Part IV - Other Intersting Features in the Data

September 24, 2016

This part answers some questions that I wanted to answer out of curiousity. Questions related to EVs: - How much power does the typical EV draw? Does it vary from household to household? - How long does a typical charge last? - How much power is drawn from a typical charge?

Questions related to total power consumption: - What does the average power consumption look like over time? Is there a day/night cycle?

In [1]:
```
%matplotlib inline

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

usages.index = range(len(usages.index))
when_charging.index = range(len(when_charging.index))
houses = usages.columns

houses_with_ev = houses[when_charging.sum() > 0]
houses_without_ev = houses[when_charging.sum() == 0]
```

0.1 Questions related to EVs

How much power does the typical EV draw?

An approach here is to look at (average consumption while charging) - (average consumption while not charging) for each household. Something to account for is that the transition intervals must be removed, since they generally count as charging, even if an EV wasn't charging for the entire interval (which would underestimate and add variance to the data).

In [2]:
```
charging_intervals = (when_charging == 1)
not_charging_intervals = (when_charging == 0)
power_while_charging = (usages[houses_with_ev]*charging_intervals[houses_with_ev])
power_while_charging[power_while_charging == 0] = np.nan
```

```
power_while_charging = power_while_charging.apply(np.nanmean)

power_while_not_charging = (usages[houses_with_ev]*not_charging_intervals[houses_with_ev])
power_while_not_charging[power_while_not_charging == 0] = np.nan
power_while_not_charging = power_while_not_charging.apply(np.nanmean)

average_ev_power = power_while_charging - power_while_not_charging
```

The plot below shows a histogram of the average EV power

It would appear different households have different models of EVs, but they are all within the same order of magnitude as each other.

Next, how long do people charge for?

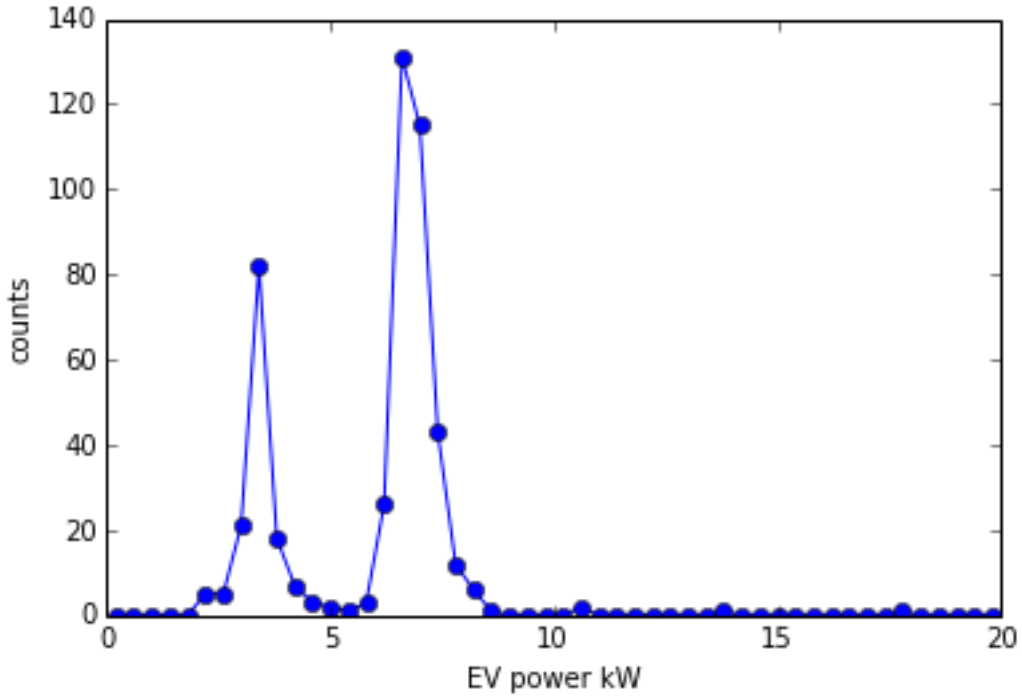
In [3]:
```
def charge_intervals(when_charging):
    intervals = []
    curr_interval = []
    prev_c = 0
    for i,curr_c in enumerate(when_charging):
        if prev_c==0 and curr_c==1:
            curr_interval.append(i)
        elif prev_c==1 and curr_c==1:
            curr_interval.append(i)
        elif prev_c==1 and curr_c==0:
            intervals.append(curr_interval)
            curr_interval = []
        prev_c = curr_c
    if prev_c==1:
        intervals.append(curr_interval)
    return intervals

def average_charge_length(when_charging):
    intervals = charge_intervals(when_charging)
    interval_lengths = [len(i) for i in intervals]
    return np.average(interval_lengths)

def power_during_charge(usage, when_charging, background):
    curr_charge_intervals = charge_intervals(when_charging)
    powers = [[usage[i]-background for i in interval] for interval in curr_charge_intervals]
    return powers

def average_EV_power(usage, when_charging, background):
    powers = power_during_charge(usage, when_charging, background)
    powers = [p[1:-1] for p in powers]
    powers = filter(lambda p: len(p)>0, powers)
    powers = [np.average(p) for p in powers]
    power = np.average(powers)
    return power

def total_EV_consumption(usage, when_charging, background):
    powers = power_during_charge(usage, when_charging, background)
    consumptions = [sum(p) for p in powers]
    consumption = np.average(consumptions)
    return consumption
```

