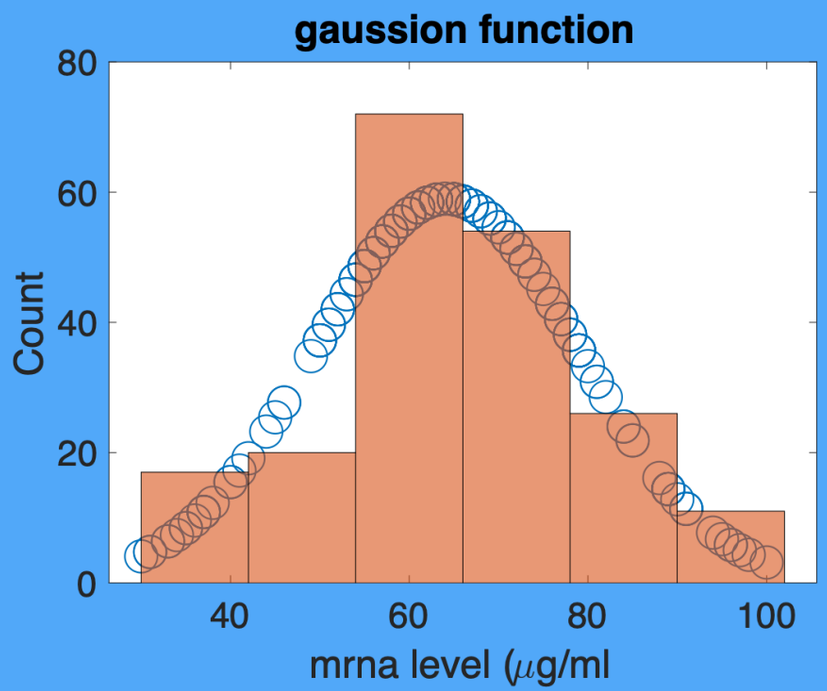
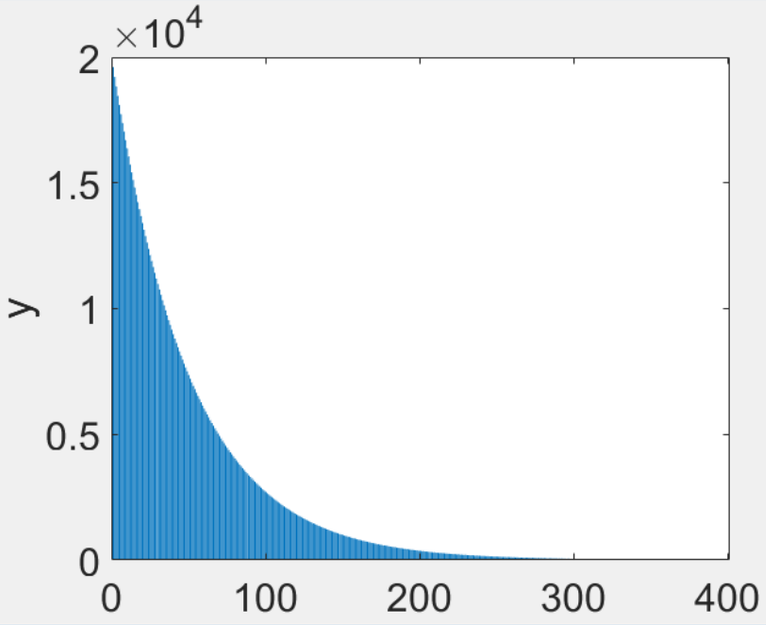
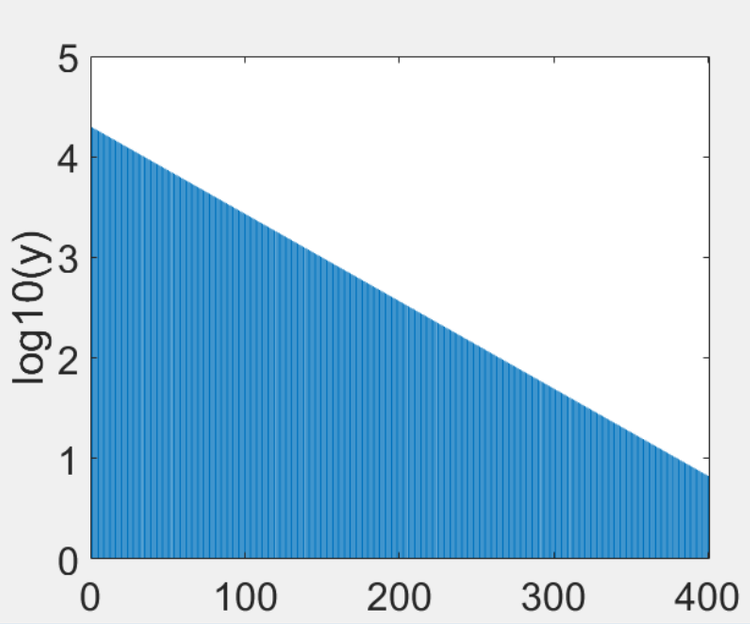
# Introduction to Scientific Computation




