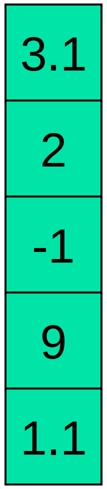
- Today's Lecture:
  - Cell arrays
  - File input/output

- Announcements:
  - Lab 3 exercise is *very important for A2 and Test2*. Make sure that you learn that material
  - Test 1 (second run, optional) Thursday 9/26 3:35p in PHS 203
  - Assignment 2 due Thursday 10/3 at 11:59pm

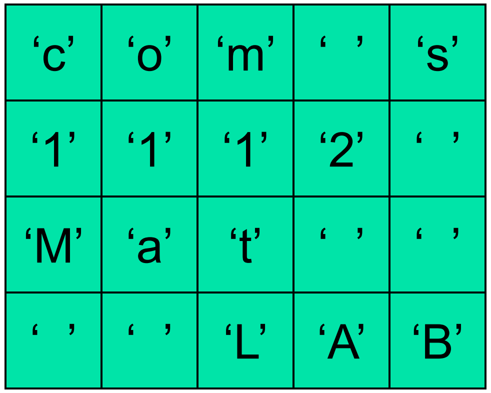
Matrix vs. Cell Array

Vectors and matrices store values of the same type in all components

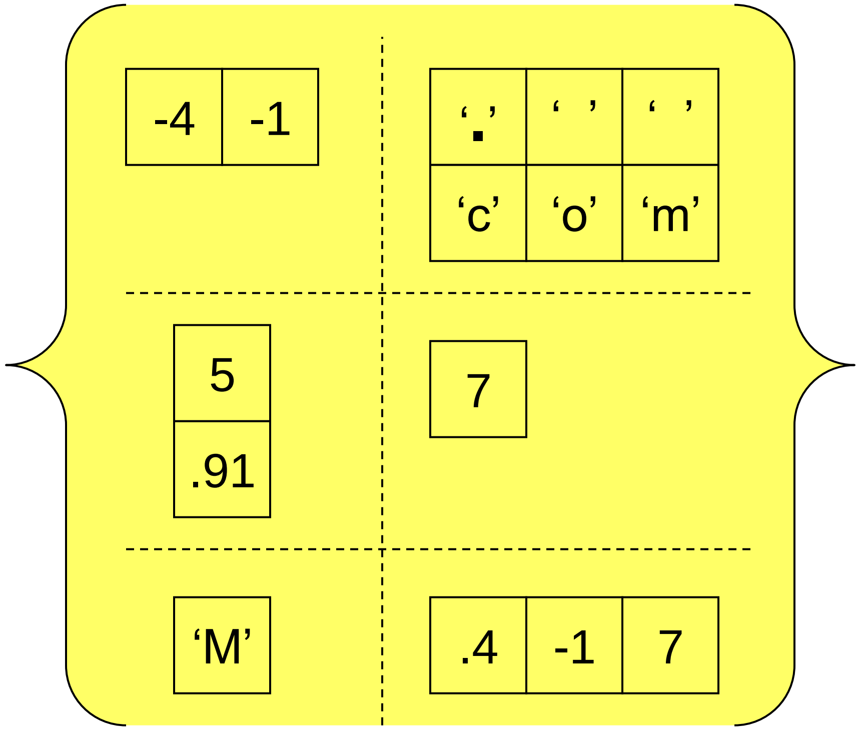
A cell array is a special array whose individual components may contain different types of data



