Laboratory 4 – Solar panel model

Goals & Questions:

- Continue climbing the R learning curve!
- Develop a sense for the structure of programming
- Create a model of a photovoltaic solar panel
- What is the capacity of solar power on campus buildings?

Outline:

As a class:

- Programs using indexing, loops, and logical statements

With your partner(s):

1. Conceptualize a solar panel model, variables and relationships
2. Solar panel model assuming a flat panel
3. Solar panel model that incorporates panel orientation

   Assignment: Rooftop solar power capacity on campus

Today's Context:

By enrolling in ES 220, you have also automatically joined an environmental consulting firm that specializes in quantitative systems modeling!

For this week, we have been hired by the Office of Sustainability at Wellesley College to estimate the potential for rooftop photovoltaic solar power generation on campus. To do this, we'll need to (1) construct a quantitative model of a solar panel and (2) use input values specific to Wellesley College to get an appropriate output.

1. Conceptualize a solar panel model, variables and relationships

Spend a few minutes with your partner and sketch out a "causal diagram" systems model of a solar panel that includes the following five variables (boxes):

Radiant Energy (W/m^2) *Solar Panel Efficiency (%)* *Electric Power (W)*

Individual Panel Area (m^2) *Number of Panels*

Draw the system and include first-order relationships (i.e. positive or negative) between variables. We'll discuss as a class before moving on.

2. Solar panel model assuming a flat panel

We'll start by building a model of the existing solar panel array on campus and compare it to the actual values of electric power produced.

Important Model Assumptions:

- The efficiency of the solar panels is assumed constant at 15 %.
- The panels are assumed to be lying flat on the ground

Begin to create a model of a solar panel through time by thinking of the main **control flow** of the program. We'll be calculating solar power produced for each hour of the day, so our main structural component will be a **loop** that represents time. The general structure of the program will be:

1. Load data and functions
2. View data (always take a look at any data that you'll be using)
3. Define necessary input objects
4. Perform initial calculations
5. Create an empty object to store electric power values (output variable)
6. Setup loop to calculate solar power
7. Convert from watts to kilowatts
8. Plot data after loop is finished
9. Calculate and display the total energy produced over the 24 hours (in kWh)

The calculation of electric power for each hour: **Power = Total Area × Radiation × Efficiency**

The file **Lab 4 Data - Jan 6 2021.RData** contains hourly solar radiation data from January 6, 2021. When loaded it will add several objects to R's environment, each with 24 values that correspond to the hour of the day:

`Hour` indicates the hour of the day (using a 24 hour clock, with zero as midnight)

`Solar.Radiation.Wm2` is solar radiation in W/m^2

`Elec.Observed.W` is the actual power (W) produced by Wellesley's solar panels that day

→ Open the file: Lab 4 Solar Panel Model.Rmd

→ Change the author to your name

The Program (multiple code chunk):

1. **Code Chunk: *Load_Data***
   Load and view the data

```
load("Lab 4 Data - Jan 6 2021.RData")  #loads the data

weplot(x = Hour, y = Solar.Radiation.Wm2, type = "line") #Radiant power
weplot(x = Hour, y = Elec.Observed.W, type = "line") #Actual electrical power
```

2. **Code Chunk: *Initial_IO***
   Define initial input an output values

```
#Input objects
Efficiency <- 0.15  #defines the panel efficiency at 15%
Panel.Area <- 1.41  #area of each panel
Num.Panels <- 48  #number of panels

#Initial calculations (from inputs)
Total.Area <- Panel.Area * Num.Panels  #total panel area
Time <- length(Hour)  #length of the dataset (used to run the loop)

#Output object
Elec.Model.W <- numeric()  #creates an empty object (length of zero)
```

3. **Code Chunk: *Solar_Model_1***
   Setup loop to calculate solar power

```
#Loop
for (t in 1:Time){

      #calculate electricity produced
      Elec.Model.W[t] <- Total.Area * Solar.Radiation.Wm2[t] * Efficiency

}
```

   Convert from watts to kilowatts

```
#Results in kilowatts
Elec.Observed.kW <- Elec.Observed.W / 1000 #watts to kilowatts
Elec.Model.kW <- Elec.Model.W / 1000 #watts to kilowatts
```

4. **Code Chunk: *Plot_Results***
   Compare modeled and observed power

```
#Overlay observed and modeled power
weplot(x = Hour, y = list(Elec.Observed.kW, Elec.Model.kW),
       type = "line",
       xlab = "Hour of the day",
       ylab = "Electrical power (kW)")
```

   Note that within the parentheses of a "function" such as *weplot*, you can separate your code onto multiple lines to make it easier to read.

How does our model compare with the actual electric power produced on 1/6/2021?

