STAT243 Lecture 2.1 Data storage and file formats on a computer

Logic

参考文献:

- Adler
- Nolan and Temple Lang, XML and Web Technologies for Data Sciences with R.
- Murrell, Introduction to Data Technologies.
- SCF tutorial: Working with large datasets in SQL, R, and Python

参考视频: bCourses Media Gallery 有 4 段 2020 年的参考视频 (使用 R):

- Text files and ASCII
- Encodings and UTF-8
- HTML
- XML and JSON

1 基本概念与术语

- **文件 (file)**：按字节顺序组织的数据集合，包含内容与少量元信息 (metadata)。
- **目录 (directory)**：文件的层级组织方式，路径使用分隔符表示层级。
- **路径 (path)**：
  - 绝对路径：从根目录开始，例如 `/home/user/data.csv`。
  - 相对路径：相对当前工作目录，例如 `../data/airline.csv`。
- **扩展名 (extension)**：约定俗成的类型标识，如 `.csv`，`.json`，`.parquet`，不决定编码方式但有提示作用。
- **MIME type**：互联网场景中文件类型的标准描述，如 `text/csv`，`application/json`。
- **bit / byte**：bit 为二进制位，取值 0/1；byte 为 8 bits。很多文件属性与编码的度量单位以 byte 计。

2 Text files vs Binary files

- **Text files**
  - 定义：文件中的比特编码为单个字符；数字以字符形式写入（例如由数字字符 0–9 组成）。
  - 例子：CSV, XML, HTML, JSON。
  - 编码：
    - ASCII：1 byte/char；最多 128 个字符（字母大小写、数字、标点与控制字符）。
    - UTF-8：1–4 bytes/char。
  - 优点：人可读、跨语言、易 diff。
  - 局限：
    - 某些格式（如 JSON、HTML）不易按“行”解释/操作，不适合用 shell 行处理工具；
    - 存储冗余、解析耗时；随机访问差。
- **Binary files**
  - 定义：比特用自定义二进制布局编码信息，而非直接对应字符；程序需按格式解释字节语义。
  - 例子：netCDF、Python pickle、R `.Rda`、HDF5、已编译代码等。
  - 性能：通常更节省空间且可随机访问；数值通常以 8 bytes/数存储（详见 Unit 8）。
  - 局限：不可读、跨语言互操作依赖库与规范。

3 Common file types

Logic

说明：此处既包含文本格式也包含二进制/容器格式。

1. **Flat text（定宽或分隔符）**
   - 一行一条记录；列为不同变量。
   - 常见分隔符：tab、逗号、空格、pipe `|`。
   - 常见扩展名：`.txt`，`.csv`。
   - 字段中若包含分隔符或换行，可使用引号包裹；`pandas.read_table()` 能处理带引号情况。
   - 换行差异与转换：
     - Windows 行尾 ，UNIX 行尾 。
     - 在 UNIX 上看到行尾 `^M` 表示混入了 DOS 行尾。
     - 可用 `fromdos` / `dos2unix` 转换为 UNIX；`todos` / `unix2dos` 转换为 DOS/Windows。
2. **逐行文本但无列含义**
   - 例如纯文本或部分生物信息学数据，一行仅是一段信息。

3. **数据交换格式**
   - XML, JSON 为自描述（元数据在文件中）。
   - Python 常用库：`lxml`，`json`（细节见后续章节）。

4. **HTML**
   - 常用于 web 抓取；结合 `requests`，`BeautifulSoup` 解析。

5. **netCDF / HDF5**
   - 科学计算常用；以变量保存多维数组，包含维度与元数据。
   - Python 可用 `netCDF4`。

6. **他软件数据格式**
   - Stata/SAS/SPSS；`pandas.read_stata` / `read_spss` / `read_sas` 可读入。

7. **Excel**
   - 不建议作为数据分发格式：
     - 专有且结构复杂、格式可能变化；
     - 行数等限制；
     - 难以用 UNIX 工具处理；
     - 常含多工作表、图表、宏等非数据内容。
   - 建议：在 Excel 导出为 CSV 再读入 Python。

8. **Databases**
   - Python 可与 SQLite, DuckDB, PostgreSQL, MySQL, Oracle 等交互。
   - 通过 SQL 查询并将结果返回至 Python（详见后续单元与 tutorial）。

4 CSV vs. specialized formats such as Parquet

- **CSV 优点**：简单、人可读、便于行处理工具。
- **CSV 劣势**：
  - 行存储类型混杂，压缩效果一般；
  - 明文存储分隔符与换行，体积偏大；
  - 随机访问不便：定位第 N 行需顺序扫描换行符（如找第 10 行需先找到第 9 个换行）。
- **Parquet 优势**：
  - 列式存储（列块），同列类型一致且重复多，便于编码与压缩；

- 可只读取所需列，I/O 更小，通常更快；常以多个文件分片存储。

- **示例（原文给出）**：
  - 读取 CSV ≈ 1.815 s；读取 Parquet ≈ 0.443 s。
  - 大小：CSV ≈ 51 MB；Parquet ≈ 8 MB。
  - 代码与命令：

Python

```python
import time, os, pandas as pd
# Read from CSV
t0 = time.time()
data_from_csv = pd.read_csv(os.path.join('..', 'data', 'airline.csv'))
print(time.time() - t0)  # ~1.815 s
# Write Parquet
data_from_csv.to_parquet(os.path.join('..', 'data', 'airline.parquet'))
# Read from Parquet
t0 = time.time()
data_from_parquet = pd.read_parquet(os.path.join('..', 'data', 'airline.parquet'))
print(time.time() - t0)  # ~0.443 s
```

Shell

```shell
ls -l ../data/airline.csv
ls -l ../data/airline.parquet
```