



Linkages between NREL's dGen, REopt, and SAM models

Brian Mirletz, Michaela Sizemore, Paritosh Das,
Jane Lockshin

National Renewable Energy Laboratory (NREL)

July 11th, 2023

SAM Webinars for 2023

Linkages between NREL's dGen, REopt, and SAM models	July 11
Financial Models for Utility-scale Projects in SAM	July 19
Modeling Utility Scale Photovoltaic Projects in SAM	August 23
Modeling Behind-the-meter (BTM) Batteries in SAM	September 20

Register for free at: <https://sam.nrel.gov/events.html>

Find webinar recordings at <https://sam.nrel.gov/>

Agenda

- 1** Introduction to dGen

- 2** Introduction to SAM

- 3** Introduction to REopt

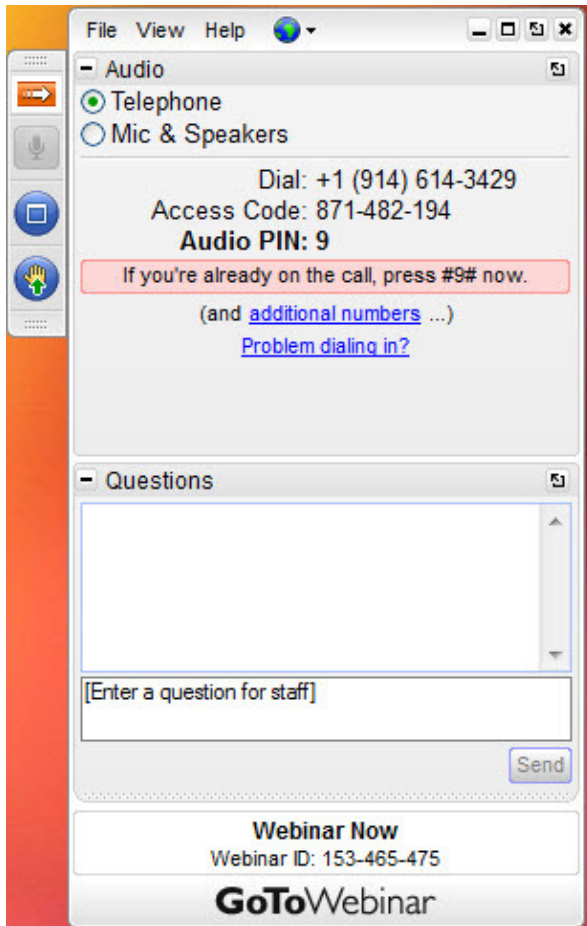
- 4** dGen-PySAM interface

- 5** SAM GUI-dGen interface demo

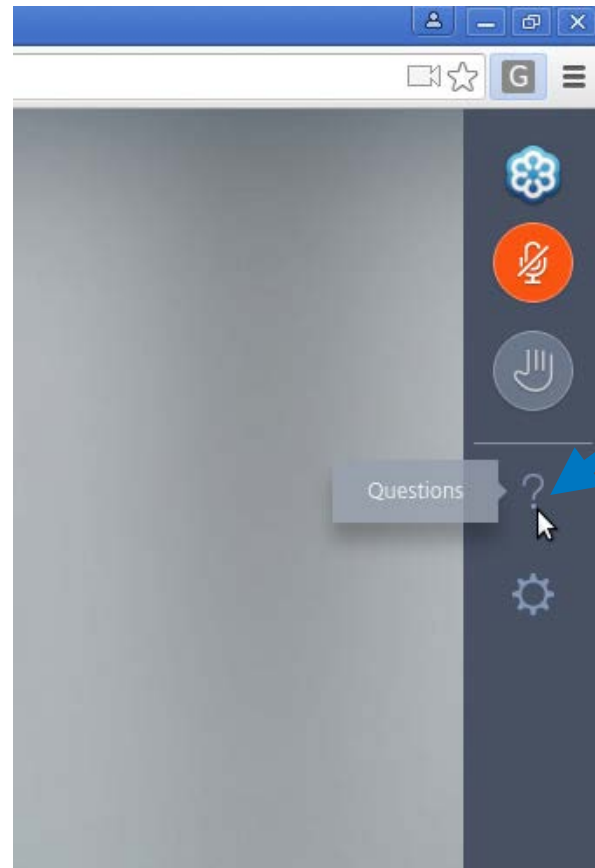
- 6** dGen-REopt interface demo

- 7** Q&A

Questions and Answers



Desktop application



Instant Join Viewer

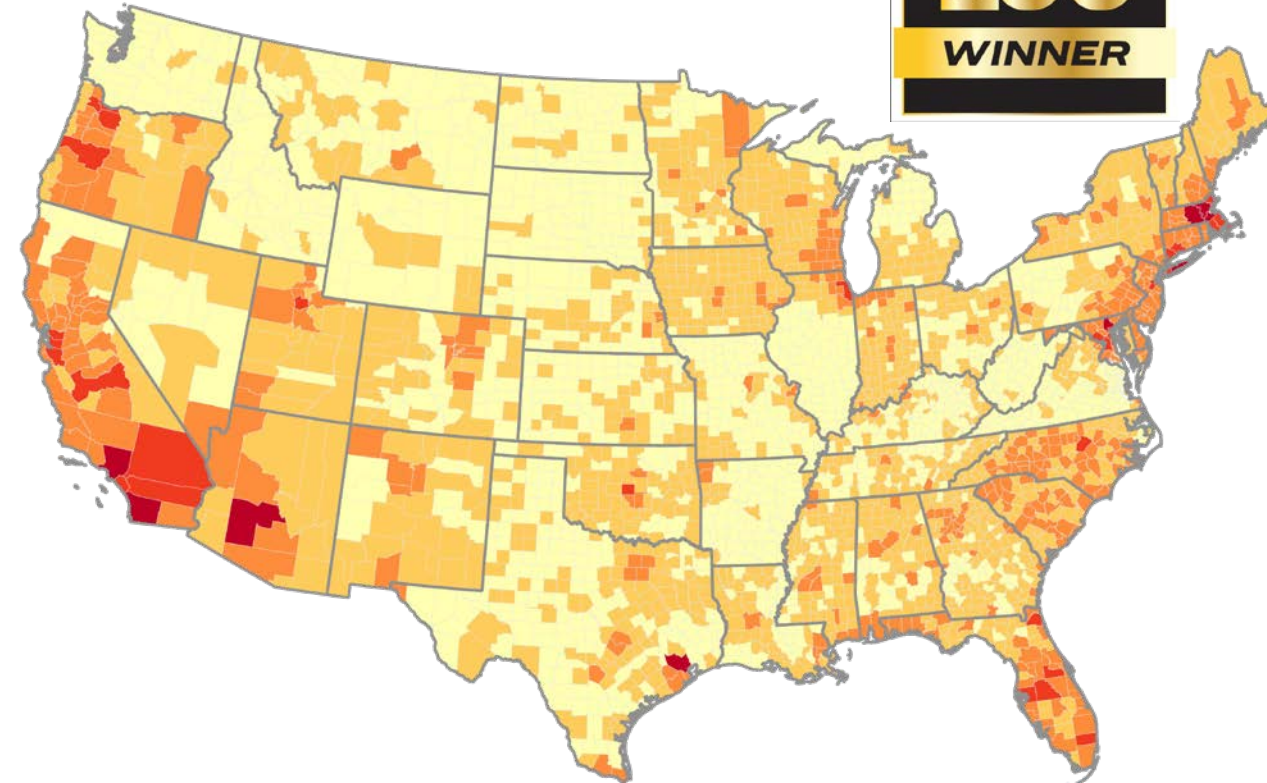
We will either type an answer to your question or answer it at the end of the presentation.

