demo_01-Battery

September 20, 2023

[ ]:
```
graphics_toolkit('gnuplot')   # 'qt', 'fltk'
```

[ ]:
```
function [T, S, i] = sim_battery(N)
    T = -1;
    S = 0;
    t = ones(N, 1) * -1;    % time series
    s = ones(N, 1) * -1;    % status series, the number of available batteries
    i = 0;                  % time stamp
    charging = 0;           % the rest of charging time
    t(1) = 0;
    s(1) = 2;
    % initial status
    r = randi(6);  % time goes here!
    while (i < N - 1)  % in fact, should be infinite
        i = i + 1;  % what's the next status? (and time)
        if (s(i) == 2)
            % one using one ready
            s(i + 1) = 1;
            t(i + 1) = t(i) + r;
            charging = 2.5;
            r = randi(6);  % go!
            % one using one charging
        elseif (s(i) == 1)
            % one using one charging
            charging = charging - r; % till the using one empty
            if (charging > 0)
                % no power!
                t(i + 1) = t(i) + r;
                s(i + 1) = 0;
                r = 0;  % the end!
                T = t(i + 1);
                S = sum((t(2:end) - t(1:end-1)) .* s(1:end-1)) / T;
                return;  % the only way to stop!
            else
                % already full, the using one just keep on using
                % equal to one using one ready.
                t(i + 1) = t(i) + 2.5;
```

```
                s(i + 1) = 2;
                charging = 0;
                r = r - 2.5;  % keep going!
                if (r < 0)
                    % assert!
                    fprinft(stdout, 'error!\n');
                    return;
                end
            end
        end
    end
    fprintf('not finish yet!\n');
    return;
end
```

[ ]:
```
Tb = 0;
Sb = 0;
K = 0;
NN = 1000;
T = zeros(NN+1,1);
S = zeros(NN+1,1);
```

[ ]:
```
for I = 0:NN
    [Ti, Si, i] = sim_battery(100000);
    if (Ti ~= -1)
        Tb = Tb + Ti;
        Sb = Sb + Si;
        K = K + 1;
        T(K) = Ti;
        S(K) = Si;
    end
end
```

[ ]:
```
Tb = Tb / K
Sb = Sb / K
```

[ ]:
```
plot(1:K, S(1:K), '.'); axis([0,NN,1,2]);
```

[ ]:
```
K
```

[ ]:
```

```

demo_01-MontyHall

September 20, 2023

[1]:
```
graphics_toolkit('gnuplot')
```

```
warning: using the gnuplot graphics toolkit is discouraged

The gnuplot graphics toolkit is not actively maintained and has a number
of limitations that are ulikely to be fixed.  Communication with gnuplot
uses a one-directional pipe and limited information is passed back to the
Octave interpreter so most changes made interactively in the plot window
will not be reflected in the graphics properties managed by Octave.  For
example, if the plot window is closed with a mouse click, Octave will not
be notified and will not update it's internal list of open figure windows.
We recommend using the qt toolkit instead.
```

[2]:
```
N = 10000;
decision = true;  % true: I want to change,    % false: I don not want to␣
 ↪change.
win = 0;
loss = 0;
```

[3]:
```
% do the test for N times
for i = 1:N
    % 1st stage: open one door by operator
    open = 1;    %% assume it
    car = randi(3);
    picked = randi(3);
    while (open == picked || open == car)
        % should not open the picked or real car
        open = open + 1;
    end
    % 2nd stage: choose by user
    if (decision)
        % change
        re_pick = 1;
        while (re_pick == picked || re_pick == open)
            re_pick = re_pick + 1;
        end
        picked = re_pick;
```

```
    else
        % do not repick, then do nothing
    end
    % 3rd stage: record the result
    if (picked == car)
        win = win + 1;
    else
        loss = loss + 1;
    end
end
```

