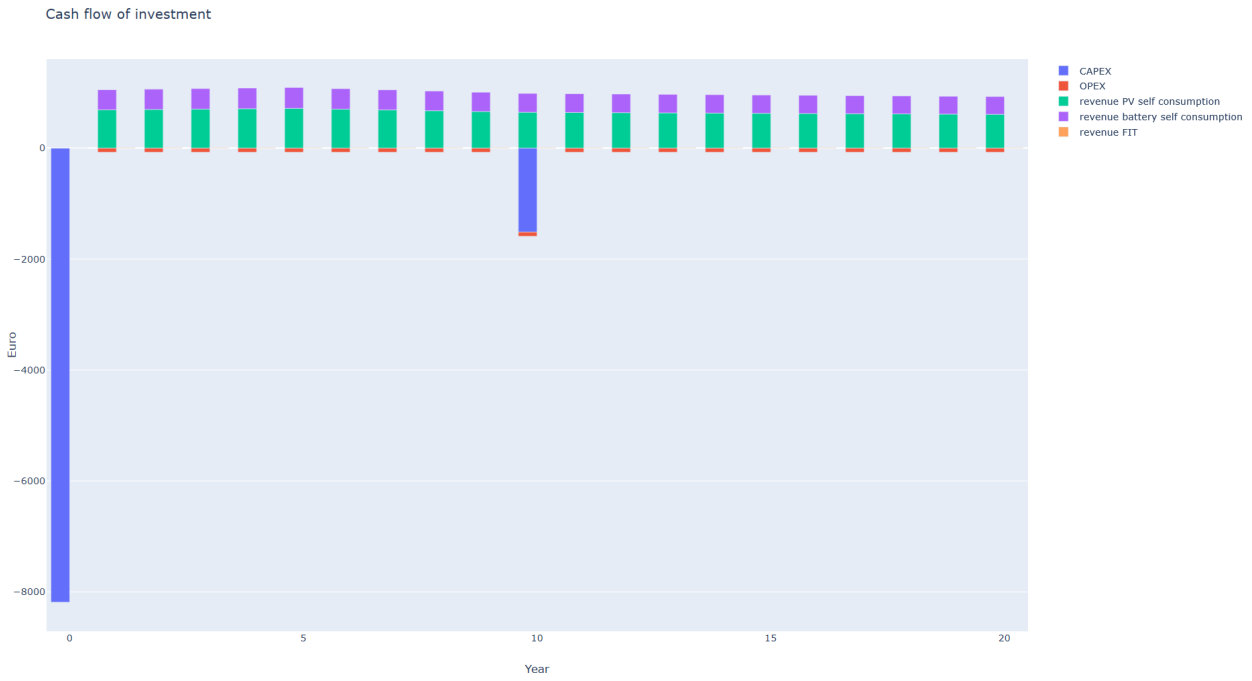
To calculate the economic efficiency of the HSS the cash flow over the technical life of the system is calculated. The cash flow is determined using the calculate_cashflow() (*API*) function. When determining the annual PV production, the degradation of the modules is taken into account in the form of an annual factor.

Costs:

Capital costs consist of investment costs (for PV and battery) and installation costs. They are incurred in year 0 and in the year the battery is replaced (defined by technical lifetime of the battery). Further expenses are the annual running costs (OPEX).

Revenues:

The revenues is made up of the PV system's own consumption, the battery's own consumption plus VAT (USTG, §19) and the grid feed-in. The annual net cash flow (income - expenditure) is discounted and accumulated using the assumed interest rate. On the basis of the cash flow, various parameters can be determined to calculate the economic efficiency of the system. All relevant data is written to the dataframe df_cashflow. Two plots of the cashflow are generated when only one system configuration and "iterables_on = False" is set and opened in the browser:



Economic Measures:

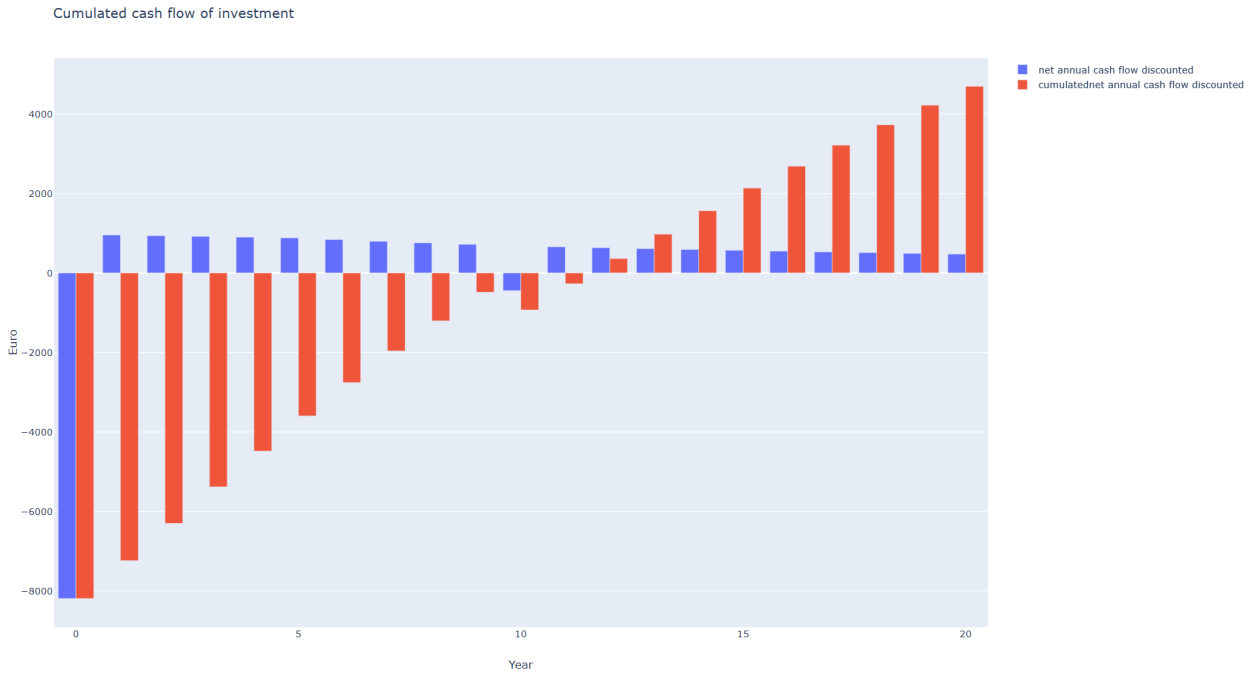
The calculate_payback function() determines the payback time.
The function calculate_npv() determines the net present value.
The function calculate_IRR() calculates the internal rate of return.

