Tyche Example with Simple PV Model

Set up.

One only needs to execute the following line once, in order to make sure recent enough packages are installed.

In [ ]:
```
#pip install numpy>=1.17.2 pandas>=0.25.1
```

Import packages.

In [1]:
```
import os
import sys
sys.path.insert(0, os.path.abspath("../src"))
```

In [2]:
```
import numpy             as np
import matplotlib.pyplot as pl
import pandas            as pd
import re                as re
import scipy.stats       as st
import seaborn           as sb

# The `tyche` package is located at <https://github.com/NREL/portfoli
o/tree/master/production-function/framework/code/tyche/>.
import tyche             as ty

from copy import deepcopy
```

Load data.

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

In [3]:
```
designs = ty.Designs("../data/utility_pv")
```

In [4]:
```
investments = ty.Investments("../data/utility_pv")
```

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

In [5]:
```
designs.compile()
```

Examine the data.

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

In [6]:
```
designs.functions
```

Out[6]:

Technology	Style	Module	Capital	Fixed	Production	Metrics	Notes
Simple pv	numpy	simple_pv	capital_cost	fixed_cost	production	metrics	

Right now, only the style `numpy` is supported.

The `indices` table defines the subscripts for variables.

In [7]:
```
designs.indices
```

Out[7]:

Technology	Type	Index	Offset	Description	Notes
Simple pv	Capital	Other Capital Cost	0	Other Capital Cost	Placeholder in case other capital costs are ne...
	Fixed	Other Fixed Cost	0	Other Fixed Cost	Placeholder in case other fixed costs are need...
	Input	Solar Radiation	0	Solar Radiation	
	Metric	GHG	1	Greenhouse gas emissions	
		LCOE	0	Cost	
	Output	Electricity	0	Electricity	

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

In [8]:
```
designs.designs
```

Out[8]:

Technology	Scenario	Variable	Index	Value	Units	Notes
Simple pv	Base PV	Input	Solar Radiation	5.5	kWh/m2/day	
		Input efficiency	Solar Radiation	0.152	1	From Kavlak et al. (2018)
		Input price	Solar Radiation	0	USD/kWh/m2/day	
		Lifetime	Other Capital Cost	20	yr	Assumed, Kavlak et al. (2019) do not provide a...
		Output efficiency	Electricity	1	1	No output inverter losses assumed
		Output price	Electricity	0.092	USD/kWh	Average commercial rate in Denver, CO
		Scale	NaN	0.05	module/yr	Inverse of lifetime. Constant needed to leveli...

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

In [9]:
```
designs.parameters
```

Out[9]:

Technology	Scenario	Parameter	Offset	Value	Units	Notes
Simple pv	Base PV	Cells per module	0	72	cell/module	From Kavlak et al. (2018)
		GHG factor for electricity	13	400	gCO2e/kWh	Rough approximation for US Grid
		Module area utilization	12	0.9	unitless	From Kavlak et al. (2018)
		Non-silicon materials cost	6	0.009433	$/cm2/cell	Calculated based on data from Kavlak et al. (2...
		Plant size	8	1000	MW/yr	From Kavlak et al. (2018). Equivalent to 3.35E...
		Polysilicon price	4	26	$/kg	2015$. From Kavlak et al. (2018)
		Production yield	11	0.95	unitless	Production waste parameter. Include as an outp...
		Reference plant cost	7	1.5513	$/cell	Calculated based on data from Kavlak et al. (2...
		Reference plant size	9	1000	MW/yr	From Kavlak et al. (2018). Equivalent to 3.35E...
		Scaling factor	10	0.27	unitless	From Kavlak et al. (2018)
		Silicon utilization	5	0.45	unitless	From Kavlak et al. (2018)
		Wafer area	1	243	cm2	From Kavlak et al. (2018)
		Wafer density	3	2.33	g/cm3	From Kavlak et al. (2018)
		Wafer thickness	2	180	um	From Kavlak et al. (2018)

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

In [10]:
```
designs.results
```

Out[10]:

Technology	Variable	Index	Units	Notes
Simple pv	Cost	Cost	USD/module	
	Metric	GHG	gCO2e/module	
		LCOE	USD/kWh	
	Output	Electricity	kWh/module	

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

In [11]:
```
investments.tranches
```

Out[11]:

Category	Tranche	Scenario	Notes
PV R&D	High PV R&D	Fast Progress on PV	
	Low PV R&D	Slow Progress on PV	
	Medium PV R&D	Moderate Progress on PV	
	No PV R&D	Base PV	

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

In [12]:
```
investments.investments
```

Out[12]:

Investment	Category	Tranche	Amount	Notes
No R&D Spending	PV R&D	No PV R&D	0.0	

Evaluate the scenarios in the dataset.