Find webinar recordings at <https://sam.nrel.gov/>

Forecasting Distributed Energy Resources: NREL's dGen™ Model



- Forecasts adoption of **distributed solar, storage, wind, and geothermal** by region and sector through 2050
- **Agent-Based Model** simulating **consumer decision-making**
- Incorporates detailed spatial data to understand **regional adoption trends**

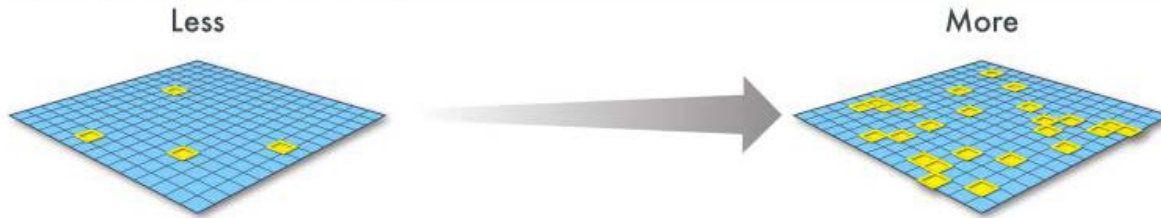


Source: Cole, Wesley, Will Frazier, Paul Donohoo-Vallett, Trieu Mai, and Paritosh Das. 2018 Standard Scenarios Report: A U.S. Electricity Sector Outlook <https://www.nrel.gov/docs/fy19osti/71913.pdf>

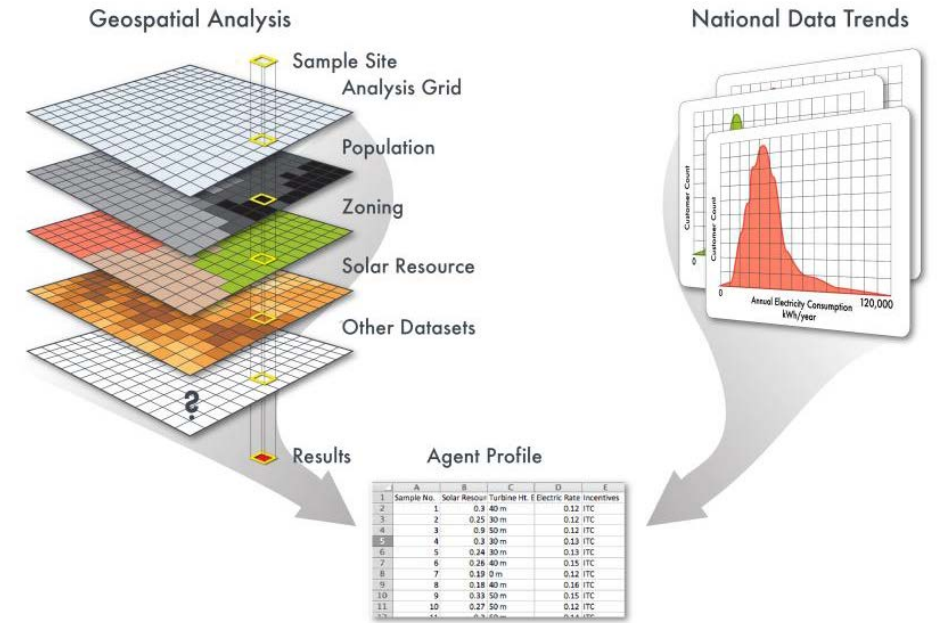
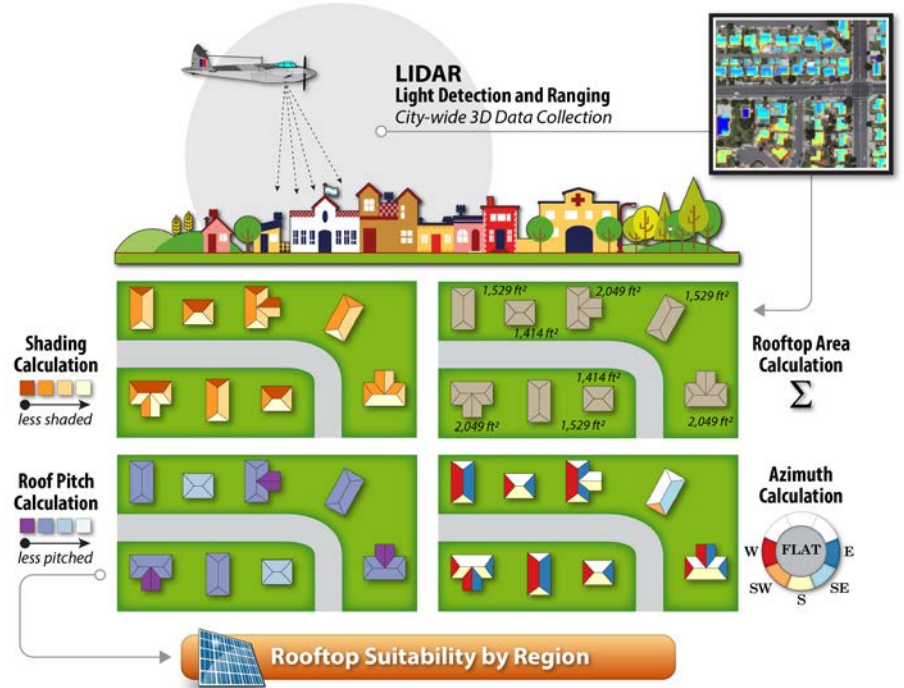
Used in several key analysis: Solar Futures Study, Storage Futures Study, Distributed Wind Energy Futures Study, LA100, PR100, LA100-Equity Strategies.

