The Uber Demand Prediction Challenge

Michael Sachs

The Uber demand prediction challenge is the classic problem of predicting future events given some past data. Anyone who has tried to do this for say, the stock market, knows that getting meaningful results can be tricky. Not surprisingly success depends on the regularity of the past data. Below I describe a simple technique for making predictions of future demand on Uber's services given two months of past data. This technique is based mostly on patterns picked out from the data by eye. However, I also describe some more advanced pattern recognition techniques and how they can be applied to Uber data.

The Data

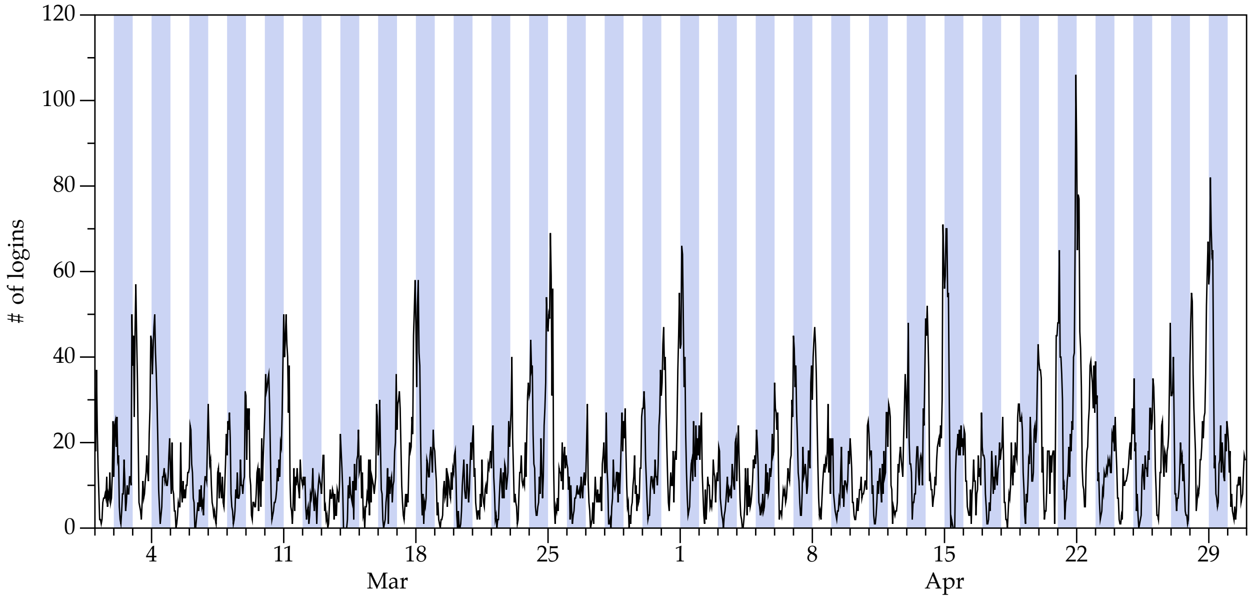

Figure 1: The logins per hour for the two months of data provided.

The data provided for this challenge was a series of timestamps representing login times for Uber users. In order to make sense of this data, I parsed it into a time-series of logins per unit time. Several examples are shown in Figures 1, 2, and 3. Although each of these time-series have interesting features, the one that I used for the analysis that follows was the logins per hour (Figure 1). A detail of the logins per hour for the two week period ending on Sunday, April 22nd is shown in Figure 4. There are many interesting features in this time-series data. The first thing that stands out is that the data are periodic. There is a strong daily cycle with peaks around midnight and troughs around 8am. Clearly Uber is more popular with the bar crowd than with commuters. There is also a weekly cycle which peaks Saturday night. Again this is most likely a bar-crowd effect.

There is another interesting feature that only shows up in Figure 1. It seems there are more logins (especially at peak times) closer to the end of each month. Why this happens is not entirely clear. The data do come from the Washington DC area, so perhaps it has something to do with the pay-day schedules of government employees. However, with only two months of data, it is hard to say whether this is part of an underlying process or just a random fluctuation.

