Data Incubator project

-Rohit Aggarwal

# Problem Statement

Given aggregate energy consumption data for one home that contains the following main appliances, process the data to extract the energy consumption time series for individual appliances listed above.

* Central AC 1 - The most common repeating pulse. (At about 2.5 KW amplitude and a width of about 10 minutes)
* Central AC 2 - Another repeating pulse but less frequent (At about 4 KW amplitude and > 30 minute width)
* Pool Pump - Runs for a duration of about 3 hours at 1.5 KW amplitude. Starts at the same time every day.
* Refrigerator - This is the smallest amplitude repeating pulse at < 200 W

Provide a summary detailing,

* Techniques used (Brief note describing why)
* Insights acquired while working on this data
* Interesting visualizations / observations
* Computational complexity of your technique(s)
* How would you solve this if you weren't provided the appliance descriptions for the home?

# Analysis

There are 2 components in this analysis:

1. Observations and cross-verification if the given information about appliance’s operations are indeed accurate?
   1. Spoiler – they seem not.
2. Algorithm to find at any given instant which devices are ON based on observations.
   1. Thresholding based algorithm.

## Part I: Observations and Checking if the given values of appliance’s operations are indeed accurate?

The data set contains the **power consumed (W) and time (seconds)**. Of all the 4 appliances, the refrigerator’s power usage is much less. So, depending on the variance of the power usage of individual devices, it is possible that disaggregation of refrigerator output may not be accurate. Now, since the other 3 devices consume much higher power, we should expect to see up to 7 discrete power consumption levels in the data:

* + Pool pump only - 1500 W (ON time = 3 hrs, 1 day period)
  + AC1 only - 2500 W (period ~ 10 min)
  + AC2 only - 4000 W (period > 30 min)
  + Pool pump + AC1 - 4000 W
  + Pool pump + AC2 - 5500 W
  + AC1+AC2 - 6500 W
  + Pool pump + AC1 + AC2 - 8000 W

**Basic idea: Since the above thresholds are different (except for (pool pump + AC1) vs AC2 only cases), we can divide the time series into its individual components by looking at the total power level. Further, since the pool pump is known to work only for 3 hours in a 1-day period, we can possibly isolate the pool pump ON/OFF time and we should also be able to distinguish between (pool pump + AC1) –vs- AC only cases.**

Upon closer observation we make 4 observations:

**Observation 1:** The original time-series data is not continuous and has total consumed power for certain seconds missing.

**Fix:** We construct a new data frame with 2 columns (power, time) and for each second for which the power series entry is missing, we fill it with the last available valid power entry. This will help in vectorized commands on the entire data without any problems (for example, to calculate the period, we can simply take the difference of indices instead of taking the difference of the times)

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import numdifftools as nd

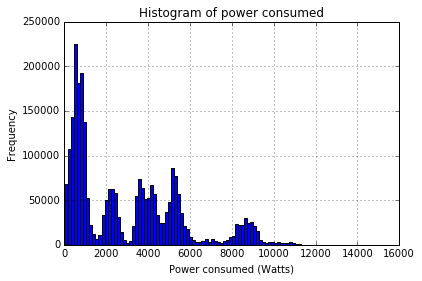
df = pd.read\_table("data.txt",sep=',')

df["time\_offset"] = df["time"] - df["time"].iloc[0]

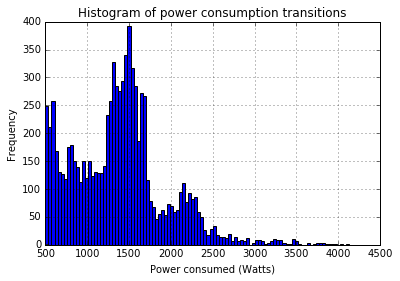
train = df.merge(how='right', on='time\_offset', right = pd.DataFrame({'time\_offset':np.arange(df.iloc[0]['time\_offset'], df.iloc[-1]['time\_offset'] + 1, 1)})).sort(columns='time\_offset').reset\_index().drop(['index'], axis=1)

train = train.fillna(method='pad')