Halil Bayraktar

Lecture 7-data normalization, missing values and functions in programming

# How to clean missing values in the data?

mdl.Rsquared | fvalues | mdl.Coefficients | subdata1 | data1

27130x34 table

| | 1<br>Gene | 2<br>ARNA | 3<br>TRNA | 4<br>ARNA1 | 5<br>TRNA1 | 6<br>ARNA2 |
|---|---|---|---|---|---|---|
| 1 | "LOC1024... | 0 | 1.0045 | 4.0185 | 0.9162 | 2.9799 |
| 2 | "ZBTB42" | 27.8394 | 37.1676 | 55.2547 | 30.2348 | 42.7112 |
| 3 | "FCAMR" | 1.1136 | 0 | 1.0046 | 0 | 0.9933 |
| 4 | "ZNF503-... | 41.2024 | 35.1586 | 40.1853 | 16.4917 | 35.7582 |
| 5 | "NFU1" | 111.3578 | 123.5573 | 118.5465 | 133.7663 | 91.3821 |
| 6 | "ELSPBP1" | 0 | 0 | 0 | 0 | 0 |
| 7 | "ZRANB3" | 190.4218 | 183.8291 | 141.6531 | 169.4984 | 155.9455 |
| 8 | "MECR" | 259.4637 | 291.3139 | 188.8708 | 237.2977 | 289.0455 |
| 9 | "LOC1057... | 0 | 0 | 0 | 0 | 0 |
| 10 | "LINC003... | 2.2272 | 2.0091 | 3.0139 | 4.5810 | 6.9530 |
| 11 | "AARSD1" | 1.1136 | 0 | 0 | 0 | 0 |
| 12 | "DEXI" | 485.5200 | 435.9663 | 492.2695 | 308.7619 | 511.5410 |
| 13 | "DCHS1" | 1.1559e+03 | 1.0035e+03 | 1.2116e+03 | 1.3138e+03 | 1.3479e+03 |
| 14 | "PSMD2" | 1.2550e+03 | 1.8232e+03 | 1.3914e+03 | 1.4861e+03 | 1.5545e+03 |
| 15 | "GABRR1" | 3.3407 | 4.0181 | 2.0093 | 1.8324 | 5.9597 |
| 16 | "PKNOX2" | 780.6181 | 676.0491 | 640.9550 | 244.6274 | 522.4672 |
| 17 | "TIPARP" | 309.5747 | 294.3275 | 372.7184 | 721.0553 | 238.3881 |
| 18 | "ADAM20" | 113.5849 | 91.4123 | 74.3427 | 89.7883 | 100.3216 |
| 19 | "LOC2847... | 0 | 0 | 0 | 0 | 0 |
| 20 | "MIR4715" | 0 | 0 | 0 | 0 | 0 |
| 21 | "PURB" | 1.3274e+03 | 1.6374e+03 | 1.3000e+03 | 1.2341e+03 | 1.2416e+03 |