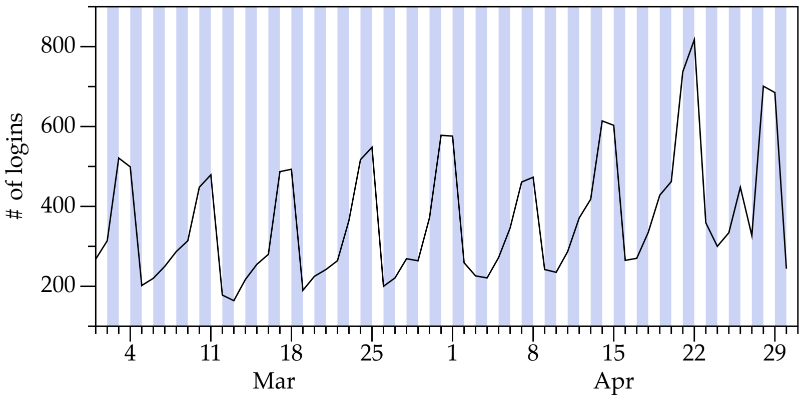

Figure 2: The logins per day for the two months of data provided.

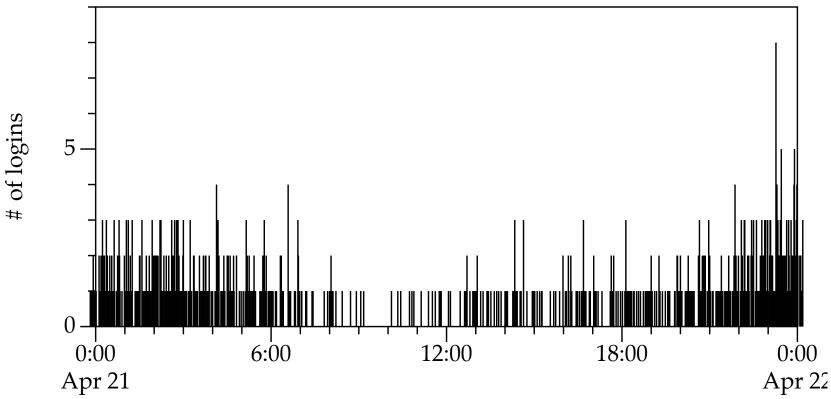

Figure 3: The logins per minute for Saturday, April 21st 2012. This was the busiest day in the dataset.

The Forecasts

The fact that the Uber login data has such a strong daily and weekly periodicity makes basic forecasting fairly simple. Because, for example, Monday, March 5 looks pretty much the same as Monday, April 2, it seems safe to assume that – at least to first order – this will hold