Agent-Based Consumer Level Understanding of DERs

1. Numer of Sample Points



2. Geographic Scale



Resilient Planning for DERs (RiDER): Data Standards and Multi-DER Evaluation

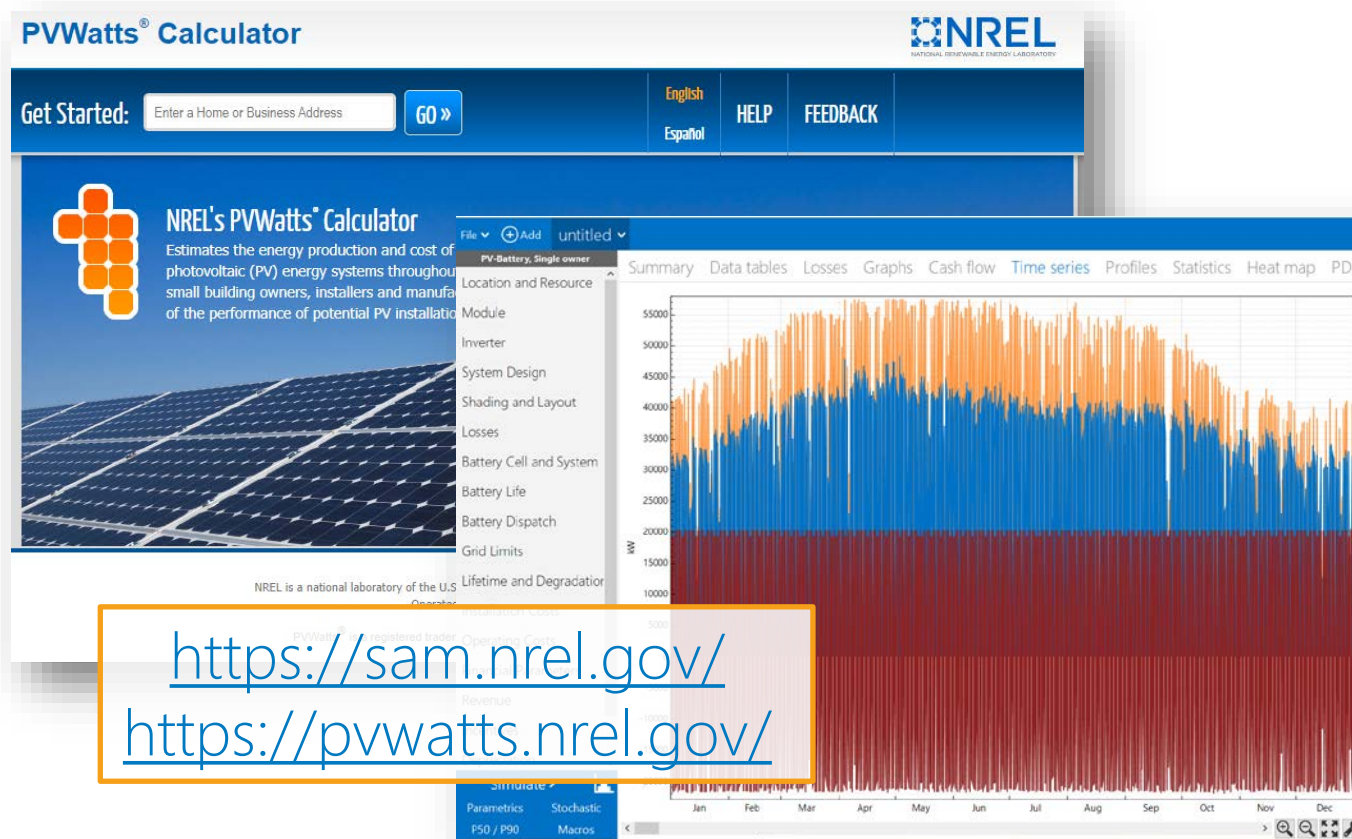
Consumers increasingly co-adopt distributed solar with (a) energy storage, b) energy efficiency appliances and (c) electric vehicles.

1. Improve Data Standardization and Access: **Publicly available** datasets
2. Model and Algorithm Development: **Standardized data (agents)** and new modules
3. Dissemination and Outreach: **Provide technical assistance** to ISO/RTOs, utilities, state and local energy planners, local governments, policymakers, and regulators.



