Perovskite App Documentation

By Daniel Saul
April 22nd, 2023

App Link
GitHub Repository

Contents

I. App Description

The web application was created using the R programming language, RStudio software, and R Shiny which is an R package used for simple web app functionality. The first section of inputs allows the user to determine their location by either zipcode or latitude/longitude for local price and solar radiation purposes. The user can then select whether they have the levelized cost of electricity (LCOE) for comparison to the electricity prices. Depending on the selection, the inputs in the sidebar will adjust to show additional inputs required. If the user does not know the LCOE, the equation inputs are total installation cost, estimated annual operating costs, system lifetime in years, average annual degradation rate of the solar cell, the discount rate, and annual energy output of the solar panel.

If the user does not have the energy output value, they will be prompted with a few simple panel characteristics filled with default values recommended from NREL's PVWatts calculator to get the estimated energy output. Those inputs are module efficiency, losses, array type, module type, array area, panel tilt, and azimuth. Once those are changed or not, the app is ready to run and calculate LCOE or not and compare to local electricity prices.

Upon running, the LCOE, name of local utility company, residential, industrial, and commercial electricity prices along with various energy-related numbers will be outputted. The local electricity prices are pulled from the NREL Utility Rate API. Furthermore, a table showing the costs and energy outputs per lifetime year and basic visualizations will be shown. Both the table and visuals can be exported as an Excel sheet or PNG. If the user wants to run another calculation or reach an error, they should navigate to the restart session tab.

II. Instructions

1. Enter your location inputs to account for differences in regional prices and solar radiation. The current default values are for Athens, Georgia, United States.
2. Select whether you want to calculate the LCOE of a solar project/panel or just compare your LCOE to electricity prices.
3. Keep or update the default values (based on NREL PVWatts).
4. Press "Calculate" to run the simulation.
5. Download the table or visual outputs if necessary or navigate to the "Restart Session" tab to reset inputs and clear outputs.

III. Uses Cases

Goal: The user wants to compare the LCOE of a perovskite solar panel to local electricity prices of the area the panel will be installed.

Case 1: The user knows the LCOE of the perovskite panel.

Case 2: The user does not know the LCOE of the perovskite panel and has some basic characteristics of the panel.

IV. Calculations & Information

Levelized Cost of Energy (LCOE) Equation

$$\text{LCOE} = \frac{\text{total lifetime cost}}{\text{total lifetime electricity production}} = \frac{I + \sum_{i=0}^{N} \frac{\text{OM}}{1+r}}{\sum_{i=0}^{N} \frac{E(1-d)^i}{1+r}}$$

LCOE Equation Inputs

- I -- total initial investment to install the PV system (including cost of PV modules, racking, interconnects, labor, and permits)
- OM -- annual cost for operation and maintenance
- E --(un-degraded) annual energy output by the system as electricity, could be calculated by the PCE of PSC
- N --system lifetime in years (researchers usually assume PSC could last more than 10 years, which is impossible based on current technique, or there is no way to be cost effective)
- d -- annual module efficiency degradation rate
- r -- the nominal discount rate

LCOE Calculation R Code

```
# LCOE calculations/outputs
years <- 1:system_lifetime

energy_outputs <- energy_output * (1 -as.numeric(degradation_rate))^years

operating_costs <- operating_cost / (1 + discount_rate)^years

output_table <- data.frame(year = years, energy_output = energy_outputs,
install_cost = initial_investment, operating_cost = operating_costs, degradation_rate =
degradation_rate, discount_rate = discount_rate)

output_table <- round(output_table,4)

overall_lcoe <- round((initial_investment +
sum(output_table$operating_cost))/sum(output_table$energy_output),4)
```

NREL-Related Definitions

*Includes some specific terms not included in the application

Term	Type/Unit	Description
National Renewable Energy Laboratory (NREL)	Organization	The National Renewable Energy Laboratory (NREL) is a research institute that is focused on advancing renewable energy technologies. NREL has developed several tools and resources to help people and organizations understand and evaluate the potential of renewable energy, including a software tool called the System Advisor Model (SAM).
System Advisor Model (SAM)	Software	SAM is a free, open-source software program that allows users to model, analyze, and optimize the performance of renewable energy systems. It is widely used by industry professionals, researchers, and policy makers to evaluate the feasibility and potential benefits of using renewable energy technologies.
National Solar Radiation Database (NSRDB)	Database	The NSRDB is a comprehensive database of solar radiation and meteorological data for the United States. It is maintained by the National Renewable Energy Laboratory NREL and is freely available for download and use. The NSRDB contains hourly solar radiation and meteorological data for over 1,000 locations across the United States, and is widely used by researchers, industry professionals, and policy makers to evaluate the potential of renewable energy technologies. The NSRDB can be used in conjunction with the SAM to model and analyze the performance of renewable energy systems.
PVWatts	Software	PVWatts is a solar power calculation tool developed by the NREL. It is a free, online tool that allows users to estimate the electricity production and cost of energy of a grid-connected photovoltaic (PV) system based on a few simple inputs, such as the location, size, and type of PV system. PVWatts uses data from the NSRDB to calculate the amount of solar radiation that a PV system is likely to receive at a given location, and then estimates the electricity production and cost of energy of the PV system based on that data. PVWatts is widely used by industry professionals, researchers, and consumers to evaluate the potential of PV systems and to compare the performance of different PV technologies.
Typical Meteorological Year (TMY)	Dataframe	A TMY is a data set that provides a representative year of hourly meteorological data for a specific location. TMY data sets are commonly used in the