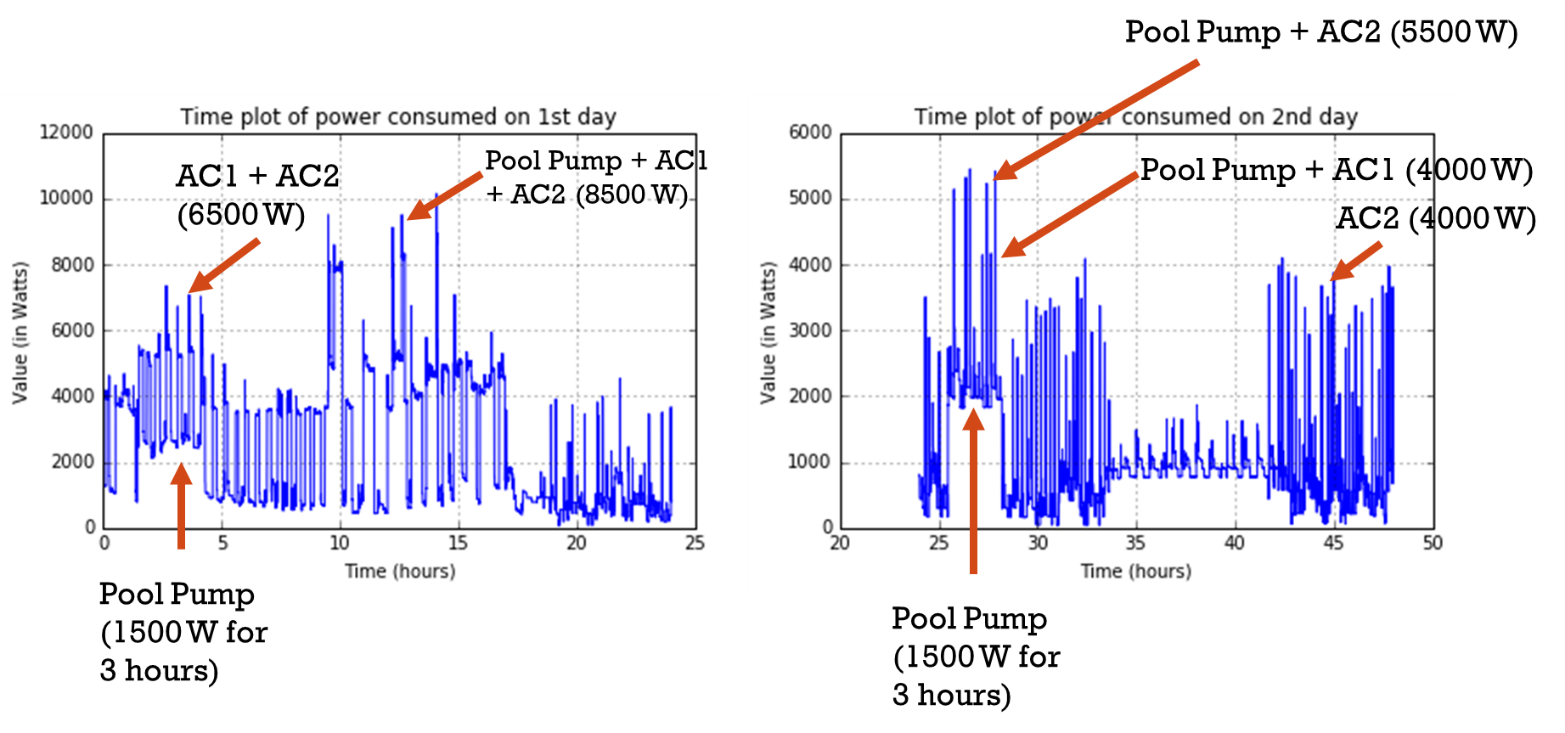
**Observation 2:** If we look at the histogram of total power consumption data from all 31 days (duration of data is 31 days after the fix in observation 1), we can see that the power series data is concentrated around discrete values (see plots below)

