PSY8219 : Week 5

Homework 4 Due Today

Homework 5 Due October 1

Readings for Today
   Attaway Chapter 3 and 6

Readings for Next Week
   Attaway Chapter 6, 10, and 12
Introduction to Random Numbers
rng(1);
r1 = rand(100,1);
rng(1);
r2 = rand(100,1);
s = sum(r1 == r2));
s
what does s equal?
Introduction to Random Numbers

`rand()`

what does `rand()` do?

returns a random number 0...1 from a uniform distribution
Introduction to Random Numbers

probability density function  random number generator
Introduction to Random Numbers

probability density function

random number generator

rand()

.42345234  .51832049  .10999311

how would you show that these are distributed as a uniform distribution?
Introduction to Random Numbers

probability density function

random number generator

rand()

.42345234 .51832049 .10999311

how would you show that these are distributed as a uniform distribution?
Introduction to Random Numbers

size = 10000;

mydata = rand(size,1);
histogram(mydata, 10);
Introduction to Random Numbers

probability mass function

random number generator

how would you show that these are distributed as a discrete uniform distribution?

randi()
Introduction to Random Numbers

size = 10000;

mydata = randi(8,size,1);
histogram(mydata, 10);
Difference between a probability density function and a probability mass function?

**Probability density function**

```
0 1
```

**Probability mass function**

```
1.125
1 2 3 4 5 6 7 8
```
Difference between a probability density function and a probability mass function?

**Probability Density Function**
- Continuous
- Area under the curve represents probability

**Probability Mass Function**
- Discrete
- Height of bars represents probability
Difference between a probability density function and a probability mass function?

- **Probability density function**:
  - Total area under the pdf is 1

- **Probability mass function**:
  - Sum of all probabilities is 1
Difference between a probability density function and a probability mass function?
Difference between a probability density function and a probability mass function?

Probability density function

Probability mass function

Likelihood of observing \(0.4804432321\) is 1

Probability of observing 5 is 0.125
Difference between a probability density function and a probability mass function?

**Probability density function**

- Probability density function is continuous.
- Area under the curve gives the probability of observing a value.
- Probability of observing 0.4804432321 is 0.

**Probability mass function**

- Probability mass function is discrete.
- Probability of observing 5 is 0.125.
- Probability of observing 6 is 0.125.
- Probability of observing 7 is 0.125.
Difference between a probability density function and a probability mass function?

**Probability Density Function**

- Area under the curve from 0.4 to 0.5 is 0.1.
- Probability of observing a number between 0.4 and 0.5 is equal to 0.1.

**Probability Mass Function**

- Probability of observing 5 is 0.125.
size = 10000;

mydata = randi([4 12],size,1);
histogram(mydata, 10);
Introduction to Random Numbers

Probability density function

Random number generator

randn()

0.53766714  1.8338850  -2.2588468

How would you show that these are distributed as a Gaussian with mean 0 and standard deviation 1?

plot(-4:.01:4, normpdf(-4:.01:4, 0, 1))
Introduction to Random Numbers

probability density function

random number generator

plot(-4:.01:4, normpdf(-4:.01:4, 0, 1))

histogram(randn(10000,1),20)
Introduction to Random Numbers

probability density function

.875

likelihood of observing
.875 is 0.27205
Introduction to Random Numbers

probability density function

probability of observing .875 is 0
Introduction to Random Numbers

probability density function

area is .341344

probability of observing number between 0 and 1 is equal to .341344
Introduction to Random Numbers

probability density function

area under the curve between -1.96 and +1.96
Introduction to Random Numbers

probability density function

what is the area under here?

what is the area under here?
Introduction to Random Numbers

probability density function

area under the curve between -1.96 and +1.96

probability of observing number between -1.96 and +1.96 is equal to .95
Introduction to Random Numbers

`rand()` is the building block for every other random number generator

"every other" in what sense?
Introduction to Random Numbers

`rand()` is the building block for every other random number generator. "every other" in what sense?

there are mathematical and computational techniques for using `rand()` to generate random samples from any pdf, e.g.,
- transformation methods
- rejection sampling
- Markov Chain Monte Carlo (MCMC)
How can we create a random sequence of N coin flips, where the probability of a head is .5 - using just rand()
How can we create a random sequence of $N$ coin flips, where the probability of a head is $p$ - using just `rand()`
Imagine an experiment where on each trial you show condition 1 with probability .45, condition 2 with probability .45, and condition 3 with probability .10.

**Purely Randomized**

condition 1

\[
\begin{array}{c}
T \\
+ \\
\Gamma
\end{array}
\]

condition 2

\[
\begin{array}{c}
T \\
+ \\
T
\end{array}
\]

condition 3

\[
\begin{array}{c}
T
\end{array}
\]
Imagine an experiment where on each trial you show show condition 1 with probability .45, condition 2 with probability .45, and condition 3 with probability .10.

*Force Exact Numbers of Trials in Each Condition*
Imagine an experiment where on each trial you show condition 1 with probability .45, condition 2 with probability .45, and condition 3 with probability .10.

