Production-Function Approach to Portfolio Evaluation

Version 1.5 Draft

19 September 2020

# Summary

Our production-function approach to R&D portfolio evaluation is mathematically formulated as a stochastic multi-objective decision-optimization problem and is implemented in the Python programming language. The framework abstracts the technology-independent aspects of the problem into a generic computational schema and enables the modeler to specify the technology-dependent aspects in a set of data tables and Python functions. This approach not only minimizes the labor needed to add new technologies, but it also enforces uniformity of financial, mass-balance, and other assumptions in the analysis.

The framework is scalable, supporting rapid computation on laptops computer and large-ensemble studies on high-performance computers (HPC). The use of vectorized operations for the stochastic calculations and of response-surface fits for the portfolio evaluations minimizes the computational resources needed for complex multi-objective optimizations. The software handles parameterized studies such as tornado plots, Monte-Carlo sensitivity analyses, and a generalization of epsilon-constraint optimization.

All values in the data tables may be probability distributions, specified by Python expressions using a large library of standard distributions, or the values may be simple numbers. Expert opinion is encoded through these distributions. The opinions may be combined prior to simulator or subsequent to it.

Four example technologies have been implemented as examples illustrating framework’s use: biorefineries, electrolysis, residential photovoltaics (PV), and utility-scale PV. A desktop user interface allows exploration of the cost-benefit trade-offs in portfolio decision problems.

Below we detail the mathematical formulation and its implementation as a Python module with user-specified data tables and technology functions. We also provide a sample analysis that exercises the framework’s main features.

# Mathematical formulation

We separate the financial and conversion-efficiency aspects of a production process, which are generic across all technologies, from the physical and technical aspects, which are necessarily specific to the particular process. The motivation for this is that the financial and waste computations can be done uniformly for any technology (even for disparate ones such as PV cells and biofuels) and that different experts may be required to assess the cost, waste, and techno-physical aspects of technological progress. Table 1 defines the indices that are used for the variables that are defined in Table 2.

Table 1: Definitions for set indices used for variable subscripts.

|  |  |  |
| --- | --- | --- |
| Set | Description | Examples |
|  | capital | equipment |
|  | fixed cost | rent, insurance |
|  | input | feedstock, labor |
|  | output | product, co-product, waste |
|  | metric | cost, jobs, carbon footprint, efficiency, lifetime |
|  | technical parameter | temperature, pressure |
|  | technology type | electrolysis, PV cell |
|  | scenario | the result of a particular investment |
|  | investment category | investment alternatives |
|  | investment | a particular investment |
|  | portfolio | a basket of investments |

Table 2: Definitions for variables.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Description | Units |
|  | calculated | unit cost | USD/unit |
|  | function | capital cost | USD |
|  | cost | lifetime of capital | year |
|  | cost | scale of operation | unit/year |
|  | function | fixed cost | USD/year |
|  | input | input quantity | input/unit |
|  | calculated | ideal input quantity | input/unit |
|  | waste | input efficiency | input/input |
|  | cost | input price | USD/input |
|  | calculated | output quantity | output/unit |
|  | calculated | ideal output quantity | output/unit |
|  | waste | output efficiency | output/output |
|  | cost | output price (+/-) | USD/output |
|  | calculated | metric | metric/unit |
|  | function | production function | output/unit |
|  | function | metric function | metric/unit |
|  | parameter | technical parameter | (mixed) |
|  | variable | scenario inputs | (mixed) |
|  | variable | scenario outputs | (mixed) |
|  | function | scenario evaluation | (mixed) |
|  | function | scenario probability | 1 |
|  | variable | investment cost | USD |
|  | random variable | investment outcome | (mixed) |
|  | random variable | portfolio outcome | (mixed) |
|  | calculated | portfolio cost | USD |
|  | parameter | minimum portfolio cost | USD |
|  | parameter | maximum portfolio cost | USD |
|  | parameter | minimum category cost | USD |
|  | parameter | maximum category cost | USD |
|  | parameter | minimum output/metric | (mixed) |
|  | parameter | maximum output/metric | (mixed) |
| , | operator | evaluate probabilities | (mixed) |

## Cost

The cost characterizations (capital and fixed costs) are represented as functions of the scale of operations and of the technical parameters in the design:

* Capital cost: .
* Fixed cost: .

The per-unit cost is computed using a simple levelization formula:

## Waste

The waste relative to the idealized production process is captured by the parameters. Expert elicitation might estimate how the s would change in response to R&D investment.

* Waste of input: .
* Waste of output: .

## Production

The production function idealizes production by ignoring waste, but accounting for physical and technical processes (e.g., stoichiometry). This requires a technical model or a tabulation/fit of the results of technical modeling.

## Metrics

Metrics such as efficiency, lifetime, or carbon footprint are also compute based on the physical and technical characteristics of the process. This requires a technical model or a tabulation/fit of the results of technical modeling. We use the convention that higher values are worse and lower values are better.

## Scenarios

A *scenario* represents a state of affairs for a technology . If we denote the scenario as , we have the tuple of input variables

and the tuple of output variables

and their relationship

given the tuple of functions

for the technology of the scenario.

## Investments

An *investment* assigns a probability distribution to scenarios:

.

such that

or ,

depending upon whether one is performing the computations discretely or continuously. Expectations and other measures on probability distributions can be computed from the . We treat the outcome as a random variable for the outcomes according to the distribution .

Because investment options may be mutually exclusive, as is the case for investing in the same R&D at different funding levels, we say is the set of mutually exclusive investments (i.e., only one can occur simultaneously) in investment category : investments in different categories can be combined arbitrarily, but just one investment from each may be chosen.

Thus the universe of all portfolios is , so a particular portfolio has components . The overall outcome of a portfolio is a random variable:

The cost of an investment in one of the constituents is , so the cost of a porfolio is:

## Decision problem

The multi-objective decision problem is

such that

,

,

,

where and are the expectation operator , the value-at-risk, or another operator on probability spaces. Recall that is a vector with components for cost and each metric , so this is a multi-objective problem.

The two-stage decision problem is a special case of the general problem outlined here: Each scenario can be considers as a composite of one or more stages.

## Experts

Each expert elicitation takes the form of an assessment of the probability and range (e.g., 10th to 90th percentile) of change in the cost or waste parameters or the production or metric functions. In essence, the expert elicitation defines for each potential scenario of each investment .

## Example: Idealized electrolysis of water

Here is a very simple model for electrolysis of water. We just have water, electricity, a catalyst, and some lab space. We choose the fundamental unit of operation to be moles of H2:

     H2O → H2 + ½ O2

Experts could assess how much R&D to increase the various efficiencies would cost. They could also suggest different catalysts, adding alkali, or replacing the process with PEM.

### Tracked quantities.

### Current design.

(due to mass transport loss on input)

(due to ohmic losses on input)

(due to mass transport loss on output)

(due to mass transport loss on output)

### Current costs.

(cost of Al-Ni catalyst)

(effective lifetime of Al-Ni catalyst)

(rough estimate for a 50W setup)

### Current prices.

### Production function (à la Leontief)

### Metric function.

### Performance of current design.

(i.e., not profitable since it is positive)

# Implementation

Database tables (one per set) hold all of the variables and the expert assessments. These tables are augmented by concise code with mathematical representations of the production and metric functions.

The Monte-Carlo computations are amenable to fast tensor-based implementation in Python.

See <<https://github.com/NREL/portfolio/tree/master/production-function/framework/code/tyche/>> for the tyche package that computes cost, production, and metrics from a technology design.

## Database tables

Each analysis case is represented by a Technology and a Scenario within that technology. In the specifications for the individual tables, we use the simple electrolysis example to populate the table.