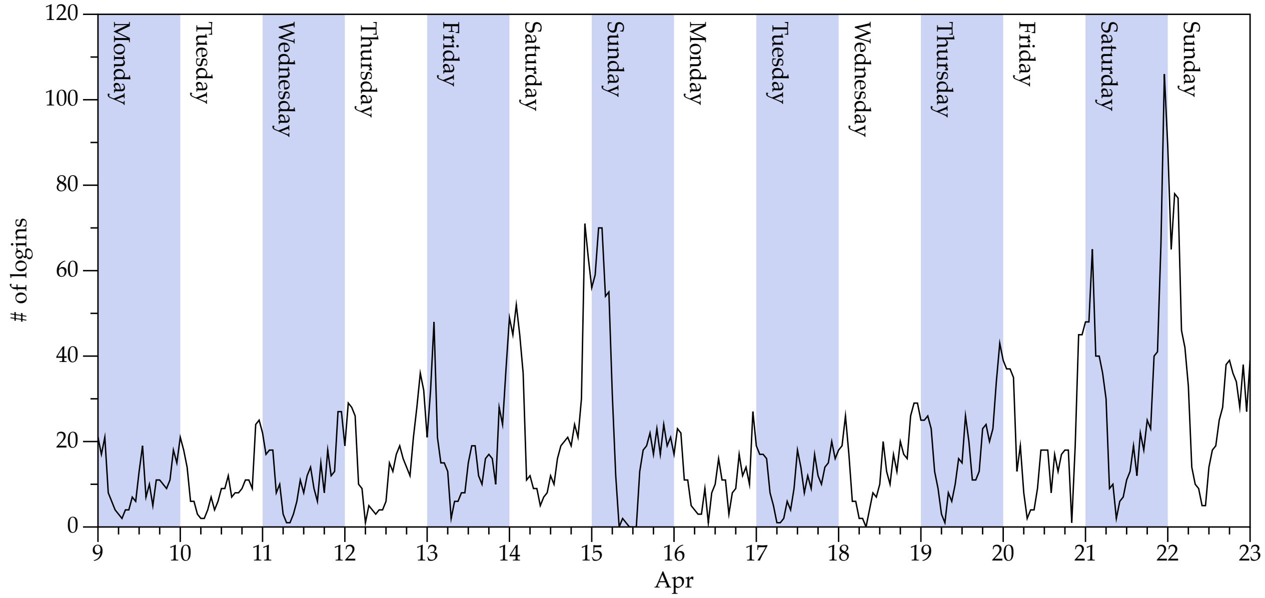

Figure 4: The logins per hour for the two weeks ending on Sunday, April 22nd. The periodic structure is clear at this scale.

for Monday, May 7. Building upon this similarity, my forecasting scheme simply averages each hour of each weekday over the number of instances of that weekday that appear in the data. So, taking Mondays as an example again, there are 9 Mondays in the data, therefore my forecast for Monday at 10am is the average of 9 values of Monday at 10am. This is expressed mathematically in Equation 1:

$$f_{hd} = \frac{\sum_d L_{hd}}{\sum_d}, \tag{1}$$

where f if the forecast, h and d are the hour and day respectively, and L is the login count.

The results of applying this forecasting scheme are shown in Figure 5. The daily and weekly periodicity is clearly captured. The forecast shows high "bar-time" demand, especially on weekends, and low morning commute demand, exactly like the data. However, because the monthly effect that I described above is subtle – and because there are only two months of data – the forecast in Equation 5 and Figure 1 does not include any monthly variations. This effect could be added by superimposing the monthly variations $M(t)$ in Equation 1:

$$f_{hd}(t) = \frac{\sum_d L_{hd}}{\sum_d} + M(t). \tag{2}$$

The form of $M(t)$ could be inferred from the data or based on a model that captures the monthly dynamics.