In [4]:
```
charge_lengths = when_charging[houses_with_ev].apply(average_charge_length)
```

```
average_powers = [average_EV_power(usages[h], when_charging[h], power_while_not_charging[h]) fo
consumptions = [total_EV_consumption(usages[h], when_charging[h], power_while_not_charging[h]) f
```

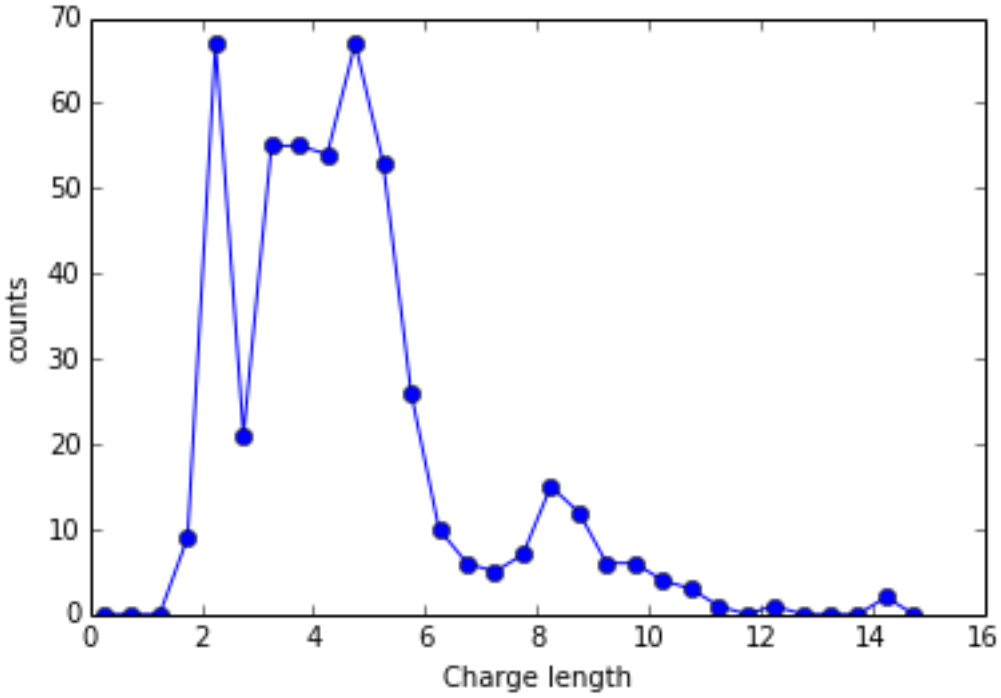
The plot below shows a histogram of the power increase from an EV.

