Part I - Introduction to the Data

September 24, 2016

1 Introduction

The actual order I did things is not exactly as presented, but it's pretty close. This section covers the first few days, which I mainly spent getting a feel for the data. Included in this part are some simple plots and aggregate statistics. The actual predictions can be found in parts II and III, while some other interesting features of the data can be found in part IV.

1.1 Preprocessing

Nothing too interesting as of yet, just some imports, data extraction, and variable definitions. Some of the residences didn't use all 2880 intervals, so the `dropna()` function was called in order to guarantee all the rows would be filled (this makes it easier to deal with numpy, since most numpy functions called with any nan's will return nan). Two of the more useful variable definitions were `houses_with_ev` and `houses_without_ev`, which list the residences with or without electric vehicles in terms of the house IDs. These can be used to filter pandas DataFrames rather easily.

In [1]:
```
%matplotlib inline

import pandas as pd
import matplotlib.pyplot as pl
import numpy as np
import scipy.stats as stats
import scipy.stats.mstats as mstats
import scipy.interpolate as interpolate
import sklearn.neighbors as neighbors
import random

# DataFrame where index = intervals, columns = house ids
usages = pd.read_csv('./data/EV_train.csv', index_col='House ID').transpose().dropna()
when_charging = pd.read_csv('./data/EV_train_labels.csv', index_col='House ID').transpose()[0:l

# make the index a bunch of integers, instead of 'Interval_1, Interval_2, ...'
usages.index = range(len(usages.index))
when_charging.index = range(len(when_charging.index))

# Int64Index
houses = usages.columns
intervals = usages.index
owns_ev = when_charging.sum() > 0

# Series where index = house ids, values = Bool
houses_with_ev = houses[when_charging.sum() > 0]
houses_without_ev = houses[when_charging.sum() == 0]
```

```
num_with_ev = len(houses_with_ev.T)
num_without_ev = len(houses_without_ev.T)
num_total = num_with_ev+num_without_ev
fraction_with_ev = float(num_with_ev) / float(num_total)
fraction_without_ev = float(num_without_ev) / float(num_total)

# intervals in units of days
days = np.linspace(0, float(len(usages.index))/(24.*2.), len(usages.index))
```

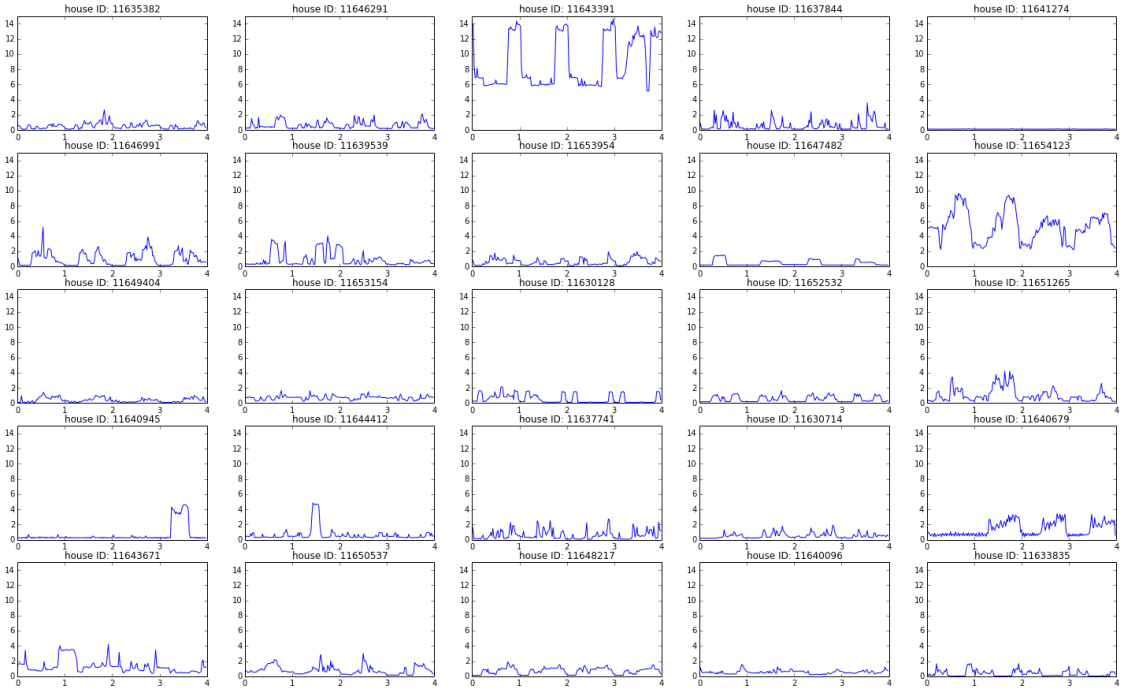
1.2 Looking at Individaul Power Consumptions

