MATLAB BASICS

TABLE OF CONTENTS

Page:

List of Figures

List of Tables

List of m.files

Introduction

In this handout, I've compiled a listing of the most common Matlab programming methods that I used in writing the m.files to accompany Larsen & Marx (2006). As examples of these programming methods, I've included Matlab programs to solve the probability and statistics problems from the MA high school MCAS exams.

Differences among Matlab versions

I've been using Matlab since 1992, years before Windows. I can remember when the first version of Matlab from Windows came out in 1994. I can recall how much slower the Windows version was. Matlab 3.5 was a speed demon and the neatest looking graphics on an all-black DOS screen.

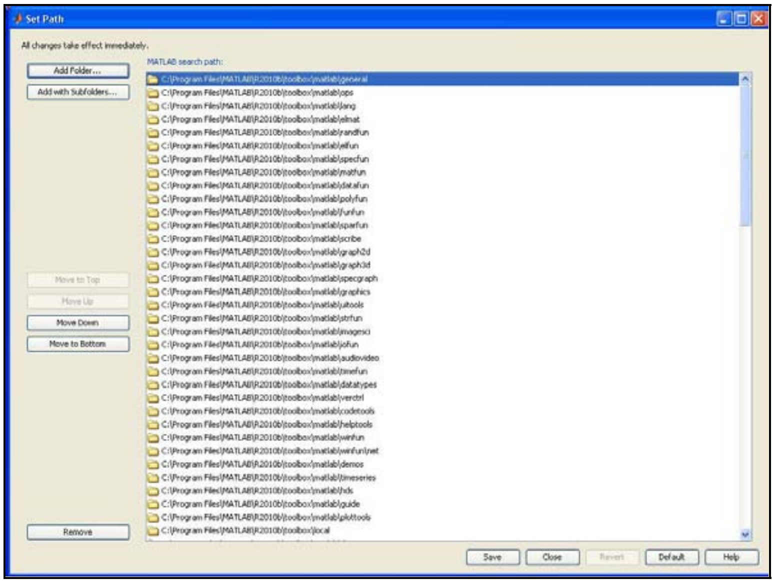
I use the professional version of Matlab (R2010b) with the optimization, symbolic math and statistics toolboxes. All of these are available in the $100 student version. This course will make extensive use of both the statistics and symbolic math toolboxes. Prior to 2004, the student version of Matlab lacked the statistics toolbox. Mathworks charged an extra $50 for this student version. I wrote almost all of my m.files so that it wouldn't require the statistics toolbox. I wrote a large number of statistics programs and others I used from the Stixbox and staxbox toolboxes (links on the syllabus). I also used the dataviz and glmlab toolboxes which implemented statistical graphics routines and performed generalized linear modeling.

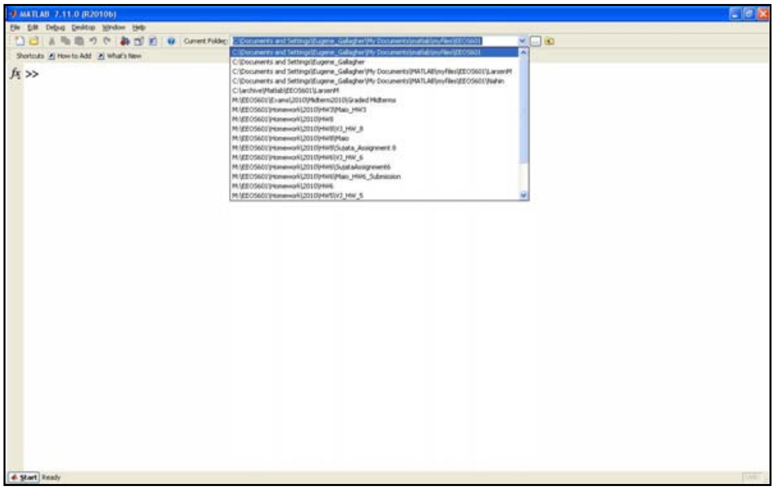
Loading Matlab

I'd recommend installing Matlab in the default directory. On my computer, the installation created a Matlab folder under 'My documents.' I created a separate folder called 'myfiles' as a subfolder of Matlab. Within 'myfiles' I created an EEOS601 folder. Within the EEOS601 folder, I have a set of folders which include LarsenMarx and Nahin. I put all of my m.files specific to Larsen & Marx (2006) in the LarsenMarx folder.

Matlab Path

When you type ABCD at the >prompt in the Matlab command screen, Matlab will go through a series of checks in sequence: 1) Is ABCD the name of a variable in memory? If not, 2) Is ABCD the name of a program ABCD.m on the current director, and finally 3) Is ABCD.m found on any of the sequential list of directories on the Matlab path. So, if you want Matlab to run your programs, the program has to be either on the current directory, or the program has to be on the Matlab path. By clicking on file \set path you can add your EEOS601 directory (see figure at right) and the appropriate subdirectories. It is vitally important that you move the names of your folders to the bottom of the file queue. You could bring Matlab to a halt if you name your programs with names like sum, mean, median and your m.files are invoked rather than those on the Matlab path.



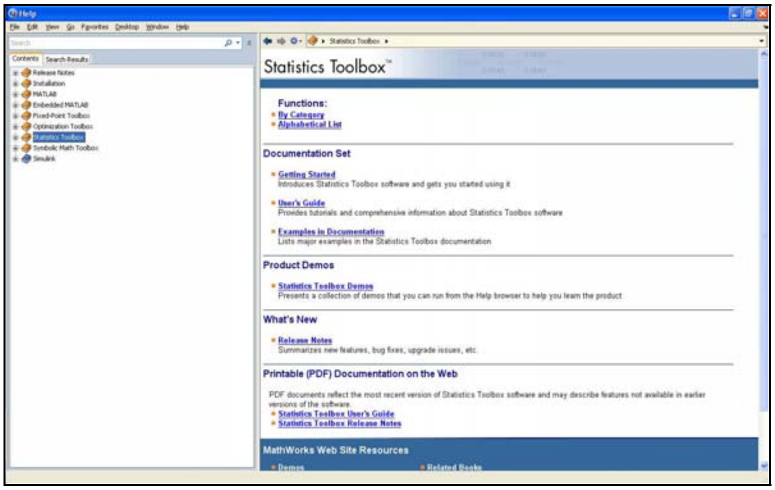


You can change Matlab directories by clicking on the \/ **or ...** next to the file window on the command window (see at right) and change to another directory. If you remember how to type the directory you can type > cd EEOS601 to move from the myfiles to the EEOS601 directory. Matlab will search through your path statement to find a file called startup.m. You can create this as a script file in the Matlab editor. I have a one-line startup.m file which tells Matlab to switch to my EEOS601 directory as the home directory:

```
cd('C:\Documents and Settings\Eugene_Gallagher\My Documents\matlab\myfiles\EEOS601')
```

Help files

I am continually typing > help function to find out how to call both Matlab's functions and my own. Like learning a foreign language, the help files are like Matlab's dictionary. For more documentation or if you can't remember the function name, you can open a help window in Matlab. Click on help product help and statistics toolbox to bring up the user's manual for the statistics toolbox (see right).