# Solution 1:
# We can use the rmmissing function that removes all rows with NaN

27130x34 table

| | 30 TRNA13 | 31 TRNA14 | 32 Young_vs_Old | 33 Old_vs_AD | 34 Young_vs_AD |
|---|---|---|---|---|---|
| 1 | 16 | 2.5888 | 0.9477 | NaN | 0.6577 | 0.9316 |
| 2 | 74 | 28.4767 | 37.9071 | 0.2173 | 0.9606 | 0.1592 |
| 3 | 0 | 0 | 0 | NaN | NaN | NaN |
| 4 | 79 | 11.2181 | 36.9594 | 0.7778 | 0.4949 | 0.2141 |
| 5 | 05 | 134.6172 | 93.8201 | 0.2529 | 0.5152 | 0.0180 |
| 6 | 0 | 0 | 0 | NaN | NaN | NaN |
| 7 | 97 | 217.4585 | 149.7331 | 0.8257 | 0.0248 | 0.0035 |
| 8 | 34 | 225.2249 | 373.3850 | 0.8962 | 0.1639 | 0.0661 |
| 9 | 0 | 0 | 0 | NaN | NaN | NaN |
| 10 | 0 | 0 | 4.7384 | NaN | 0.0820 | 0.4243 |
| 11 | 0 | 0 | 0.9477 | NaN | NaN | NaN |
| 12 | 15 | 356.3903 | 397.0770 | 0.6785 | 0.7987 | 0.3801 |
| 13 | 03 | 1.2452e+03 | 1.4860e+03 | 0.8661 | 0.9385 | 0.7631 |
| 14 | 03 | 1.6439e+03 | 1.5068e+03 | 0.4814 | 0.2136 | 0.0121 |
| 15 | 0 | 1.7259 | 1.8954 | NaN | 0.3701 | 0.0193 |
| 16 | 94 | 220.0473 | 470.9958 | 0.3814 | 0.3493 | 0.9747 |
| 17 | 78 | 739.5315 | 273.8789 | 0.3138 | 0.7563 | 0.0845 |
| 18 | 97 | 43.1465 | 71.0758 | 0.1019 | 0.7264 | 0.1319 |
| 19 | 0 | 0 | 0 | NaN | NaN | NaN |
| 20 | 0 | 0 | 1.8954 | NaN | NaN | NaN |
| 21 | 03 | 1.1926e+03 | 981.7941 | 0.5825 | 0.8988 | 0.3862 |
| 22 | 03 | 1.4256e+03 | 1.1903e+03 | 0.3570 | 0.9632 | 0.2343 |
| 23 | 60 | 73.3491 | 60.6514 | 0.6285 | 0.2586 | 0.0371 |
| 24 | 24 | 46.5983 | 49.2792 | 0.8579 | 0.3470 | 0.1649 |

128x33 double

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 27.8394 | 37.1676 | 55.2547 | 30.2348 | 42.7112 | 32.2643 |
| 2 | 41.2024 | 35.1586 | 40.1853 | 16.4917 | 35.7582 | 32.2643 |
| 3 | 111.3578 | 123.5573 | 118.5465 | 133.7663 | 91.3821 | 86.7800 |
| 4 | 190.4218 | 183.8291 | 141.6531 | 169.4984 | 155.9455 | 92.3428 |
| 5 | 259.4637 | 291.3139 | 188.8708 | 237.2977 | 289.0455 | 189.1358 |
| 6 | 485.5200 | 435.9663 | 492.2695 | 308.7619 | 511.5410 | 493.9782 |
| 7 | 1.1559e+03 | 1.0035e+03 | 1.2116e+03 | 1.3138e+03 | 1.3479e+03 | 1.4652e+03 |
| 8 | 1.2550e+03 | 1.8232e+03 | 1.3914e+03 | 1.4861e+03 | 1.5545e+03 | 1.3206e+03 |
| 9 | 780.6181 | 676.0491 | 640.9550 | 244.6274 | 522.4672 | 406.0857 |
| 10 | 309.5747 | 294.3275 | 372.7184 | 721.0553 | 238.3881 | 354.9078 |
| 11 | 113.5849 | 91.4123 | 74.3427 | 89.7883 | 100.3216 | 160.2091 |
| 12 | 1.3274e+03 | 1.6374e+03 | 1.3000e+03 | 1.2341e+03 | 1.2416e+03 | 1.4953e+03 |
| 13 | 1.3285e+03 | 1.1422e+03 | 977.5067 | 1.1333e+03 | 1.3072e+03 | 1.1738e+03 |
| 14 | 150.3330 | 145.6569 | 190.8800 | 127.3528 | 113.2343 | 140.1830 |
| 15 | 66.8147 | 36.1631 | 46.2131 | 39.3969 | 31.7851 | 52.2905 |
| 16 | 153.6738 | 171.7747 | 161.7457 | 154.8391 | 164.8851 | 141.2956 |
| 17 | 146.9923 | 150.6796 | 110.5095 | 95.2856 | 85.4224 | 74.5418 |
| 18 | 197.1033 | 381.7216 | 150.6948 | 362.8182 | 73.5030 | 70.0915 |
| 19 | 20.0444 | 12.0544 | 21.0973 | 6.4135 | 20.8590 | 66.7538 |
| 20 | 1.4610e+03 | 747.3707 | 816.7656 | 209.8115 | 345.6627 | 269.2404 |
| 21 | 228.2835 | 198.8971 | 337.5563 | 285.8567 | 128.1336 | 133.5076 |

