Markdown_Felix

Felix Bigler

2/1/2022

Contents

1 Intrudction

In this group work for the R bootcamp module, we would like to analyze a photovoltaic system dataset. The dataset was created during the operation of AEW-owned PV plants in 2019. In total there are 3 differnt plants but we will only concentrate on one (Plant A). The plants are all located in Aargau, Switzerland. The power values in kW refer to the average over the 15min period. The data was first time published for the Energy Data Hackdays 2020 in Brugg.

We then merge this dataset with weather data from 2019 in order to evaluate the influence of the weather as well as be aple to build a simple prediction model. Moreover we would like to create a simple battery charging algorithm in order to store surplus energy.

2 Data Preperation

2.1 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 =",")
```

2.2 Inspect Data Structure

```
str(df_weather)
str(df_plantA)
```

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. There are no missing values so we don't have to

2.3 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.

```
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")
```

2.4 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))
```

2.5 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")
```

2.6 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)
 )
```

3 Graphical Observations of Datasets

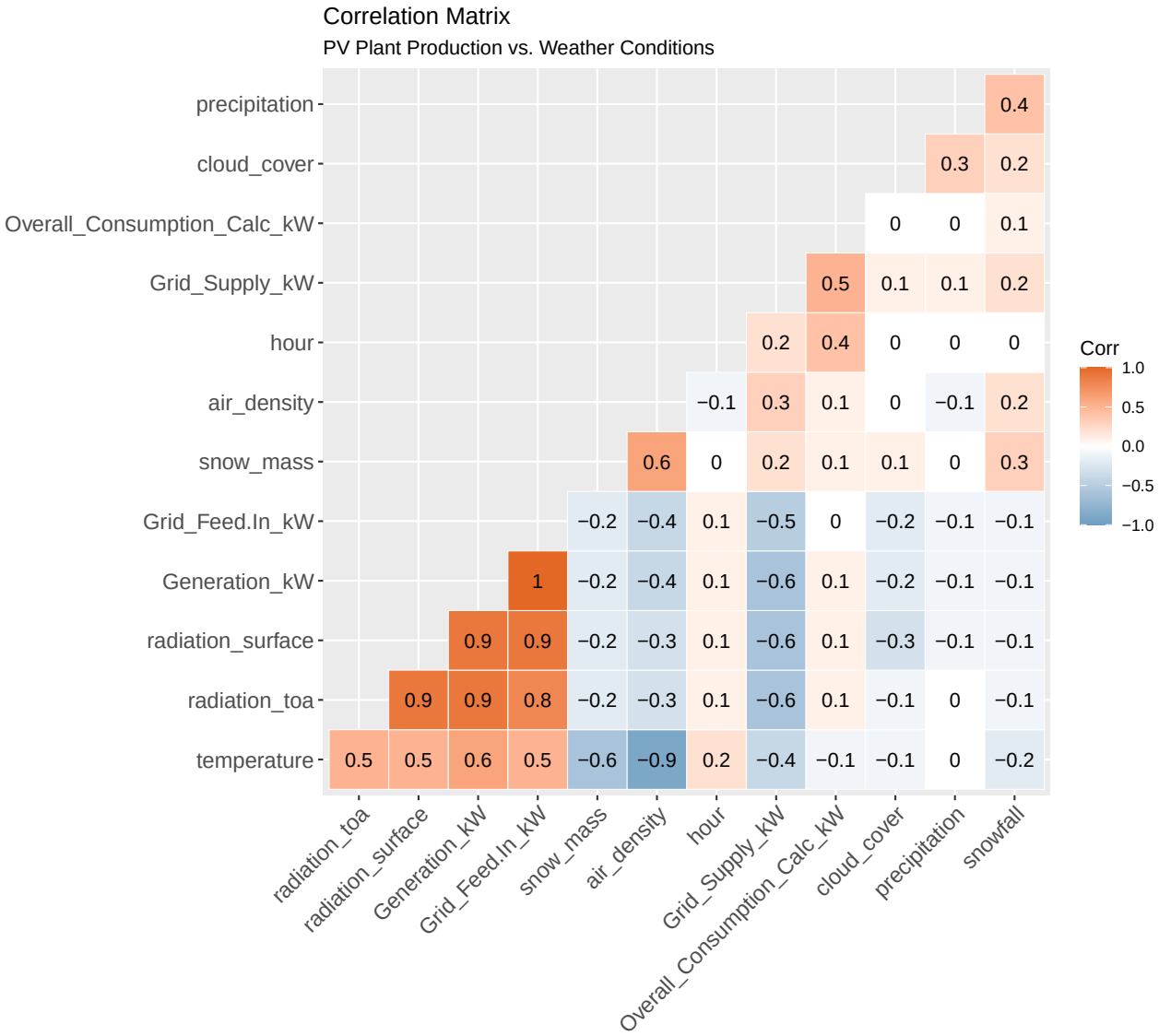
In this chapter we would like to explore the dataset graphically. First we are going create a correlation matrix in order to see which variables correlate with the PV plant production rate. Then we explore the continuous as well as the categorical variables graphically.