In [5]:
```
bins = np.linspace(0, 10, 51)
xs = (bins[1:] + bins[:-1])/2
xs *= 2 # units of kW
ys = np.histogram(average_powers, bins)
ys = ys[0]
pl.plot(xs, ys, marker='o')
pl.xlabel('EV power kW')
pl.ylabel('counts')
pl.show()
```



This is really cool, since it's bimodal. This means there are two types of electric vehicles! Next, how long do people typically charge for.

In [6]:
```
bins = np.linspace(0, 15, 31)
xs = (bins[1:] + bins[:-1])/2
ys = np.histogram(charge_lengths, bins)
ys = ys[0]
pl.plot(xs, ys, marker='o')
pl.xlabel('Charge length')
pl.ylabel('counts')
pl.show()
```

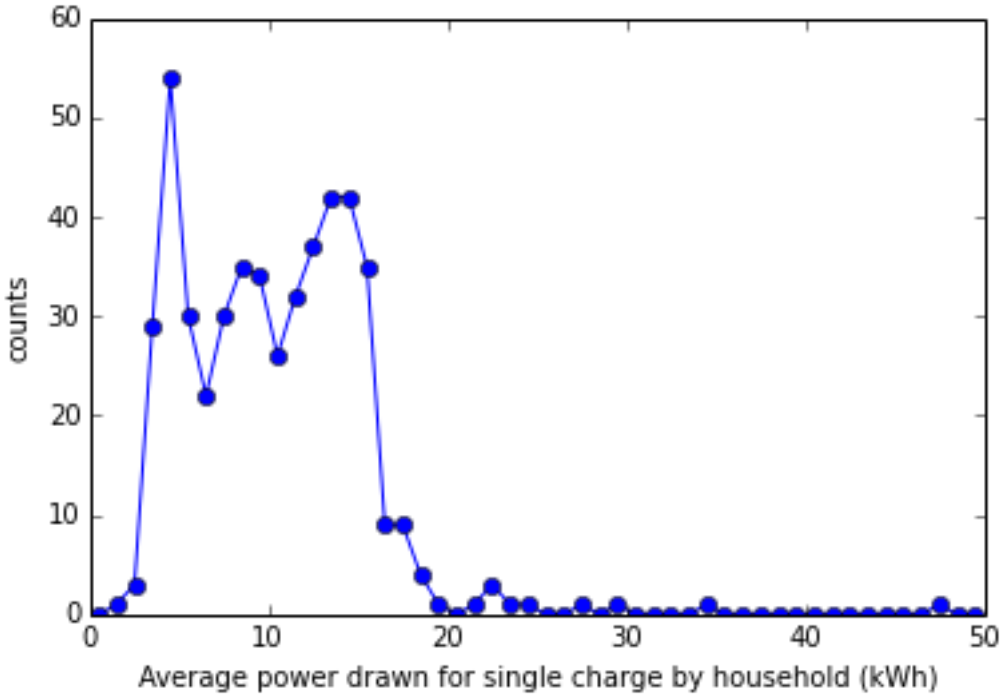


And the average/standard deviation

```
In [7]: print "Average charge length: %.4f" % np.average(charge_lengths)
        print "Standard deviation among houses: %.4f" % np.std(charge_lengths)

Average charge length: 4.6300
Standard deviation among houses: 2.0830
```

This is also interesting, there seem to be several peaks.
Next, how much total power is drawn from a typical charge?