		renewable energy industry to model and analyze the performance of solar, wind, and other renewable energy technologies. TMY data sets are generated by selecting the most representative hour of data from each day of the year, based on long-term averages of meteorological data for a specific location. This allows users to easily evaluate the potential of renewable energy technologies for a specific location, without having to use multiple years of data. TMY data sets are widely used in conjunction with software tools like the SAM to model and analyze the performance of renewable energy systems.
Photovoltaic (PV)	Technology	Photovoltaic (PV) technology converts sunlight directly into electricity. PV systems are typically made up of many solar cells, which are arranged in panels or modules. When sunlight hits a solar cell, it excites the electrons in the cell, causing them to flow and generate a current. This current can then be used to power electrical devices or be sent to the grid for use by others. PV systems are a clean, renewable, and increasingly cost-effective source of electricity, and are widely used in a variety of applications, including residential and commercial rooftops, utility-scale power plants, and portable or off-grid systems.
Application Programming Interface (API)	Software	In the context of software development, an API is a set of rules and protocols that allows different software programs to communicate with each other and share data and functionality. An API defines how software components should interact and provides a standard way for developers to access and use the capabilities of a particular system or service. APIs are widely used in the software industry to enable different applications and services to work together, and to make it easier for developers to build new products and services on top of existing systems.
Latitude	Decimal Degree	Latitude is a geographical coordinate that specifies the north-south position of a point on the Earth's surface. It is usually expressed in degrees and is measured relative to the equator, with values ranging from 90 degrees north to 90 degrees south. The equator is the line of 0 degrees latitude, while the North Pole is at 90 degrees north and the South Pole is at 90 degrees south. Latitude is used along with

		longitude to specify the precise location of a point on the Earth's surface.
Longitude	Decimal Degree	Longitude is a geographical coordinate that specifies the east-west position of a point on the Earth's surface. It is measured relative to the Prime Meridian, which is the line of 0 degrees longitude that runs through the Royal Observatory in Greenwich, London, England. Longitude is typically expressed in degrees and ranges from 180 degrees west to 180 degrees east. It is used together with latitude to specify the precise location of a point on the Earth's surface.
Dew Point	Degrees Celsius	The dew point is the temperature at which air becomes saturated with water vapor, resulting in the formation of dew. It is an important measure of atmospheric moisture and is often used to predict the likelihood of fog, precipitation, and other forms of moisture in the air. The dew point is calculated based on the temperature and relative humidity of the air. When the temperature of the air cools to the dew point, moisture in the air will condense and form dew. This typically occurs at night when the air cools down, but it can also happen during the day if the air is cooled by other means, such as evaporation or contact with a cool surface.
Elevation	Meters	The elevation of the selected location where weather data is extracted.
Direct Normal Irradiance (DNI)	Watts Per Square Meter	DNI is a measure of the amount of solar radiation that is incident on a surface perpendicular to the sun's rays. It is typically measured in units of watts per square meter (W/m2) and is used to assess the potential efficiency of solar power systems. DNI is an important factor in the design and performance of solar collectors and other solar energy technologies, as it directly affects the amount of energy that can be harvested from the sun. In general, the higher the DNI, the more solar energy that can be collected and converted into usable power.
Global Horizontal Irradiance (GHI)	Watts Per Square Meter	GHI is a measure of the amount of solar radiation that is incident on a horizontal surface. It is typically measured in units of watts per square meter (W/m2) and is used to assess the potential efficiency of solar power systems. GHI is an important factor in the design and performance of solar collectors and other solar energy technologies, as it directly affects the

		amount of energy that can be harvested from the sun. In general, the higher the GHI, the more solar energy that can be collected and converted into usable power. GHI is different from direct normal irradiance (DNI), which is a measure of the solar radiation that is incident on a surface perpendicular to the sun's rays.
Diffuse Horizontal Irradiance (DHI)	Watts Per Square Meter	DHI is a measure of the amount of solar radiation that is incident on a horizontal surface after being scattered by the atmosphere. It is typically measured in units of watts per square meter (W/m2) and is used to assess the potential efficiency of solar power systems. DHI is an important factor in the design and performance of solar collectors and other solar energy technologies, as it can affect the amount of solar energy that can be collected and converted into usable power. In general, the higher the DHI, the more solar energy that can be collected, even on cloudy or overcast days. DHI is different from global horizontal irradiance (GHI), which is a measure of the total amount of solar radiation incident on a horizontal surface, including direct and diffuse radiation.
Pressure	Millibar	Atmospheric pressure is the pressure exerted by the weight of the atmosphere on the surface of the Earth. It is also known as barometric pressure and is typically measured in units of pascals or millibars. At sea level, the average atmospheric pressure is 1013.25 millibars. Atmospheric pressure varies depending on altitude, temperature, and weather conditions. It decreases with increasing altitude and can rise or fall due to changes in temperature and weather. Atmospheric pressure is an important factor in determining the weather and is used by meteorologists to make weather forecasts.
Wind Direction	Degrees	Wind direction refers to the direction from which the wind is blowing. It is typically reported in degrees, with 0 degrees being a wind blowing from the north, 90 degrees being a wind blowing from the east, 180 degrees being a wind blowing from the south, and 270 degrees being a wind blowing from the west. Wind direction is often reported in weather forecasts, along with wind speed, to help people understand the current wind conditions and how they may affect outdoor activities. Wind direction is determined

		using a wind vane, which is a device that uses the wind to move a pointer in the direction from which the wind is blowing.
Wind Speed	Meters Per Second	Wind speed is a measure of the speed at which wind is blowing. It is typically measured in units of meters per second, miles per hour, or knots. Wind speed is an important factor in determining the weather and can affect various activities, such as flying a kite, sailing a boat, or driving a car. Wind speed is often reported in weather forecasts, along with wind direction, to help people understand the current wind conditions and how they may affect outdoor activities. Wind speed is determined using an anemometer, which is a device that measures the speed of the wind as it blows past.
Surface Albedo	Ratio	Albedo is a measure of the amount of sunlight reflected by the ground. Most of the sunlight that reaches the surface of a photovoltaic module comes directly from the sun (direct or beam irradiance) or reflected from clouds or particles in the atmosphere (diffuse irradiance), but a small amount is also reflected from the ground (ground diffuse irradiance), depending on the array's orientation and the position of the sun.
Azimuth	Degrees	For a fixed array, the azimuth angle is the angle clockwise from true north describing the direction that the array faces. An azimuth angle of 180° is for a south-facing array, and an azimuth angle of zero degrees is for a north-facing array. For an array with one-axis tracking, the azimuth angle is the angle clockwise from true north of the axis of rotation. The azimuth angle does not apply to arrays with two-axis tracking. The default value is an azimuth angle of 180° (south-facing) for locations in the northern hemisphere and 0° (north-facing) for locations in the southern hemisphere. These values typically maximize electricity production over the year, although local weather patterns may cause the optimal azimuth angle to be slightly more or less than the default values. For the northern hemisphere, increasing the azimuth angle favors afternoon energy production, and decreasing the azimuth angle favors morning energy production. The opposite is true for the southern hemisphere. N: 0, NE: 45, E: 90, SE: 135, S: 180, SW: 225, W: 270, NW: 315.