Other useful commands are who and whos that lists all of the variables in memory and > which function that reveals the directory in which the location of the file is located.

Matlab Syntax

Script m.files

Almost all of my Matlab programs for this course are written as script m.files. In a way, this is just like cutting and pasting the commands one after another at the prompt in the command window. For my research, I invariably transfer all of my script m.files to function m.files. Function m.files are compiled into fast machine code, so they will run MUCH faster than script m.files. Variable names remain internal to the function m.files, so I can call 20 different functions each calling a different data matrix DATA. When these results are output to the main program, I assign these DATA matrices to different names.

A third type of Matlab file is a mat file, which saves data in a compressed binary format. For massive datasets, the saving in space can be massive.

Entering data

Data Assignment Statements

The equal sign = in Matlab does not have its standard algebraic meaning. As in most programming languages, like Fortran, the equal sign is a data assignment command. It means take the expression on the right side of the = and assign it to the memory location associated with the variable name on the left side of the = sign.

So, 2+2=4 and 4=2+2 are meaningless statements in Matlab, but N=2+2 is a meaningful statement. N(1)=2+2 assigns the number 4 to the 1st element of the N matrix. N(K+1)=2+2 checks the value of K and assigns the number 4 to the K+1 memory location of matrix N. If K is 1000, then K(1001) is assigned the value 4.

data can also be stored as an array. All matlab variables are cointained within square brackets and the semicolon is equivalent to a hard return. So, the following one line program could be saved as an m.file called mydata.m
DATA=[1 3;4 2]
When mydata is typed on the screen then the statements in the program are processed and the variable DATA is in memory. It will remain in memory until the session ends, or the statements clear DATA, clear variables, or clear DATA are entered

The variable DATA could also be stored on the default directory using the following **save** command:
save mydata DATA
This command will create a binary file with a **.mat** extension called **mydata.mat**. It can be brought back into the programs memory using the load statement
load mydata
DATA can be transferred to spreadsheets — Excel, Quattro Pro, or 123 — using tab delimited ascii files using the following syntax
save mydata.txt DATA -ascii -tabs
This data file can be loaded into the spreadsheet program after clicking on the gui in the spreadsheet program indicating that it is a tab-delimited file. Similarly, data from spreadsheets can be loaded into Matlab by either a cut-and-paste into the Matlab editor and then adding square brackets at the beginning and end
DATA = ["paste the data from a clipboard copy command here"
]
Or, the data can be saved in the spreadsheet as a tab or comma-delimited file. Using the Matlab load statement, Matlab will load the data into memory assigning the variable name mydata to the matrix.
load mydata.prn

Displaying data

To prevent the result from being echoed on the screen, add a semicolon. Multiple lines can be placed side by side if they are separated by semicolons.

```
>> M=2+2;N=3+3;Q=M+N
Q =
    10
```

You can get rid of the equal sign by using the display command

```
>> disp(Q)
    10
```

I prefer to use the C language print statement **fprintf** because it provides more control of the output:

>> M=2+2;N=3+3;Q=M+N;**fprintf**('The sum of %1.0f and %1.0f is %2.0f.\n',M,N,Q)
The sum of 4 and 6 is 10.
%2.0f specifies a 2-digit floating point number with 0 digits to the right of the decimal point The \n tells Matlab to enter a hard line return

sprintf can be used to create a character string. This is particularly useful in creating strings for labeling the axes of graphs.

s=sprintf('The sum of %1.0f and %1.0f is %2.0f.\n',M,N,Q)

Algebraic operators

+, -, * have their conventional meanings of addition, subtraction and multiplication if one or both of the variables is a scalar (a single number). For example, magic(3) is a matrix in which all row, column and diagonal sums are identical. This 3x3 matrix can be added from and subtracted from 1:

```
>> magic(3)
ans =
    8    1    6
    3    5    7
    4    9    2
>> 1-magic(3)
ans =
   -7    0   -5
   -2   -4   -6
   -3   -8   -1

>> 1+magic(3)
ans =
    9    2    7
    4    6    8
    5   10    3
```

Now, Matlab considers all variables to be matrices, so A*B is matrix mulitiplication. To multiply each element of a matrix times another, one must use '.*' instead of '*'

```
>> magic(3)*magic(3)
```

```
ans =
   91    67    67
   67    91    67
   67    67    91
>> magic(3).*magic(3)
ans =
   64     1    36
    9    25    49
   16    81     4
```

Similarly exponentiation can be done on the full matrix '^' or on individual elements '.^'

```
>> magic(3)^2
ans =
   91    67    67
   67    91    67
   67    67    91
>> magic(3).^2
ans =
   64     1    36
    9    25    49
   16    81     4
```

Other Algebraic Operators

abs(N)	Absolute value
exp(N)	e^N
log(N)	ln(N)
log10(N)	$\log_{10}$(N)
sqrt(N)	√N
factorial(N)	N! Factorial. I almost never use the factorial in programming. In probability and combinatorics, it is usually the ratio and product of factorials that is of interest. It is more computationally appropriate to use the gamma distribution and the fact that N! = Γ(N+1)=gamma(N+1). Matlab uses the gammaln function for the natural log of the gamma distribution. So,

```
>> factorial(37) /(factorial(10)* factorial(27))

ans =
  348330136
>> exp(gammaln(38)-(gammaln(11)+gammaln(28)))
ans =
   3.483301359999959e+008
```

Matrices and vectors

Matlab stores all data as matrices. These include numeric matrices, character matrices, and symbolic matrices. All character strings are enclosed in single quotations. Matrices are contained within square brackets [ ]. Sometimes in working with character strings, it is preferable to store the strings using the curly bracket notation. This creates a cell array. Species = {'Capitella', 'Polydora', Streblospio'}.

