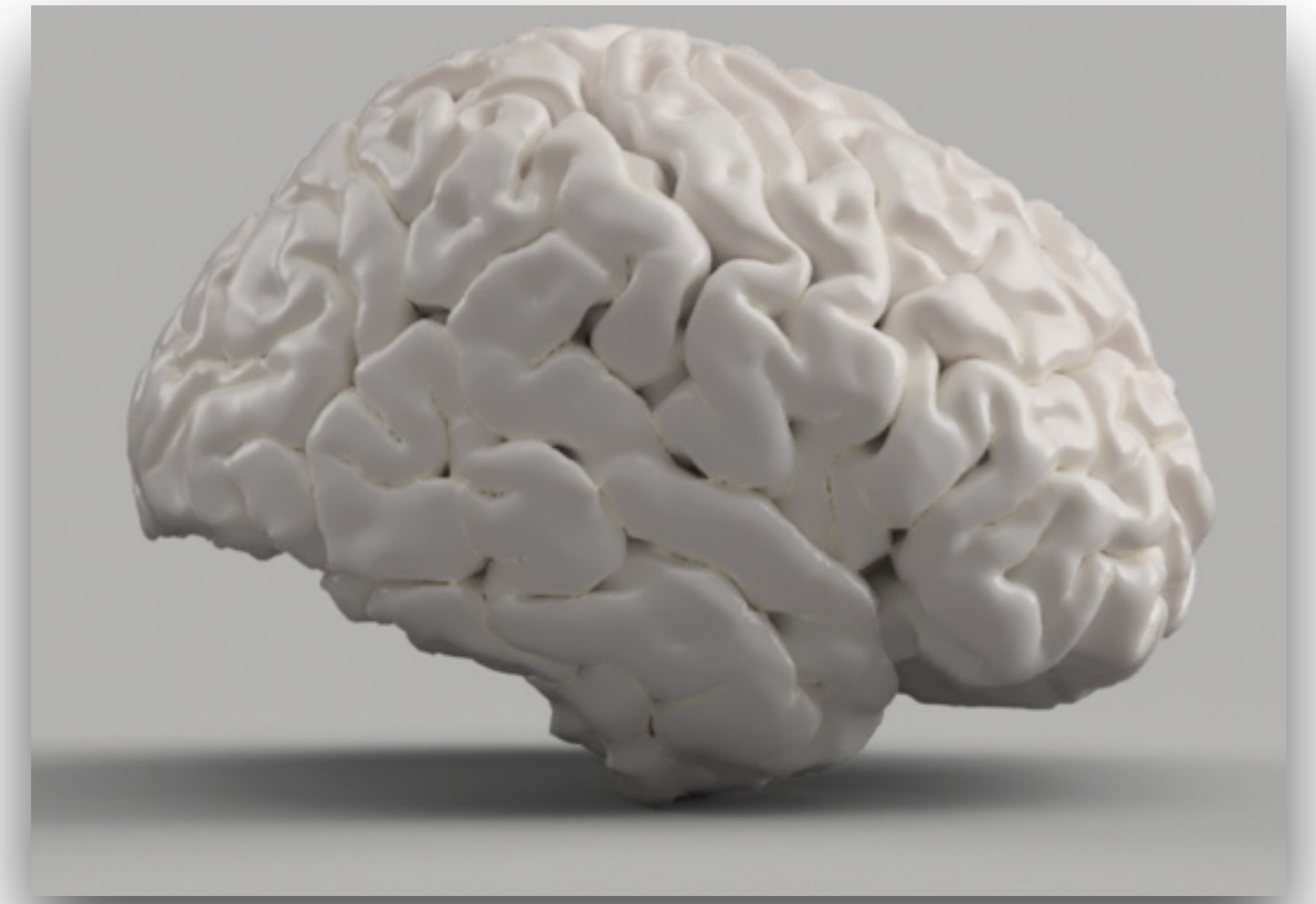
COGS109: Lecture 3



Data I/O, python and matlab syntax, basic visualization

July 11, 2023

Modeling and Data Analysis

Summer Session 1, 2023

C. Alex Simpkins Jr., Ph.D.
RDPRobotics LLC | Dept. of CogSci, UCSD

Plan for today

- Announcements
- Review of last time
- Data files, loading, saving, binary and ASCII, structure and form
- Python implementation and matlab/octave mentions for comparisons
- Data cleaning, wrangling, and the notion of ‘tidy’ data

