MATLAB Visualisation

Because of its matrix orientation, MATLAB can easily perform statistical analysis on data sets - by convention stored in *column-oriented matrices* - ie each column represents a different variable and each row represents individual samples of these variables.

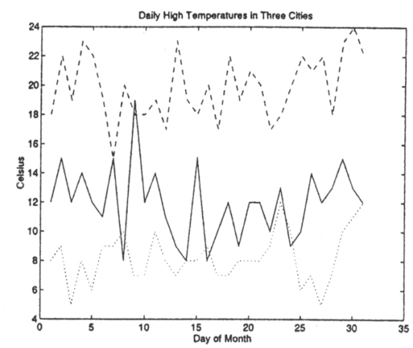
For example, if we have 3 cities and record temperature temp:

```
» temps
temps =
    12     8    18
    15     9    22
    12     5    19
    14     8    23
    12     6    22
    11     9    19
    15     9    15
     8    10    20
    19     7    18
    12     7    18
    14    10    19
    11     8    17
     9     7    23
     8     8    19
    15     8    18
     8     9    20
    10     7    17
    12     7    22
     9     8    19
    12     8    21
    12     8    20
    10     9    17
    13    12    18
     9    10    20
    10     6    22
    14     7    21
    12     5    22
    13     7    18
    15    10    23
    13    11    24
    12    12    22
```


Plotting the Data

Each row contains the high temperatures for a given day of the month and each column corresponds to different city - visualise:

```
d=1:31; % number days of the month
plot(d,temps)
xlabel('Day of Month'),ylabel('Celcius')
title('Daily High Temperatures in Three Cities')
```



The plot command above illustrates how it can be used with vector d and matrix temps to plot each column of temps against d.

MATLAB has many data analysis capabilities which we can easily demonstrate for this dataset.

Data Analysis

For example, suppose we want the mean temperature for each city:

avg_temp=mean(temps)

avg_temp =

11.9677 8.2258 19.8710

MATLAB shows mean of each column individually - take again:

avg_avg=mean(avg_temp)

avg_avg =

13.3548

Similarly, we can find maximum and minimum values in the data:

max_temp=max(temps)

max_temp =

19 12 24

MATLAB shows max or min of each column individually:

[min_temp,n]=min(temps)

min_temp =

8 5 15

n =

8 3 7

Standard Deviation and Differences

The standard deviation for each column of temps is found:

s_dev=std(temps)

s_dev =

2.5098 1.7646 2.2322

Computing the difference between daily high temperatures:

daily_change=diff(temps)

daily_change =

3	1	4
–3	–4	–3
2	3	4
–2	–2	–1
–1	3	–3
4	0	–4
–7	1	5
11	–3	–2
–7	0	0
2	3	1
–3	–2	–2
–2	–1	6
–1	1	–4
7	0	–1
–7	1	2
2	–2	–3
2	0	5
–3	1	–3
3	0	2
0	0	–1
–2	1	–3
3	3	1
–4	–2	2
1	–4	2
4	1	–1
–2	–2	1
1	2	–4
2	3	5
–2	1	1
–1	1	–2

Data Analysis Summary

Data analysis in MATLAB is performed on column-oriented matrices where different variables are stored in individual columns and each row represents a different observation of each variable.

MATLAB statistical functions include:

Statistical Function	Description
corrcoef(x)	Correlation coefficients.
cov(x)	Covariance matrix.
cumprod(x)	Cumulative product of columns.
cumsum(x)	Cumulative sum of columns.
diff(x)	Compute differences between elements.
hist(x)	Histogram or bar chart.
mean(x)	Mean or average value of columns.
median(x)	Median value of columns.
prod(x)	Product of elements in columns.
rand(x)	Uniformly distributed random numbers.
randn(x)	Normally distributed random numbers.
sort(x)	Sort columns in ascending order.
std(x)	Standard deviation of columns.
sum(x)	Sum of elements in each column.

Next we continue with just a few examples to extend visualisation.

Linestyles, Markers and Colours

It is possible to specify colour, linestyle, and markers, such as plus signs or circles, with:

```
plot(x,y,'color_style_marker')
```

color_style_marker is a 1-, 2-, or 3-character string (delineated by single quotation marks) constructed from a colour, a linestyle, and a marker type:

Color strings are ‘c’, ‘m’, ‘y’, ‘r’, ‘g’, ‘b’, ‘w’, and ‘k’ - these corresp to cyan, magenta,yellow, red, green, blue, white, and black.

Linestyle strings are ‘-’ for solid, ‘- -’ for dashed, ‘:’ for dotted, ‘-.’ for dash-dot, and ‘none’ for no line.

The most common marker types include ‘+’, ‘o’, ‘*’, and ‘x’.

For example, the statement:

```
plot(x,y,'y:+')
```

plots yellow dotted line and places plus sign markers at data points

If you specify a marker type but not a linestyle, MATLAB draws only the marker.

Adding Plots to an Existing Graph

The plot function automatically opens a new figure window if there are no figure windows already on the screen -if figure window exists, plot uses that window by default.

To open a new figure window and make it the current figure, type:

```
figure
```

The hold command allows you to add plots to an existing graph - when you type:

```
hold on
```

MATLAB does not remove the existing graph; it adds the new data to the current graph, rescaling if necessary.

For example, these statements first create a contour plot of some peaks, then superimpose a pseudocolor plot of the same function:

```
[x,y,z] = peaks;
contour(x,y,z,20,'k')
hold on
pcolor(x,y,z)
shading interp
```

The subplot function allows you to display multiple plots in the same window or print them on the same piece of paper, type:

```
subplot(m,n,p)
```


Controlling Axes

The axis function has a number of options for customizing scaling, orientation, and aspect ratio of plots.

Ordinarily, MATLAB finds the maxima and minima of the data and chooses an appropriate plot box and axes labeling - axis function overrides the default by setting custom axis limits:

```
axis([xmin xmax ymin ymax])
```

axis also accepts number of keywords for axes control -for example:

```
axis square
```

makes the entire x-axes and y-axes the same length and

```
axis equal
```

makes the individual tick mark increments on the x- and y-axes the same length - plot(exp(i*t)) followed by either axis square or axis equal turns an oval into a proper circle.

```
axis auto
```

returns the axis scaling to its default, automatic mode.

```
axis on
```

turns on axis labeling and tick marks.

```
axis off
```

turns off axis labeling and tick marks.