Matrix manipulations

N'	transpose of N: rows become columns
fliplr	flip a matrix left-right
flipud	flip a matrix up-down
N=N(:)	convert a matrix into a single column vector
N=magic(4);N=reshape(N,2,8)	reshape a matrix
repmat(N,4,1)	Reproduce matrix N 4 times down
repmat(N,1,4)	Reproduce matrix N 4 times across
N=1:10	Create a row vector with the numbers 1 to 10
N=2:2:100	Creates all of the even numbers from 2 to 100, the middle number is the increment
N=logspace(0,3,7)	Creates row vector of 7 $\log_{10}$ spaced numbers from 10^0 to 10^3
N=[logspace(0,3,7)]'	Creates a column vector of 7 $\log_{10}$ spaced numbers from 10^0 to 10^3
ceil(N)	Round N up to the nearest integer
floor(N)	Round N down to the nearest integer
round(N)	Round N to the nearest integer, up if equidistant
M = mod(X,Y)	returns the remainder X - Y.*floor(X./Y) for nonzero Y, and returns X otherwise. mod(X,Y) always differs from X by a multiple of Y.

Column-wise operations

min, max, mean, median   Calculate the min, max, mean and median of each column

```
>> median(magic(3))
ans =
    4     5     6
>> mean(magic(3))
ans =
    5     5     5
```

Logical Operators

All logical operators evaluate a statement or pairs of statements and return a 0 for false and 1 for true.

<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
~	Not
~=	Not equal to

The logical operators can operate elementwise on entire matrices. these commands find all of the elements in a matrix greater than or equal to 7

```
>> N=magic(3)
N =
     8     1     6
     3     5     7
     4     9     2
>> M = N>= 7
M =
     1     0     0
     0     0     1
     0     1     0
```

The not operator ~ will change all non-zero elements to 0 and all zero elements to 1. So:

```
>> ~(magic(3)>=7)

ans =
     0     1     1
     1     1     0
     1     0     1
```

Indices

All variables in Matlab are treated as matrices. The variable N created by N=2 is regarded as a 1,1 matrix. Matlab counts matrices starting from 1 (C and C++ begin their indices with 0). N(10,2)=ones(10,1) creates a matrix of all ones that has 10 rows and 2 columns. The indices are numbered by column, so the 1st element of the 2nd column of the 10 x 2 matrix would be 11.

Logical statements

any	true if any element of vector is nonzero
all	true if all elements of a matrix are nonzero
isempty	true for empty matrices
isnan	true for a NaN

Special characters & matrices

ones(2,3)	Creates a 2 x 3 matrix of all ones
zeros(3,2)	Creates a 3x2 matrix of all zeros
NaN	stands for not a number
[]	An empty matrix
eye(3)	creates a 3x3 identity matrix (1's on main diagonal, zeros elsewhere)

Search operators

find finds the indices of all of the non-zero elements in a matrix. So,

```
>> N=magic(3);M=N>=7; i=find(M)
i =
     1
     6
     8
>> N(i)
ans =
     8
     9
     7
```

Deleting elements from matrices

```
A=1:10;A(7)=[]; A
A =
     1     2     3     4     5     6     8     9    10
```

Building matrices (Concantenation)

```
A=[A(1:6) 77 A(7:9)]
A =
     1     2     3     4     5     6    77     8     9    10
```

Deleting elements through indexing

```
>> i=find(A<77);A=A(i)
A =
     1     2     3     4     5     6     8     9    10
```

Elementwise operators

abs, sqrt,

Column-wise operators

sum
mean
median
std
var

[K,i]=sort(N)   Sorts the rows of matrix N and returns the sort indices in I

```
N=magic(3);[K,i]=sort(N)
K =
    3    1    2
    4    5    6
    8    9    7
i =
    2    1    3
    3    2    1
    1    3    2
```

Sets

intersect   INTERSECT(A,B) for vectors A and B, returns the values common to the two vectors. MATLAB sorts the results. A and B can be cell arrays of strings.

setdiff   SETDIFF(A,B) when A and B are vectors returns the values in A that are not in B. The result will be sorted. A and B can be cell arrays of strings.

union   UNION(A,B) for vectors A and B, returns the combined values of the two vectors with no repetitions. MATLAB sorts the results. A and B can be cell arrays of strings.

unique   B = UNIQUE(A) for the array A returns the same values as in A but with no repetitions. B will also be sorted. A can be a cell array of strings.

Control flow

for and **while** statements are used to perform subsections of the program while the statement

for i=1:20   perform the statements after the for statement and before the end statement 20 times, to produce 20 factorial. Other examples might be for i=1:2:1000 or for i=logspace(0,1000,10)

```
tally=1
for i=1:20
        tally=i*tally;
end
```

while   peform the statements after the **while** line as long as the expression is true.

```
i=1;tally=1;
while i <=20
        tally=i*tally;
        i=i+1;
end
```

Graphics

The basic matlab statements are

plot(X,Y) plots variables X and Y with a line plot. The line styles can include the basic colors, -b, -r, -g, -y, -w, -k, -m, for blue, red, green, yellow, white, blcak and magenta. Any combination of red, green and blue can be specified with optional commands. Other linestyles include '- -' or '..

plot(X,Y,'.g') This will plot a green period symbol at each X and Y point. Other symbols include o, or s for a square. There are a dozen other symbols.

The full array of plotting symbols and linestyles can be obtained using help plot:

```
b     blue          .     point              -     solid
g     green         o     circle             :     dotted
r     red           x     x-mark             -.    dashdot
c     cyan          +     plus               --    dashed
m     magenta       *     star             (none)  no line
y     yellow        s     square
k     black         d     diamond
w     white         v     triangle (down)
                    ^     triangle (up)
                    <     triangle (left)
                    >     triangle (right)
                    p     pentagram
                    h     hexagram
```

plot(X,Y,'LineWidth',2,'Color',[.6 0 0]) This changes the linewidth to double thickness and specifies the color using the red, green blue triple. This would plot a medium thickness maroon line.

```
plot(x,y,'--rs','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','g',...
            'MarkerSize',10)
```

Multiple lines can be plotted with the same plot command. If no colors are specified Matlab will cycle through colors specified by the axes colororder command.

axis The minimum and maximum dimensions for X, Y (and Z) are specified by the axis command. These will be set automatically, but often a more esthetically pleasing graph can be obtained by setting these axis limits. The command has the following form axis([xmin xmax ymin ymax])

xlabel('text') Adds a label to the x axis

ylabel('text') Adds a label to the y axis

title('text') Adds a label to the title

xlabel('text','FontSize', 16) Matlab labels are unreadable when transferred into a presentation slideshow unless the FontSize is increased. I almost always increase the fontsize to 16 point for labels and 20 points for the title of the graph.

figure(gcf) I usually work with only the command screen open. I'll put in the figure (gcf) command in my programs which will bring the figure window with the **g**et **c**urrent **f**igure or gcf option to bring the most recently plotted figure to the fore. I usually follow this with a **pause** so that I can examine the graph before the program proceeds by the viewer hitting any key.