3.1 Create Correlation Matrix

```
knitr::opts_chunk$set(fig.width=12, fig.height=8)
df_corr <- select(df_A_joined,-c(datetime,time,month_label,year,day,month))
# Exclude datetime entries in order to create correlation matrix
corr <- round(cor(df_corr), 1)
#head(corr[, 1:6])
```

```
head(df_corr)
```

```
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).

The other weather variables don't have a significant influence, therefore we concentrate on the variable irradiance in the further analysis of the data set.

3.2 Exploration of Contineous Variables

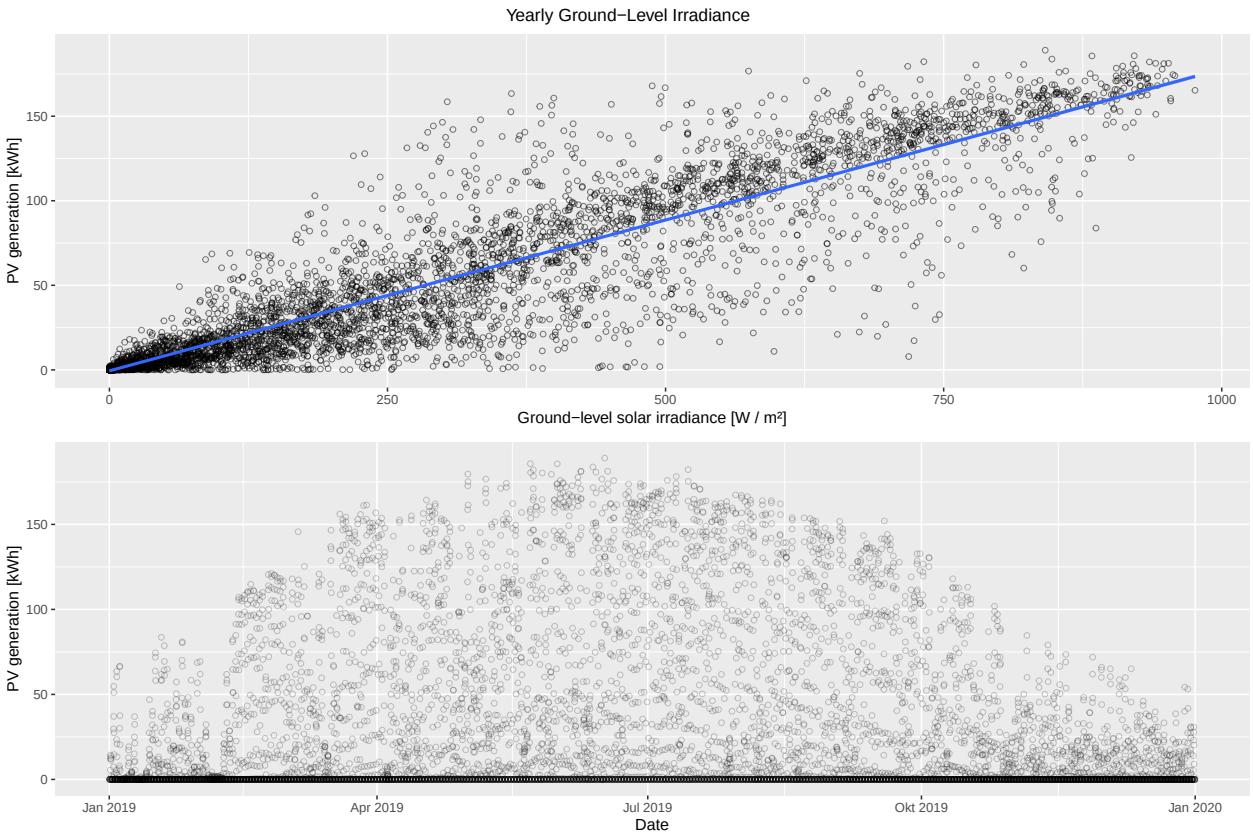
3.2.1 Yearly Ground-Level Irradiance