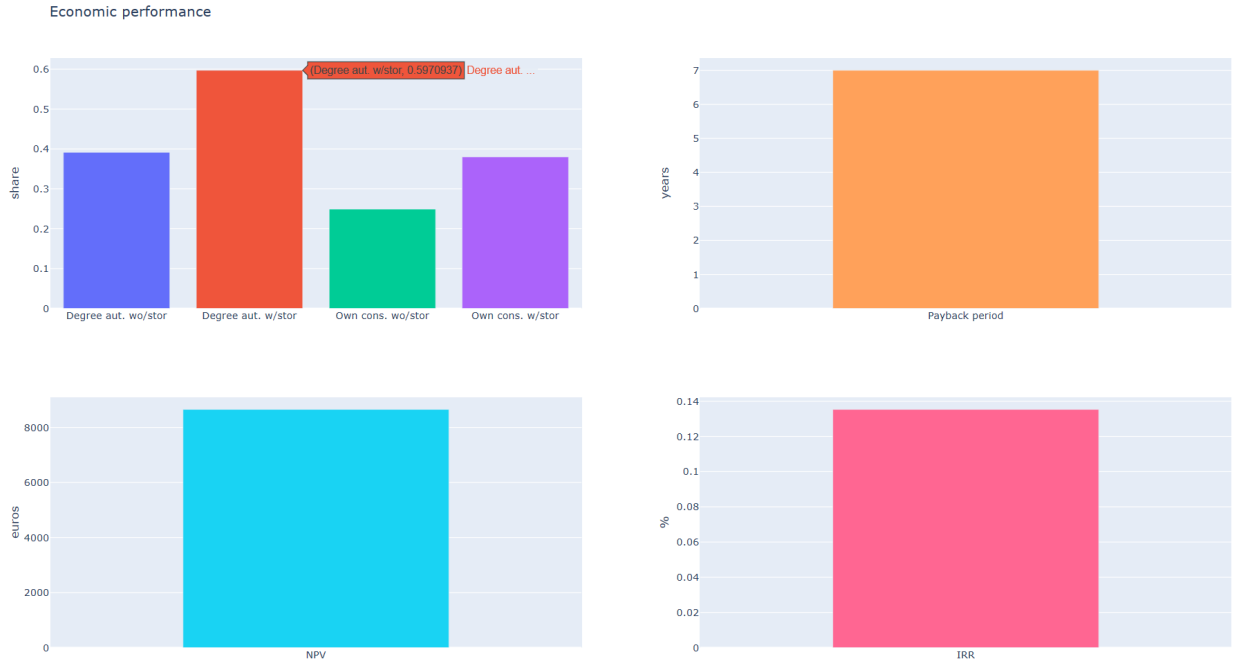
A plot of the economic measures and the autarky as well as own consumption shares is generated:

Iterations:

Since there are many possible variations regarding the degree of autarky as well as the economic feasibility of the HSS, the script *calc_UCM_economics* was designed in a way that a number of parameters can be specified as iteration parameters.

Iterables are:


Cumulated cash flow of investment
net annual cash flow discounted
cumulatednet annual cash flow discounted
Euro
Year
4000
2000
0
−2000
−4000
−6000
−8000
0
5
10
15
20

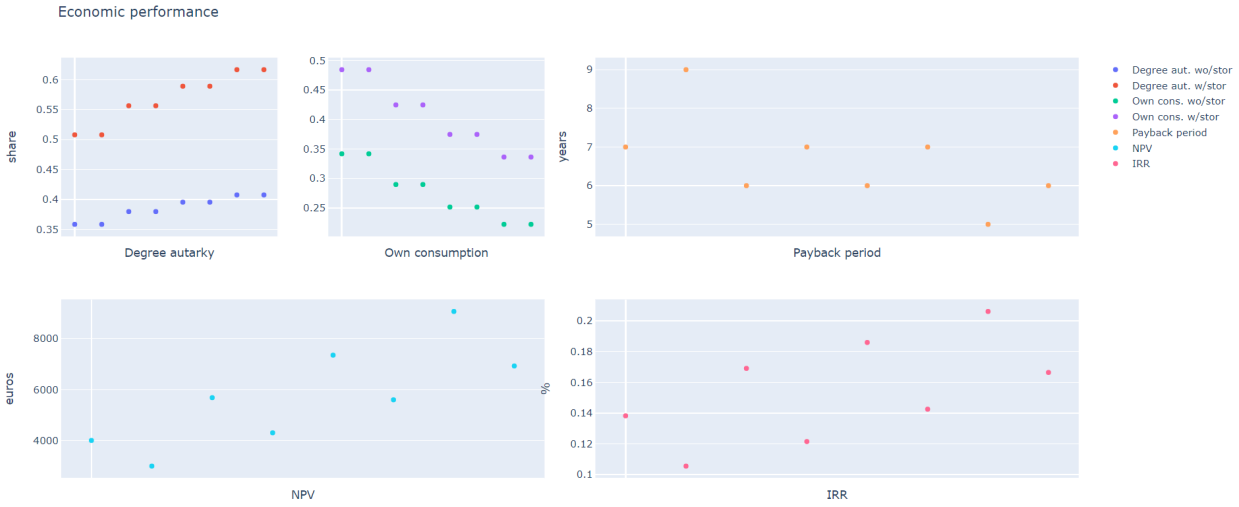

Economic performance
share
(Degree aut. w/stor, 0.597093...) Degree aut. ...
Degree aut. wo/stor
Degree aut. w/stor
Own cons. wo/stor
Own cons. w/stor
years
Payback period
euros
NPV
%
IRR