Diagnostic graphics

boxplot Used throughout the course, see LMex070401_4th.m

normplot plot to assess normality of a distribution, used in LMex070401_4th.m

Pretty 3-d graphics

hist2d.m plots 3-d histograms

mesheq.m 3d solution to a binary logistic regression of California earthquakes.

Random numbers and combinatorics

allperms.m All permutations

rand(7) generates 7 uniformly distributed random numbers on the interval 0,1

randi(7) Uniformly distributed random integers on the interval 1:7

randperm(7) randomly permutes the integers 1:7

randn(17,3) generates a 17 x 3 matrix of standard normal deviates (mean 0 and variance=1)

factorial(7) 7!

exp(gammaln(8)) 7! Note that Γ(N+1) =N! And gammaln(N) is ln(gamma(N))

nchoosek(n,r) $\binom{n}{r} = \frac{n!}{(n-r)!\,r!}$ NCHOOSEK Binomial coefficient or all combinations.

NCHOOSEK(N,K) where N and K are non-negative integers returns N!/K!(N-K)!. This is the number of combinations of N things taken K at a time. When a coefficient is large, a warning will be produced indicating possible inexact results. In such cases, the result is only accurate to 15 digits for double-precision inputs, or 8 digits for single-precision inputs.

NCHOOSEK(V,K) where V is a vector of length N, produces a matrix with N!/K!(N-K)! rows and K columns. Each row of the result has K of the elements in the vector V. This syntax is only practical for situations where N is less than about 15.

perms(n,r) PERMS All possible permutations.
PERMS(1:N), or PERMS(V) where V is a vector of length N, creates a matrix with N! rows and N columns containing all possible permutations of the N elements.

Matlab pdfs, cdfs & inverse functions

pdf	cdf	inv	Probability distribution
binocdf	binocdf	binoinv	Binomial
chi2pdf	chi2cdf	chi2inv	Chi Square
exppdf	expcdf	expinv	Exponential
fpdf	fcd	finv	F distribution
geopdf	geocdf	geoinv	Geometric
hygepdf	hygecdf	hygeinv	Hypergeometric
normpdf	normcdf	norminv	Normal
poisspdf	poisscdf	poissinv	Poisson
tpdf	tcdf	tinv	Student's t

Probability functions

bayestheorem.m Simple 1-line implementation of Bayes theorem
hypergeop.m pdf for the hypergeometric
multinomial.m User contributed m.file

Statistical functions

Categorical data

chisquarecont User-contributed m.file for chi-square analysis of 2x2 tables.
contincy.m Chi-square analysis of rxc contingency tables.
fishertest.m User contributed m.file for Fisher's exact hypergeometric test.(c) Jos van der Geest
hetchi.m Heterogeity chi-square statistic

One-sample tests

binom1sample.m	
demoivre.m	Evaluates the DeMoivre-Laplace rule for the adequacy of the Normal approximation to the binomial distribution
onesamplebinom.m	
signtest.m	Matlab's sign test
stud1sample	
student1group	
ttest.m	Matlab's 1-sample t test

Two-sample tests

binom2sample	Two-sample binomial test, covered in Larsen & Marx Section 9.4 (Case Study 9.4.1 and 9.4.2)
exactptratio	Exact probability test that two means are different, used in Advanced Problem 2 in Week 8
randp2sample	Test for difference between 2 means using random permutations
ranksum	Matlab's Wilcoxon rank sum test
signrank	Matlab's Wilcoxon's signed rank test.
stud2sample	Gallagher's Student's 2-sample t test
student2group	This program will solve the independent samples t test using aggregated data (means, sd's, and n's). It is used in LMcs090202_4th.m
ttest.m	Matlab's 1-sample t test and paired t test.
ttest2.m	Matlab's independent samples and Welch's Student's t test
unequalvariances.m	Gallagher's unequalvariances.m
welch2sample	Gallagher's Welch's t test
wilcoxranksum	Gallagher's Wilcoxon rank sum test
wilcoxrsexact	Gallagher's Wilcoxon signed rank test

Correlation

Kendall
Spearman

Regression

boxcoxlm	Performs box-cox transformation analysis
glm.m	Gallagher's least squares with Draper & Smith equations
loess.m	Lowess (c) 1998 by Datatool
leastsqu.m	Gallagher's OLS regression program with estimate of lack of fit MS, F & P
ls.m	Least squares regression: B=X\Y; nYest=X*B
polyfit, polyconf	Matlab's polyfit will fit a polynomial regression including a 1-explanatory variable OLS regression. Why would you want to do that? Because, polyonf.will generate prediction and confidence regions, including the

	simultaneous Hotelling-Working or Scheffé prediction and confidence limits for supplemental cases. Used in LMex110303_4th.m
regress.m	Matlab's statistics toolbox regression program

ANOVA

adtukeyaov2.m	An m.file that performs Tukey's additivity test for unreplicated blocked and factorial ANOVA's
anova2	Matlab's 2-way ANOVA, used in LMcs130201_4th.m
anovalc	Gallagher's linear contrasts for ANOVA
anovan	
multcompare	Performs multiple comparison procedures (used throughout chapter 13, especially LMcs130201_4th.m

Multivariate

factor.m
Leggallfigs.m
Pca.m
Pcah.m
Polypcah.m

Ecological statistics

Diversity indices

brillouin.m	Brillouin diversity
gini.m	Gini-Simpson diversity
logseries.m	Fisher's logseries alpha
Shannon.m	Shannon-Wiener diversity