```
p2 <-  ggplot(data = df_A_joined,
         mapping = aes(y = Generation_kW ,
                       x = radiation_surface)) +
  xlab("Ground-level solar irradiance [W / m²]") +
  ylab("PV generation [kWh]") +
     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")
```

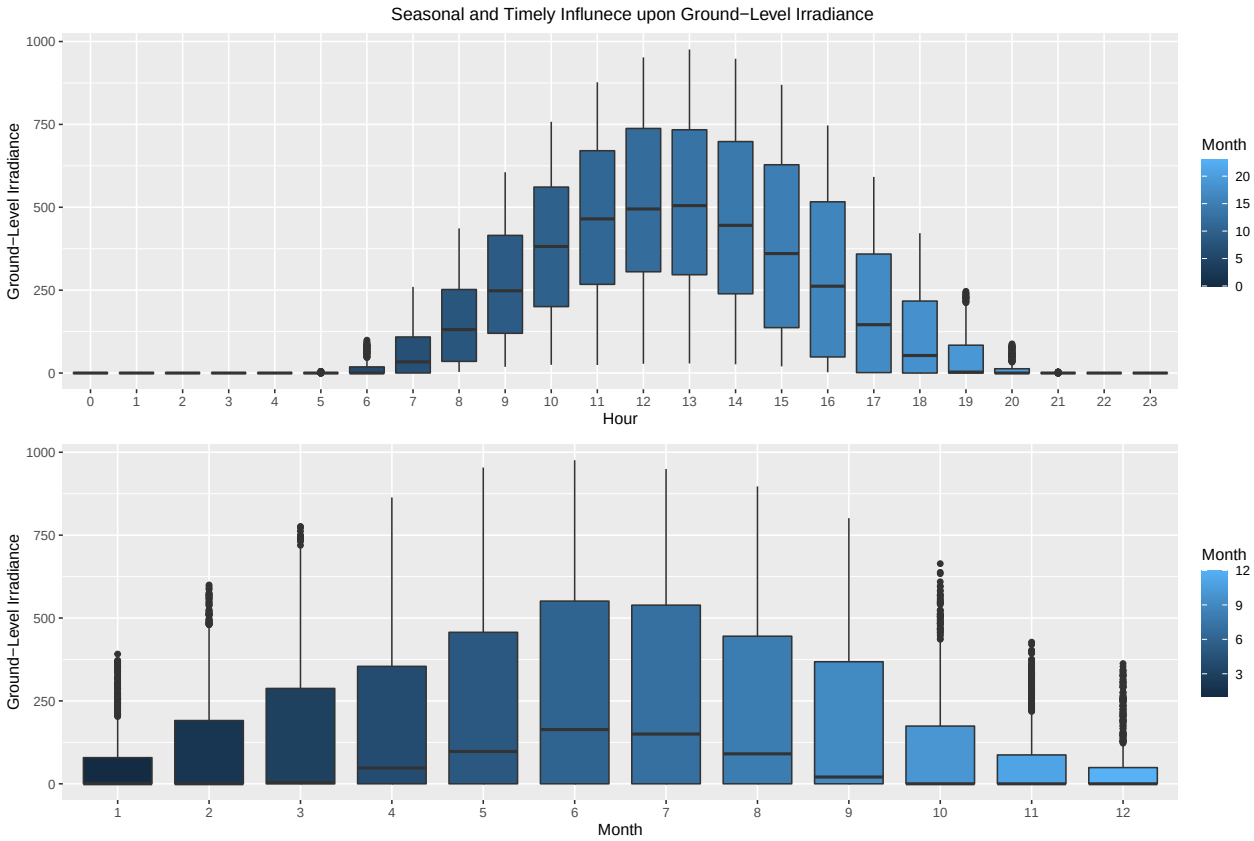
```
## `geom_smooth()` using formula 'y ~ x'
```



* we can conclude from the plot above, that the PV production highly correlates with the gorund