5 x 1 matrix



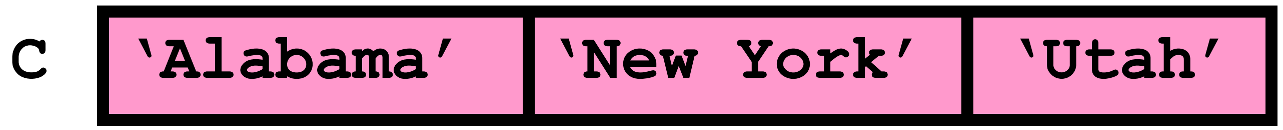
4 x 5 matrix



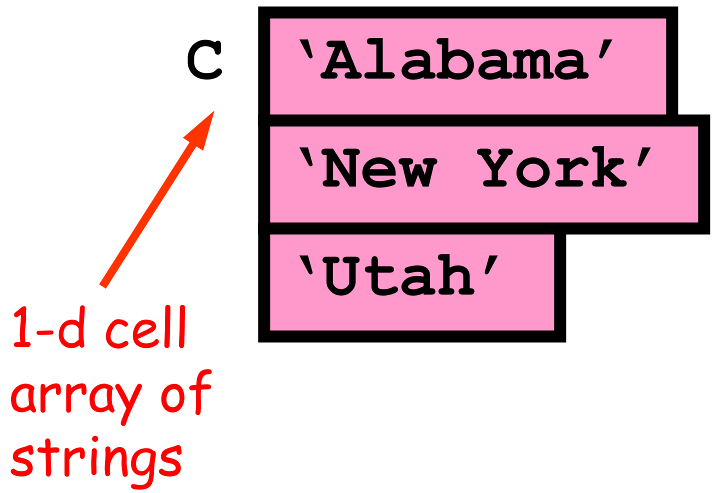
3 × 2 cell array

Cell Arrays of Strings

```
C= { ‘Alabama’,’New York’,’Utah’}
```



```
C= { ‘Alabama’;’New York’;’Utah’}
```



Contrast with
2-d array of characters

```
M= [‘Alabama ’; ...
    ‘New York’; ...
    ‘Utah    ’]
```

M							
‘A’	‘l’	‘a’	‘b’	‘a’	‘m’	‘a’	‘ ‘
‘N’	‘e’	‘w’	‘ ‘	‘Y’	‘o’	‘r’	‘k’
‘U’	‘t’	‘a’	‘h’	‘ ‘	‘ ‘	‘ ‘	‘ ‘

Use braces { } for creating and addressing cell arrays

Matrix

- Create

```
m= [ 5, 4 ; …
     1, 2 ; …
     0, 8 ]
```

- Addressing

```
m(2,1)= pi
```

Cell Array

- Create

```
C= { ones(2,2), 4          ; …
     'abc'    , ones(3,1) ; …
      9       , 'a cell'   }
```

- Addressing

```
C{2,1}= 'ABC'
C{3,2}= pi
disp(C{3,2})
```

Creating cell arrays…

```
     C= {'Oct', 30, ones(3,2)};
```

is the same as

```
     C= cell(1,3); % not necessary
     C{1}= 'Oct';
     C{2}= 30;
     C{3}= ones(3,2);
```

You can assign the empty cell array: `D = {}`

Example: Build a cell array of Roman numerals for 1 to 3999

```
C{1} = ‘I’
C{2} = ‘II’
C{3} = ‘III’
    :
C{2007} = ‘MMVII’
    :
C{3999} = ‘MMMXMXCIX’
```

Example

1904 = 1*1000 + 9*100 + 0*10 + 4*1

= M CM IV

= MCMIV

1	9	0	4

MCMIV

1	9	0	4

'M' concat. 'CM' concat. '' concat. 'IV'

''	''	''	''
M	C	X	I
MM	CC	XX	II
MMM	CCC	XXX	III
	CD	XL	IV
	D	L	V
	DC	LX	VI
	DCC	LXX	VII
	DCCC	LXXX	VIII
	CM	XC	IX

Concatenate entries from these cell arrays!

Ones-Place Conversion

```
function r = Ones2R(x)
% x is an integer that satisfies
%        0 <= x <= 9
% r is the Roman numeral with value x.

Ones = {'I', 'II', 'III', 'IV', ...
        'V', 'VI','VII', 'VIII', 'IX'};

if x==0
     r = '';
else
     r = Ones{x};
end
```

Ones-Place Conversion

```
function r = Ones2R(x)
% x is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value x.

Ones = {'I', 'II', 'III', 'IV', ...
        'V', 'VI','VII', 'VIII', 'IX'};

if x==0
     r = '';
else
     r = Ones{x};
end
```

Similarly, we can implement these functions:

```
function r = Tens2R(x)
% x is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value 10*x.
```

```
function r = Hund2R(x)
% x is an integer that satisfies
%       0 <= x <= 9
% r is the Roman numeral with value 100*x
```

```
function r = Thou2R(x)
% x is an integer that satisfies
%       0 <= x <=3
% r is the Roman numeral with value 1000*x
```

Now we can build the Roman numeral cell array for 1,…,3999

```
for a = 0:3
   for b = 0:9
      for c = 0:9
         for d = 0:9
            n = a*1000 + b*100 + c*10 + d;
            if n>0
               C{n} = [Thou2R(a) Hund2R(b)...
                                       Tens2R(c) Ones2R(d)];
            end
         end
      end
   end
end
```

