Smart City R Analysis

```r
# install.packages('pacman')
setwd("~/Spring 2021/hackathon/EDA")
# read in weather, electric, and water data
weather <- read.csv("2017-2020_weather_data.csv", na.strings = "NULL")
# electric usage
electric <- read.csv("elec_final_data_hackathon.csv", na.strings = "NULL")
# water usage
water <- read.csv("water_final_data_hackathon.csv", na.strings = "NULL")
# View(head(df1)); View(head(df2)); View(head(df3))

# weather analysis
table(latitude = weather$latitude, longitude = weather$longitude)
```

```
##            longitude
## latitude    -117.151885 -105.256111 -97.699662 -76.498564
##   30.292432           0           0      35427          0
##   32.778033       35004           0          0          0
##   40.027278           0       34990          0          0
##   42.421658           0           0          0       9480
```

```r
# the only one of these locations that is in Austin is 30.2, -97.7
# drop all of the weather locations that are not in austin
weather.atx <- weather[which(weather$latitude == 30.292432), ]

# convert date to type 'POXITct'
weather.atx$date.time <- as.POSIXct(weather.atx$localhour)
electric$date.time <- as.POSIXct(electric$hourly_time)
water$date.time <- as.POSIXct(water$hourly_time)

# remove non-utilized variables from weather factors/ logicals
# cannot be aggregated (remove)
c(sum(is.na(weather.atx$ozone)), sum(is.na(weather.atx$irradiance)))/nrow(weather.atx)
```

```
## [1] 0.9327631 1.0000000
```

```r
# ozone and irradiance are largely NA (remove)
(weather.names <- which(colnames(weather.atx) %in% setdiff(colnames(weather.atx),
    c("localhour", "latitude", "longitude", "tz_offset", "summary",
        "precip_type", "location", "irradiance", "ozone")) & sapply(weather.atx,
    class) != "logical"))
```

```
##          temperature            dew_point             humidity
##                    8                   10                   12
##           visibility apparent_temperature             pressure
```

```