level solar irradiance as the correlation plot already explained.The blue line indicates a linear regression line.

3.3 Exploration of Categorical Variables

3.3.1 Seasonal and Timely Ground-Level Irradiance



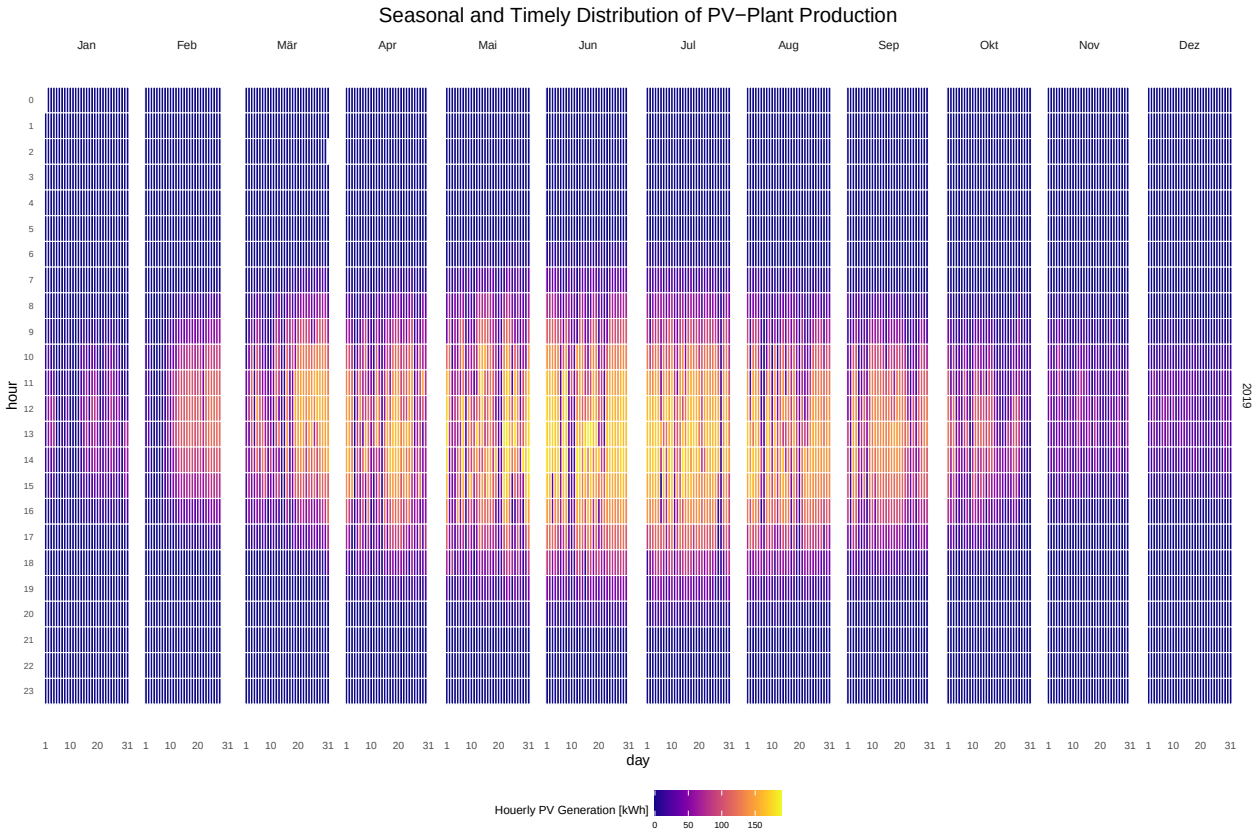
- as we have expected, the most solar irradiance occurs between April and September and during the hours from 10am to 3pm as the boxplots above indicates.

3.4 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
```