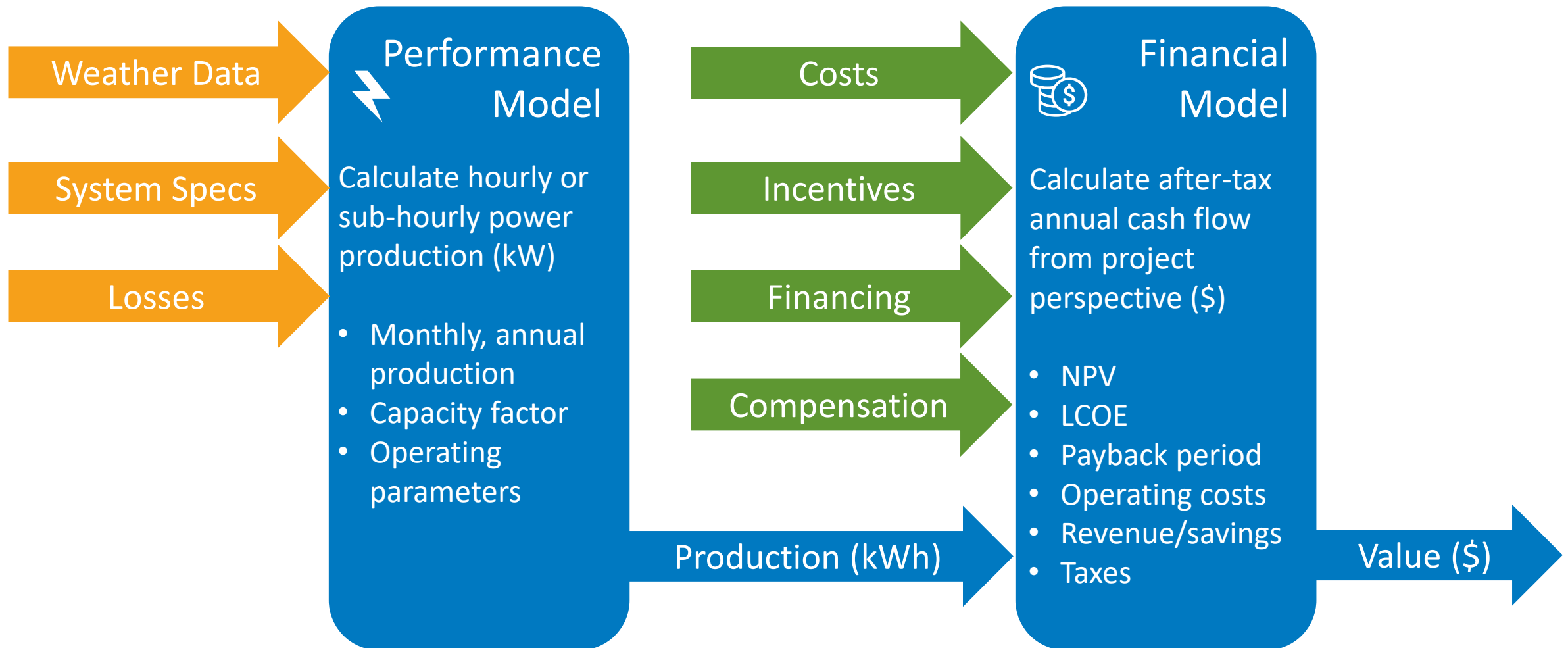
System Advisor Model (SAM) & PVWatts

Free software that enable detailed performance and financial analysis for renewable energy systems



- ✓ Desktop application
- ✓ PVWatts web tool & API
- ✓ Software development kit
- ✓ PySAM Python package
- ✓ Open source code
- ✓ Extensive documentation
- ✓ User support

Model Structure





Technologies

- Photovoltaic
- Energy storage
 - Electric battery
 - Electric thermal storage
- Concentrating solar power
- Industrial process heat
- Marine energy
- Wind power
- Fuel cell
- Geothermal power
- Solar water heating
- Biomass combustion
- Generic system

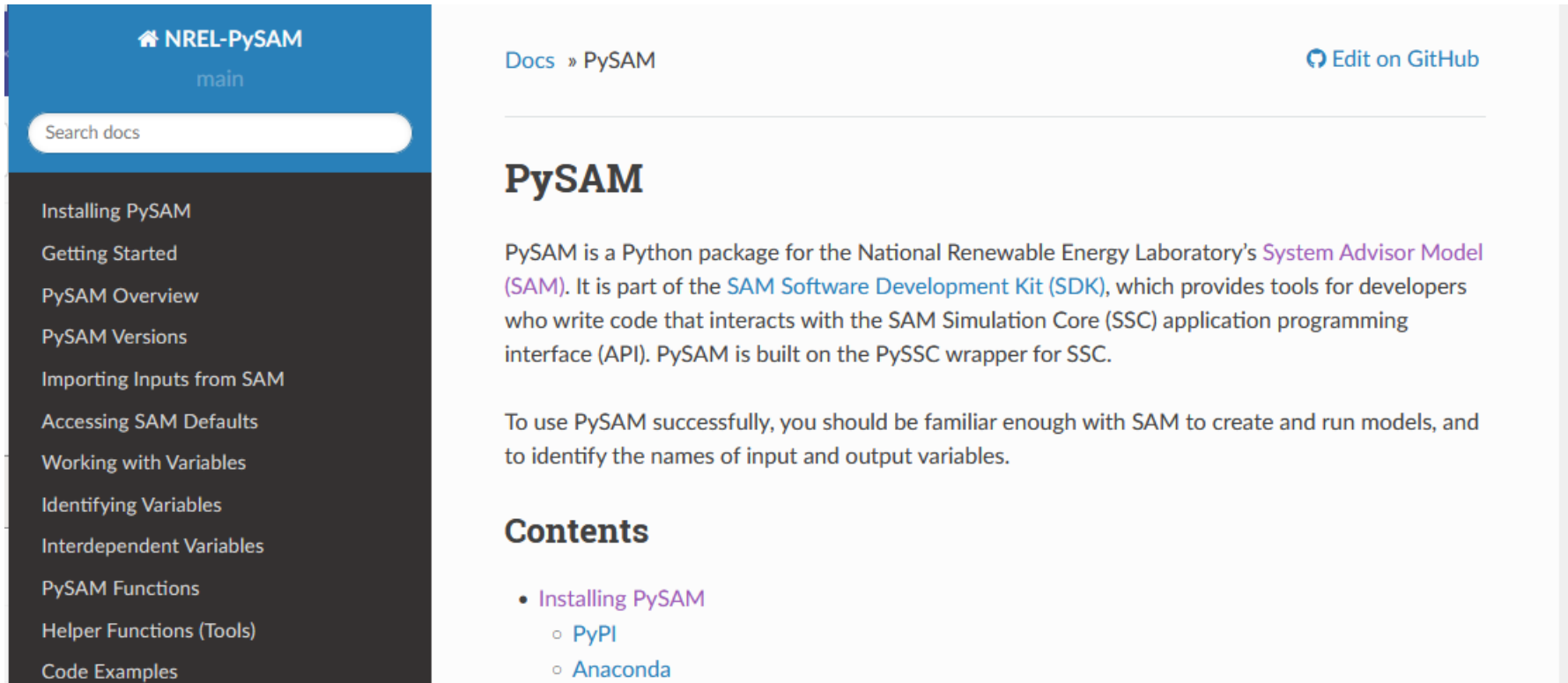
Financial Models

- Power purchase agreements
 - Single owner
 - Partnership flips
 - Sale leaseback
- Residential
- Commercial
- Third party ownership
- Merchant plant
- Community solar
- Simple LCOE calculator

What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Package name nrel-pysam
- Unit modules called compute_modules in the SSC code



The screenshot shows the documentation page for PySAM. The page has a blue header with the NREL-PySAM logo and a search bar. The main content area is white and contains the following text:

PySAM

PySAM is a Python package for the National Renewable Energy Laboratory's [System Advisor Model \(SAM\)](#). It is part of the [SAM Software Development Kit \(SDK\)](#), which provides tools for developers who write code that interacts with the SAM Simulation Core (SSC) application programming interface (API). PySAM is built on the PySSC wrapper for SSC.