To get a feel for the power consumption of a typical household, the first four days of power consumption were plotted for a random sample of households, split into those with and without electric vehicles.

In [2]:
```
grid_size = 5
house_sample_size = grid_size ** 2

random.seed(0)
house_sample = random.sample(houses, house_sample_size)
house_with_ev_sample = random.sample(houses_with_ev, house_sample_size)
house_without_ev_sample = random.sample(houses_without_ev, house_sample_size)

intervals_to_plot = 48*4
small_interval = intervals[0:intervals_to_plot]
days_to_plot = days[0:intervals_to_plot]
```

First up are the households without electric vehicles.
The x-axis is which day, and the y-axis is kWh consumed per 30-minute interval.
The y-axis was also scaled consistently so the usages can be compared with each other.

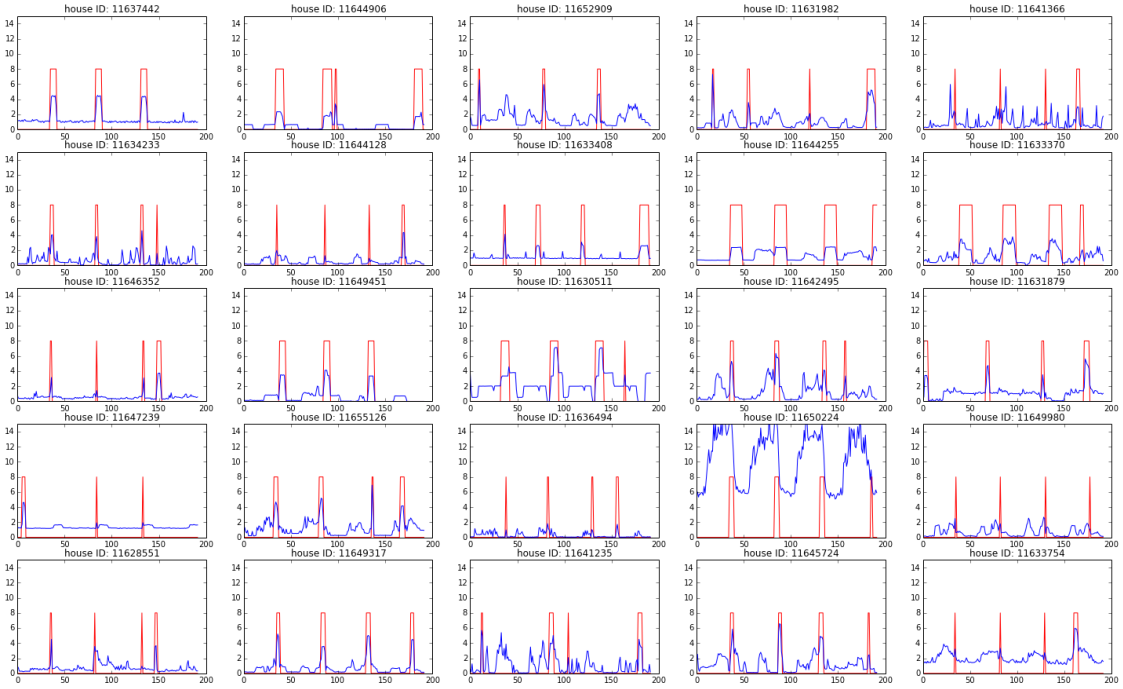
In [3]:
```
f = pl.figure(figsize=(25, 15), dpi=80)
for i,house_id in enumerate(house_without_ev_sample):
    curr_usage = usages[house_id]
    pl.subplot(grid_size, grid_size, i+1) # subplots have 1-based indexing
    pl.plot(days_to_plot, curr_usage[small_interval])
    pl.title('house ID: ' + str(house_id))
    pl.ylim([0, 15])
    pl.xticks([0, 1, 2, 3, 4])
pl.show()
```



There's some interesting patterns to note. First is the day/night cycle. Many of the usage graphs have exactly 4 peaks and 4 troughs, and some residences have usage patterns which seem to repeat daily. Second, the power consumptions are mostly consistent across households (generally between 0.5 - 2 kWh per interval), but can vary by more than an order of magnitude. For example, the residence in the top-middle is likely using more power than the bottom row combined, while the residence in the top-right appears to use almost no power at all.