Minimization & optimization

fminbnd As used in LMex030203_4th.m and LMcs010201_4th.m, **fminbnd** is a bounded minimization routine. In the following program, pwin is the probability that the better team will win the Stanley Cup. It is bounded to find the probability between 0.49 and 1.0, because the better team can't have a probability of winning less than 0.5

```
f=@(pwin,ObservedP) ...
    sum(  ( (binopdf(3,3:6,pwin)*   pwin   + ...
    binopdf(3,3:6,1-pwin)*(1-pwin))-ObservedP).^2);
    % Call the function f to find pwin that minimizes the function
    % with bounds 0.49 <= p <= 1
    pwino = fminbnd(@(pwin) f(pwin,ObservedP),0.49,1);
```

fsolve FSOLVE solves systems of nonlinear equations of several variables. FSOLVE attempts to solve equations of the form: F(X) = 0 where F and X may be vectors or matrices. Used in LMex070501_4th.m among others.

fzero fzero finds the value of parameter for which an equation is zero. The following statement is found in LMex030204_4th.m. and it finds that p is 1/3.

```
Penginefails = fzero(@(p) (3*p-1)*(p-1), 0.001, 0.999);
```

solve Solve, part of the symbolic math toolbox will find the solution to equations. The following use of solve, from LMex020205_4th.m, finds the roots of a quadratic equation and then finds the values of a, b, and c that produce equal roots.

```
A = solve(a*x^2 + b*x + c);
fprintf('Membership is A is contingent upon a, b and c satisfying:\n')
a=solve(A(1)-A(2),a)
b=solve(A(1)-A(2),b)
c=solve(A(1)-A(2),c)
```

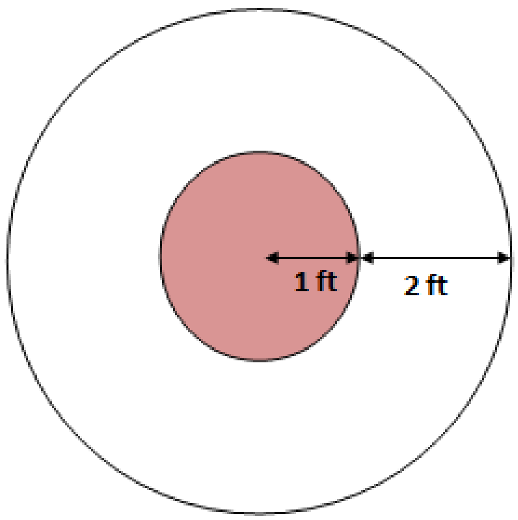
solve The following statements, from LMex020207_4th.m, solves the following problem: *Let A bet the set of x's for which* $x^2 + 2x=8$*; let B be the set for which* $x^2 + x = 6$*, Find* $A \cap B$ *and* $A \cup B$*.*

```
syms A B x
A=solve(x^2+2*x-8)
B=solve(x^2+x-6)
AintB=solve(x^2+2*x-8,x^2+x-6)
AunionB=unique([A B])
```

MCAS Questions

There is a toolkit of geometric equations on page **29**.

9 Julie designed a target computer game. On her computer screen, the circular targets look like the circular areas shown below.



If the computer randomly generates a dot that lands within the circular areas. What is the approximate probability that the dot will land in the shaded area?

A. $\frac{1}{9}$

B. $\frac{2}{9}$

C. $\frac{1}{3}$

D. $\frac{2}{3}$

Estimating π using a random number generator

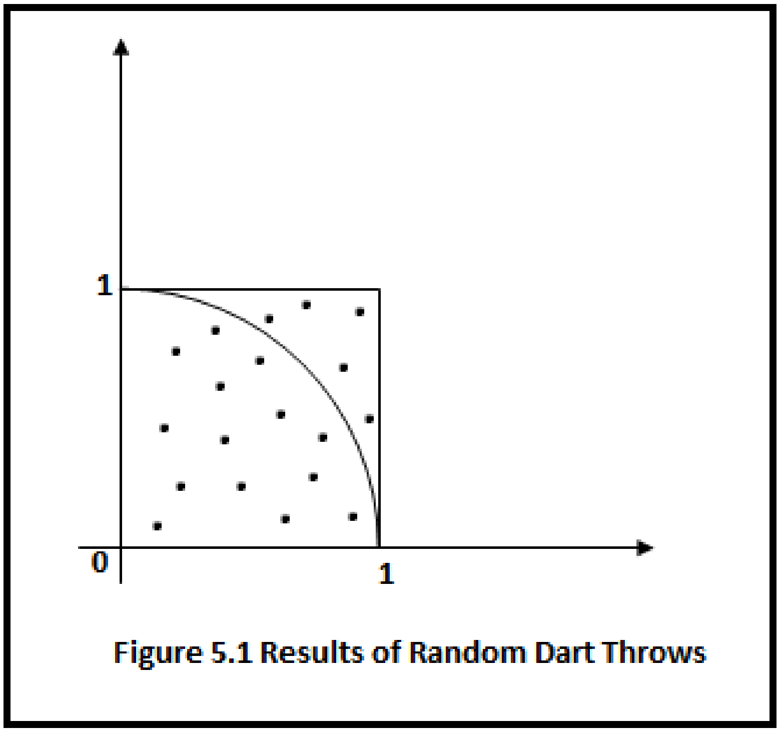


Figure 1. Figure 5.1

```
%pisim.m/created by PJNahin for "Duelling Idiots"(10/22/98)
%This m-file estimates pi by randomly tossing darts at the
%unit square and counting how many land inside a quarter-
%circle with unit radius contained in the square.
%
%
rand('state',100*sum(clock))  %set new seed for generator;
darts=0;                        %initialize number of darts
                             %inside quarter-circle region;
for i=1:10000;
  x=rand;              %toss a
  y=rand;              %dart;
  r=x*x+y*y;           %compute distance squared from origin to dart;
  if r<1             %is dart inside quarter-circle region?
    darts=darts+1;   %yes
  end
end
pi_estimate=4*darts/10000
```

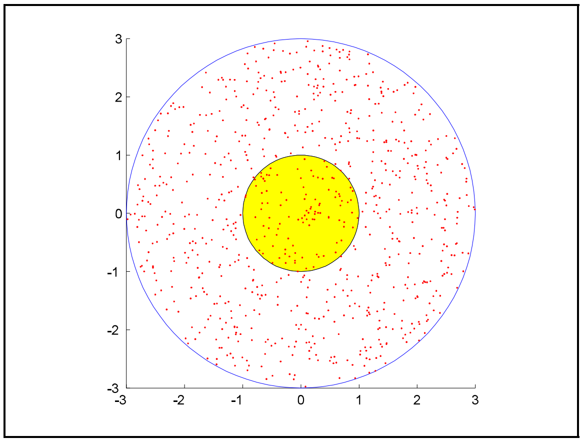


Figure 1. MCAS Question 9 simulation. If the radius of the inner circle is 1 and the outer circle 3, what is the probability that a dot placed uniformly within the outer circle will land within the inner circle.