##                  14                   16                   18
##          wind_speed          cloud_cover         wind_bearing
##                  20                   22                   24
##     precip_intensity   precip_probability            date.time
##                  25                   27                   31
```

```
weather.atx1 <- weather.atx[, weather.names]
# which variables have been removed from 'weather.atx'?
colnames(weather.atx)[!colnames(weather.atx) %in% colnames(weather.atx1)]
```

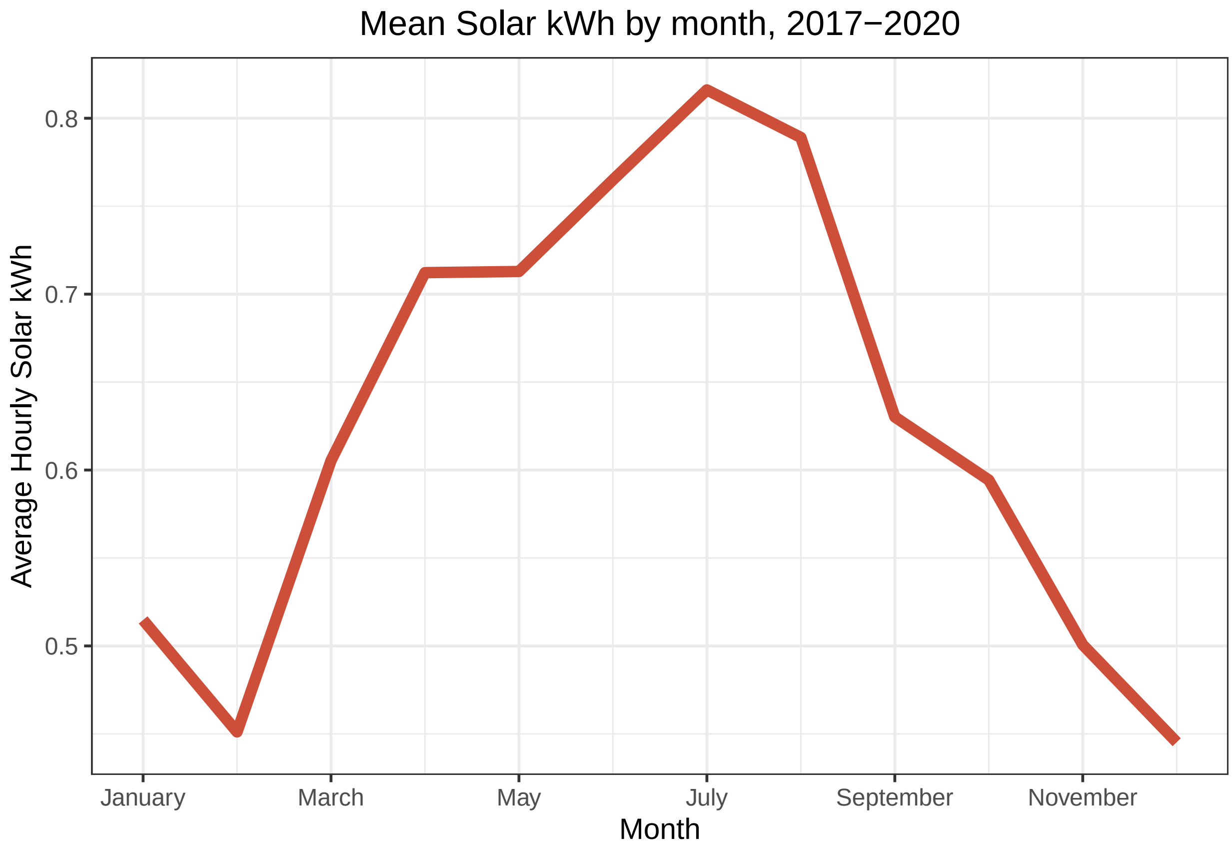
```
##  [1] "localhour"                  "latitude"
##  [3] "longitude"                  "tz_offset"
##  [5] "summary"                    "ozone"
##  [7] "ozone_error"                "temperature_error"
##  [9] "dew_point_error"            "humidity_error"
## [11] "visibility_error"           "apparent_temperature_error"
## [13] "pressure_error"             "wind_speed_error"
## [15] "cloud_cover_error"          "precip_intensity_error"
## [17] "precip_type"                "irradiance"
## [19] "location"
```

```
# remove 'hourly_time' from electric and water
electric1 <- electric[, setdiff(colnames(electric), "hourly_time")]
water1 <- water[, setdiff(colnames(water), "hourly_time")]

# collapse electric and water dataframes across households
# aggregate by mean for each numeric variable
electric.agg <- aggregate(electric1, list(electric1$date.time), mean)[-1]
water.agg <- aggregate(water1, list(water1$date.time), mean)[-1]
# drop home ID
electric.agg1 <- electric.agg[, setdiff(names(electric.agg), "home_id")]
water.agg1 <- water.agg[, setdiff(names(water.agg), "home_id")]

# merge all three dataframes on date.time
df <- merge(merge(weather.atx1, electric.agg1, "date.time"), water.agg1,
    "date.time")

# get month and time of today
df$month <- as.POSIXlt(df$date.time)$mon + 1
df$hour <- as.POSIXlt(df$date.time)$hour

# temperature analysis
df.temp0 <- df[, c("date.time", "temperature")]
df.temp <- aggregate(df.temp0, list(as.POSIXlt(df.temp0$date.time)$mon),
    mean)
names(df.temp)[1] <- "month"
df.temp1 <- df.temp[, -2]

# solar kWh by month

# spread
df.days.sd <- aggregate(df, list(as.POSIXlt(df$date.time)$mon), sd)$hourly_solar_kWh
```

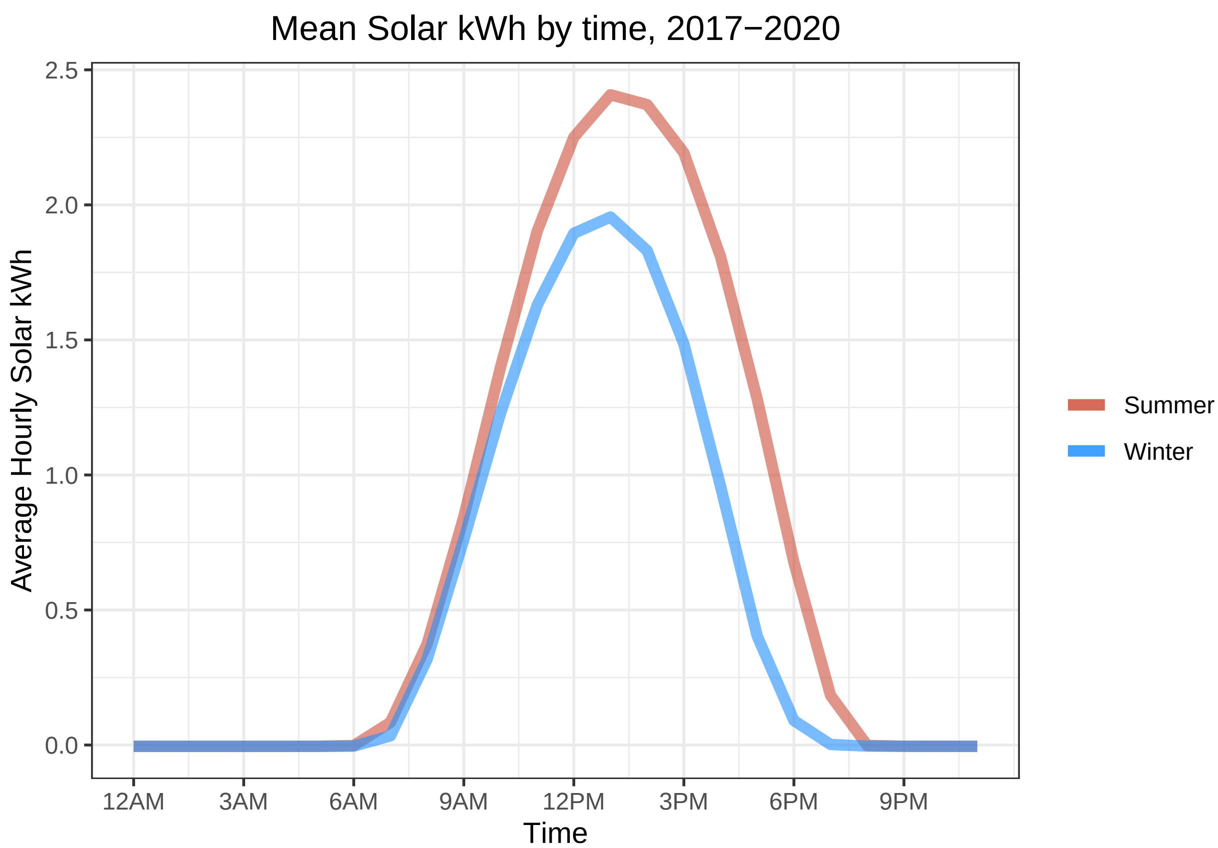