5. **Code Chunk: *Total_Energy***
   Calculate and display the total energy produced over the 24 hours (kWh)

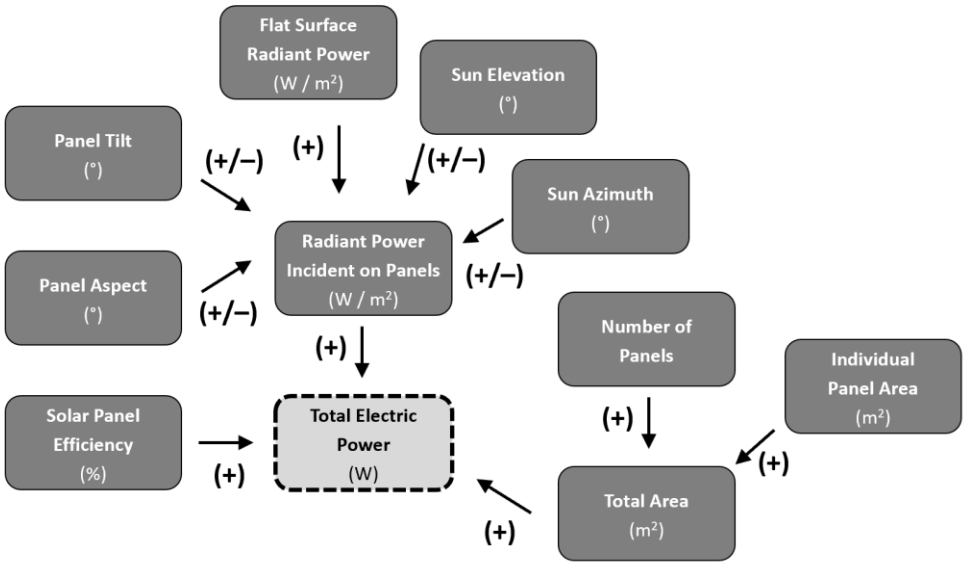
```
Elec.Observed.kWh <- mean(Elec.Observed.kW)*Time #total energy in kWh
Elec.Model.kWh <- mean(Elec.Model.kW)*Time #total energy in kWh

print(Elec.Observed.kWh)
print(Elec.Model.kWh)
```

   The calculation for kilowatt-hours above calculates the *average flow* of electrical power and multiplies by the number of hours over which this power was flowing.

How does your model compare with the actual solar electricity produced?

3. Solar panel model with panel and sun angles

As you can imagine, it makes a big difference whether the solar panels are actually facing toward or away from the sun! The incorporate this, we'll expand our systems diagram of our model by incorporating the orientation of the panels and the position of the sun in the sky in order to figure out the actual radiant power that is falling directly ("incident") onto the solar panels:



This more complicated model introduces four new variables (and R objects):

- Tilt of the solar panels (from horizontal to vertical)
- Aspect of the solar panels (i.e. north, south, east, west)
- Elevation of the sun in the sky (from below the horizon to directly overhead)
- Azimuth of the sun in the sky (i.e. north, south, east west)

We'll calculate these values using three custom *functions* that spare you from having to do a series of pretty ugly calculations:

> The function **sun_elevation** will return the elevation of the sun in the sky based on the time of day and the day of year. It assumes that we are located in Wellesley.
>
> < 0° the sun is below the horizon
> 0° sunset, sunrise
> 90° the sun is directly overhead

Similarly, the function **sun_azimuth** will return the azimuth (direction) of the sun in the sky based on the time of day and the day of year.

0°	North
90°	East
180°	South
270°	West

The function **get_incident_rad** will return the amount of radiation directly falling ("incident") on the panels (W/m^2) based on the tilt of the panel (from horizontal), the aspect of the panel (degrees from north) and the solar elevation and azimuth values provided by the functions above.

→ Copy your code from the *Solar_Model_1* code chunk into the *Solar_Model_2* code chunk

→ Add the following code above your loop:

```
source("solar panel functions.R") #loads functions for calculating angles
```

→ Within your loop you'll need to calculate the position of the sun in the sky and the solar incident solar radiation on the panels. Be sure to read through this next section before adding lines of code.

The *sun_elevation* and *sun_azimuth* functions:

Usage – These functions requires two inputs:

1. Day of the year
2. Hour of the day

```
sun_elevation(Day of the year, Hour of the day)
sun_elevation(Day of the year, Hour of the day)
```

In our case, January 6 is the 6th day of the year, and the hour of the day can be found by the values within the object **Hour**. The following lines of code will calculate the elevation and azimuth of the sun:

```
Elev <- sun_elevation(6, Hour[t]) #calculates sun elevation
Azi <- sun_azimuth(6, Hour[t]) #calculates sun azimuth
```

The *get_incident_rad* function:

Usage – The *get_incident_rad* function requires five inputs:

1. Solar radiation in W/m^2 on a flat panel
2. Panel tilt angle in degrees from horizontal
3. Panel aspect (direction) in degrees
4. Elevation of the sun in degrees
5. Azimuth (direction) of the sun in degrees

Output – A single value of solar radiation perpendicular to panel in W/m^2

We already have solar radiation on a flat panel from the object `Solar.Radiation`. The campus solar panels have a tilt of 25° and an aspect of 155° (Southeast). The values for solar elevation (`Elev`) and azimuth (`Azi`) were created by the `sun_elevation` and `sun_azimuth` functions. We can enter all of this into the `get_incident_rad` function with the following line of code:

```
#calculates radiation falling onto the panel
Incident.Solar.Radiation.Wm2[t] <- get_incident_rad(Solar.Radiation.Wm2[t],
                                                    Tilt, Aspect,
                                                    Elev, Azi)
```

Note that above the loop you'll need to 1) create an empty object for `Incident.Solar.Radiation.Wm2`, and 2) create the objects `Tilt` and `Aspect` and assign them their corresponding values.

→ Modify the code in *Solar_Model_2* so that the values contained in the object *Incident.Solar.Radiation.Wm2* are incorporated into the calculation of electrical power .

- How does your new model compare to the previous mode that assumed flat solar panels?

- Note that you can run existing code chunks by adding `<<  >>` around the chunk name:

```
#plots results using current object values
<<Plot_Results>>

#total energy from current object values
<<Total_Energy>>
```