To use PySAM successfully, you should be familiar enough with SAM to create and run models, and to identify the names of input and output variables.

Contents

- [Installing PySAM](#)
 - [PyPI](#)
 - [Anaconda](#)

What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Unit modules called `compute_modules` in the SSC code

A single simulation is a process chaining together multiple unit modules

- Order
- Information needs to be passed from one to the next

Generic System-Battery – Commercial Owner

Generic system model with battery storage. Renewable energy system displaces commercial building electric load.

Configuration name for defaults: *"GenericBatteryCommercial"*

[GenericSystem](#), [Battery](#), [Grid](#), [Utilityrate5](#), [Cashloan](#)

What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Unit modules called `compute_modules` in the SSC code

A single simulation is a process chaining together multiple unit modules

- Order
- Information needs to be passed from one to the next
- Assembled behind the scenes in SAM user interface

PySAM, and SAM's other software development kits, expose these unit modules so that they can be customized and embedded in software applications

What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

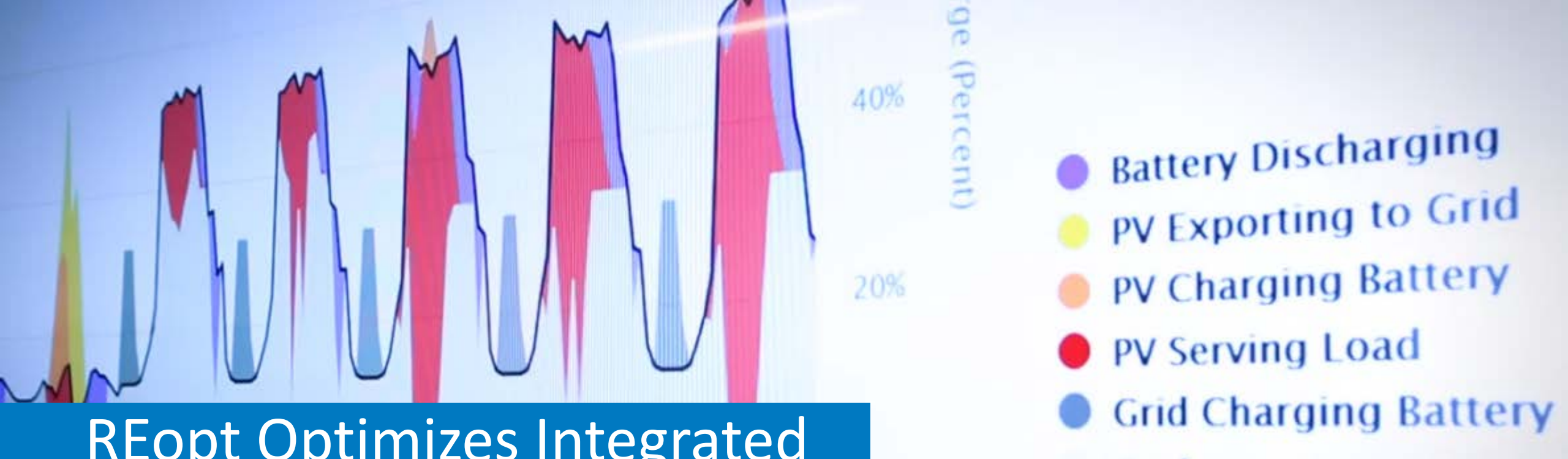
- Unit modules called `compute_modules` in the SSC code

A single simulation is a process chaining together multiple unit modules

- Order
- Information needs to be passed from one to the next
- Assembled behind the scenes in SAM user interface

PySAM, and SAM's other software development kits, expose these unit modules so that they can be customized and embedded in software applications