% MCAS Question 9

```
% Solves question 9, 10th grade math, 2001
MCAS
% Written by E Gallagher 9/5/01
% Eugene.Gallagher@umb.edu
% Last revised 9/6/01
clf % clears the previous graph
hold on; % no graphs will be erased;
theta = 0:pi/50:2*pi;
% first draw a circle of radius 3, centered at 0
r=3;
x = r * cos(theta) + 0;
y = r * sin(theta) + 0;
plot(x, y);axis('square');
% Now draw a circle of radius 1, centered at 0
r=1;
x = r * cos(theta) + 0;
y = r * sin(theta) + 0;
plot(x, y);
% Now fill this circle with yellow;
h =fill(x,y,'y');
hold on
figure(gcf)
% Modify dart throwing code from Nahin's pisim.m
rand('state',100*sum(clock))  %set new seed for generator;
trials=0;            % initialize trials
success=0;           % initialize successes
tic
h = waitbar(0,'Please wait...');
for i=1:1000;
   x=rand*6-3;       % generate a uniform random number on the interval -3 3
   y=rand*6-3;       % generate a uniform random number on the interval -3 3
   r=sqrt(x^2+y^2);   % compute Euclidean distance from origin to dart;
   if r<3
      trials=trials+1;     % don't count darts outside the outer circle
      plot(x,y,'.r','MarkerSize',4);
      if r<1            % is dart inside the inner circle?
        success=success+1;  % yes
      end              %yes
   end
   waitbar(i/1000,h)
end
clf(h)5
figure(gcf)
pause
```

```
p_est=success/trials
1/9-p_est
toc
```

Now a fast version of Question 9

```
% MCAS10gm01Q9b.m, a vectorized versioin of MCAS10gm10Q9.m
% MCAS Question 9
% Solves question 9, 10th grade math, 2001 MCAS
% Written by E Gallagher 9/5/01
% Eugene.Gallagher@umb.edu
% Last revised 1/18/11
clf % clears the previous graph
hold on; % no graphs will be erased;
theta = 0:pi/50:2*pi;
% first draw a circle of radius 3, centered at 0
r=3;
x = r * cos(theta) + 0;
y = r * sin(theta) + 0;
plot(x, y);axis('square');
% Now draw a circle of radius 1, centered at 0
r=1;
x = r * cos(theta) + 0;
y = r * sin(theta) + 0;
plot(x, y);
% Now fill this circle with yellow;
h =fill(x,y,'y');
hold on
figure(gcf)
% Modify dart throwing code from Nahin's pisim.m
% Vectorized
rand('state',100*sum(clock))  %set new seed for generator;
tic
darts=1000;
X=rand(darts,2)*6-3;
r=sqrt(sum((X.^2)'));
i=find(r<3);
trials=length(i);
success=length(r(i)<1);
plot(X(i,1),X(i,2),'.r','MarkerSize',4);figure(gcf)
p_est=success/trials
1/9-p_est
toc
```

What if you forgot $A = \pi r^2$?

Figure removed due to copyright

Figure 2. Figure 5.1 from Bevington &Taylor (1992).
$A_c = A_s N_c / N_s$

26 A set of 36 cards is numbered with the positive integers from 1 to 36. If the cards are shuffled and one is chosen at random, what is the probability that the number on the card is a multiple of **both 4 and 6**?

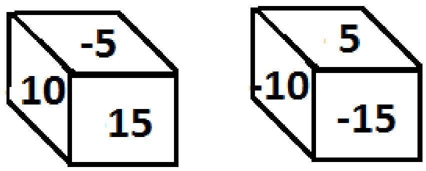
A. $\frac{1}{12}$

B. $\frac{1}{6}$

C. $\frac{5}{12}$

D. $\frac{2}{3}$

37 Joseph has two number cubes, each with faces labeled by the numbers -15, -10, -5, 5, 10 and 15.

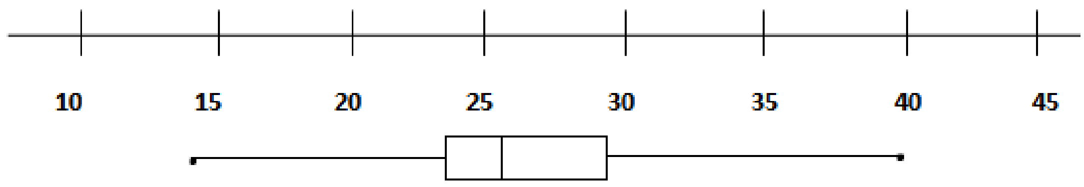


If Joseph rolls the two cubes and adds the resulting numbers, what is the probability that the sum will be 0?

A. $\frac{1}{36}$
B. $\frac{1}{12}$
C. $\frac{1}{4}$
D. $\frac{1}{6}$

Mathematics, Grade 10

39 The box and whisker graph show below represents the results of a survey of the estimated gas mileage of 100 car models.



Which statistics – mean, median, mode, range – can be determined from this graph?