```r
# center
df.days <- aggregate(df, list(as.POSIXlt(df$date.time)$mon), mean)
df.days2 <- df.days[order(df.days$Group.1), ]

df.days2$hourly_solar_kWh
```

```
##  [1] 0.5147800 0.4511100 0.6053597 0.7122225 0.7128844 0.7649739 0.8161253
##  [8] 0.7890889 0.6301972 0.5943417 0.5006943 0.4452512
```

```r
pacman::p_load(ggplot2)

g1 <- ggplot(df.days2, aes(Group.1, hourly_solar_kWh)) + geom_line(color = "tomato3",
    size = 2) + xlab("Month") + ylab("Average Hourly Solar kWh") +
    scale_x_continuous(breaks = seq(0, 11, 2), labels = c("January",
        "March", "May", "July", "September", "November")) + theme_bw() +
    ggtitle("Mean Solar kWh by month, 2017-2020") + theme(plot.title = element_text(hjust = 0.5))

# solar kwh by hour of the day, split by time of the year

df.winter <- df[which(df$month %in% c(10, 11, 12, 1, 2, 3)), ]
df.summer <- df[-which(df$month %in% c(10, 11, 12, 1, 2, 3)), ]

df.winter.hrs <- aggregate(df.winter, list(as.POSIXlt(df.winter$date.time)$hour),
    mean)
df.winter.hrs2 <- df.winter.hrs[order(df.winter.hrs$Group.1), ]

df.summer.hrs <- aggregate(df.summer, list(as.POSIXlt(df.summer$date.time)$hour),
    mean)
df.summer.hrs2 <- df.summer.hrs[order(df.summer.hrs$Group.1), ]

g2 <- ggplot(df.winter.hrs2, aes(Group.1, hourly_solar_kWh)) + geom_line(data = df.summer.hrs2,
    aes(Group.1, hourly_solar_kWh, color = "Summer"), size = 2, alpha = 0.6) +
    geom_line(data = df.winter.hrs2, aes(Group.1, hourly_solar_kWh,
        color = "Winter"), size = 2, alpha = 0.6) + xlab("Time") +
    ylab("Average Hourly Solar kWh") + scale_x_continuous(breaks = seq(0,
    23, 3), labels = c("12AM", "3AM", "6AM", "9AM", "12PM", "3PM",
    "6PM", "9PM")) + theme_bw() + ggtitle("Mean Solar kWh by time, 2017-2020") +
    theme(plot.title = element_text(hjust = 0.5)) + scale_color_manual(values = c("tomato3",
    "dodgerblue")) + theme(legend.title = element_blank())

g1
```


Mean Solar kWh by month, 2017−2020
Average Hourly Solar kWh
0.8
0.7
0.6
0.5
January
March
May
July
September
November
Month

```
g2
```




```
pacman::p_load(officer)