Solution 2:

Can we write our own code that removes the data with zeros?

How to find Nan values in the data and remove these arrays?

```
%%

[r,c,l]=find(isnan(subdata1))
rc=sort(r)
rcs=circshift(rc,-1,1)
rrun=rcs-rc
[r,c,l] =find(rrun~=0)
 uniqlst= rc(r,c)
 uniqlst1= uniqlst(:,1)
%%
clear sdata
j=1
for i=1:200
    [r,c,l]= find(uniqlst1(:,1)==i)
      if isempty(r)
          sdata(j,:)=subdata1(i,:)
          j=j+1



    end
end
%%
```

# Data normalization

1. Divide by the max value or
2. Find largest of smallest value and use the following formula, you can scale the values between 0 and 1

Data-min/max-min

$$x' = (x - x_{min})/(x_{max} - x_{min})$$

```
%%
data1=randi([10,100],10,5)
%%
mad=max(max(data1))
mid=min(min(data1))
ndata=(data1-mid)/(mad-mid)
```

10x5 double

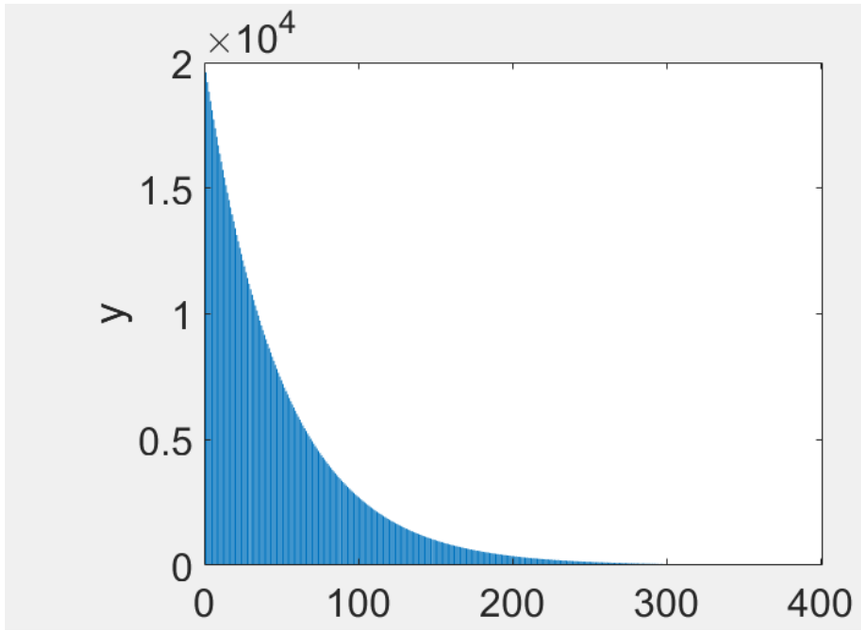
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 87 | 66 | 74 | 20 | 17 |
| 2 | 15 | 46 | 81 | 81 | 51 |
| 3 | 42 | 42 | 61 | 51 | 80 |
| 4 | 33 | 68 | 12 | 37 | 84 |
| 5 | 49 | 76 | 10 | 42 | 67 |
| 6 | 37 | 76 | 99 | 53 | 43 |
| 7 | 97 | 97 | 28 | 49 | 90 |
| 8 | 21 | 14 | 84 | 73 | 32 |
| 9 | 29 | 48 | 42 | 71 | 16 |
| 10 | 91 | 10 | 51 | 10 | 76 |
| 11 | | | | | |