[4]:
```
if (win + loss ~= N)
    % assert!
    fprintf('error!\n');
else
    win_rate = win / N;
    fprintf('The win rate is : %5.2f \n', win_rate*100);
end
```

```
The win rate is : 66.71
```

[ ]:
```

```

demo_01-PIcalc

September 20, 2023

[ ]:
```
# graphics_toolkit('gnuplot')
```

[ ]:
```
format long
```

[ ]:
```
tic;
N = 100000;
m = 0;
for i = 1:N
    x = rand() * 2 - 1.0;  %(-1,1)
    y = rand() * 2 - 1.0;
    if ((x*x + y*y) < 1.0)
        m = m + 1;
    end
end
PI = 4 * m / N
toc;
```

[ ]:
```
%% try the vectorized, faster than for-loop
N=1000000; tic; num=rand(N,2); est_pi=sum(num(:,1).*num(:,1) + num(:,2).*num(:
 ↪,2) < 1.0)/N*4.0;toc; est_pi
```

[ ]:
```

```

demo_01-Quadrature

September 20, 2023

[1]:
```
graphics_toolkit('gnuplot')
```

```
warning: using the gnuplot graphics toolkit is discouraged

The gnuplot graphics toolkit is not actively maintained and has a number
of limitations that are ulikely to be fixed.  Communication with gnuplot
uses a one-directional pipe and limited information is passed back to the
Octave interpreter so most changes made interactively in the plot window
will not be reflected in the graphics properties managed by Octave.  For
example, if the plot window is closed with a mouse click, Octave will not
be notified and will not update it's internal list of open figure windows.
We recommend using the qt toolkit instead.
```