A. Mean only
B. Median only
C. Range and Mean
D. Range and Median

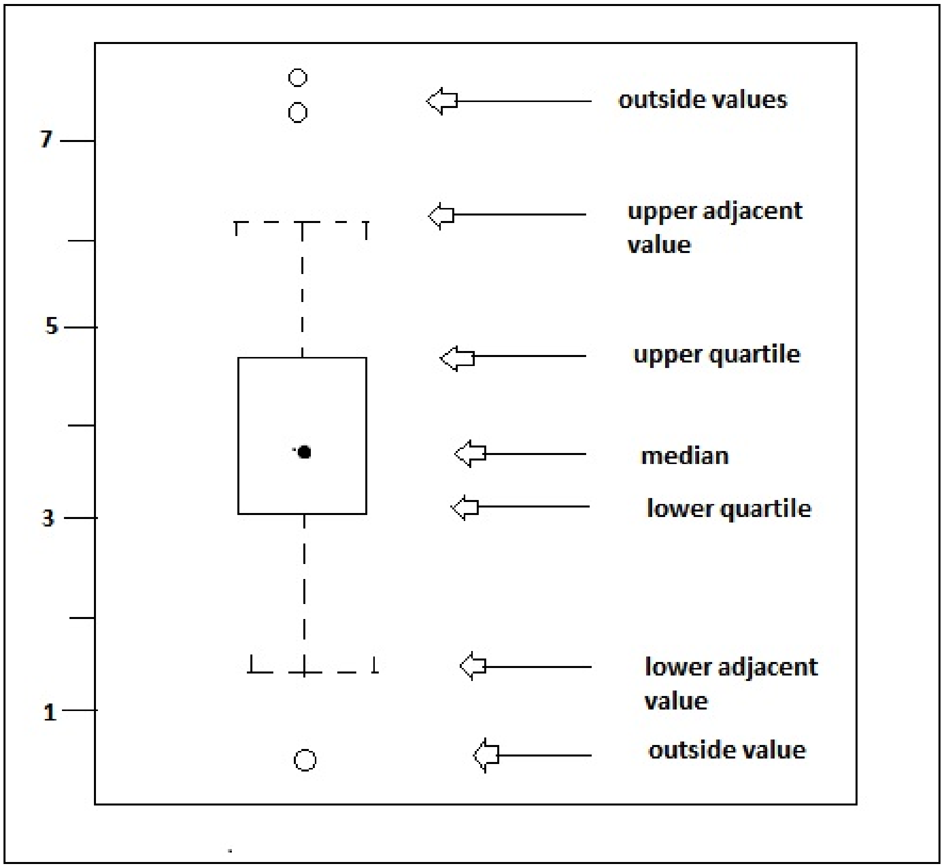

7
5
3
1
outside values
upper adjacent value
upper quartile
median
lower quartile
lower adjacent value
outside value

Figure 2.

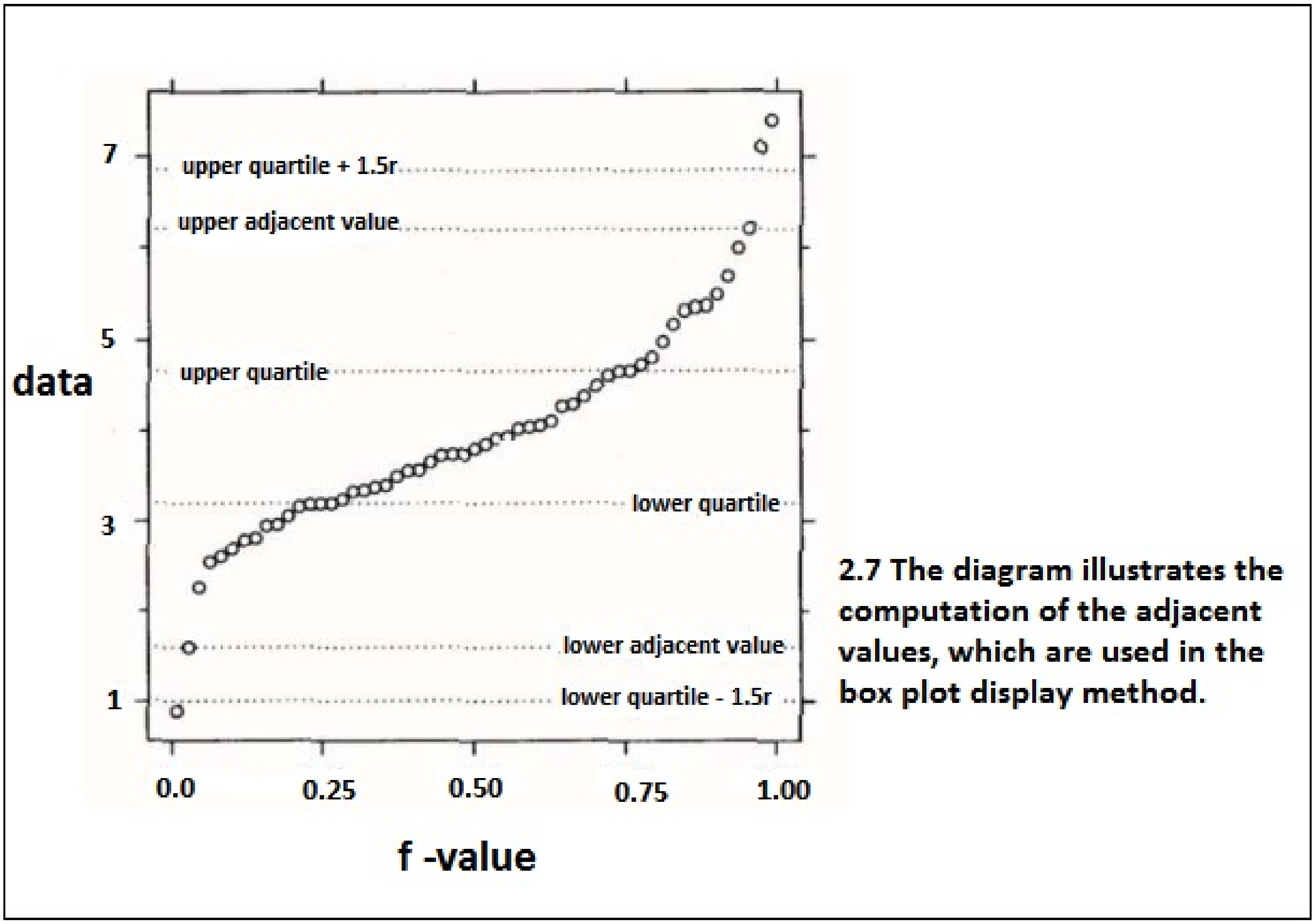

data
f -value
7
5
3
1
0.0
0.25
0.50
0.75
1.00
upper quartile + 1.5r
upper adjacent value
upper quartile
lower quartile
lower adjacent value
lower quartile - 1.5r
2.7 The diagram illustrates the computation of the adjacent values, which are used in the box plot display method.

Figure 2.

Mathematics, Grade 10

Session 3, Open-Response Questions

(40) A class of 25 students is asked to determine approximately how much time the average student spends on homework during a one-week period. Each student is to ask one of his/her friends for the information, making sure that no one student is asked more than once. The numbers of hours spent on homework per week are as follows:

8, 0, 25, 9, 4, 19, 25, 9, 9, 8, 0, 8, 25, 9, 8, 3, 7, 8, 5, 3, 25, 8, 10

a. Find the mean, median, and mode for those data. Explain or show how you found each answer.
b. Based on this sample, which measure (or measures) that you found in part a best describes the typical student? Explain your reasoning.
c. Describe a sampling procedure that would have led to more representative data.

