CBE 20258 Project 3

Due by 4pm 2/26/16

In the "old days" weather forecasts were both *very* short range and notoriously unreliable. Now, of course, there are far more sophisticated weather models, vastly more data sensors available, and computers that are actually fast enough to process it all. It is still a question of just how good weather forecasts really are. An extreme example is Accuweather.com, which will give you a daily weather forecast a month and a half out – which seems likely to be rather wishful thinking. In this problem we will analyze the forecasts for the month of January and test their reliability.

So now we come to the problem:

1) **Getting the data in:** Download the excel file "januaryforecasts.xls" and get it into a format you can read in matlab. Each morning last January I downloaded the daily forecast for the month of January, and all of these have been compiled together. Thus, there are 31 "same day" forecasts, 30 "next day" forecasts, etc., down to one 30.5 day forecast. Each succeeding chunk of data has one less actual prediction: the rest of the values in the chunk are the actuals. I have also appended columns containing the historical weather and the actual weather for each day. In the first sheet everything is broken up and labeled so you can see what is going on, and in the second sheet only the 961 (e.g., 312) rows of data are included. How you get all this into matlab is up to you, just describe in the first section the method you chose and your data structure – but don't display all the data!

2) **Daily High Predictions:** Extract from your data file the absolute value of the difference between the daily high temperature prediction and the actual high temperature as a function of the number of days out. Note that the length of these vectors (number of predictions) will change with the number of days out – so be careful manipulating your indices! Do the same for the absolute value of the difference between the historical average high temperature and the actual high temperature (e.g., the "no knowledge" prediction) for the same periods. Describe in the comments for this section the data structure you are using. Plot the averages of these up as a function of the number of days out (e.g., a vector [1:31]-0.5) for up to 21 days (use the axis command to trim the plot to this length). Use a dashed line for the no-knowledge prediction and label your plot!

3) **Error Bars:** For the high temperature data in part 2, calculate the standard deviation of the averages (not the sample standard deviations!) and add them to the plot. The "errorbar" command is useful here. Don't forget that the number of predictions changes with the number of days out! In the section description, comment on whether the length of time over which the predictions appear to be useful makes sense in terms of your results from part 6 of the last project.

4) **Covariance:** The error calculation in part 3 relies on assuming the independence of the prediction error from one day to the next. We test this by focusing on the 30 "next day" predictions. From this vector, calculate the difference from the mean (e.g., the "residual") and plot it up vs. index. Does it appear random? Confirm this by

comparing the ratio of the covariance of adjacent forecast errors (e.g., shifting the column down by one, as in the last project) to the variance. Is this ratio small?

5) **Probabilities:** Here we turn the results from part 3 into probabilities. We have two possible predictions: the Accuweather.com prediction and the "know nothing" prediction of just using the historical average high. Plot up the probability that we are better off with the historical average high as a function of the number of days out (this is essentially the null hypothesis) using a semilogy scale. Draw horizontal lines on your graph at probabilities of 1%, 2.5%, 15%, and 50% (the last is just flipping a coin between the two predictions). In the descriptive part of this section, comment on the point where Accuweather.com predictions fail.

6) **Precipitation:** A marker for whether a day is nice or not (important in planning garden weddings and pool parties!) is "measureable precipitation". Using this, and matlab's "ceil" command, turn the predicted and actual precipitation into zeros and ones (note: you can choose a different threshold for measureable precipitation if you like by subtracting off your threshold before using the "ceil" command – just describe what you choose in your text!). This works nicely provided that there is less than an inch of precipitation (predicted or actual) on any day, satisfied in January: otherwise you would need to tweak the code a bit using the "min" command. By taking the absolute value of the difference of these binary vectors, you get a "1" anywhere there was an error in predicting the precipitation. Using this approach, calculate the error rate as a function of the number of days out for the predictions and plot it up. Add in errorbars for the prediction as well. The error in this case is governed by the binomial distribution: If p is the error rate and n is the number of predictions, the standard deviation in the error rate is just (p*(1-p)/n)^.5. Graphically compare this to the average fraction of the days that there was measureable precipitation (or whatever threshold you choose!) in January.