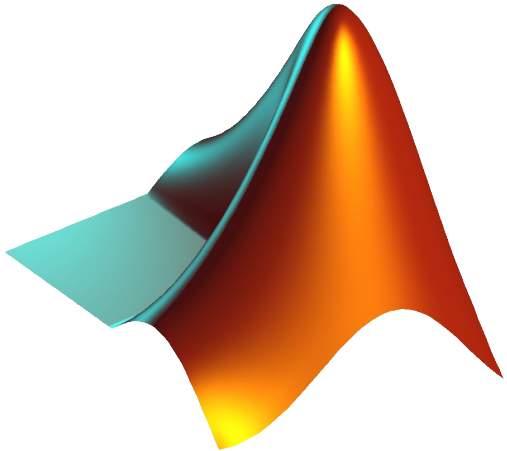
CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz



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

Today: Char arrays and cell arrays

Agenda and announcements

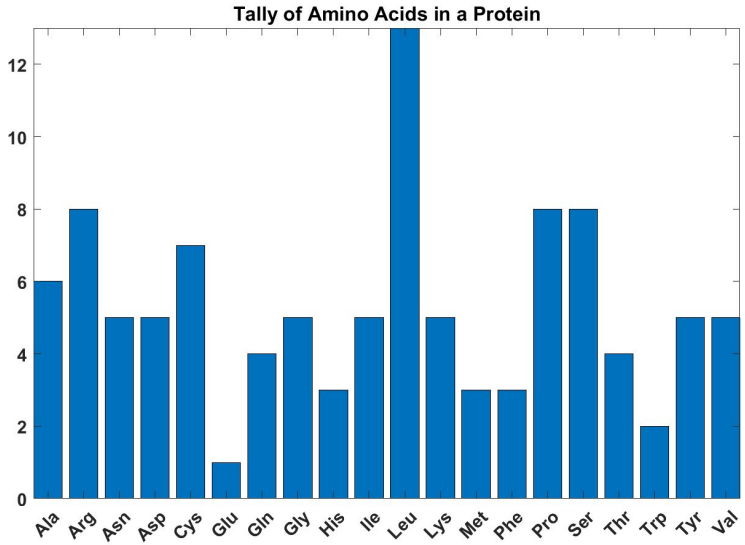
- Last time
  - Finished images
  - Char arrays
- Today
  - More char arrays
  - Cell arrays
  - Comparison of different types of arrays in MATLAB
- Announcements
  - Project 4 due 10/26
    - 3 problems, and an optional 4th problem if you want more practice
  - Prelim 1 grades out
    - Submit a regrade until Sunday!
  - Grading guides posted for P1-3. Go to CMS, then click on the solutions to find the grading guide.

Visualize the distribution of amino acids in a protein

- Given a gene sequence defining a protein

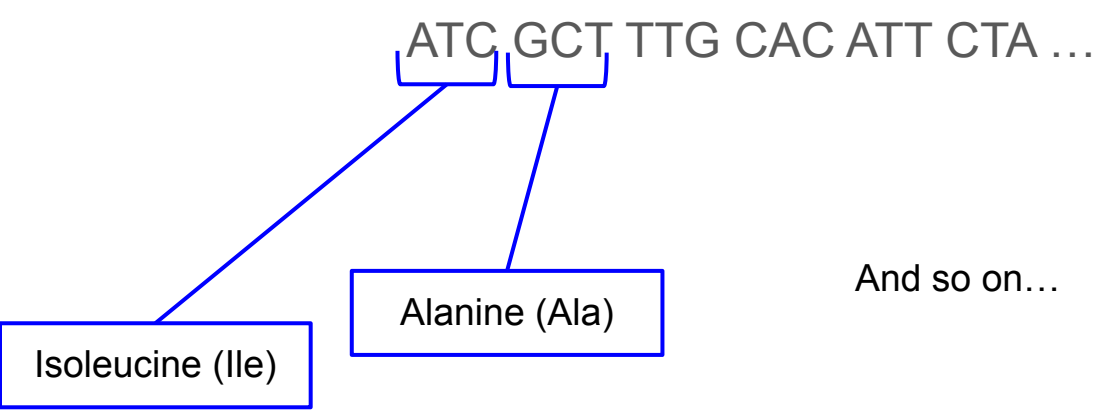
  ATC GCT TTG CAC ATT CTA …

- Make a bar plot showing counts of amino acids that make up a protein
  - 3 letter “codons” identify the amino acid