- start_years (year of calculation e.g. 2020, 2030, etc.)
- demand_scenarios (Defined in database or input file)
- regions (NUTS3-code)
- PV_orientations (East, South-West)
- ratio_PV_demands (e.g. 1)
- ratio_PV_Batterys (e.g. 1)
- WACCs (e.g. 0.018)
- scenario_CAPEX_PVs (Defined in database or input file)
- scenario_CAPEX_bats (Defined in database or input file)
- scenario_FITs (Defined in database or input file)
- scenario_EEXs (Defined in database or input file)
- scenario_consumption_prices (Defined in database or input file)

These can each be specified as a list. Some of the iterables can be entered directly in the script. Others are specified by scenarios stored in the database or in the Input folder. The user can also add his own new scenarios. The results of the iteration are stored in a csv-file (results_df) with the following variables and stored in the model output folder.

result_df = pd.DataFrame(columns = ['degree_autarky_wo_storage', 'degree_autarky_w_storage', 'own_consumption_wo_stor', 'own_consumption_w_stor', 'pay_back', 'NPV', 'IRR'], index = pd.MultiIndex.from_product(iterables))

For the graphical display of the results (iterables_on = True) should be set.



3&4 Utilities and preference shares

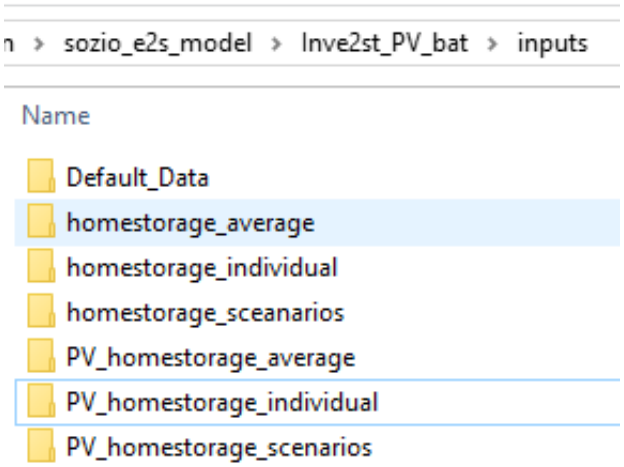
To calculate purchasing preferences, the attribute levels (time of realization, CAPEX, IRR/Paybacktime, Degree of autarky and Environmental impact) are defined in the form of scenarios.The scenarios are based on the possible characteristics of a use case for a particular year. The script *calc_UCM_economics* was developed for the parameters Degree of Autarky and IRR/Paybacktime in combination with the CAPEX, the results of which can be used for the calculation of preferences. Similar to the case of passenger cars, the numerical values for which interpolation (but no extrapolation) between the values is possible (CAPEX,IRR/Paybacktime,Degree of autarky) and those for which interpolation is not possible (time of realization, environmenatl impact) differ.

The calculation of the utilities is done with the function calculate_utilities(). The calculation of the preference probability can be performed by the functions calculate_logit_probabilities() or calculate_first_choice(). The functions were described in detail for the Use Case passenger cars.

Input Data

For the use case PV-homestorage system there is also the possibility of database connection as well as calculation without database. The setting can be made in the script Preferences_HSS (databaseconnection = True or False).

databaseconnection = False For the calculation without database, the data for the cases (investment_options) homestorage and PV-homestorage are provided, for average and individual resolution. It is explicitly stated that the individual data sets should be used for the calculation. The use of the average data sets leads to a significantly shorter calculation time and can therefore be used for test runs, but they show high inaccuracies. There is also a scenario folder for the two investment options, in which the attributes can be specified by year. Any new scenarios can be added at this point. Note that the attribute level for the continuous attributes must remain within the queried Discrete Choice values and the Discrete takes one of the predefined level values. The name of the csv file must be structured as follows: Scenario_name_attribute_level_per_year.csv



databaseconnection = True

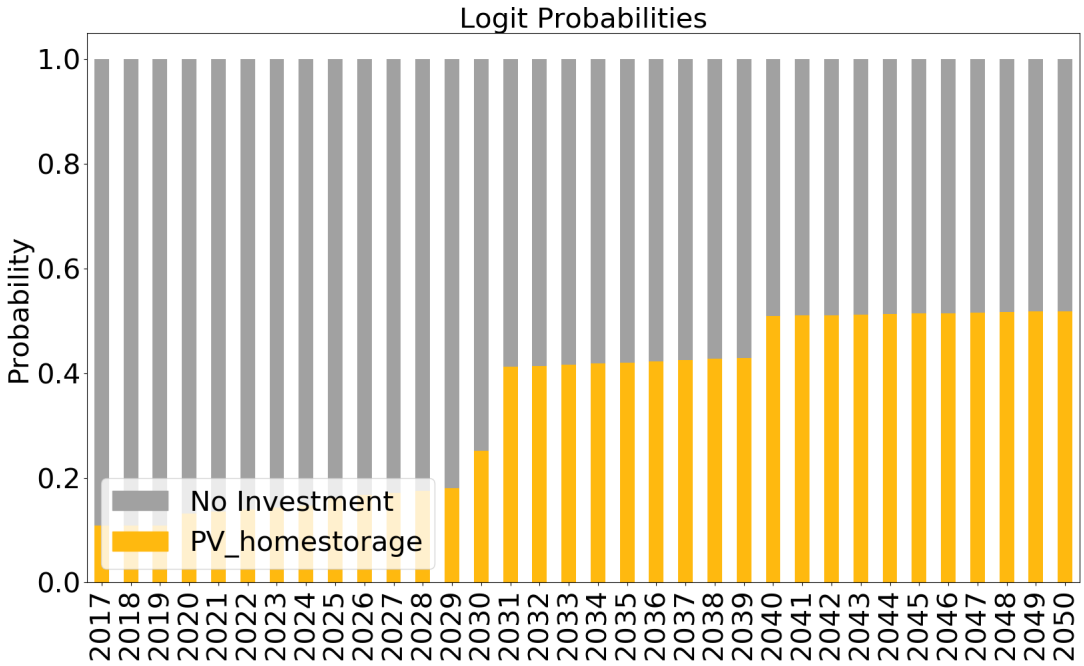
If the database connection is used, for each scenario a new folder is created in the Input folder, which contains all the data that is used.

Results:

As a result, a folder of the scenario is created under Results, which contains a csv file for the partial utilities and preferences. The plot of the preferences is saved.

