Working with Files

Table of Contents

Directories and File Paths: Organizing & Finding Your Files

In a standard file system, Folders are used to organize Files. The location of a file is called a "**file path**". Being able to specify a file path either "absolutely" (where exactly it is on a computer) or "relatively" (where it is relative to another path, often the *working directory*) makes it much easier to read and write data files.

In the section below, we'll get some practice organizing folders, specifying paths, and creating folders.

Code	
`path = "C:/Program Files/file.m"` `path = "C:\Program Files\file.m"`	(Microsoft Windows) Absolute filepaths to a file. (Note that it starts with a drive letter, and that both forward and backward slashes work equally well."
`path = "/usr/folder/file.m"`	(Apple Mac OS, Linux) Absolute filepath to a file. (note that it starts with a slash, and only forward slashes are accepted."
`path =  "./folder/file.m"` `path =   "folder/file.m"`	Relative filepaths to the working directory
`path = "../folder/file.m"`	Relative filepath to the *parent folder* of the working directory
---	---
`pwd()`	Get the current working directory
`dir(".")`	Show the files and folders in the current working directory
`mkdir(path)`	Make a directory at the given path
`rmdir(path)`	Delete the directory at the given path
`cd(path)`	Change the working direcotory to the given path
`[folder, filename] = fileparts(path)`	get the folder and filename, separated
`[folder, base, ext] = fileparts(path)`	get the folder, filename (without the extension), and the file extension (e.g. ".mat"), separated
`fullfile("project", "data", "raw")`	Build a path with the correct path seperator connecting the parts given.

matlab.desktop.editor.getActiveFilename	The path to the current file.

Exercises

Get the current working directory

```
pwd()
```

```
ans =
'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3'
```

```

```

Make the folder "project" in the current directory

```
mkdir("project")
```

```
Warning: Directory already exists.
```

Change the working directory to the project folder.

```
cd("project")
```

Inside the project folder, make the "data/raw" folder

```
mkdir("data/raw")
```

```
Warning: Directory already exists.
```

Make the other standard data subfolders: "data/processed" and "data/final"

```
mkdir("data/processed")
```

```
Warning: Directory already exists.
```

```
mkdir("data/final")
```

```
Warning: Directory already exists.
```

Have Matlab check: What folders are inside the "data" folder?

```
dir("data")
```

```
.          ..         final      processed  raw
```

Save all the information from the `dir()` function to a variable. What additional information does matlab provide about the "data" folder?

```
data = dir("data")
```

data = 5×1 struct

Fields	name	folder	date	bytes	isdir	datenum
1	'.'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project/data'	'28-Nov-2023 14:03:55'	0	1	7.3922e+05
2	'..'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project/data'	'28-Nov-2023 15:48:20'	0	1	7.3922e+05
3	'final'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project/data'	'28-Nov-2023 14:03:55'	0	1	7.3922e+05
4	'processed'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project/data'	'28-Nov-2023 14:03:55'	0	1	7.3922e+05
5	'raw'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project/data'	'28-Nov-2023 14:03:49'	0	1	7.3922e+05

Text: Writing and Reading

In a standard file system, Folders are used to organize Files, and Files are used to store Data. If the data is formatted in a way that can be reaad by a standard text editor, the file is generally referred to as a "text file".

Text files can contain arbitrary text, like a journal article or code, but it can also contain data. If the data is written in a consistent way, it can be read back into a data analysis program.

In this section, we'll use the **`readlines()`** and **`writelines()`** functions to read and write arbitrary text data into files. We'll also quickly look at the **`jsonencode()`**

Code	Description
`readlines(filename)`	Read the text from a text file
`writelines(["a", "b"], filename)`	Write each string in the array as a line in the text file.
`writelines(["a", "b"], filename, "LineEnding", '\t')`	At the end of each string in the array, put a tab character ("\t") instead of the default newline character ("\n")
`writelines(["a", "b"], filename, "WriteMode", "append")`	Instead of overwriting the file completely, append the new text to the end of the file

Exercises

Example: Create a file called "hello.txt" and write the phrase, "Hello, World!" in the file.

```
writelines("Hello, World!", "hello.txt")
```

Read back the text from the file.

```
readlines("hello.txt")
```

```
ans = 2×1 string
"Hello, Wo…
""
```

Create a file called "bye.txt" and write the phrase, "Goodbye, World!" in the file.

```
writelines("Goodbye, World!", "bye.txt")
```

Read back the text from the file.

```
readlines("bye.txt")
```

```
ans = 2×1 string
"Goodbye, Wo…
""
```

Create a file called "aloha.txt" and put both "Hello" and "Goodbye" in the file, each on its own line.

```
writelines(["Hello", "Goodbye"], "aloha.txt")
```

Read back the text from the file.

```
readlines("aloha.txt")
```

```
ans = 3×1 string
"Hello"
"Goodbye"
""
```

Create a CSV File "bdays.csv" that looks like this:

```
name,date
Adrian,22.04.2013
Emma,23.07.2022
```

Tip: Write the data once for each line, using **`WriteMode "append"`** to keep from overwriting the data.
Use commas to separate each string in a line using the **`join()`** function (e.g. **`join(["hello", "world", ','))`**.