Next up is the same plot, except for households with electric vehicles. One extra add-on for these plots is the red overlay, which shows when people are charging their electric vehicles (note the red overlay is for visual purposes only, the "scale" doesn't represent anything).

```
In [4]: f = pl.figure(figsize=(25, 15), dpi=80)
        for i,house_id in enumerate(house_with_ev_sample):
            curr_usage = usages[house_id][small_interval]
            curr_charging_data = when_charging[house_id][small_interval]
            pl.subplot(grid_size, grid_size, i+1) # subplots have 1-based indexing
            pl.plot(curr_charging_data*8, color='red')
            pl.plot(curr_usage, color='blue')
            pl.title('house ID: ' + str(house_id))
            pl.ylim([0, 15])
        pl.show()
```



In addition to the features listed earlier, these households also have short, sudden "pulses" corresponding to when an EV is charging. It is worth noting that not every pulse is associated with a charging EV, and not every charging EV is associated with a pulse (the top-rightmost plot includes examples of both). However, as long as there exists enough overlap between EVs and pulses, the pulses should carry enough information to make predictions about when a residence is charging an EV.

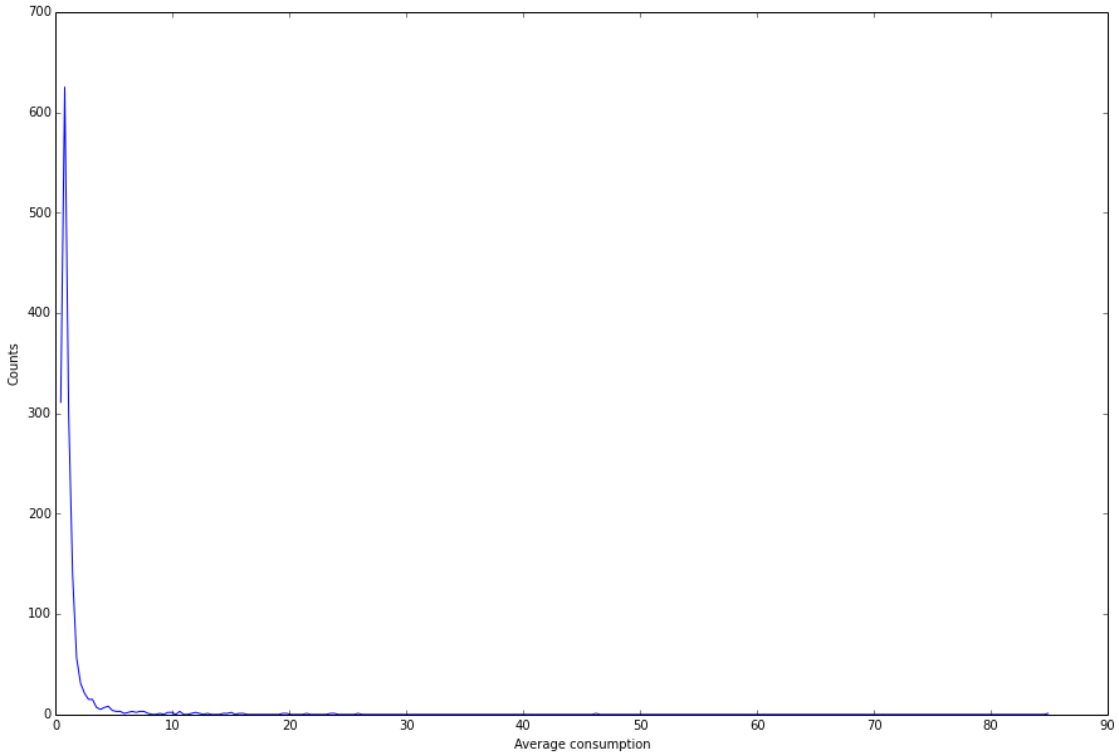
1.3 Looking at Aggregate Statistics

In order to better understand the data, another useful step is to examine some of the aggregate statistics about power usage. What is the average power consumption per household? What are their standard deviations? How do they compare when split into households with or without EVs? It's difficult to know beforehand which (if any) of these particular statistics will have any predictive power, but their simplicity makes them a good starting point.

```
In [5]: stat = pd.DataFrame(index=houses)
        stat['sum'] = usages.apply(np.sum)
        stat['average'] = usages.apply(np.average)
        stat['std'] = usages.apply(np.std)
```