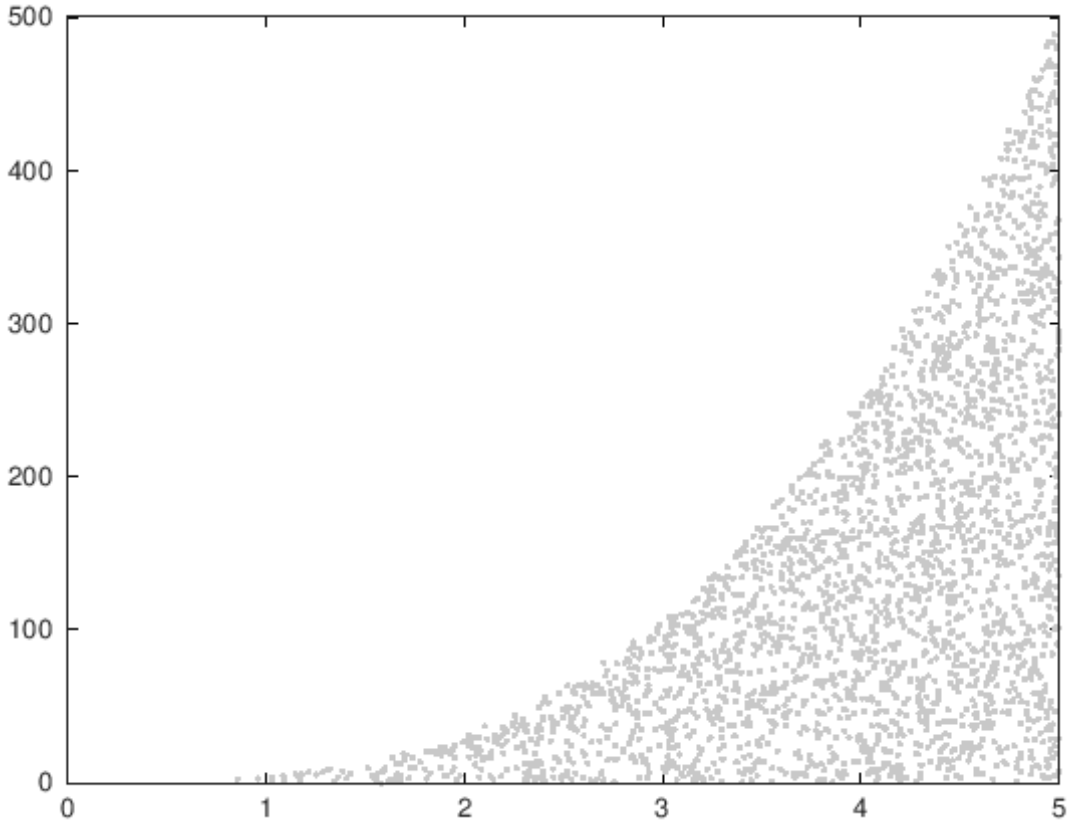
0.1 1. inegrate 2d examples

[2]:
```
ntrials = 10000;
x =   5*rand(1,ntrials);
y = 500*rand(1,ntrials);
nhits = sum(y<4*x.^3);
p = nhits/ntrials;
area = p*500*5
```

```
area = 644.75
```

[3]:
```
idx_pos = y<4*x.^3;
```

[4]:
```
plot(x(idx_pos),y(idx_pos),'.','color',[0.8,0.8,0.8]);
```



[ ]:

0.2 2. Integration in 3D

[5]:
```
N = [10,1000,100000];   % last case run 10s+ on my tablet
for i =1:length(N)
    x=2*rand(N(i),1);
    y=rand(N(i),1);
    z=2*rand(N(i),1);
    %
    nhit=sum((x.^2+y.^2+z.^2<=4)&((x-1).^2+y.^2<=1));
    V(i)=16*nhit/N(i);
        V(i)
end
V
```

```
ans = 12.800
ans = 9.4400
ans = 9.6429
V =
```

```
   12.8000    9.4400    9.6429
```

[6]:
```
pkg load statistics;
```

```
warning: function /usr/share/octave/packages/statistics-1.4.3/signtest.m shadows
a core library function
warning: called from
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7

warning: function
/usr/share/octave/packages/statistics-1.4.3/tests/wilcoxon_test.m shadows a core
library function
warning: called from
    /usr/share/octave/packages/statistics-1.4.3/PKG_ADD at line 16 column 5
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7

warning: function /usr/share/octave/packages/statistics-1.4.3/tests/u_test.m
shadows a core library function
warning: called from
    /usr/share/octave/packages/statistics-1.4.3/PKG_ADD at line 16 column 5
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7
```

0.3 3. Integrate with mean sampleing method

[7]:
```
ntrials = 1000000;
x = unifrnd(0,5,1,ntrials);
y = 20*x.^3;
s = mean(y)
```

```
s = 625.48
```

[ ]:
```
```

demo_rand_octave

September 20, 2023

1 Octave

1.1 0.

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

1.2 1.

[4]:
```
randi(3)
```

```
ans = 2
```

[5]:
```
randn(5,5)
```

```
ans =

   0.370102  -0.212593  -0.191943   0.695009  -0.601658
  -0.160841  -0.185804   2.377961   0.568279  -0.923177
  -0.285608  -0.439174   0.664263   1.458301  -2.079801
  -0.657213   0.021676  -0.772529  -0.158223  -0.678895
   0.882294   0.889398   1.682335   0.392982   0.375361
```

[ ]:

[ ]:

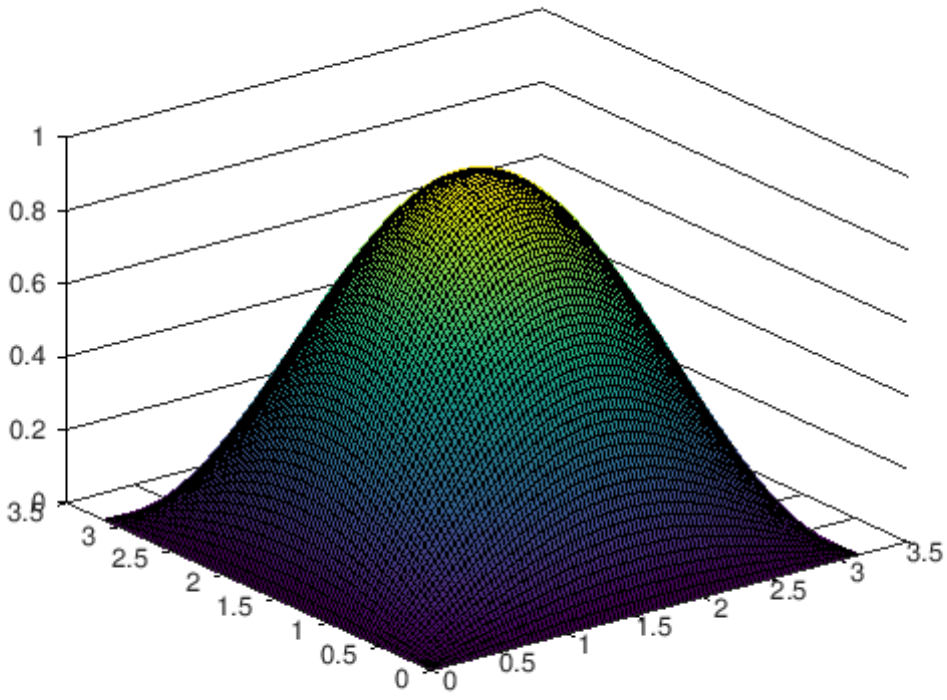
1.3 2.

[1]:
```
graphics_toolkit('gnuplot')  %    7.2.0 qt5  ,gnuplot
```

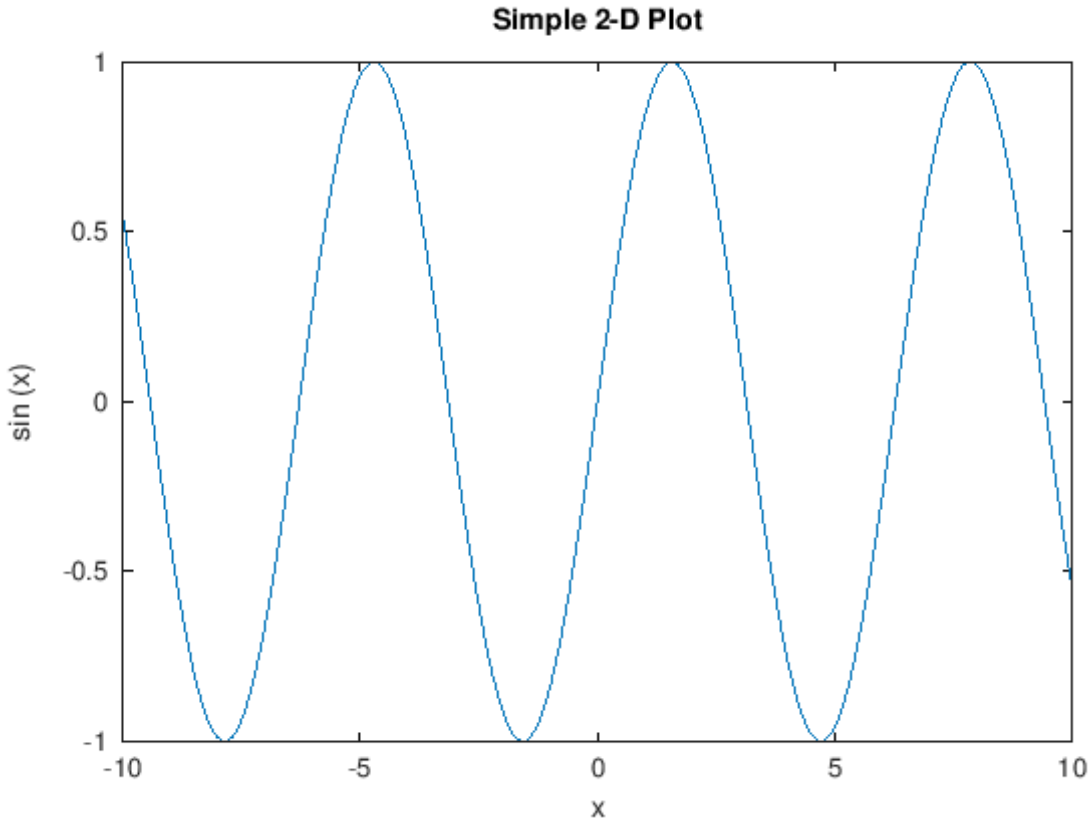
```
warning: using the gnuplot graphics toolkit is discouraged

The gnuplot graphics toolkit is not actively maintained and has a number
of limitations that are ulikely to be fixed.  Communication with gnuplot
uses a one-directional pipe and limited information is passed back to the
Octave interpreter so most changes made interactively in the plot window
will not be reflected in the graphics properties managed by Octave.  For
example, if the plot window is closed with a mouse click, Octave will not
be notified and will not update it's internal list of open figure windows.
We recommend using the qt toolkit instead.
```

