Markdown_Felix

Felix Bigler

2/1/2022

Contents

Data Preperation

Load Data

```
# set directory read data
setwd(dirname(getActiveDocumentContext()$path))
df_weather <- read.csv("./data/weather.csv",header=TRUE,sep =",",comment.char ="#")
df_plantA <- read.csv("./data/A.csv",header=TRUE,sep =",")
df_plantB <- read.csv("./data/B.csv",header=TRUE,sep =",")
df_plantC <- read.csv("./data/C.csv",header=TRUE,sep =",")
```

Inspect Data Structure

```
## 'data.frame':    8760 obs. of  10 variables:
##  $ time             : chr  "2019-01-01 00:00" "2019-01-01 01:00" "2019-01-01 02:00" "2019-01-01 03:0
##  $ local_time       : chr  "2019-01-01 01:00" "2019-01-01 02:00" "2019-01-01 03:00" "2019-01-01 04:0
##  $ temperature      : num  0.772 0.434 0.349 0.568 0.899 ...
##  $ precipitation    : num  0.002 0.002 0.002 0.001 0.001 0.002 0.002 0.003 0.003 0.002 ...
##  $ snowfall         : num  0 0 0 0 0 0 0 0.001 0 0 ...
```

```
##  $ snow_mass        : num  2.1 2.1 2.1 2.1 2.1 ...
##  $ air_density      : num  1.21 1.21 1.21 1.21 1.21 ...
##  $ radiation_surface: num  0 0 0 0 0 ...
##  $ radiation_toa    : num  0 0 0 0 0 ...
##  $ cloud_cover      : num  0.543 0.824 0.965 0.969 0.97 0.97 0.966 0.968 0.991 0.968 ...
```

```
## 'data.frame':    35040 obs. of  5 variables:
##  $ Timestamp                 : chr  "2019-01-01 00:00:00" "2019-01-01 00:15:00" "2019-01-01 00:30:0
##  $ Generation_kW             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Grid_Feed.In_kW           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Grid_Supply_kW            : num  4.21 4.21 4.21 4.22 4.21 ...
##  $ Overall_Consumption_Calc_kW: num  4.21 4.21 4.21 4.22 4.21 ...
```

```
## 'data.frame':    35040 obs. of  5 variables:
##  $ Timestamp                 : chr  "2019-01-01 00:00:00" "2019-01-01 00:15:00" "2019-01-01 00:30:0
##  $ Generation_kW             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Grid_Feed.In_kW           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Grid_Supply_kW            : num  5.4 6 5.7 6 5.7 5.7 5.4 5.4 5.7 6 ...
##  $ Overall_Consumption_Calc_kW: num  5.4 6 5.7 6 5.7 5.7 5.4 5.4 5.7 6 ...
```

```
## 'data.frame':    35040 obs. of  3 variables:
##  $ Timestamp      : chr  "2019-01-01 00:00:00" "2019-01-01 00:15:00" "2019-01-01 00:30:00" "2019-01-
##  $ Grid_Feed.In_kW: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Grid_Supply_kW : num  2.8 2.8 3 3 3 3.6 4.8 4.8 5 4.8 ...
```