Now we can build the Roman numeral cell array for 1,…,3999

```
for a = 0:3  % possible values in thous place
   for b = 0:9  % values in hundreds place
      for c = 0:9  % values in tens place
         for d = 0:9  % values in ones place
            n = a*1000 + b*100 + c*10 + d;
            if n>0
               C{n} = [Thou2R(a) Hund2R(b)...
                            Tens2R(c) Ones2R(d)];
            end
         end
      end
   end
end
```

Four strings concatenated together

The nth component of cell array C

Example: subset of clicker IDs

IDs

```
['d091314'; ...
 'h134d83'; ...
 'h4567s2'; ...
 'fr83209']
```

Find subset that begins with ‘h’

L

```
{'h134d83', ...
 'h4567s2'}
```

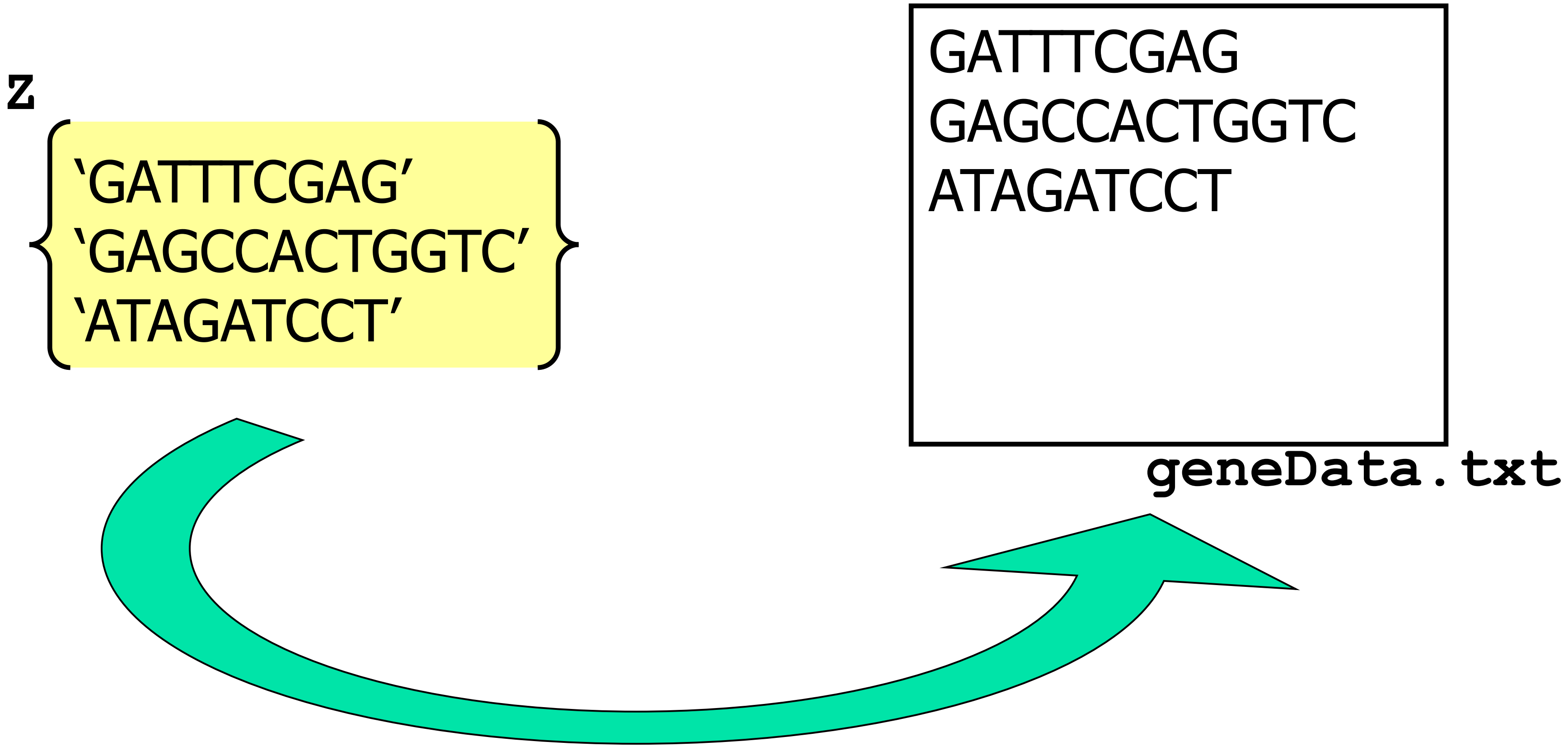
```
L= {};
k= 0;
for r=1:size(IDs,1)
  if IDs(r,1)==‘h’
    k= k+1;
    L{k }= IDs(r,:);
  end
end
```

