

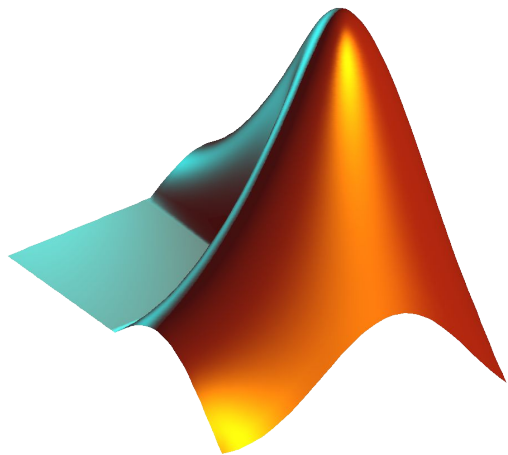
CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website:

<https://www.cs.cornell.edu/courses/cs1112/2022fa/>

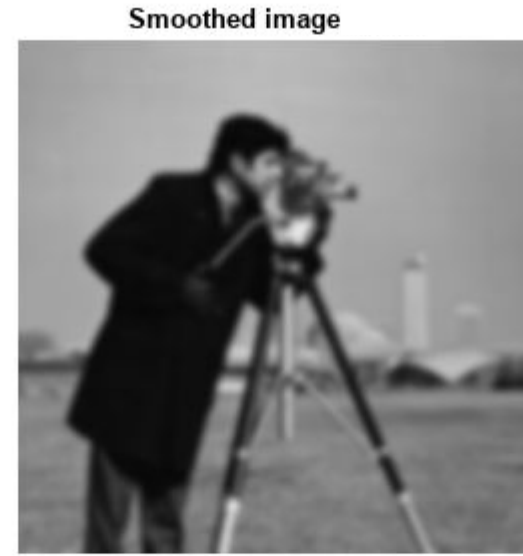
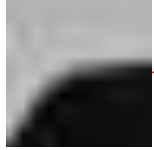
Today: images and char arrays



Agenda and announcements

- Last time
 - Image processing
 - Imread and imwrite
 - Vectorized computation
- Today
 - Finish images
 - Char arrays
- Announcements
 - Project 4 released (due 10/26)
 - 3 problems, and an optional 4th problem if you want more practice
 - Partner service out! Submit now if you need a partner!
 - Prelim 1 grades out Monday
 - Numeric arrays, char arrays, and cell arrays are where people get lost

Blurring an image



We blur an image by taking the average pixel value around each pixel

56	75	39	96	94
32	65	46	74	34
25	109	158	245	237
224	235	224	226	234
254	255	251	242	247

```
img = imread('ManTakingPhoto.png')  
% assuming we are finding the average pixel value around  
% the pixel in the row 252 and column 283  
block = img(250:254, 281:285);
```

```
img = imread('ManTakingPhoto.png');
imgBlur = zeros(size(img));
imgBlur = uint8(imgBlur);
% assuming we are finding the average pixel value around
% the pixel in the row 252 and column 283
block = img(250:254, 281:285);
```

```
avgPixel = 0;
for i = 1:size(block,1)
    for j = 1:size(block,2)
        avgPixel = avgPixel + block(i,j);
    end
end
```

```
imgBlur(252,283) = avgPixel/25;
```

Block →



Matrix storing pixel values of block ↓

56	75	39	96	94
32	65	46	74	34
25	109	158	245	237
224	235	224	226	234
254	255	251	242	247

Bad code because it uses hard coded values 252 and 283. This is bad when we change the center pixel.


```
img = imread('ManTakingPhoto.png');
imgBlur = zeros(size(img));
imgBlur = uint8(imgBlur);
% assuming we are finding the average pixel value around
% the pixel in the row 252 and column 283
row = 252; col = 283;
block = img(row-2:row+2, col-2:col+2);


avgPixel = 0;
for i = 1:size(block,1)
    for j = 1:size(block,2)
        avgPixel = avgPixel + block(i,j);
    end
end

numPixels = size(block,1)*size(block,2);
imgBlur(row,col) = avgPixel/numPixels ;
```

Update: This code does not deal with uint8 correctly. See next slide...

Block 

Matrix storing pixel values of block 



56	75	39	96	94
32	65	46	74	34
25	109	158	245	237
224	235	224	226	234
254	255	251	242	247

But now this code will not fully work. Why?

