# Computer Programming with MATLAB


## Lesson 8: File Input/Output

by

Akos Ledeczi and Mike Fitzpatrick

# File Input/Output

VANDERBILT UNIVERSITY

- File:
  - Area in permanent storage (disk drive)
  - Stores information
  - Managed by the operating system
  - Can be copied or moved
  - Can be accessed by programs
- File Input/Output (I/O)
  - Data exchange between programs and computers
  - Data exchange between the physical world and computers
  - Saving your work so you can continue with it later
- MATLAB can handle
  - Mat-files and M-files AND text, binary, and Excel files

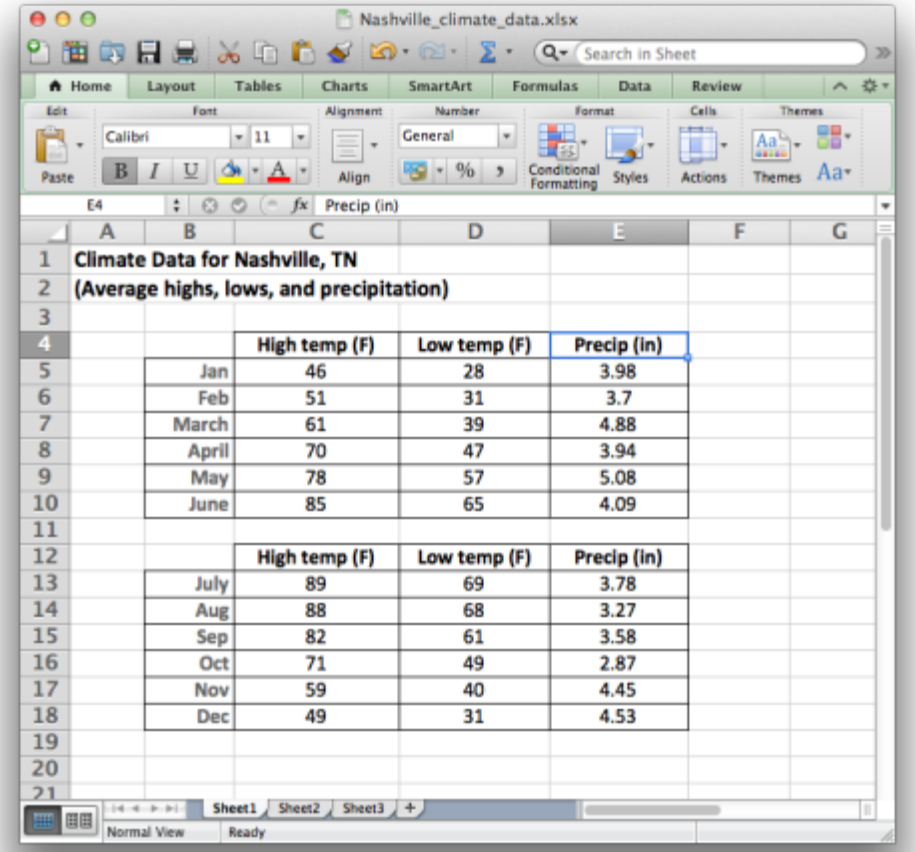
# Excel files

VANDERBILT UNIVERSITY

- Microsoft Excel® is a widely used data-analysis tool
- Many other programs support reading and writing Excel files
- MATLAB does too with two built-in functions
  - **`xlsread`**
  - **`xlswrite`**