3.3 3. Power-to-Gas

Work in progress


Logit Probabilities
Probability
1.0
0.8
0.6
0.4
0.2
0.0
No Investment
PV_homestorage
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050

CHAPTER 4

License

CHAPTER 5

Quick Start

5.1 Installation

Download the code from https://www.github.com

Hint: If working in Spyder IDE, please set the working directory to the root folder (Inve2st_Passenger_car).

5.2 Prerequisites

Please use Python 3.7. Following python modules are required and should be installed before running the framework.

- numpy
- scipy
- pandas
- scikit-learn
- psycopg2

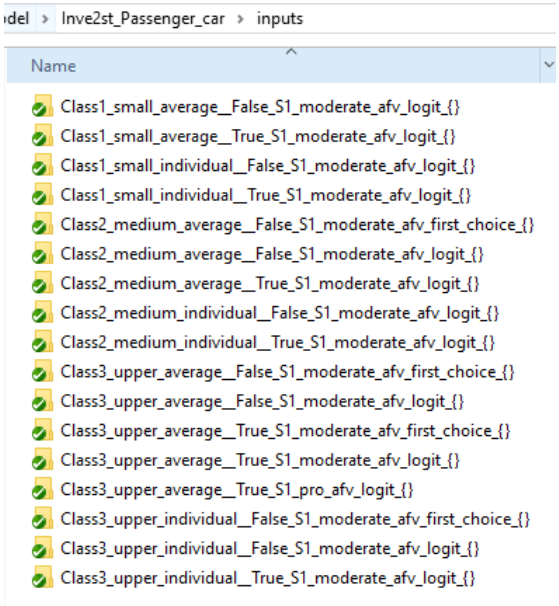
5.3 Minimum working example Passenger Cars

To run the simulation for passenger cars car_simulation.py needs to be executed. If no own PostgreSQL Database – according to the Inve2st schema is set up, the model can be run without database – reading in csv data from the Input folder. Within the car_simulation.py the following settings need to be put to work with csv files:

```
10
11 db_on = False
12 write_data = False # control variable to w
13
14 read_data = True
15
```

The csv files are provided for different cases:

```
92
93 #-------CALLING FUNCTIONS-------------------------------------------------------------------------------------------------------------
94
95 folder_name = investment_option + '_' + aggregation_level + '_' +'_'+str(tec_none)+ '_'+attribute_sceanrio + '_' + probabil
96
97 io = Investment_Options(folder_name,investment_option=investment_option,starting_year=201,DB=False,d_on=db_on,write_data=
98
99
100
101 #Creates folder for results
```



The options (scenarios) that are available are put as comments in the car_simulation.py and can be replaced by the other available options(e.g. investment_option = 'Class1_small' can be replaced by investment_option = 'Class2_medium'). The folder name in the input order shows which scenarios are available by default. A detailed description of the data can be found under "Data and Database". A short description of the options is provided in the code. In the folder inputs/scenario_data an excel sheet is provided containing attribute developments for 4 different scenarios, which can be filtered an replaced in the input/query_attribute_level_per_year.csv - if the scenario is not supplied by default. Ensure that the folder name is build according to the "folder_name" specification

All user settings can be made in the car_simulation.py

```
#######################################################################################################
#-------USER INPUT-----------------------------------------------------------------------------------

#------Car class-----------------------------------------------------------------

#'Class1_small'  -  passenger car containing minis, small cars and compact class
#'Class2_medium' -  passenger car containing mid-sized cars
#'Class3_upper'  -  passenger car containg upper mid-sized cars and luxury cars

investment_option='Class3_upper'

#------Value resolution for utilities----------------------------------------

#'individual_raw'   - individual values (containing the full sample)
#'average_raw'      - average values (average of full sample)
value_resolution = 'average_raw'

# Value resolution for utilities

#------Aggregation level-----------------------------------------------------

#'individual'  - individual values (containing the full sample) (Same as value resolution)
#'average'     - average values (containing the full sample) (Same as value resolution)
aggregation_level = 'average'


#------Attribute scenario development ----------------------------------------
#Defines a scenario in which the attributes (CAPEX, fuel price, range, CO2-tax, CO2-emissions and i

#'S1_contra_afv'   - in favor of conventional vehicles (CV) and against alternative vehicles (AV) (
#'S1_moderate_afv' - moderate for CV and AV
#'S1_pro_afv'      - in favor of AV
attribute_sceanrio ='S1_moderate_afv'

#------None option -----------------------------------------------------------
# additional choice for the investment decision tec_none = True means that also no car can be prefe
#tec_none = True    -  not suggested for stock calculation as exchange rate is given by growth facto
tec_none = False

#------Method to calculated preferences -------------------------------------
#'first_choice'  - always choice for alternative with highes utility
#'logit'         - Assumes logit distribution for choices

probability_calc_type = 'logit'

#------Assumption for growth rate of car stock  ------------------------------
#'S_decreasing_1' - 1% degrease per year
#'S_constant'     - constant car stock
#'S_decreasing_05'- decrease by 0.5% per year
growth_scenario = 'S_constant'

#------Assumptions for CO2-calculation----------------------------------------
# assumption of average passenger kilometers for calculating CO2-emissions [Pkm/car]
average_passenger_kilometers = 20900
specific_consumption_scenario = 'S_base'  #average consumption of cars [kWh/100km]
```

CHAPTER 6

Data and Database

6.1 Database

The foundation for the Inve2st Framework is a PostgreSQL Database. Queries have been defined to be able to use the framework as flexible as possible, e.g. for choosing different scenario_IDs for different kinds of data. All queries can be found in the modules.Queries *Link Einfügen API*

If the database is installed locally by the user, the database needs to be set as TRUE (self.db_on = True) within the module investment_options.

If the data from the database should be written into the input folder the function needs to be set as True (self.write_data = True)

6.2 Equipment Data

To be able to use the framework without the database, equipment data is provided for different scenarios and aggregation levels.

The data provided are the results of the database queries for specific settings. The provided data can be found in ... sozio_e2s_modelInve2st_Passenger_carinputs.