**Force Exact Numbers of Trials in Each Condition**

* e.g., 100 total trials

<table>
<thead>
<tr>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>+</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>45 trials</td>
<td>45 trials</td>
<td>10 trials</td>
</tr>
</tbody>
</table>
First, how do we create this array (efficiently)?
First, how do we create this array (efficiently)?

Second, how do randomize it?
First, how do we create this array (efficiently)?

Second, how do randomize it?

`randperm()`

`randperm` Random permutation.

\[ P = \text{randperm}(N) \] returns a vector containing a random permutation of the integers 1:N. For example, `randperm(6)` might be [2 4 5 6 1 3].
First, how do we create this array (efficiently)?

Second, how do randomize it?

`randperm()`

How can this be used to randomize the array?
Functions
## Functions

We’ve already used lots of built-in functions ...

<table>
<thead>
<tr>
<th>sin()</th>
<th>ceil()</th>
<th>size()</th>
</tr>
</thead>
<tbody>
<tr>
<td>log()</td>
<td>floor()</td>
<td>length()</td>
</tr>
<tr>
<td>abs()</td>
<td>round()</td>
<td>rand()</td>
</tr>
</tbody>
</table>

Many of these are written themselves in Matlab. Some are ultimately written in C++ or Fortran or assembly and compiled.
Functions

Why do we use functions?
Functions

Why do we use functions?
- reuse code over and over again
- makes code far more readable
- easier to write modular code
- easier to debug modular code
Functions in Matlab

calling a function

\[
\begin{align*}
\gg a &= \text{myfun}(x, y, z) \\
\end{align*}
\]
Functions in Matlab

calling a function

>> a = myfun(x, y, z)

function name

parameters
Functions in Matlab

calling a function

\[ a = \text{myfun}(x, y, z) \]
Functions in Matlab

calling a function

$$a = \text{myfun}(x, y, z)$$

declaring a function in file myfun.m

```matlab
function retval = myfun(p1, p2, p3)
    % comment describing the function
    retval = (p1 + p2) / p3
end
```
Functions in Matlab

calling a function

```
>> a = myfun(x, y, z)
```

```
function retval = myfun(p1, p2, p3)

% comment describing the function
    retval = (p1 + p2) / p3

end
```

parameters are mapped 1:1

function names must match
Functions in Matlab

function c = myfun1(a, b)
    a = a/10;
    b = b/10;
    c = a + b;
end

clear all;
x = 100;
y = 200;
z = myfun1(x, y)
DEBUGGING in Matlab

function c = myfun1(a, b)
    a = a/10;
    b = b/10;
    c = a + b;
end

clear all;
x = 100;
y = 200;
z = myfun1(x, y)
DEBUGGING in Matlab

- setting and using breakpoints
- command window to evaluation expressions (K>>)
- workspace to see variables
- "Step" to step and don’t step into a function
- "Step in" to step into a function
- "Step out" to continue until that function is over
- "Continue" to continue until next breakpoint if there is one
- "Exit debug mode" does as it says
- "Stack" lets you pop around workspaces
Variable Scope in Matlab

function c = myfun1(a, b)
    a = a/10;
    b = b/10;
    c = a + b + q;
end

clear all;
q = 999;
x = 100;
y = 200;
z = myfun1(x, y)
Variable Scope in Matlab

function c = myfun1(a, b)
    clear all;
    a = a/10;
    b = b/10;
    c = a + b;
end

clear all;
x = 100;
y = 200;
z = myfun1(x, y)
Matlab passes parameters “by value”

function c = myfun1(a, b)
    a = a/10;
    b = b/10;
    c = a + b;
end

clear all;
x = 100;
y = 200;
z = myfun1(x, y)
x
y
How about this?

function \( z = \text{myfun1}(x, y) \)
   \[
   x = x/10;
   y = y/10;
   z = x + y;
   \]
end

----------------------------------------

clear all;
x = 100;
y = 200;
z = myfun1(x, y)
x y
Matlab passes parameters “by value”

```matlab
function z = myfun1a(x)
    x = x + 1000;
    z = sum(x);
end
```

clear all;
x = [1 20 5 100];
z = myfun1a(x)
x
Functions can return no value

A “Procedure”

function print_data(data)
% this functions prints formatted data
for i=1:length(data)
    fprintf(‘%3d\t %f\n’, i, data(i));
end
Functions can take no parameters too

This can be used to break up a program into pieces.

```
function print_info()
% this functions prints a message
for i=1:length(data)
    fprintf('This is an important message.\n');
    fprintf('Make sure you read it.\n');
    fprintf('Or else.\n');
end
```
Functions can return multiple values/variables

function [davg, dstd, dmin, dmax] = ...
    myfun1b(data)
% this functions analyzes data
    davg = mean(data);
    dstd = std(data);
    dmin = min(data);
    dmax = max(data);
end
Functions can return multiple values/variables

data = [1 3 4; 3 2 1];
Sz = size(data);
[nr nc] = size(data);
function [davg dstd dmin dmax] = myfun(data)
% this functions analyzes data
if (length(data) < 2)
    return;
end

davg = mean(data);
dstd = std(data);
dmin = min(data);
dmax = max(data);
end
function [davg dstd dmin dmax] = myfun(data)
% this functions analyzes data

davg = []; dstd = []; dmin = []; dmax = [];
if (length(data) < 2)
    return;
end

davg = mean(data);
dstd = std(data);
dmin = min(data);
dmax = max(data);
end
You can break out of for and while loops as well:

```plaintext
s = 0;
for i = 1:10
    s = s+1;
    if i == 4
        break;
    end
end
s
```
break in general