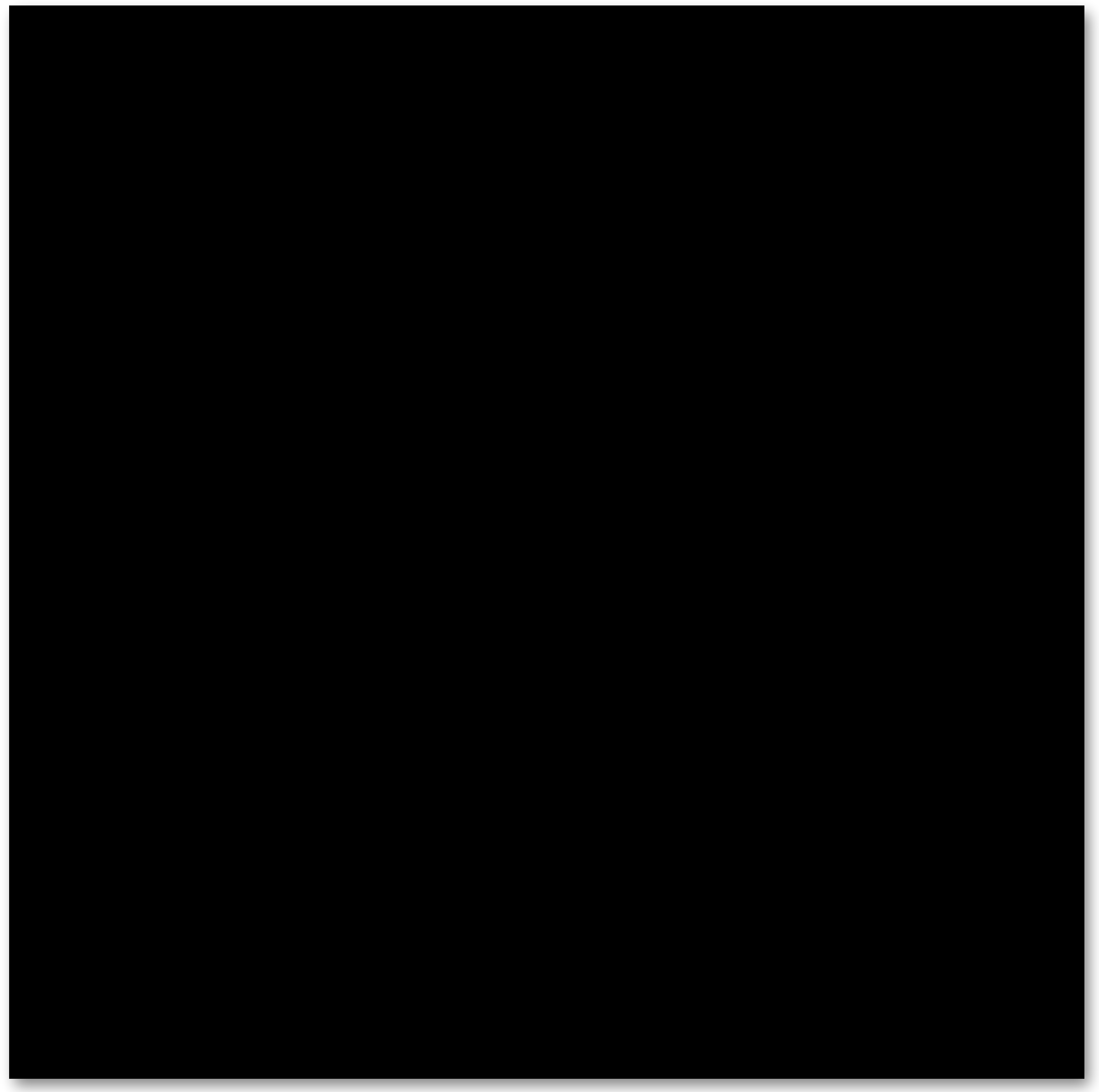
Announcements

- Groups
  - https://canvas.ucsd.edu/courses/47870/modules
  - Check the module link to the google doc, you can edit
  - If you need to find a group, we need to resolve today because of upcoming group work
- Q1 - Due this Friday (7/14)
  - Open notes open book
- D1 - Due this Friday (7/14)
- Group paper review due this Friday (7/14)
- D2 - Due next Friday (7/21)

1.23456789

About data files

- Extension of what we talked about last time
- Data must be recorded and stored for later retrieval
- Information must be ‘represented’ in some digital form
- Data files are a way to do this
- We can store all types of information in this way
  - Images
  - Numbers
  - Sounds
  - Video
  - etc