Data to use:

```
adrian = ["Adrian", "22.04.2013"];
emma = ["Emma", "23.07.2022"];
```

Write Code:

```
writelines(join(["name", "date"], ','), 'bdays.csv');
writelines(join(adrian, ','), 'bdays.csv', "WriteMode", "append");
writelines(join(emma, ','), 'bdays.csv', "WriteMode", "append");
```

Read the data back into Matlab as an array of strings:

```
readlines("bdays.csv")
```

```
ans = 4×1 string
"name,date"
"Adrian,22.04.…
"Emma,23.07.2022"
""
```

Table Data Files: Writing & Reading to Text and Binary Files

When saving data, we want to consider how **interoperable** our data files are: How much work will it be to read the data back in again in other scripts, and will other programs be able to easily work with it? Using established, open file formats help with this a lot!

Here, we'll look at two different formats that are made for writing *tabular* data (i.e. tables): CSV (a text-based format) and Parquet(a binary format). Each of these formats is very popular, and Matlab comes with built-in functions for reading and writing to these formats. All they need is a table!

Why two different formats for one data structure? Text files often have the advantage with small files: they are easy to work with and are often quite small. But as they files get bigger, it's good to switch to binary formats like Parquet, which can compress the data into smaller filesizes, plus can add extra features, like data validation (making sure the data is saved correctly for analysis) and data consistency (making sure that the data is loaded the same way, even when different toolboxes or even different programming languages are used).

`writetable(t, filename)`	Write a Matlab table to a text CSV file.
`readtable(filename)`	Read a text CSV file into a Matlab table
---	---
`parquetwrite(filename, t)`	Write a Matlab table to a binary Parquet file.
`parquetread(filename)`	Read a binary Parquet file into a Matlab table.
---	---
`t = table()`	Create an empty Matlab Table
`t.var1 = data`	assign `data` to a the column `var1` of a table.
`data'`	transpose `data` (e.g. from rows to columns, or vice-versa)

Mouse Data

Do the exercises using the following data:

```
mouse = ["Müller", "Cori", "Radnitz"]';
dob = ["09.11.2016", "09.11.2016", "09.11.2016"]';
sex = ["M", "F", "F"]';
```

Make a Matlab Table from the data with the following columns: `mouse_name`, `date_of_birth`, and `sex`.

```
t = table();
t.mouse_name =  mouse;
t.date_of_birth = dob;
t.sex = sex;
t
```

`t = 3×3 table`

	mouse_name	date_of_birth	sex
1	"Müller"	"09.11.2016"	"M"
2	"Cori"	"09.11.2016"	"F"
3	"Radnitz"	"09.11.2016"	"F"

Exercises

Write the table to a text CSV file mice.csv.

```
writetable(t, "mice.csv");
```

Verify the data by reading the CSV file back into a Matlab table. Did it load correctly?

```
readtable("mice.csv")
```

ans = 3×3 table

	mouse_name	date_of_birth	sex
1	'Müller'	09-Nov-2016	'M'
2	'Cori'	09-Nov-2016	'F'
3	'Radnitz'	09-Nov-2016	'F'

Write the table to a binary Parquet file subjects.parquet.

```
parquetwrite("mice.parquet", t);
```

Verify the data by reading the CSV file back into a Matlab table. Did it load correctly?

```
parquetread("mice.parquet")
```

ans = 3×3 table

	mouse_name	date_of_birth	sex
1	"Müller"	"09.11.2016"	"M"
2	"Cori"	"09.11.2016"	"F"
3	"Radnitz"	"09.11.2016"	"F"

Which file is bigger: the CSV file or the Parquet file? Use [i] = dir("mice*") to view the file sizes of each file that starts with the word "subjects".

```
[a] = dir("mice*")
```

a = 2×1 struct

Fields	name	folder	date	bytes	isdir	datenum
1	'mice.csv'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project'	'28-Nov-2023 18:02:59'	89	0	7.3922e+05
2	'mice.parquet'	'/home/ben/ibots/iBOTS-Tools/workshops/	'28-Nov-2023 18:03:00'	1158	0	7.3922e+05

Fields	name	folder	date	bytes	isdir	datenum
		matlab-workshop-1/plan/day3/project'				

Subject Groups

Use the data below:

```
sid = 1:500;
tgroup = randsample({'control', 'treatment'}, 500, true)';
sex = randsample({'M', 'F'}, 500, true);
```