PySAM does NOT contain all the features in the SAM GUI

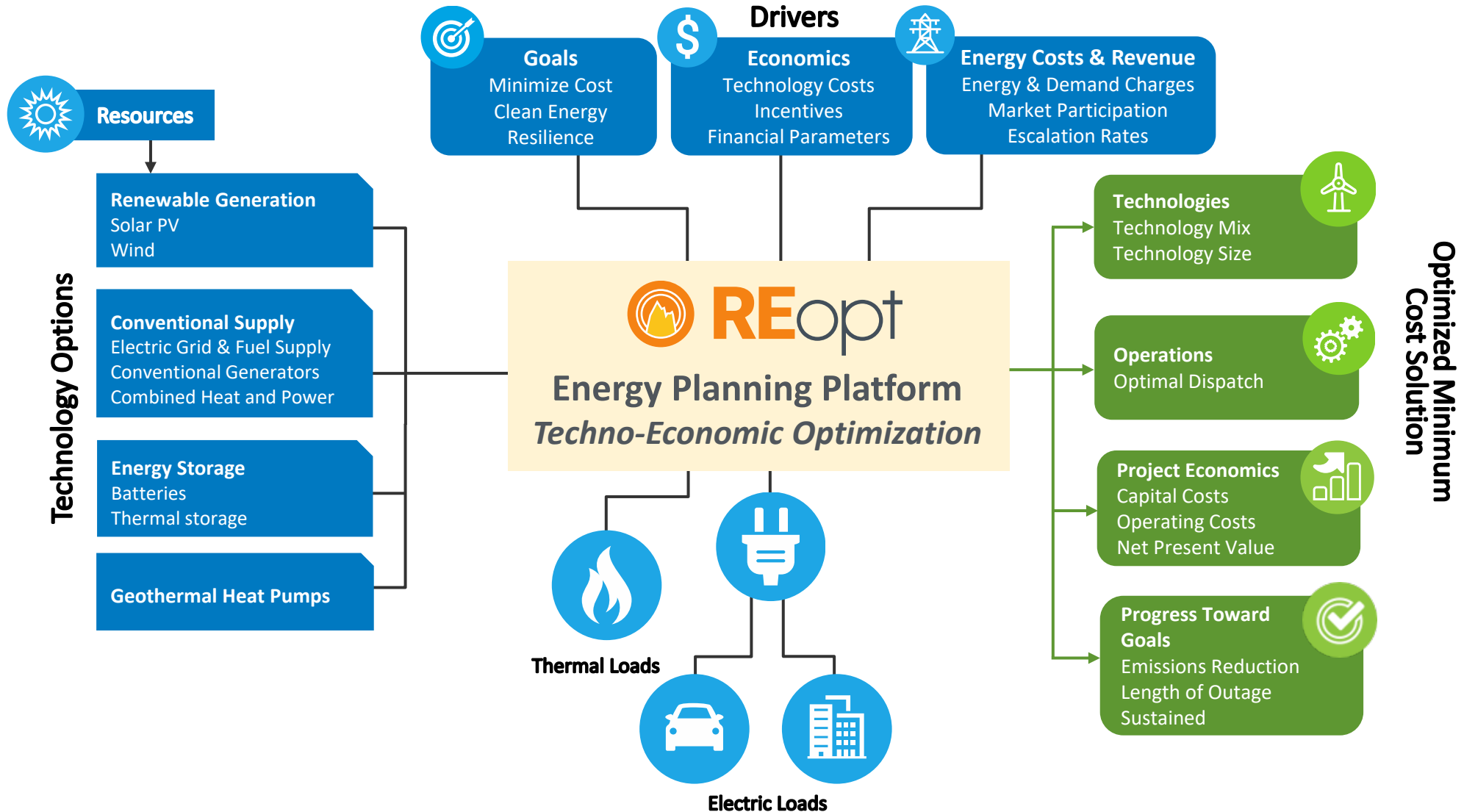


REopt Optimizes Integrated Energy Systems

- NREL's REopt® platform optimizes planning of generation, storage, and controllable loads to maximize the value of integrated systems.
- REopt considers electrical, heating, and cooling loads and technologies simultaneously to identify the optimal technology or mix of technologies.
- It transforms complex decisions into actionable results for building owners, utilities, developers, and industry.
- REopt analysis guides investment in economic, resilient, sustainable energy technologies.

REopt Energy Planning Platform

Formulated as a mixed integer linear program, REopt provides an integrated, cost-optimal energy solution.



REopt Provides Solutions for a Range of Users

Including researchers, developers, building owners, utilities, and industry



What is the optimal size of DERs to minimize my cost of energy?



How do I optimize system control across multiple value streams to maximize project value?



Where do market opportunities for DERs exist? Now and in the future?



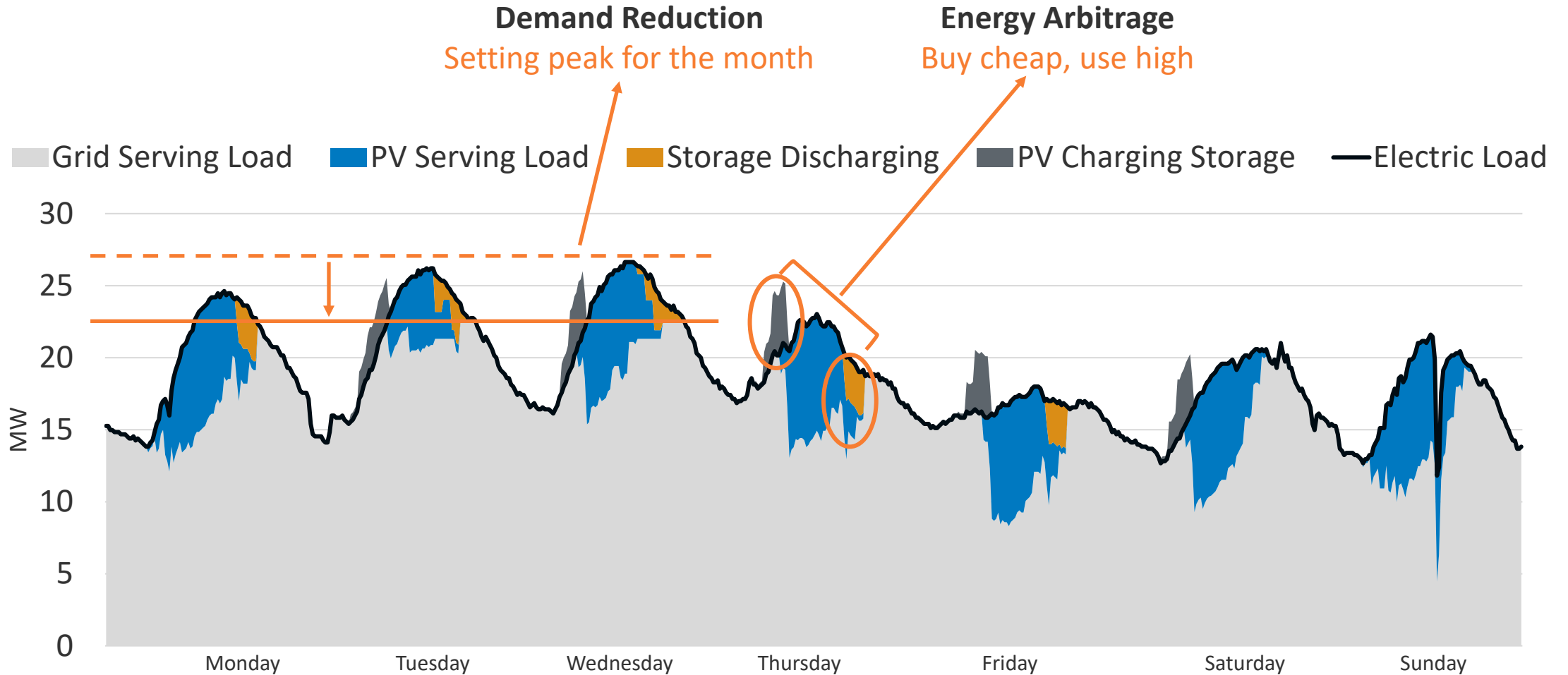
What will it cost to meet my sustainability or renewable energy goal?



What is the most cost-effective way to sustain a grid outage spanning 1 day? What about 9 days?

How Does REopt Work?

REopt considers the trade-off between ownership costs and savings across multiple value streams to recommend optimal size and dispatch.