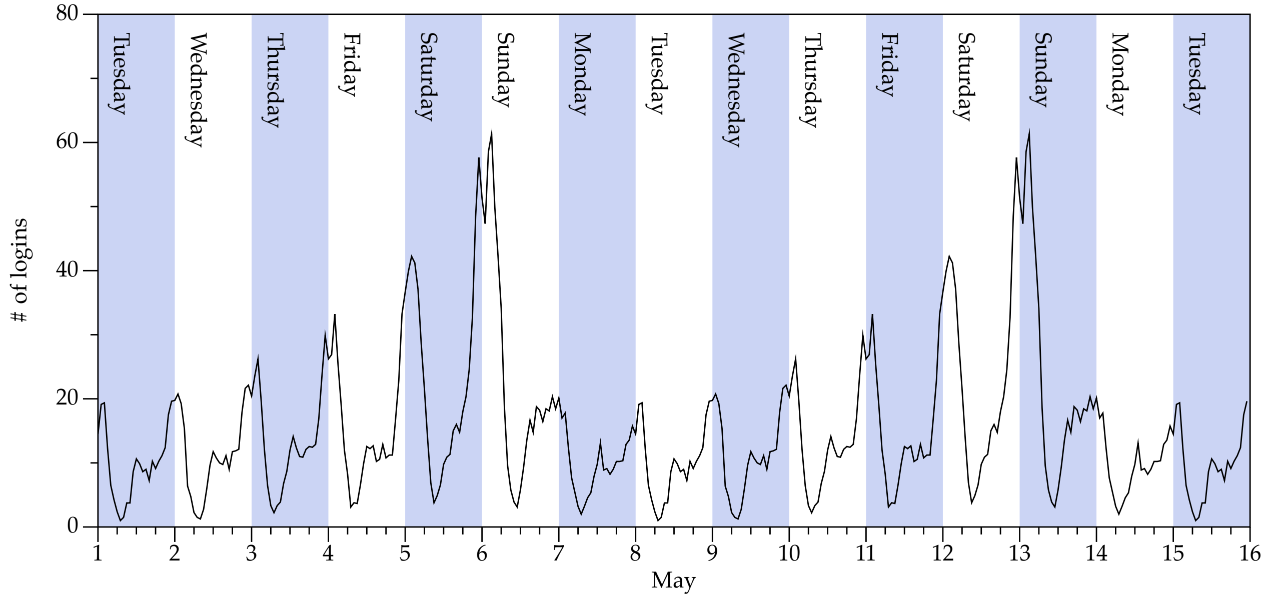

Figure 5: Forecast logins per hour for the time period May 1 2012 - May 15 2012. The daily and weekly periodic structure from Figure 4 is captured in this forecast.

Further Explorations

The forecast described above is effective but fairly basic. I was excited by the data, and wanted to explore some additional techniques that could be used to improve forecasting, but are also just interesting ways to look at time-series data. In what follows I am not drawing any conclusions or suggesting applications. I am merely poking around to see if I can find anything interesting.

Fourier Transforms

The first tool that is used in the analysis of any stationary time series is usually the Fourier transform. This technique is based on the fact that any discrete time series X can be rewritten as a sum of sines and cosines (Equation 3).

$$X_n = \sum_{n=0}^{N-1} x_n \exp^{-i2\pi k \frac{n}{N}} . \tag{3}$$

The x_n are a measure of the amount, or power, of each frequency present in the signal. Equation 3 can be inverted to extract the x_n in terms of the X_n; this is a Fourier transform. The particular algorithm I used is the fast Fourier transform that is implemented as part of the python Numpy package, details can be found at `http://docs.scipy.org/doc/numpy/reference/routines.fft.html` and Cooley and Tukey [1965], Press et al. [2007].

The results of the fast Fourier transform applied to the Uber logins per hour data (Figure 1) are shown in Figure 6. The top of Figure 6 shows the normalized power over all periods in the data while the bottom focusses in on the region marked in blue. The first thing that pops out in Figure 6 is the periodicity mentioned above. The strongest cycle is, as guessed, at the one day mark with the next strongest cycle at the 7 day mark. It is also interesting to note that my observation of a monthly cycle seems to also be correct with some power in the ∼30 and ∼60 day modes. Concentrating on the bottom plot of Figure 6, there are some other active modes that were not immediately apparent, particularly the power in the 0.5 day mode and the 3.5 day mode.

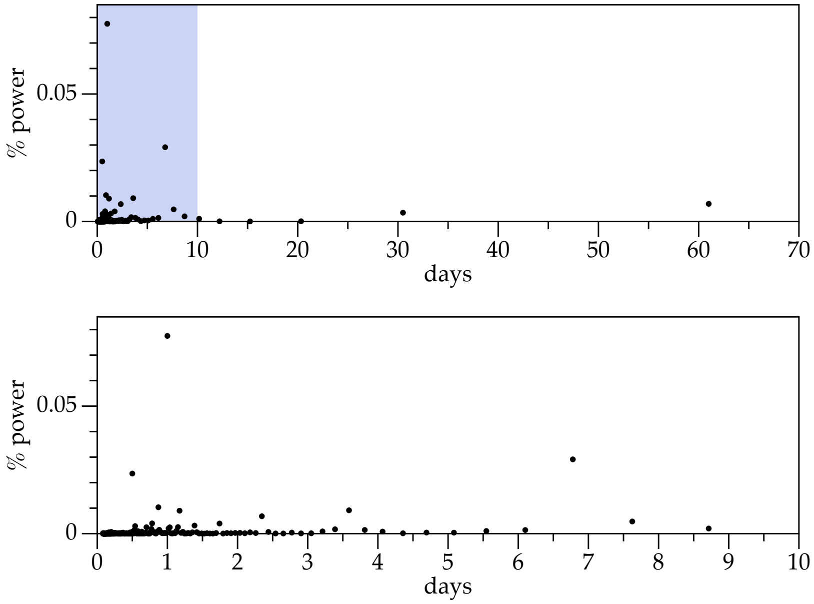

Figure 6: The normalized power spectrum of the Uber logins per hour timeseries data. **Top** The power spectrum for all periods in the data. **Bottom** The power spectrum for the region marked in blue.