System Capacity (Size)	Kilowatt	System capacity is a measure of the maximum amount of power that a solar panel system can produce. It is typically measured in units of watts or kilowatts, and is determined by the number and size of the solar panels in the system, as well as the efficiency of the panels and other components. Solar panel system capacity is an important factor in the design and performance of a solar power system, as it determines the amount of energy that the system can produce. In general, a larger solar panel system with high-efficiency panels will have a higher capacity and be able to generate more electricity than a smaller system with lower-efficiency panels. The DC system size is the DC (direct current) power rating of the photovoltaic array in kilowatts (kW) at standard test conditions (STC). In this instance, it is calculated by multiplying the array area by the module efficiency. It is calculated by multiplying the module efficiency and array area.
Inverter Efficiency	Percent	The inverter's nominal rated DC-to-AC conversion efficiency, defined as the inverter's rated AC power output divided by its rated DC power output. The efficiency of a solar panel inverter is a measure of how effectively the inverter converts the direct current (DC) electricity produced by the solar panels into alternating current (AC) electricity that can be used by household appliances and other devices. It is typically expressed as a percentage and can range from about 80% to more than 99%. The efficiency of a solar panel inverter is an important factor in the overall efficiency of a solar power system, as it affects the amount of electricity that can be generated and used by the system. In general, a higher-efficiency inverter will be able to convert more of the DC electricity produced by the solar panels into usable AC electricity, resulting in a more efficient solar power system.
Losses	Percent	Solar panel losses refer to the amount of solar energy that is not converted into usable electricity by the solar panels. There are several factors that can cause losses in a solar panel system, including the efficiency of the panels, the temperature of the panels, and shading or obstructions that block the sun's rays from reaching the panels. Solar panel losses are an important factor in the performance of a solar power system, as they can affect the amount of

		electricity that the system can generate. In general, reducing solar panel losses can improve the efficiency of a solar power system and increase the amount of electricity it can produce. These total losses include soiling, shading, snow, mismatch, wiring, connections, light-induced degradation, nameplate rating, age, and availability.
Ground Coverage Ratio (GCR)	Ratio	The ground coverage ratio (GCR) is the ratio of module surface area to the area of the ground or roof occupied by the array. The GCR applies only to arrays with one-axis tracking. A GCR of 0.5 means that when the modules are horizontal, half of the surface below the array is occupied by the array. An array with wider spacing between rows of modules has a lower GCR than one with narrower spacing. A GCR of 1 would be for an array with no space between modules, and a GCR of 0 for infinite spacing between rows. The default value is 0.4, and typical values range from 0.3 to 0.6.
DC to AC Ratio	Ratio	The DC to AC size ratio is the ratio of the array's DC rated size to the inverter's AC rated size. For the default value of 1.2, a 4 kW system size would be for a 4 DC kW array at standard test conditions (STC) and 4 DC kW / 1.2 = 3.33 AC kW inverter. For a system with a high DC to AC size ratio, for times when the array's DC power output exceeds the inverter's rated DC input power, the inverter limits the array's power output by increasing the DC operating voltage, which moves the array's operating point down its current-voltage (I-V) curve. The default value of 1.20 is reasonable for most systems. A typical range is 1.10 to 1.25, although some large-scale systems have ratios as high as 1.50. The optimal value depends on the system's location, array orientation, and module and inverter costs.
Fixed - Open Rack	PV Array Type	A fixed open rack solar panel is a type of solar panel that is mounted on a fixed, open frame or rack. This type of solar panel is typically used in ground-mounted solar installations, where the panels are positioned on a fixed angle and orientation to maximize their exposure to sunlight. Fixed open rack solar panels are a popular choice for large-scale solar installations, such as utility-scale solar power plants, because they are relatively easy to install and maintain. They are also more cost-effective than

		other types of solar panels, such as those that use tracking systems to follow the movement of the sun.
Fixed - Roof Mounted	PV Array Type	A fixed roof-mounted solar panel is a type of solar panel that is mounted on a fixed, angled frame or rack on the roof of a building. This type of solar panel is typically used in residential and commercial solar installations, where the panels are positioned at a fixed angle and orientation to maximize their exposure to sunlight. Fixed roof-mounted solar panels are a popular choice for many buildings because they are relatively easy to install and maintain. They are also a more cost-effective option than other types of solar panels, such as those that use tracking systems to follow the movement of the sun. In addition, fixed roof-mounted solar panels are less intrusive and less likely to cause visual disruptions than other types of solar panels.
1-Axis	PV Array Type	A 1-axis solar panel is a type of solar panel that is mounted on a single axis of rotation. This allows the panel to rotate and follow the movement of the sun throughout the day, which helps to maximize its exposure to sunlight and improve its overall efficiency. 1-axis solar panels are typically used in residential and commercial solar installations, where they are mounted on a fixed frame or rack on the roof of a building. The single axis of rotation allows the panels to track the sun as it moves across the sky, improving their efficiency and making them a more cost-effective option than fixed solar panels. Some 1-axis solar panels are manually adjusted, while others use automated tracking systems to follow the sun's movement.
Backtracked	PV Array Type	1-axis tracking with backtracking is a tracking algorithm that rotates the array toward the horizontal during early morning and late evening hours to reduce the effect of self-shading.
2-Axis	PV Array Type	A 2-axis solar panel is a type of solar panel that is mounted on a frame or rack with two axes of rotation. This allows the panel to rotate and tilt in two different directions, which allows it to follow the movement of the sun more precisely and maximize its exposure to sunlight. 2-axis solar panels are typically used in residential and commercial solar installations, where they are mounted on the roof of a building. The dual axes of rotation allow the panels