****

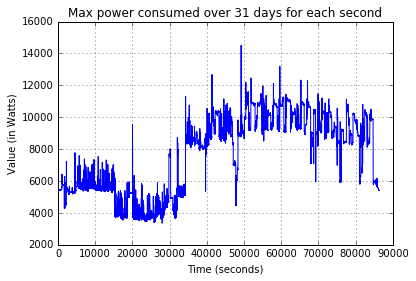
One can also look at the first discrete derivative of consumed power and look at the histogram. If we look at the histogram while ignoring all transitions that are less than 500W, we get the following figure. Assuming that only 1 device is either turned ON or OFF, the plot below gives a tentative idea of how many appliances are being used with their power ratings. **Note that we cannot read too much into this plot because a lot of the high transitions may not occur instantly, and therefore may be absorbed in the lower values in the plot.**



**Observation 3: Contrary to the notion** that pool pump is ON only for 3 hours during a day, if we look at the total power consumption data from 1 day, we can see that there are power values > 8000 W at times that are outside of the 3-hour window where pool pump is switched ON for 3 hours (see image on left below). **Note that 8000W is only possible when all appliances (AC1+AC2+pool pump) are ON.**

****

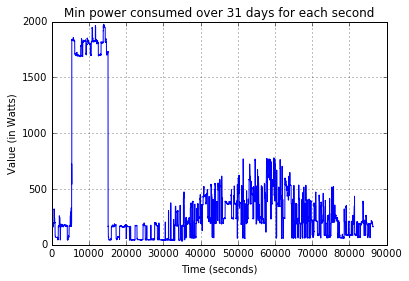
**Assuming that the pool pump is ON in the beginning of the day for 3 hours (as shown in the above pictures)**, if we look at the maximum power series data over 1 day (power maximized for each second over 31 days), we confirm our finding that pool pump (or a new device) has to be ON outside the 3 hours mentioned in the problem statement.



## Part II: Algorithm

We take into account all observations and propose thefollowing algorithm:

1. We take the minimum of power series data for each day, and take the minimum of power over the same seconds each day. That will isolate where the pool pump is ON each day (since we know that it starts at the same time each day). The plot is below:



The duration of the spike in the above plot is 9741 seconds (i.e., 2.7 hours). So, it confirms that if we look for a window of power series data that is above 1500W and extended for more than or equal to 9741 seconds, we will know where exactly each day is the pool pump ON. Once we find that, we can remove the pool pump power from the power series and use thresholding to identify the rest of the appliances’ ON duration. We write a function that takes into the dataframe containing power data for each second and finds all instances where the power exceed a threshold *“power\_appliance“* for more than *“period”* (in seconds).

# dataframe is the dataframe that contains power series data, power\_applicance is the power threshold above which the values are valid, period is the time for which the power > power\_appliance, string\_to\_add is the name of the column that has values 0 and 1, 1 = valid value, 0 means invalid value

def extract\_and\_remove(dataframe, power\_appliance, period, string\_to\_add):

array\_valid\_power = dataframe["power"] \* 0

high\_values\_indices = (dataframe["power"] >= power\_appliance) # Where values are low

array\_valid\_power[high\_values\_indices] = 1

last\_index\_ER = 0

first\_index\_ER = 0

dataframe[string\_to\_add] = dataframe["power"] \*0

while(first\_index\_ER<=(len(array\_valid\_power)-period)):

if(min(array\_valid\_power[first\_index\_ER:first\_index\_ER+period-1]) == 1):

# This is a valid case. Find the last index when array\_valid\_power = 1 after index

last\_index\_ER = np.argmax(array\_valid\_power[first\_index\_ER+1:] == 0, axis=0)-1

first\_index\_ER = int(first\_index\_ER)

last\_index\_ER = int(last\_index\_ER)

dataframe["power"].loc[first\_index\_ER:last\_index\_ER] = dataframe["power"].loc[first\_index\_ER:last\_index\_ER] - power\_appliance

dataframe[string\_to\_add].loc[first\_index\_ER:last\_index\_ER] = 1

first\_index\_ER = max(first\_index\_ER+1,last\_index\_ER)

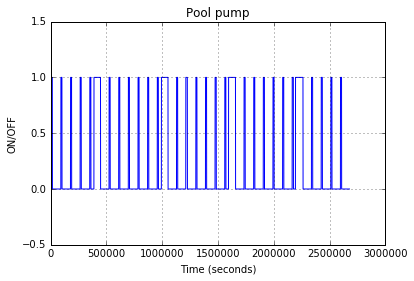
else:

# Find the last occurrence of 0 in first\_index:first\_index+period-1 and go to the next element

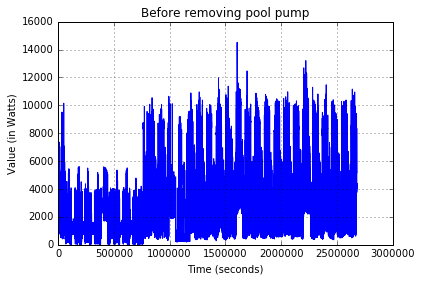
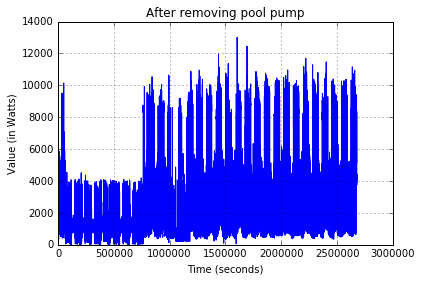
first\_index\_ER = first\_index\_ER + max(max(np.where(array\_valid\_power[first\_index\_ER:first\_index\_ER+period-1] == 0)))+1

return dataframe

1. We call the above function *“extract\_and\_remove”* to isolate pool pump, and then remove the power consumed by the pool pump (i.e., 1500W) from the power series data to simplify it. Value of 1 means pool pump is ON, 0 means pool pump is OFF. **Note that every 7th day, pool pump is ON for > 3Hrs.** This is perhaps on the weekends, the pool pump could be ON for longer durations.

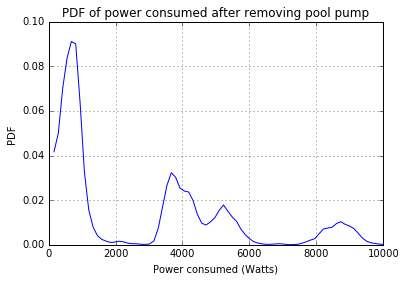
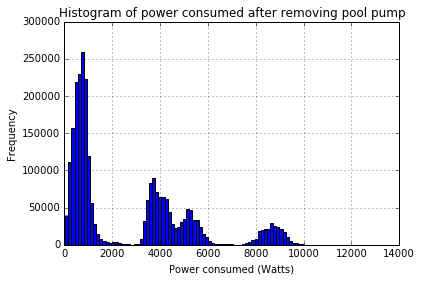


Below is the power series data before and after removing pool pump power values:

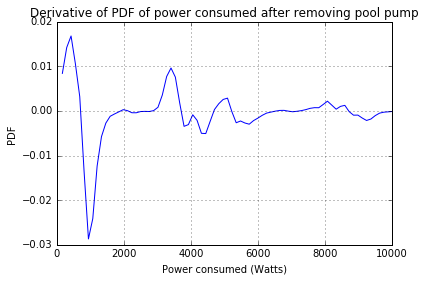
 

|  |  |
| --- | --- |
| **PEAKS:** | **TROUGHS:** |
| 562.577 | 1381.115 |
| 2004.467 | 2685.222 |
| 3253.32 | 3518.234 |
| 3717.421 | 4173.99 |
| 4704.455 | 5769.063 |
| 6099.675 | 6174.701 |
| 6306.05 | 6671.245 |
| 7802.093 | 8725.949 |
| 8838.431 | 9027.299 |
| 9160.823 | 9477.464 |
| 9688.319 | 10664.01 |
| 10961.16 | 11584.2 |
| 11778.9 | 12103.4 |

1. If we look at the power histogram now, it suggests that there are either 2 or 3 main appliances.



|  |  |
| --- | --- |
| **Resulting Peaks** | **Resulting Troughs:** |
| 562.577 | 3518.23 |
| 3717.42 | 4704.45 |
| 5769.06 | 6671.24 |
| 8838.431 | 10664.01 |



