# Input & Output

Scientific Computing
Fall, 2018
Paul Gribble

---



---

Most of the time you will be writing programs that analyse data—whether those data are collected from experiments, or generated by models and simulations. We will need to be able to read in data from a file. It will also be useful to be able to write data out to a file.

The MathWorks online documentation has a page devoted to importing and exporting data, here:

Data Import and Export

Here we will go over how to read and write to some common types of files including ASCII files (plain text), MATLAB `.mat` files, as well as other binary formats. The MathWorks has a page listing all of the various file formats that MATLAB knows how to import, it is quite lengthy:

Supported File Formats for Import and Export

In general, there are two types of file formats, ASCII files (otherwise known as plain text files) and binary files. In fact, this is a lie and there really is only one file type, namely binary files, since all data are ultimately stored as 0s and 1s (binary)—but we have conventions, like the ASCII code, which allow us to make assumptions, to make life easier. So we know that if a (binary) file is coded using a series of bytes, each of which corresponds to an ASCII code, then this file is in fact a "plain text" or ASCII file (and you can read it using any plain text editor like vim, emacs, sublime text, notepad, even MS Turd will open plain text files). Binary files include things like image formats such as `.png`, `.jpeg`, sound files such as `.mp3` and video files such as `.mp4` and `.mov`. Like I said before though, really, all files are binary. It's just that we can open files containing ASCII code using many programs which know how to interpret the 0s and 1s as ASCII codes.

## 1 Plain text files

If your data are stored in a plan text file (ascii) then you can use the MATLAB function `load` to load in the file. For example let's say we have a plain text file called `mydata.txt` that contains the following:

```
2 3
4 5
6 7
8 9
```

Then we can use the `load` command to load the data:

```
>> d = load('mydata.txt');
>> whos
```

```
  Name      Size            Bytes  Class     Attributes

  d         4x2                64  double

>> d

d =

     2     3
     4     5
     6     7
     8     9
```

You can also load the data without giving the `load` function an output variable to store the data—in this case the data will be stored in a new variable with the same name as the filename (but any file suffix, such as `.txt` stripped):

```
>> load mydata.txt
>> whos
  Name        Size            Bytes  Class     Attributes

  mydata      4x2                64  double

>> mydata

mydata =

     2     3
     4     5
     6     7
     8     9
```

For loading ASCII files, the file must contain a rectangular table of numbers, with an equal number of elements in each row. Delimiters such as spaces, commas, semicolons or tabs can be used—but they have to the same througout the file. If these conditions are not met, MATLAB will complain. For example if our data file `mydata2.txt` looks like this:

```
2 3
4 5
6 7 8
8 9
```

MATLAB will complain about number of columns not being the same:

```
>> load mydata2.txt
Error using load
Number of columns on line 3 of ASCII file mydata2.txt must be the same as previous lines.
```

To save data to an ASCII file, you can use the `save` command. For example let's say we have data store in a variable called `data` that looks like this:

```
data =
```

```
    0.6557    0.7577
    0.0357    0.7431
    0.8491    0.3922
    0.9340    0.6555
    0.6787    0.1712
```

Then we can use `save` with the `-ascii` flag to save this into an ASCII file:

```
>> save mynewdata.txt data -ascii
```

The first argument (`mynewdata.txt`) is the filename of the new file to be created. The second argument (`data`) is the name of the variable to be saved to the file, and the third argument (`-ascii`) is a flag to the `save` command that tells MATLAB to save the data in plain text (ASCII) format. Now if we look at the new file (for example by opening it in the MATLAB text editor) that was created, `mynewdata.txt` it looks like this:

```
   6.5574070e-01   7.5774013e-01
   3.5711679e-02   7.4313247e-01
   8.4912931e-01   3.9222702e-01
   9.3399325e-01   6.5547789e-01
   6.7873515e-01   1.7118669e-01
```

Note how it has been saved in scientific notation.

If you want finer control over how things are stored in an ASCII file, you can read (as well as write) using lower-level control using MATLAB's built-in functions `fprintf` and `fscanf`. These mirror the functions with the same name that may be familiar to you if you have programmed in C before. Here is an example of writing to an ASCII file where we want a very specific format:

```
data = [
    0.6557    0.7577
    0.0357    0.7431
    0.8491    0.3922
    0.9340    0.6555
    0.6787    0.1712
];

fid = fopen('myfile.txt','w');
fprintf(fid, 'myfile.txt contains some data\n');
for i=1:size(data,1)
    fprintf(fid,'item 1.1: %.4f, item 1.2: %.4f\n', data(i,1), data(i,2));
end
fprintf(fid, 'end of data\n');
fclose(fid);
```

This creates a file called `myfile.txt` that looks like this:

```
myfile.txt contains some data
item 1.1: 0.6557, item 1.2: 0.7577
item 1.1: 0.0357, item 1.2: 0.7431
item 1.1: 0.8491, item 1.2: 0.3922
item 1.1: 0.9340, item 1.2: 0.6555
```

```
item 1.1: 0.6787, item 1.2: 0.1712
end of data
```

## 2 Binary files

MATLAB has its own binary format for files, denoted using a `.mat` file suffix. The `save` and `load` functions in MATLAB with no other options use this default binary format. The advantage of MATLAB's binary format over an ASCII format is (1) your data files will be smaller in size, and (2) with MATLAB's `.mat` format you can store more than one variable (and variables of different kinds) in a single file. For example here we store a scalar variable called `mynumber`, a vector called `myvector`, a matrix called `mymatrix` and a structure called `mystructure` in a single binary `.mat` file called `myfile.mat`:

```
>> whos
  Name             Size            Bytes  Class     Attributes

  mymatrix         4x2                64  double
  mynumber         1x1                 8  double
  mystructure      1x1               840  struct
  myvector         1x7                56  double

>> save myfile mynumber myvector mymatrix mystructure
```

Now there is a file in my working directory called `myfile.mat`. I can now load the file (and all the variables contained within it) into MATLAB's memory using the `load` function. First I clear the memory to demonstrate that I'm not cheating:

```
>> clear
>> whos
```

Now I load the file:

```
>> load myfile
>> whos
  Name             Size            Bytes  Class     Attributes

  mymatrix         4x2                64  double
  mynumber         1x1                 8  double
  mystructure      1x1               840  struct
  myvector         1x7                56  double
```

## 3 ASCII or binary?

The question of which file format to use as you go forward and write programs for analysing your data is an interesting one to consider. For long-term archival purposes, I would suggest storing your data in an ASCII format, so that it remains readable by human eyes. There will always be programs to read ASCII files. The risk of storing data in a binary format is that (a) whatever program you used to save the data will no longer be easily accessible in the future, and/or (b) you may not even remember what the binary format is. The disadvantage of storing data in ASCII format is that the files will be larger than if they were stored in a binary format. The availability and affordability of large amounts of storage is growing so quickly however,

so perhaps one does not have to worry too much about this.