We got 8760 observations for the weather dataset and 35040 observations for each PV plant Dataset. Which makes sense since we got weather data for every hour for the year 2019 and on the other hand, operational data of the PV plants for every 15 min in kW. The values are all of the type num except the datetime entries. These are of type char. In a next step we convert the datetime entries to a datetime object to have the possibility to work with the datetime entries.

```
##             Timestamp Generation_kW Grid_Feed.In_kW Grid_Supply_kW
## 1 2019-01-01 00:00:00             0               0          4.212
## 2 2019-01-01 00:15:00             0               0          4.212
## 3 2019-01-01 00:30:00             0               0          4.212
## 4 2019-01-01 00:45:00             0               0          4.220
## 5 2019-01-01 01:00:00             0               0          4.212
## 6 2019-01-01 01:15:00             0               0          4.212
##   Overall_Consumption_Calc_kW
## 1                       4.212
## 2                       4.212
## 3                       4.212
## 4                       4.220
## 5                       4.212
## 6                       4.212
```

```
##             Timestamp Generation_kW Grid_Feed.In_kW Grid_Supply_kW
## 1 2019-01-01 00:00:00             0               0            5.4
## 2 2019-01-01 00:15:00             0               0            6.0
## 3 2019-01-01 00:30:00             0               0            5.7
## 4 2019-01-01 00:45:00             0               0            6.0
## 5 2019-01-01 01:00:00             0               0            5.7
## 6 2019-01-01 01:15:00             0               0            5.7
##   Overall_Consumption_Calc_kW
## 1                         5.4
## 2                         6.0
## 3                         5.7
```

```
## 4                        6.0
## 5                        5.7
## 6                        5.7
```

```
##             Timestamp Grid_Feed.In_kW Grid_Supply_kW
## 1 2019-01-01 00:00:00               0            2.8
## 2 2019-01-01 00:15:00               0            2.8
## 3 2019-01-01 00:30:00               0            3.0
## 4 2019-01-01 00:45:00               0            3.0
## 5 2019-01-01 01:00:00               0            3.0
## 6 2019-01-01 01:15:00               0            3.6
```

```
##               time       local_time temperature precipitation snowfall
## 1 2019-01-01 00:00 2019-01-01 01:00       0.772         0.002        0
## 2 2019-01-01 01:00 2019-01-01 02:00       0.434         0.002        0
## 3 2019-01-01 02:00 2019-01-01 03:00       0.349         0.002        0
## 4 2019-01-01 03:00 2019-01-01 04:00       0.568         0.001        0
## 5 2019-01-01 04:00 2019-01-01 05:00       0.899         0.001        0
## 6 2019-01-01 05:00 2019-01-01 06:00       0.801         0.002        0
##   snow_mass air_density radiation_surface radiation_toa cloud_cover
## 1     2.095       1.208                 0             0       0.543
## 2     2.095       1.209                 0             0       0.824
## 3     2.095       1.210                 0             0       0.965
## 4     2.095       1.209                 0             0       0.969
## 5     2.095       1.209                 0             0       0.970
## 6     2.096       1.209                 0             0       0.970
```

Convert datetime entries from char to datetime object

Since all datetime are from the type char we convert the local time to a datetime object.

```r
df_weather$local_time <- as.POSIXct(df_weather$local_time,tz="GMT",format="%Y-%m-%d %H:%M")
df_plantA$Timestamp <- as.POSIXct(df_plantA$Timestamp,tz="GMT",format="%Y-%m-%d %H:%M:%S")
df_plantB$Timestamp <- as.POSIXct(df_plantB$Timestamp,tz="GMT",format="%Y-%m-%d %H:%M:%S")
df_plantC$Timestamp <- as.POSIXct(df_plantC$Timestamp,tz="GMT",format="%Y-%m-%d %H:%M:%S")
head(df_plantA)
```

```
##             Timestamp Generation_kW Grid_Feed.In_kW Grid_Supply_kW
## 1 2019-01-01 00:00:00             0               0          4.212
## 2 2019-01-01 00:15:00             0               0          4.212
## 3 2019-01-01 00:30:00             0               0          4.212
## 4 2019-01-01 00:45:00             0               0          4.220
## 5 2019-01-01 01:00:00             0               0          4.212
## 6 2019-01-01 01:15:00             0               0          4.212
##   Overall_Consumption_Calc_kW
## 1                       4.212
## 2                       4.212
## 3                       4.212
## 4                       4.220
## 5                       4.212
## 6                       4.212
```

Resample PV plant Timeseries from 15min to Hourly Intervalls

Since the PV plant entries are 15 minute observations and the weather data is hourly we resample the PV plant dataset by grouping every 15 min observation to its corresponding hour and take the sum of it. We got then directly the energy gained in kWh instead of the 15 min PV power in kW which is more convenient.

```
df_plantA_resample <- df_plantA %>%
  mutate(datetime = floor_date(Timestamp, "1 hour")) %>%
  group_by(datetime) %>%
  summarise(across(Generation_kW:Overall_Consumption_Calc_kW, sum))
```

```
head(df_plantA_resample,echo=FALSE)
```

```
## # A tibble: 6 x 5
##   datetime            Generation_kW Grid_Feed.In_kW Grid_Supply_kW
##   <dttm>                      <dbl>           <dbl>          <dbl>
## 1 2019-01-01 00:00:00             0               0           16.9
## 2 2019-01-01 01:00:00             0               0           16.8
## 3 2019-01-01 02:00:00             0               0           17.5
## 4 2019-01-01 03:00:00             0               0           16.9
## 5 2019-01-01 04:00:00             0               0           17.5
## 6 2019-01-01 05:00:00             0               0           16.9
## # ... with 1 more variable: Overall_Consumption_Calc_kW <dbl>
```