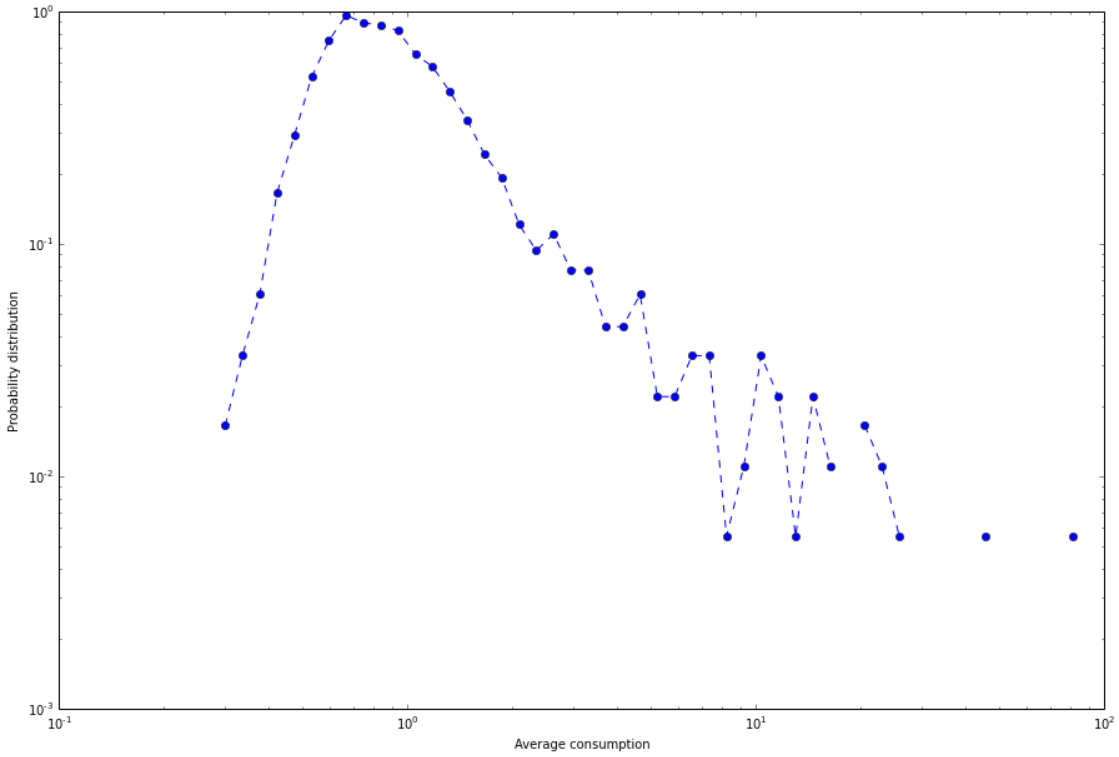
The first statistic plotted below is a histogram of the average 30-minute usage per household.

```
In [6]: hist = np.histogram(stat['average'], bins = 250)
        counts = hist[0]
        bins = hist[1]
        xs = (bins[1:] + bins[:-1]) / 2
        pl.figure(figsize=(15, 10))
        pl.plot(xs, counts)
        pl.xlabel('Average consumption')
        pl.ylabel('Counts')
        pl.show()
```