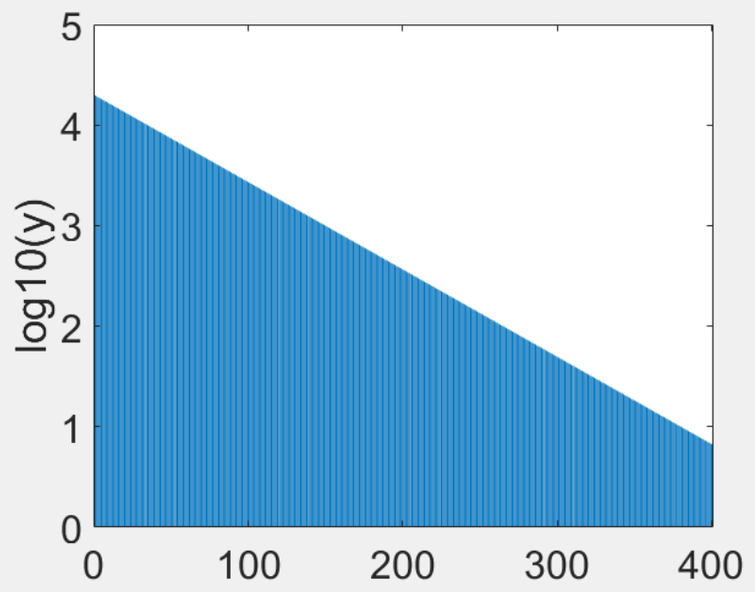
10x5 double

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0.8652 | 0.6292 | 0.7191 | 0.1124 | 0.0787 | |
| 2 | 0.0562 | 0.4045 | 0.7978 | 0.7978 | 0.4607 | |
| 3 | 0.3596 | 0.3596 | 0.5730 | 0.4607 | 0.7865 | |
| 4 | 0.2584 | 0.6517 | 0.0225 | 0.3034 | 0.8315 | |
| 5 | 0.4382 | 0.7416 | 0 | 0.3596 | 0.6404 | |
| 6 | 0.3034 | 0.7416 | 1 | 0.4831 | 0.3708 | |
| 7 | 0.9775 | 0.9775 | 0.2022 | 0.4382 | 0.8989 | |
| 8 | 0.1236 | 0.0449 | 0.8315 | 0.7079 | 0.2472 | |
| 9 | 0.2135 | 0.4270 | 0.3596 | 0.6854 | 0.0674 | |
| 10 | 0.9101 | 0 | 0.4607 | 0 | 0.7416 | |
| 11 | | | | | | |
| 12 | | | | | | |

# Log scaling

It is used to scae the values and compress a wide range to a narrow range of values.




```
%%
%log scaling
% example
clear y
x=1:1:400

for i=1:400
    y(1,i)=20000/exp(0.02*x(1,i))
end
figure(1)
subplot(1,2,1)
bar(x,y)
ylabel('y')
set(gca,'fontsize',24)

subplot(1,2,2)
bar(x,log10(y))
ylabel('log10(y)')
set(gca,'fontsize',24)
```

# Subgrouping the data

```
% mean values of each gene for young, old, and AD samples
allmeansdata=subyoungmean';
allmeansdata(:,2)=suboldmean';
allmeansdata(:,3)=subADmean';
% young old diff
allmeansdata(:, 4)= allmeansdata (:, 1)- allmeansdata (:, 2);
% young alz diff
allmeansdata(:, 5)= allmeansdata (:, 1)- allmeansdata (:, 3);
% old alzhemier diff
allmeansdata(:, 6)= allmeansdata (:, 2)- allmeansdata (:,3);
```