- the plot above shows the seasonal and timely distribution of the PV production of the plant A.

4 Battery Loading Algorithm

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

4.1 Self-Consumption-Rate Without Storage System

Lets first calculate the self-consumption-rate of the system without a battery. We define a function as follows to determine when the produced energy is directly consumed and not feed into the grid:

```
self_consumption_rate <- function(generation,consumption,i){
  self_con <- as.data.frame(matrix(1:8759,ncol=1))
    if ((generation - consumption) >= 0) {
      self_con$V1[i] = consumption
    }else {
      self_con$V1[i] = 0
    }
}
```

We create then a new data frame self_consumption and apply the function with a for loop over the original data frame df_A_joined as follows:

```
self_consumption <- as.data.frame(matrix(1:8759,ncol=1))
for(i in 1:nrow(df_A_joined)){
    self_consumption$V1[i] <- self_consumption_rate(df_A_joined$Generation_kW[i],
                                                    df_A_joined$Overall_Consumption_Calc_kW[i],
                                                    i)
}
```

Now we can calculate the self-consumption-rate as follows:

```
sum(self_consumption$V1)/sum(df_A_joined$Generation_kW)
```

```
## [1] 0.199887
```

- we can see that only 20 % of the produced energy is used directly, the other part is feed into the grid.

4.1.1 Create Simple Battery Loading Algorithm

The next step is to create a simple battery loading function. The input variables for the function are battery capacity, max. battery capacity, min. battery capacity, the PV generation as well as the consumption and the initial battery charging state.

```
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
  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)
}
```

We apply now the battery loading algorithm. We us different battery sizes from 45 kWH up to 450 kWh in 45 kWh steps.

```
max_battery_capacity <- 95 # in %
min_battery_capacity <- 15 # in %

battery_states <- as.data.frame(matrix(0,nrow=8759,ncol=10))
x <- 1

for (j in seq(45, 450, length.out=10)) {
    for(i in 2:nrow(df_A_joined)) {
      battery_states[i,x] <- battery_state(j,max_battery_capacity,min_battery_capacity,
                                           df_A_joined$Generation_kW[i],
                                           df_A_joined$Overall_Consumption_Calc_kW[i],
                                           battery_states[i-1,x])
    }
  x = x + 1
}
```

4.1.2 Calculate Self-Consumption-Rate for Differnt Dattery Sizes

```
self_consumption_battery <- as.data.frame(matrix(0,nrow=8759,ncol=10))
for(j in seq(1,10,length.out = 10)){
  for(i in 1:8758) {
    if ((battery_states[i+1,j] - battery_states[i,j]) < 0) {
      self_consumption_battery[i+1,j] <- self_consumption[i+1,1] + abs(battery_states[i+1,j] - battery_s
    } else{self_consumption_battery[i+1,j] <- self_consumption[i+1,1]}
  }
}
```

- The next step is to create a dataframe containing the different self-consumption-ratios and the corresponding battery capacity.

