Solar Panels at NAIT

Alexander Ondrus

07/03/2020

Loading the Data

First, I load the `tidyverse` library and read the downloaded CSV file using the `read_csv()` command. As long as the csv is stored in the same directory as this file, you can use tab-autocomplete to find the file quickly.

The notation `<-` to store the result of a command in another variable can be used interchangably with `=`.

Data is stored internally as a *data frame*. Click on the view icon on the right side of the 'Environment' pane to see the first few rows. Note that R automatically recognized the date column as dates and the time column as times, and so we can perform calculations on them and integrate the results in the text.

To access columns in a data frame we use the `$` notation, and we can perform functions such as `max()`, `min()`, `mean()`, etc. on the column in a single command.

To use these results integrated into the text, we use single tic-marks as follows; the data begins on 2012-05-23 and ends on 2014-08-26. When we knit the document the variable names are replaced by the results of the calculations. This is useful when generating reports that have external data references. The text then automatically updates with the data source.

Data Wrangling - Combining, Selecting, Pivoting, Filtering, Aggregating

Combining

Notice that the date and time is currently contained in two separate columns. The `lubridate` package contains tools to easily combine our date and time columns into a single 'date-time' object, which we can then use for our x-axis in a plot later on.

To add the two together using `lubridate`, I literally add the two columns (first using the functions `ymd()` and `hms()` to have the two columns recognized as year-month-day and hour-minute-second, respectively)

Selecting

For simplicity, I am going to look at only the columns that deal with power. I can pick out specific columns using the `select()` function. Note that all of the columns that I am interested in (excluding the `DATE_TIME` column) end in `POWER`. The `tidyselect` package (part of the `tidyverse` that we already loaded) has convenient function for just this occasion: `ends_with()`.

Pivoting

R works best with tidy data. In our case, the data is not "tidy" because the column names contain some of the data (the angles of the various solar panels).

To tidy the data, I want to gather the data in the column headers in its own column. This is done by `gather`ing the columns into more rows using the `gather()` function. This is sometimes known as *long* format, because it increases the number of rows while reducing the number of columns.

I am going to clean up the `Angle` column by removing the characters "_" and "POWER" from each of the entries. This can be done more elegantly using regular expressions, but for simplicity I just do a direct character replacement.

Filtering

Notice that there are 6912 entries in the `Power` column that have values of `NA`. This is R's way of indicating that a value is missing. I don't want to include these values in my plots below, so I am going to remove them using the `filter()` command. Similar to Python, the exclamation point (!) is used to indicate 'not'.

Aggregating

There are too many points to make a meaningful plot of all of the data, so I want to take the maximum power values on each day. To do this I will use `group_by()` to set the levels that I want R to take for my groups, then I will use the `summarise()` function to summarise over each of those groups. I connect the two functions with the *pipe* operator `%>%`, which takes the output of the first and feeds it to the second.