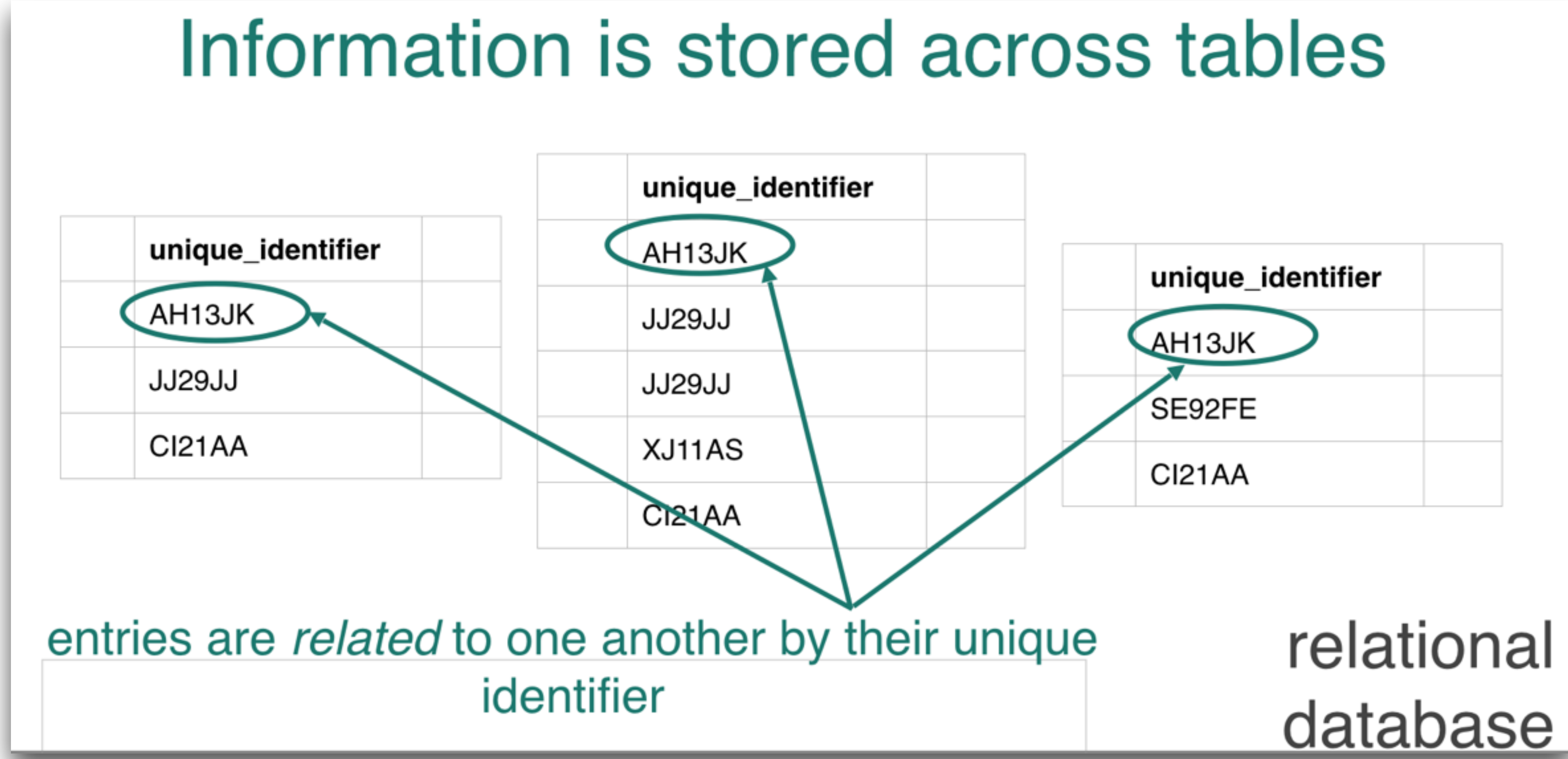


Quick review- Data structures

- Structured data
- Semi-structured data
- Unstructured data

Structured data

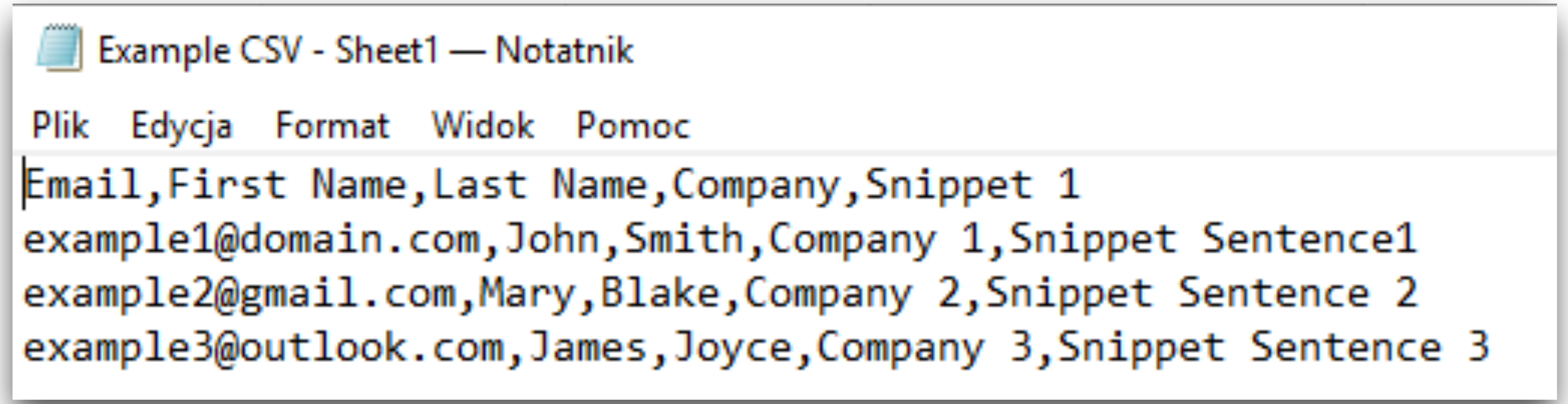
- Can be stored in database SQL
- Tables with rows and columns
- Requires a relational key
- 5-10% of all data



