Laboratory 3 – Wind power

Goals & Questions:

- Continue to develop familiarity with R
- Develop a model of a wind turbine
- Incorporate realities/complexities of wind power into models
- Assess the wind power capacity and variability of the Block Island Wind Farm and for a location of your choosing

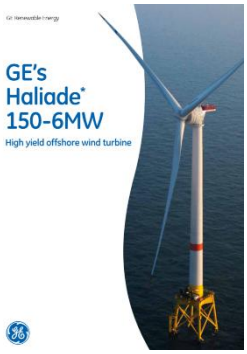
Lab Outline:

1. Import and visualize wind data
2. A "simple" wind turbine model with fixed efficiency
3. Correcting for variable efficiency
4. Correcting for low and high wind speed limitations

You'll work with your partner to produce a model of a wind turbine, which will estimate power production based on wind speed. Similar to last week, the model will start off relatively simple and then you'll add in more complexity/reality.

We'll be specifically modeling a GE Haliade 150 wind turbine.

Blade radius:	75.5 m
Tower height:	100 m
"Cut-in" wind speed:	3 m/s
"Cut-out" wind speed:	25 m/s
Maximum power rating:	6,000 kW
Efficiency:	Varies with wind speed



Input Data:

Block Island.Rdata – contains three objects that we'll use to examine the Block Island Wind Farm. Each spans the entire year of 2013 and contains 6-hour values (4 times a day).

`Velocity.Wind`	6-hour average wind velocity at 100 m above the ground/ocean
`Date`	Object containing date/time values

Import and visualize wind data

→ **Open: "Lab 3 - Template.Rmd" and save with a new name: "Lab 3 - Model 1.Rmd"**

- Load *Block Island.Rdata*: `load("Block Island.RData")`
- Load the *weplot* function: `source("weplot function.R")`

First, take a look at the data to get a sense of what we're working with.

→ **Make a plot that shows average 6-hour wind speeds off Block Island in 2013.**

```
weplot(x = Date, y = Velocity.Wind, type = "line")
```

- Modify the above code to label your y axis

```
weplot(x = Date, y = Velocity.Wind, type = "line", ylab = "__")
```

- Try changing the type from **"line"** to "area" to fill in the area under the curve.
- Change the color by including a color argument in the weplot function:

```
weplot(..., color = "orange")
```

→ **Choose a time period of approximately two weeks to get a zoomed in view of the data.** (Two weeks is about 50 6-hour observations)

```
timespan <- 50:100  #Index values between the 12th and 24th days
weplot(x = Date[timespan], y = Velocity.Wind[timespan], ...)
```

Model 1 – A "simple" wind turbine model

Work with your partner to develop the general structure of code that will calculate wind power based on wind speeds through time. (Read the rest of this page before actually writing any code.)

- Remember that the power of wind (in watts) is determined by the kinetic energy of moving air:

$$Power = \frac{1}{2}\rho A v^3$$

  Where ρ is the density of air (kg/m^3), A is the area that the turbine blades rotate through (m^2), and v is the velocity (m/s).

- Electric power produced by the turbine (in watts) can then be calculated by multiplying the power of the wind by the efficiency of converting wind power to electric power.

The goal of this program is very similar to that of the solar panel model, and we can again make use of a **loop**. This is because we want to do something over and over again – in this case, calculate wind power for 6-hour period. (But also because we need to build up experience working with loops!)

1. **Define Initial Objects**
   - Which objects do you need to define and assign values to? (*Inputs*)
     - Some may simply be a number, while others may require an equation.
   - Which objects do you need to define as empty, to be filled in during the loop? (*Outputs*)
2. **Loop**
   - How long should we make the loop?
   - What needs to be calculated within the loop?

→ **Write the code in R for a program that calculates wind power captured based on average wind speed.**

Assume: Efficiency = 0.20 Air Density = 1.2 kg / m^3

- Refer to last week's lab as a template. The overall larger structure of the model will be very similar, but the object names and calculations will be different.
- It's probably best to divide your electric power values by 1 million (1e6) in order to work in units of megawatts (MW). (This can be done after the loop)

→ **After the loop, calculate the <u>average power</u> and <u>total energy</u> that your model predicts for 2013 from one Haliade 150 turbine off Block Island.**

- The easiest way to calculate <u>total energy</u> is to multiply the average power by the number of hours over which it is produced, i.e. number of hours in a year:
    - You can use the `mean` function to calculate the average power

→ **Make plots showing how the electric power varies through time (`x = Date`) and with wind speed (`x = Velocity.Wind`).** Does this relationship make sense to you given the way you have specified the model?
    - Probably best to use `type = "point"` when examining the relationship between electric power and wind velocity.