		to track the sun as it moves across the sky, improving their efficiency and making them a more cost-effective option than fixed solar panels. Some 2-axis solar panels are manually adjusted, while others use automated tracking systems to follow the sun's movement.
Standard: Polycrystalline	PV Module Type	A standard polycrystalline solar panel is a type of solar panel that uses polycrystalline silicon photovoltaic cells to convert sunlight into electricity. Polycrystalline solar panels are made up of many small silicon crystals, which gives them a distinctive blue or purple color. They are a popular choice for residential and commercial solar installations because they are relatively efficient, durable, and cost-effective. Standard polycrystalline solar panels are typically rectangular in shape and are mounted on a fixed frame or rack on the roof of a building. They are a good choice for many solar installations because they can be easily integrated with existing electrical systems and have a relatively low environmental impact. Approximate nominal efficiency is 19% with glass with anti-reflective coating as the module cover. Temperature coefficient of power is -0.37%/degrees Celsius.
Premium: Monocrystalline	PV Module Type	A premium monocrystalline solar panel is a type of solar panel that uses a single, large crystal of silicon to convert sunlight into electricity. Monocrystalline solar panels are made up of a single, continuous crystal of silicon, which gives them a distinctive black or dark blue color. They are the most efficient type of solar panel. Premium monocrystalline solar panels are typically used in high-end residential and commercial solar installations, where efficiency and performance are a top priority. They are also more expensive than other types of solar panels, such as polycrystalline solar panels, but they offer a higher return on investment over the long term. Approximate nominal efficiency is 21% with glass with anti-reflective coating as the module cover. Temperature coefficient of power is -0.35%/degrees Celsius.
Thin film	PV Module Type	A thin film solar panel is a type of solar panel that uses a thin layer of photovoltaic material to convert sunlight into electricity. Unlike traditional solar panels, which use crystalline silicon cells, thin film

		solar panels use a layer of photovoltaic material that is only a few micrometers thick. This allows them to be much more flexible and lightweight than traditional solar panels, and they can be easily integrated into a variety of different surfaces and structures. Thin film solar panels are a popular choice for portable solar installations, such as solar-powered backpack chargers, and for building-integrated photovoltaic (BIPV) systems, where the panels are integrated into the roof or facade of a building. They are also a more cost-effective option than traditional solar panels for large-scale solar installations, such as utility-scale solar power plants.
Analysis Period	Years	The amount of time in years you want to run the simulation against. In other words, what is the financial and system lifetime degradation output during the analysis period.
DC Degradation	Percent	DC degradation is a term used to describe the decrease in performance of a photovoltaic (PV) system over time. In a PV system, the solar panels generate direct current (DC) electricity, which is then converted into alternating current (AC) electricity by an inverter. DC degradation occurs when the solar panels experience a loss of efficiency or a decrease in their ability to generate electricity. This can be caused by a number of factors, including aging of the solar panels, damage or wear and tear, and exposure to extreme weather conditions. DC degradation can have a significant impact on the performance of a PV system, reducing its overall output and the amount of electricity it generates. It is important to monitor and maintain a PV system to prevent or minimize DC degradation.
Array Type	Panel Characteristic	A solar panel array is a group of solar panels that are connected together to generate electricity. The type of solar panel array that is used in a particular installation will depend on a number of factors, including the size of the installation, the amount of electricity needed, and the location and orientation of the panels. The array type describes whether the PV modules in the array are fixed, or whether they move to track the movement of the sun across the sky with one or two axes of rotation. It includes Fixed - Open Rack, Fixed - Roof Mounted, 1-Axis, Backtracked, and 2-Axis.

Module Type	Panel Characteristic	The type of the actual module in the solar panel arrays gathering sunlight. They include Standard: Polycrystalline, Premium: Monocrystalline, and Thin Film.
Capacity Factor	Percent	The ratio of the system's predicted electrical output in the first year of operation to the nameplate output, which is equivalent to the quantity of energy the system would generate if it operated at its nameplate capacity for every hour of the year. (AC-to-DC)
AC Output	Kilowatt-Hours or Watts	AC output refers to the alternating current (AC) electricity that is produced by a solar power system. AC electricity is the type of electricity that is typically used in homes and businesses. Solar panels produce direct current (DC) electricity, which must be converted to AC electricity by an inverter before it can be used. The AC output of a solar power system depends on the efficiency of the solar panels and inverters, the amount of sunlight that is available, and the load on the system.
DC Input	Kilowatt-Hours or Watts	DC output refers to the direct current (DC) electricity that is produced by a solar power system. Solar panels produce DC electricity, which is the type of electricity that is typically used in batteries and other low-voltage systems. The DC output of a solar power system depends on the efficiency of the solar panels, the amount of sunlight that is available, and the load on the system. To use the DC electricity that is produced by a solar panel, it must be converted to alternating current (AC) electricity by an inverter before it can be used to power most appliances and devices.
Solar Radiation	Kilowatt-Hours Per Square Meter Per Day or Watts per Meter squared	Solar radiation, also known as the solar resource or insolation, is a measure of the amount of solar energy that is available at a particular location and time. The amount of solar radiation that an area receives can vary depending on factors such as the time of year, the weather, and the location. In general, areas closer to the equator tend to receive more solar radiation than areas closer to the poles.
Ambient Temperature	Celsius	The temperature of the air or the immediate surroundings of a photovoltaic (PV) module or solar panel. It is an important parameter for determining the performance and efficiency of a solar panel, as temperature can have a significant impact on the electrical output of a PV system.