Directly assign into a particular cell—<u>good</u>!

```
L= {};

for r=1:size(ID,1)
  if IDs(r,1)==‘h’

    L= [L, IDs(r,:)];
  end
end
```

Concatenate cells or cell arrays—<u>prone to problems</u>!

Example: Write a cell array of gene sequences to a file



A 3-step process to read data from a file or write data to a file

1. (Create and) open a file
2. Read data from or write data to the file
3. Close the file

1. Open a file

```
fid = fopen('geneData.txt', 'w');
```

An open file has a file ID, here stored in variable **`fid`**

Built-in function to open a file

Name of the file (created and) opened. **`txt`** and **`dat`** are common file name extensions for plain text files

'**`w`**' indicates that the file is to be opened for writing

Use '**`a`**' for appending

2. Write (print) to the file

```
fid = fopen(‘geneData.txt’, 'w');

for i=1:length(Z)
    fprintf(         '%s\n', Z{i});
end
```

2. Write (print) to the file

```
fid = fopen(‘geneData.txt’, 'w');

for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end
```

Printing is to be done to the file with ID `fid`

Substitution sequence specifies the *string* format (followed by a new-line character)

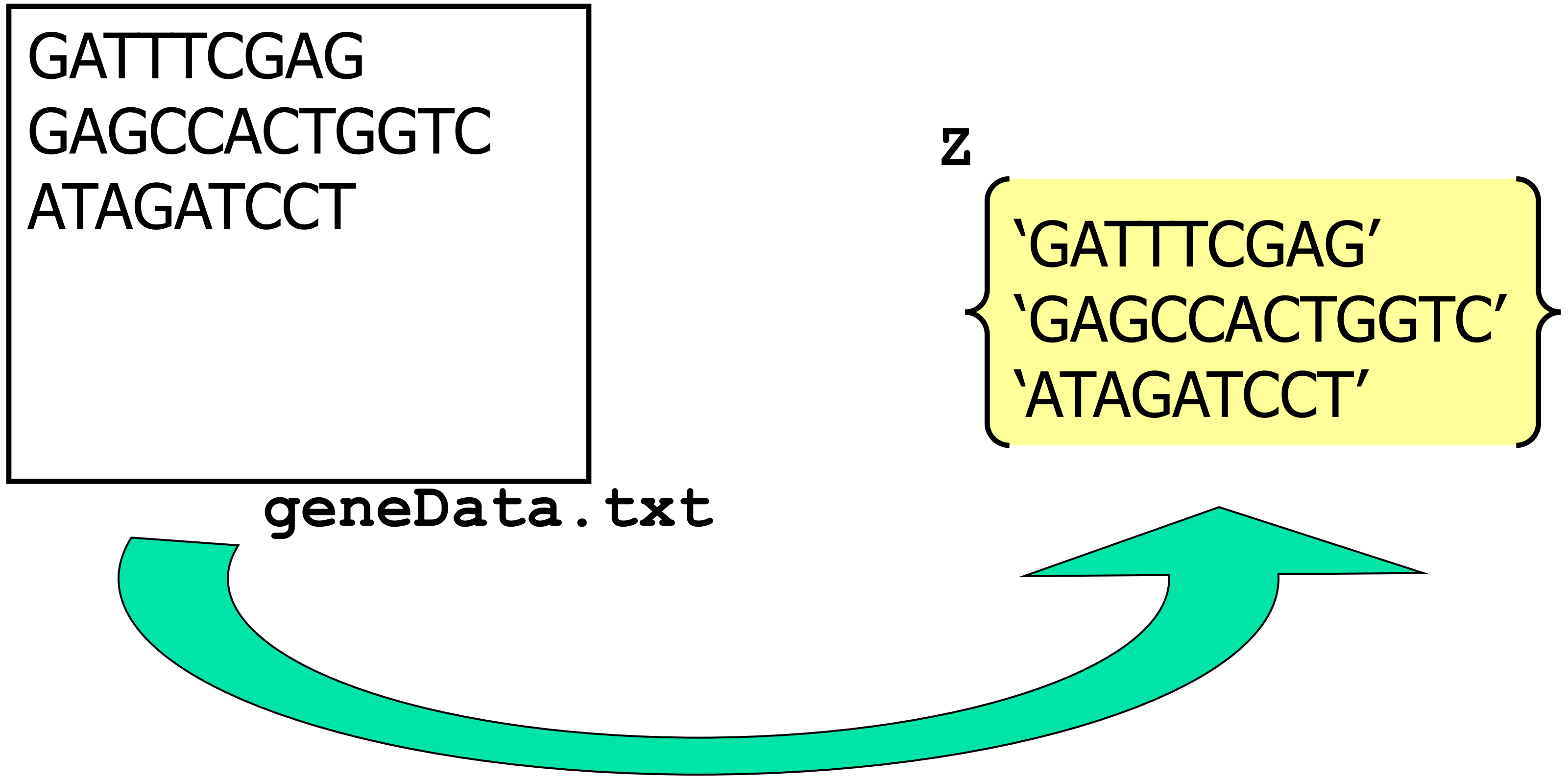
The **`i`** th item in cell array **`Z`**