Merge Datasets

In this step we merge both datasets with an inner_join by the entry datetime. For that we have to rename the column local_time to datetime in the weather dataset.

```
df_weather <- df_weather %>%
  rename(
    datetime = local_time,
    )
```

```
df_A_joined <- inner_join(df_plantA_resample, df_weather, by = "datetime")
```

```
head(df_A_joined,echo=FALSE)
```

```
## # A tibble: 6 x 14
##   datetime            Generation_kW Grid_Feed.In_kW Grid_Supply_kW
##   <dttm>                      <dbl>           <dbl>          <dbl>
## 1 2019-01-01 01:00:00             0               0           16.8
## 2 2019-01-01 02:00:00             0               0           17.5
## 3 2019-01-01 03:00:00             0               0           16.9
## 4 2019-01-01 04:00:00             0               0           17.5
## 5 2019-01-01 05:00:00             0               0           16.9
## 6 2019-01-01 06:00:00             0               0           17.4
## # ... with 10 more variables: Overall_Consumption_Calc_kW <dbl>, time <chr>,
## #   temperature <dbl>, precipitation <dbl>, snowfall <dbl>, snow_mass <dbl>,
## #   air_density <dbl>, radiation_surface <dbl>, radiation_toa <dbl>,
## #   cloud_cover <dbl>
```

Create Categorical Variable

A prerequisite of the course assignment is that the data set contains at least one categorical variable. Since our data set does not have any categorical variable we create two of them by extracting the month as well as the day out of the date time entries with the package lubridate as follows.

```
df_A_joined <-df_A_joined %>%
 mutate(
  month = month(datetime),
  month_label = month(datetime, label=TRUE),
  hour = hour(datetime),
  day = day(datetime),
  year = year(datetime)
 )
```

Graphical Observations of Datasets

In this chapter we would like to explore the dataset graphically. First we are going create a correlation matrix in order to see which variables correlate with the PV plant production rate. Then we explore the continuous as well as the categorical variables graphically.

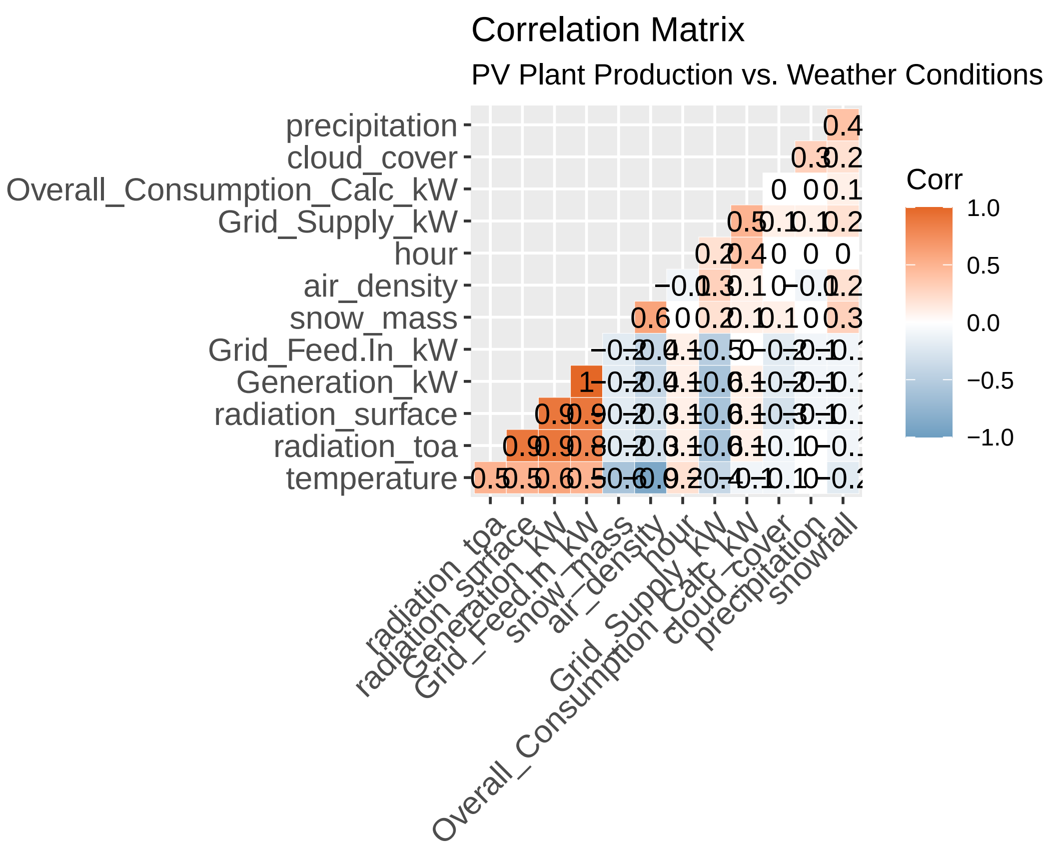
Create Correlation Matrix

```
df_corr <- select(df_A_joined,-c(datetime,time,month_label,year,day,month)) # Exclude datetime entries
corr <- round(cor(df_corr), 1)
#head(corr[, 1:6])
```

```
head(df_corr)
```

```
## # A tibble: 6 x 13
##   Generation_kW Grid_Feed.In_kW Grid_Supply_kW Overall_Consumption_~ temperature
##           <dbl>           <dbl>          <dbl>                 <dbl>       <dbl>
## 1             0               0           16.8                  16.8       0.772
## 2             0               0           17.5                  17.5       0.434
## 3             0               0           16.9                  16.9       0.349
## 4             0               0           17.5                  17.5       0.568
## 5             0               0           16.9                  16.9       0.899
## 6             0               0           17.4                  17.4       0.801
## # ... with 8 more variables: precipitation <dbl>, snowfall <dbl>,
## #   snow_mass <dbl>, air_density <dbl>, radiation_surface <dbl>,
## #   radiation_toa <dbl>, cloud_cover <dbl>, hour <int>
```

```
p1 <- ggcorrplot(corr, hc.order = TRUE, type = "lower",
   outline.col = "white",
   ggtheme = ggplot2::theme_gray,
   lab = TRUE,
   colors = c("#6D9EC1", "white", "#E46726")) +
   labs(title = "Correlation Matrix",
       subtitle = "PV Plant Production vs. Weather Conditions")
p1
```



