Homework 6 (well, week 6)

Due by 9PM October 22nd

Please submit R code, explanations, and / or plots for any section marked with the **[To submit]** heading by emailing jakeporway+itp@gmail.com.

READINGS

- Go back and read through the notes in Lecture 6
- Read up to the section titled "Forecasts using Exponential Smoothing" in http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html

Back to Twitter

In class we looked at the number of NYPD stops during November and explored seasonality as well as identified significant peaks in stops. Let's flex our timeseries muscles using what we learned in class on a set of Twitter data.

You can download a dataset of tweets from http://jakeporway.com/teaching/data/tweets2009.csv that we'll use for this assignment. These are tweets that were collected during a week in June 2009 with just a few basic columns of information – human-readable time of tweet, tweet time in seconds, user screen name, and text of the tweet. There is no header in this data so we need to set the argument "header" to FALSE in read.csv():

```
tweets <- read.csv("http://jakeporway.com/teaching/data/tweets2009.csv",
header=FALSE, as.is=TRUE)
```

Prep

When you first load the data you may notice that it doesn't have column names. Running names(tweets) just returns "V1 V2 V3 V4", the default naming scheme when headers aren't given. Hmm. How can we fix that?

Well, there's a cool trick we can do with it, and many other functions in R. You can actually *assign* values to the output of functions. I know, crazy, right? If we assign a vector to the names() function like this:

```
names(tweets)  <- c("time", "seconds", "screen_name", "text")
```

it will set our column names to what we want. Neat, huh? In general, you can assign values to the results of almost any R function and it'll stick as if you were changing that thing itself. We'll see more of this soon.

The Rhythms of Twitter

This dataset spans 6/11/09 to 6/15/09 and each row in this dataset is a tweet. I'd be curious to know what tweet volume over time looked like and if there were any significant trends. Using what we learned in class, let's create a histogram of tweets using breaks=500.

[To submit]

- Make a plot of the timeseries as a line (plot() with type='l') just so we can see our data (recall you can create a timeseries by pulling out the "counts" entry of the histogram object). What do you see?
- Let's figure out if there are any cycles / seasonal trends in this data. Use the acf() function to identify cycles in the tweet frequency, just as we did with the NYPD data. Recall that acf() only looks at a small time frame so you'll want to pass it a lag.max argument that's about 200 or more. Where is it most likely that we have a cycle and how can you tell?
- OK, let's remove the cycles and analyze this data. Create an official timeseries object with frequency equal to the cycle length. Use decompose() to decompose the timeseries into its components and plot the results. What do you see in terms of an overall trend?

Let's Get Political

Overall, I don't see a lot of spikes in activity in the tweets over time. Let's breakdown our tweets around a certain topic. How about, oh, say, Iran? So how do we pull tweets out that have a certain word in them?

grep() to the rescue! If you've used the grep function on the command-line, this should look familiar. grep() takes as arguments a phrase you're searching for, a set of text to look through, and optional arguments about how to search. It will then return the row numbers of any rows that match your search. To pull out Iran tweets, we can use the code:

```
iran.tweets <- tweets[grep("iran", ignore.case=TRUE, tweets$text), ]
```

[To submit]

- Plot the time series for iran.tweets using a histogram with breaks=100. Add red vertical lines to the plot at the 3 largest peaks using abline().
- There's not a lot of seasonality in this plot, so let's go straight to analyzing the trend. Use SMA() with the default settings to smooth the signal and plot it.
- Let's build a basic event detection algorithm, but let's not use the total number of tweets, since that misses the "velocity" of the signal. Use the diff() function with a lag of 5 to look at the differences in tweet volume over time on the smoothed signal (use ?diff if you need a refresher). Create a figure with two graphs – one with the smoothed signal above and one with the diff() of the signal below it. What do you see?
- I'd like to know why all these tweets started increasing. Can we figure out what time the tweets started increasing using your results from diff, i.e. where is the biggest jump in tweets? (hint: there are lots of ways to do this, many of which require you to remove the NAs created by SMA) Pull 20 or so tweets from around that time and write down why you think they're increasing based on what people are saying.

You may see this as just a homework to get through, but what you just built is not very far off from what Twitter's trending topics uses. You can use what you did in this homework to automatically identify super important historical events (at least on Twitter). Neato!