User-defined functions

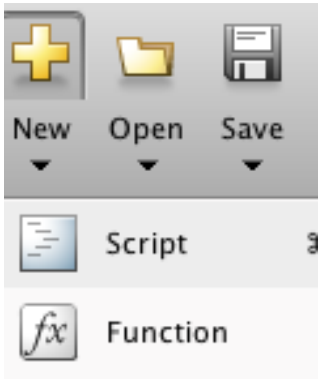
- Function handles, anonymous functions
  - One-liners, defined in the command window or in a script
    `>> f=@(x)x.^2` to be read: f is the function which "at x" returns the value x^2. (In math: $f = x \to x^2$)
    Several inputs allowed:
    `>> g=@(x,y,z)sqrt(x.^2+y.^2+z.^2).`
- Functions in m-files
  If more lines are needed, local variables, control structures (`for`, `while`, `if - else`, etc.), then write an m-file
- Inline-function is older, more restrictive version of function handle. We will not use them actively, the only reason to know about them, is old Matlab-codes. (`help inline`)

User-defined functions, m.file

function [out1,out2,out3]=funname(in1,in2)
file: funname.m on matlabpath.

- Keyword function
- Each out_k-argument must be assigned a value, the last assignement is the value returned.
- Variable scope: All variables defined in the function body are local, i.e. they are cleared when function stops running. (Note the difference with a script).
- Function needn't have output-arguments it can display text or graphics, write to files etc. In such cases it may often be more natural to use a script, though.

Examples of writing functions

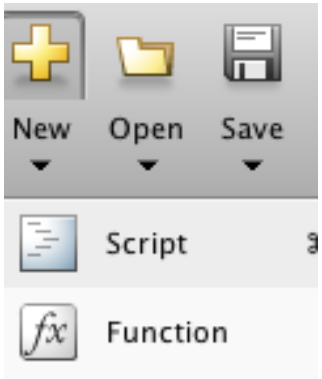


To start editing a function, open the editor on the top left "New"-button. Instead of script, this time click Function. Or on the command line:

```
>> edit myfunction
```

As our first example, let's write a function that computes the mean of the components of the input vector.
Let's first give some thought of the expression.

Examples of writing functions



To start editing a function, open the editor on the top left "New"-button. Instead of script, this time click Function. Or on the command line:

```
>> edit myfunction
```

As our first example, let's write a function that computes the mean of the components of the input vector.
Let's first give some thought of the expression.

```
x=1:10;
avg=sum(x)/length(x)
```

Example 1, mean of a vector


```
function y=mymean(x)
% Compute the mean (average) ox x-values.
% Input: vector x
% Result : mean of x
% Exampe call: r=mymean(1:10)
%
y=sum(x)/length(x);
```

```
>> help mymean
  Compute the mean (average) ox x-values.
  ...
>> r=mymean(1:10)
r =
    5.5000
```

Example function **stats**

Standard deviation is given by: $\sigma = \sqrt{\frac{1}{N}\sum_{k=1}^{n}(x_k - \mu)^2}$.

```
function [avg,sd,range] = stats(x)
% Returns the average (mean), standard deviation
% and range of input vector x
N=length(x);
avg=sum(x)/N;
sd = sqrt(sum(x - avg).^2)/N);
range=[min(x),max(x)];
```

- **Keyword:** "function"
- **Input:** x (can be more than one input)
- **Outputs:** [avg,sd,range]
- filename must be funname.m (here stats.m).
- help stats displays the first contiguous set of comments.

Calling example function *stats*

- Example call of function stats:

```
x=linspace(0,pi);
y=sin(x);
[a,s,r]=stats(y)   % Function call
plot(x,y,'b')      % 'b' for blue
hold on
plot([0;pi],[a;a],'k') % 'k' for blacK
shg                % show graphics
% help errorbar
```

- Create a script with headerline: %% `statscript.m`
- Add errorbars of length $0.1 \times s$ at each 10^{th} point of the graph.
- Add title and legend and make own modifications.
- Run and then publish your file.