We can conclude from the correlation matrix, that the variables radiation_surface (Ground-level solar irradiance (W / m²)) and radiation_toa (Top of atmosphere solar irradiance (W / m²)) have a strong positive correlation with the PV production rate (variable Generation_kW). The coefficients are very high with a value of 0.9 which seems obvious, since a solar cell converts solar radiation into electrical energy when sunlight is present on the surface of the cell. The two variables certainly have a strong colinearity since the values differ only in respect to the fact that one value was measured at the ground and the other value above the atmosphere. For the actual solar production, however, only the value at the ground is relevant.

The variable temperature has also a rather high correlation with a value of 0.6. However, this correlation may have more in common with the fact that the temperatures are also higher in the case of strong solar irradiation. In fact, solar modules have a negative temperature coefficient in relation to the efficiency rating (https://www.pveurope.eu/solar-modules/global-warming-growing-importance-temperature-coefficient-solar-modules).

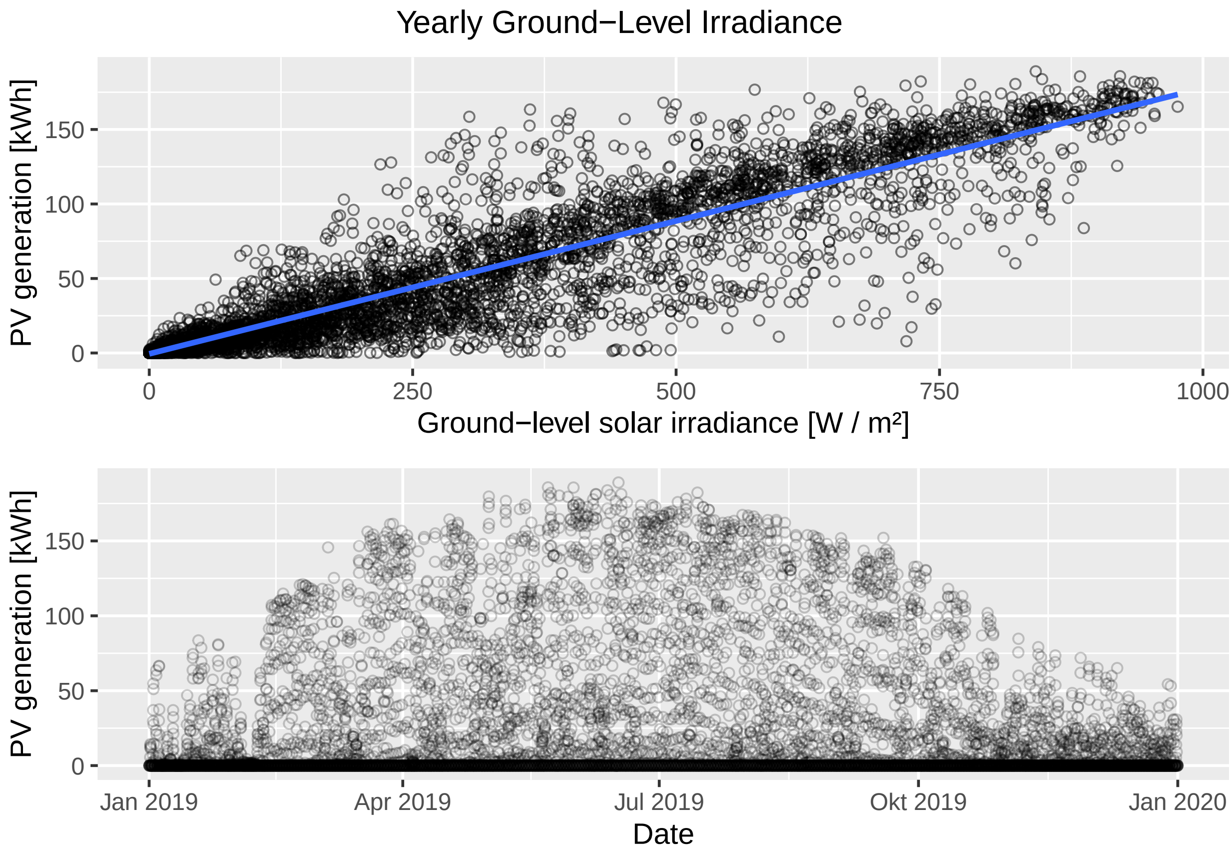
Exploration of Contineous Variables

Yearly Ground-Level Irradiance

```
p2 <-  ggplot(data = df_A_joined,
         mapping = aes(y = Generation_kW ,
                       x = radiation_surface)) + xlab("Ground-level solar irradiance [W / m²]") + ylab("PV
     geom_point(alpha = 0.5,shape = 1) +
     geom_smooth(method = "lm")

p3 <-  ggplot(data = df_A_joined,
         mapping = aes(y = Generation_kW ,
                       x = datetime)) + xlab("Date") + ylab("PV generation [kWh]") +
```

```
    geom_point(alpha = 0.2,shape = 1) +
    labs(colour = "Method")

grid.arrange(p2,p3, nrow=2,top = "Yearly Ground-Level Irradiance")
```