# Reading Excel files

VANDERBILT UNIVERSITY

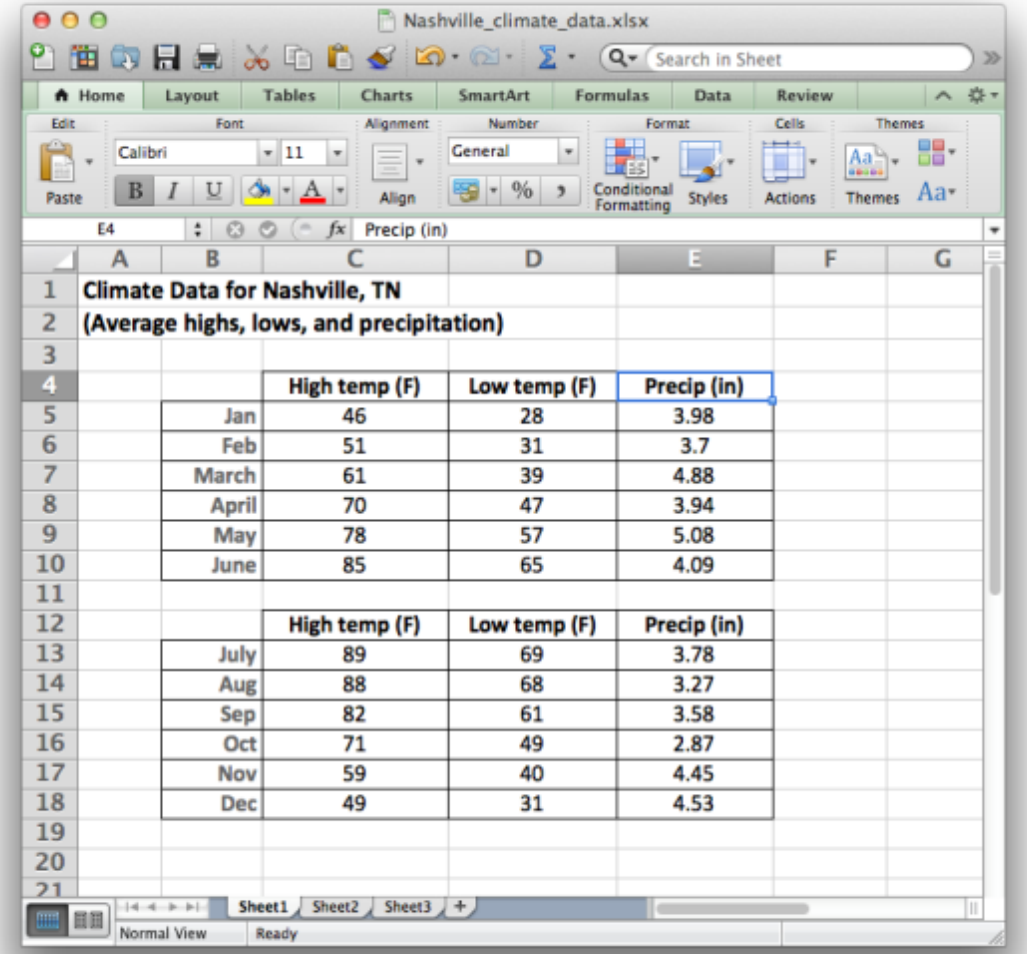



```
>> [num,txt,raw] = xlsread('Nashville_climate.xlsx');
```