3. Close the file

```
fid = fopen(‘geneData.txt’ ,'w');

for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end

fclose(fid);
```

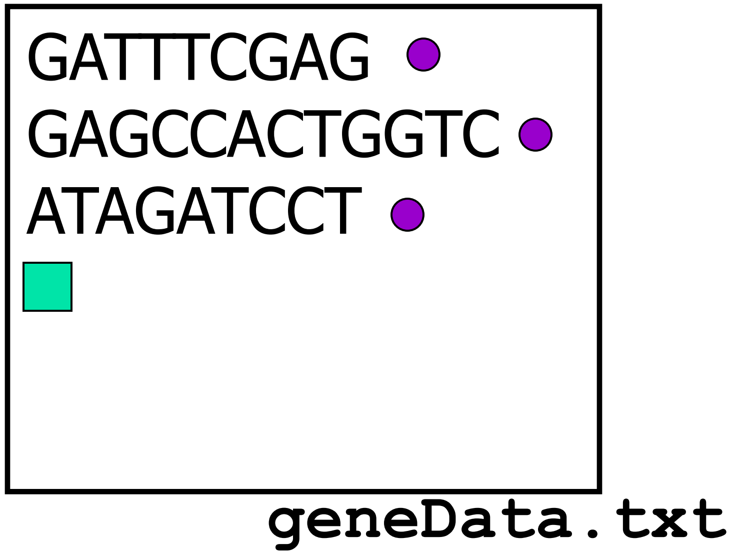
```
function cellArray2file(CA, fname)
% CA is a cell array of strings.
% Create a .txt file with the name
% specified by the string fname.
% The i-th line in the file is CA{i}

fid= fopen([fname `.txt’], 'w');
for i= 1:length(CA)
    fprintf(fid, '%s\n', CA{i});
end
fclose(fid);
```

Reverse problem: Read the data in a file line-by-line and store the results in a cell array



How are lines separated?
How do we know when there are no more lines?

In a file there are hidden “markers”



● Carriage return marks the end of a line

🟩 eof marks the end of a file

Read data from a file

1. Open a file
2. Read it line-by-line until eof
3. Close the file

1. Open the file

```
fid = fopen('geneData.txt', 'r');
```

An open file has a file ID, here stored in variable **`fid`**

Built-in function to open a file

Name of the file opened. **`txt`** and **`dat`** are common file name extensions for plain text files

'**`r`**' indicates that the file has been opened for **r**eading

2. Read each line and store it in cell array

```
fid = fopen('geneData.txt', 'r');

k= 0;
while ~feof(fid)
    k= k+1;
    Z{k}= fgetl(fid);
end
```

False until end-of-file is reached

Get the next line

3. Close the file

```
fid = fopen('geneData.txt', 'r');


k= 0;
while ~feof(fid)
    k= k+1;
    Z{k}= fgetl(fid);
end

fclose(fid);
```

```
function CA = file2cellArray(fname)
% fname is a string that names a .txt file
%   in the current directory.
% CA is a cell array with CA{k} being the
%   k-th line in the file.

fid= fopen([fname '.txt'], 'r');
k= 0;
while ~feof(fid)
   k= k+1;
   CA{k}= fgetl(fid);
end
fclose(fid);
```

A Detailed Read-File Example

From the protein database at

http://www.rcsb.org

we download the file `1b18.dat` which encodes the amino acid information for the protein with the same name. We want the xyz coordinates of the protein's "backbone."