# plots = read_docx() plots = body_add_gg(x = plots, value = g1,
# style = 'centered', res = 1200) print(plots, target =
# 'plots.docx')


# remove temperatures above 75 and months outside of November and
# March
df1 <- df[which(df$temperature <= 75 & df$month %in% c(11, 12, 1,
    2, 3)), setdiff(names(df), "date.time")]

# prepare the test data set
test.kwh <- read.csv("test_kwh.csv")
test.kwh$hourly_time <- as.POSIXct(strptime(as.character(test.kwh$hourly_time),
    "%m/%d/%Y %H:%M"))
test.kwh$hour <- as.POSIXlt(test.kwh$hourly_time)$hour
test.kwh$month <- as.POSIXlt(test.kwh$hourly_time)$mon + 1
test.kwh$hourly_solar_kWh <- test.kwh$solar
test.kwh$hourly_gal <- test.kwh$hourly_gal_water_consumed
test.kwh1 <- test.kwh[, names(test.kwh) %in% names(df)]

# prepare the training and testing data set
df.train <- df1[, setdiff(names(df1), c("month", "hour"))]
test.names <- setdiff(names(df.train), "hourly_kwh")
df.test <- test.kwh1[, test.names]
```

```r
pacman::p_load(glmnet)

# fit a LASSO
x <- model.matrix(hourly_kwh ~ ., data = df.train)[, -1]
y <- df.train$hourly_kwh
set.seed(1)
cv <- cv.glmnet(x, y, trace.it = F, alpha = 1, nfolds = 10)
# fit the model using the best value of lambda
lm <- glmnet::glmnet(x, y, lambda = cv$lambda.min, alpha = 1)
coef(lm)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                                  s0
## (Intercept)          -9.795649e-01
## temperature            .
## dew_point              .
## humidity             -1.665419e-01
## visibility             .
## apparent_temperature -6.885221e-04
## pressure              1.900966e-03
## wind_speed             .
## cloud_cover            .
## wind_bearing         -9.795224e-05
## precip_intensity       .
## precip_probability     .
## hourly_solar_kWh     -2.954233e-02
## hourly_gal             .
```

```r
lasso.pred <- predict(lm, newx = x)
# training RMSE
sqrt(mean((lasso.pred - y)^2))
```

```
## [1] 0.2249425
```

```r
sd(df.train$hourly_kwh)
```

```
## [1] 0.2314771
```

```r
# predictions
lasso.pred.test <- predict(lm, newx = data.matrix(df.test))
fileConn <- file("test_preds.kwh")
writeLines(as.character(lasso.pred.test), fileConn)
close(fileConn)
```