# 1 Data Storage and I/O

## 1.1 Data storage and I/O

## 1.2 File Formats

- Binary Formats
  - Numeric values stored encoded in binary representation
    - 32 bi float
    - 16 but integer
  - Properties
    - Compact
    - High Performance I/O
    - Not Human Readable
    - Worry about int size, endiness, un/signed
- Plain text formats (ASCII/Unicode)
  - Numeric values encoded as ASCII/Unicode strings
    - float $\rightarrow$ "3.1415926"
    - int $\rightarrow$ "44"
  - Peroperties:
    - Human Readable
    - Self-documenting
    - Slower I/O
    - Less compact
- Choice depends on app
  - Smaller files $\rightarrow$ Plain Text
  - Larger files $\rightarrow$ Binary

## 1.3 ASCII Formats for Tabular Data

- Two common formats:
  - CSV: Comma Separated Values
  - FWF: Fixed Width Format

## 1.4 CSV

- Plain txt format
- Rows on Lines
- Column sep by commas
- Not only commas (Tabs: TSV)

- Strins Containing commas are quoted
- Advantages
  - Read by multiple apps e.g. Excel
  - Fast to parse/generate
  - can be compressed (.csv.gz)
  - Do not need to load all into mem. (Streamable)
- Disadvantages:
  - Not standardised
  - Bulkier than binary formats (esp. if not compressed)
  - No types

## 1.5 FWF

- Plain text format
- Rows on Lines
- Columns of fixed width (fixed no. of chars)
- Colum,ns padded using padding char (usually space)
- Advantages:
  - Easier to read in plain txt
  - Can be read by most apps
  - Fast to parse
  - Can be compressed (.fwf.gz)
  - Streamable
- Disadvantages:
  - Not standardised
  - Bulkier than CSV (padding chars)
  - Need to establlish field width before can write first row
  - No types

## 1.6 Binary formats for tabular data

Binary formats are more compact and performant. Not human readable!

Common binary formats:

- HDF5
- MATLAB
- NumPy
- Apache Arrow and Feather
- Excel

## 1.7 HDF5

- Hierarchical Data Format Version 5
- Industry standard for numeric structured tab. data
- Used in scientific community
- Can store mult. datasets in single file
- organize datasets into hierarchical struct. like filesys
- Metadata using ättributes"
- Memory mapping so entire dataset not need to be loaded into memory. Mem. can be shared between processes
- Library suppor in many langs.
- Advantages:
  - High performance
  - Compact
  - Mult. datasets in one file
  - Include type info
  - Metadata
  - Parallel reads, share mem
  - Optional compression
- Disadvantages:
  - Not human readable
  - Can be cumbersome when table size not known in advance

## 1.8 MATLAB Files

- Proprietary format used by MATLAB
- Common for data exchange in industry/academia
- Store mult named arrays and various other structs
- Newer versions of MATLAB (2006+) use HDF5

## 1.9 NumPy

- NumPy has builtin support for 2 bin formats
  - .npy contains single numpy arrays
  - .npz contains multiple arrays
- Format supports compression
- Advantages/Disadvantages

  - Fast, compact
  - Supports mem. mapping
  - Built in support for numPy
  - No metadata
  - Not standardized
  - Main use - store temporary results

## 1.10 Excel Format

- Common proprietary format used in Excel
  - XLS/XLSX files
- Advantage:
  - Works with MS products
  - Keeps formatting (colours etc.
- Disadvantages:
  - Keeps formatting
  - Not portable
  - Proprietary

## 1.11 Common formats for structured data

More complex structures than plain tables (although can be reduced to tables)
Structure: encode data types, attributes, relationships, hierarchies.