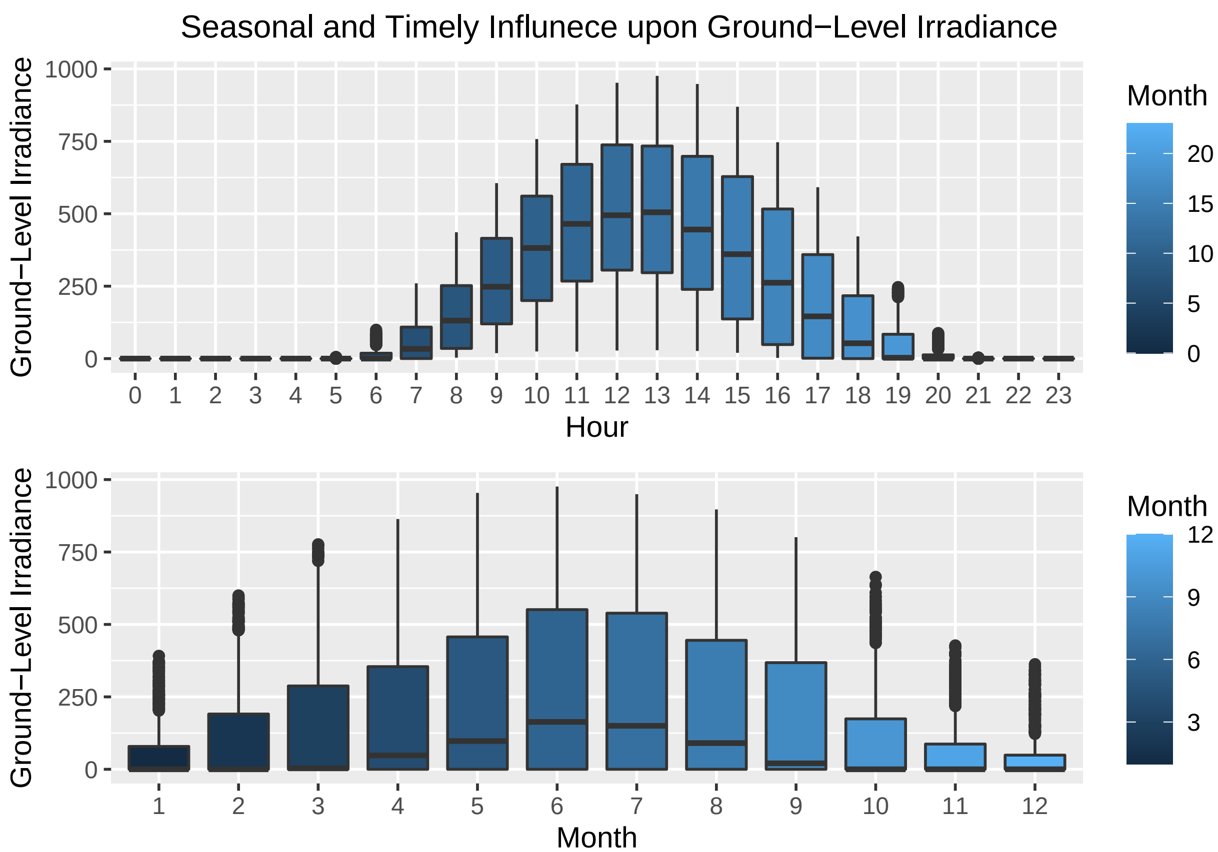
```
## `geom_smooth()` using formula 'y ~ x'
```




```
fig.align = 'center'
```

Exploration of Categorical Variables

Seasonal and Timely Ground-Level Irradiance



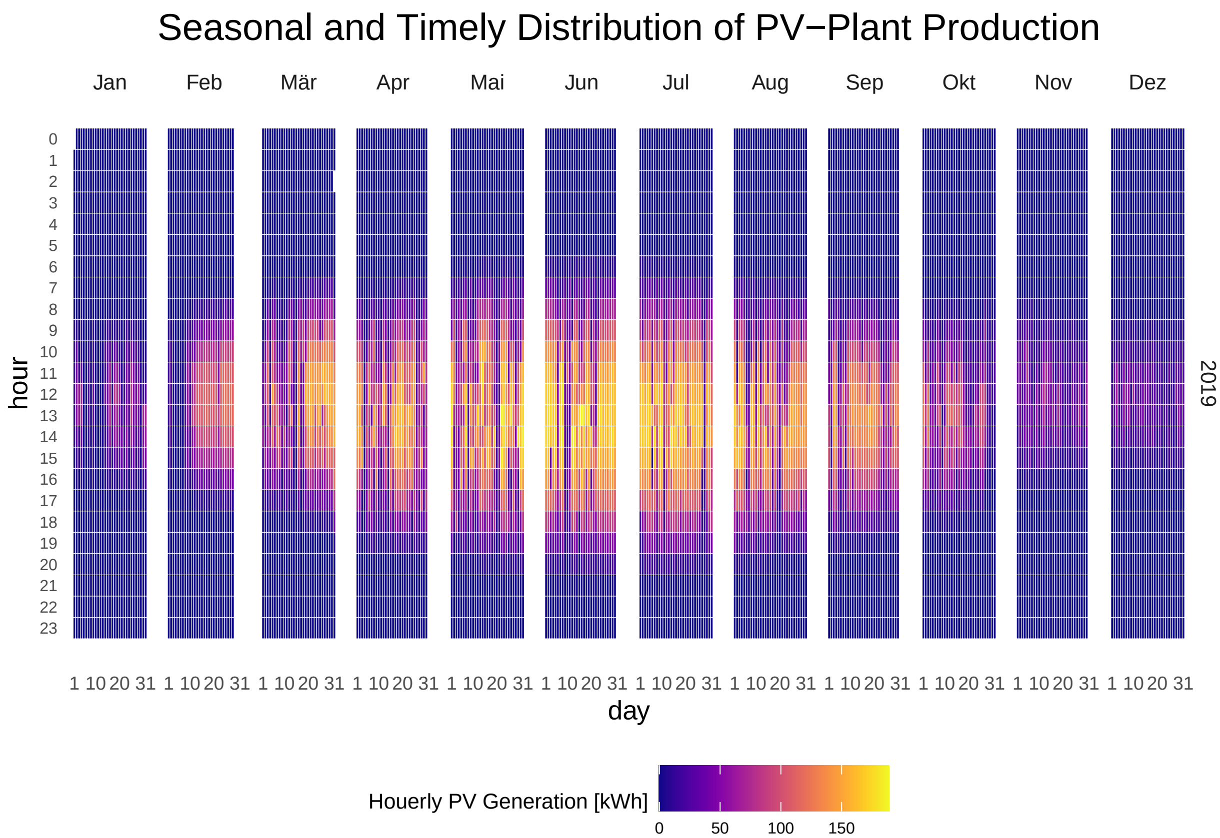
PV Plant Production

```
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(ggExtra)
p6 <- ggplot(data = df_A_joined, aes(x = day,y = hour,fill=Generation_kW))+
  geom_tile(color= "white",size=0.1) +
  scale_fill_viridis(name="Houerly PV Generation [kWh]",option ="C") +
  facet_grid(year ~ month_label) +
  scale_y_continuous(trans = "reverse", breaks = unique(df_A_joined$hour)) +
  scale_x_continuous(breaks =c(1,10,20,31)) +
  theme_minimal(base_size = 10)+
  labs(title= "Seasonal and Timely Distribution of PV-Plant Production")+
  theme(legend.position = "bottom") +
  theme(plot.title=element_text(size = 14)) +
  theme(axis.text.y=element_text(size=6)) +
  theme(strip.background = element_rect(colour="white")) +
  theme(plot.title=element_text(hjust=0)) +
  theme(axis.ticks=element_blank()) +
  theme(axis.text=element_text(size=7)) +
```

```
  theme(legend.title=element_text(size=8)) +
  theme(legend.text=element_text(size=6)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  removeGrid()
p6
```