Cell Temperature	Celsius	The temperature of the individual solar cells that make up the panel. It is an important parameter for determining the performance and efficiency of a photovoltaic (PV) system, as it affects the electrical output of the cells and, in turn, the overall output of the panel.
Module Efficiency	Percent	The percentage of solar energy that is converted into electricity by a photovoltaic (PV) module. In other words, it measures how effectively a solar panel can convert the energy from the sun into usable electrical power.
Temperature	Celsius	The temperature of the air at a given time in the weather data tables.
Plane of Array Irradiance	Kilowatts or Watts per Meter squared	The amount of solar irradiance that reaches a photovoltaic (PV) module or array in a specific plane, which is typically defined as a plane perpendicular to the axis of the module. In other words, it is the amount of sunlight that is received by the PV module after considering factors such as shading, reflection, and orientation.
Bifaciality	Ratio	A bifacial photovoltaic module is a type of module that converts solar irradiance incident on both the front and rear of the module into electricity. For a system with bifacial modules, PVWatts calculates the plane-of-array irradiance on the rear side of the array in each hour based on the available solar resource, position of the sun, and array orientation, and considering ground reflectance (albedo) and spacing between rows of modules. It assumes a bifaciality factor of 0.7, transmission factor of 0.013, and ground clearance height of 1 meter. The bifacial option is not available for the Fixed Roof Mount array type because bifacial modules require space between the module and roof surface to allow sunlight to reach the back of the module.
Soiling Loss	Percent	Losses due to dirt and other foreign matter on the surface of the PV module that prevent solar radiation from reaching the cells. Soiling is location- and weather-dependent. There are greater soiling losses in high-traffic, high-pollution areas with infrequent rain. The default value is 2%. You can use monthly irradiance losses for losses that vary by month instead of including soiling in the system losses.
Shading Loss	Percent	Reduction in the incident solar radiation from shadows caused by objects near the array such as

		buildings or trees, or by self-shading for modules arranged in rows when modules in one row cause shadows on those in an adjacent row. The default value of 1% represents an array with little or no shading. Shading analysis tools can determine a loss percentage for shading by nearby objects. You should not use the shading loss to account for self-shading with the one-axis tracking option.
Snow Loss	Percent	Reduction in the system's annual output due to snow covering the array. The default value is zero, assuming either that there is never snow on the array, or that the array is kept clear of snow. You can use the monthly irradiance losses to represent snow loss instead of including it in the system losseds.
Mismatch Loss	Percent	Electrical losses due to slight differences caused by manufacturing imperfections between modules in the array that cause the modules to have slightly different current-voltage characteristics. The default value of is 2%.
Wiring Loss	Percent	Resistive losses in the DC and AC wires connecting modules, inverters, and other parts of the system. The default value is 2%.
Connections Loss	Percent	Resistive losses in electrical connectors in the system. The default value is 0.5%.
Light-Induced Degradation Loss	Percent	Effect of the reduction in the array's power during the first few months of its operation caused by light-induced degradation of photovoltaic cells. The default value is 1.5%.
Nameplate Rating Loss	Percent	The nameplate rating loss accounts for the accuracy of the manufacturer's nameplate rating. Field measurements of the electrical characteristics of photovoltaic modules in the array may show that they differ from their nameplate rating. A nameplate rating loss of 5% would indicates that testing yielded power measurements at Standard Test Conditions (STC) that were 5% less than the manufacturer's nameplate rating. The default value is 1%.
Age Loss	Percent	Effect of weathering of the photovoltaic modules on the array's performance over time. The default value is zero.
Availability Loss	Percent	Reduction in the system's output cause by scheduled and unscheduled system shut down for maintenance, grid outages, and other operational factors. The default value is 3%.

System Capacity (Size) Equation

Size (kW) = Array Area (m^2) × 1 kW/m^2 × Module Efficiency (%)

System Capacity (Size) Calculation R Code

```
system_capacity <- array_area * 1 * module_efficiency
```

V. Application Programming Interfaces (APIs) Used

NREL PVWatts V8
https://developer.nrel.gov/docs/solar/pvwatts/v8/

- This API is used to estimate the annual energy output of a solar panel based on location, system, and panel inputs.

NREL Utility Rates V3
https://developer.nrel.gov/docs/electricity/utility-rates-v3/

- This API is used to pull the location-based power company’s residential, commercial, and industrial electricity rates.

FRED Inflation Rate
https://fred.stlouisfed.org/docs/api/fred/series_observations.html

- This API is for pulling the current inflation rate in the United States to account for cost changes over time. The code is commented out and not currently used for this application and remains in the file for future use.

VI. Appendix A: Sources

https://www.sciencedirect.com/science/article/pii/S2542435118301910#undfig1
https://pubs.rsc.org/en/content/articlehtml/2020/se/c9se00948e
https://pvwatts.nrel.gov/pvwatts.php
https://www.nrel.gov/docs/fy14osti/62641.pdf
https://nsrdb.nrel.gov/
https://nsrdb.nrel.gov/data-sets/tmy