### Metadata about indices

The indices table (see Table 3) simply describes the various indices available for the variables. The Offset column specifies the memory location in the argument for the production and metric functions.

Table 3: Example of the indices table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Technology | Type | Index | Offset | Description | Notes |
| Simple electrolysis | Capital | Catalyst | 0 | Catalyst |  |
| Simple electrolysis | Fixed | Rent | 0 | Rent |  |
| Simple electrolysis | Input | Water | 0 | Water |  |
| Simple electrolysis | Input | Electricity | 1 | Electricity |  |
| Simple electrolysis | Output | Oxygen | 0 | Oxygen |  |
| Simple electrolysis | Output | Hydrogen | 1 | Hydrogen |  |
| Simple electrolysis | Metric | Cost | 0 | Cost |  |
| Simple electrolysis | Metric | Jobs | 1 | Jobs |  |
| Simple electrolysis | Metric | GHG | 2 | GHGs |  |

### Design variables

The design table (see Table 4) specifies the values of all of the variables in the mathematical formulation of the design. Note that the Value column can either contain numeric literals or Python expressions specifying probability distribution functions. For example, a normal distribution with mean of five and standard deviation of two would be written st.norm(5, 2). All of the [Scipy probability distribution functions](https://docs.scipy.org/doc/scipy-1.4.1/reference/tutorial/stats/continuous.html#continuous-distributions-in-scipy-stats) are available for use, as are two special functions, constant and mixture. The constant distribution is just a single constant value; the mixture distribution is the mixture of a list of distributions, with specified relative weights. The mixture function is particularly important because it allows one to specify a first distribution in the case of an R&D breakthrough, but a second distribution if no breakthrough occurs.

Table 4: Example of the designs table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Variable | Index | Value | Units | Notes |
| Simple electrolysis | Base | Input | Water | 19.04 | g/mole |  |
| Simple electrolysis | Base | Input Efficiency | Water | 0.95 | 1 |  |
| Simple electrolysis | Base | Input | Electricity | 279 | kJ/mole |  |
| Simple electrolysis | Base | Input Efficiency | Electricity | 0.85 | 1 |  |
| Simple electrolysis | Base | Output Efficiency | Oxygen | 0.90 | 1 |  |
| Simple electrolysis | Base | Output Efficiency | Hydrogen | 0.90 | 1 |  |
| Simple electrolysis | Base | Lifetime | Catalyst | 3 | yr |  |
| Simple electrolysis | Base | Scale |  | 6650 | mole/yr |  |
| Simple electrolysis | Base | Input price | Water | 4.8e-3 | USD/mole |  |
| Simple electrolysis | Base | Input price | Electricity | 3.33e-5 | USD/kJ |  |
| Simple electrolysis | Base | Output price | Oxygen | 3.0e-3 | USD/g |  |
| Simple electrolysis | Base | Output price | Hydrogen | 1.0e-2 | USD/g |  |

### Metadata for functions

The functions table (see Table 5) simply documents which Python module and functions to use for the technology and scenario. Currently only the numpy style of function is supported, but later plain Python functions and tensorflow functions will be allowed.

Table 5: Example of the functions table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Technology | Style | Module | Capital | Fixed | Production | Metrics | Notes |
| Simple electrolysis | numpy | simple\_electrolysis | capital\_cost | fixed\_cost | production | metrics |  |

### Parameters for functions

The parameters table (see Table 6) contains ad-hoc parameters specific to the particular production and metrics functions. The Offset column specifies the memory location in the argument for the production and metric functions.

Table 6: Example of the parameters table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Parameter | Offset | Value | Units | Notes |
| Simple electrolysis | Base | Oxygen production | 0 | 16.00 | g |  |
| Simple electrolysis | Base | Hydrogen production | 1 | 2.00 | g |  |
| Simple electrolysis | Base | Water consumption | 2 | 18.08 | g |  |
| Simple electrolysis | Base | Electricity consumption | 3 | 237 | kJ |  |
| Simple electrolysis | Base | Jobs | 4 | 1.5e-4 | job/mole |  |
| Simple electrolysis | Base | Reference scale | 5 | 6650 | mole/yr |  |
| Simple electrolysis | Base | Reference capital cost for catalyst | 6 | 0.63 | USD |  |
| Simple electrolysis | Base | Reference fixed cost for rent | 7 | 1000 | USD/yr |  |
| Simple electrolysis | Base | GHG factor for water | 8 | 0.00108 | gCO2e/g | based on 244,956 gallons = 1 Mg CO2e |
| Simple electrolysis | Base | GHG factor for electricity | 9 | 0.138 | gCO2e/kJ | based on 1 kWh = 0.5 kg CO2e |

### Units for results

The results table (see Table 7) simply specifies the units for the results.

Table 7: Example of the results table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technology | Variable | Index | Units | Notes |
| Simple electrolysis | Cost | Cost | USD/mole |  |
| Simple electrolysis | Output | Oxygen | g/mole |  |
| Simple electrolysis | Output | Hydrogen | g/mole |  |
| Simple electrolysis | Metric | Cost | job/gH2 |  |
| Simple electrolysis | Metric | Jobs | job/gH2 |  |
| Simple electrolysis | Metric | GHG | gCO2e/gH2 |  |

### Tranches of investments.

In the tranches table (see Table 8), each *category* of investment contains a set of mutually exclusive *tranches* that may be associated with one or more *scenarios* defined in the designs table. Typically, a category is associated with a technology area and each tranche corresponds to an investment strategy within that category.

Table 8: Example of the tranches table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Tranche | Scenario | Amount | Notes |
| Electrolysis R&D | No Electrolysis R&D | Base Electrolysis | 0 |  |
| Electrolysis R&D | Low Electrolysis R&D | Slow Progress on Electrolysis | 1000000 |  |
| Electrolysis R&D | Medium Electrolysis R&D | Moderate Progress on Electrolysis | 2500000 |  |
| Electrolysis R&D | High Electrolysis R&D | Fast Progress on Electrolysis | 5000000 |  |

### Investments

In the investments table (see Table 9), each *investment* is associated with a single *tranche* in one or more *categories*. An investment typically combines tranches from several different investment categories.

Table 9: Example of the investments table.

|  |  |  |  |
| --- | --- | --- | --- |
| Investment | Category | Tranche | Notes |
| No R&D Spending | Electrolysis R&D | No Electrolysis R&D |  |
| Low R&D Spending | Electrolysis R&D | Low Electrolysis R&D |  |
| Medium R&D Spending | Electrolysis R&D | Medium Electrolysis R&D |  |
| High R&D Spending | Electrolysis R&D | High Electrolysis R&D |  |

## Python module and functions for a technology

Each technology design requires a Python module with a capital cost, a fixed cost, a production, and a metrics function. Listing 1 shows these functions for the simple electrolysis example.

Listing 1: Capital-cost, fixed-cost, production, and metrics functions for the simple electrolysis example.