[2]:
```
x = (0:100)/100*pi;
```

[3]:
```
[xx,yy] = meshgrid(x,x);
surf(xx,yy,sin(xx).*sin(yy));
```



[7]:
```
graphics_toolkit('gnuplot')
x = -10:0.1:10; # Create an evenly-spaced vector from -10..10
y = sin (x);    # y is also a vector
plot (x, y);
title ("Simple 2-D Plot");
xlabel ("x");
ylabel ("sin (x)");
```



[ ]:
```
```

1.4 3.

[10]:
```
pkg load statistics;
```

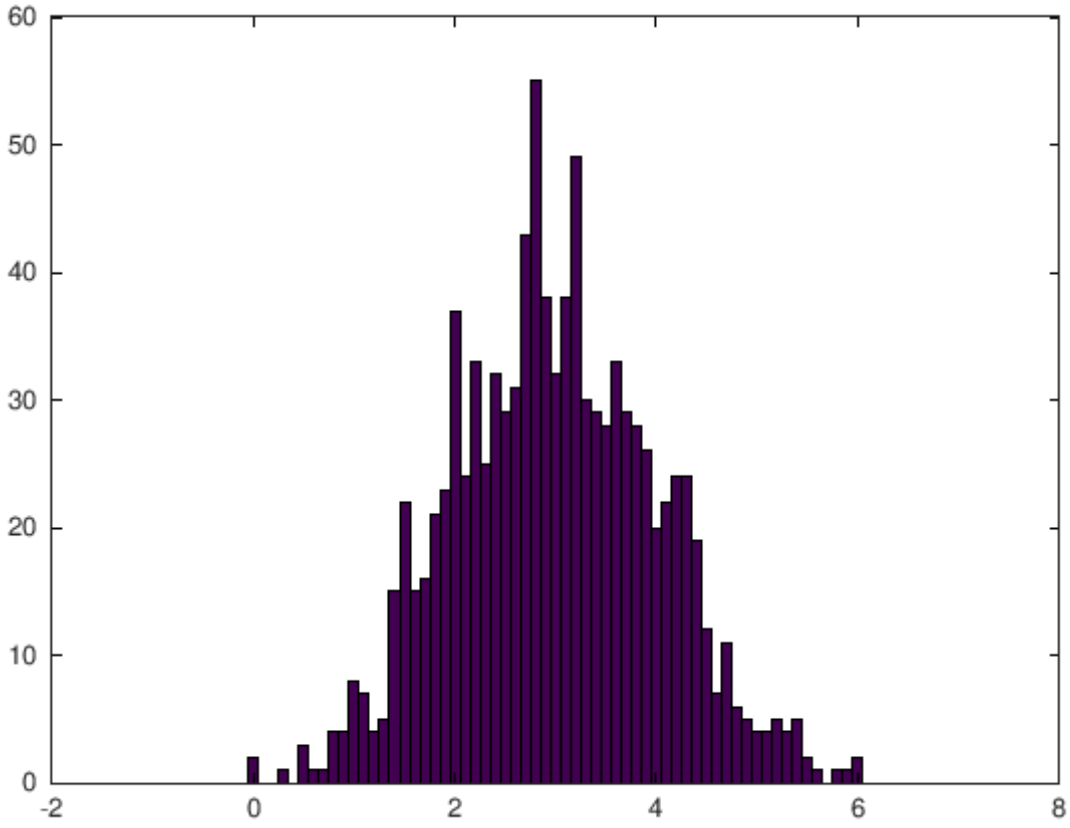
```
warning: function /usr/share/octave/packages/statistics-1.4.3/signtest.m shadows
a core library function
warning: called from
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7
```

```
warning: function
/usr/share/octave/packages/statistics-1.4.3/tests/wilcoxon_test.m shadows a core
library function
warning: called from
    /usr/share/octave/packages/statistics-1.4.3/PKG_ADD at line 16 column 5
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7

warning: function /usr/share/octave/packages/statistics-1.4.3/tests/u_test.m
shadows a core library function
warning: called from
    /usr/share/octave/packages/statistics-1.4.3/PKG_ADD at line 16 column 5
    load_packages_and_dependencies at line 56 column 5
    load_packages at line 53 column 3
    pkg at line 588 column 7
```