Common plain-text formats for structuresd data:

- XML
- JSON
- YAML

Binary formats:

- MessagePack
- Google Protobuf

## 1.12 XML

- Extensible Markup Language
- Format for structured data - Human and Machine readable
- Self describing - No additional documentation
- Hierarchy made up of
  - Elements

  - Attributes
  - Content
- Advantages:
  - Self-Describing
  - Human Readable
- Disadvantages:
  - High Overhead
  - Verbose
  - Slow parsers
  - Not well duited to big datasets
  - XML namespaces are nightmare
  - Seperate validation and DTD
  - Painful to manually type

### 1.13 JSON

- JS Object Notation
- Subset of JS for desc. data
- Value types:
  - String
  - Number
  - Boolean
  - Object
  - Array
  - Null
- Type implicit in syntax
  - 35 is a number
  - "35ïs a string
  - ä": 1 is an object

1,2,3 is an array

- Advantages:
  - Has types
  - Hierarchical
  - Human readable
  - Self-describing
  - Easy to write by hand

  - More compact
  - Faster to parse
  - Most languages have parsers
- Disadvantages:
  - Some overhead
  - Not well suited to big datasets

## 1.14 YAML

- YAML Ain't Markup Language
- Like JSON but easier to write
- Very useful for config/meta files
- Slower than JSON

## 1.15 MessagePack

- Binary encoded JSON
- More compact than JSON
- Advantages
  - Compact
  - High I/O performmance
  - Good for network I/O and DBs
  - Good for bigger data
  - Streamable
- Disadvantages:
  - Not Human readable
  - Less suited to large numerical arrays

## 1.16 Google Protobuf

- Serializing data over the wire, can be used as file format
- Defines protocol specification language (prototxt) and wire format
- Used for model spec. and distribution by well-known deep learning library developed by Berkeley

## 1.17 Things to consider when choosing a data format

- I/O performance
- Structure support
- Streamable
- Scalability
- Appendable
- Portability
- Compactness
- Metadata
- Type support
- Readability
- Write by hand

## 1.18 Databases

- Common for data acq. and storage
- Advantages
  - Network Access
  - concurrency
  - Enforced consistency
  - Fast indexes
  - Query languages
- Not efficient for large binary data
- Two important types:
  - Relational Databases (SQL)
  - NoSQL Databases

# 2 Data Wrangling

## 2.1 Real-World Datasets

Real world datasets often "dirty": messy, inconsistent sources of problems

- Use incomplete standards
- Manyal entry errors
- Measurement errors

- Inconsustent notations
- Redundancies and duplicates
- Missing values
- E.G:
  - Headers needing to be removed
  - Blank Lines
  - Missing Column names
  - Subheadings embedded in table
  - Hierarchy of categories implied using whitespace

## 2.2 Data Wrangling

- Process of transforming "raw"data into data that can be analysed to generate actionable insights
- AKA:
  - Preprocessing
  - Munging
  - Cleaning
  - Scrubbing
  - Preparation
  - Transformation

## 2.3 Typical tasks

- Fixing ugly/broken formats
- Handling missing values
- Remoiving redundant attributes and records
- Fixing inconsitencies
- Shaping data
- Fusing data sources
- Scraping and gathering data from external sources
- Extracting info from unstructured sources

## 2.4 Ugly and Broken Formats

- Examples:
  - Badly formatted tables
  - Broken XML/JSON (Syntax Errors)
  - Hand entered data with syntax errors
  - Log files with strange formatting
- First Step: Transform to more machine friendly parsable format
- Toolbox:
  - Python csv module
  - Text editor like Vim
  - Custom parsing scripts
  - RegEx
  - Tabular
  - Pandas

## 2.5 Missing Values

- Common, can have significant effects on analysis and conclusions
- Causes
  - Non-response
  - Unobserved or unknown values
  - Sensor or measurement errors
  - Censorship
  - Erros in data collection or data entry
- Often show up in datasets as
  - Special NA values
  - NaN
  - null or None
  - Sentinel values (e.g. age == -1)
  - Blanks
- Three types:
  1. Missing completely at random (MCAR): missing values random;ly distributed for all observations
  2. Missing at random (MAR): Prob. of value being missing depends on other observed variables
  3. Missing not at random (MNAR): Prob. of value being missing depends on value of missing variable or another unobserved variable