# Simple electrolysis.  
  
  
# All of the computations must be vectorized, so use `numpy`.  
import numpy as np  
  
  
# Capital-cost function.  
def capital\_cost(scale, parameter):  
  
 # Scale the reference values.  
 return np.stack([np.multiply(parameter[6], np.divide(scale, parameter[5]))])  
  
  
# Fixed-cost function.  
def fixed\_cost(scale, parameter):  
  
 # Scale the reference values.  
 return np.stack([np.multiply(parameter[7], np.divide(scale, parameter[5]))])  
  
  
# Production function.  
def production(capital, fixed, input, parameter):  
  
 # Moles of input.  
 water = np.divide(input[0], parameter[2])  
 electricity = np.divide(input[1], parameter[3])  
  
 # Moles of output.  
 output = np.minimum(water, electricity)  
  
 # Grams of output.  
 oxygen = np.multiply(output, parameter[0])  
 hydrogen = np.multiply(output, parameter[1])  
  
 # Package results.  
 return np.stack([oxygen, hydrogen])  
  
  
# Metrics function.  
def metrics(capital, fixed, input\_raw, input, img/output\_raw, output, cost, parameter):  
  
 # Hydrogen output.  
 hydrogen = output[1]  
  
 # Cost of hydrogen.  
 cost1 = np.divide(cost, hydrogen)  
  
 # Jobs normalized to hydrogen.  
 jobs = np.divide(parameter[4], hydrogen)  
  
 # GHGs associated with water and electricity.  
 water = np.multiply(input\_raw[0], parameter[8])  
 electricity = np.multiply(input\_raw[1], parameter[9])  
 co2e = np.divide(np.add(water, electricity), hydrogen)  
  
 # Package results.  
 return np.stack([cost1, jobs, co2e])

## Python API for module tyche

The tyche module is a Python package for R&D pathways analysis and evaluation. It contains five Python classes for R&D pathway decision support.

* Designs for specifying and evaluating technology decisions in the presence of uncertainty.
* Investments for specifying and evaluating R&D portfolios consisting of multiple technology-investment options.
* Evaluator for rapidly evaluating the costs and benefits for sets of portfolios.
* EpsilonConstraints for multi-objective optimization using a generalization of the epsilon-constraint technique.
* DecisionGUI for interactively exploring the costs and benefits of R&D portfolios.

### DecisionWindow Objects

class DecisionWindow()

Class for displaying an interactive interface to explore cost-benefit tradeoffs for a technology.

#### \_\_init\_\_

| \_\_init\_\_(evaluator)

*Parameters*

evaluator : Evaluator

The evaluation object for the technology.

#### reevaluate

| reevaluate(next=lambda: None, delay=200)

Recalculate the results after a delay.

*Parameters*

next : function

The operation to perform after completing the recalculation.

delay : int

The number of milliseconds to delay before the recalculation.

#### reevaluate\_immediate

| reevaluate\_immediate(next=lambda: None)

Recalculate the results immediately.

*Parameters*

next : function

The operation to perform after completing the recalculation.

#### refresh

| refresh()

Refresh the graphics after a delay.

#### refresh\_immediate

| refresh\_immediate()

Refresh the graphics immediately.

#### mainloop

| mainloop()

Run the interactive interface.

### Designs Objects

class Designs()

Designs for a technology.

*Attributes*

indices : DataFrame

The *indices* table.

functions : DataFrame

The *functions* table.

designs : DataFrame

The *designs* table.

parameters : DataFrame

The *parameters* table.

results : DataFrame

The *results* table.

#### \_\_init\_\_

| \_\_init\_\_(path=None, indices="indices.tsv", functions="functions.tsv", designs="designs.tsv", parameters="parameters.tsv", results="results.tsv")

*Parameters*

path : str

Location of the data files.

indices : str

Filename for the *indices* table.

functions : str

Filename for the *functions* table.

designs : str

Filename for the *designs* table.

parameters : str

Filename for the *parameters* table.

results : str

Filename for the *results* table.

#### vectorize\_technologies

| vectorize\_technologies()

Make an array of technologies.

#### vectorize\_scenarios

| vectorize\_scenarios(technology)

Make an array of scenarios.

#### vectorize\_indices

| vectorize\_indices(technology)

Make an array of indices.

#### vectorize\_designs

| vectorize\_designs(technology, scenario\_count, sample\_count=1)

Make an array of designs.

#### vectorize\_parameters

| vectorize\_parameters(technology, scenario\_count, sample\_count=1)

Make an array of parameters.

#### compile

| compile()

Compile the production and metrics functions.

#### evaluate

| evaluate(technology, sample\_count=1)

Evaluate the performance of a technology.

*Parameters*

technology : str

The name of the technology.

sample\_count : int

The number of random samples.

#### evaluate\_scenarios

| evaluate\_scenarios(sample\_count=1)

Evaluate scenarios.

*Parameters*

sample\_count : int

The number of random samples.

### EpsilonConstraintOptimizer Objects

class EpsilonConstraintOptimizer()

An epsilon-constration multi-objective optimizer.

*Attributes*

evaluator : tyche.Evaluator

The technology evaluator.

scale : float

The scaling factor for output.

#### \_\_init\_\_

| \_\_init\_\_(evaluator, scale=1e6)

*Parameters*

evaluator : tyche.Evaluator

The technology evaluator.

scale : float

The scaling factor for output.

#### maximize

| maximize(metric, max\_amount=None, total\_amount=None, min\_metric=None, statistic=np.mean, initial=None, tol=1e-8, maxiter=50, verbose=0)

Maximize the objective function.

*Parameters*

metric : str

The metric to maximize.

max\_amount : DataFrame

The maximum amounts that can be invested in each category.

total\_amount : float

The maximum amount that can be invested *in toto*.

min\_metric : DataFrame

The minimum constraint for each metric.

statistic : function

The statistic used on the sample evaluations.

initial : array of float

The initial value for the search.

tol : float

The search tolerance.

maxiter : int

The maximum iterations for the search.

verbose : int

Verbosity level.

#### max\_metrics

| max\_metrics(max\_amount=None, total\_amount=None, statistic=np.mean, tol=1e-8, maxiter=50, verbose=0)

Maximum value of metrics.

*Parameters*

max\_amount : DataFrame

The maximum amounts that can be invested in each category.

total\_amount : float

The maximum amount that can be invested *in toto*.

min\_metric : DataFrame

The minimum constraint for each metric.

statistic : function

The statistic used on the sample evaluations.

initial : array of float

The initial value for the search.

tol : float

The search tolerance.

maxiter : int

The maximum iterations for the search.

verbose : int

Verbosity level.

### Evaluator Objects

class Evaluator()

Evalutate technology investments using a response surface.

*Attributes*

amounts : DataFrame

Cost of tranches.

categories : DataFrame

Categories of investment.

metrics : DataFrame

Metrics for technologies.

units : DataFrame

Units of measure for metrics.

interpolators : DataFrame

Interpolation functions for technology metrics.

#### \_\_init\_\_

| \_\_init\_\_(tranches, summary)

*Parameters*

tranches : DataFrame

The tranches of investment.

summary : DataFrame

The summary of evaluating the tranches.

#### evaluate

| evaluate(amounts)

Sample the distribution for an investment.

*Parameters*

amounts : DataFrame

The investment levels.

#### evaluate\_statistic

| evaluate\_statistic(amounts, statistic=np.mean)

Evaluate a statistic for an investment.

*Parameters*

amounts : DataFrame

The investment levels.

statistics : DataFrame

The statistic to evaluate.

### Investments Objects

class Investments()

Investments in a technology.

*Attributes*

tranches : DataFrame

The *tranches* table.

investments: DataFrame

The *investments* table.

#### \_\_init\_\_

| \_\_init\_\_(path=None, tranches="tranches.tsv", investments="investments.tsv")

*Parameters*

tranches : str

Filename for the *tranches* table.

investments: str

Filename for the *investments* table.

#### evaluate\_tranches

| evaluate\_tranches(designs, sample\_count=1)

Evaluate the tranches of investment for a design.

*Parameters*

designs : tyche.Designs

The designs.

sample\_count : int

The number of random samples.

#### evaluate\_investments

| evaluate\_investments(designs, sample\_count=1)

Evaluate the investments for a design.

*Parameters*

designs : tyche.Designs

The designs.

sample\_count : int

The number of random samples.