The names of the folders are a combination of the settings for the queries: investment_option _ aggregation_level _ (tec_none) _ttribute_sceanrio _ probability_calc_type_main_sub

One exemplary set of input files (as csv) is described here:

6.2.1 Class1_small_average__False_S1_moderate_afv_logit_{}

Description of the folder name

investment_option = Class1_small aggregation_level = average tec_none = False growth_scenario = S1_moderate_afv probability_calc_type = logit main_sub = { }

A subset of each input csv files is shown below:

List of investment option alternatives **(query_investment_option_alternatives)**

Table 1: query_investment_option_alternatives

0
tec_bev
tec_fcev
tec_cv
tec_none

Description of the alternatives:

List of all alternatives related to the car type alternatives:

1. tec_bev (battery electric vehicles)
2. tec_cv (conventional vehicles, diesel/gasoline)
3. tec_fecv (fuel cell electric vehicle)
4. tec_none (no car/optional)

Following are the options that the table columns could contain;

The attribute scenarios are in **(query_attribute_level_per_year)**

where: year: integer range between [2018,2050]

The cars have a list of attributes which have different attribute_level s (which are part of the discrete choice experiment)

1. **att_CAPEX (investment cost)**
    a. numeric range []
2. **att_Co2_tax (additional CO2-tax on gazoline and diesel)**
    a. co2_tax
    b. no_co2_tax
3. **att_v_type (vehicle type)**
    a. BEV (battery electric vehicle)
    b. CV (conventional vehicle)
    c. FCEV (fuel cell electric vehicle)
4. **att_fuel_cost (fuel cost per 100 km)**
    a. numeric range []
5. **att_infrast (charging infrastructure)**
    a. strong_res (with strong restrictions)
    b. with_res (with restrictions)
    c. no_res (without restrictions)
6. **att_range (average maximum range for the vehicle)**
    a. numeric range []
7. **att_w2w (Weel2well CO2-emissions)**

a. low_co2

b. medium_co2

c. high_co2

The query_attribute_level_per_year delivers the following table

Table 2: query_attribute_level_per_year

alternative	year	attribute	attribute_level
tec_bev	2018	att_CAPEX	23967
tec_cv	2050	att_CAPEX	22250
tec_fcev	2018	att_CAPEX	46218
tec_cv	2018	att_Co2_tax	no_co2_tax
tec_cv	2050	att_Co2_tax	no_co2_tax
tec_bev	2018	att_fuel_cost	3.58
tec_cv	2050	att_fuel_cost	5.62
tec_fcev	2018	att_fuel_cost	5.61
tec_bev	2018	att_infrast	with_res
tec_bev	2050	att_infrast	no_res
tec_fcev	2018	att_infrast	strong_res
tec_bev	2018	att_range	202
tec_bev	2018	att_v_type	BEV
tec_cv	2050	att_v_type	CV
tec_fcev	2050	att_v_type	FCEV
tec_bev	2018	att_w2w	low_co2
tec_cv	2018	att_w2w	medium_co2
tec_fcev	2050	att_w2w	low_co2

The attribute and attribute_level should be same as defined above, whereas the attribute levels for the continuous values need to stay within the limits of the discrete chocie experiment.

List of discrete attributes **(query_discrete_attributes)**

Table 3: query_discrete_attributes

0
att_Co2_tax
att_infrast
att_v_type
att_w2w

The **query_attribute_level_putility**

Data source is the discrete chocice experiment data. The average values are the average from individual values. In this case the respondend_ID equals a numeric value. Note that individual values are prefered because the uncertainty with average values is high.

Table 4: query_attribute_level_putility

attribute	attribute_level	partial_utility_value	respondend	value_resolution
att_v_type	FCEV	-0.204667348	resp_average	average_raw
att_v_type	CV	0.072138788	resp_average	average_raw
att_CAPEX	15600	1.69783607	resp_average	average_raw
att_CAPEX	26000	0.732658309	resp_average	average_raw
att_CAPEX	36400	-0.492675808	resp_average	average_raw
att_range	800	0.891915896	resp_average	average_raw
att_CAPEX	52000	-1.937818571	resp_average	average_raw
att_fuel_cost	2	0.813541675	resp_average	average_raw
att_fuel_cost	4	0.474195979	resp_average	average_raw
att_fuel_cost	7.2	-0.353221	resp_average	average_raw
att_fuel_cost	10	-0.934516343	resp_average	average_raw
att_range	160	-1.313911791	resp_average	average_raw
att_range	320	-0.139127507	resp_average	average_raw
att_range	560	0.561123402	resp_average	average_raw
att_infrast	strong_res	-0.791598823	resp_average	average_raw
att_infrast	with_res	0.080949569	resp_average	average_raw
att_infrast	no_res	0.710649254	resp_average	average_raw
att_w2w	no_co2	0.342227117	resp_average	average_raw
att_w2w	low_co2	0.350126564	resp_average	average_raw
att_w2w	medium_co2	0.017020044	resp_average	average_raw
att_Co2_tax	co2_tax	-0.228203433	resp_average	average_raw
att_Co2_tax	no_co2_tax	0.228203433	resp_average	average_raw
att_w2w	high_co2	-0.709373725	resp_average	average_raw
att_v_type	BEV	0.13252856	resp_average	average_raw

The **query_utility_none_option** is separated from the other alternatives, as it is optional

Table 5: query_utility_none_option

attribute	attribute_level	utility_value	respondend	value_resolution
att_none	none	3.131344833	resp_average	average_raw

The query **query_stock** delivers the historical stock, containing the registration year of the cars.

Table 6: query_stock

technology	registeration_year	stock_year	num_cars	sources	quality	date	modifier
tec_car	1979	2001	71701	KBA.2017b	1	26.02.2019	D. Baumann
tec_car	1979	2002	58515	KBA.2017b	1	26.02.2019	D. Baumann
tec_car	2016	2018	3130587	KBA.2017b	1	26.02.2019	D. Baumann
tec_car	2017	2018	3152304	KBA.2017b	1	26.02.2019	D. Baumann

The **query_stoc_init_year** delivers the actual stock (2018), containing the registration year of the cars.

