L12. More on Functions

Header, Specification, Body
Input Parameter List
Output Parameter List
Built-Ins: randn, imag,
real, max, min, ginput

Eg. 1: “Gap N”
Keep tossing a fair coin until

| Heads - Tails | == N

Score = total # tosses
Write a function Gap(N) that returns the score and estimate the average value.

The Packaging...

function nTosses = Gap(N)

Heads = 0; Tails = 0; nTosses = 0;
while abs(Heads-Tails) < N
    nTosses = nTosses + 1;
    if rand <.5
        Heads = Heads + 1;
    else
        Tails = Tails + 1;
    end
end

The Header...

function nTosses = Gap(N)

output parameter list

input parameter list

The Body

Heads = 0; Tails = 0; nTosses = 0;
while abs(Heads-Tails) < N
    nTosses = nTosses + 1;
    if rand <.5
        Heads = Heads + 1;
    else
        Tails = Tails + 1;
    end
end

The necessary output value is computed.

Local Variables

Heads = 0; Tails = 0; nTosses = 0;
while abs(Heads-Tails) < N
    nTosses = nTosses + 1;
    if rand <.5
        Heads = Heads + 1;
    else
        Tails = Tails + 1;
    end
end
A Helpful Style

Heads = 0; Tails = 0; n = 0;
while abs(Heads-Tails) < N
    n = n + 1;
    if rand < .5
        Heads = Heads + 1;
    else
        Tails = Tails + 1;
    end
end
nTosses = n;

Explicitly assign output value at the end.

The Specification...

function nTosses = Gap(N)

% Simulates a game where you
% keep tossing a fair coin
% until |Heads - Tails| == N.
% N is a positive integer and
% nTosses is the number of
% tosses needed.

Estimate Expected Value of Gap(N)

Strategy:

Play "Gap N" a large number of times.

Compute the average "score."

That estimates the expected value.

Solution...

N = input('Enter N:');
nGames = 10000;
s = 0;
for k=1:nGames
    s = s + Gap(N);
end
ave = s/nGames;

A very common methodology for the estimation of expected value.

Sample Outputs

<table>
<thead>
<tr>
<th>N</th>
<th>Expected Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>98.67</td>
</tr>
<tr>
<td>20</td>
<td>395.64</td>
</tr>
<tr>
<td>30</td>
<td>889.11</td>
</tr>
</tbody>
</table>

Solution...

N = input('Enter N:');
nGames = 10000;
s = 0;
for k=1:nGames
    s = s + Gap(N);
end
ave = s/nGames;

Program development is made easier by having a function that handles a single game.
What if the Game Was Not "Packaged"?

\[
s = 0;
\text{for } k=1:n\text{Games}
\begin{align*}
\text{score} &= \text{Gap}(N) \\
    s &= s + \text{score};
\end{align*}
\text{end}
\text{ave} = s/n\text{Games};
\]

Is there a Pattern?

- \(N = 10\) Expected Value = 98.67
- \(N = 20\) Expected Value = 395.64
- \(N = 30\) Expected Value = 889.11

New Problem

Estimate expected value of Gap(N) for a range of N-values, say, N = 1:30

Pseudocode

\[
\text{for } N=1:30
\]
\begin{align*}
\text{Estimate expected value of Gap}(N) \\
\text{Display the estimate.}
\end{align*}
\text{end}

Pseudocode

\[
\text{for } N=1:30
\]
\begin{align*}
\text{Estimate expected value of Gap}(N) \\
\text{Display the estimate.}
\end{align*}
\text{end}

Refine this!
Done that..

\begin{verbatim}
nGames = 10000;
s = 0;
for k=1:nGames
    s = s + Gap(N);
end
ave = s/nGames;
\end{verbatim}

Sol’n Involves a Nested Loop

\begin{verbatim}
for N = 1:30
    s = 0;
    for k=1:nGames
        s = s + Gap(N);
    end
    ave = s/nGames;
    fprintf('%3d   %16.3f
',N,ave))
end
\end{verbatim}

But during derivation, we never had to reason about more than one loop.

Sol’n Involves a Nested Loop

\begin{verbatim}
for N = 1:30
    s = 0;
    for k=1:nGames
        s = s + Gap(N);
    end
    ave = s/nGames;
    fprintf('%3d   %16.3f
',N,ave))
end
\end{verbatim}

Output

<table>
<thead>
<tr>
<th>N</th>
<th>Expected Value of Gap(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>4.009</td>
</tr>
<tr>
<td>3</td>
<td>8.985</td>
</tr>
<tr>
<td>4</td>
<td>16.094</td>
</tr>
<tr>
<td>28</td>
<td>775.710</td>
</tr>
<tr>
<td>29</td>
<td>838.537</td>
</tr>
<tr>
<td>30</td>
<td>885.672</td>
</tr>
</tbody>
</table>

Looks like $N^2$.