In [13]:
```
scenario_results = designs.evaluate_scenarios()
```

In [14]:
```
scenario_results.xs(1, level="Sample", drop_level=False)
```

Out[14]:

Technology	Scenario	Sample	Variable	Index	Value	Units
Simple pv	Base PV	1	Cost	Cost	-7.177702e+02	USD/module
			Metric	GHG	4.508260e+06	gCO2e/module
				LCOE	-6.368489e-02	USD/kWh
			Output	Electricity	1.127065e+04	kWh/module

Save results.

In [15]:
```
scenario_results.to_csv("output/utility_pv/results.csv")
```

NOTE: Items below have not been updated for simple PV module...

Plot GHG metric.

In [ ]:
```
g = sb.boxplot(
    x="Scenario",
    y="Value",
    data=scenario_results.xs(
        ["Metric", "GHG"],
        level=["Variable", "Index"]
    ).reset_index()[["Scenario", "Value"]],
    order=["Base Electrolysis", "Slow Progress on Electrolysis", "Mod
erate Progress on Electrolysis", "Fast Progress on Electrolysis"]
)
g.set(ylabel="GHG Footprint [gCO2e / gH2]")
g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Plot cost metric.

In [ ]:
```
g = sb.boxplot(
    x="Scenario",
    y="Value",
    data=scenario_results.xs(
        ["Metric", "Cost"],
        level=["Variable", "Index"]
    ).reset_index()[["Scenario", "Value"]],
    order=["Base Electrolysis", "Slow Progress on Electrolysis", "Mod
erate Progress on Electrolysis", "Fast Progress on Electrolysis"]
)
g.set(ylabel="Cost [USD / gH2]")
g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Plot employment metric.

```
In [ ]: g = sb.boxplot(
            x="Scenario",
            y="Value",
            data=scenario_results.xs(
                ["Metric", "Jobs"],
                level=["Variable", "Index"]
            ).reset_index()[["Scenario", "Value"]],
            order=["Base Electrolysis", "Slow Progress on Electrolysis", "Mod
        erate Progress on Electrolysis", "Fast Progress on Electrolysis"]
        )
        g.set(ylabel="Employment [job / gH2]")
        g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Evaluate the investments in the dataset.

```
In [ ]: investment_results = investments.evaluate_investments(designs, sample
        _count=50)
```

Costs of investments.

```
In [ ]: investment_results.amounts
```

Benefits of investments.

```
In [ ]: investment_results.metrics.xs(1, level="Sample", drop_level=False)
```

```
In [ ]: investment_results.summary.xs(1, level="Sample", drop_level=False)
```

Save results.

```
In [ ]: investment_results.amounts.to_csv("example-investment-amounts.csv")
```

```
In [ ]: investment_results.metrics.to_csv("example-investment-metrics.csv")
```

Plot GHG metric.

In [ ]:

```
g = sb.boxplot(
    x="Investment",
    y="Value",
    data=investment_results.metrics.xs(
        "GHG",
        level="Index"
    ).reset_index()[["Investment", "Value"]],
    order=["No R&D Spending", "Low R&D Spending", "Medium R&D Spending", "High R&D Spending"]
)
g.set(ylabel="GHG Footprint [gCO2e / gH2]")
g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Plot cost metric.