Table 7: query_stoc_init_year

reg_year	stock_year	num_cars
1979	2018	30069
1980	2018	23614
1981	2018	25607
1982	2018	28930
1983	2018	35351
1984	2018	34813
1985	2018	36810
1986	2018	47968
1987	2018	59563
1988	2018	70841
1989	2018	95171
1990	2018	127771
1991	2018	172523
1992	2018	179023
1993	2018	175977
1994	2018	219672
1995	2018	330614
1996	2018	447139
1997	2018	600044
1998	2018	805294
1999	2018	1073831
2000	2018	1093988
2001	2018	1260961
2002	2018	1402213
2003	2018	1616812
2004	2018	1765529
2005	2018	2010822
2006	2018	2310465
2007	2018	2198307
2008	2018	2331512
2009	2018	3142495
2010	2018	2409507
2011	2018	2710883
2012	2018	2681385
2013	2018	2573462
2014	2018	2681496
2015	2018	3028961
2016	2018	3130587
2017	2018	3152304

The **query_car_class_share** delivers percentage of the chosen car classes (small, medium or upper and luxury on the basis of the investment-option)

Table 8: query_car_class_share

0
0.74

The **query_sub_technology_share** delivers total number of cars by vehicle type (tec_bev, tec_fcev, tec_cv)

Table 9: query_sub_technology_share

year	stock	alternative
2009	1452	tec_bev
2010	1588	tec_bev
2011	2307	tec_bev
2012	4541	tec_bev
2013	7114	tec_bev
2014	12156	tec_bev
2015	18948	tec_bev
2016	25502	tec_bev
2017	34022	tec_bev
2009	41318779	tec_cv
2010	41734193	tec_cv
2011	42281656	tec_cv
2012	42922141	tec_cv
2013	43421478	tec_cv
2014	43836993	tec_cv
2015	44382343	tec_cv
2016	45044025	tec_cv
2017	45758644	tec_cv
2009	0	tec_fcev
2010	0	tec_fcev
2011	0	tec_fcev
2012	0	tec_fcev
2013	0	tec_fcev
2014	0	tec_fcev
2015	0	tec_fcev
2016	0	tec_fcev
2017	0	tec_fcev
2018	0	tec_fcev
2018	53861	tec_bev
2018	46410016	tec_cv

The **query_investor_stock_share** delivers percentage of the investor (private owners) of the total car stock

Table 10: query_investor_stock_share

0
0.892

The **query_car_stock_scenario** delivers the annual percentage increase or decrease of the total car stock (e.g. 0.01,0, -0.01)

Table 11: query_car_stock_scenario

0
0

The **query_car_target_value** defines the total number of cars per vehicle type as a target value

Table 12: query_car_target_value

0	1	2
17349566	2030	tec_bev
33090754	2050	tec_bev
34430399	2030	tec_cv
1121569	2050	tec_cv
163	2030	tec_fcev
17072836	2050	tec_fcev

The **query_spec_emissions_cv** gives values for the specific emissions of conventional vehicles for the car size of the investment option per registration year [gCO2/km]

Table 13: query_spec_emissions_cv

c_specific_emissions_construction_year	c_specific_emissions_value
1979	288
1980	283
1981	277
1982	272
1983	266
1984	261
1985	255
1986	250
1987	244
1988	239
1989	233
1990	228
1991	222
1992	217
1993	211
1994	206
1995	200
1996	199
1997	198
1998	198
1999	197
2000	196
2001	195
2002	194
2003	194
2004	193
2005	192
2006	191
2007	190
2008	190
2009	189
2010	188
2011	188
2012	187
2013	186
2014	186

Continued on next page

Table 13 – continued from previous page

c_specific_emissions_construction_year	c_specific_emissions_value
2015	185
2016	185
2017	184
2018	183
2019	158
2020	154
2021	151
2022	147
2023	144
2024	141
2025	137
2026	134
2027	132
2028	129
2029	126
2030	123
2031	121
2032	118
2033	116
2034	114
2035	112
2036	110
2037	108
2038	106
2039	104
2040	103
2041	101
2042	100
2043	99
2044	98
2045	96
2046	96
2047	95
2048	94
2049	93
2050	93

The **query_spec_emissions_electricity_mix** gives values for the specific emissions of the electricty mix for a choosen scenario per year [kgCO2/kWh]

Table 14: query_spec_emissions_electricity_mix

c_specific_emissions_electricity_mix_year	c_specific_emissions_value
2018	0.51
2019	0.51
2020	0.51
2021	0.48
2022	0.45
2023	0.42
2024	0.38

Continued on next page

Table 14 – continued from previous page

c_specific_emissions_electricity_mix_year	c_specific_emissions_value
2025	0.35
2026	0.32
2027	0.29
2028	0.26
2029	0.23
2030	0.2
2031	0.19
2032	0.18
2033	0.16
2034	0.15
2035	0.14
2036	0.13
2037	0.12
2038	0.11
2039	0.09
2040	0.08
2041	0.08
2042	0.07
2043	0.07
2044	0.06
2045	0.05
2046	0.05
2047	0.04
2048	0.04
2049	0.03
2050	0.03

The **query_specific_consumption** gives values for the specific consumption pf BEV and FCEV per construction year and choosen car size [kWh/100km]

Table 15: query_specific_consumption

c_specific_consumption_construction_year	c_specific_consumption_technology	c_specific_consumption_value
2019	tec_fcev	28.4
2020	tec_fcev	28.4
2021	tec_fcev	28.4
2022	tec_fcev	28.4
2023	tec_fcev	28.4
2024	tec_fcev	28.4
2025	tec_fcev	28.4
2026	tec_fcev	28.4
2027	tec_fcev	28.4
2028	tec_fcev	28.4
2029	tec_fcev	28.4
2030	tec_fcev	28.4
2031	tec_fcev	28.4
2032	tec_fcev	28.4
2033	tec_fcev	28.4
2034	tec_fcev	28.4
2035	tec_fcev	28.4

