Solar Plant Diagnostics in R

This analysis is a case study for the last course in the Google Data Analytics Certificate

The purpose of this notebook is twofold:

1. to conduct functional analysis on each inverter in the plant to visualize trends in performance and identify any under-performing equipment
2. to predict future power generation in the short-term

Overview of solar plants

First, it will be important to establish some context and explain what metrics will be used in this analysis.

How do solar plants generate energy?

- Solar panels receive photons from sunlight, and in turn produce DC (Direct Current) power.
- An inverter then converts this DC to AC (Alternating Current) power to then be used for various purposes.

In this analysis we will be using DC and AC power metrics that provide the amount of power (kW) generated per 15 minute interval.

We will also be looking at the daily yield of each inverter, as well as the cumulative total yield of each inverter over a 34 day period.

First, we will install the necessary packages to perform the analysis:

```
install.packages('tidyverse')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages('lubridate')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages('skimr')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages('janitor')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

```
install.packages('chron')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.1'
## (as 'lib' is unspecified)
```

Now Load the libraries we just installed:

```
library(tidyverse)  #helps wrangle data
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)  #helps wrangle date attributes
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)  #helps visualize data
library(skimr)  #summary statistics
library(janitor)  #helps with cleaning data
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(chron)  #helps deal with dates
```

```
##
## Attaching package: 'chron'

## The following objects are masked from 'package:lubridate':
##
##     days, hours, minutes, seconds, years
```

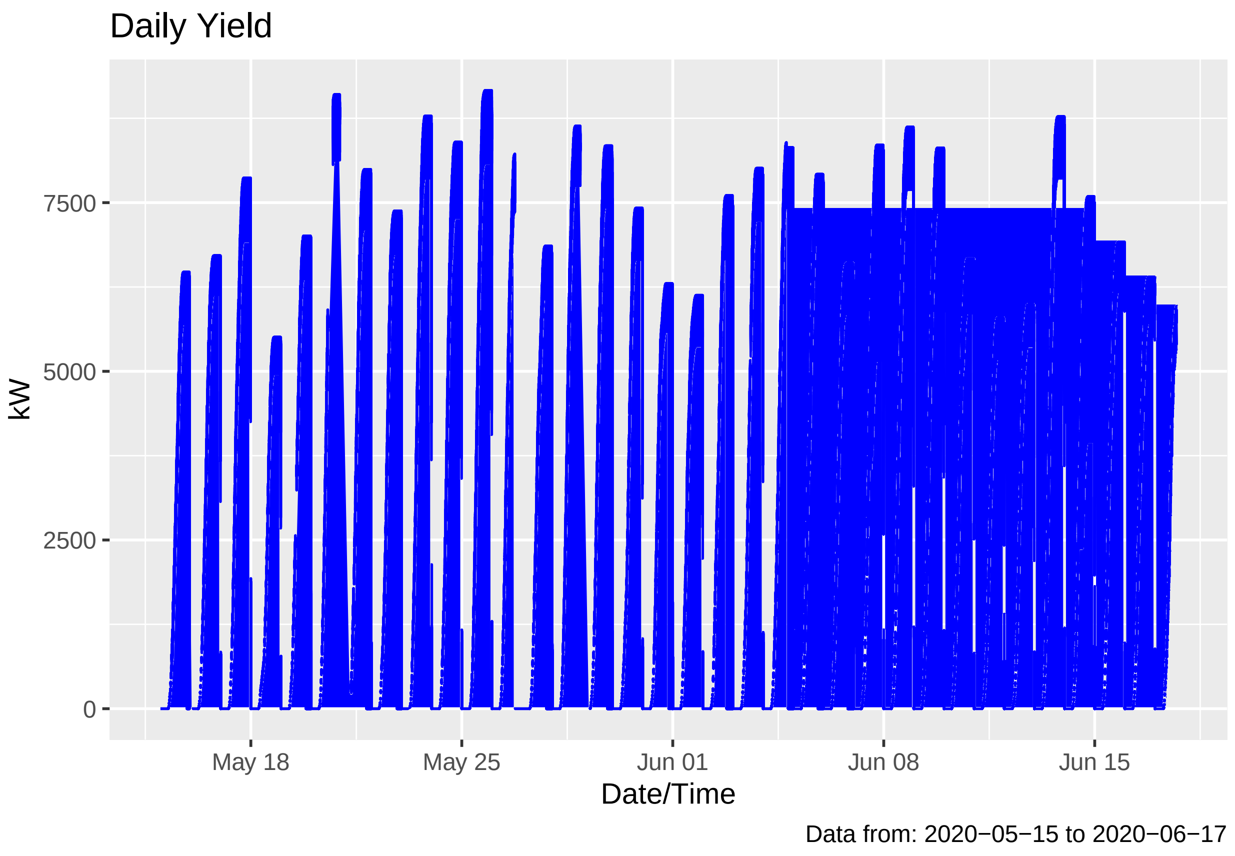
```
# Load data set
plant0 <- read_csv('Plant_1_Generation_Data.csv')
```

```
## Rows: 68175 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): DATE_TIME, SOURCE_KEY
## dbl (5): PLANT_ID, DC_POWER, AC_POWER, DAILY_YIELD, TOTAL_YIELD

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Make copy of data frame with unique and lowercase column names
plant1 <- plant0 %>%
  rename_with(tolower)%>%
  clean_names()

# Change 'date_time' from 'chr' to 'dttm' data type
plant1 <- plant1 %>%
  mutate(date_time = dmy_hm(date_time))

# Create new variables 'date' and 'time' from existing variable 'date_time'
plant1 <- plant1 %>%
  mutate(date = as_date(date_time))%>%
  mutate(time = format(date_time, format = "%H:%M:%S"))
```

```r
# Change 'time' variable to type 'chron' to make it continuous
plant1 <- plant1 %>%
  mutate(time = chron(times(plant1$time)))

# Calculate mean DC production for each inverter by time of day
# and save as new variable
plant1 <- plant1 %>%
  group_by(time, source_key)%>%
  mutate(mean_dc_power = mean(dc_power))

mindate <- min(plant1$date)
maxdate <- max(plant1$date)
```

```r
# Visualize daily yield for entire date range
ggplot(plant1, aes(date_time, daily_yield))+
  geom_line(color="blue")+
  labs(title="Daily Yield",
       caption=paste0("Data from: ", mindate, " to ", maxdate),
       x="Date/Time",
       y="kW")
```


Daily Yield
kW
May 18
May 25
Jun 01
Jun 08
Jun 15
Date/Time
Data from: 2020−05−15 to 2020−06−17