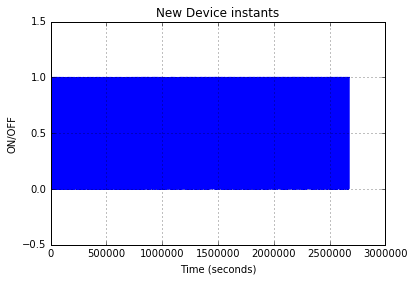
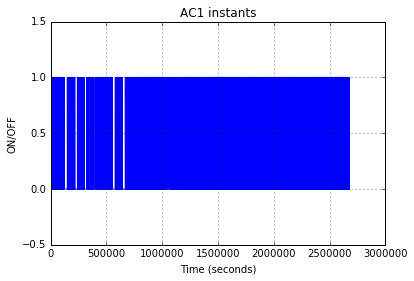
For the case of 2 appliances, the associated power levels are around 3.7KW, and 5.8KW. Since we know from the problem statement that the available devices are refrigerator, AC1, and AC2 (at 200W, 2.5KW and 4KW), we conjecture that there is a third device of around 1.5KW that is only operational outside the 3 hour period of pool pump (and for some reason is perfectly aligned with the usage of AC1 and AC2). **Pretty strong assumption, if you ask me. However, I am still going with it to work under the construct of the “given” problem statement. Personally, I would change the AC1 and AC2 power ratings to be around 3.5KW and 5KW.**

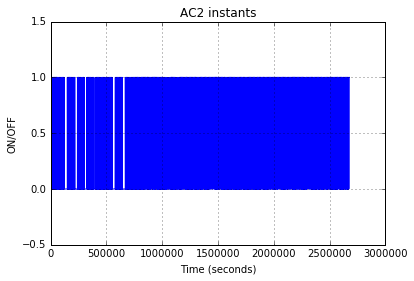
1. We now do threshold-based decisions based on the relaxation that there is a third device with power rating of 1500W. We select 1500W only because the problem statement only mentioned AC1, AC2, refrigerator and pool pump as the main appliances. **This means that our algorithm will once again be confused between AC2 and AC1+third device, and that is an error we have to live with.** We select the decision boundaries at the middle of the discrete values that the data set is supposed to take, making this a ML based estimator (based on a uniform data). Note that we can reduce noise by looking at the peaks and troughs and selecting the decision boundary based on it. However, since our “assumed” appliances are not matching up with the devices we infer from the available data, we will provide the algorithm based on the provided information in the problem statement.

The assumption we will make is that any power consumption value near the mark of 4000W will be termed as AC2 only (and not new device + AC1). If the power value lies between the low and high threshold, then we assume that the corresponding appliances are ON.

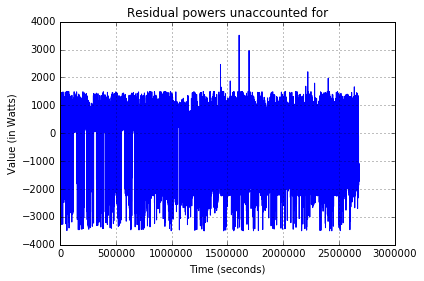
|  |  |  |  |
| --- | --- | --- | --- |
| Category | Power Consumption (W) | Low threshold | High threshold |
| New device | 1500 | 1500 | (1500+2500)/2 |
| AC1 | 2500 | (1500+2500)/2 | (2500+4000)/2 |
| AC2 | 4000 | (2500+4000)/2 | (4000+5500)/2 |
| New device + AC2 | 5500 | (4000+5500)/2 | (5500+6500)/2 |
| AC1+AC2 | 6500 | (5500+6500)/2 | (6500+8000)/2 |
| New device + AC1 + AC2 | 8000 | (6500+8000)/2 | Inf |

This results in the following ON/OFF timings for AC1, AC2 and some additional pool pump instants.



