1.017/1.010 Recitation 3
MATLAB Tests and Loops

More MATLAB operations

Feature	**Classical Notation** (Each element of result is true or false)	**MATLAB notation** (Each element of result is 1or 0)
Relational Operations	*a* ,*b* scalars or arrays of numbers	a, b scalars or arrays of numbers
Greater than	$a > b$, $a(i) > b(i)$	a > b
Less than	$a < b$, $a(i) < b(i)$	a < b
Greater than or equal	$a \geq b$, $a(i) \geq b(i)$	a >= b
Less than or equal	$a \leq b$, $a(i) \leq b(i)$	a <= b
Logical Operations	*A*,*B* scalars or arrays of true or false	A,B arrays of 1 or 0
Logical OR	*A* OR *B*, $A \cup B$	A|B
Logical AND	*A* AND *B*, $A \cap B$	A&B
Logical NOT	NOT *A*	~A

Relational and logical operations for arrays

Operations are elementwise:

`[1 3 -2] > [0 4 1]` yields `[1 0 0]`

1 in this result signifies true, 0 signifies false

`([1 5] > [2 3])|([-3 9] <= [-3 8])` yields `[1 1]`

Using relational and logical operations in tests

Runoff generation:

```
if (precip>infilt)  % begin test
  runoff = precip - infilt % if inequality true
else
  runoff=0   % if inequality false
end   %end test
```

Recursive computations and loops

1D translation of a particle in a time-dependent velocity field:

```
delta_t=.1     % define time step
x(1)=2         % initialize x
for(t=1:40)    % begin time loop
v(t)=10*cos(t);
x(t+1) = x(t) + v(t)*delta_t;  % update x(t)
end  % end time loop
plot(0:40,x)
```

User-defined functions

User defined function is stored in m-file of same name:

stats.m . . . .

```
function [mdata, vdata]=stats(data)
% Computation of sample mean and variance
   n=length(data)
   mdata=sum(data)/n   % sample mean
   vdata=sum((data-mdata).^2)/n % sample variance
return
```

Run function by typing name, specifying appropriate input and output variables:

```
>> load arsenic.txt
>> [marsenic, varsenic]=stats(arsenic)
```

Function returns `marsenic` and `varsenic`

Application to Virtual Experiments

MATLAB's vector, loop and logical operations can be used to derive probabilities from virtual experiments. The procedure is:

1. Generate a random outcome for each replicate (repeated experiment)

2. Test the outcome to see if it satisfies the requirements that define a particular event

3. Record the number of replicates that yield the event

4. Compute the probability of the event by dividing the number of event occurrences by the total number of replicates

Sometimes one or more steps are combined.

Example: Coin toss

Use virtual experiment to compute probability of obtaining at least 3 heads in 5 coin tosses.
Write user-defined function `toss`:

```
function toss
nrep=10000   % number of replicates
ntoss=5      % number of tosses
for i=1:nrep
% generate random outcome for repl. i
    toss=unidrnd(2,ntoss,1)-1; % head=1, tail=0
    heads(i)=sum(toss);  % count heads
end
% test outcomes and count event occurrences
n_atleast_3=sum(heads>=3)
% compute probability of event
p_atleast_3=n_atleast_3/nrep
return
```