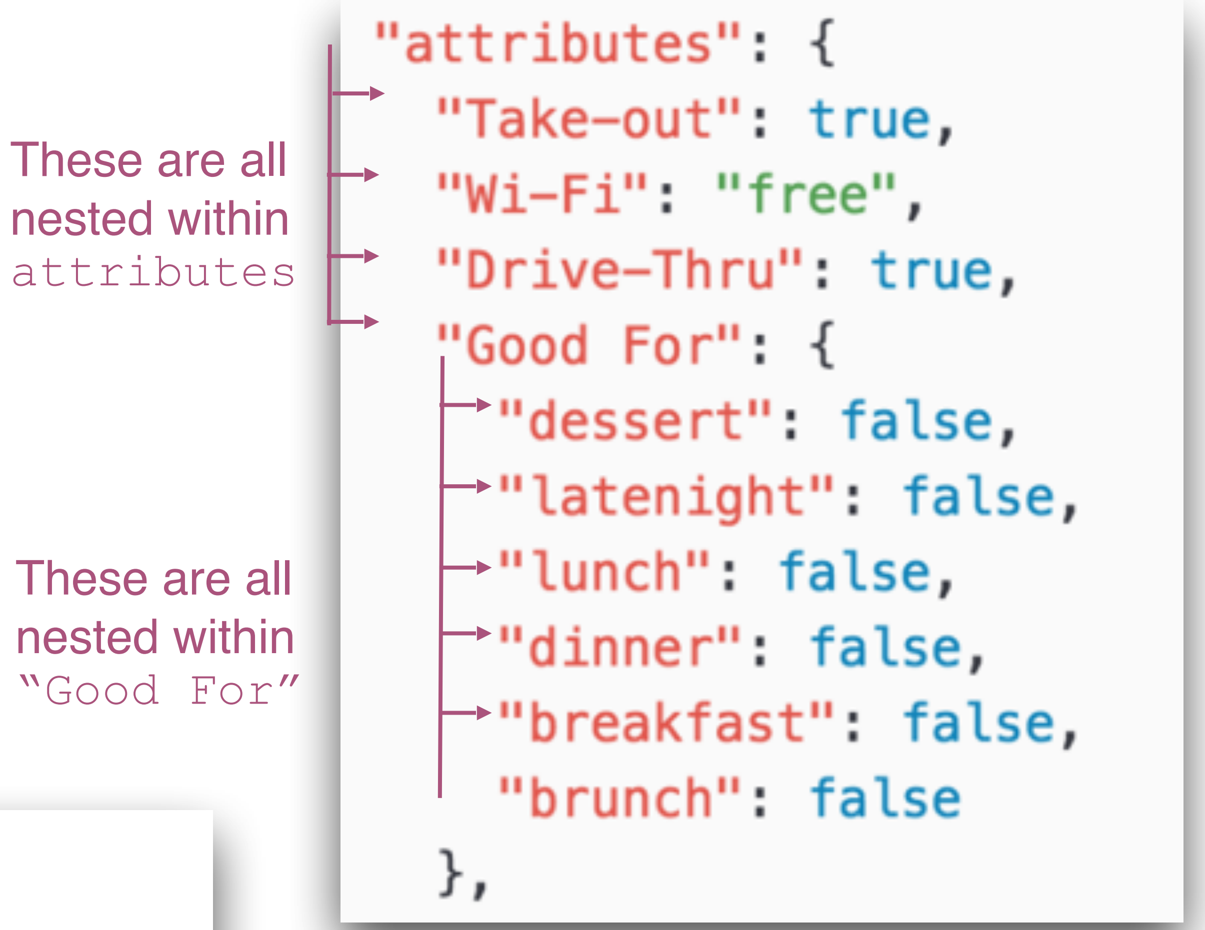
Semi-structured data

- Doesn't reside in a relational database
- Has organizational properties (easier to analyze)
- CSV, XML, JSON