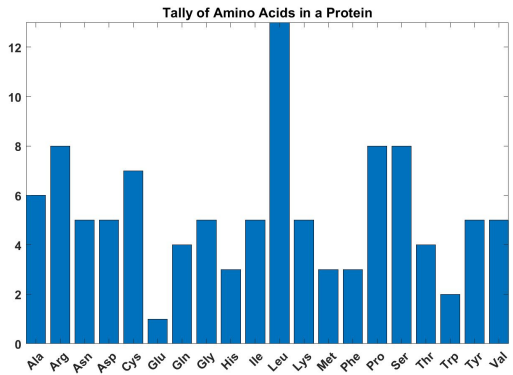


Program sketch

- 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'];



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

codon would store:

```
codon = 'TTC'
codon = 'GGG'
codon = 'AGC'
...
```

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

```
function a = getMnemonic(s)
% s is length 3 row vector of chars (i.e. 'GCC')
% 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

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

Examples:

```
s = 'GCC';
cDict(1, 1:3) = 'GTC';
strcmp(s, cDict(r, 1:3)) == false

s = 'GCC';
cDict(2, 1:3) = 'GCC';
strcmp(s, cDict(r, 1:3)) == true
```

```
% 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

How can I use mnem to increment counts?

Let's see this in the function on the next slide.

```
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

```
%%% Exercise: fill in the missing comments that describe each line
p = ['TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...  % _____________________________
'ATATGTACCAACGACAATGACATTGAAAAC'];

counts = zeros(1,20);

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

    ind = getAAIndex(mnem);                % _____________________________
    counts(ind) = counts(ind) + 1;         % _____________________________
end

bar(counts)                                % _____________________________
```

```
p = ['TTCGGGAGCCTGGGCGTTACGTTAATGAAA' ...  % store protein sequence
'ATATGTACCAACGACAATGACATTGAAAAC'];

counts = zeros(1,20);

for k = 1:3:length(p)-2                    % loop thru each codon
    codon = p(k:k+2);                      % store codon
    mnem = getMnemonic(codon);             % transform codon to name

    ind = getAAIndex(mnem);                % transform name to index
    counts(ind) = counts(ind) + 1;         % Increment count
end

bar(counts)                                % plot histogram
```

Linear search

```
function i = LinSearch(v, x)
% Searches for x in a vector v
% Returns the index of the first
% occurrence of x in v (returns
% -1 if x is not found).

k = 1;
while k <= length(v) && v(k) ~= x
    k = k + 1;
end
if k > length(v)
    i = -1;
else
    i = k;
end
```

Linear search

```
function i = LinSearch(v, x)
% Searches for x in a vector v
% Returns the index of the first occurrence of x in v (returns
% -1 if x is not found).

k = 1;
while k <= length(v) && v(k) ~= x
    k = k + 1;
end
if k > length(v)
    i = -1;          % x is not found in v
else
    i = k;
end
```

Does this work for numeric arrays?

```
i = LinSearch([1, 4, 3, 5], 2)
```

yes!

Does this work for char arrays?

```
j = LinSearch('dad', 'd')
```

yes!

Limitations of char arrays and numeric arrays

- Homogeneous data type
  - Cannot represent tables

```
a = ['Dominic', 25, true];
Will output an error!
```

- Rows of 2D arrays must have the same length

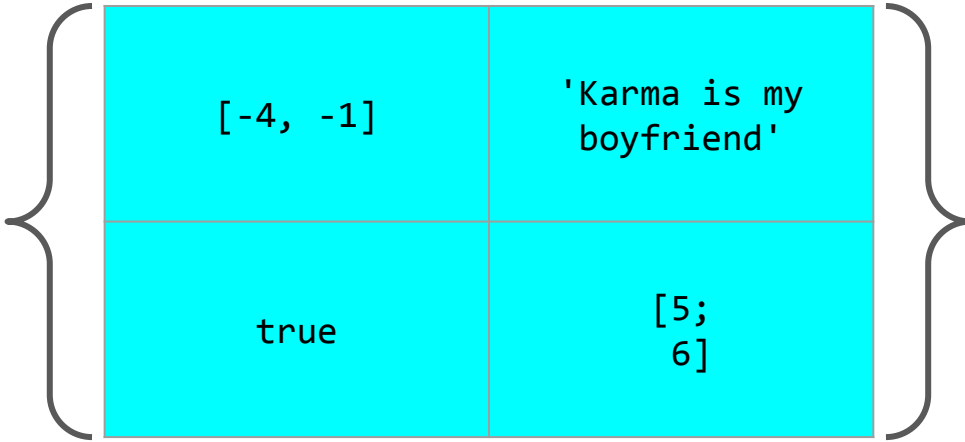
```
b = [1, 2, 3, 4;
     2, 3;
     5, 6, 7, 8];
Will output an error!
```