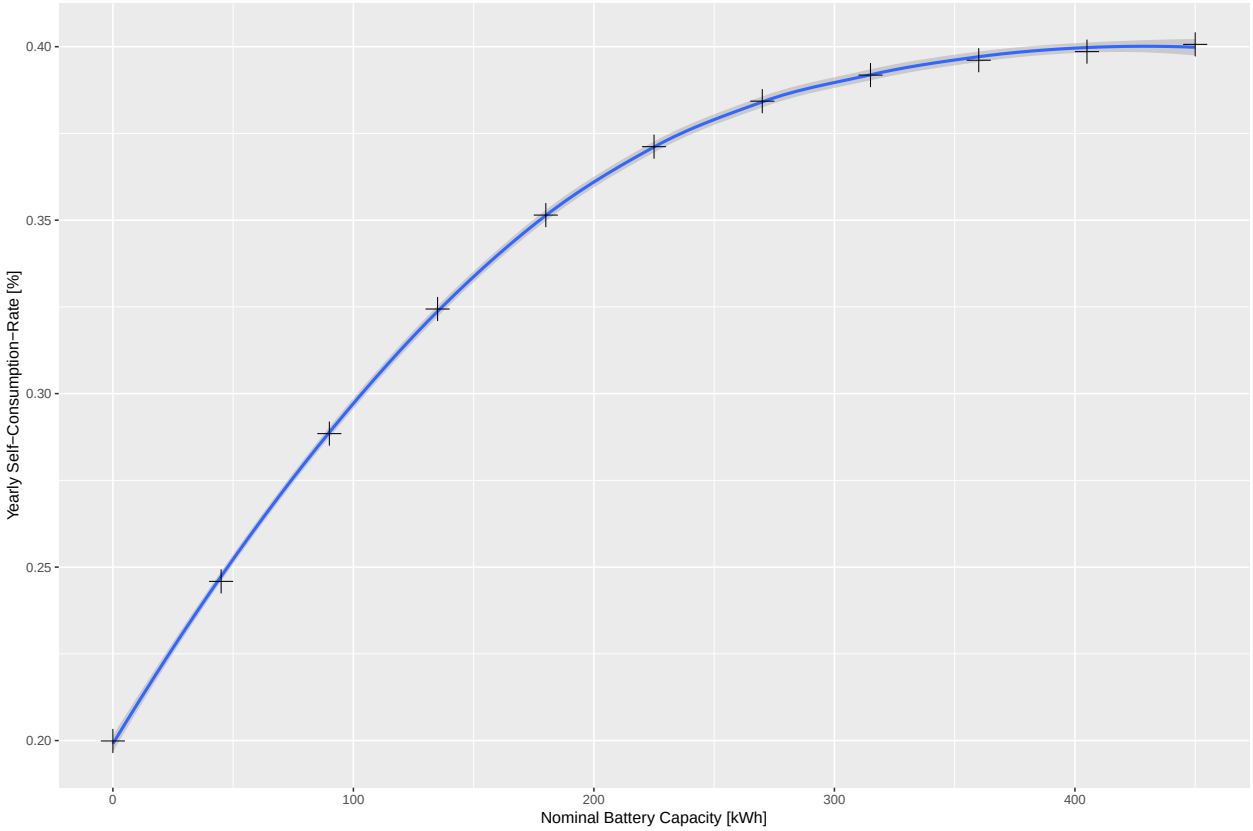
```
df_self_con_ratio <- as.data.frame(colSums(self_consumption_battery)/sum(df_A_joined$Generation_kW))

df_self_con_ratio$battery_capacity <- seq(45, 450, length.out=10)
colnames(df_self_con_ratio) <- c("self_consumption_ratio_kWh","battery_capacity_kWh")

new_row <- c(sum(self_consumption$V1)/sum(df_A_joined$Generation_kW),0)
df_self_con_ratio  <- rbind(new_row,df_self_con_ratio)
```

```
## <ScaleContinuousPosition>
##  Range:
##  Limits:    0 --  500
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



* as we can see from the plot above, the self-consumption-rate starts to flatten out at around a capacity of 220 kWh. This can be explained by the fact, that after a certain point, an increase in battery capacity no longer causes a noticeable increase in the self-consumption rate, since only individual peaks of the PV production are additionally stored.

In the next chapters we try to fit a timeseries model and a GAM model to predict the PV production.