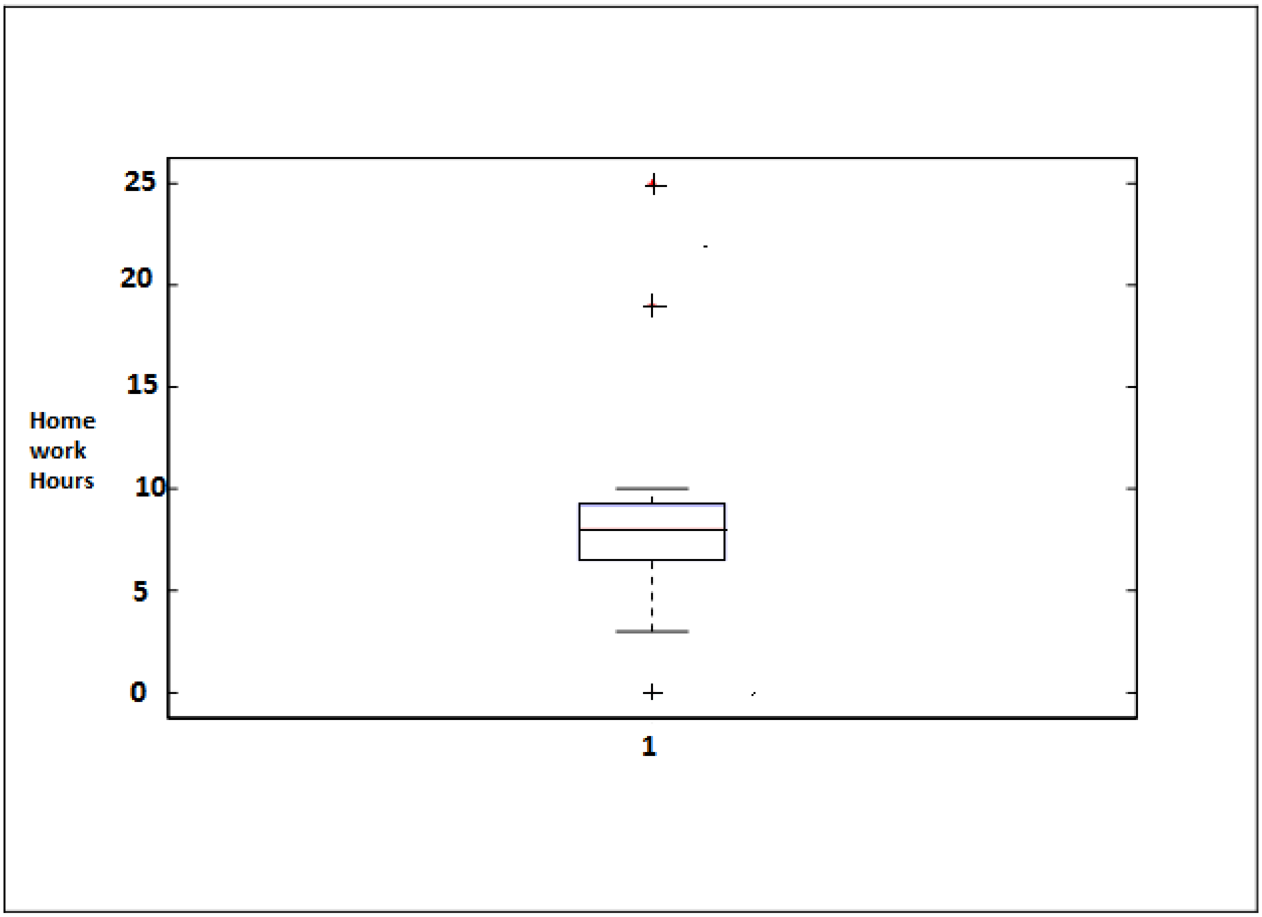


Figure 2. Boxplot of MCAS Question 40 data

Final Question: read the coin tossing procedure before answering. Provide only one T or F.

Results: There were 2 T and 7 F.

Question 1. True or False: **I don't want to take this class**.

1. Take out a coin and flip it.
2. If the coin showed a tail, answer Question 1 truthfully and hand in your paper.
3. If the coin showed a head, flip the coin a second time and answer question 2:
4. **Question 2**. True or False: "The coin showed heads on the second toss."

Massachusetts Department of Elementary & Secondary Education

Massachusetts Comprehensive Assessment System
Grade 10 Mathematics Reference Sheet

AREA FORMULAS

square .... $A = s^2$

rectangle .... $A = bh$

parallelogram .... $A = bh$

triangle .... $A = \frac{1}{2}bh$

trapezoid .... $A = \frac{1}{2}h(b_1 + b_2)$

circle .... $A = \pi r^2$

LATERAL SURFACE AREA FORMULAS

right rectangular prism .... $LA = 2(hw) + 2(lh)$

right circular cylinder .... $LA = 2\pi rh$

right circular cone .... $LA = \pi r\ell$
(ℓ = slant height)

right square pyramid .... $LA = 2s\ell$
(ℓ = slant height)

TOTAL SURFACE AREA FORMULAS

cube .... $SA = 6s^2$

right rectangular prism .... $SA = 2(lw) + 2(hw) + 2(lh)$

sphere .... $SA = 4\pi r^2$

right circular cylinder .... $SA = 2\pi r^2 + 2\pi rh$

right circular cone .... $SA = \pi r^2 + \pi r\ell$
(ℓ = slant height)

right square pyramid .... $SA = s^2 + 2s\ell$
(ℓ = slant height)

VOLUME FORMULAS

cube .... $V = s^3$
(s = length of an edge)

right rectangular prism............ $V = lwh$
OR
$V = Bh$
(B = area of a base)

sphere .... $V = \frac{4}{3}\pi r^3$

right circular cylinder .... $V = \pi r^2 h$

right circular cone .... $V = \frac{1}{3}\pi r^2 h$

right square pyramid .... $V = \frac{1}{3}s^2 h$

CIRCLE FORMULAS

$C = 2\pi r$
$A = \pi r^2$

SPECIAL RIGHT TRIANGLES