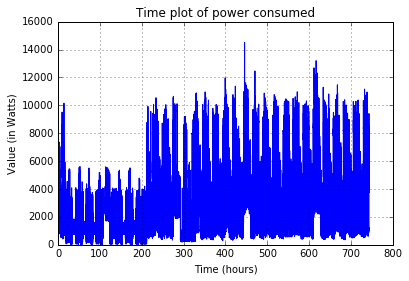
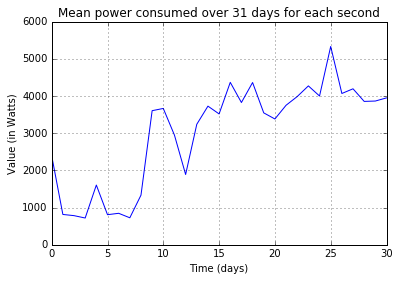
The plot of residual powers after removal of all the above categories are as follows. This contains the power consumed by the refrigerator (<200W)



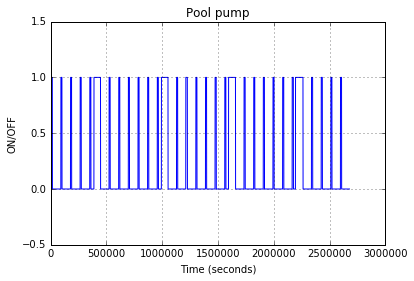
The negative values here is because for a range of values between low/high threshold, we remove the power level equal to the corresponding appliances’ power rating. As mentioned earlier, we can reduce the error by using the decision boundaries based on the peaks and troughs in the data.

# Additional observations/Interesting visualizations

**Observation 1:** After 8 days, the average power consumption increases, with values frequency hovering around 8KW more frequently. This is due to increased usage of AC2.

**** 

**Observation 2: Pool pump is used for extended durations every 7th day.**



Note the extended duration in the above plot for every 7th transition from 0->1 state.

# Computational complexity of your technique(s)

The algorithm requires the following steps:

* 1. Clean up to fill missing seconds and corresponding power data - O(n)
  2. Find minimum over an entire day to find the duration for which water pump is on – O(n)
  3. Calculation for each day, the exact time duration each appliance was ON and removal of its power consumption from the data – O(n) for comparison with threshold and adjustment of power consumption level.

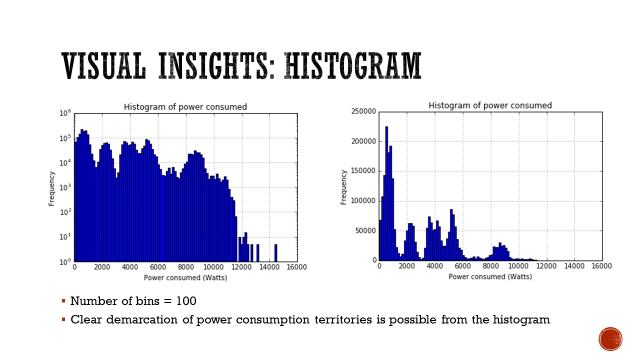
The best-possible computation complexity of our technique is O(n).

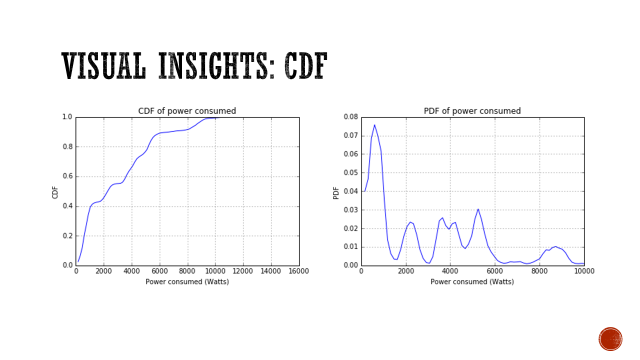
Note that calculation complexity to find the minimum of an array of size n is O(n). Since we use calculation of minimum extensively in our implementation, **the function extract\_and\_remove() actually has a complexity of O(n^2) making the complexity of the entire algorithm O(n^2)**. While all operations can also be done in O(n) duration by using a simple “for” loop, we chose not to do that because we use Python’s built-in vectorization commands. This makes the algorithm runs much faster in practice than a serialized “for” loop doing the same thing.