# Extended example

## Set up.

Import packages.

import numpy as np  
import matplotlib.pyplot as pl  
import pandas as pd  
import seaborn as sb  
import tyche as ty  
  
from copy import deepcopy  
from IPython.display import Image

Load data.

The data are stored in a set of tab-separated value files in a folder.

designs = ty.Designs("data/residential\_pv\_multiobjective")  
investments = ty.Investments("data/residential\_pv\_multiobjective")

Compile the production and metric functions for each technology in the dataset.

designs.compile()

## Examine the data.

The functions table specifies where the Python code for each technology resides.

designs.functions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Technology | Style | Module | Capital | Fixed | Production | Metrics | Notes |
| Residential PV | numpy | residential\_pv\_multiobjective | capital\_cost | fixed\_cost | production | metrics |  |

The indices table defines the subscripts for variables.

designs.indices

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Technology | Type | Index | Offset | Description | Notes |
| Residential PV | Capital | BoS | 2 | balance of system |  |
| Residential PV | Capital | Inverter | 1 | system inverters |  |
| Residential PV | Capital | Module | 0 | system module |  |
| Residential PV | Fixed | System | 0 | whole system |  |
| Residential PV | Input | nan | 0 | no inputs |  |
| Residential PV | Metric | GHG | 2 | reduction in GHGs |  |
| Residential PV | Metric | LCOE | 0 | reduction in levelized cost of energy |  |
| Residential PV | Metric | Labor | 1 | increase in spending on wages |  |
| Residential PV | Output | Electricity | 0 | electricity generated |  |

The designs table contains the cost, input, efficiency, and price data for a scenario.

designs.designs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Variable | Index | Value | Units | Notes |
| Residential PV | 2015 Actual | Input | nan | 0 | 1 | no inputs |
| Residential PV | 2015 Actual | Input efficiency | nan | 1 | 1 | no inputs |
| Residential PV | 2015 Actual | Input price | nan | 0 | 1 | no inputs |
| Residential PV | 2015 Actual | Lifetime | BoS | 1 | system-lifetime | per-lifetime computations |
| Residential PV | 2015 Actual | Lifetime | Inverter | 1 | system-lifetime | per-lifetime computations |
| Residential PV | 2015 Actual | Lifetime | Module | 1 | system-lifetime | per-lifetime computations |
| Residential PV | 2015 Actual | Output efficiency | Electricity | 1 | W/W | see parameter table for individual efficiencies |
| Residential PV | 2015 Actual | Output price | Electricity | 0 | $/kWh | not tracking electricity price |
| Residential PV | 2015 Actual | Scale | nan | 1 | system/system | no scaling |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| Residential PV | Module Slow Progress | Input | nan | 0 | 1 | no inputs |
| Residential PV | Module Slow Progress | Input efficiency | nan | 1 | 1 | no inputs |
| Residential PV | Module Slow Progress | Input price | nan | 0 | 1 | no inputs |
| Residential PV | Module Slow Progress | Lifetime | BoS | 1 | system-lifetime | per-lifetime computations |
| Residential PV | Module Slow Progress | Lifetime | Inverter | 1 | system-lifetime | per-lifetime computations |
| Residential PV | Module Slow Progress | Lifetime | Module | 1 | system-lifetime | per-lifetime computations |
| Residential PV | Module Slow Progress | Output efficiency | Electricity | 1 | W/W | see parameter table for individual efficiencies |
| Residential PV | Module Slow Progress | Output price | Electricity | 0 | $/kWh | not tracking electricity price |
| Residential PV | Module Slow Progress | Scale | nan | 1 | system/system | no scaling |

The parameters table contains additional techno-economic parameters for each technology.

designs.parameters

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Parameter | Offset | Value | Units | Notes |
| Residential PV | 2015 Actual | Customer Acquisition | 19 | st.triang(0.5, loc=2000, scale=0.2) | $/system | BCA |
| Residential PV | 2015 Actual | DC-to-AC Ratio | 15 | st.triang(0.5, loc=1.4, scale=0.00014) | 1 | IDC |
| Residential PV | 2015 Actual | Direct Labor | 17 | st.triang(0.5, loc=2000, scale=0.2) | $/system | BLR |
| Residential PV | 2015 Actual | Discount Rate | 0 | 0.07 | 1/year | DR |
| Residential PV | 2015 Actual | Hardware Capital | 16 | st.triang(0.5, loc=80, scale=0.008) | $/m^2 | BCC |
| Residential PV | 2015 Actual | Insolation | 1 | 1000 | W/m^2 | INS |
| Residential PV | 2015 Actual | Installer Overhead & Profit | 20 | st.triang(0.5, loc=0.35, scale=3.5e-5) | 1 | BOH |
| Residential PV | 2015 Actual | Inverter Capital | 11 | st.triang(0.5, loc=0.3, scale=3e-5) | $/W | ICC |
| Residential PV | 2015 Actual | Inverter Efficiency | 14 | st.triang(0.5, loc=0.9, scale=9e-5) | 1 | IEF |
| Residential PV | 2015 Actual | Inverter Lifetime | 12 | st.triang(0.5, loc=16, scale=0.0016) | yr | ILT |
| Residential PV | 2015 Actual | Inverter Replacement | 13 | st.triang(0.5, loc=0.5, scale=5e-5) | 1 | IRC |
| Residential PV | 2015 Actual | Location Capacity Factor | 9 | st.triang(0.5, loc=0.2, scale=2e-5) | 1 | MCF |
| Residential PV | 2015 Actual | Module Aperture | 6 | st.triang(0.5, loc=0.9, scale=9e-5) | 1 | MAP |
| Residential PV | 2015 Actual | Module Capital | 3 | st.triang(0.5, loc=110, scale=0.11) | $/m^2 | MCC |
| Residential PV | 2015 Actual | Module Degradation | 8 | st.triang(0.5, loc=0.0075, scale=7.5e-7) | 1/yr | MDR |
| Residential PV | 2015 Actual | Module Efficiency | 5 | st.triang(0.5, loc=0.16, scale=1.6e-5) | 1 | MEF |
| Residential PV | 2015 Actual | Module Lifetime | 4 | st.triang(0.5, loc=25, scale=0.0025) | yr | MLT |
| Residential PV | 2015 Actual | Module O&M Fixed | 7 | st.triang(0.5, loc=20, scale=0.002) | $/kWyr | MOM |
| Residential PV | 2015 Actual | Module Soiling Loss | 10 | st.triang(0.5, loc=0.05, scale=5e-6) | 1 | MSL |
| Residential PV | 2015 Actual | Permitting | 18 | st.triang(0.5, loc=600, scale=0.06) | $/system | BPR |
| Residential PV | 2015 Actual | System Size | 2 | 36 | m^2 | SSZ |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . |

The results table specifies the units of measure for results of computations.

designs.results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Technology | Variable | Index | Units | Notes |
| Residential PV | Cost | Cost | $/system |  |
| Residential PV | Metric | GHG | ΔgCO2e/system |  |
| Residential PV | Metric | LCOE | Δ$/kWh |  |
| Residential PV | Metric | Labor | Δ$/system |  |
| Residential PV | Output | Electricity | kWh |  |

The tranches table specifies multually exclusive possibilities for investments: only one Tranch may be selected for each Category.