128x11 double

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0.4866 | 0.6443 | 0.6451 | -0.1577 | -0.1585 | -8.7756e-04 |
| 2 | 0.5456 | 0.6937 | 0.6966 | -0.1481 | -0.1510 | -0.0029 |
| 3 | 0.2530 | 0.4804 | 0.4837 | -0.2274 | -0.2307 | -0.0033 |
| 4 | 0.6238 | 0.7190 | 0.7131 | -0.0952 | -0.0893 | 0.0059 |
| 5 | 0.6202 | 0.4621 | 0.4699 | 0.1581 | 0.1504 | -0.0077 |
| 6 | 0.7676 | 0.6858 | 0.6771 | 0.0818 | 0.0905 | 0.0087 |
| 7 | 0.8241 | 0.6859 | 0.6761 | 0.1382 | 0.1480 | 0.0098 |
| 8 | 0.4072 | 0.5010 | 0.4909 | -0.0938 | -0.0837 | 0.0101 |
| 9 | 0.3752 | 0.4150 | 0.4031 | -0.0398 | -0.0279 | 0.0119 |
| 10 | 0.5472 | 0.6547 | 0.6675 | -0.1075 | -0.1202 | -0.0127 |
| 11 | 0.7443 | 0.6082 | 0.6215 | 0.1361 | 0.1229 | -0.0132 |
| 12 | 0.6721 | 0.5565 | 0.5408 | 0.1157 | 0.1314 | 0.0157 |
| 13 | 0.6649 | 0.5888 | 0.6067 | 0.0761 | 0.0581 | -0.0180 |
| 14 | 0.4952 | 0.3110 | 0.3305 | 0.1843 | 0.1647 | -0.0196 |
| 15 | 0.8474 | 0.7785 | 0.8004 | 0.0689 | 0.0470 | -0.0219 |
| 16 | 0.4533 | 0.6328 | 0.6089 | -0.1795 | -0.1556 | 0.0239 |
| 17 | 0.7804 | 0.6416 | 0.6657 | 0.1388 | 0.1147 | -0.0241 |
| 18 | 0.5196 | 0.5083 | 0.5332 | 0.0113 | -0.0136 | -0.0250 |
| 19 | 0.3145 | 0.3653 | 0.3352 | -0.0508 | -0.0207 | 0.0301 |
| 20 | 0.5720 | 0.5746 | 0.6052 | -0.0026 | -0.0332 | -0.0306 |
| 21 | 0.6814 | 0.6760 | 0.6448 | 0.0054 | 0.0367 | 0.0313 |

# IF **statement**

I**f** statement is used to choose whether or not a statement,

or group of statements, is executed. **if** statement is written as,

If logic expression or relational expression
arguments
elseif logic or relational expression
Arguments
end
end

## Switch Statement

A switch statement can often be used in place of a nested if-else or an if statement with many elseif clauses.
Switch statements are used when an expression is tested to see whether it is equal to one of several possible values.

```
switch switch_expression
        case caseexp1
action1
case caseexp2
action2
case caseexp3
action3
% etc: there can be many of these
otherwise
actionn end
```

# Numeric values as a case

```
month=input('select a month:')

switch month
    case {1,3,5,7,8,10,12} % group different case varibles if return same results
        days=31;
        fprintf('days: %i',days)
    case {4,6,9,11}
        days=30
        fprintf('days: %i',days)
    case 2
        days=28
        fprintf('days: %i',days)
    otherwise
        disp('entered valuer is not present, please select a number between 1 and 12')
end
```

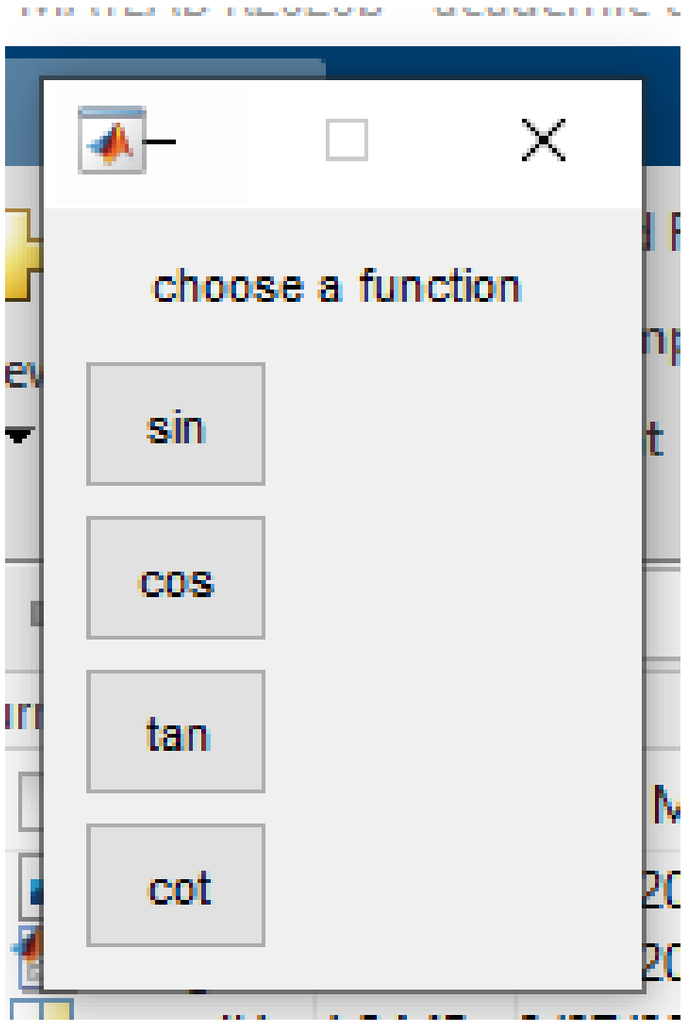
# String values as a case

```
%% enter a string
% if and case statements
food=input('enter a food: ')

switch food
   case {'pizza','burger','chips'}
      disp('food is not healthy')
   case {'soup','vegetable','rice'}
      disp('food is healthy')
      if food=='soup'
       disp('soup has high protein nutrients')

      else
       disp('it has high carbohydrate nutrient')
      end
   otherwise
      disp('i do not know')
end
```

```
Error in matlab.internal.editor.evaluateCode


enter a food: 'pizza'

food =

     'pizza'

food is not healthy
fx >>
```