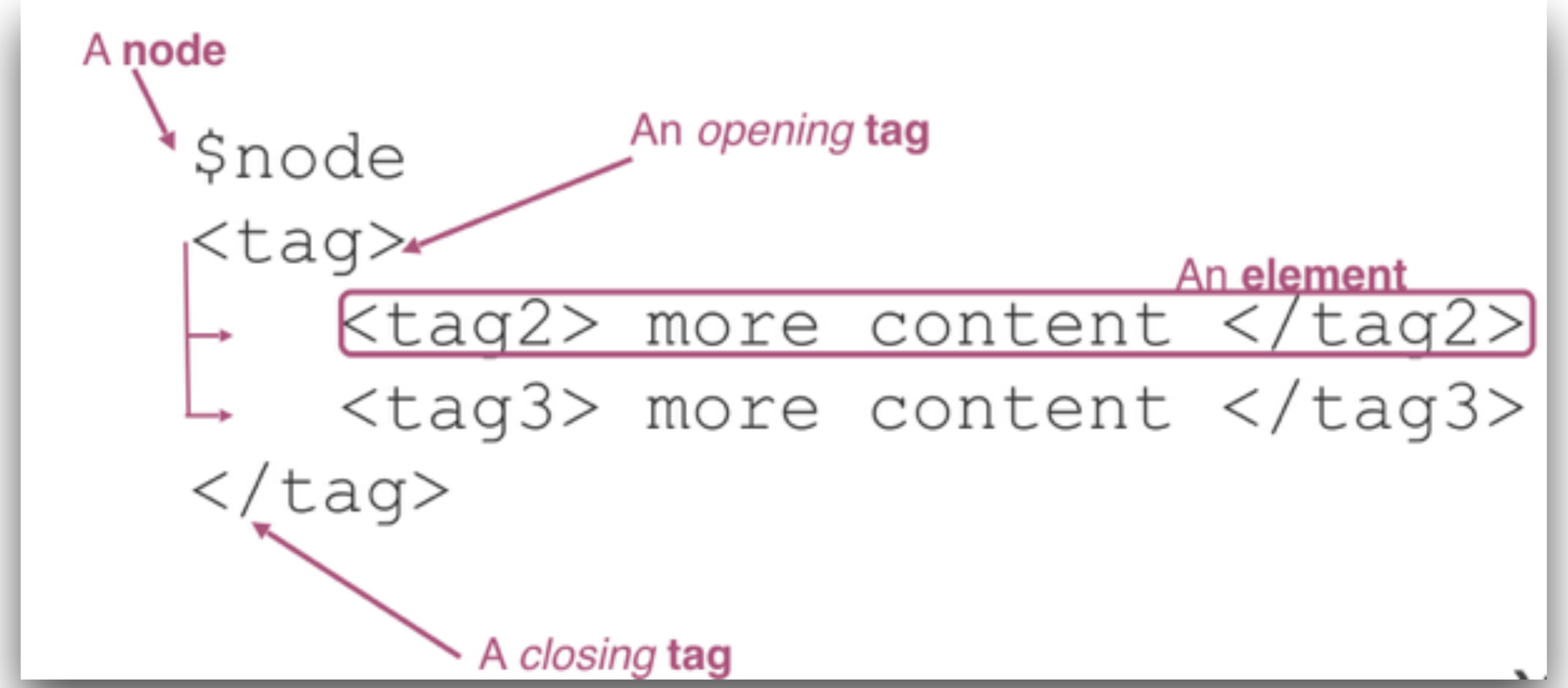
CSV



JSON

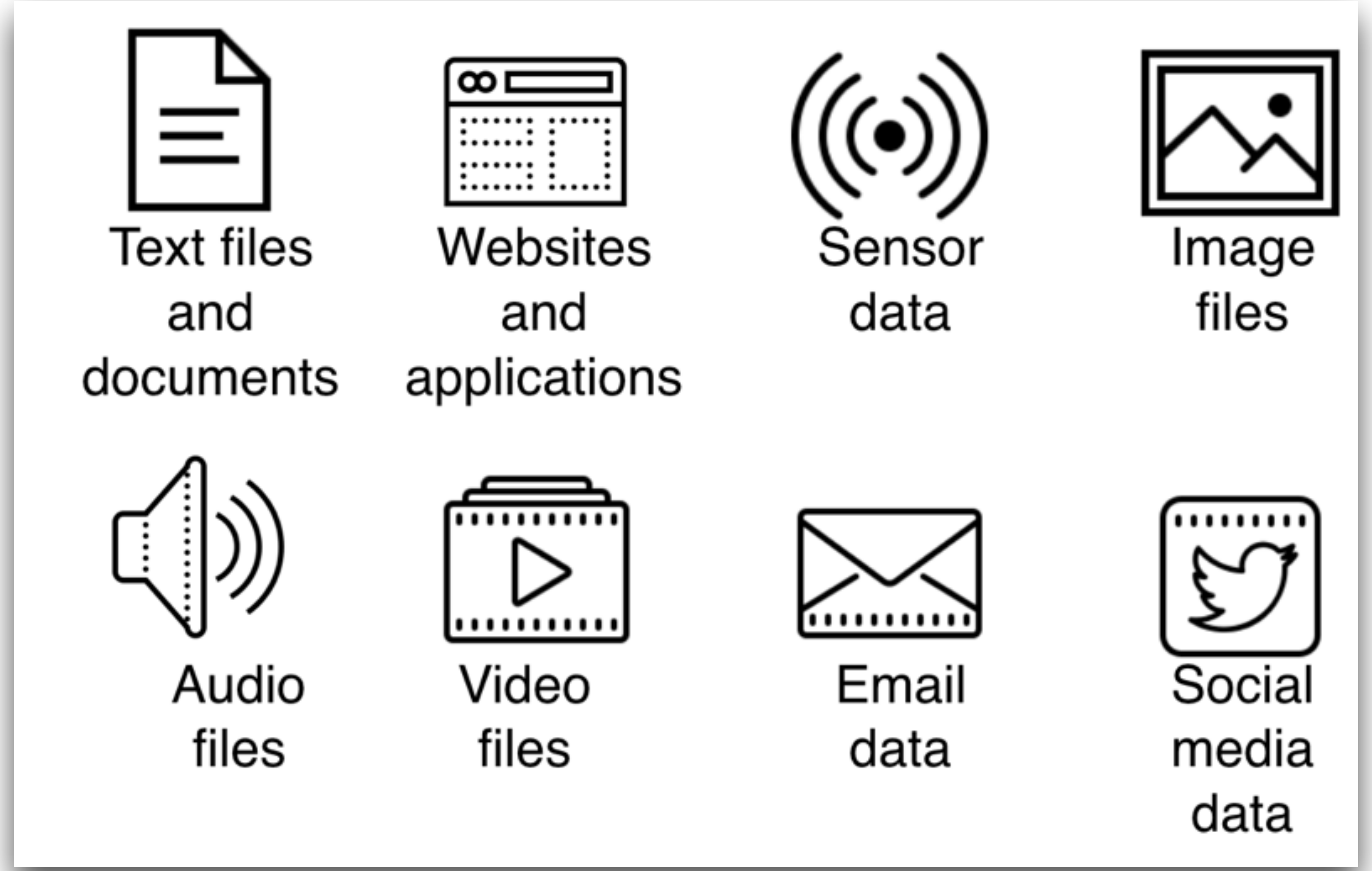


XML



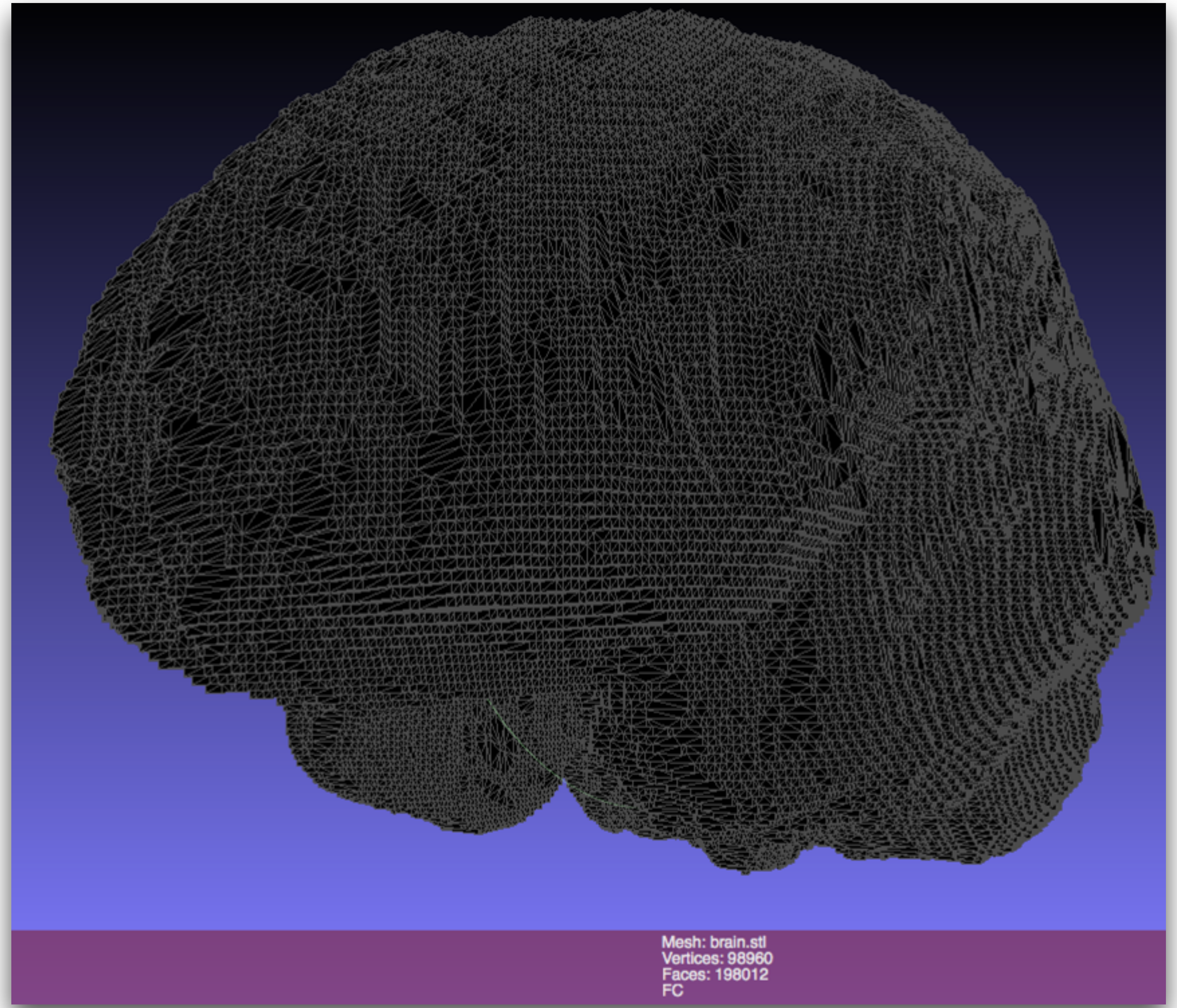
Unstructured data

- Non-tabular data
- 80% of the world's data
- Images, text, audio, videos



Types of data files (low level format)