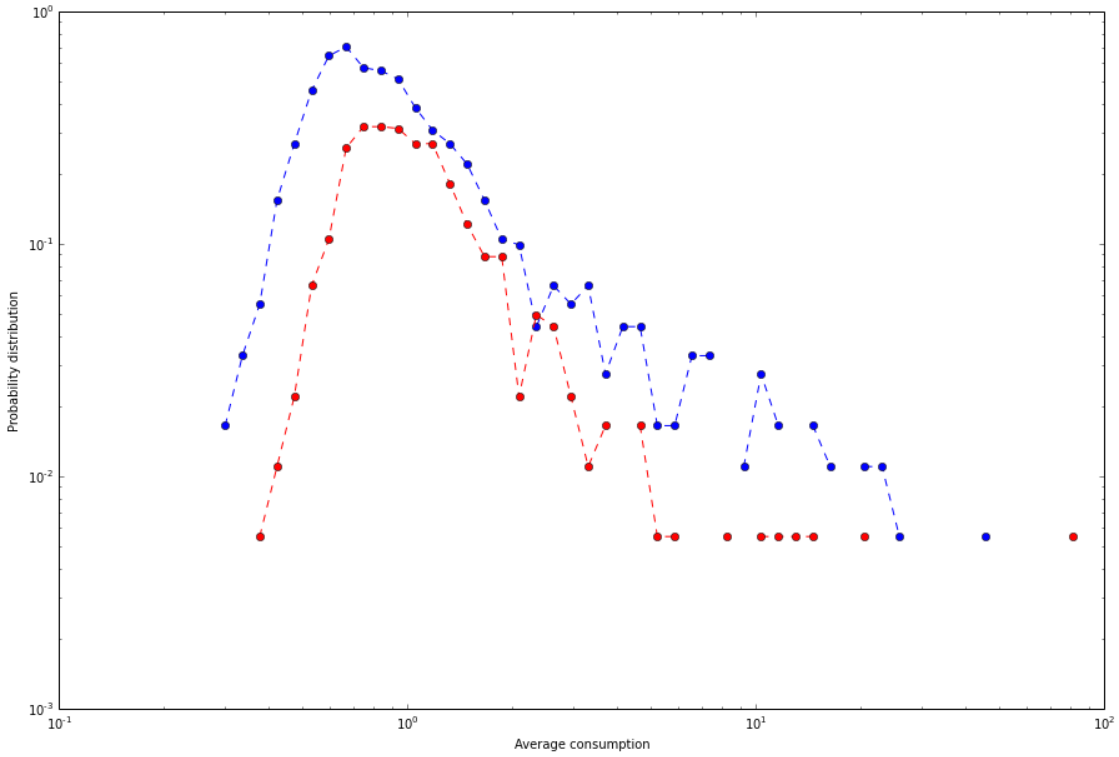
Interesting, it starts really high and falls really fast. Maybe this would convey more information if plotted in logspace. `logbins` is a helper function for generating and plotting histograms in logspace. It also normalizes the histogram, so the returned values represent a probability distribution rather than a list of counts.

```
In [7]: def logbins(data, num_bins=50, bins=None):
            if bins == None:
                bins = np.logspace(0.999*np.log(min(data)), 1.001*np.log(max(data)), num=num_bins+1, bas
            xs = (bins[:-1]+bins[1:]) / 2
            ys = np.histogram(data, bins)[0]
            ys = ys.astype(np.float)
            delta = np.log(xs[1]) - np.log(xs[0])
            ys = ys / (len(data)*delta) # normalize
            return (xs, ys)

In [8]: xs, ys = logbins(stat['average'], num_bins=50)
        xs_i = np.logspace(np.log(min(xs)), np.log(max(xs)), 100, base=np.e)
        pl.figure(figsize=(15, 10))
        pl.loglog(xs, ys, linestyle='--', marker='o')
        pl.xlabel('Average consumption')
        pl.ylabel('Probability distribution')
        pl.show()
```



Interesting, the average household consumption appears to follow a log-normal distribution. Next is the same plot, except split into households with or without electric vehicles. (My initial prediction was that households with EVs would use more power on average, but I wasn't sure if the effect would be strong enough to build a classifier.)

```
In [9]: num_bins = 50
        bins = np.logspace(0.999*np.log(min(stat['average'])), 1.001*np.log(max(stat['average'])), num=n

        xs1, ys1 = logbins(stat['average'][houses_without_ev], bins=bins)
        xs2, ys2 = logbins(stat['average'][houses_with_ev], bins=bins)

        # normalize to total
        ys1 *= fraction_without_ev
        ys2 *= fraction_with_ev

        pl.figure(figsize=(15, 10))
        pl.loglog(xs1, ys1, linestyle='--', marker='o', color='blue')
        pl.loglog(xs2, ys2, linestyle='--', marker='o', color='red')
        pl.xlabel('Average consumption')
        pl.ylabel('Probability distribution')
        pl.show()
```