# Numerical

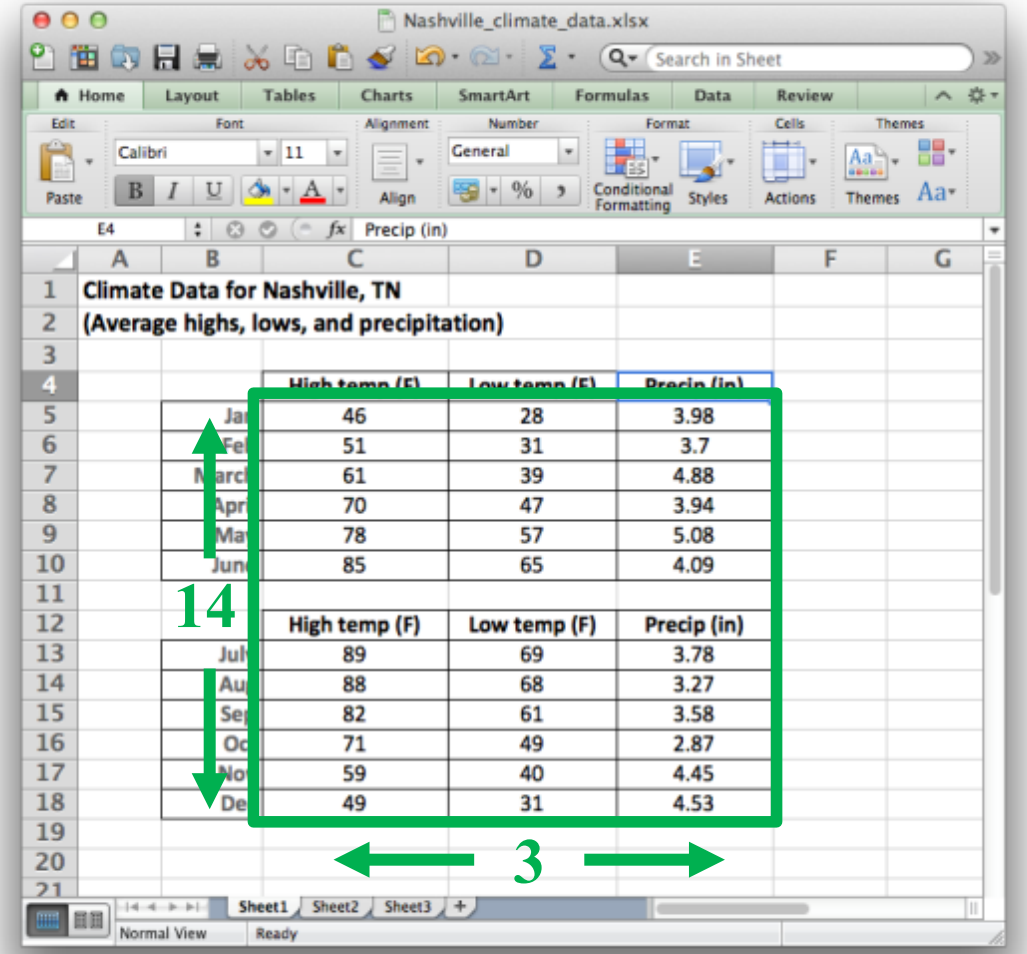
VANDERBILT UNIVERSITY

```
>> num
num =
        46     28     3.98
        51     31     3.7
        61     39     4.88
        70     47     3.94
        78     57     5.08
        85     65     4.09
       NaN    NaN    NaN
       NaN    NaN    NaN
        89     69     3.78
        88     68     3.27
        82     61     3.58
        71     49     2.87
        59     40     4.45
        49     31     4.53
```