- But how do we encode files in 1's and 0's?
- Files can typically be classified into two different formats
  - ASCII ("Text")
  - Binary
- STL example
  - Brain.STL (ASCII - 52MB)
  - Brain.STL (BINARY - 9MB)



ASCII Files

- “American Standard Code for Information Interchange”
- Any word processor, straight text
  - py-files and M-files are ASCII text files, so any word processor can create them,
    - As long as you are saving as ASCII text
    - Word, pages, google docs have their own format, but can create ASCII text files

ASCII Files (II)

- Python, Matlab, C, JAVA, etc can load and save specialized data files and standard text files (look for the extension on the end of the file name, ie “demo.m,” “data.dat,” “data.txt,” “demo.py”)
- Often you will be dealing with data, either in survey format, or in files which come from data acquisition systems (stored in text or binary files), sound or video files, etc.

Decimal - Binary - Octal - Hex – ASCII
Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	“	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	‘	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

source: https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-3-5-ascii.html

How to load text files in Python

- Depends on the file type (generic or specific)
- https://www.geeksforgeeks.org/reading-writing-text-files-python/
- Python can load either generic text or binary files
- **`open()`** function
- No special module needed
- very similar to C
- Add that ‘r’ if the file is not in the same folder as the script/current directory in order to make the string raw and avoid processing special characters

```
File_object = open(r"File_Name","Access_Mode")
```

Loads a file into ‘primary memory’ or RAM

- Secondary memory is your nonvolatile storage
- If successful, returns a file ‘handle’ that allows you to then access that memory
- Pay attention to the mode it is opened in (r, r+, w, w+, a, a+)
- other operations:
  - file_handle.close()
  - file_handle.write()
  - file_handle.read()
  - more…(e.g. https://www.geeksforgeeks.org/reading-writing-text-files-python/)

```
# Open function to open the file "MyFile1.txt"
# (same directory) in append mode and
file1 = open("MyFile1.txt","a")

# store its reference in the variable file1
# and "MyFile2.txt" in D:\Text in file2
file2 = open(r"D:\Text\MyFile2.txt","w+")
```

File access modes using open()

1. **Read Only (‘r’) :** Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exists, raises the I/O error. This is also the default mode in which a file is opened.
2. **Read and Write (‘r+’):** Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. **Write Only (‘w’) :** Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. **Write and Read (‘w+’)** : Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. **Append Only (‘a’)**: Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. **Append and Read (‘a+’) :** Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.

Source: https://www.geeksforgeeks.org/reading-writing-text-files-python/

Load various files using Pandas

- https://pandas.pydata.org/pandas-docs/stable/reference/io.html
- **pd.read_csv('data.csv')**
- **pd.read_table(‘data.csv’)**
- A bit simpler?

How to load text files in Matlab/Octave

- *Import wizard* menu (also works for matlab binary files)
  - Demo
- *M=xlsread('filename')*
  - Reads an excel spreadsheet file and stores it into a matrix of your choosing (here it's *M*)
- *Load filename .ext*
  - loads the data in the ASCII text file *filename.ext (*where *.ext* is the extension of the filename, such as .txt)
- Octave documentation: https://docs.octave.org/latest/Simple-File-I_002fO.html

Saving files in ASCII format (with Python)

To open a file for writing, use:

```
f = open('data_new.txt', ‘wb')
```

Then simply use `f.write()` to write any content to the file, for example:

```
f.write("Hello, World!\n")
```

If you want to write multiple lines, you can either give a list of strings to the `writelines()` method:

```
f.writelines(['spam\n', 'egg\n', 'spam\n'])
```

or you can write them as a single string:

```
f.write('spam\negg\nspam')
```

To close a file, simply use:

```
f.close()
```

- *https://python4astronomers.github.io/files/asciifiles.html*

Writing files with Pandas

- https://realpython.com/pandas-read-write-files/
- Series and DataFrame objects have write data methods to write to the clipboard or to a file
- Naming convention is **DataFrame.to_<file-type>()**
  - **file-type is the type of target file**
  - **.to_csv(), .to_excel(), .to_json(), .to_html(), .to_sql(), .to_pickle(), and more!**
  - **e.g. df.to_csv('data.csv')**
- **Precision example**

Saving files in ASCII format (with Matlab)

- *Save filename -ASCII*
  - Saves files in ASCII single precision format
  - Numbers are represented by 1.249E+002 format for 1.249x10^2
  - `The range for single is:`
    - `-3.40282e+038 to -1.17549e-038 and`
    - `1.17549e-038 to  3.40282e+038`
- *Save filename -double*
  - *Double precision format*
  - *1.249D+002*
  - `The range for double is:`
    - `-1.79769e+308 to -2.22507e-308 and`
      `2.22507e-308 to  1.79769e+308`
- *Dlmwrite('my.data.out',data, ';')* Delimited files, data separated by some character

ASCII data files

- Typically (semi-structured) have a header of text
- Then columns of data representing each variable

```
subject #: 18 Date: 6-6-2006 23:20

===settings===

  database name M, # of files: 50, stereo: 1, scale: 0.603774, full screen: 1, vision: no

121.0000   119.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   152.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   84.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
121.0000   118.0000   123.0000   151.0000   98.0000   84.0000   85.0000   77.0000   83.0000   85.0000
```

Binary files and .mat files

- A more efficient way to store files generally is binary format
  - Smaller
  - But…Less platform independent - ie need to know exactly what the format is to read the file
  - Can’t load these files into just any text editor like you can with ASCII
  - Image files are examples of binary
  - Matlab stores a binary format with the extension .mat
  - Python and Pandas can read/write binary files fairly simply as well
    - Have to choose carefully what techniques you use - with large files the slower approaches might not work due to being too slow or memory intensive