investments.tranches

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Tranche | Scenario | Amount | Notes |
| BoS R&D | BoS High R&D | BoS Fast Progress | 900000 |  |
| BoS R&D | BoS Low R&D | BoS Slow Progress | 300000 |  |
| BoS R&D | BoS Medium R&D | BoS Moderate Progress | 600000 |  |
| Inverter R&D | Inverter High R&D | Inverter Fast Progress | 3e+06 |  |
| Inverter R&D | Inverter Low R&D | Inverter Slow Progress | 1e+06 |  |
| Inverter R&D | Inverter Medium R&D | Inverter Moderate Progress | 2e+06 |  |
| Module R&D | Module High R&D | Module Fast Progress | 4.5e+06 |  |
| Module R&D | Module Low R&D | Module Slow Progress | 1.5e+06 |  |
| Module R&D | Module Medium R&D | Module Moderate Progress | 3e+06 |  |

The investments table bundles a consistent set of tranches (one per category) into an overall investment.

investments.investments

|  |  |  |  |
| --- | --- | --- | --- |
| Investment | Category | Tranche | Notes |
| High R&D | BoS R&D | BoS High R&D |  |
| High R&D | Inverter R&D | Inverter High R&D |  |
| High R&D | Module R&D | Module High R&D |  |
| Low R&D | BoS R&D | BoS Low R&D |  |
| Low R&D | Inverter R&D | Inverter Low R&D |  |
| Low R&D | Module R&D | Module Low R&D |  |
| Medium R&D | BoS R&D | BoS Medium R&D |  |
| Medium R&D | Inverter R&D | Inverter Medium R&D |  |
| Medium R&D | Module R&D | Module Medium R&D |  |

## Evaluate the scenarios in the dataset.

scenario\_results = designs.evaluate\_scenarios(sample\_count=50)  
scenario\_results.xs(1, level="Sample", drop\_level=False)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technology | Scenario | Sample | Variable | Index | Value | Units |
| Residential PV | 2015 Actual | 1 | Cost | Cost | 19541.3 | $/system |
| Residential PV | 2015 Actual | 1 | Metric | GHG | -0.00371397 | ΔgCO2e/system |
| Residential PV | 2015 Actual | 1 | Metric | LCOE | -1.85062e-05 | Δ$/kWh |
| Residential PV | 2015 Actual | 1 | Metric | Labor | -0.00191143 | Δ$/system |
| Residential PV | 2015 Actual | 1 | Output | Electricity | 184103 | kWh |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| Residential PV | Module Slow Progress | 1 | Cost | Cost | 19680.6 | $/system |
| Residential PV | Module Slow Progress | 1 | Metric | GHG | 15.2892 | ΔgCO2e/system |
| Residential PV | Module Slow Progress | 1 | Metric | LCOE | 0.0159126 | Δ$/kWh |
| Residential PV | Module Slow Progress | 1 | Metric | Labor | 0.00873305 | Δ$/system |
| Residential PV | Module Slow Progress | 1 | Output | Electricity | 218158 | kWh |

Plot GHG metric.

g = sb.boxplot(  
 x="Scenario",  
 y="Value",  
 data=scenario\_results.xs(  
 ["Metric", "GHG"],  
 level=["Variable", "Index"]  
 ).reset\_index()[["Scenario", "Value"]],  
 order=[  
 "2015 Actual" ,  
 "Module Slow Progress" ,  
 "Module Moderate Progress" ,  
 "Module Fast Progress" ,  
 "Inverter Slow Progress" ,  
 "Inverter Moderate Progress",  
 "Inverter Fast Progress" ,  
 "BoS Slow Progress" ,  
 "BoS Moderate Progress" ,  
 "BoS Fast Progress" ,  
 ]  
)  
g.set(ylabel="GHG Reduction [gCO2e / system]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=90);

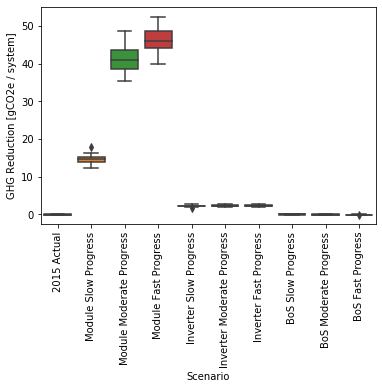


Figure 1: GHG metrics for scenarios.

Plot LCOE metric.

g = sb.boxplot(  
 x="Scenario",  
 y="Value",  
 data=scenario\_results.xs(  
 ["Metric", "LCOE"],  
 level=["Variable", "Index"]  
 ).reset\_index()[["Scenario", "Value"]],  
 order=[  
 "2015 Actual" ,  
 "Module Slow Progress" ,  
 "Module Moderate Progress" ,  
 "Module Fast Progress" ,  
 "Inverter Slow Progress" ,  
 "Inverter Moderate Progress",  
 "Inverter Fast Progress" ,  
 "BoS Slow Progress" ,  
 "BoS Moderate Progress" ,  
 "BoS Fast Progress" ,  
 ]  
)  
g.set(ylabel="LCOE Reduction [USD / kWh]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=90);

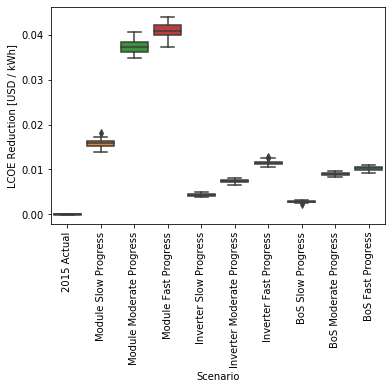


Figure 2: LCOE metrics for scenarios.

Plot labor metric.

g = sb.boxplot(  
 x="Scenario",  
 y="Value",  
 data=scenario\_results.xs(  
 ["Metric", "Labor"],  
 level=["Variable", "Index"]  
 ).reset\_index()[["Scenario", "Value"]],  
 order=[  
 "2015 Actual" ,  
 "Module Slow Progress" ,  
 "Module Moderate Progress" ,  
 "Module Fast Progress" ,  
 "Inverter Slow Progress" ,  
 "Inverter Moderate Progress",  
 "Inverter Fast Progress" ,  
 "BoS Slow Progress" ,  
 "BoS Moderate Progress" ,  
 "BoS Fast Progress" ,  
 ]  
)  
g.set(ylabel="Labor Increase [USD / system]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=90);

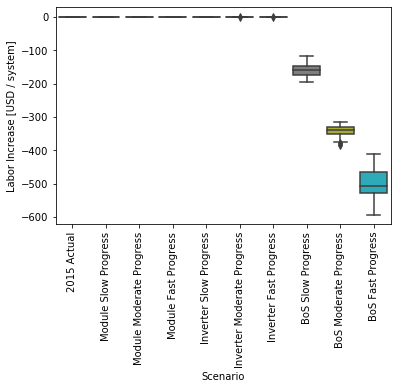


Figure 3: Labor metrics for scenarios.