  - MCAR and MAR assumption is common
  - If addunp. made when dealing with missing values, make them explicit

## 2.6 Strategies for handling missing values

- Three common approaches:
  1. Ignore
     - Drop missing values when computiong summary stats. (mean, variance)
  2. Remove
     - If plenty of data available, may be possible to ignore rows with missing values
  3. Impute
     - Try to fill in blanks
     - Common imputation techniques
       * Mean/mode sub
       * Predict from other attrib.
       * Interpolation (e.g. time series)
- May introduce bias for MNAR values

## 2.7 Redundant attributes

- For example:
  - Useless attribs
  - Duplicated attribs
  - Attribs easily derived from other attribs
- Can cause problems for some stat analysis
- Eliminate redundancy where possible

## 2.8 Inconsistent Categories (Nominal Attributes)

- E.G:
  - Misspellings
  - Inconsistent Spellings
  - Hyphenation
  - Inconsistent case
  - Inconsistent abbreviations
- Techniques:
  - Print unique vals and try to detect outliers and splits
  - Normalize case and spelling
- Tools:
  - Unix: sort | uniq
  - Py: sort, set()
  - str.lower, str.replace,
  - regex

## 2.9 Dates and Times

- Variation in ways can be represented
- Cleaning should be standardized to single format
- Preferably include timezone info.

## 2.10 Parsing dates and times

parsedatetime library will parse almost anything
Consider arrow library if a lot of date and time amnipulatio

## 2.11 Outlliers

- Data points extremely unlikely under data distribution
- Causes:
  - Measurement Error
  - Recording Error
  - Statistical anomalies
- Oftern want to identify prior to further analysis
  - Measure quantity of outliers
  - label outlier
  - Completely remove

## 2.12 Detecting Outliers

- Can detect by plotting data/visual inspection
  - Boxplots, jitter plots, histograms
- Alt. can make assumptions about distrib. of attrib. and find items unlikelyt under distrib
- e.g.assume data normally distributed
  - Estimate sample mean and std. dev from data, compute Z scores

$$z = \frac{z - \mu}{\delta} \mu = Samplemean \delta = samplestd.dev.$$

## 2.13 Data shaping

- Data often stored in "stacked/record format
- Sometimes more convenient to have one öbservation"per row with mult. attib.
- Achieved by reshapoing or pivot operations

## 2.14 Dealing wit unstructured data

So far looked at cleaning structured data.
Whar about semi/unstructured data

- Natural language plain text
- HTML files

Usually want to extract some features from data for analysis
One way of encoding text features: bag-of-words

## 2.15 Web scraping: getting data from webpages

- Idea: extract structured info from unstructured web pages by auto downloading and parsing

| Task | Issues | |
|---|---|---|
| Getting the data | HTTP requests, cookies, headers, download, JS, timing | wget, curl, requ |
| Figuring out which data to get | Link crawling and spidering | |
| Extracting structured info | Robust parsing, DOM querying | PyQuery, |

## 2.16 Web scraping

- wget, curl
  - Convenient cmd line tools
- requests
  - Easily make HTTP req. directly in Py
- mechanize
  - Stateful programmatic web browsing
  - Pretend to be a browser (cookies, headers, and all)
- selenium
  - Remote control an actual browser

## 2.17 PyQuery

- Parsing and querying HTML with PyQuery

## 2.18 Processing log files

- Good example of semi-structured data
- Log analysis: making sense and extracting info from log files
- Regex and string partitioning are useful tools:
  - Built-in python re module
  - Python string function
  - Unix tools: sed, awk, grep, vim

## 2.19 Data wrangling best practices

- Keep raw data separate from cleaned data
  - Never overwrite raw data
- Script data wrangling steps as much as possible
  - Need to change something later on, easier if steps are scripted
- Document all transforms, all assumptions made
  - Distribute documentation with cleaned dataset
  - Make wrangling process as reproducable as possible
- For large datasets, start with small random sample
  - Iterate faster: once perfected cleaning steps, apply to full datasets