Loading binary files in Python

- One approach (https://stackoverflow.com/questions/16573089/reading-binary-data-into-pandas)
- There are many ways to do this
  - Often you will work with standardized formats or formats that provide tools if from a commercial system
  - Not always
  - Knowing how binary files and text files work as well as both simplifying functions and low level python functions allows you to work with anything

Non-standardized binary data

- So some file you know the structure
  - Data acquisition, image file (there would be a module normally though), other arbitrary type
  - **Need documentation for how the bytes encode the data**
  - Typically either just a sequence of numbers and you have to know the order or…
  - A file with a header then body, header specifies the rest
  - A series of records consisting of a header (identifying info) and the record one after the other
  - https://towardsdatascience.com/loading-binary-data-to-numpy-pandas-9caa03eb0672

Saving binary files in Python

- https://pandas.pydata.org/docs/user_guide/io.html

Loading binary files in Matlab

- *Load filename*
  - Loads all workspace variables from the file filename.mat
  - They appear as the same names of variables as when they were stored in the file

Saving binary .mat files in Matlab

- *Save filename*
  - Saves all the workspace variables in the file filename.mat
  - Saves in the current workspace directory
- *Save filename variable1 variable2…*
  - Saves only the variables you choose from the workspace into the file

Matlab/octave vs. Python-Pandas

- Matlab is sometimes simpler, Pandas sometimes simpler
- Use the tool for your application
- Matlab/Octave tends to be very simple if you stick with their proprietary binary format, but it is therefore limiting
- As usual with python, there's a module for that!
    - https://stackoverflow.com/questions/38197449/matlab-data-file-to-pandas-dataframe
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.loadmat.html

```
import scipy.io as sio
test = sio.loadmat('test.mat')
```

Python syntax demo

- https://pandas.pydata.org/docs/user_guide/io.html
- So we are not going to provide an exhaustive list of functions here, but show you what different file types are and how to use the documentation to accomplish what you need to
- Our exercises in upcoming workbooks will be taking you through examples loading various types of data and performing basic visualization

Tidy Data

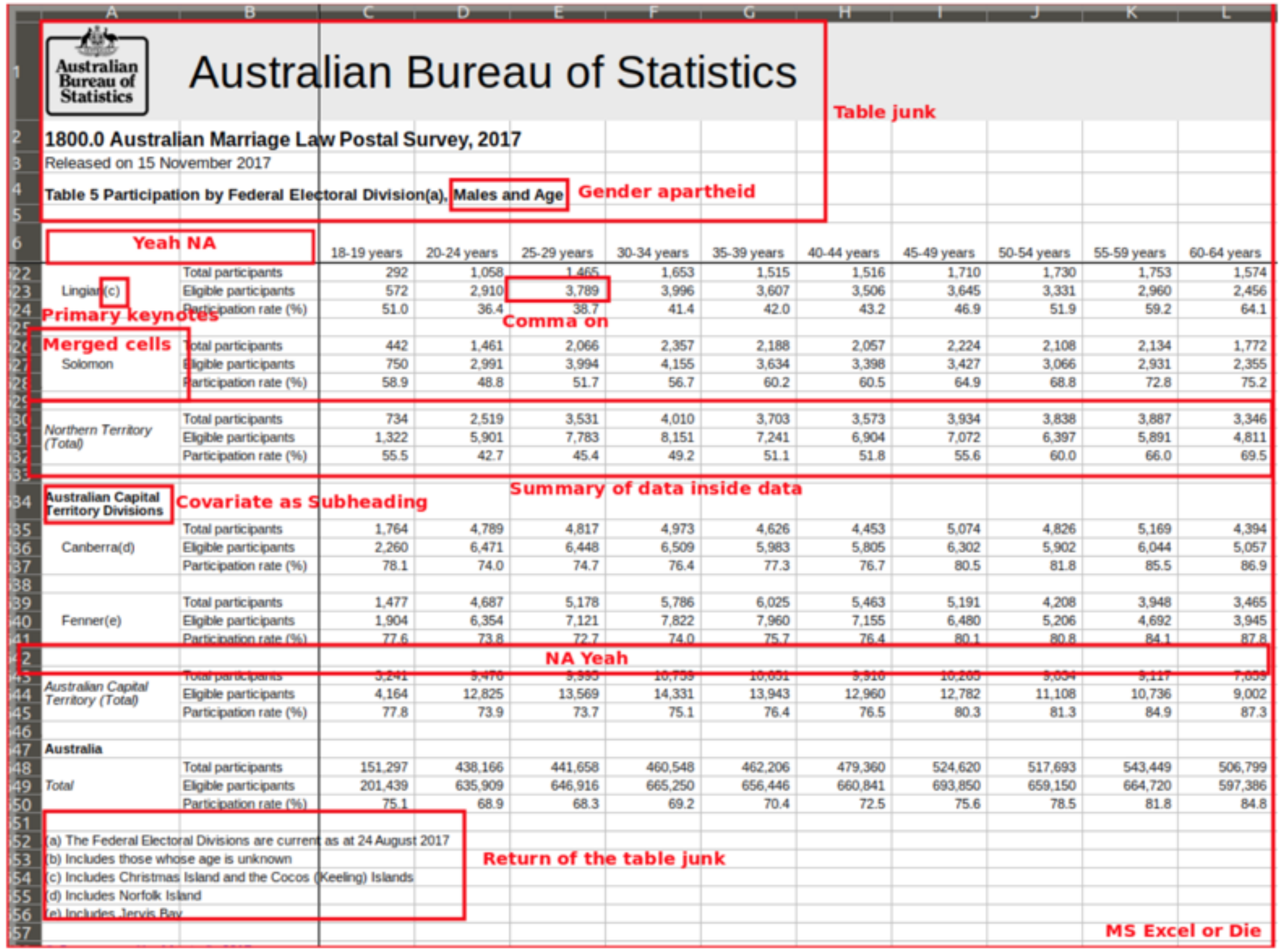
Making data usable

Image Source: https://www.yahoo.com/entertainment/star-wars-turns-40-dirty-secrets-trash-compactor-scene-144820052.html