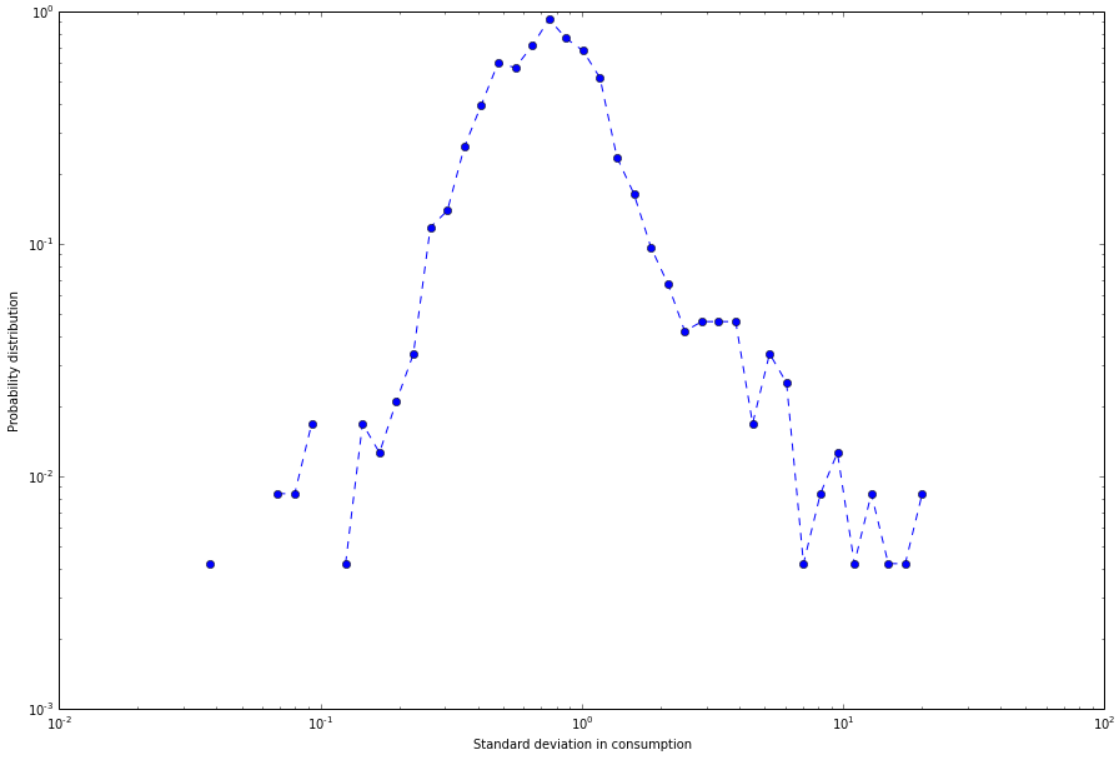


In the plot above, the probability distributions are normalized with each other, so that integrating both and summing the results would equal 1. Both the households with EVs and without EVs follow their own log-normal distributions, but the households with EVs peak at a higher average consumption. It would appear that owning an EV causes a household to use more power on average.