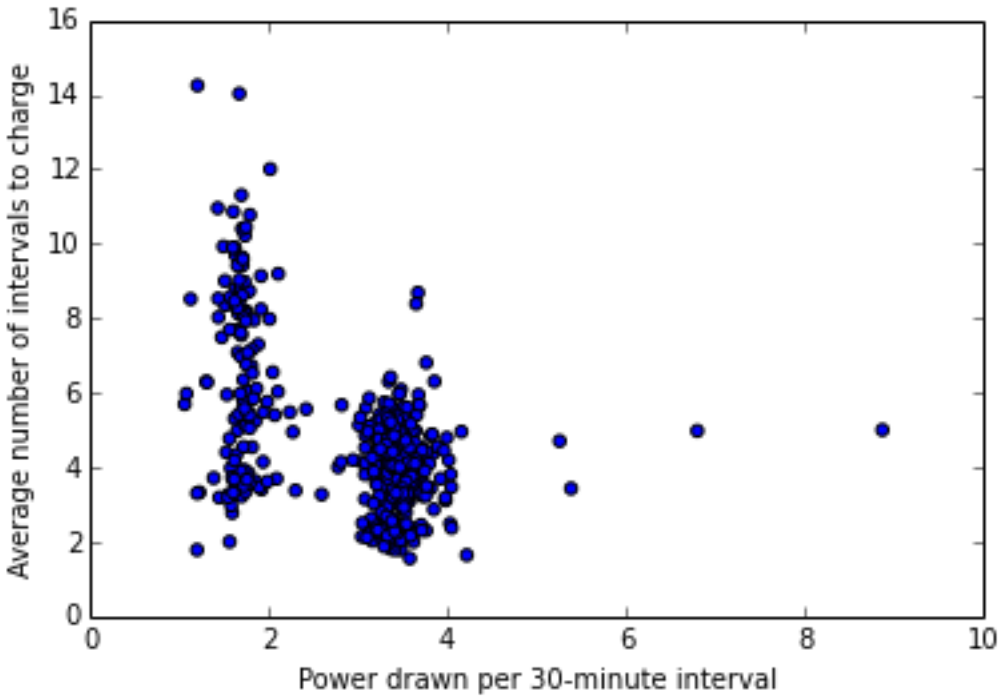
```
In [8]: bins = np.linspace(0, 50, 51)
        xs = (bins[1:] + bins[:-1])/2
        ys = np.histogram(consumptions, bins)
        ys = ys[0]
        pl.plot(xs, ys, marker='o')
        pl.xlabel('Average power drawn for single charge by household (kWh)')
        pl.ylabel('counts')
        pl.show()
```



One last thing, does the average power consumption of an EV correlate with how long it takes to charge?

```
In [9]: pl.scatter(average_powers, charge_lengths)
        pl.xlabel('Power drawn per 30-minute interval')
        pl.ylabel('Average number of intervals to charge')
```

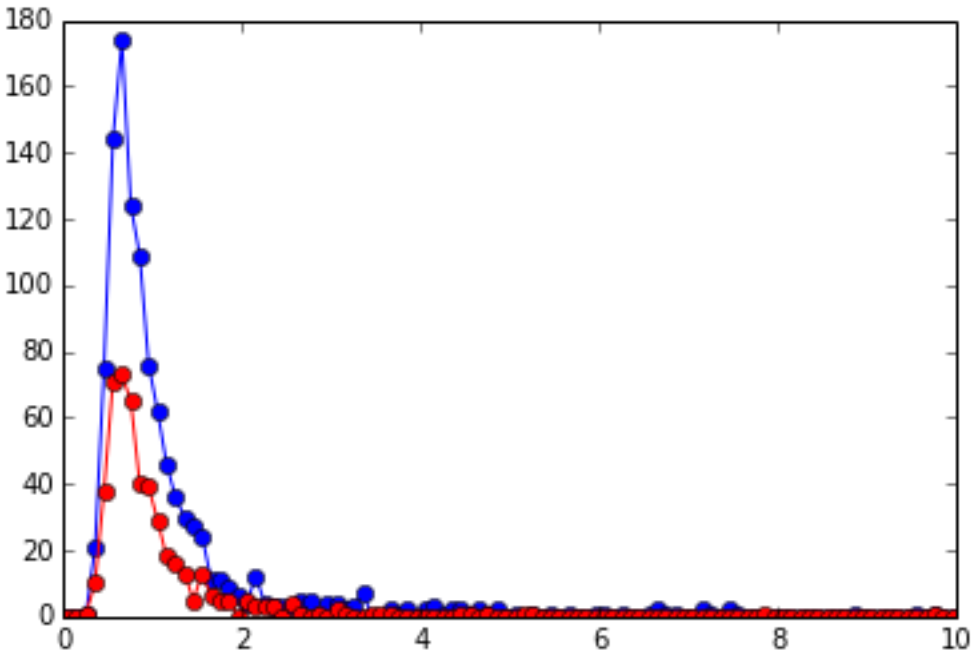
```
Out[9]: <matplotlib.text.Text at 0xae67e8cc>
```



Interesting! It seems the EVs which draw more power also take less time to charge.