Data wrangling

- The process of restructuring a dataset from whatever form it is initially in to a computationally usable form suitable for data science
- Generally a large amount of effort goes into change data from raw recordings to a usable form

untidy data



data wrangling

tidy data

	area	gender	age	State	Area (sq km)	Eligible participants	Participation rate (%)	Total participants	Total Paticipants
2	Adelaide	Female	18-19 years	SA	76	1341	83.5	1120	1120
3	Adelaide	Female	20-24 years	SA	76	4620	81.2	3750	3750
4	Adelaide	Female	25-29 years	SA	76	4897	81.8	4004	4004
5	Adelaide	Female	30-34 years	SA	76	4784	79.8	3820	3820
6	Adelaide	Female	35-39 years	SA	76	4319	79	3411	3411
7	Adelaide	Female	40-44 years	SA	76	4310	80.6	3472	3472
8	Adelaide	Female	45-49 years	SA	76	4579	81.4	3728	3728
9	Adelaide	Female	50-54 years	SA	76	4475	84.7	3791	3791
10	Adelaide	Female	55-59 years	SA	76	4622	87.3	4033	4033
11	Adelaide	Female	60-64 years	SA	76	4342	89.3	3879	3879
12	Adelaide	Female	65-69 years	SA	76	3970	90.7	3602	3602
13	Adelaide	Female	70-74 years	SA	76	3009	90.3	2716	2716
14	Adelaide	Female	75-79 years	SA	76	2156	88.5	1908	1908
15	Adelaide	Female	80-84 years	SA	76	1673	85.1	1423	1423

https://medium.com/@miles.mcbain/tidying-the-australian-same-sex-marriage-postal-survey-data-with-r-5d35cea07962

Tidy Data

1. Each variable you measure should be in a single column

	A	B	C	D	E	F	G
1	ID	LastName	FirstName	Sex	City	State	Occupation
2	1004	Smith	Jane	female	Frederick	MD	Welder
3	4587	Nayef	Mohammed	male	Upper Darby	PA	Nurse
4	1727	Doe	Janice	female	San Diego	CA	Doctor
5	6879	Jordan	Alex	male	Birmingham	AL	Teacher

2. Every observation of a variable should be in a different row

	A	B	C	D	E	F	G
1	ID	LastName	FirstName	Sex	City	State	Occupation
2	1004	Smith	Jane	female	Frederick	MD	Welder
3	4587	Nayef	Mohammed	male	Upper Darby	PA	Nurse
4	1727	Doe	Janice	female	San Diego	CA	Doctor
5	6879	Jordan	Alex	male	Birmingham	AL	Teacher

3. There should be one table for each type of data

Demographic Survey Data

	A	B	C	D	E	F	G
1	ID	LastName	FirstName	Sex	City	State	Occupation
2	1004	Smith	Jane	female	Frederick	MD	Welder
3	4587	Nayef	Mohammed	male	Upper Darby	PA	Nurse
4	1727	Doe	Janice	female	San Diego	CA	Doctor
5	6879	Jordan	Alex	male	Birmingham	AL	Teacher

Doctor's Office Measurements Data

	A	D	E	F	G
1	ID	Height_inches	Weight_lbs	Insulin	Glucose
2	1004	65	180	0.60	163
3	4587	75	215	1.46	150
4	1727	62	124	0.72	177
5	6879	77	160	1.23	205

4. If you have multiple tables, they should include a column in each *with the same column label* that allows them to be joined or merged

	A	B	C	D	E	F	G
1	ID	LastName	FirstName	Sex	City	State	Occupation
2	1004	Smith	Jane	female	Frederick	MD	Welder
3	4587	Nayef	Mohammed	male	Upper Darby	PA	Nurse
4	1727	Doe	Janice	female	San Diego	CA	Doctor
5	6879	Jordan	Alex	male	Birmingham	AL	Teacher

	A	D	E	F	G
1	ID	Height_inches	Weight_lbs	Insulin	Glucose
2	1004	65	180	0.60	163
3	4587	75	215	1.46	150
4	1727	62	124	0.72	177
5	6879	77	160	1.23	205

Tidy data == rectangular data

A

	A	B	C	D	E
1	id	sex	glucose	insulin	triglyc
2	101	Male	134.1	0.60	273.4
3	102	Female	120.0	1.18	243.6
4	103	Male	124.8	1.23	297.6
5	104	Male	83.1	1.16	142.4
6	105	Male	105.2	0.73	215.7

Broman KW, Woo KH. (2017) Data organization in spreadsheets. *PeerJ Preprints* 5:e3183v1 https://doi.org/10.7287/peerj.preprints.3183v1

Tidy Data Benefits

1. Consistent data structure
2. Foster tool development
3. Require only a small set of tools to be learned
4. Allow for datasets to be combined

Data Intuition

1. Think about your question and your expectations
2. Do some Fermi calculations (back of the envelope calculations)
3. Write code & look at outputs <- think about those outputs
4. Use your gut instinct / background knowledge to guide you
5. Review code & fix bugs
6. Create test cases - “Sanity checks”

What is data cleaning?

- Fixing/removing incorrect, corrupted, incorrectly formatted, duplicate, incomplete, data within a dataset
- Many issues combining data sources and types, researcher styles, standards, recording errors, etc

Consequences of poorly cleaned data

- Unreliable outcomes and algorithms
- Difficult to detect these issues
- Biased results
- Failure to process algorithms (for example NANs causing errors)

Variability in cleaning

- There is no one process to clean data
- Varies from set to set, project to project, software to software
- But can establish a ‘template’ procedure/process of ‘check-offs’ to make sure you’ve done your best to address it

Methods can be

- Interactive through ‘wrangling tools’
- Automated through scripts, programs or other software (batch processing)

Data wrangling vs. data cleaning

- Data wrangling focuses on transforming the data from a ‘raw’ format into a format suitable for computational use
- Data cleaning focuses on, as discussed, fixing/removing incorrect, corrupted, incorrectly formatted, duplicate, incomplete, data within a dataset

Let’s take a break and come back in 10min