We need to do something special for edge pixels... otherwise we would go out of bounds of the image.

```
img = imread('ManTakingPhoto.png');
imgBlur = zeros(size(img));
imgBlur = uint8(imgBlur);
% assuming we are finding the average pixel value around
% the pixel in the row 252 and column 283
row = 252; col = 283;
block = img(row-2:row+2, col-2:col+2);

avgPixel = 0;
for i = 1:size(block,1)
    for j = 1:size(block,2)
        avgPixel = avgPixel + block(i,j)/numPixels;
    end
end

numPixels = size(block,1)*size(block,2);
imgBlur(row,col) = avgPixel;
```

Block →



Matrix storing pixel values of block ↙

56	75	39	96	94
32	65	46	74	34
25	109	158	245	237
224	235	224	226	234
254	255	251	242	247

The code on the last slide does not deal with overflow correctly. In particular, avgPixel would become uint8 and would not be able to store anything larger than 255.

New topic: text in programming

- We've seen text already
 - `fprintf('Hello \n'), title('golf ball trajectory'), imread('ManTakingPhoto.png'), etc.`
 - Time to dive in to the details and *compute* on text
- Vocabulary you should know
 - A single letter (or digit, or symbol, or space) is a **character**
 - A sequence of characters is an array of character, but I will call it a **char array**
 - In other programming languages this is called a string
 - There are strings in MATLAB but we will not be going over them.
 - A char array could be a word, a sentence, gibberish, ...

Text–sequences of characters–are important in computation

Numerical data is often encoded in strings. For example, a file containing weather data in Ithaca might begin with the char array

```
'W7629N4226'
```

Meaning

```
Longitude: 76° 29' West
```

```
Latitude: 42° 26' North
```

We may need to grab hold of the substring `W07629`, convert `076` and `29` to the numeric values 76 and 29, and do some computation.

Character array (an array of type char)

- We have used arrays of characters in programs already:
 - `n= input('Next number: ')`
 - `sprintf('Answer is %d', ans)`
- `'CS1112 rocks!'` is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.
- Can have 2-d array of characters as well (below would be a 2x6 array):

```
charArr = ['CS1112';  
           'rocks!'];
```

charArr =

C	S	1	1	1	2
r	o	c	k	s	!

Numeric vectors versus char vectors

Numeric vectors

- Assignment

```
v = [7, 0 ,5];
```

- Indexing

```
x = v(3);
```

```
v(1) = 1;
```

```
w = v(2:3);
```

- Appending

```
v = [7 0 5];
```

```
v(4) = 2;
```

- Concatenation

```
v = [7 0 5];
```

```
v = [v 2];    % v = [7 0 5 2]
```

Char vectors

- Assignment

```
s = 'hi';      % shortcut
```

```
s = ['h','i']; % formal
```

- Indexing

```
c = s(2)
```

```
s(1) = 'j';    % s = 'ji'
```

```
t = s(1:2);    % t = 'ji'
```

- Appending

```
s = 'tree';
```

```
s(5) = 's';    % s = 'trees'
```

- Concatenation

```
s = [s ' quack'];
```

```
% s = 'trees quack'
```

Syntax: single quotes enclose char arrays in MATLAB

Anything enclosed in single quotes is a char array (even if it looks like something else like a number).

- `x = '100';` is a char array of length 3
- `x = 100;` is a numeric value
- `p = 'pi';` is a char array of length 2
- `p = pi;` is the built-in constant 3.1415...
- `v = 'x';` is a character (char vector of length 1)
- `v = x;` may be a variable name in your program

Variable types so far: char, double, uint8, logical

```
a = 'CS1';  
a = ['C', 'S', '1'];
```

```
b = [3, 9];
```

```
c = uint8(b)
```

```
d = rand() > 0.5
```

a is a 1-d array with type char components. Often called a string; NOT the same as a type in Matlab called string.

b is a 1-d array with type double components. Double is the default type for numbers in Matlab. We call b a “numeric array”.

c is a 1-d array with type uint8 components. We call c a “uint8 array”.

d is a scalar of the type logical. We call d a “Boolean value” or “logical value”.

Example: working with char arrays

A gene is a DNA fragment that codes for a protein:

ATC GCT TTG CAC ATT CTA ...