Maybe increase nTrials to solidify conjecture.

Eg. 2: Random Quadratics

Generate random quadratic
\[ q(x) = ax^2 + bx + c \]

If it has real roots, then plot $q(x)$ and highlight the roots.

Sample Output
% Uniform
for k=1:1000
    x = rand;
end

% Normal
for k=1:1000
    x = randn;
end

% Built-In Function: randn

x = 3 + 4*sqrt(-1);
y = real(x);  % Assigns 3 to y.
z = imag(x);  % Assigns 4 to z.

% Built-In Functions: imag and real

a = 3, b = 4;
y = min(a,b);    % Assigns 3 to y.
z = max(a,b);    % Assigns 4 to z.

% Built-In Functions: min and max

function [a,b,c] = randomQuadratic
    % a, b, and c are random numbers, normally distributed.
    a = randn;
b = randn;
c = randn;
end

% Packaging the Coefficient Computation

function [r1,r2] = rootsQuadratic(a,b,c)
    % a, b, and c are real.
    % r1 and r2 are roots of ax^2 + bx + c = 0.
    r1 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
r2 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
end

% Computing the Roots

function [a,b,c] = randomQuadratic
    % a, b, and c are real.
    % A function can have more than one output parameter.
    % Syntax: [v1,v2,...]
end

function [v1,v2,...] = randomQuadratic
    % A function can have no input parameters.
    Syntax: Nothing
Question Time

function [r1,r2] = rootsQuadratic(a,b,c)
r1 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
r2 = (-b + sqrt(b^2 - 4*a*c))/(2*a);

a = 4; b = 0; c = -1;
[r2,r1] = rootsQuadratic(c,b,a);
r1 = r1

Output?
A. 2  B. -2  C. .5  D. -.5

Answer is B.

We are asking rootsQuadratic to solve
\(-x^2 + 4 = 0\)  \(\text{roots} = +2 \text{ and } -2\)

Since the function call is equivalent to
\([r2,r1] = \text{rootsQuadratic}(-1,0,4);\)

Script variable \(r1\) is assigned the value that
rootsQuadratic returns through output parameter \(r2\). That value is \(-2\)

Script Pseudocode

for k = 1:10
    Generate a random quadratic
    Compute its roots
    If the roots are real
        then plot the quadratic and
        show roots
end

if imag(r1)==0 && imag(r2)==0
Script Pseudocode

for k = 1:10
    [a,b,c] = randomQuadratic;
    [r1,r2] = rootsQuadratic(a,b,c);
    if imag(r1)==0 && imag(r2)==0
        then plot the quadratic and show roots
    end
end

Plot the Quadratic and Show the Roots

m = min(r1,r2);
M = max(r1,r2);
x = linspace(m-1,M+1,100);
y = a*x.^2 + b*x + c;
plot(x,y,0*y,:k',r1,0,'or',r2,0,'or')

This determines a nice range of x-values.

Array ops get the y-values.

Graphs the quadratic.

A black, dashed line x-axis.
Plot the Quadratic and Show the Roots

\[
m = \min(r_1, r_2);
M = \max(r_1, r_2);
x = \text{linspace}(m-1, M+1, 100);
y = a \cdot x^2 + b \cdot x + c;
\text{plot}(x, y, x, 0*y, ':k', r_1, 0, 'or', r_2, 0, 'or')
\]

Highlight the root \(r_1\) with red circle.

Plot the Quadratic and Show the Roots

\[
m = \min(r_1, r_2);
M = \max(r_1, r_2);
x = \text{linspace}(m-1, M+1, 100);
y = a \cdot x^2 + b \cdot x + c;
\text{plot}(x, y, x, 0*y, ':k', r_1, 0, 'or', r_2, 0, 'or')
\]

Highlight the root \(r_2\) with red circle.

Complete Solution

\begin{verbatim}
for k=1:10
    [a,b,c] = randomQuadratic;
    [r1,r2] = rootsQuadratic(a,b,c);
    if imag(r1)==0
        m = \min(r1, r2); M = \max(r1, r2);
        x = \text{linspace}(m-1, M+1, 100);
        y = a \cdot x^2 + b \cdot x + c;
        \text{plot}(x, y, x, 0*y, ':k', r_1, 0, 'or', r_2, 0, 'or')
        shg
        pause(1)
    end
end
\end{verbatim}