# Numerical

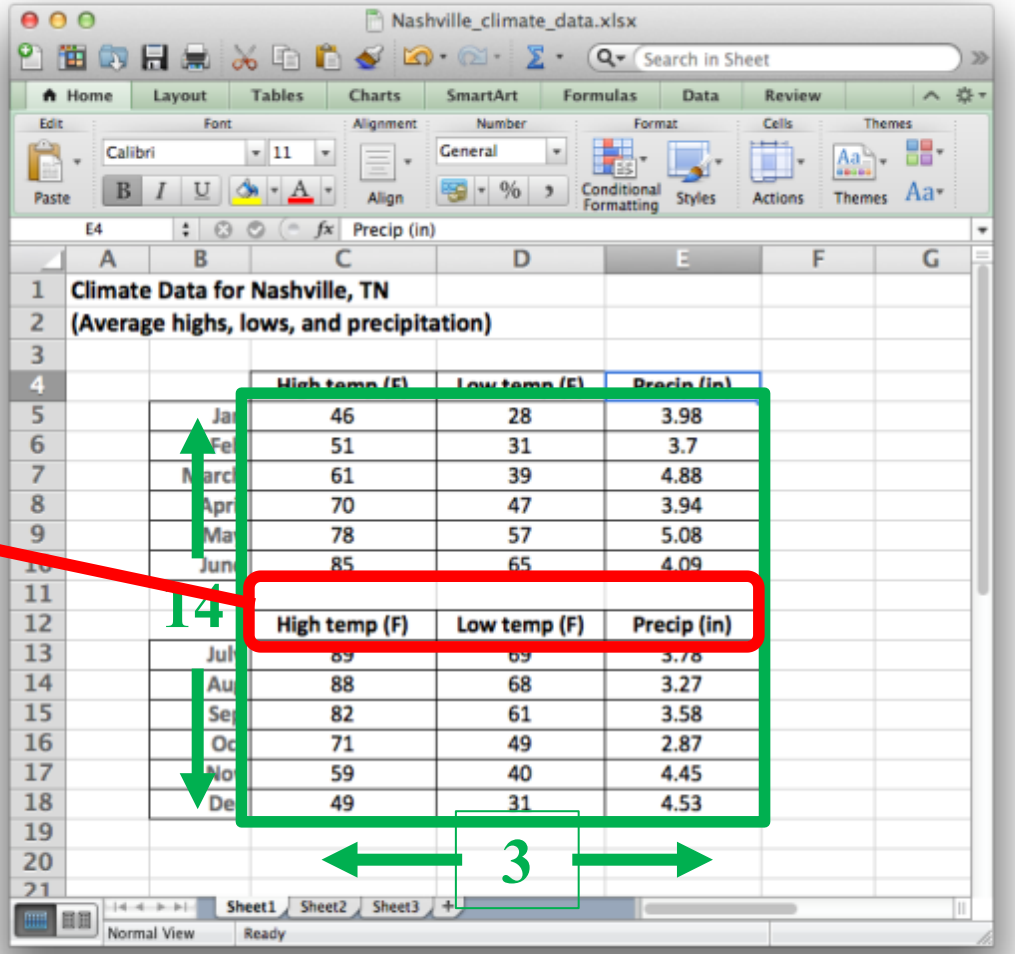
```
>> num
num =
        46     28     3.98
        51     31     3.7
        61     39     4.88
        70     47     3.94
        78     57     5.08
        85     65     4.09
       NaN    NaN    NaN
       NaN    NaN    NaN
        89     69     3.78
        88     68     3.27
        82     61     3.58
        71     49     2.87
        59     40     4.45
        49     31     4.53
```



# Numerical

VANDERBILT UNIVERSITY

```
>> num
num =
        46     28     3.98
        51     31     3.7
        61     39     4.88
        70     47     3.94
        78     57     5.08
        85     65     4.09
       NaN    NaN    NaN
       NaN    NaN    NaN
        89     69     3.78
        88     68     3.27
        82     61     3.58
        71     49     2.87
        59     40     4.45
        49     31     4.53
```



# Text

VANDERBILT UNIVERSITY

```
>> txt =
txt =
      [1x30 char] ''      ''               ''               ''
      [1x40 char] ''      ''               ''               ''
      ''          ''      ''               ''               ''
      ''          ''      'High temp (F)'  'Low temp (F)'   'Precip (in)'
      ''          'Jan'   ''               ''               ''
      ''          'Feb'   ''               ''               ''
      ''          'March' ''               ''               ''
      ''          'April' ''               ''               ''
      ''          'May'   ''               ''               ''
      ''          'June'  ''               ''               ''
      ''          ''      ''               ''               ''
      ''          ''      'High temp (F)'  'Low temp (F)'   'Precip (in)'
      ''          'July`  ''               ''               ''
      ''          'Aug'   ''               ''               ''
      ''          'Sep'   ''               ''               ''
      ''          'Oct'   ''               ''               ''
      ''          'Nov'   ''               ''               ''
      ''          'Dec'   ''               ''               ''
```

