Tutorial 4: Statistics & Graphics

Getting Started

1. In this tutorial, I will lead you through a statistical analysis of a set of data. We will do some basic stats and simple plots.

2. Download Basins.mat from the website, and save it in some neutral location.

3. Start Matlab. Change directories to the one containing Basins.mat.

Mapping

4. Load Basins.mat ("load Basins") and type "whos". You should have two arrays:

   Basins: 88-by-5 array with the temp. data
   nm: 4-by-4 character array with the names of the basins

   The first column of Basins contains the years when the temperatures were taken. Columns 2:5 are the temperatures from the 4 regions: Emerald Basin, Georges Basin, Jordan Basin, and Wilkinson Basin, in the indicated years. The rows of nm contain a 4-letter name for each basin. Type "nm" to see the names.

5. Let's get some descriptive statistics for our data using mean and std (standard deviation function):

   ```
   >>mn=mean(Basins(:,2:5);
   >>sd=std(Basins(:,2:5);
   ```

   Look at mn, notice anything? Yup, mn (and sd) contain NaN's, because Basins contains nans if no temperatures were taken in a basin in a given year. We need to exclude nans from our computations. If you have the stats toolbox, you have the functions nanmean and nanstd that do exactly what we want. If you don't have the toolbox, you'll have to do it yourself:

   ```
   >>M=Basins(:,2:5);%the data
   >>nonNaNs=~isnan(M);%logic array—1 if M(j,k) is not a nan
   >>I=find(isnan(M));%location of nans
   >>M(I)=0;%replace nans with 0's
   >>mn=sum(M);%sum of columns of M
   >>n=sum(nonNaNs);%total number of good data in each column
   >>I=find(n==0):
   >>n(I)=nan;%divide by nan rather than 0
   >>mn=mn./n;%the mean.
   ```

Rather than typing all of this out, implement this as a function.

6. Assuming you've compute mn and sd, let's look at them. To plot the means as a bar plot, type

```
>>bar(mn);
```

You should see that temperature decreases to the right. This means that Emerald is warmest, followed by Georges, etc. Plot the standard deviations—do they follow the same pattern?

7. We want to know whether a change in temperature in one basin is seen in any others. We can start by plotting the time series together.

```
>>plot(Basins(:,1),Basins(:,2:5));
```

Matlab was nice enough to plot each series using a different color; however, we don't know which color goes with which series. To add a legend, we can use nm and the legend command. Type help legend to see how it works. Then, type

```
>>legend(nm, 4);
```

8. We have another problem, though. Because the means and variances are so different, it is hard to compare the series. One solution is to normalize the series so that the each have mean 0 and standard deviation 1:

```
>>BasinsN=Basins(:,2:5)-ones(88,1)*mn; %subtract means
>>BasinsN=BasinsN./(ones(88,1)*sd); %divide by sd
```

Note the use of the outer product in both lines. Clear the figure by typing "clf", and then plot the normalized series. How do they compare?

9. Comparing one series with another in this way is not quantitative. To make the analysis quantitative, we want to compute the correlation between each series and all of the others. Visually, this amounts to plotting samples of one series vs. samples of another and seeing if they fall on a line. The nice thing about Matlab, is that we can program the graphics. Type

```
>>for j=1:3;          %Loop over the first 3 series
for k=j+1:4;       %Loop over the series we haven't compared
subplot(3,3,(j-1)*3+k-1);              %create a subplot
plot(BasinsN(:,j),BasinsN(:,k),'x');   %want x's not lines
axis([-2 2 –2 2]);                     %sets axis limits
xlabel(nm(j,:));ylabel(nm(k,:));       %label the axes
end;end;
```

Do the data seem related?

10. To measure the correlations, we want to use the command corrcoef that will give the correlation coefficients (higher number means stronger correlation) and the probability that the correlations are due to chance alone (low values are better). To use corrcoef, type

```
>>[R,P]=corrcoef(BasinsN)
```

What happens? Yup, the nans are screwing us up again.

11. To deal with the nans, we will create a new array with the rows of BasinsN where all values are good. We'll start by creating a matrix with 1's where BainsN has a value and 0 where it has a nan.

```
>>notnan=~isnan(BasinsN);
```

From notnan, we want a vector with a 1 if the corresponding row of nontnan is all 1's (all good data) and a 0 if the row contains any 0's (nans). The command "all" will do this for us. Type "help all" to read about it. Notice that like most Matlab commands, all operates on columns. To get it to work on rows, we could fool around with transposes or specify the dimension we want (in this case, 2):

```
>>goodrows=all(notnan,2);
```

Finally, we want an index to only the rows where goodrows is 1. The find command will do this:

```
>>I=find(goodrows);
```

12. I is now the index we want, and we can use it to pass the good rows to corrcoef

```
>>[R,P]=corrcoef(BasinsN(I,:))
```

Notice that R is 1 along the diagonal—this means that each series is perfectly correlated with itself (this is a good thing!). Also, notice that the R is symmetric around the diagonal. This means that the correlation does not depend on the order of the data. Finally, notice that all of the correlations are high and significant (P-values <0.05) and that the correlation decreases slightly with distance.