Frequency-Magnitude

My dissertation work involved (among other things) looking at scaling relations in complex systems [Sachs et al., 2012]. One particularly important scaling relation is the Gutenberg-Richter frequency-magnitude relation in seismology [Gutenberg and Richter, 1954]. This relation compares the cumulative number of earthquakes N to their magnitudes m:

$$\log N = a - bm, \tag{4}$$

where a and b are constants. The basic idea is that for a certain number of small earthquakes (the exact number depending on the value of b), one can expect a large earthquake.

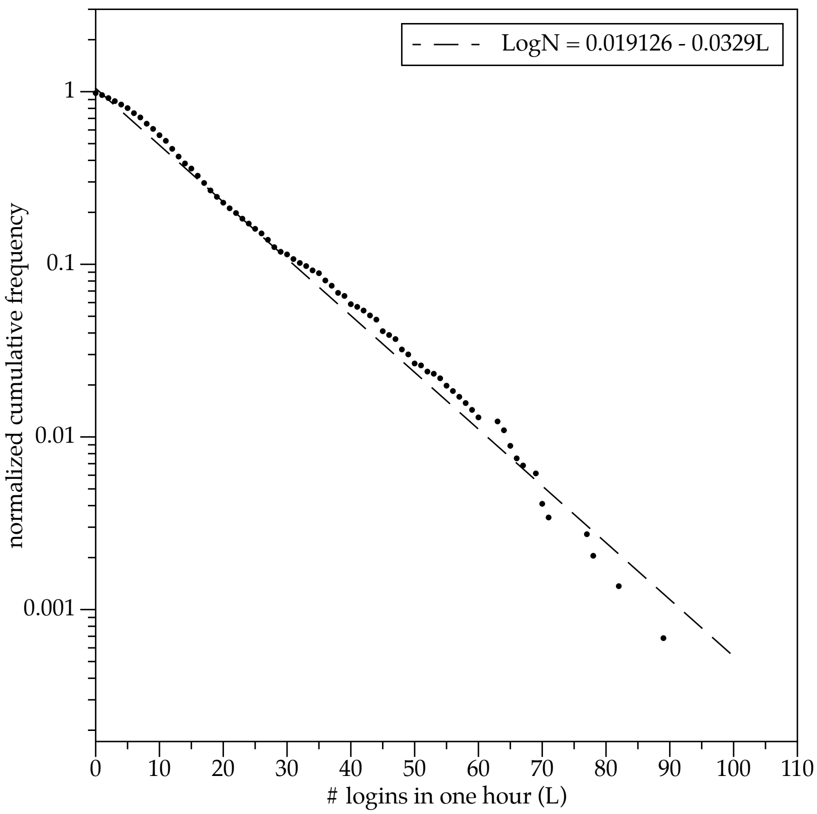

Figure 7: The cumulative frequency-login count for the Uber logins per hour data.

The Uber login data inspired me to see what a "frequency-login count" relation looks like. The results are shown in Figure 7. The basic idea in this figure is that we are counting the number of hours that have login counts greater than or equal to the numbers listed on the x-axis. So, for example, around 40% of the hours in the Uber data have 10 or more logins in them while only about 1% have 50 or more. There is a nice linear relationship that is expressed in Figure 7:

$$\log N = 0.019126 - 0.0329L. \tag{5}$$

Above about $L = 70$ there is a deviation from the linear relationship. This usually indicates some kind of change in the process that generates the data, although here it my just be due to the small sample size.

Shannon Entropy

The idea of entropy in information was first introduced by Shannon [1948] as a measure of the amount of information in a signal. It is usually defined as:

$$H(X) = -\sum_i P(X_i) \log P(X_i). \tag{6}$$

In Equation 6, X is a discrete random variable, and $P(X_i)$ is the probability of the variable being X_i. The logarithm is usually taken as base two, which results in H being in bits. Highly random signals will have a high entropy, while signals that are predictable will have a low entropy. The definition of entropy in Equation 6 can be difficult to apply to signals that are not binary. In order to avoid this difficulty a new definition called permutation entropy [Brandt and Pompe, 2002] can be employed. The permutation entropy is defined as:

$$H(n) = -\sum_i p(\pi) \log p(\pi). \tag{7}$$

$p(\pi)$ is the probability of finding permutations π of order n in the signal. The logarithm is base two. For example: assume that our signal is $x = (4, 7, 9, 10, 6, 11, 3)$. Looking at each pair of numbers – this is order $n = 2$ – there are four pairs where the first number is smaller than the second. There are two pairs where the first number is bigger than the second. These are the only possibilities at $n = 2$. There are six total pairs. The permutation entropy is:

$$H(2) = -\frac{4}{6} \log \frac{4}{6} - \frac{2}{6} \log \frac{2}{6} \simeq 0.918. \tag{8}$$

The permutation entropy in Equation 7 is bounded: $0 \leq H(n) \leq \log n!$. It is convenient to define a new value $h_n = H(n)/\log n!$ which is zero for a completely predictable signal and one for a completely random signal.

The results of applying this to the Uber logins per hour data are shown in Figure 8. Here h_n was calculated for several values of n on a moving window of $T_{win} = 336$ hours (two weeks). It can be tricky choosing the correct value of n to use, if n is too small the signal can look more random than it actually is, if n is too high the signal may not have a representative number of permutations to sample. I have chosen $n = 6$ here (the bold line in Figure 8) because it retains some of the structure of the lower n values but is still small enough to be reasonably well represented in the 336 hour sample. The first two weeks have $h_n = 0$ because of the value of T_{win}.

The first interesting feature in Figure 8 is the fact that h_n changes very little throughout the two month period. This again confirms that the dominant features of the original data are the daily and weekly cycles. Next, there does seem to be a dip towards the end of April. It is not clear if this is reflecting a longer term dynamic or if it is just noise.

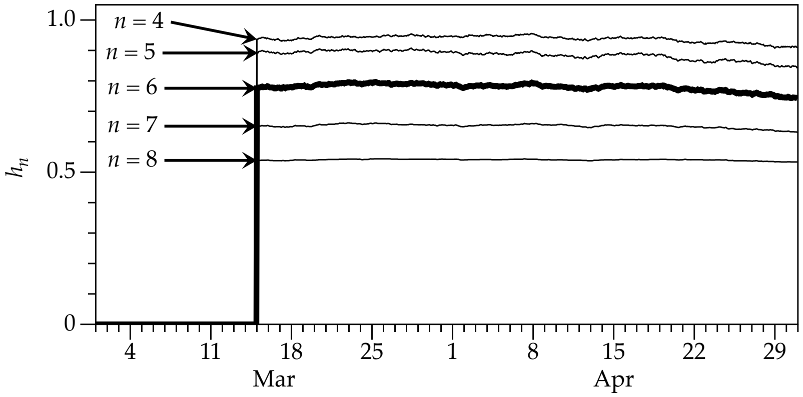

Figure 8: The permutation entropy h_n calculated for various values of n and a moving two week window ($T_{win} = 336$ hours).

Epilogue

Data from Uber's services are extremely interesting. I feel like I have only been able to scratch the surface of what is possible. I would love to explore how more than two months of data would change my results. Also, how would data from different cities compare? There is a trove of information here for cultural anthropologists. It would also be interesting to develop a more comprehensive model of transportation demand based on this data. Lastly, it would be fascinating to include geo-spatial data in these results.

References

C. Brandt and B. Pompe. Permutation entropy: A natural complexity measure for time series. *Phys. Rev. Let.*, 88(17):174102, 2002.

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput.*, 19:297–301, 1965.

B. Gutenberg and C. F. Richter. *Seismicity of the Earth and Associated Phenomena*. Princeton University Press, Princeton, NJ, 1954.

W. Press, S. Teukolsky, W. T. Vetterline, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, chapter 12-13. Cambridge Univ. Press, Cambridge, UK, 2007.

M. K. Sachs, M. R. Yoder, D. L. Turcotte, J. B. Rundle, and B. D. Malamud. Black swans, power laws, and dragon-kings: Earthquakes, volcanic eruptions, landslides, wildfires, floods, and SOC models. *Eur. Phys. J. Special Topics*, 205:167–182, 2012.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, July/October 1948.