VII. Appendix B: Code

```
# Import packages
library(shiny) # Package required for shiny
library(rsconnect) # Connection to shinyapps.io servers
library(shinythemes) # UI themes for shiny
library(shinydashboard) # Dashboard functionality
library(DT) # Data table functionality
library(tidyverse) # Data wrangling/transformation
library(shinyalert) # Used to create the popups/alerts
```

```
library(shinyBS) # Tooltip descriptions
library(shinyjs) # 'Quit' page to prevent freezes on local machine
library(leaflet) # Location and mapping
library(zipcodeR) # Zipcode input functionality
library(plotly) # Nice plots
library(lubridate) # Dealing with time series data
library(httr) # API calls
library(forecast)
library(jsonlite)
library(reshape2)
library(writexl) # Download excel files
library(xlsx)
library(readxl)
library(lpSolve) # Linear programming for financial model

ui <- fluidPage(

  theme = shinytheme("lumen"),
  navbarPage(title="Perovskite Cell Model",
             windowTitle = "Perovskite Cell Model",


             tabPanel(
               'Calculator',
               icon = icon("calculator"),

               h2('Instructions'),
               p('1. Enter your location inputs to account for differences in regional prices
and solar radiation. The current default values are for Athens, Georgia, United
States.'),
               p('2. Select whether or not you want to calculate the LCOE of a solar
project/panel or just compare your LCOE to electricity prices.'),
               p('3. Keep or update the default values (based on NREL PVWatts).'),
               p('4. Press "Calculate" to run the simulation.'),
               p('5. Download the table or visual outputs if necessary, or navigate to the
"Restart Session" tab to reset inputs and clear outputs.'),
               hr(),

               sidebarLayout(
                 sidebarPanel(
                   titlePanel("Inputs"),
                   width = 3,

                   selectInput(inputId = "location_type",
                               label = "Location Input Type:",
                               width = "200px",
```

```
                    choices = c("Latitude/Longitude", "Zipcode")),

            conditionalPanel(condition = "input.location_type == 'Latitude/Longitude'",
                         numericInput(inputId = "lat",
                              label = "Latitude:",
                              width = '200px',
                              value = '33.94',
                              min=-90,
                              max=90),
                         bsTooltip('lat',
                                "Latitude of weather location. Please round to two
decimals and provide a value between -90 and 90.",
                                'right',
                                options = list(container = 'body')),

                         numericInput(inputId = "lon",
                              label = "Longitude:",
                              width = '200px',
                              value = '-83.37',
                              min=-180,
                              max=180),
                         bsTooltip('lon',
                                "Longitude of weather location. Please round to two
decimals and provide a value between -180 and 180.",
                                'right',
                                options = list(container = 'body'))),

            conditionalPanel(condition = "input.location_type == 'Zipcode'",

                         numericInput(inputId = "zipcode",
                              label = "Zipcode:",
                              width = '200px',
                              value = '30601',
                              min=00000,
                              max=99999),
                         bsTooltip('zipcode',
                                "The 5-digit zipcode of the weather location. Please do
not add any spaces or dashes.",
                                'right',
                                options = list(container = 'body'))

                         ),

            selectInput(inputId = "calc_type",
                    label = "Calculation Type:",
                    width = "200px",
```

```
                        choices = c("Find LCOE", "Compare LCOE")),

                conditionalPanel(condition = "input.calc_type == 'Find LCOE'",


                            numericInput(inputId = "initial_investment",
                                     label = "Total Installation Cost ($):",
                                     width = '200px',
                                     value = '20000',
                                     min=0),
                            bsTooltip('initial_investment',
                                   "The total initial investment to install the PV system
(including cost of PV modules, racking, interconnects, labor, and permits).",
                                   'right',
                                   options = list(container = 'body')),

                            numericInput(inputId = "operating_cost",
                                     label = "Annual Operating Cost ($):",
                                     width = '200px',
                                     value = '10000',
                                     min=0),
                            bsTooltip('operating_cost',
                                   "The average annual cost for operation and
maintenance.",
                                   'right',
                                   options = list(container = 'body')),

                            numericInput(inputId = "system_lifetime",
                                     label = "System Lifetime (years):",
                                     width = '200px',
                                     value = '25',
                                     min=0),
                            bsTooltip('system_lifetime',
                                   "The system lifetime in years (researchers usually
assume PSC could last more than 10 years, which is impossible based on current
technique, or there is no way to be cost effective).",
                                   'right',
                                   options = list(container = 'body')),

                            numericInput(inputId = "degradation_rate",
                                     label = "Degradation Rate (%):",
                                     width = '200px',
                                     value = '0.5',
                                     min=0),
                            bsTooltip('degradation_rate',
                                   "The annual module efficiency degradation rate.",
```

```
                        'right',
                        options = list(container = 'body')),

                  numericInput(inputId = "discount_rate",
                        label = "Discount Rate (%):",
                        width = '200px',
                        value = '5',
                        min=0),
                  bsTooltip('discount_rate',
                        "The nominal discount rate.",
                        'right',
                        options = list(container = 'body')),

                  # Sections for whether or not user has energy output
                  radioButtons(inputId = "has_energy_output",
                        label = "Do you have the annual energy output?:",
                        width = "200px",
                        choices = c("Yes", "No")),

                  conditionalPanel(condition = "input.has_energy_output ==
'Yes'",