Battery Loading Algorithm

The main drawback of photovoltaics is that most of the energy production falls in the summer months, which leads to a surplus of energy for this period. This surplus can be stored with a battery, for example. We would like create a simple battery charging algorithm to simulate a battery in the solar system and evaluate how the self-consumption of the system can be increased by adding a storage.

Lets first calculate the total energy production as well as the total consuption:

```
sum(df_A_joined$Generation_kW) / sum((df_A_joined$Overall_Consumption_Calc_kW))
```

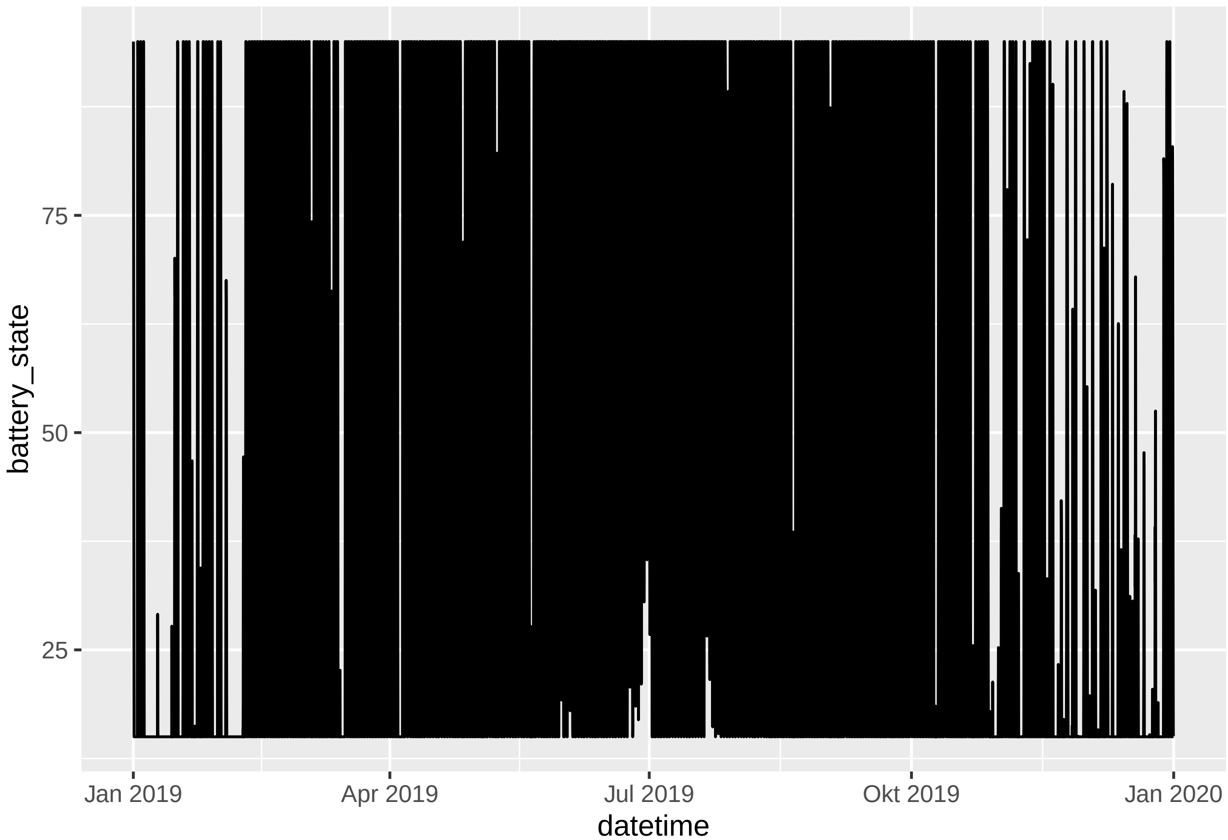
```
## [1] 1.765006
```

```
#battery_capacity <- 10 # Value in kWh
#max_battery_capacity <- 95 # in %
#min_battery_capacity <- 15 # in %

battery_state <- function(battery_capacity,max_battery_capacity,min_battery_capacity,pv_generation, cons

  min_battery_load = battery_capacity * (min_battery_capacity)/100
  max_battery_load = battery_capacity * (max_battery_capacity)/100
```

```r
  battery_load = battery_state

  if ((pv_generation - consumption) > 0 & battery_load < max_battery_load) {
    if (battery_load + (pv_generation - consumption) > max_battery_load) {
      battery_load = max_battery_load
    } else
    {battery_load = battery_load + (pv_generation - consumption)}
  }
  else if ((pv_generation-consumption) < 0 & (battery_load > min_battery_load)) {
    if (battery_load + (pv_generation - consumption) < min_battery_load) {
      battery_load = min_battery_load
    } else {
      battery_load = battery_load + (pv_generation - consumption) }
  }
   return(battery_load)
}