## Evaluate the investments in the dataset.

investment\_results = investments.evaluate\_investments(designs, sample\_count=50)  
investment\_results.amounts

|  |  |
| --- | --- |
| Investment | Amount |
| High R&D | 8.4e+06 |
| Low R&D | 2.8e+06 |
| Medium R&D | 5.6e+06 |

Benefits of investments.

investment\_results.metrics.xs(1, level="Sample", drop\_level=False)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Investment | Category | Tranche | Scenario | Sample | Technology | Index | Value | Units |
| High R&D | BoS R&D | BoS High R&D | BoS Fast Progress | 1 | Residential PV | GHG | -0.00612191 | ΔgCO2e/system |
| High R&D | BoS R&D | BoS High R&D | BoS Fast Progress | 1 | Residential PV | LCOE | 0.0099879 | Δ$/kWh |
| High R&D | BoS R&D | BoS High R&D | BoS Fast Progress | 1 | Residential PV | Labor | -473.404 | Δ$/system |
| Medium R&D | BoS R&D | BoS Medium R&D | BoS Moderate Progress | 1 | Residential PV | GHG | -0.00672276 | ΔgCO2e/system |
| Medium R&D | BoS R&D | BoS Medium R&D | BoS Moderate Progress | 1 | Residential PV | LCOE | 0.0091519 | Δ$/kWh |
| Medium R&D | BoS R&D | BoS Medium R&D | BoS Moderate Progress | 1 | Residential PV | Labor | -342.64 | Δ$/system |
| Low R&D | BoS R&D | BoS Low R&D | BoS Slow Progress | 1 | Residential PV | GHG | -0.00246448 | ΔgCO2e/system |
| Low R&D | BoS R&D | BoS Low R&D | BoS Slow Progress | 1 | Residential PV | LCOE | 0.00257484 | Δ$/kWh |
| Low R&D | BoS R&D | BoS Low R&D | BoS Slow Progress | 1 | Residential PV | Labor | -125.705 | Δ$/system |
| High R&D | Inverter R&D | Inverter High R&D | Inverter Fast Progress | 1 | Residential PV | GHG | 1.96063 | ΔgCO2e/system |
| High R&D | Inverter R&D | Inverter High R&D | Inverter Fast Progress | 1 | Residential PV | LCOE | 0.0117868 | Δ$/kWh |
| High R&D | Inverter R&D | Inverter High R&D | Inverter Fast Progress | 1 | Residential PV | Labor | 0.0208266 | Δ$/system |
| Medium R&D | Inverter R&D | Inverter Medium R&D | Inverter Moderate Progress | 1 | Residential PV | GHG | 1.95647 | ΔgCO2e/system |
| Medium R&D | Inverter R&D | Inverter Medium R&D | Inverter Moderate Progress | 1 | Residential PV | LCOE | 0.00749456 | Δ$/kWh |
| Medium R&D | Inverter R&D | Inverter Medium R&D | Inverter Moderate Progress | 1 | Residential PV | Labor | 0.0321275 | Δ$/system |
| Low R&D | Inverter R&D | Inverter Low R&D | Inverter Slow Progress | 1 | Residential PV | GHG | 2.28385 | ΔgCO2e/system |
| Low R&D | Inverter R&D | Inverter Low R&D | Inverter Slow Progress | 1 | Residential PV | LCOE | 0.00425453 | Δ$/kWh |
| Low R&D | Inverter R&D | Inverter Low R&D | Inverter Slow Progress | 1 | Residential PV | Labor | -0.0462818 | Δ$/system |
| High R&D | Module R&D | Module High R&D | Module Fast Progress | 1 | Residential PV | GHG | 46.1992 | ΔgCO2e/system |
| High R&D | Module R&D | Module High R&D | Module Fast Progress | 1 | Residential PV | LCOE | 0.040478 | Δ$/kWh |
| High R&D | Module R&D | Module High R&D | Module Fast Progress | 1 | Residential PV | Labor | -0.0398534 | Δ$/system |
| Medium R&D | Module R&D | Module Medium R&D | Module Moderate Progress | 1 | Residential PV | GHG | 47.5765 | ΔgCO2e/system |
| Medium R&D | Module R&D | Module Medium R&D | Module Moderate Progress | 1 | Residential PV | LCOE | 0.0399647 | Δ$/kWh |
| Medium R&D | Module R&D | Module Medium R&D | Module Moderate Progress | 1 | Residential PV | Labor | 0.024476 | Δ$/system |
| Low R&D | Module R&D | Module Low R&D | Module Slow Progress | 1 | Residential PV | GHG | 12.9021 | ΔgCO2e/system |
| Low R&D | Module R&D | Module Low R&D | Module Slow Progress | 1 | Residential PV | LCOE | 0.0144974 | Δ$/kWh |
| Low R&D | Module R&D | Module Low R&D | Module Slow Progress | 1 | Residential PV | Labor | 0.0317123 | Δ$/system |

investment\_results.summary.xs(1, level="Sample", drop\_level=False)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Investment | Sample | Index | Value | Units |
| High R&D | 1 | GHG | 48.1537 | ΔgCO2e/system |
| High R&D | 1 | LCOE | 0.0622528 | Δ$/kWh |
| High R&D | 1 | Labor | -473.423 | Δ$/system |
| Medium R&D | 1 | GHG | 49.5263 | ΔgCO2e/system |
| Medium R&D | 1 | LCOE | 0.0566111 | Δ$/kWh |
| Medium R&D | 1 | Labor | -342.583 | Δ$/system |
| Low R&D | 1 | GHG | 15.1835 | ΔgCO2e/system |
| Low R&D | 1 | LCOE | 0.0213268 | Δ$/kWh |
| Low R&D | 1 | Labor | -125.719 | Δ$/system |

Plot GHG metric.

g = sb.boxplot(  
 x="Investment",  
 y="Value",  
 data=investment\_results.metrics.xs(  
 "GHG",  
 level="Index"  
 ).reset\_index()[["Investment", "Value"]],  
 order=[  
 "Low R&D" ,  
 "Medium R&D",  
 "High R&D" ,  
 ]  
)  
g.set(ylabel="GHG Reduction [gCO2e / system]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=15);

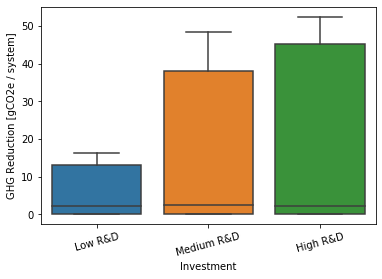


Figure 4: GHG metrics for investments.

Plot LCOE metric.

g = sb.boxplot(  
 x="Investment",  
 y="Value",  
 data=investment\_results.metrics.xs(  
 "LCOE",  
 level="Index"  
 ).reset\_index()[["Investment", "Value"]],  
 order=[  
 "Low R&D" ,  
 "Medium R&D",  
 "High R&D" ,  
 ]  
)  
g.set(ylabel="LCOE Reduction [USD / kWh]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=15);

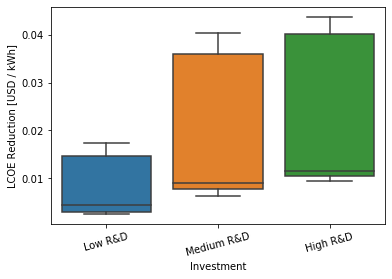


Figure 5: LCOE metrics for investments.

Plot labor metric.

g = sb.boxplot(  
 x="Investment",  
 y="Value",  
 data=investment\_results.metrics.xs(  
 "Labor",  
 level="Index"  
 ).reset\_index()[["Investment", "Value"]],  
 order=[  
 "Low R&D" ,  
 "Medium R&D",  
 "High R&D" ,  
 ]  
)  
g.set(ylabel="Labor Increase [USD / system]")  
g.set\_xticklabels(g.get\_xticklabels(), rotation=15);

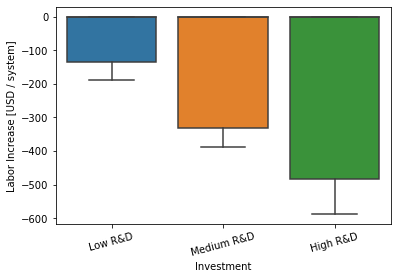


Figure 6: Labor metrics for investment

## Multi-objective decision analysis.

Compute costs and metrics for tranches. Tranches are atomic units for building investment portfolios. Evaluate all of the tranches, so we can assemble them into investments (portfolios).