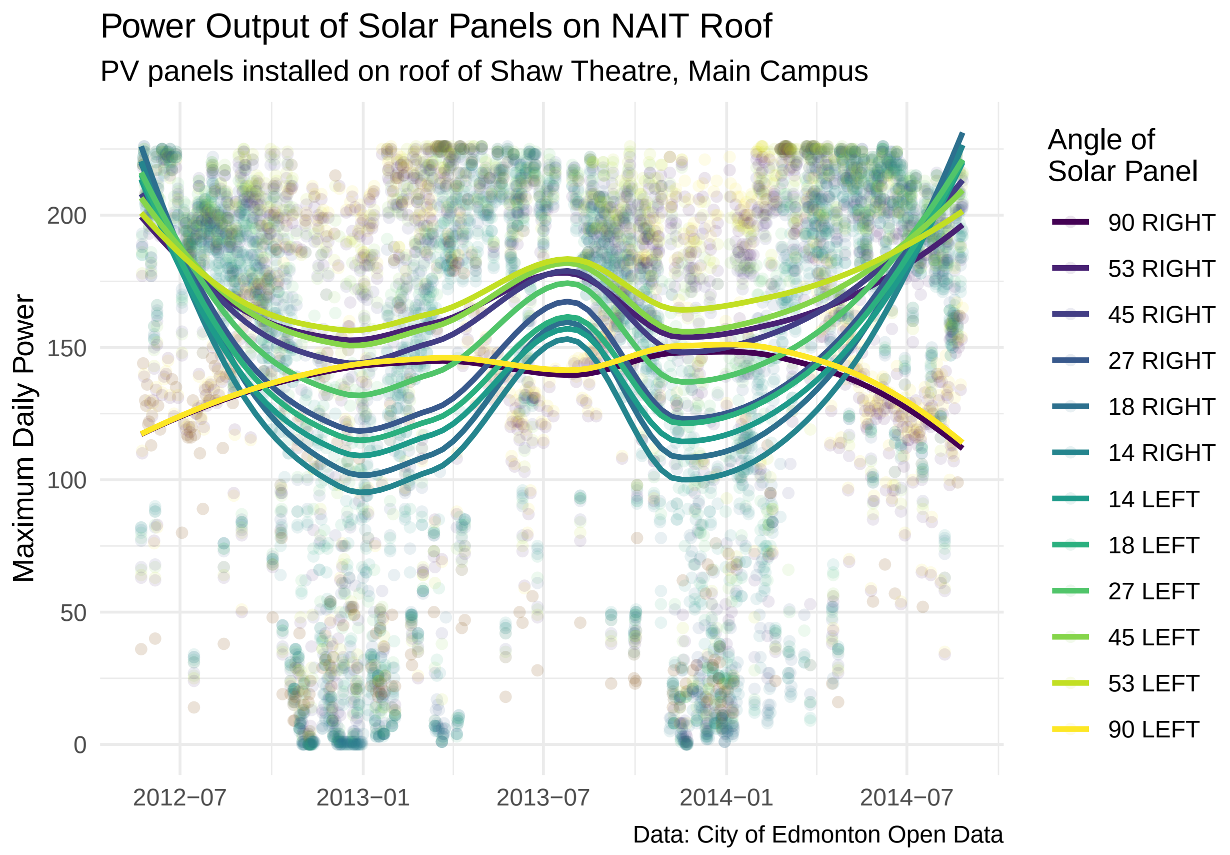
Visualizing

Question 1 - How does power depend on the date?

The first way that I want to visualize the data is to plot the power on the y-axis, the date on the x-axis, and colour the points by angle. I will generate this (and all of the other plots today) using the `ggplot2` package. The `ggplot2` package is part of the `tidyverse` and so it has already been loaded. The basics of how to generate plots in `ggplot2` are:

1. Plots are generated by starting with a `ggplot()` command which contains the data source and the `aes`thetics (mappings from data to visual elements) that will be used on all subsequent layers (by default, but this can be changed).
2. Layers are added to the plot by using the `+` symbol.
3. Plots are stored to a variable and then plotted using the `plot()` command or saved to an external file using the `ggsave()` command.

In my case, the data is `power_daily` and the `aes`thetics I specify are `x = DATE`, `y = Max_Power`, and `colour = Angle`. The layers we will need are `geom_point()` (for points), `geom_smooth()` (to add smoothing curves), and `labs()` for labels.



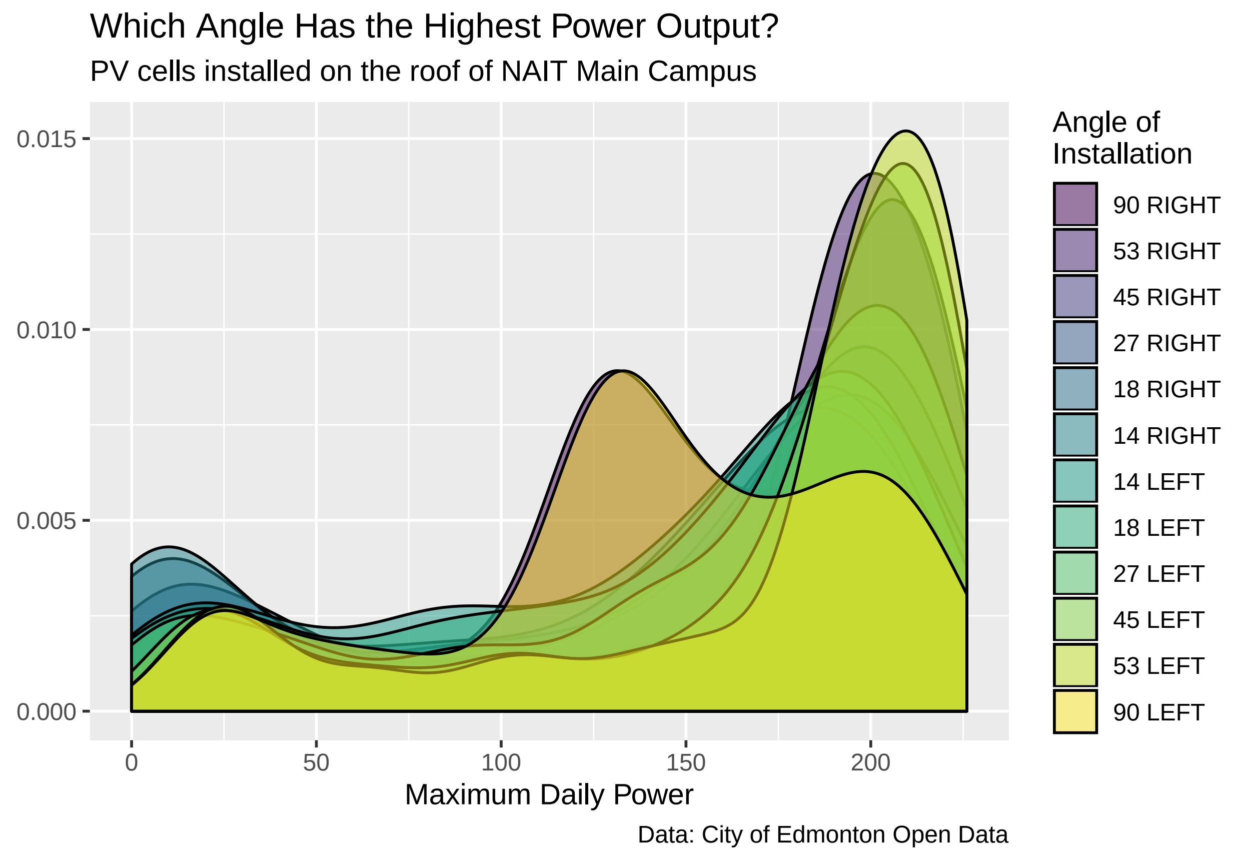
This broadly answers the first question. The two trends that I notice are:

- The panels that are *not* angled at 90^0 have peaks in the summer and troughs in the winter
- The panels that *are* angled at 90^0 have the opposite trend

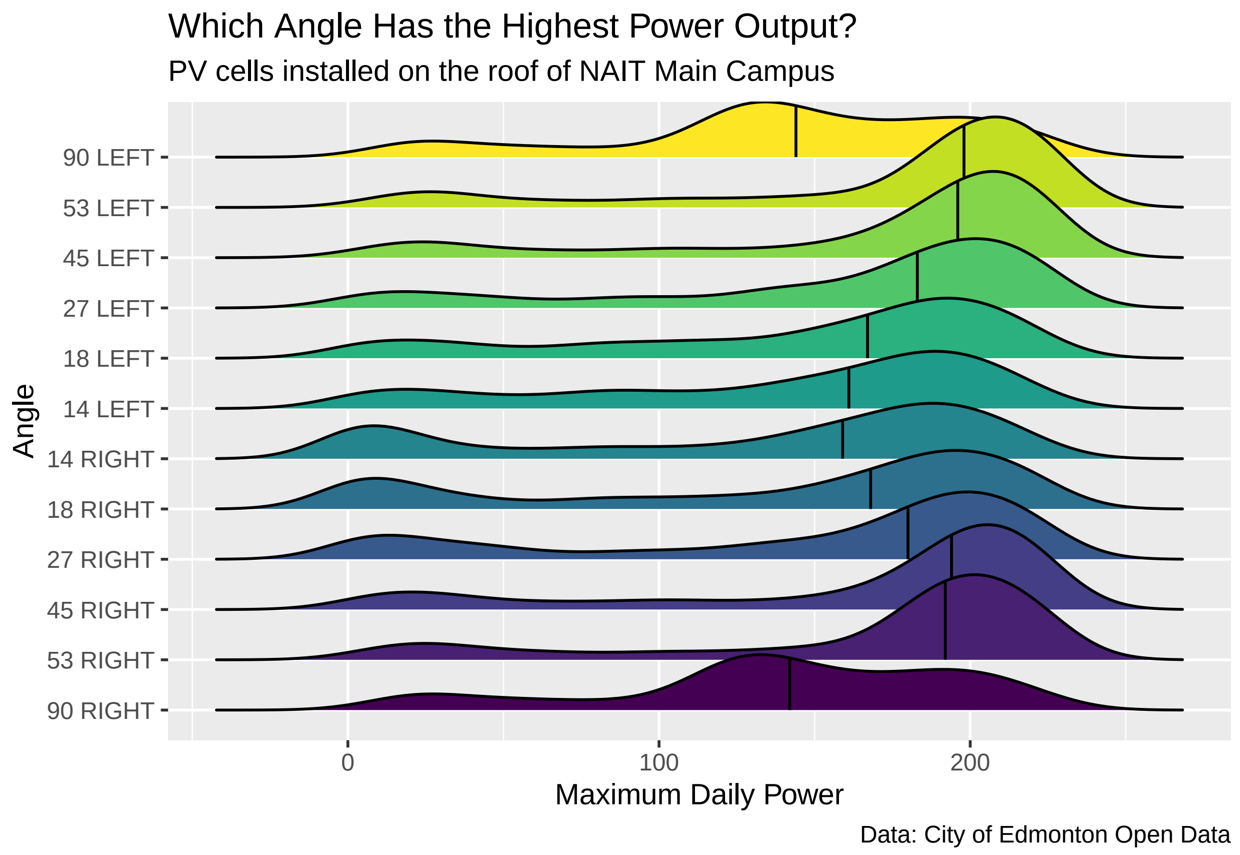
Question 2 - Which angle produces the best power?