                              numericInput(inputId = "energy_output",
                                    label = "Annual Energy Output (kWh):",
                                    width = '200px',
                                    value = '18000',
                                    min=0),
                              bsTooltip('energy_output',
                                    "The un-degraded, annual energy output by
the system as electricity, could be calculated by the PCE of PSC.",
                                    'right',
                                    options = list(container = 'body'))
                  ),

                  conditionalPanel(condition = "input.has_energy_output ==
'No'",

                              # More inputs button
                              actionButton(inputId = "more_inputs",
                                    label = "More Inputs",
                                    width = '100px'),
                              helpText("If you do not want to use the default
energy calculation inputs, you must click the 'More Inputs' button and fill in those
inputs before calculating.")

                  ),
```

```
                    bsModal(id = 'modal1', title = 'Energy Inputs', trigger =
"more_inputs", size = 'large',

                         fluidRow(
                          column(4,

                              # Module efficiency input
                              numericInput(inputId = "module_efficiency",
                                     label = "Module Efficiency (%):",
                                     width = '200px',
                                     value = '18',
                                     min=0),
                              bsTooltip('module_efficiency',
                                   "The module efficiency (how much radiation is
converted to energy).",
                                   'right',
                                   options = list(container = 'body')),

                              # Losses input
                              numericInput(inputId = 'losses',
                                     label = 'Losses (%):',
                                     width = '200px',
                                     value = '14',
                                     min=-5,
                                     max=99),
                              bsTooltip('losses',
                                   "Estimated total system losses.",
                                   'right',
                                   options = list(container = 'body'))


                              ),

                          column(4,

                              helpText('For perovskite solar sells, please select
the thin film module type.'),

                              # Array type input
                              radioButtons(inputId = 'array_type',
                                     label = 'Array Type:',
                                     choiceNames = c('Fixed - Open Rack','Fixed
- Roof Mounted','1-Axis','Backtracked','2-Axis'),
                                     choiceValues = c(0,1,2,3,4)),
```

```
                                        # Module type input
                                        radioButtons(inputId = 'module_type',
                                                  label = 'Module Type:',
                                                  choiceNames = c('Standard:
Polycrystalline','Premium: Monocrystalline','Thin film'),
                                                  choiceValues = c(0,1,2))

                                        ),

                                   column(4,



                                        # Array area input
                                        numericInput(inputId = 'array_area',
                                                  label = 'Array Area (m\u00B2):',
                                                  width = '200px',
                                                  value = '40',
                                                  min=0,
                                                  max=100000),
                                        bsTooltip('array_area',
                                                "The total estimate area of the array, inlcuding
all panel faces. It can also be a value of roof or ground area.",
                                                'right',
                                                options = list(container = 'body')),

                                        # Panel tilt input
                                        numericInput(inputId = 'tilt',
                                                  label = 'Panel Tilt (degrees):',
                                                  width = '200px',
                                                  value = '0',
                                                  min=0,
                                                  max=90),
                                        bsTooltip('tilt',
                                                "Degree tilt of the solar panels.",
                                                'right',
                                                options = list(container = 'body')),

                                        # Azimuth input
                                        numericInput(inputId = 'azimuth',
                                                  label = 'Azimuth (degrees):',
                                                  width = '200px',
                                                  value = '180',
                                                  min=0,
                                                  max=360),
                                        bsTooltip('azimuth',
```

```
                                        "Azimuth is the angle that the solar panels are
facing and is measured in a clockwise direction from north.",
                                        'right',
                                        options = list(container = 'body'))

                                  )


                            )




                        ),

                        br(),

                        ), # End of find LCOE conditional panel


              conditionalPanel(condition = "input.calc_type == 'Compare LCOE'",

                        # LCOE input
                        numericInput(inputId = 'lcoe_in',
                                 label = 'LCOE ($/kWh):',
                                 width = '200px',
                                 value = '',
                                 min=0),
                        bsTooltip('lcoe_in',
                               "The estimated levelized cost of electricity of your solar
project.",
                               'right',
                               options = list(container = 'body'))

                        ),

              # LCOE submit button
              actionButton(inputId = "lcoe_button",
                       label = "Calculate",
                       width = '100px',
                       icon("arrows-rotate"))




             ),
             mainPanel(
```

```
fluidRow(
  column(4,
        HTML(paste0("<b>","Utility Company:","</b>")),
        textOutput('utility_name_out'),
        br(),
        br(),
        HTML(paste0("<b>","Commercial Electricty Rate:","</b>")),
        textOutput('commercial_rate_out')),
  column(4,
        HTML(paste0("<b>","Industrial Electricty Rate:","</b>")),
        textOutput('industrial_rate_out'),
        br(),
        br(),
        HTML(paste0("<b>","Residential Electricty Rate:","</b>")),