[ ]:

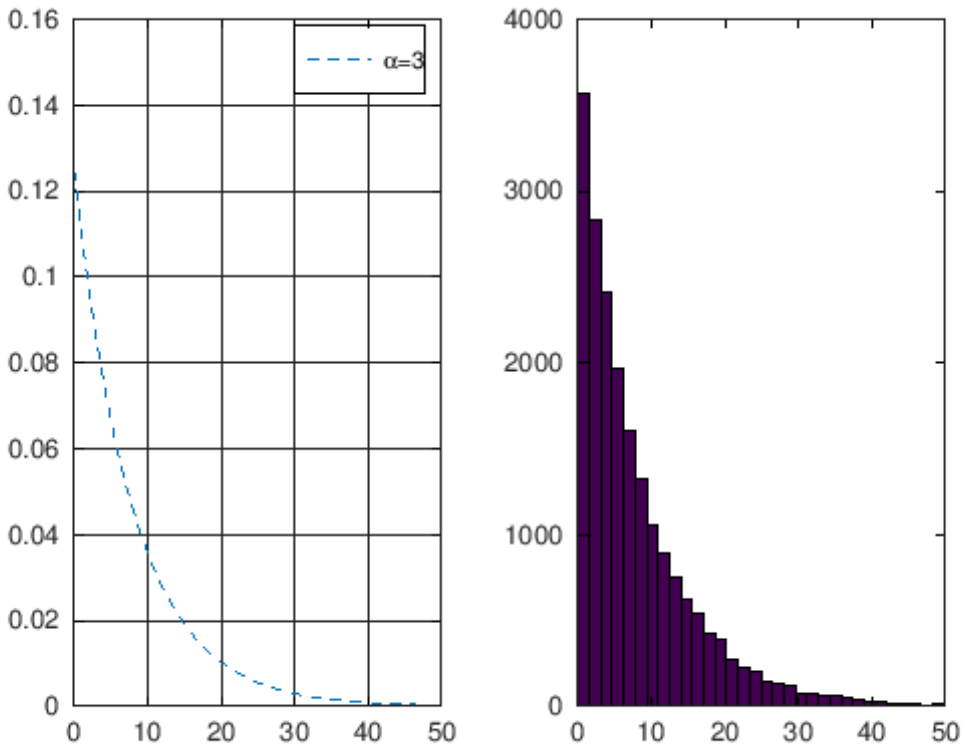
[14]:
```
u = normrnd(3,1,1000,1);  % help normrnd/exprnd
```

[15]:
```
hist(u,[0:0.1:6])
```



[ ]:

[16]:
```
alpha = 8;  Nmaxplot = 50; t = (0:0.2:Nmaxplot)';
subplot(1,2,1);  plot(t, exppdf(t,alpha), '--');   legend('\alpha=3');  grid on;
subplot(1,2,2);  hist(exprnd(alpha,20000,1), 50); axis([0,Nmaxplot])
```



[ ]:

1.5 5.