I am not interested in the power production for a specific day, but the *distribution* of the power production for the cells installed at different angles. This means that my `x` aesthetic will be the max power, and I will choose my `fill` aesthetic to be my angle so that I can differentiate between them. I'll make the base plot and add some labels, storing the object as `q`.

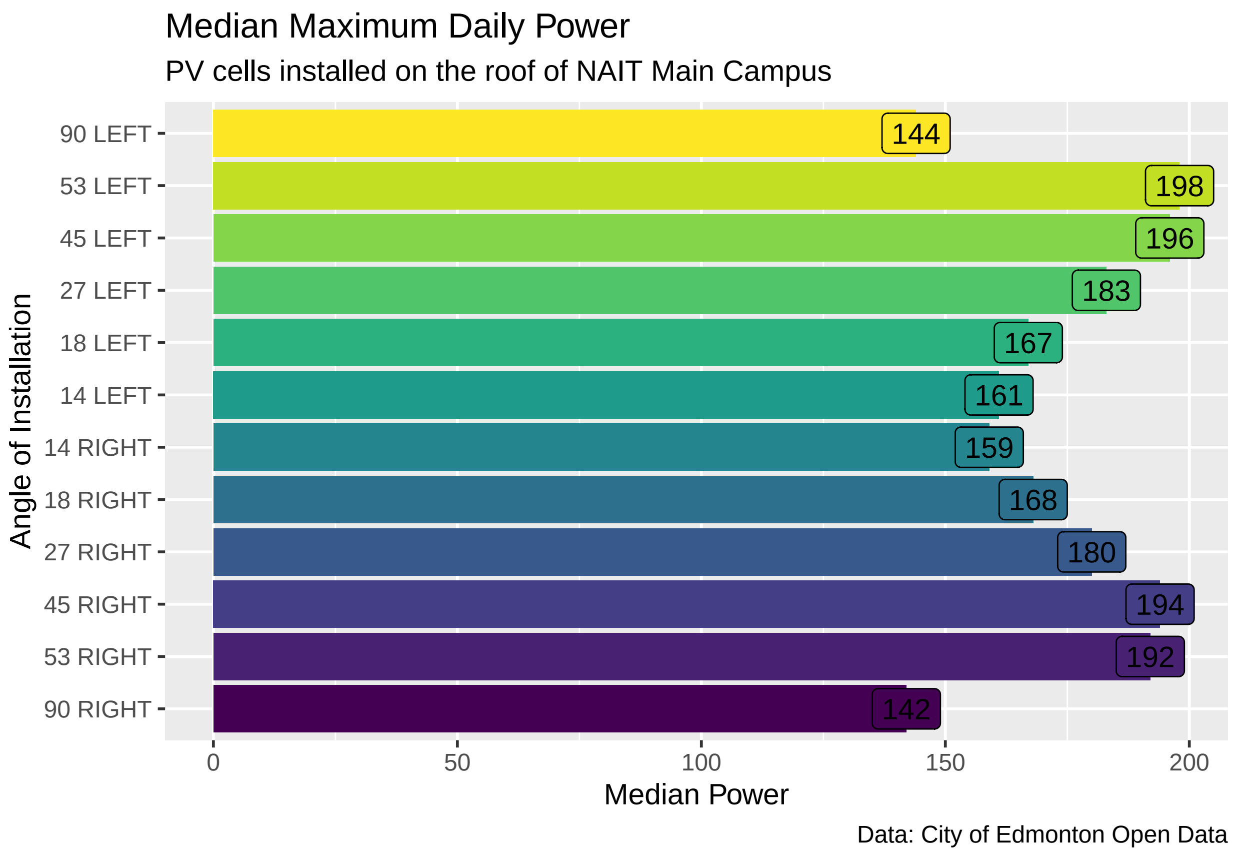
The first way that I will try and get a sense for the distribution is to use the `geom_density` to build *probability density distributions* for each angle.



Even using the opacity to account for the overlap, it is difficult to distinguish the curves. In order to make that easier, I will generate a ridge plot. Notice that the order of the angles has already been set using the `factor()` command in the last plot. This requires us to load (after installing) the `ggridges` package.



It looks like 53^0 left is the winner! Just to double-check, I will calculate the median values for each of the distributions and make a simple bar graph.


Median Maximum Daily Power
PV cells installed on the roof of NAIT Main Campus
Angle of Installation
90 LEFT
53 LEFT
45 LEFT
27 LEFT
18 LEFT
14 LEFT
14 RIGHT
18 RIGHT
27 RIGHT
45 RIGHT
53 RIGHT
90 RIGHT
144
198
196
183
167
161
159
168
180
194
192
142
0
50
100
150
200
Median Power
Data: City of Edmonton Open Data