One last question - is the increase in power usage for EV owners simply a result of EVs drawing more power, or do people who own EVs also use more power for other things?

```
In [10]: bins = np.linspace(0, 10, 101)
         xs = (bins[1:] + bins[:-1])/2
         non_ev_averages = usages[houses_without_ev].apply(np.average)
         h0 = np.histogram(non_ev_averages, bins=bins)
         h1 = np.histogram(power_while_not_charging, bins=bins)
         pl.plot(xs, h0[0], color='blue', marker='o')
         pl.plot(xs, h1[0], color='red', marker='o')
         pl.show()
```



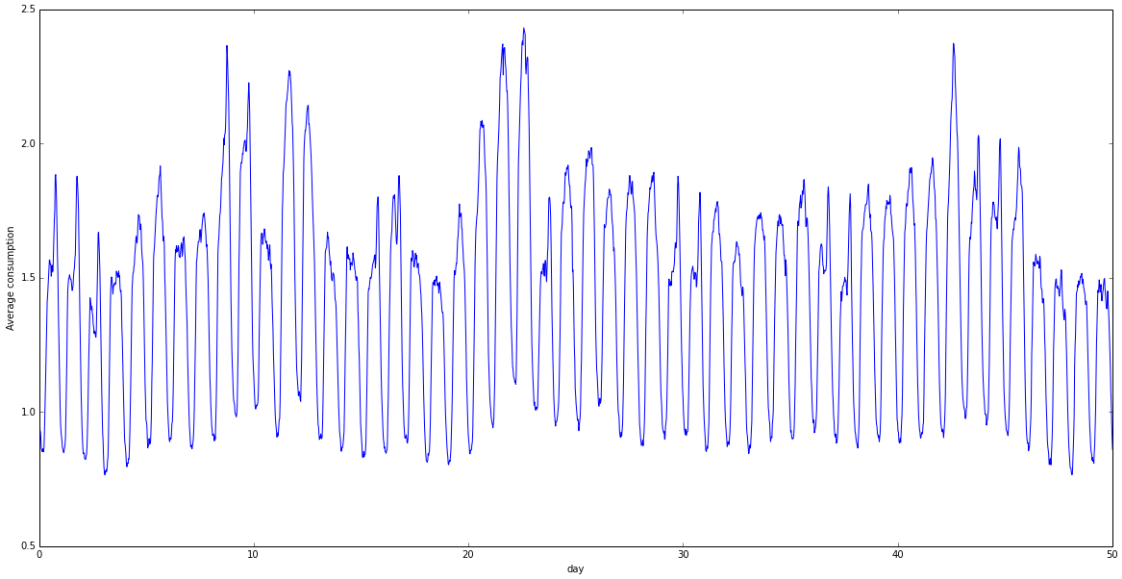

```
In [11]: non_ev_avg = np.average(non_ev_averages)
         non_ev_unc = np.std(non_ev_averages) / len(non_ev_averages)
         with_ev_avg = np.average(power_while_not_charging)
         with_ev_unc = np.std(power_while_not_charging) / len(power_while_not_charging)
         print "houses without EV: %.3f (%.3f)" % (non_ev_avg, non_ev_unc)
         print "houses with EV (while not charging): %.3f (%.3f)" % (with_ev_avg, with_ev_unc)
```

```
houses without EV: 1.395 (0.002)
houses with EV (while not charging): 1.283 (0.008)
```

Nope, it appears the increase in power is solely derived from charging EVs (the houses with EVs could be made more accurate by including the times while charging and subtracting out the power from the car, which could matter if people only charge during the day, for example).