# All data: cell array

VANDERBILT UNIVERSITY

```
>> raw
raw =
      [1x30 char] [ NaN] [ NaN]           [ NaN]          [ NaN]
      [1x40 char] [ NaN] [ NaN]           [ NaN]          [ NaN]
      [ NaN]      [ NaN] [ NaN]           [ NaN]          [ NaN]
      [ NaN]      [ NaN] 'High temp (F)' 'Low temp (F)' 'Precip (in)'
      [ NaN]     'Jan'   [ 46]            [ 28]           [ 3.98]
      [ NaN]     'Feb'   [ 51]            [ 31]           [ 3.7]
      [ NaN]     'March' [ 61]            [ 39]           [ 4.88]
      [ NaN]     'April' [ 70]            [ 47]           [ 3.94]
      [ NaN]     'May'   [ 78]            [ 57]           [ 5.08]
      [ NaN]     'June`  [ 85]            [ 65]           [ 4.09]
      [ NaN]     [ NaN]  [ NaN]           [ NaN]          [ NaN]
      [ NaN]     [ NaN]  'High temp (F)' 'Low temp (F)' 'Precip (in)'
      [ NaN]     'July`  [ 89]            [ 69]           [ 3.78]
      [ NaN]     'Aug'   [ 88]            [ 68]           [ 3.27]
      [ NaN]     'Sep'   [ 82]            [ 61]           [ 3.58]
      [ NaN]     'Oct'   [ 71]            [ 49]           [ 2.87]
      [ NaN]     'Nov'   [ 59]            [ 40]           [ 4.45]
      [ NaN]     'Dec'   [ 49]            [ 31]           [ 4.53]
```

# Text files

VANDERBILT UNIVERSITY

- Text files contain characters
- They use an encoding scheme:
  - ASCII or
  - Any one of many other schemes
  - MATLAB takes care of encoding and decoding
- Before using a text file, we need to open it
- Once done with the file, we need to close it

# Opening text files

VANDERBILT UNIVERSITY

- Opening: `fid = fopen(filename, permission)`
- Closing: `fclose(fid)`
- `fid`: Unique file identifier for accessing file
- Permission: what we want to do with the file—
  - read, write, overwrite, append, etc.

| 2ND ARGUMENT | PERMISSION |
|---|---|
| `'rt'` | open text file for reading |
| `'wt'` | open text file for writing; discard existing contents |
| `'at'` | open or create text file for writing; append data to end of file |
| `'r+t'` | open (do not create) text file for reading and writing |
| `'w+t'` | open or create text file for reading and writing; discard existing contents |
| `'a+t'` | open or create text file for reading and writing; append data to end of file |

# Reading text files

VANDERBILT UNIVERSITY

- One line at a time
- **`type`** prints a text file in the command window
- Let's re-implement it:

```
function view_text_file(filename)
fid = fopen(filename,'rt');
if fid < 0
    error('error opening file %s\n\n', filename);
end

% Read file as a set of strings, one string per line:
oneline = fgets(fid);
while ischar(oneline)
    fprintf('%s',oneline) % display one line
    oneline = fgets(fid);
end
fprintf('\n');
fclose(fid);
```

# Reading text files


- Reading lines into string variables is easy
- Parsing these strings to get numerical data is much harder
- Not covered
- Binary files are more suited for numerical data

# Binary files

VANDERBILT UNIVERSITY

- Binary file = “not a text file”
- Many different ways to represent numbers
- All we need to know are their types.
- Binary files need to be
  - Opened with **`fopen`**
  - Closed with **`fclose`**

| 2ND ARGUMENT | PERMISSION |
|---|---|
| `'r'` | open binary file for reading |
| `'w'` | open binary file for writing; discard existing contents |
| `'a'` | open or create binary file for writing; append data to end of file |
| `'r+'` | open (do not create) binary file for reading and writing |
| `'w+'` | open or create binary file for reading and writing; discard existing contents |
| `'a+'` | open or create binary file for reading and writing; append data to end of file |

# Writing binary files

VANDERBILT UNIVERSITY

- Data type is important
- Example: write a double array into a binary file

```
function write_array_bin(A,filename)
fid = fopen(filename,'w+');
if fid < 0
    error('error opening file %s\n', filename);
end

fwrite(fid,A,'double');

fclose(fid);
```

# Reading binary files

VANDERBILT UNIVERSITY

- Example: read a double array from a binary file

```
function A = read_bin_file(filename,data_type)
fid = fopen(filename,'r');
if fid < 0
    error('error opening file %s\n',filename);
end

A = fread(fid,inf,data_type);

fclose(fid);
```