3 letter "codons" identify the amino acid sequence that defines a protein

Isoleucine (Ile)

Alanine (Ala)

Leucine (Leu)

Histidine(His)

Isoleucine(Ile)

Leucine (Leu)

Codon dictionary

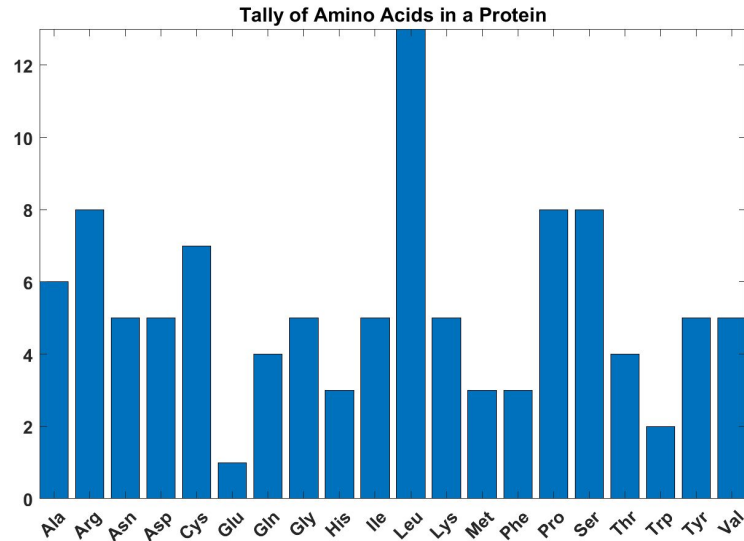
Index	Amino Acid	Mnemonic	DNA Codons
1	Alanine	Ala	GCT GCC GCA GCG
2	Arginine	Arg	CGT CGC CGA CGG AGA AGG
3	Asparagine	Asn	AAT AAC
4	Aspartic Acid	Asp	GAT GAC
5	Cysteine	Cys	TGT TGC
6	Glutamic Acid	Glu	CAA CAG
7	Glutamine	Gln	GAA GAG
8	Glycine	Gly	GGT GGC GGA GGG
9	Histidine	His	CAT CAC
10	Isoleucine	Ile	ATT ATC ATA
11	Leucine	Leu	CTT CTC CTA CTG TTA TTG
12	Lysine	Lys	AAA AAG
13	Methionine	Met	ATG
14	Phenylalanine	Phe	TTT TTC
15	Proline	Pro	CCT CCC CCA CCG
16	Serine	Ser	TCT TCC TCA TCG AGT AGC
17	Threonine	Thr	ACT ACC ACA ACG
18	Tryptophan	Trp	TGG
19	Tyrosine	Tyr	TAT TAC
20	Valine	Val	GTT GTC GTA GTG

Visualize the distribution of amino acids in a protein

- Given a gene sequence defining a protein

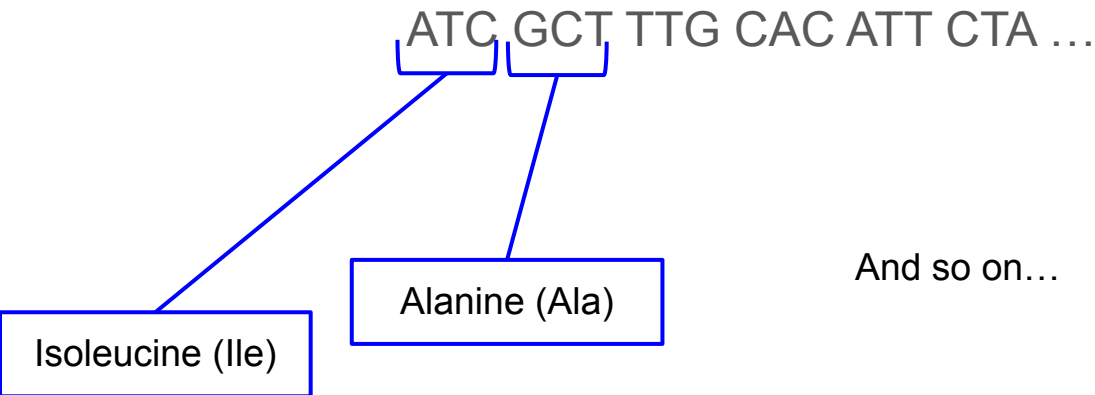
TTCGGGAGCCTGGGCGTTACG...

- Make a bar plot showing counts of amino acids that make up a protein

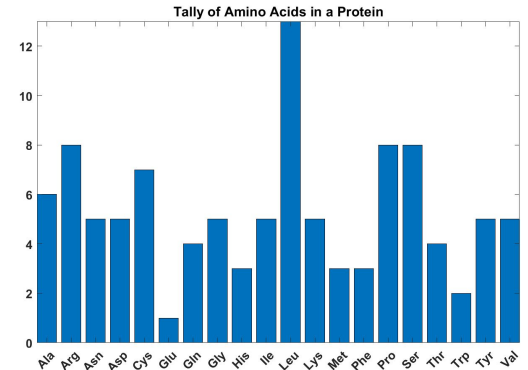


Program sketch

- Given a dna sequence representing a protein
- For each codon (subvector of 3 chars)
 - Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)
- Tally the counts of the 20 amino acids
- Draw a bar plot



And so on...