# How would you solve this if you weren't provided the appliance descriptions for the home?

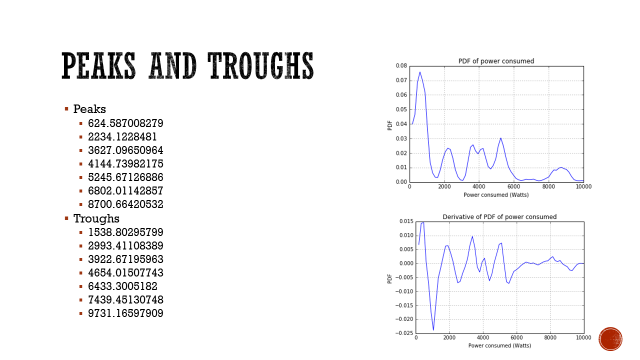
If the appliance descriptions were not available, I would follow the following steps:

1. Using the histogram, get the smoothened CDF and PDF of the available power-series data. See the 2 slides below:



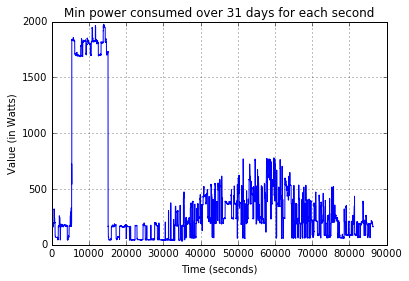


1. Using the PDF from the above plot, one can obtain the thresholds on which various appliances (and their combinations) operate:

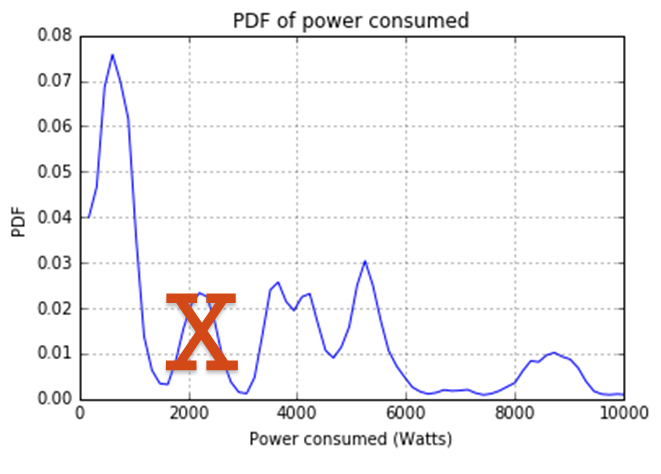


Note that for 7 peaks, the number of appliances can only be between 3 or 7. Also note that this shows that the minimum power consumption is occurring around 624W (which is larger than the refrigerator usage of 200W). This suggests that the data has some other appliances that were not mentioned in the problem statement.

1. I would now follow our original algorithm by finding the minimum power consumption in each second of 1 day minimized over all 31 days and eliminating one of the appliances (pool pump in this case). This shows that there is 1 device (+ minimum residual power usage) around 1700W that is ON for 2.7 hours. This corresponds to the peak 2234W in the above set of all peaks. This is because here we are plotting the minimum value possible. The actual value is higher than the minimum value. The corresponding mean power value of the appliance can be selected as the next higher value of all possible peaks listed above.



1. We can then remove the contribution of all values from 1538W-2993W in that time region for each day. This means that now the resulting pdf plot will not have the second peak.



|  |  |
| --- | --- |
| **PEAKS:** | **TROUGHS:** |
| **562.577** | **1381.115** |
| 3518.234 | 3717.421 |
| 4173.99 | 4704.455 |
| 5769.063 | 6099.675 |
| 8838.431 | 10664.01 |

Now, the first peak is much lower than the rest of the peaks. So, if we treat it as one device that has very low power consumption (around 562W), we are left with 3-4 peaks. This can happen for either 2-4 devices. However, we notice that the largest peak (8838W) is within +/- 600W (minimum peak) to the sum of the rest 2 peaks (3700W + 4700W). So, it is more likely that there are 2-3 devices. Since the difference between the 2 peaks around 4000W is of the order of 500W, the 2 peaks could be due to switching ON of the 1st device (with power 562W) when the 2nd device (with power slightly less than 4KW) simultaneously. So, we can assume that there are 2 devices ON – one with power rating of ~3.7KW and other with power rating of 4.7KW.

1. The final step is to use thresholding based algorithm (based on the thresholds in the above table) to find the time instants of switching ON/OFF of each device (similar to how we used earlier).