tranche\_results = investments.evaluate\_tranches(designs, sample\_count=50)  
tranche\_results.amounts

|  |  |  |
| --- | --- | --- |
| Category | Tranche | Amount |
| BoS R&D | BoS High R&D | 900000 |
| BoS R&D | BoS Low R&D | 300000 |
| BoS R&D | BoS Medium R&D | 600000 |
| Inverter R&D | Inverter High R&D | 3e+06 |
| Inverter R&D | Inverter Low R&D | 1e+06 |
| Inverter R&D | Inverter Medium R&D | 2e+06 |
| Module R&D | Module High R&D | 4.5e+06 |
| Module R&D | Module Low R&D | 1.5e+06 |
| Module R&D | Module Medium R&D | 3e+06 |

Display the metrics for each tranche.

summary.xs(1, level="Sample", drop\_level=False)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Tranche | Sample | Index | Value | Units |
| BoS R&D | BoS High R&D | 1 | GHG | -0.00509628 | ΔgCO2e/system |
| BoS R&D | BoS High R&D | 1 | LCOE | 0.0101198 | Δ$/kWh |
| BoS R&D | BoS High R&D | 1 | Labor | -512.124 | Δ$/system |
| BoS R&D | BoS Medium R&D | 1 | GHG | -0.00347462 | ΔgCO2e/system |
| BoS R&D | BoS Medium R&D | 1 | LCOE | 0.00871829 | Δ$/kWh |
| BoS R&D | BoS Medium R&D | 1 | Labor | -326.32 | Δ$/system |
| BoS R&D | BoS Low R&D | 1 | GHG | -0.00207459 | ΔgCO2e/system |
| BoS R&D | BoS Low R&D | 1 | LCOE | 0.00236172 | Δ$/kWh |
| BoS R&D | BoS Low R&D | 1 | Labor | -106.82 | Δ$/system |
| Inverter R&D | Inverter High R&D | 1 | GHG | 2.23156 | ΔgCO2e/system |
| Inverter R&D | Inverter High R&D | 1 | LCOE | 0.0114439 | Δ$/kWh |
| Inverter R&D | Inverter High R&D | 1 | Labor | -0.0363715 | Δ$/system |
| Inverter R&D | Inverter Medium R&D | 1 | GHG | 2.32106 | ΔgCO2e/system |
| Inverter R&D | Inverter Medium R&D | 1 | LCOE | 0.00755465 | Δ$/kWh |
| Inverter R&D | Inverter Medium R&D | 1 | Labor | 0.0521195 | Δ$/system |
| Inverter R&D | Inverter Low R&D | 1 | GHG | 2.40101 | ΔgCO2e/system |
| Inverter R&D | Inverter Low R&D | 1 | LCOE | 0.00445806 | Δ$/kWh |
| Inverter R&D | Inverter Low R&D | 1 | Labor | 0.0482813 | Δ$/system |
| Module R&D | Module High R&D | 1 | GHG | 46.2551 | ΔgCO2e/system |
| Module R&D | Module High R&D | 1 | LCOE | 0.0410125 | Δ$/kWh |
| Module R&D | Module High R&D | 1 | Labor | -0.016474 | Δ$/system |
| Module R&D | Module Medium R&D | 1 | GHG | 44.0851 | ΔgCO2e/system |
| Module R&D | Module Medium R&D | 1 | LCOE | 0.0388401 | Δ$/kWh |
| Module R&D | Module Medium R&D | 1 | Labor | 0.0607401 | Δ$/system |
| Module R&D | Module Low R&D | 1 | GHG | 15.8667 | ΔgCO2e/system |
| Module R&D | Module Low R&D | 1 | LCOE | 0.017083 | Δ$/kWh |
| Module R&D | Module Low R&D | 1 | Labor | 0.00647508 | Δ$/system |

## Response surface.

Fit a response surface to the results. The response surface interpolates between the discrete set of cases provided in the expert elicitation. This allows us to study funding levels intermediate between those scenarios.

evaluator = ty.Evaluator(investments.tranches, tranche\_results.summary)

Here are the categories of investment and the maximum amount that could be invested in each:

evaluator.max\_amount

|  |  |
| --- | --- |
| Category | Amount |
| BoS R&D | 900000 |
| Inverter R&D | 3e+06 |
| Module R&D | 4.5e+06 |

Here are the metrics and their units of measure:

evaluator.units

|  |  |
| --- | --- |
| Index | Units |
| GHG | ΔgCO2e/system |
| LCOE | Δ$/kWh |
| Labor | Δ$/system |

### Example interpolation.

Let’s evaluate the case where each category is invested in at half of its maximum amount.

example\_investments = evaluator.max\_amount / 2  
example\_investments

|  |  |
| --- | --- |
| Category | Amount |
| BoS R&D | 450000 |
| Inverter R&D | 1.5e+06 |
| Module R&D | 2.25e+06 |

evaluator.evaluate(example\_investments).xs(1, level="Sample", drop\_level=False)

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Index | Sample | Value |
| BoS R&D | GHG | 1 | -0.00277461 |
| BoS R&D | LCOE | 1 | 0.00554001 |
| BoS R&D | Labor | 1 | -216.57 |
| Inverter R&D | GHG | 1 | 2.36104 |
| Inverter R&D | LCOE | 1 | 0.00600635 |
| Inverter R&D | Labor | 1 | 0.0502004 |
| Module R&D | GHG | 1 | 29.9759 |
| Module R&D | LCOE | 1 | 0.0279616 |
| Module R&D | Labor | 1 | 0.0336076 |

Let’s evaluate the mean instead of outputing the whole distribution.

evaluator.evaluate\_statistic(example\_investments, np.mean)

Index  
GHG 30.229573  
LCOE 0.038244  
Labor -248.531825  
Name: Value, dtype: float64

Here is the standard deviation:

evaluator.evaluate\_statistic(example\_investments, np.std)

Index  
GHG 1.626180  
LCOE 0.000927  
Labor 12.830230  
Name: Value, dtype: float64

A risk-averse decision maker might be interested in the 10% percentile:

evaluator.evaluate\_statistic(example\_investments, lambda x: np.quantile(x, 0.1))

Index  
GHG 28.254546  
LCOE 0.037052  
Labor -259.124007  
Name: Value, dtype: float64

## ε-Constraint multiobjective optimization

optimizer = ty.EpsilonConstraintOptimizer(evaluator)

In order to meaningfully map the decision space, we need to know the maximum values for each of the metrics.

metric\_max = optimizer.max\_metrics()  
metric\_max

GHG 49.671071  
LCOE 0.062720  
Labor 0.045590  
Name: Value, dtype: float64

### Example optimization.

Limit spending to $3M.

investment\_max = 3e6

Require that the GHG reduction be at least 40 gCO2e/system and that the Labor wages not decrease.

metric\_min = pd.Series([40, 0], name = "Value", index = ["GHG", "Labor"])  
metric\_min

GHG 40  
Labor 0  
Name: Value, dtype: int64

Compute the ε-constrained maximum for the LCOE.

optimum = optimizer.maximize(  
 "LCOE" ,  
 total\_amount = investment\_max,  
 min\_metric = metric\_min ,  
 statistic = np.mean ,  
)  
optimum.exit\_message

'Optimization terminated successfully.'