[8]:
```
b = [4; 9; 2] # Column vector
A = [ 3 4 5;
      1 3 1;
      3 5 9 ]
x = A \ b     # Solve the system Ax = b
```

```
b =
```

```
   4
   9
   2

A =

   3   4   5
   1   3   1
   3   5   9

x =

  -1.5000
   4.0000
  -1.5000
```

[ ]:

demo_rand_python

September 20, 2023

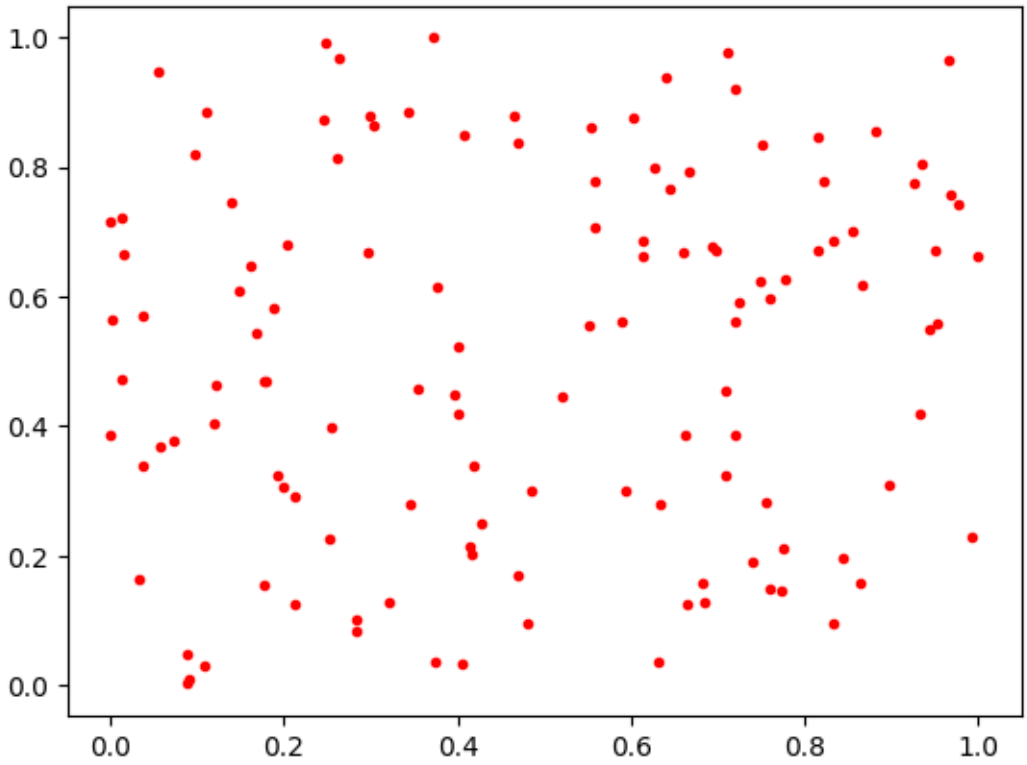
[1]:
```
import pylab
```

[2]:
```
import math
```

[3]:
```
import random
```

[4]:
```
x = []
y = []
```

[5]:
```
xval = 0.0
while xval < math.pi*4:
    x.append(random.random())   ## xval
    y.append(random.random())   ## (math.sin(xval))
    xval += 0.1
pylab.plot(x,y,'r.')
pylab.show()
```



[6]:
```
print(len(x))
```

126

[ ]:
```

```