Make a Matlab table with three columns: `subject_id`, `treatment_group`, and `sex`. (Note: you may have to transpose some of the variables in order to make them into columns).

```
t = table();
t.subject_id = sid';
t.treatment_group = tgroup;
t.sex = sex';
t
```

t = 500×3 table

	subject_id	treatment_group	sex
1	1	'treatment'	'F'
2	2	'treatment'	'F'
3	3	'control'	'F'
4	4	'treatment'	'M'
5	5	'treatment'	'M'
6	6	'control'	'M'
7	7	'control'	'F'
8	8	'treatment'	'F'
9	9	'treatment'	'M'
10	10	'treatment'	'F'
11	11	'control'	'F'
12	12	'treatment'	'F'
13	13	'treatment'	'F'
14	14	'control'	'F'

⋮

Write the table to a text CSV file `subjects.csv`.

```
writetable(t, "subjects.csv");
```

Verify the data by reading the CSV file back into a Matlab table. Did it load correctly?

```
readtable("subjects.csv")
```

```
ans = 500×3 table
```

	subject_id	treatment_group	sex
1	1	'treatment'	'F'
2	2	'treatment'	'F'
3	3	'control'	'F'
4	4	'treatment'	'M'
5	5	'treatment'	'M'
6	6	'control'	'M'
7	7	'control'	'F'
8	8	'treatment'	'F'
9	9	'treatment'	'M'
10	10	'treatment'	'F'
11	11	'control'	'F'
12	12	'treatment'	'F'
13	13	'treatment'	'F'
14	14	'control'	'F'

⋮

Write the table to a binary Parquet file `subjects.parquet`.

```
parquetwrite("subjects.parquet", t);
```

Verify the data by reading the CSV file back into a Matlab table. Did it load correctly?

```
parquetread("subjects.parquet")
```

```
ans = 500×3 table
```

	subject_id	treatment_group	sex
1	1	"treatment"	"F"
2	2	"treatment"	"F"
3	3	"control"	"F"
4	4	"treatment"	"M"
5	5	"treatment"	"M"
6	6	"control"	"M"

	subject_id	treatment_group	sex
7	7	"control"	"F"
8	8	"treatment"	"F"
9	9	"treatment"	"M"
10	10	"treatment"	"F"
11	11	"control"	"F"
12	12	"treatment"	"F"
13	13	"treatment"	"F"
14	14	"control"	"F"

⋮

Which file is bigger: the CSV file or the Parquet file? Use `[i] = dir("subjects*")` to view the file sizes of each file that starts with the word "subjects".

```
[a] = dir("subjects*")
```

a = 2×1 struct

Fields	name	folder	date	bytes	isdir	datenum
1	'subjects.csv'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project'	'28-Nov-2023 18:03:01'	7409	0	7.3922e+05
2	'subjects.parquet'	'/home/ben/ibots/iBOTS-Tools/workshops/matlab-workshop-1/plan/day3/project'	'28-Nov-2023 18:03:01'	3904	0	7.3922e+05

Extra: Saving Metadata as a JSON File:

Text files can also be used to encode arbitrary data, if there is a specific format that they are written in.

One popular format for small datasets is the `json` format. Let's try saving some metadata from a structure into it!

Code	Description
`text = jsonencode(data);`	Turn any data structure into json-formatted text.
`text = jsonencode(data, "PrettyPrint", true)`	format the json text in a way that makes it nice to view in text editors.

`data = jsondecode(text)`	Turn a json-formatted text into a data structure for analysis.
`text = join(["string1", "string2"])`	Concatenate an array of strings into a single text string (great for use before `jsondecode()`.

Data to Use:

```
data = struct();
data.firstname = "Marty";
data.lastname = "McFly";
data.mph = 88;
data.to_date = "26.10.1985";
data
```

```
data = struct with fields:
    firstname: "Marty"
     lastname: "McFly"
          mph: 88
      to_date: "26.10.1985"
```

Exercises:

Encode the data as a JSON-formatted string using the `jsonencode()` function, then write the string to the text file `b2f.json`. (Example: `jsonencode([1 2], "PrettyPrint", true)`)

```
text = jsonencode(data, "PrettyPrint", true)
```

```
text =
    '{
      "firstname": "Marty",
      "lastname": "McFly",
      "mph": 88,
      "to_date": "26.10.1985"
     }'
```

```
writelines(text, 'b2f.json')
```

Read the data in `b2f.json` back into Matlab as an array of strings, join the array to be a single string using the `join()` function, then decode the first string back into a Matlab Struct using `jsondecode()`.

```
jsondecode(join(readlines("b2f.json")))
```

```
ans = struct with fields:
    firstname: 'Marty'
     lastname: 'McFly'
          mph: 88
      to_date: '26.10.1985'
```