Example of optimal dispatch of PV and BESS

Existing dGen->PySAM interface

- Agent data is translated into PySAM format
 - PV generation profiles (8760 of kWh/kW) already in database
- Remaining model chain:
 - Battery
 - UtilityRate5
 - Cashloan
- Defaults vary by residential vs commercial agent

Existing dGen->PySAM interface

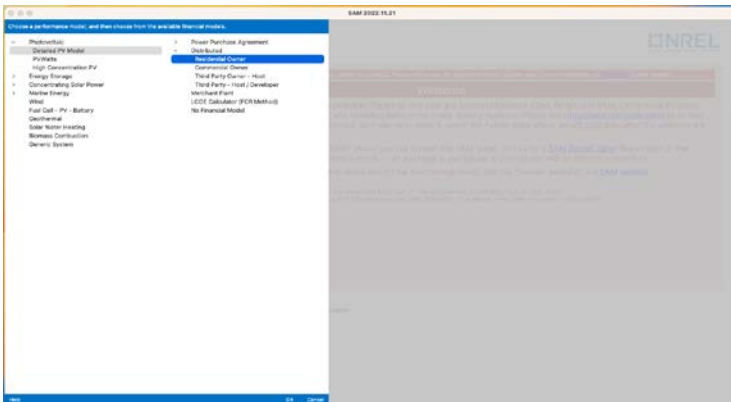
- Outputs include system size, net present value (NPV), bill savings, payback period
 - System size is based on best NPV after iterating
- These feed into adoption decisions

Leveraging SAM Output JSON Files with dGen Model

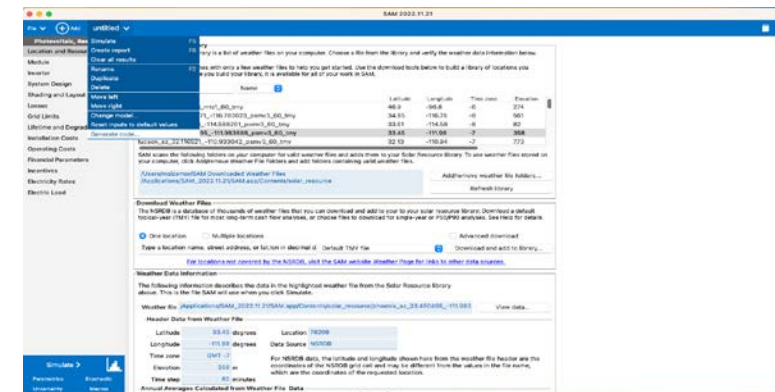
- SAM allows users to export system input parameters to a JSON file format.
- Update existing dGen PySAM integration to utilize exported SAM JSON file
 - Built in Python functionality to read from input JSON values to PySAM Utilityrate function
- Reduce hard coded values, increase accuracy in model

QuickStart for exporting JSON from SAM

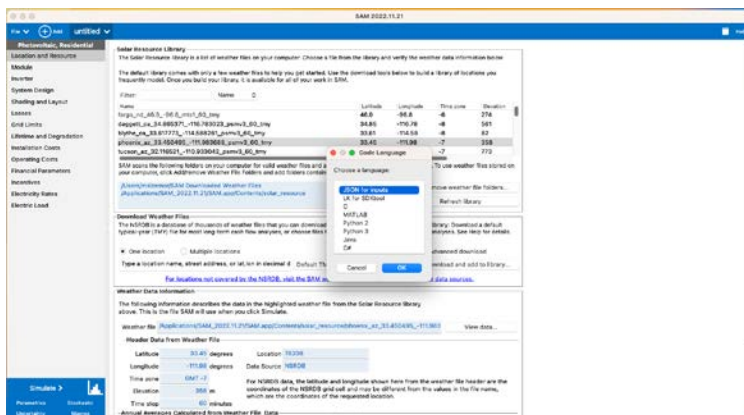
Step 1: Create new SAM project



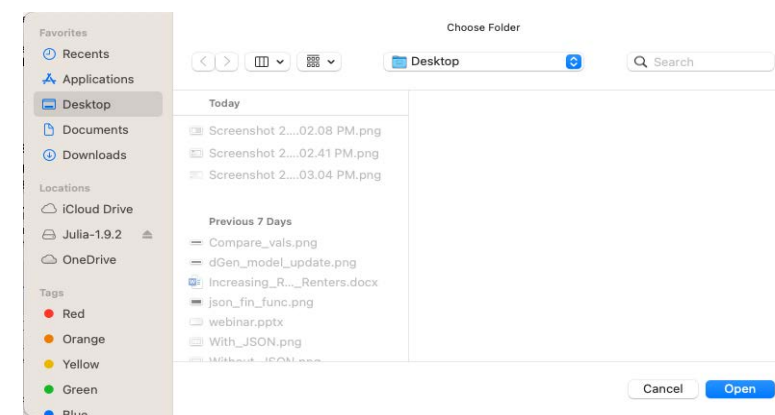
Step 2: From drop down next to project name, select “Generate code...”



Step 3: select “JSON for inputs”



Step 4: Select save location and click “open”



JSON file will save to the selected location in the format `project_name.json`

Proof of Concept: Impact of SAM JSON File Integration in dGen Model

Sample dGen Model Values Changing Utilityrate5 ONLY

Without JSON input

	first_year_elec_bill_with_system numeric	first_year_elec_bill_without_system numeric	first_year_elec_bill_savings numeric
1	637.1142253921997	646.4410813625852	9.326855970385509
2	603.149971836993	613.5194653798018	10.369493542808755
3	637.4838127293683	653.9727280590814	16.48891532971311
4	109.99949419425508	111.69874570356288	1.6992515093077998
5	531.0001055813658	543.1628681954838	12.162762614118037
6	478.15066582064236	493.50566395300274	15.354998132360379
7	360.3760762431331	371.2352873547047	10.859211111571597
8	688.2056269469812	705.4406726627893	17.235045715808155
9	154.7762162341478	156.88319837593855	2.106982141790752
10	253.26495654869936	261.76045795278094	8.49550140408158
11	231.3550465904813	234.53856174827988	3.183515157798581
12	278.15338778671816	290.54227445118215	12.388886664463996
13	241.38520382891164	245.60406464557508	4.218860816663437