→ **What is the maximum amount of power that your turbine is able to produce?**

- You can use the `max` function on your vector of power values to find this out.
- How does this line up with the maximum power rating for the Haliade 150 turbine? (first page of this handout)

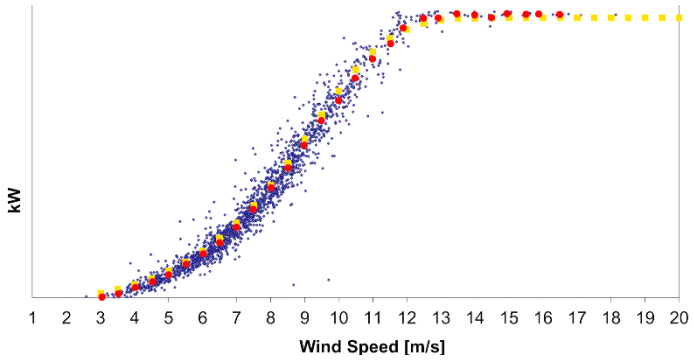
<u>Model 2 – Correcting for variable efficiency</u>

→ **Before moving on…**

1. **Save your current file**
2. **Save to a new file (save as): "Lab 3 - Model 2.Rmd")**

We've assumed an average efficiency ("coefficient of power") of 0.20, which is about average for this wind turbine. However, at high wind speeds this value is too high and can greatly overestimate the amount of electric power produced. At low wind speeds, turbines attempt to maximize their efficiency in order to generate as much electrical power as possible. However, at higher wind speeds, turbines need to actually *reduce* their efficiency so as to not exceed their maximum (*and safe*) power limit. Essentially, at high wind speeds there can be way too much wind power for any turbine to actually capture.

Instead of calculating how efficiency changes with wind speed, we can actually use a *statistical model* that directly describes the relationship between wind speed and the electric power produced by the turbine. The figure on the right shows the actual 'power curve' from the testing and certification process of the Haliade 150 turbine. Note that the turbine will achieve its maximum power (6 MW) at a wind speed of around 13 m/s.



The model has the following equation:

$$Electric\ Power\ (W) = MaxPower\ \times\ \frac{e^{(a\ +\ b \times WindSpeed)}}{1 + e^{(a\ +\ b \times WindSpeed)}}$$

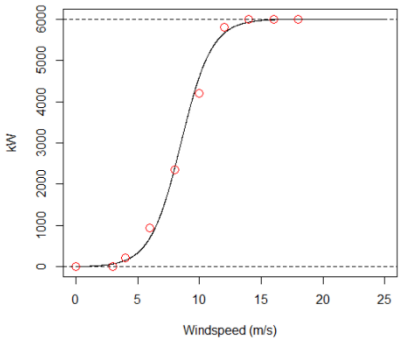
Note that efficiency is no longer directly incorporated into the calculation of electric power

- We are representing the electric power produced using a *statistical model* built on observed relationships.

For the Haliade 150 turbine the 'best fit' parameters are:

***MaxPower* =**	**6 million (watts)**
***a* =**	**-6.8409**
***b* =**	**0.8043**

The figure on the right shows the fitted model (black line) compared to a subset of the actual data (red circles).



→ **Modify your model to incorporate variable efficiency based on windspeed.**

- Note that the **`exp`** function can be used to raise *e* to a power:
  - **`exp(3)`** is the same as e^3

→ **Check the maximum amount of electric power that is produced to see if it is more characteristic of the Haliade 150 turbine.**

Model 3 – Correcting for low and high wind speeds

→ **Before moving on…**

1. **Save your current file**
2. **Save to a new file (save as): "Lab 3 - Model 3.Rmd")**

Your current model assumes that the turbine will capture power at any wind speed, but in reality this is not the case. Below a certain wind speed ("cut-in speed") most turbines simply won't turn at all. Above a high wind speed ("cut-out speed"), turbines are shut down to prevent danger by tilting the blades and applying brakes to stop movement.

Cut-in wind velocity = 3 m/s **Cut-out wind velocity** = 25 m/s

→ **Modify your code to include the fact that sometimes power is not being captured if the wind speed is too low or too high.**

To do this, you'll need to incorporate what is known as a *logical statement*. This allows the code to take different actions depending on the value(s) of one or more objects. In other words: **IF** *this*, **THEN** *that*…

The structure of a logical statement in R looks a lot like a loop, but typically uses the `if` function. For example:

```
if (x > 10){
  y <- 45
}
```

The code above will assign a value of 45 to the object `y`, but *only if* the value of `x` is greater than 10. If the value of `x` is 10 or less, the code within the brackets `{ }` will be completely ignored.

The most straightforward way to incorporate the cut-in and cut-out velocities is to allow your existing code to calculate what the electric power *would* have been, and then use two logical statements to assign a value of zero to electric power if the wind velocity is too low or too high.

(continued…)

→ **Examine the *variation* in the power output of your turbine.**

- Create a histogram that shows the distribution of power values:

```
weplot(x = Power.Turbine, type = "hist")
```

- Calculate the percentage of time that the turbine is producing zero or maximum power:

```
mean(Power.Turbine == 0)*100
mean(Power.Turbine > 5.9)*100
```

  - This code assumes that your power values are in MW.
  - It's best to use 5.9 as maximum power, as this model will *approach* 6 MW, but not get there exactly. For our purposes we can assume that anything about 5.9 MW is essentially maximum power.
  - The "double equals sign" indicates that you are *asking a question* about equality, not setting something equal to a value. For example, `Power.Turbine == 0` asks R which of the values within Power.Turbine are equal to zero. This will return an object containing TRUE or FALSE values that correspond to the values in Power.Turbine. TRUE/FALSE values are also interpreted as numerical values of FALSE = 0 and TRUE = 1. Thus, taking the average will provide the fraction of values that are TRUE, e.g. if half the values are TRUE, the result will be 0.5.