Here are the optimal spending levels:

np.round(optimum.amounts)

Category  
BoS R&D 0.0  
Inverter R&D 0.0  
Module R&D 3000000.0  
Name: Amount, dtype: float64

Here are the three metrics at that optimum:

optimum.metrics

Index  
GHG 42.079306  
LCOE 0.037732  
Labor 0.023559  
Name: Value, dtype: float64

*Thus, by putting all of the investment into Module R&D, we can expected to achieve a mean 3.75 ¢/kWh reduction in LCOE under the GHG and Labor constraints.*

It turns out that there is no solution for these constraints if we evaluate the 10th percentile of the metrics, for a risk-averse decision maker.

optimum = optimizer.maximize(  
 "LCOE" ,  
 total\_amount = investment\_max,  
 min\_metric = metric\_min ,  
 statistic = lambda x: np.quantile(x, 0.1),  
)  
optimum.exit\_message

'Positive directional derivative for linesearch'

Let’s try again, but with a less stringent set of constraints, only constraining GHG somewhat but not Labor at all.

optimum = optimizer.maximize(  
 "LCOE" ,  
 total\_amount = investment\_max ,  
 min\_metric = pd.Series([30], name = "Value", index = ["GHG"]),  
 statistic = lambda x: np.quantile(x, 0.1) ,  
)  
optimum.exit\_message

'Optimization terminated successfully.'

np.round(optimum.amounts)

Category  
BoS R&D 0.0  
Inverter R&D 0.0  
Module R&D 3000000.0  
Name: Amount, dtype: float64

optimum.metrics

Index  
GHG 38.525518  
LCOE 0.036185  
Labor -0.022495  
Name: Value, dtype: float64

### Pareto surfaces.

#### Metrics constrained by total investment.

pareto\_amounts = None  
for investment\_max in np.arange(1e6, 9e6, 0.5e6):  
 metrics = optimizer.max\_metrics(total\_amount = investment\_max)  
 pareto\_amounts = pd.DataFrame(  
 [metrics.values] ,  
 columns = metrics.index.values ,  
 index = pd.Index([investment\_max / 1e6], name = "Investment [M$]"),  
 ).append(pareto\_amounts)  
sb.relplot(  
 x = "Investment [M$]",  
 y = "Value" ,  
 col = "Metric" ,  
 kind = "line" ,  
 facet\_kws = {'sharey': False},  
 data = pareto\_amounts.reset\_index().melt(id\_vars = "Investment [M$]", var\_name = "Metric", value\_name = "Value")  
)

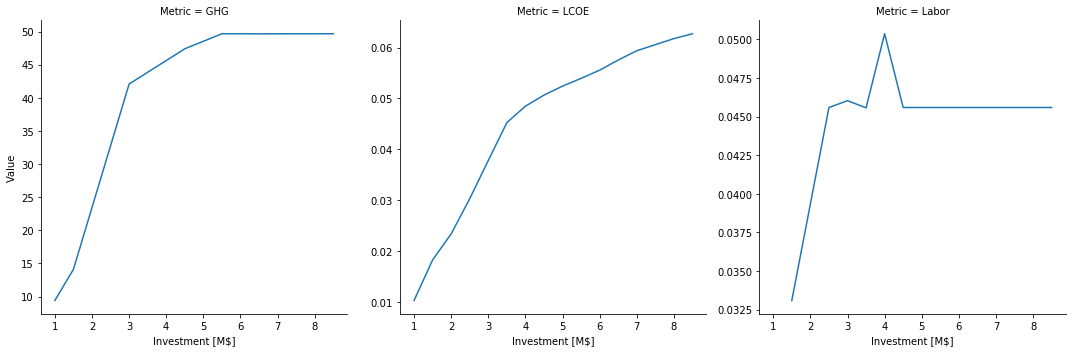


Figure 7: Pareto surface.

*We see that the LCOE metric saturates more slowly than the GHG and Labor ones.*

#### GHG vs LCOE, constrained by total investment.

investment\_max = 3  
pareto\_ghg\_lcoe = None  
for lcoe\_min in 0.95 \* np.arange(0.5, 0.9, 0.05) \* pareto\_amounts.loc[investment\_max, "LCOE"]:  
 optimum = optimizer.maximize(  
 "GHG",  
 max\_amount = pd.Series([0.9e6, 3.0e6, 1.0e6], name = "Amount", index = ["BoS R&D", "Inverter R&D", "Module R&D"]),  
 total\_amount = investment\_max \* 1e6 ,  
 min\_metric = pd.Series([lcoe\_min], name = "Value", index = ["LCOE"]),  
 )  
 pareto\_ghg\_lcoe = pd.DataFrame(  
 [[investment\_max, lcoe\_min, optimum.metrics["LCOE"], optimum.metrics["GHG"], optimum.exit\_message]],  
 columns = ["Investment [M$]", "LCOE (min)", "LCOE", "GHG", "Result"] ,  
 ).append(pareto\_ghg\_lcoe)  
pareto\_ghg\_lcoe = pareto\_ghg\_lcoe.set\_index(["Investment [M$]", "LCOE (min)"])  
sb.relplot(  
 x = "LCOE",  
 y = "GHG",  
 kind = "scatter",  
 data = pareto\_ghg\_lcoe#[pareto\_ghg\_lcoe.Result == "Optimization terminated successfully."]  
)

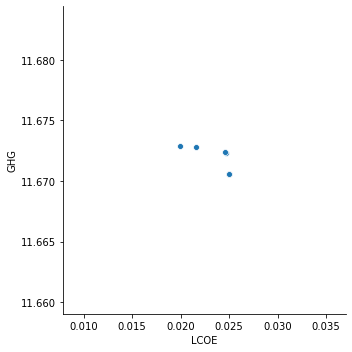


Figure 8: GHG vs LCOE.

*The three types of investment are too decoupled to make an interesting pareto frontier, and we also need a better solver if we want to push to lower right.*

## Run the interactive explorer for the decision space.

w = ty.DecisionWindow(evaluator)  
w.mainloop()

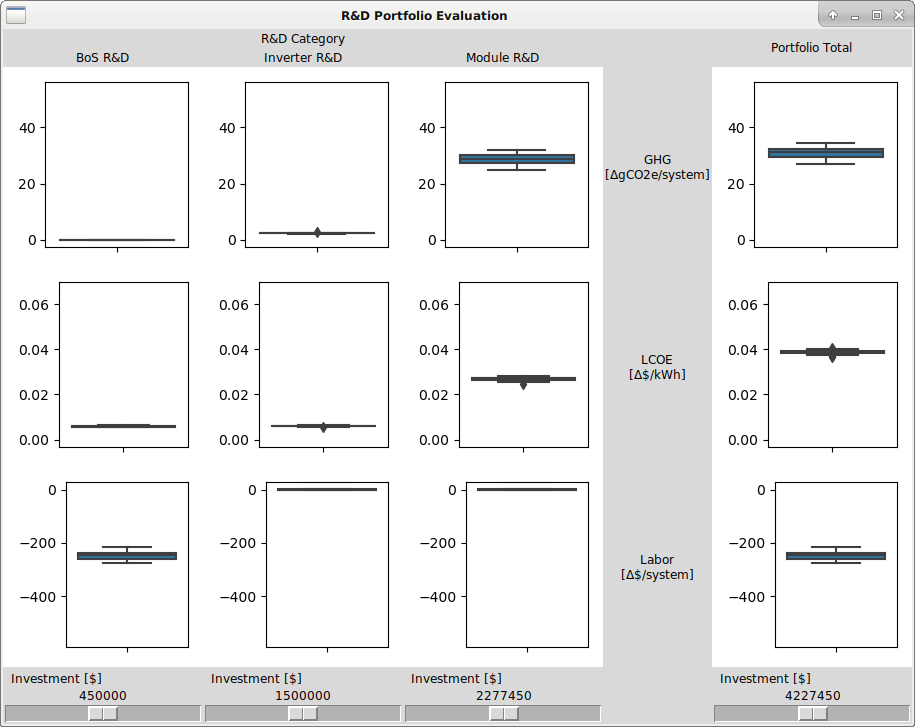


Figure 9: Interactive explorer for R&D portfolios.