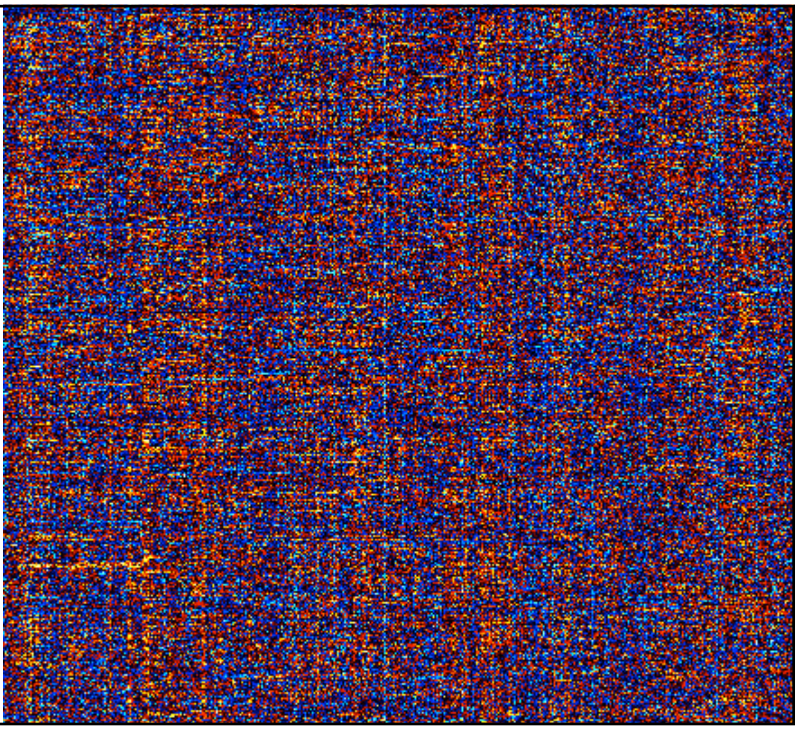
BIG DATA ISSUES

1

Big data comes at a price.

There are challenges…



2

Data format issues

0 0 1 0 1 0 1 1

- Some MATLAB data types: double (8 bytes), single (4 bytes), int16 (2 bytes), logical (1 byte)
- (Most) computation needs to be done in double or single, but to save memory and/or disk space, we can consider storing data in smaller formats
- Be aware of variables that are potentially huge; when saving to disk, consider casting to a small format

3

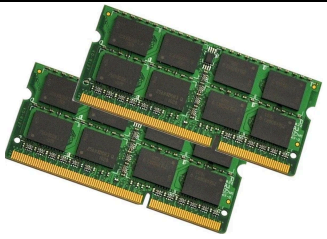
Data format issues

0 0 1 0 1 0 1 1

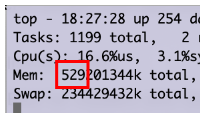
- Basic dilemma for file formats:
  Compression reduces file size (e.g. this is the default when using save –v7.3 for .mat files) but this costs computational time when saving and loading.
  - E.g., there is overhead when loading .nii.gz
  - Note that some data are highly compressible (e.g. ROIs)
- Nowadays, disk space is generally "cheaper" than computational time, so consider saving large data in uncompressed format?

4

RAM/memory



- Typical computers have 8–16 GB. This is not a lot.
- Need enough RAM to hold data and to compute on it
- If data grow too big to fit into RAM, need to chunk the analysis (load some data, compute, save results, clear, and repeat)
- In MATLAB, can use ‘whos’ to monitor usage (also see checkmemoryworkspace.m)
- Can use ‘top’ (or Activity Monitor) to monitor RAM on the entire computer.
- Hitting swap (i.e. requiring the OS to offload memory to disk) is likely a death knell. ☠️
- If money is no object, buy lots of RAM



```
top - 18:27:28 up 254 d
Tasks: 1199 total,   2
Cpu(s): 16.6%us,  3.1%s
Mem:  529201344k total,
Swap: 234429432k total,
```

5

Disk space

- Disk space is cheap. Buy lots.
- Type of disk (SSD vs. HDD) [Tradeoff speed vs. cost]
- If certain files are accessed very often, consider storing them on a fast device
- Disk access is time-consuming. Avoid writing and reading unneccessarily.
- It is generally faster to consolidate data into a small number of files compared to having to access a large number of files.
- Try to load only the data you need
  - For example: load(‘test.mat’,’var1’)
  - For example: HDF5 format and random access

6

Execution time

- Many MATLAB operations are automatically multithreaded
- To speed things up, consider:
  - Opening multiple MATLAB sessions
  - Using parallel computing (parfor)
  - Farming the code to a cluster
  - Implementing code on GPUs
  - Writing more efficient code
- MATLAB profiler is extremely useful to isolate slow code
- Vectorization is good; for-loops are bad
- In general, DO NOT optimize until it becomes a problem: human time is more expensive than computer time.

7

Network issues

- If the data live on a server, network speed to the computer performing the analysis is a potential bottleneck when loading or saving
- Consider performing expensive computations on the machine that has direct access to the data

8

Miscellaneous ideas

- Carefully test code on small data (e.g. one subject, one session) before deploying at scale
- Separate loading from analysis (this way, you can load once and then use trial-and-error to develop the analysis)
- Cache computationally expensive results
- The larger the data, the more costly coding errors are. (The roundtrip between developing and seeing results takes more and more time.) Thus, it is important to develop coding proficiency.