Extrapolation and Matlab Scripting

In this lab assignment, you will

1. begin using the extrapolation techniques you have learned to project data for your own region,

2. learn how to create and use matlab scripts.

A matlab script is a separate file that contains a list of matlab commands that you can run all at once. In previous labs, you wrote commands at the command prompt and execxuted them one at a time. This is called interactive programming. While it allows you to try new things out and see if they work, it can be very tedious, since you often have to type the same commands repeatedly. Writing a script is one solution to this problem. A script also allows you to save commands from one session to another so that you can repeat you work whenever you want to.

Besides allowing you to write commands, a script allows you to write comments. A comment is anything that follows a # symbol. Matlab ignores anything after this symbol until the next line is reached. Why would we want to write something Matlab will just ignore. It is good programming practice to write down what we are trying to do. Since the math can be cryptic at times, we might want to tell other people who look at our program what we are doing, or we might just want to remind ourselves what we were doing when we next look at our program.

A script is nothing other than a file with a series of commands in it. A script file has the extension "**.m**". You can give your file anyname you want to, but if it has the same name as a matlab function, then you will not be able to use the matlab function.

I will give an example of a script below, that calculates a projection for the population of Redding using a Gompertz curve.

The data is on the website. Under class 3, there will be a variety of files, one for each BEA. You should find data for your region, but for this lab, we will use the data for Redding. Download the file `ea_165.csv` to the `C:\` folder (use Explorer, not Netscape). From Matlab, click on `File --> Import File`. Select `ea_`65.csv!. Matlab should properly identify that there is 1 line of text header, and create variables for the columns and the text.

If you look at the data, you will note that the first column just has the area identifier, in this case 165 for the Redding BEA. This column is pretty much useless, so we can delete this. Also, note that the last column is the year. The last column represents year 2040. This is a projection calculated by the California Department of Finance (DOF). We will not want to use this number for making our own projections, but it might help us to compare out projections with the DOFs.

New commands that I will use in my script include:

- **subplot**
  Subplot creates a page with more than 1 graph on it

Example: **subplot(2,1,1)** creates a page with 2 rows and 1 column of graphs, i.e. one plot above the other. The last number 1, tells it that I want to work on the 1st plot for now. When I want to work on the second plot, I type **subplot(2,1,2)**

- **figure**
  Creates a new page of graphs without erasing the old page of graphs.

My script looks like this:

```
% Script to project Hispanic population in Redding.
% Input the Hispanic Data
% The data for Redding starts out in a matrix called "data".
hispanic = data(:,8); %The hispanic data is in the 4th column.
year = data(:,10); %The year is in the 10th column.
% The 2040 projection is in the 8th row.
hispanic2040 = hispanic(8); %Store the 2040 projection elsewhere.
hispanic(8)=[]; %Delete the 2040 projection from the hispanic vector.
year(8)=[];

% Plot the historical population data.
subplot(2,1,1)
plot(year,hispanic,'.')
xlabel('Year')
ylabel('Population')
title('Hispanic population of Redding')

% GOMPERTZ PROJECTION
% I will use 75000 as my ceiling
c = 75000;

% Perform input evaluation:
top = log(hispanic(3:7))-log(hispanic(2:6)); %log(P(t+1))-log(P(t))
bottom = log(hispanic(2:6))-log(hispanic(1:5)); %log(P(t))-log(P(t-1))
z = top./bottom
%Plot z value.  It should be relatively constant if the data follows a
%Gompertz trend.
subplot(2,1,2)
x = 1:5 ; %There are only 5 differences
plot(x,z,'.');
title('Growth rate of the increments')

%Perform least squares regression
y = log(log(c)-log(hispanic)); %First linearize data
OLSgomp = regres(y,year-1970); %Perform OLS regression of y on year.
% I subtracted 1970 from the year because it avoids taking the exponent of
% very large numbers.
```

```
%Calculate Gompertz coefficients
%Remember, OLSgomp.b is a 2x1 vector, with the first number being the
%intercept and the 2nd number is the slope.
b0 = exp(exp(OLSgomp.b(1)));
b1 = exp(OLSgomp.b(2));
predictyear = [1970:1:2040]; % Create a vector of years from 1970-2040 for prediction.

%Predict hispanic population
% Can do this do ways, either by
%   1. predicting yhat linear coefficients and transforming to population
%   2. predicting population from Gompertz coefficients.
%Method 1
% Predict yhat
yhat = OLSgomp.b(1) + OLSgomp.b(2)*(predictyear-1970); %y = m+b*x;
%Transform yhat to population
prediction1 = exp(log(c)-exp(yhat));
%Method 2;
prediction2 = c*b0.^(-b1.^(predictyear-1970));

%Create two plots of prediction, one that is linear, one that is in actual
%population.
figure; %Create new page for plots
subplot(2,1,1)
plot(predictyear,yhat)
hold on
plot(year,y,'.') %Plot actual data on top of prediction.
hold off
title('Linearized Gompertz projection of hispanic population')
xlabel('year')
ylabel('Transformed population')
subplot(2,1,2)
plot(predictyear,prediction2) %I could have used prediction1 as well
hold on
plot(year,hispanic,'.') %Plot actual data on top of prediction
hold off
title('Gompertz projection of hispanic population')
xlabel('year')
ylabel('Population')
```

You can save your script by clicking on `File --> Save As`. If you save it in a location other than the default, you may have to add that location to your path. For example, if you save the script to a floppy, you may have to type `addpath 'A:\'` at the command line.

When you are ready to run your script, you can click on `Debug --> Run` to run the script, or you can press `F5` or you can click on the little icon of a page

with down arrow (usually the second icon from the left in the script window).

This is it for the prepared lab this week. You can spand the rest of lab modifying this script to analyze your own data. You will need to import your own data, and change the matrix **data** to have your own data and not the Redding Data. This script uses column 4 to look at the hispanic data, but you can change this to look at the other variables in the other columns of **data**. You will also need to change your plot titles. You will also need to replace the Gompertz section with the proper commands for the other models, i.e. logistic, geometric with constraint, etc.

Something to watch out for: When the population goes up, the ceiling for the gompertz, logistic and geometric with constraint is greater than the population. Thus, we see terms in the linearization like: $y = log(log(c) - log(P))$. Since $c > P$, this term is positive, and we can take the log. When the population is declining however, then $c < P$, i.e. we have a floor constraint and not a ceiling constraint. When this happens, $log(c) - log(P)$ is negative, and the log of this term will return complex numbers. To fix this, we need to rearrange the c and P, i.e $log(log(P) - log(c))$. Keep this in mind as you use floors rather than ceilings.

How will you know whether to use a floor or a ceiling? Before you start anything, you should always look at your data to see whether the series is increasing (implies a ceiling) or decreasing (implies a floor). Also, one of the only ways we have of choosing a value for a floor or ceiling is to look at the data first and make an educated guess.