        textOutput('residential_rate_out')),
  conditionalPanel(condition = "input.calc_type == 'Find LCOE'",
  column(4,
        HTML(paste0("<b>","Levelized Cost of Electricity (LCOE):","</b>")),
        textOutput('lcoe_out')))
),

br(),
br(),
conditionalPanel(condition = "input.has_energy_output == 'No'",
fluidRow(
  column(4,
        HTML(paste0("<b>","Solar Radiation:","</b>")),
        textOutput('solrad_annual_out'),
        br(),
        br(),
        HTML(paste0("<b>","Capacity Factor:","</b>")),
        textOutput('capacity_factor_out')),
  column(4,
        HTML(paste0("<b>","State of Weather Station:","</b>")),
        textOutput('station_state_out'),
        br(),
        br(),
        HTML(paste0("<b>","Elevation of Weather Station:","</b>")),
        textOutput('station_elev_out'))
)

),


conditionalPanel(condition = "input.calc_type == 'Find LCOE'",
```

```
          hr(),
          downloadButton('download_table', 'Download Table'),
          br(),
          br(),
          DT::dataTableOutput("output_table"),
          br(),
          br(),
          plotlyOutput(outputId = "cost_plot", height = 'auto', width = 'auto'),
          br()
          )


        )
      )
    ),



    # Restart session tab
    tabPanel(title = "Restart Session",
         icon = icon("circle-xmark"),
         actionButton("reset_session", "Restart Session")
    )

  )

)

server <- function(input, output, session) {

  observeEvent(input$reset_session, {
    session$reload()
  })


  # Initialize variables for API calls
  api_key <- '9UUX1nAVhZoj90XY9R1QHD4U5foWHVABoQxlbnxt'
  timeframe <- 'hourly'
  dataset <- 'nsrdb'
  interval <- '60'
  email <- 'danielsaul@uga.edu'
  names <- 'tmy-2020'
  use_wf_albedo <- 1
```

```
# Run LCOE simulation
observeEvent(input$lcoe_button, {
  shinyalert("Calculating...", showConfirmButton = FALSE, timer = 0)

  # Convert Zipcode input to latitude and longitude
  if (input$location_type == "Zipcode") {

    codes <- geocode_zip(input$zipcode)

    lat <- codes[[2]]
    lon <- codes[[3]]


  } else {

    lat = input$lat
    lon = input$lon

  }


  # Establish inputs
  initial_investment <- as.numeric(isolate(input$initial_investment))
  operating_cost <- as.numeric(isolate(input$operating_cost))
  system_lifetime <- as.numeric(isolate(input$system_lifetime))
  degradation_rate <- as.numeric(isolate(input$degradation_rate))/100
  discount_rate <- as.numeric(isolate(input$discount_rate))/100

  # If user has energy output number inputs
  if (input$has_energy_output == 'Yes'){

    energy_output <- as.numeric(isolate(input$energy_output))

  } else {

    module_efficiency <- as.numeric(isolate(input$module_efficiency))/100
    tilt <- isolate(input$tilt)
    azimuth <- isolate(input$azimuth)
    array_type <- as.numeric(isolate(input$array_type))
    module_type <- as.numeric(isolate(input$module_type))
    array_area <- as.numeric(isolate(input$array_area))
    losses <- as.numeric(isolate(input$losses))/100

    system_capacity <- array_area * 1 * module_efficiency

    # Run the PVWatts simulation
```

```
      solar_data_api_outputs <-
jsonlite::fromJSON(paste0('https://developer.nrel.gov/api/pvwatts/v8.json?api_key=',a
pi_key,'&lat=',lat,'&lon=',lon,'&system_capacity=',system_capacity,'&azimuth=',azimut
h,'&tilt=',tilt,'&array_type=',array_type,'&module_type=',module_type,'&losses=',losses
,'&timeframe=',timeframe,'&dataset=',dataset,'&use_wf_albedo=',use_wf_albedo))

      # Get annual outputs
      station_elev <- round(solar_data_api_outputs[["station_info"]][["elev"]], 2)
      station_state <- solar_data_api_outputs[["station_info"]][["state"]]
      energy_output <- round(solar_data_api_outputs[["outputs"]][["ac_annual"]], 2)
      solrad_annual <- round(solar_data_api_outputs[["outputs"]][["solrad_annual"]], 2)
      capacity_factor <- round(solar_data_api_outputs[["outputs"]][["capacity_factor"]],
2)

      # Show general api outputs
      output$solrad_annual_out <- renderText({
        paste0(solrad_annual, " kWh/m\u00B2/day")
      })

      output$capacity_factor_out <- renderText({
        paste0(capacity_factor, "%")
      })

      output$station_state_out <- renderText({
        paste0(station_state, ", United States")
      })

      output$station_elev_out <- renderText({
        paste0(station_elev, " meters")
      })

    }


    # # Get the most recent inflation rate data from FRED API
    # fred_url <-
"https://api.stlouisfed.org/fred/series/observations?series_id=CPALTT01USM657N&a
pi_key=5b303891d1c1292239660c62b36805a4&file_type=json&limit=1&sort_order=d
esc"
    # response <- GET(fred_url)
    # inflation_data <-
jsonlite::fromJSON(rawToChar(response$content))$observations
    #
    # # Extract the most recent inflation rate value
    # inflation_rate <- round(as.numeric(inflation_data$value),4)
```

```
  # Electricity/utility rate apis
  utility_outputs <-
jsonlite::fromJSON(paste0('https://developer.nrel.gov/api/utility_rates/v3.json?api_key
=',api_key,'&lat=',lat,'&lon=',lon))
  utility_name <- utility_outputs[["outputs"]][["utility_info"]][["utility_name"]]
  commercial_rate <- utility_outputs[["outputs"]][["commercial"]]
  industrial_rate <- utility_outputs[["outputs"]][["industrial"]]
  residential_rate <- utility_outputs[["outputs"]][["residential"]]

  output$utility_name_out <- renderText({
    paste0(utility_name)
  })

  output$commercial_rate_out <- renderText({
    paste0('$', commercial_rate, "/kWh")
  })

  output$industrial_rate_out <- renderText({
    paste0('$', industrial_rate, "/kWh")
  })

  output$residential_rate_out <- renderText({
    paste0('$', residential_rate, "/kWh")
  })



  # LCOE calculations/outputs
  years <- 1:system_lifetime
  energy_outputs <- energy_output * (1 - as.numeric(degradation_rate))^years
  operating_costs <- operating_cost / (1 + discount_rate)^years

  output_table <- data.frame(year = years, energy_output = energy_outputs,
install_cost = initial_investment, operating_cost = operating_costs, degradation_rate =
degradation_rate, discount_rate = discount_rate)
  output_table <- round(output_table,4)

  overall_lcoe <- round((initial_investment +
sum(output_table$operating_cost))/sum(output_table$energy_output),4)

  colnames(output_table) <- c('Lifetime Year', 'Energy Output (kWh)', 'Installation
Cost ($)', 'Operating Cost ($)', 'Degradation Rate', 'Discount Rate')

  output$lcoe_out <- renderText({
    paste0('$', overall_lcoe, "/kWh")
```

```
  })




  output$output_table <- DT::renderDataTable(output_table,
                                             options = list(autoWidth = TRUE,
                                                            pageLength = 5),
                                             rownames = FALSE)

  # Cost and energy plot
  output$cost_plot <- renderPlotly({

    theme_set(theme_classic())
    p1 <- ggplot(data = output_table, aes(x = `Lifetime Year`, y = `Operating Cost
($)`)) + geom_line() + geom_point()
    p2 <- ggplot(data = output_table, aes(x = `Lifetime Year`, y = `Energy Output
(kWh)`)) + geom_line() + geom_point()
    p <- subplot(p1,p2, titleX = TRUE, titleY = TRUE)

    p <- ggplotly(p, tooltip = "text")

  })



  output$download_table <- downloadHandler(
    filename = function() {
      paste("LCOE-Data-", Sys.Date(), ".xlsx", sep="")
    },
    content = function(file) {


      write.xlsx(output_table, file, col.names = TRUE, row.names = FALSE, showNA =
FALSE)

    }
  )







  shinyalert("Calculation Complete.", showConfirmButton = TRUE, type = "success",
immediate = TRUE)
```

```
  })



} # End of server
```