The file has a long “header”

```
HEADER     MEMBRANE PROTEIN                        23-JUL-98   1BL8
TITLE      POTASSIUM CHANNEL (KCSA) FROM STREPTOMYCES LIVIDANS
COMPND     MOL_ID: 1;
COMPND    2 MOLECULE: POTASSIUM CHANNEL PROTEIN;
COMPND    3 CHAIN: A, B, C, D;
COMPND    4 ENGINEERED: YES;
COMPND    5 MUTATION: YES
SOURCE     MOL_ID: 1;
SOURCE    2 ORGANISM_SCIENTIFIC: STREPTOMYCES LIVIDANS;
```

Need to read past hundreds of lines
that are not relevant to us.

Eventually, the xyz data is reached…

```
MTRIX1   2 -0.736910 -0.010340  0.675910      112.17546    1
MTRIX2   2  0.004580 -0.999940 -0.010300       53.01701    1
MTRIX3   2  0.675980 -0.004490  0.736910      -43.35083    1
MTRIX1   3  0.137220 -0.931030  0.338160       80.28391    1
MTRIX2   3  0.929330  0.002860 -0.369240      -33.25713    1
MTRIX3   3  0.342800  0.364930  0.865630      -31.77395    1

ATOM      1  N   ALA A  23      65.191  22.037  48.576  1.00181.62           N
ATOM      2  CA  ALA A  23      66.434  22.838  48.377  1.00181.62           C
ATOM      3  C   ALA A  23      66.148  24.075  47.534  1.00181.62           C
```

Signal: Lines that begin with 'ATOM'

x

y

z

Where exactly are the xyz data?

```
  1-4      14-15             33-38   41-46   49-54   ←   Column nos. of interest

  ATOM     14  N   HIS A  25      68.656  24.973  44.142  1.00128.26           N
● ATOM     15  CA  HIS A  25      69.416  24.678  42.939  1.00128.26           C
  ATOM     16  C   HIS A  25      68.843  23.458  42.227  1.00128.26           C
  ATOM     17  O   HIS A  25      68.911  23.354  41.007  1.00128.26           O
  ATOM     18  CB  HIS A  25      70.881  24.416  43.300  1.00154.92           C
  ATOM     19  CG  HIS A  25      71.188  22.977  43.573  1.00154.92           C
  ATOM     20  ND1 HIS A  25      71.886  22.184  42.689  1.00154.92           N
  ATOM     21  CD2 HIS A  25      70.877  22.182  44.625  1.00154.92           C
  ATOM     22  CE1 HIS A  25      71.993  20.963  43.183  1.00154.92           C
  ATOM     23  NE2 HIS A  25      71.388  20.935  44.356  1.00154.92           N
  ATOM     24  N   TRP A  26      68.271  22.546  43.005  1.00 87.09           N
● ATOM     25  CA  TRP A  26      67.702  21.311  42.475  1.00 87.09           C
  ATOM     26  C   TRP A  26      66.187  21.378  42.339  1.00 87.09           C
  ATOM     27  O   TRP A  26      65.577  20.508  41.718  1.00 87.09           O

                                    x       y       z
```

Just getting what you need from a data file

- Read past all the header information
- When you come to the lines of interest, collect the xyz data
  - Line starts with ‘**`ATOM`**’
  - Cols 14-15 is ‘**`CA`**’

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Open the file.

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Initialize xyz arrays

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Iterate Until End of File

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Get the next line from file.

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
          x = [x; str2double(s(33:38))];
          y = [y; str2double(s(41:46))];
          z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Make Sure It's a
Backbone Amino Acid

```
fid = fopen('1bl8.dat', 'r');
x=[];y=[];z=[];
while ~feof(fid)
   s = fgetl(fid);
   if strcmp(s(1:4),'ATOM')
       if strcmp(s(14:15),'CA')
         x = [x; str2double(s(33:38))];
         y = [y; str2double(s(41:46))];
         z = [z; str2double(s(49:54))];
       end
   end
end
fclose(fid);
```

Update the x, y, z arrays