Continued on next page

Table 15 – continued from previous page

c_specific_consumption_construction_year	c_specific_consumption_technology	c_specific_consumption_value
2036	tec_fcev	28.4
2037	tec_fcev	28.4
2038	tec_fcev	28.4
2039	tec_fcev	28.4
2040	tec_fcev	28.4
2041	tec_fcev	28.4
2042	tec_fcev	28.4
2043	tec_fcev	28.4
2044	tec_fcev	28.4
2045	tec_fcev	28.4
2046	tec_fcev	28.4
2047	tec_fcev	28.4
2048	tec_fcev	28.4
2049	tec_fcev	28.4
2050	tec_fcev	28.4
2019	tec_bev	24.0
2020	tec_bev	23.84
2021	tec_bev	23.67
2022	tec_bev	23.51
2023	tec_bev	23.35
2024	tec_bev	23.18
2025	tec_bev	23.02
2026	tec_bev	22.85
2027	tec_bev	22.69
2028	tec_bev	22.53
2029	tec_bev	22.36
2030	tec_bev	19.9
2031	tec_bev	19.69
2032	tec_bev	19.48
2033	tec_bev	19.27
2034	tec_bev	19.06
2035	tec_bev	18.85
2036	tec_bev	18.64
2037	tec_bev	18.43
2038	tec_bev	18.22
2039	tec_bev	18.01
2040	tec_bev	17.8
2041	tec_bev	17.59
2042	tec_bev	17.38
2043	tec_bev	17.17
2044	tec_bev	16.96
2045	tec_bev	16.75
2046	tec_bev	16.54
2047	tec_bev	16.33
2048	tec_bev	16.12
2049	tec_bev	15.91
2050	tec_bev	17.0

6.3 Database Credentials

To be able to connect to database, the login credentials should be set in credentials.json.

```
{
        "dbname": "sozio_e2s",
        "host": "db_host",
        "user": "my_user",
        "password" : "mypassword"
}
```

CHAPTER 7

Funding

The Inve2st framework was developed in the context of the research project “Open source Energiesystemmodellierung – Einfluss von soziokulturellen Faktoren auf Transformationspfade des deutschen Energiesystems (Sozio-E2S), Teilvorhaben: Open source Energiesystemmodellierung und umweltpsychologisches Akzeptanzverhalten“, funded by the federal ministry for economic affairs and energy BMWi, Förderkennzeichen 03ET4041A.

Supported by:

CHAPTER 8

Publications and Links

The projekt homepage can be found under the following link:

https://www.ise.fraunhofer.de/de/forschungsprojekte/sozio-e2s.html

List of publications:

Jülch, V., Senkpiel, C., Kost, C., Hartmann, N., & Schlegl, T. (2018). Meta Study on Future Cross-sectoral Decarbonization Target Systems in Comparison to Current Status of Technologies. Freiburg. Retrieved from https://www.ise.fraunhofer.de/content/dam/ise/en/documents/publications/studies/Meta_Study_Crossectoral_Decarbonization_Target_Systems.pdf

Schrage, A., Wassermann, S., Berneiser, J., & Gölz, S. (2018). Sozialwissenschaftliche Determinanten von Investitionsentscheidungen in erneuerbare Energietechnologien (Stuttgarter Beiträge zur Risiko- und Nachhaltigkeitsforschung). Retrieved from https://elib.uni-stuttgart.de/bitstream/11682/9607/1/%c3%9cberblickstudie_Schrage%20et%20al%202018.pdf

CHAPTER 9

API

class modules.Queries.**Query**

Bases: object

This Class contains methods which help in generating SQL queries for various purposes in the Investment_Options Class.

query_applications(*c_investment_options_description*, *c_application_characteristics_validity_time_pk*, *c_application_description*, *c_application_characteristics_scenario_pk*, *c_regions_name*)

Query applications

c_investment_options_description: str Investment option description

c_application_characteristics_validity_time_pk: str Application characteristics validity time

c_application_description: str Application Description

c_application_characteristics_scenario_pk: str application characteristics scenario

c_regions_name: str Regions Name

SQL string

query_applications_characteristics(*c_investment_options_description*, *start_year*, *end_year*, *c_application_description*, *c_application_characteristics_scenario_pk*, *c_regions_name*, *tb_application_characteristic_types_description*)

Query Application Characteristics Parameters ———- c_investment_options_description: str

Investment options description

start_year: str Start year

end_year: str End year

c_application_description: str Application description

c_application_characteristics_scenario_pk: str Application characteristic scenario

c_regions_name: str Regions Name

tb_application_characteristic_types_description: str Application characteristic types description

SQL string

query_applications_characteristics_opts(*c_investment_options_description*, *start_year*, *end_year*, *c_application_description*, *c_regions_name*, *tb_application_characteristic_types_description*)

c_investment_options_description: str Investment options description

start_year: str Start year

end_year: str End year

c_application_description: str Application description

c_regions_name: str Regions name

tb_application_characteristic_types_description: str Application characteristic types description

SQL string

query_applications_demand(*c_investment_options_description*, *start_year*, *end_year*, *c_application_description*, *c_application_characteristics_scenario_pk*, *c_regions_name*, *tb_application_characteristic_types_description*)

c_investment_options_description: str Investment options description

start_year: str Start year

end_year: str End year

c_application_description: str Application description

c_application_characteristics_scenario_pk: str Application characteristics scenario

c_regions_name: str Regions name

tb_application_characteristic_types_description: str Application characteristic types description

SQL string

query_applications_demand_opts(*c_investment_options_description*, *start_year*, *end_year*, *c_application_description*, *c_regions_name*, *tb_application_characteristic_types_description*)

c_investment_options_description: str Investment options description

start_year: str Start year

end_year: str End year

c_application_description: str Application description

c_regions_name: str Regions name

tb_application_characteristic_types_description: str Application characteristic types description

SQL string

query_applications_size(*c_investment_options_description*, *starting_year*, *c_application_description*, *c_application_characteristics_scenario_pk*, *c_regions_name*, *tb_application_characteristic_types_description*)

c_investment_options_description: str Investment options description

starting_year: str Starting year

c_application_description: str Application description

c_application_characteristics_scenario_pk: str Application characteristic scenario

c_regions_name: str Regions name

tb_application_characteristic_types_description: str Application characteristic types