```
c = ['Vermont';
     'California'];
Will output an error!
```

```
d = ['Vermont    ';
     'California'];
Will work but is awkward.
```

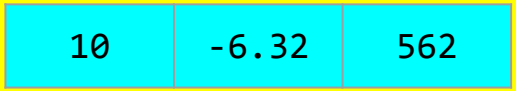
New data type: cell array

- The elements in a cell array can be of any type!
  - Array of doubles
  - Unit8
  - Char array
  - Boolean (logical) values
  - Another cell array
- Each cell in a cell array may have different type and size!
- Cell arrays are still rectangular



Different kinds of arrays in MATLAB

Numeric arrays



- Stores numbers
- Assignment
```
v = [10, -6.32, 562];
```
- Indexing
```
v(2) = -0.5;
```
- Appending
```
v(4) = 6;
```
- Concatenation
```
v = [v, 5];
```

Char arrays

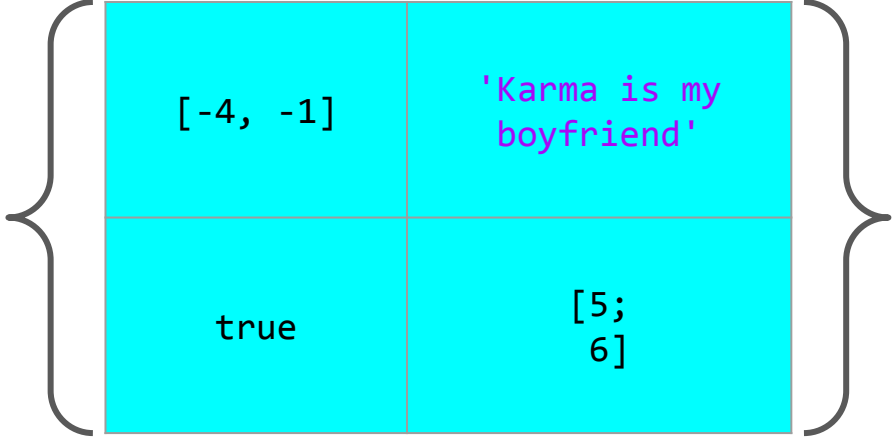
- Stores characters
- Assignment
```
s = 'bot';
```
- Indexing
```
s(3) = 'y';  % 'boy'
```
- Appending
```
s(4) = 's';  % 'boys'
```
- Concatenation
```
s = [s, 'enberries'];
    % 'boysenberries'
```

Cell arrays

- Stores anything!
- Assignment
```
c = {10, 'r', 'hi'};
```
- Indexing
```
c{2} = -0.5;
```
- Appending
```
c{4} = 'Midnight';
```
- Concatenation
```
c = [c, 5];
```

Simple example:

How can we store the following cell array?




```
% method 1
c = {[4,1], 'Karma is my boyfriend';
     true, [5;6]};
```

```
% method 2
c = {};
c{1,1} = [-4, -1];
c{1,2} = 'Karma is my boyfriend';
c{2,1} = true;
c{2,2} = [5;6];
```

```
% method 3
c1 = {[4,1], 'Karma is my boyfriend'};
c2 = {true, [5;6]};
cc = [c1; c2];
```

Comparison of bracket operators

- Square brackets [ ]
  - Create numeric arrays and char arrays
  - Concatenate (any) array contents

```
[3, 5, 8, 3, 0]
[ 'a' {'b' ['c' 'd']} ]    % { 'a', 'b', 'cd' }
```

- Curly braces { }
  - Create cell array enclosing contents

```
c = { 3 [1 4] 1 [5 9] }                % length(c) = ??
d = {'a' {'b' 'cd'}; true, 10 }        % [nr, nc] = size(d)
                                       % nr = ??, nc = ??
```