You can break out of for and while loops as well:

```plaintext
while true
    s = sum(rand(4,1));
    if s > 3
        break;
    end
end
s
```
Global variables

Global variables can be visible from any function anywhere in a program – nearly all of the time, using them is considered poor programming style.

global myglobvar;   % declares global
myglobvar = 1000;
a = myfun(1);
----------------------
function retval = myfun(x)
    global myglobvar;
    myglobvar = myglobvar*2;
    retval = x+myglobvar;
end
Persistent variables stick around within a function. Most of the time, you do not need them.

```matlab
function retval = myfun(x)
    persistent cnt;
    if isempty(cnt)
        cnt = 0;
    end
    cnt = cnt + 1;
    retval = cnt;
end
```
Variable number of parameters to a function

Sometimes, you may want “optional” parameters.

```matlab
function x = myanalyze(data, varargin)

    nargin
    special internal name in Matlab
    number of
    additional parameters
    it will be a cell array containing
    any and all parameters passed
    after those that are explicit –
    this cell array must be interpreted
```
Passing functions TO functions

Sometimes you may want to pass a function to a function.

e.g., you create a general purpose function for plotting a graph with lots of formatting commands built into the function (some of which we’ll talk about soon) – you could pass the function you want to plot to the function that creates the graph.

e.g., optimization algorithms pass the function to be optimized to the optimization function
Passing functions TO functions

\[ x = \text{myfun}(); \quad \text{x is assigned the value returned by myfun()} \]

vs.

\[ x = \text{@myfun}; \quad \text{x is assigned the handle for myfun()} \]
Passing functions TO functions

e.g.,

\[ z = \text{fminsearch}(\text{myfun}, \, z0); \]

vs.

\[ z = \text{fminsearch}(@\text{myfun}, \, z0); \]
Example

data = [1 1.2 ; 2 1.7 ; 3 2.3 ; 4 2.5 ; 5 3.1 ;
       6 3.2 ; 7 4.0 ];

How would you fit a linear regression line to this data?
Example: Linear Regression (review)

data = [1 1.2 ; 2 1.7 ; 3 2.3 ; 4 2.5 ; 5 3.1 ;
       6 3.2 ; 7 4.0 ];

Xobs = [repmat(1,length(data),1) data(:,1)];
yobs = data(:,2);

BLR = inv(Xobs'*Xobs) * Xobs' * yobs;

xobs = min(data(:,1))-1:.1:max(data(:,1))+1;
yprd = BLR(2)*xobs + BLR(1);

figure;
hold on;
plot(data(:,1),data(:,2),'r+');
plot(xobs,yprd,'b.');
Example: Optimization

data = [1 1.2 ; 2 1.7 ; 3 2.3 ; 4 2.5 ; 5 3.1 ; 6 3.2 ; 7 4.0 ];

How you might fit using an optimization approach?
Example: Optimization

data = [1 1.2 ; 2 1.7 ; 3 2.3 ; 4 2.5 ; 5 3.1 ;
       6 3.2 ; 7 4.0 ];

m0 = 1;       b0 = 1;
[best,fval] = fminsearch(@myLR, [m0 b0]);
m = best(1);   b = best(2);

x = min(data(:,1))-1:max(data(:,1))+1;
y = m*x + b;

figure;
hold on;
plot(data(:,1),data(:,2),'r+');
plot(x,y,'g-');
Local Functions

mystats.m

function [avg, med] = mystats(x)
% mystats  The only function visible from the outside
    n = length(x);
    avg = mymean(x,n);
    med = mymedian(x,n);
end

function a = mymean(v,n)
% mymean  Example of a local function
    a = sum(v)/n;
end

function m = mymedian(v,n)
% mymedian  Another example of a local function
    w = sort(v);
    if rem(n,2) == 1
        m = w((n + 1)/2);
    else
        m = (w(n/2) + w(n/2 + 1))/2;
    end
end
Local Functions

mystats.m

function [avg, med] = mystats(x)
% mystats   The only function visible from the outside
    n = length(x);
    avg = mymean(x,n);
    med = mymedian(x,n);
end

function a = mymean(v,n)
% mymean    Example of a local function
    a = sum(v)/n;
end

function m = mymedian(v,n)
% mymedian    Another example of a local function
    w = sort(v);
    if rem(n,2) == 1
        m = w((n + 1)/2);
    else
        m = (w(n/2) + w(n/2 + 1))/2;
    end
end

visible to outside files
only visible within mystats.m
only visible within mystats.m