With JSON input

	first_year_elec_bill_with_system numeric	first_year_elec_bill_without_system numeric	first_year_elec_bill_savings numeric
1	778.1142253921998	787.4410813625852	9.326855970385395
2	744.1499718369929	754.5194653798018	10.369493542808868
3	778.4838127293684	794.9727280590814	16.488915329712995
4	370.39949419425506	372.0987457035628	1.6992515093077714
5	749.1601055813658	761.3228681954838	12.162762614118037
6	696.3106658206426	711.6656639530026	15.354998132360038
7	578.5360762431333	589.3952873547047	10.859211111571653
8	829.205626946981	846.4406726627893	17.23504571580827
9	372.9362162341478	375.0431983759386	2.106982141790752
10	394.2649565486994	402.76045795278094	8.495501404081551
11	372.3550465904813	375.53856174827985	3.1835151577985243
12	419.15338778671816	431.54227445118215	12.388886664463996
13	382.38520382891164	386.60406464557514	4.2188608166634936

Proof of Concept: Changes to dGen Model Code

dgen model.py

```
233
234     if json_file:
235         # If there is a json file, use functions tailored to json inputs
236         # Calculate System Financial Performance
237         solar_agents.chunk_on_row(jff.calc_system_size_and_performance, sectors=scenario_settings.sectors, json_file=json_file, cores=cores, rate_switch_table=rate_switch_table)
238
239         # Calculate Maximum Market Share
240         solar_agents.on_frame(jff.calc_max_market_share, max_market_share)
241
242     else:
243         # Otherwise, use original financial functions
244         # Calculate System Financial Performance
245         solar_agents.chunk_on_row(financial_functions.calc_system_size_and_performance, sectors=scenario_settings.sectors, cores=cores, rate_switch_table=rate_switch_table)
246
247         # Calculate Maximum Market Share
248         solar_agents.on_frame(financial_functions.calc_max_market_share, max_market_share)
249
```

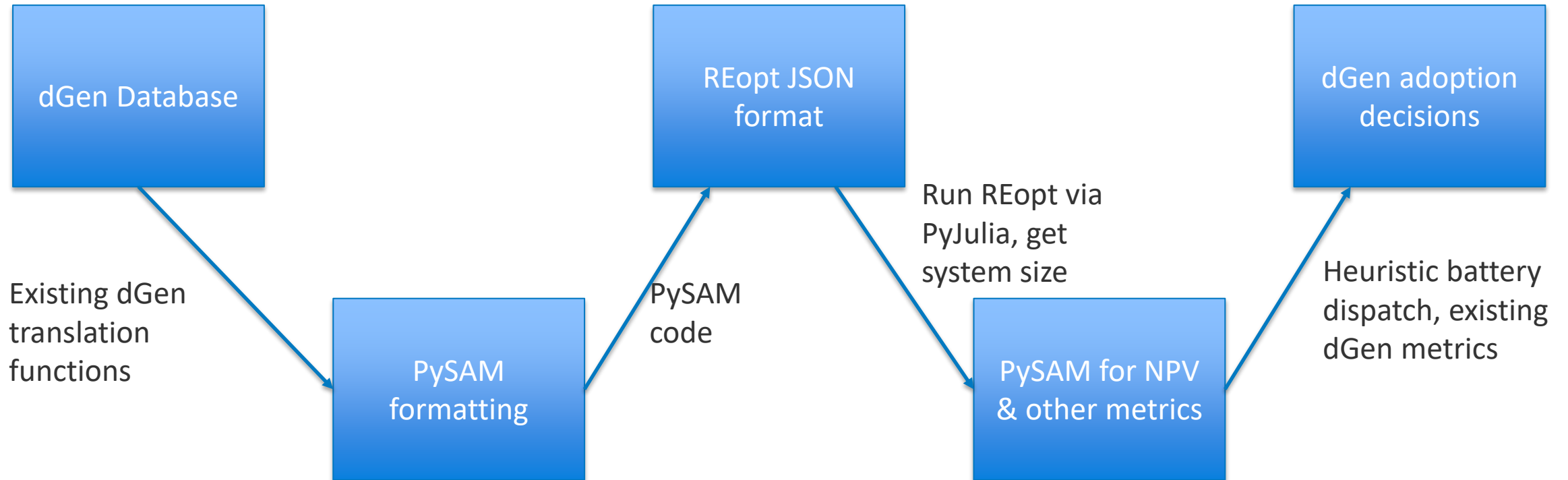
json financial functions.py (new)

```
717 def process_tariff(utilityrate, tariff_dict, json_file, net_billing_sell_rate):
718     """
719     Instantiate the utilityrate5 PySAM model and process the agent's rate json object to conform with PySAM input formatting.
720
721     Parameters
722     -----
723     agent : 'pd.Series'
724     | Individual agent object.
725     Returns
726     -----
727     utilityrate: 'PySAM.Utilityrates'
728     """
729
730     #####
731     ###----- UTILITYRATES -----###
732     ###---- FIXED AND ANNUAL CHARGES ----###
733     #####
734
735     # Monthly fixed charge [$]
736     utilityrate.ElectricityRates.ur_monthly_fixed_charge = json_file['ur_monthly_fixed_charge']
737     #tariff_dict['fixed_charge']
738     # Annual minimum charge [$]
739     utilityrate.ElectricityRates.ur_annual_min_charge = json_file['ur_annual_min_charge']
740     #0. # not currently tracked in URDB rate attribute downloads
741     # Monthly minimum charge [$]
742     utilityrate.ElectricityRates.ur_monthly_min_charge = json_file['ur_monthly_min_charge']
743     #0. # not currently tracked in URDB rate attribute downloads
744
```


Next Steps in dGen JSON Integration

- Reconfigure original `financial_functions.py` from dGen model to accept JSON file for input
 - Reduce overall addition of code to software
 - Increase user accessibility to new JSON functionality
- Testing edge cases from SAM options
 - Scenarios without Battery
 - No input JSON file
- Add additional PySAM JSON scenario values
 - Battery
 - Cashloan

dGen-REopt interface



Preliminary Results

- Overall adoption lower: 8.7 MW in 2022 with REopt vs 12.6 MW for SAM iterative method
- Payback periods \leq SAM calculated payback for 48/58 agents
- Still some QC to do around utility rate and financial representation
 - Validation with historical data is planned

Thank you! Questions?

SAM software: <https://sam.nrel.gov/>

SAM Repository: <https://github.com/NREL/SAM/>

PySAM Repository: <https://github.com/NREL/pysam>

REopt Repository: <https://github.com/NREL/REopt.jl>

dGen Repository: <https://github.com/NREL/dgen>

Q&A

www.nrel.gov

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number 38437. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

Photo from iStock-627281636

NREL/PR-7A40-86898