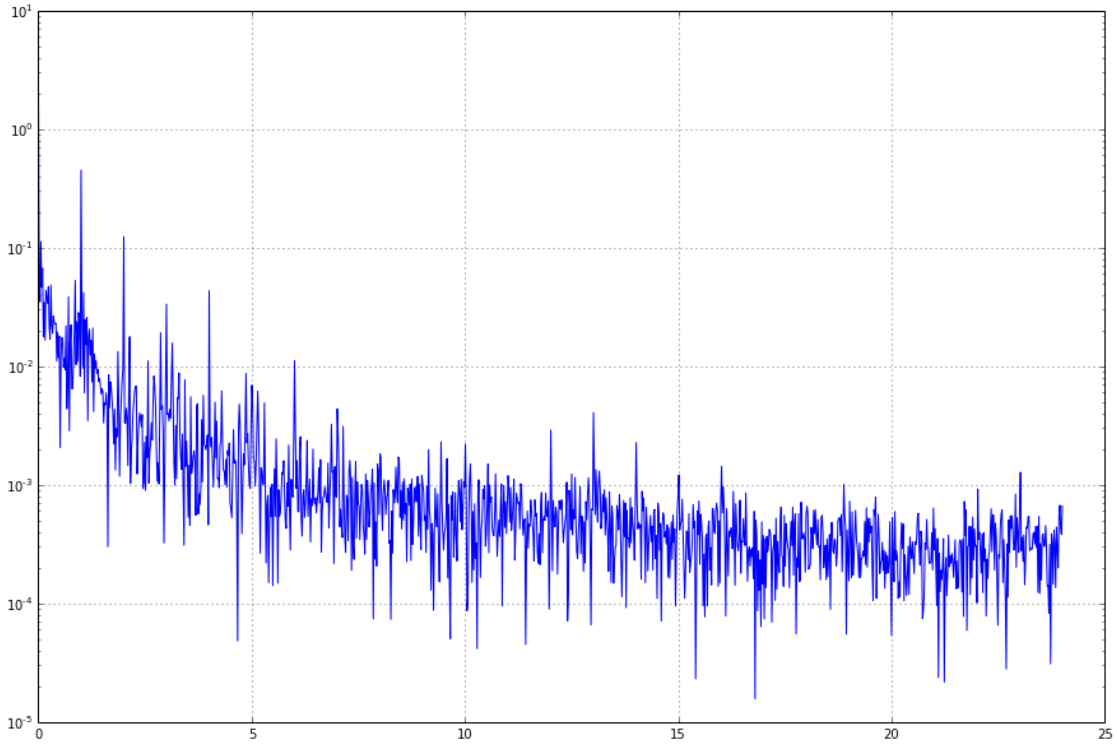
0.2 Power Consumption Averaged Across Households

```
In [12]: pl.figure(figsize=(20,10))
         days = usages.index/48.
         averages = usages.T.apply(np.average)
         pl.plot(days, averages)
         pl.xlabel('day')
         pl.ylabel('Average consumption')
         pl.xlim([0, 50])
         pl.show()
```



Interesting! There's a clear day/night cycle. What are the Fourier Modes?

In [13]:
```
from numpy.fft import fft

N = len(averages)
T = 1.0/48.0
x = np.linspace(0,N*T,N)

f_averages = fft(averages)
xf = np.linspace(0, 1.0/(2.0*T), N/2)

pl.figure(figsize=(15,10))
pl.semilogy(xf, 2.0/N * np.abs(f_averages[:N/2]))
pl.grid()
pl.show()
```



Just as expected - there are large peaks at 1 and 2 days (and other integer days in general).

0.3 Conclusions

The results in the first part (questions related to EVs) include some very interesting results which could potentially improve the accuracy in parts II and III.

In Part II (predicting when an EV is charging), a second-order Markov Model could look for the total change in usage over two timesteps, and see how close it is to the either of the two peaks. It might even be possible to fit a household to which type of EV it owns, and then fit a Markov Model to each type of EV.

In part III (predicting which houses own EVs), it might be possible to use the results of the Markov Model to see if the amount of usage from an EV is close to either of the two peaks, and if not, then the house likely doesn't own an EV.