SQL string

query_attribute_level_per_year(*investment_option*, *attribute_sceanrio*)

query_attribute_level_putility(*investment_option*, *value_resolution*)

query_attributes(*investment_option*)

query_car_class_share(*investment_option*)

query_car_stock_scenario(*investment_option*, *car_stock_scenario_ID*)

query_car_target_system(*investment_option*, *car_target_system_scenario*)

query_car_target_value(*investment_option*, *car_target_scenario*)

query_consumer_prices(*c_investment_options_description*, *c_consumertype_description*, *c_consumption_prices_scenario*, *start_year*, *end_year*)

c_investment_options_description: str Investment options description

c_consumertype_description: str Consumer type desciption

c_consumption_prices_scenario: str Consumer prices scenario

start_year: str Start year

end_year: str End year

SQL string

query_consumer_prices_opts(*c_investment_options_description*, *c_consumertype_description*, *start_year*, *end_year*)

c_investment_options_description: str Investment options description

c_consumertype_description: str Consumer type description

start_year: str Start year

end_year: str End year

SQL string

query_consumer_prices_yearly_avg(*c_investment_options_description*, *c_consumertype_description*, *c_consumption_prices_scenario*, *start_year*, *end_year*)

c_investment_options_description: str Investment options description

c_consumertype_description: str Consumer type description

c_consumption_prices_scenario: str Consumption prices scenario

start_year: str Start year

end_year: str End year

SQL string

query_discrete_attributes(*investment_option*)

query_economic_parameters(*c_investment_options_description*, *c_regions_name*, *c_economic_parameter_validity_time*, *c_economic_parameter_scenario_pk*)

c_investment_options_description: str Investment options description

c_regions_name: str Regions name

c_economic_parameter_validity_time: str Economic parameter validity time

c_economic_parameter_scenario_pk: str Economic parameter scenario

SQL string

query_economic_parameters_2(*c_investment_options_description*, *c_regions_name*, *starting_year*, *c_economic_parameter_scenario_pk*, *c_economic_parameter_description*)

c_investment_options_description: str Investment options description

c_regions_name: str Regions name

starting_year: str Starting year

c_economic_parameter_scenario_pk: str Economic parameter scenario

c_economic_parameter_description: str Economic parameter description

SQL string

query_economic_parameters_economic_lifetime(*c_investment_options_description*, *c_regions_name*, *starting_year*, *c_economic_parameter_scenario_pk*, *c_economic_parameter_description*)

c_investment_options_description: str Investment options description

c_regions_name: str regions name

starting_year: str starting year

c_economic_parameter_scenario_pk: str Economic parameter scenario

c_economic_parameter_description: str Economic parameter description

SQL string

query_economic_parameters_economic_lifetime_opts(*c_investment_options_description*, *c_regions_name*, *starting_year*, *c_economic_parameter_description*)

c_investment_options_description: str Investment options description

c_regions_name: str Regions name

starting_year: str Starting year

c_economic_parameter_description: str economic parameter description

SQL string

query_emissions_electricity_mix(*specific_emissions_electricity_mix_scenario*)

query_financial_parameters(*c_investors_description*, *c_financial_parameter_scenario_pk*, *c_regions_name*, *c_financial_parameter_types_description*, *starting_year*)

c_investors_description: str Investors description

c_financial_parameter_scenario_pk: str Financial parameter scenario

c_regions_name: str Regions name

c_financial_parameter_types_description: str Financial parameter types description

starting_year: str Starting year

SQL string

query_financial_parameters_opts(*c_investors_description*, *c_regions_name*, *c_financial_parameter_types_description*, *starting_year*)

c_investors_description: str Investors description

c_regions_name: str Regions name

c_financial_parameter_types_description: str Financial parameter types description

starting_year: str Starting year

SQL string

query_historical_annual_installations()

SQL string

query_historical_annual_installations_main(*c_investment_options_description*, *c_application_description*, *c_regions_name*)

c_investment_options_description: str Investment options description

c_application_description: str Application description

c_regions_name: str Regions name

SQL string

query_historical_annual_installations_pv()

SQL string

query_historical_annual_installations_sub(*c_regions_name*, *c_application_description*, *tec*)

c_regions_name: str Regions name

c_application_description: str Application description

tec: str technology

SQL string

query_importances(*investment_option*, *aggregation_level*)

query_investment_costs(*c_economic_parameter_description*, *c_investment_options_description*, *c_regions_name*, *c_economic_parameter_validity_time*, *c_economic_parameter_scenario_pk*)

c_economic_parameter_description: str Economic parameter description

c_investment_options_description: str Investment options description

c_regions_name: str Regions name

c_economic_parameter_validity_time: str Economic parameter validity time

c_economic_parameter_scenario_pk: str Economic parameter scenario

SQL string

query_investment_option_alternatives(*investment_option*)

query_investor_stock_share(*investment_option*)

query_market_phases()
SQL string

query_meteological_data()

query_nominal_rate(*start_year*)

start_year: str Start year

SQL string

query_political_incentives(*c_investment_options_description*, *c_application_description*, *c_political_instrument_scenario*, *tb_political_instrument_types_description*)

c_investment_options_description: str Investment options description

c_application_description: str Application description

c_political_instrument_scenario: str Political Instrument scenario

tb_political_instrument_types_description: str Political instrument types description

SQL string

query_political_incentives_opt(*c_investment_options_description*, *c_application_description*, *tb_political_instrument_types_description*)

c_investment_options_description: str Investment options description

c_application_description: str Application description

tb_political_instrument_types_description: str Political instrument types description

SQL string

query_potential(*c_technology_pfk*)

c_technology_pfk: str technology primary key

SQL string

query_specific_consumption(*investment_option*, *specific_consumption_scenario*)

query_specific_emissions(*investment_option*, *specific_emissions_scenario*)

query_stoc_init_year(*investment_option*, *year*)

query_stock_scenario(*c_application_description*, *c_investment_options_description*, *c_regions_name*)

c_application_description: str Application description

c_investment_options_description: str Investment options description

c_regions_name: str Regions name

SQL string

query_sub_group(*main*, *sub*, *investment_option*)

query_sub_technology_share(*investment_option*)

query_technical_lifetime(*c_investment_options_description*, *c_technical_characteristics_types_description*, *c_technology_characteristics_validity_period_pk*)

c_investment_options_description: str Investment options description

c_technical_characteristics_types_description: str Technical characteristics types description

c_technology_characteristics_validity_period_pk: str Technology characteristics validity period

SQL string

query_technologies_name(*tec_pk*)

query_technology_relation(*c_investment_options_description*)

c_investment_options_description: str Investment options description

SQL string

query_total_car_stock(*investment_option*)

query_total_car_stock_w_registration_year(*investment_option*)

query_utility_energy_transition()

query_utility_grid_independence()

query_utility_none_option(*investment_option*, *value_resolution*)

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

M

Q