```

```r
battery_capacity <- 100 # Value in kWh
max_battery_capacity <- 95 # in %
min_battery_capacity <- 15 # in %

df_A_joined$battery_state <- battery_capacity * (max_battery_capacity)/100

for(i in 2:nrow(df_A_joined)) {       # for-loop over rows
  df_A_joined$battery_state[i] <- battery_state(battery_capacity,max_battery_capacity,min_battery_capac
                                                df_A_joined$Generation_kW[i],
                                                df_A_joined$Overall_Consumption_Calc_kW[i],
                                                df_A_joined$battery_state[i-1]
  )
}
```

```r
ggplot(data=df_A_joined, aes(x=datetime, y=battery_state)) +
  geom_line()
```




```
library(viridis)
library(ggExtra)
p <- ggplot(data = df_A_joined, aes(x = day,y = hour,fill=battery_state))+
  geom_tile(color= "white",size=0.1) +
  scale_fill_viridis(name="Hrly PV Generation [kW]",option ="C")
p <-p + facet_grid(year ~ month_label)
p <-p + scale_y_continuous(trans = "reverse", breaks = unique(df_A_joined$hour))
p <-p + scale_x_continuous(breaks =c(1,10,20,31))
p <-p + theme_minimal(base_size = 10)
#p <-p + labs(title= paste("Hourly Temps - Station",statno), x="Day", y="Hour Commencing")
p <-p + theme(legend.position = "bottom")+
  theme(plot.title=element_text(size = 14))+
  theme(axis.text.y=element_text(size=6)) +
  theme(strip.background = element_rect(colour="white"))+
  theme(plot.title=element_text(hjust=0))+
  theme(axis.ticks=element_blank())+
  theme(axis.text=element_text(size=7))+
  theme(legend.title=element_text(size=8))+
  theme(legend.text=element_text(size=6))+
  removeGrid()
```