% dna sequence encoding protein

```
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...  
      'ATATGTACCAACGACAATGACATTGAAAAC' ];
```



Given a dna sequence representing a protein

- For each codon (subvector of 3 chars)
 - Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)
- Tally the counts of the 20 amino acids
- Draw a bar plot

```
% dna sequence encoding protein
```

```
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...  
      'ATATGTACCAACGACAATGACATTGAAAAC' ];
```

```
for k = 1:3:length(p)-2  
    codon = p(k:k+2); % length 3 subvector
```

```
    % search codon dictionary to find  
    % the corresponding amino acid name
```

```
end
```



Given a dna sequence representing a protein



For each codon (subvector of 3 chars)

- Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)

- Tally the counts of the 20 amino acids
- Draw a bar plot

 Treat this as an independent task to be written as a function!

```
function a = getMnemonic(s)
% s is length 3 row vector of chars
% If s is codon of an amino acid then
% a is the mnemonic of that amino acid
% Search for s in codon dictionary
```

```
cDict = ['GCT Ala'; ...
        'GCC Ala'; ...
        'GCA Ala'; ...
        'GCG Ala'; ...
        'CGT Arg'; ...
        ... ];
```

```
r = 1;
while strcmp(s, cDict(r, 1:3)) == false
    r = r + 1;
end
a = cDict(r, 5:7);
```

cDict

G	T	C		A	l	a
G	C	C		A	l	a
G	C	A		A	l	a
C	G	T		A	l	a
C	G	C		A	r	g

This should be filled in with the rest of the dictionary

Built in MATLAB function that compares two char vectors. Returns true if they are identical; otherwise false.

```
% dna sequence encoding protein
```

```
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...  
      'ATATGTACCAACGACAATGACATTGAAAAC' ];
```

```
for k = 1:3:length(p)-2  
    codon = p(k:k+2); % length 3 subvector  
    mnem = getMnemonic(codon);
```

```
end
```



Given a dna sequence representing a protein



For each codon (subvector of 3 chars)



Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)

- Tally the counts of the 20 amino acids
- Draw a bar plot

```

% dna sequence encoding protein
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...
      'ATATGTACCAACGACAATGACATTGAAAAC' ];

counts = zeros(1,20); % to store tallies

for k = 1:3:length(p)-2
    codon = p(k:k+2); % length 3 subvector
    mnem = getMnemonic(codon);

    % Tally: build histogram data

end

```

- ✓ Given a dna sequence representing a protein
- ✓ For each codon (subvector of 3 chars)
 - ✓ Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)
- Tally the counts of the 20 amino acids
- Draw a bar plot

Let's write a function that converts mnem to an index for each unique amino acid.

```
function ind = getAAIndex(aa)
```

```
% Returns index of amino acid named by char vector aa.
```

We will not write this function but you could create a char array dictionary as follows, loop through all rows, and return the index of the row containing the correct mnemonic.

A	l	a
A	r	g
A	s	n
A	s	p
C	y	s
G	l	u

⋮

```

% dna sequence encoding protein
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...
      'ATATGTACCAACGACAATGACATTGAAAAC' ];

counts = zeros(1,20); % to store tallies

for k = 1:3:length(p)-2
    codon = p(k:k+2); % length 3 subvector
    mnem = getMnemonic(codon);
    % Tally: build histogram data
    ind = getAAIndex(mnem);
    counts(ind) = counts(ind) + 1;
end

```

- ✓ Given a dna sequence representing a protein
- ✓ For each codon (subvector of 3 chars)
 - ✓ Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)
- ✓ Tally the counts of the 20 amino acids
 - Draw a histogram

```

% dna sequence encoding protein
p = [ 'TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...
      'ATATGTACCAACGACAATGACATTGAAAAC' ];

counts = zeros(1,20); % to store tallies

for k = 1:3:length(p)-2
    codon = p(k:k+2); % length 3 subvector
    mnem = getMnemonic(codon);
    % Tally: build histogram data
    ind = getAAIndex(mnem);
    counts(ind) = counts(ind) + 1;
end

bar(counts) % Draw bar chart

```

- ✓ Given a dna sequence representing a protein
- ✓ For each codon (subvector of 3 chars)
 - ✓ Use codon dictionary to determine which amino acid the codon represents (get the 3-letter mnemonic for that amino acid)
- ✓ Tally the counts of the 20 amino acids
- ✓ Draw a histogram