In [ ]:

```
g = sb.boxplot(
    x="Investment",
    y="Value",
    data=investment_results.metrics.xs(
        "Cost",
        level="Index"
    ).reset_index()[["Investment", "Value"]],
    order=["No R&D Spending", "Low R&D Spending", "Medium R&D Spending", "High R&D Spending"]
)
g.set(ylabel="Cost [USD / gH2]")
g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Plot employment metric.

In [ ]:

```
g = sb.boxplot(
    x="Investment",
    y="Value",
    data=investment_results.metrics.xs(
        "Jobs",
        level="Index"
    ).reset_index()[["Investment", "Value"]],
    order=["No R&D Spending", "Low R&D Spending", "Medium R&D Spending", "High R&D Spending"]
)
g.set(ylabel="Employment [job / gH2]")
g.set_xticklabels(g.get_xticklabels(), rotation=15);
```

Sensitity analysis.

Vary the four efficiencies in the design.

In [ ]:
```
# Four variables are involved.
variables = [
    ("Input efficiency" , "Water"       ),
    ("Input efficiency" , "Electricity"),
    ("Output efficiency", "Oxygen"      ),
    ("Output efficiency", "Hydrogen"    ),
]
```

In [ ]:
```
# Let efficiencies range from 0.75 to 0.975.
efficiencies = np.arange(0.750, 1.000, 0.025)
efficiencies
```

Start from the base case.

In [ ]:
```
base_design = designs.designs.xs("Base Electrolysis", level=1, drop_l
evel=False)
base_design
```

In [ ]:
```
base_parameters = designs.parameters.xs("Base Electrolysis", level=1,
drop_level=False)
base_parameters
```

Generate the new scenarios and append them to the previous ones.

In [ ]:
```
sensitivities = deepcopy(designs)
sensitivities.designs = sensitivities.designs[0:0]
sensitivities.parameters = sensitivities.parameters[0:0]
```

In [ ]:
```
# Iterate over variables and efficiencies.
for variable, index in variables:
    for efficiency in efficiencies:

        # Name the scenario.
        scenario = "Let " + variable + " @ " + index + " = " + str(ro
und(efficiency, 3))

        # Alter the base case.
        vary_design = base_design.rename(index={"Base Electrolysis" :
scenario}, level=1)
        vary_design.loc[("Simple electrolysis", scenario, variable, i
ndex), "Value"] = efficiency

        # Keep the parameters the same.
        vary_parameters = base_parameters.rename(index={"Base Electro
lysis" : scenario}, level=1)

        # Append the results to the existing table of scenarios.
        sensitivities.designs = sensitivities.designs.append(vary_des
ign)
        sensitivities.parameters = sensitivities.parameters.append(va
ry_parameters)
```

Remember to compile the design, since we've added scenarios.

In [ ]:
```
sensitivities.compile()
```

See how many rows there are in the tables now.

In [ ]:
```
sensitivities.designs.shape
```

In [ ]:
```
sensitivities.parameters.shape
```

In [ ]:
```
sensitivities.designs
```

Compute the results.

In [ ]:
```
results = sensitivities.evaluate_scenarios(1)
results
```

Plot the cost results.

In [ ]:
```
cost_results = results.xs("Cost", level="Variable").reset_index()[["S
cenario", "Value"]]
```

In [ ]:
```
cost_results[0:10]
```

In [ ]:
```
cost_results["Variable"  ] = cost_results["Scenario"].apply(lambda x:
re.sub(r'^Let (.*) @ (.*) =.*$', '\\1[\\2]', x))
cost_results["Efficiency"] = cost_results["Scenario"].apply(lambda x:
float(re.sub(r'^.*= (.*)$', '\\1', x)))
cost_results["Cost [USD/mole]"] = cost_results["Value"]
```

In [ ]:
```
cost_results = cost_results[["Variable", "Efficiency", "Cost [USD/mol
e]"]]
cost_results[0:10]
```

In [ ]:
```
# Here is a really simple plot.
cost_results.plot(
    x="Efficiency",
    y="Cost [USD/mole]",
    c=cost_results["Variable"].apply(lambda v: {
        "Input efficiency[Water]"       : "blue"   ,
        "Input efficiency[Electricity]" : "orange",
        "Output efficiency[Oxygen]"     : "green" ,
        "Output efficiency[Hydrogen]"   : "red"    ,
    }[v]),
    kind="scatter"
)
```