# Create a menu tab in the matlab

```
%% menu in matlab
degree=input('enter a angle in degrees: ')
choice=menu('function','sin','cos','tan','cot')
switch choice
    case {1}
        result=sind(degree)
    case {2}
        result=cosd(degree)
    case {3}
        result=tand(degree)
    case {4}
        result=cotd(degree)
end
fprintf('%i',result)
```




```
result =

     0.5000

5.000000e-01>>
```

# Is Functions in Matlab

```
%%  is functions In Matlab

isnumeric(56)
%%
isletter('334678888 Maslak')
%%
y=[]
x=[1,2,3]
isempty(y)
x=45
isstring(x)
%%
isnumeric(34)
%%
a=[1,2]
isempty(a)
a=["maslak"]
isstring(a)
isletter('a')
a=[1,2]
isempty(a)
isreal(0)
%%
a=[1,2,3,NaN]
find(isnan(a))
```

# How to write a function in Matlab

function [output value 1, output value 2, ...] =name(input1, input2,input3…])

body of the function

end

```
function [result]=name(x,y,z,.....)
% function to compute the factorial of a number

body of the function



end
```

# Any matlab function consists of

1. Each function in the matlab starts with a word of “function”, please do not use this word any other place in your code.

2. Find a name for your function
(note: The name should be the same as the name of the M-file in which this function is stored)

3. The input values are shown in parentheses. They are separated by commas if there is more than one input values.

4. Output values are shown in square brackets. If there are more than one inputs, they are separated by comma.

## Example: Function to compute the factorial of a number

```
function [fac]=factfun(n)
% function to compute factorial of a number


   fac=1;
   for i=1:n;
       fac=fac*i;
   end
   format short
     %result=fac;
     fprintf('%i factorial equals to %i \n',n,fac)
```

# Example: Function to compute the factorial of a number

```
function [fac]=factfun2(n)
% function to compute factorial of a number
 if n==0
     fac=1
      fprintf('%i factorial equals to %i \n',n,fac)
     return
 else
     fac=prod(1:n)
 end
   format short
     %result=fac;
     fprintf('%i factorial equals to %i \n',n,fac)
     fprintf('it works')
end
```

# Anonymous Functions

Anonymous functions are functionsdefine d in your program.

it does not need other file to save it. We use a symbol of @

@(input variable) expression

quad function in matlab evaluates the integration of a function between two values.

```
a = @(x) sin(x).*cos(x);
    quad(a,0,3)
```

Ans = 0.01

$$I = \frac{2}{2}\int \sin x \cos x dx$$

$$I = \frac{1}{2}\int 2 \sin x \cos x dx$$

$$I = \frac{1}{2}\int \sin 2x dx$$

$$I = -\frac{1}{2}\frac{\cos 2x}{2}$$

$$\boxed{I = \frac{-\cos 2x}{4} + C}$$

# Examples of Anonymous Functions

```
sqr = @(x) x.^2;
    a = sqr(5)
```

```
function1=@(x,y,z) x.^2+y.^3+z.^4
function1(1,2,3)
```

```
      function1 =

  function_handle with value:

    @(x,y,z)x.^2+y.^3+z.^4

ans =

    90
```