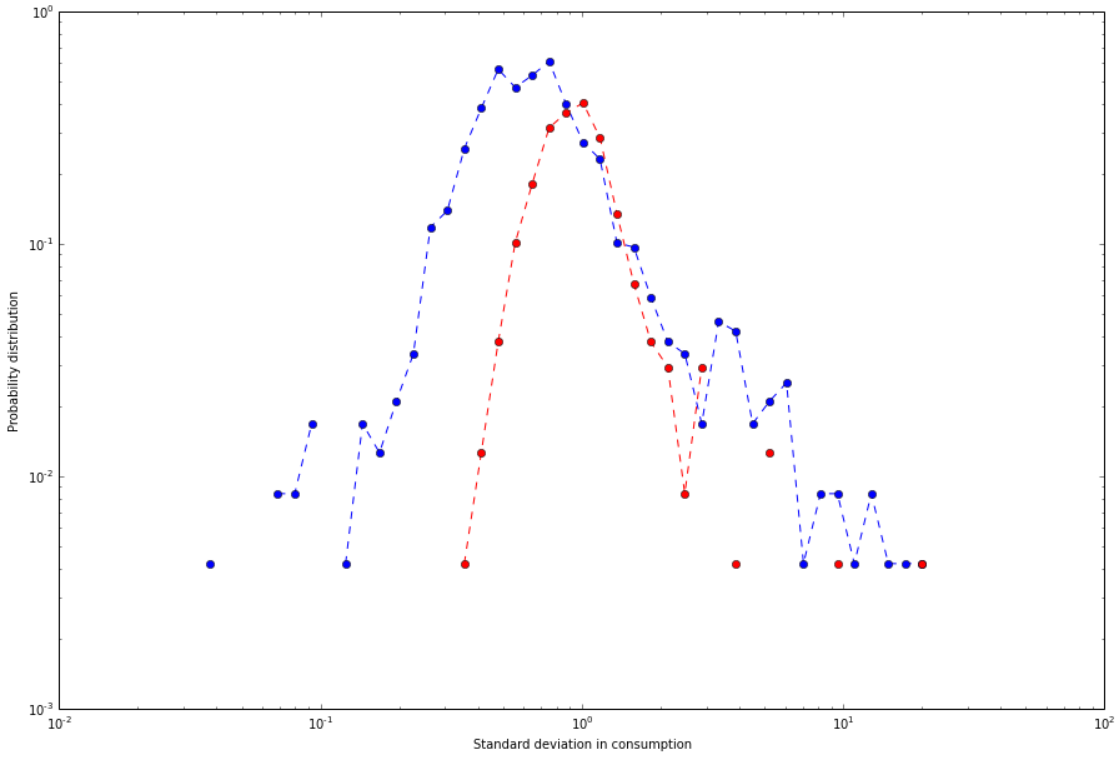
Even though there exists a statistical difference in the average power consumption between houses with EVs and houses without EVs, this effect alone is not strong enough to predict which particular houses own EVs, since the PDF for houses with EVs is always lower than houses without EVs.

Another interesting statistic worth examining is the standard deviation of a household's usage. Below are the same plots as above, except using the standard deviation instead. (My initial prediction was that if a household owned an EV, the consumption graph would contain many 'pulses', which would create a more 'jagged' usage graph, and hence a larger standard deviation. Again I wasn't sure how strong the effect would be, but I had a hunch it would be more significant than average usage).

```
In [10]: xs, ys = logbins(stat['std'], num_bins=50)
         xs_i = np.logspace(np.log(min(xs)), np.log(max(xs)), 100, base=np.e)
         pl.figure(figsize=(15, 10))
         pl.loglog(xs, ys, linestyle='--', marker='o')
         pl.xlabel('Standard deviation in consumption')
         pl.ylabel('Probability distribution')
         pl.show()
```



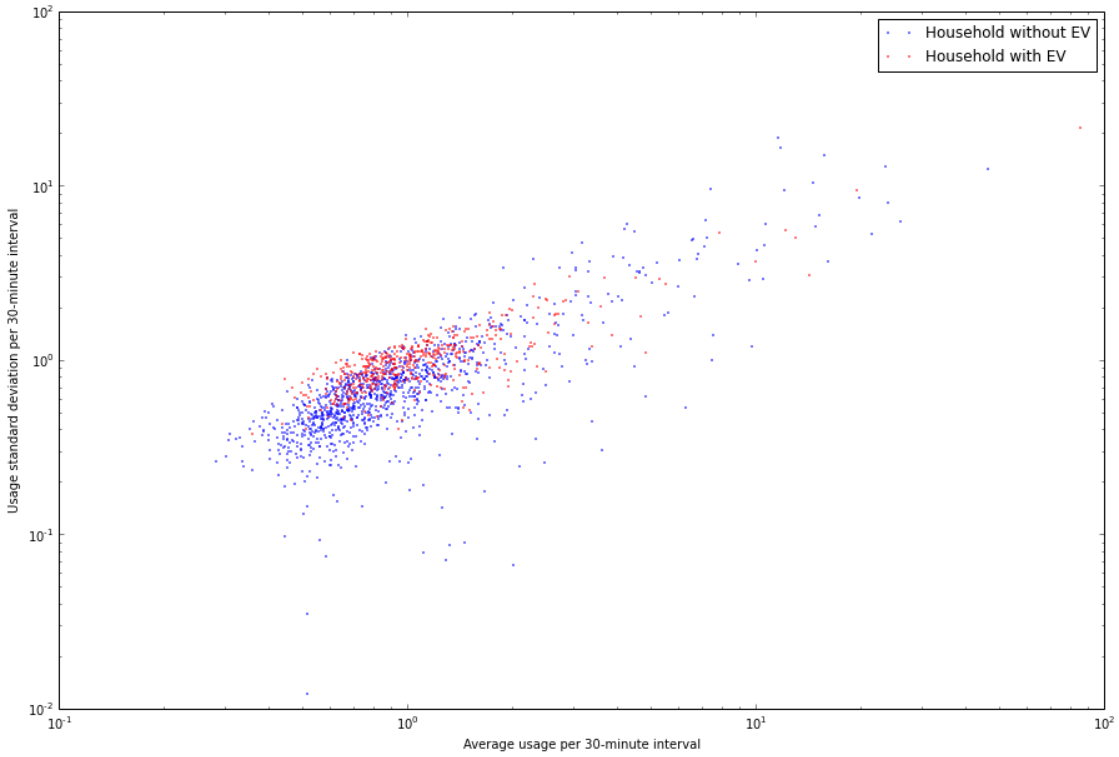

```
In [11]: num_bins = 50
         bins = np.logspace(0.999*np.log(min(stat['std'])), 1.001*np.log(max(stat['std'])), num=num_bins

         xs1, ys1 = logbins(stat['std'][houses_without_ev], bins=bins)
         xs2, ys2 = logbins(stat['std'][houses_with_ev], bins=bins)

         # normalize to total
         ys1 *= fraction_without_ev
         ys2 *= fraction_with_ev

         pl.figure(figsize=(15, 10))
         pl.loglog(xs1, ys1, linestyle='--', marker='o', color='blue')
         pl.loglog(xs2, ys2, linestyle='--', marker='o', color='red')
         pl.xlabel('Standard deviation in consumption')
         pl.ylabel('Probability distribution')
         pl.show()
```



It would appear households with EVs do in general have higher standard deviations in power consumption. Although the difference here is more pronounced than before, using standard deviation alone probably wouldn't make a great classifier (the PDF for houses with EVs never gets much above the PDF for houses without EVs).

Finally, the scatter plot below shows a combination of the two. Each blue dot represents a household without an EV, and each red dot represents a household with an EV.

```
In [12]: pl.figure(figsize=(15, 10))
         without_plt = pl.loglog(stat['average'][houses_without_ev], stat['std'][houses_without_ev],
                                 label='Household without EV', marker='.', markersize=2, linestyle='Non
         with_plt = pl.loglog(stat['average'][houses_with_ev], stat['std'][houses_with_ev],
                                 label='Household with EV', marker='.', markersize=2, linestyle='None',
         pl.legend()
         pl.xlabel('Average usage per 30-minute interval')
         pl.ylabel('Usage standard deviation per 30-minute interval')
         pl.show()
```



There tends to be an upward trend - if a household uses more power, then it will have a higher standard deviation. The households with electric vehicles are clustered a little higher on both axes, but still overlap signficantly with the households without electric vehicles. A classifier based on 2D K-Nearest-Neighbors would probably have low accuracy because of all the overlap.

1.4 Conclusion

The goal of this section was to explore the dataset at a high level and brainstorm some possibilities for classifiers, but not necessarily build a classifier yet. The main finding here was that households with electric vehicles tended to use more power and have higher variances in consumption. While these effects alone didn't seem pronounced enough to build a reliable classifier, they can help serve as a guide for building one. Here are some ideas: - Instead of averaging the usages, maybe average (usage ^ k) for some polynomial k. This would cause higher consumptions to be weighted more. This could be useful since households with EVs probably have more usages at larger values. - Same as above, but (usage - mean) ^ k for some polynomial k (standard deviation is essentially k = 2). - Try these ideas with finite differences or derivatives instead of usages values. - Look at (standard deviation / average) for each residence. If a residence has more pulses, this is a property of the shape of the graph, and not the actual values. Therefore, this statistic should be higher for households with electric vehicles. - Preprocess the data somehow before looking at the statistics. For example, using a high-pass filter could remove the daily swings while preserving any sharp pulses.