# STATS 769
# Data Formats

Paul Murrell

The University of Auckland

August 19, 2019

# Overview

- In this section of the course we will discuss some modern data formats and databases.
- JSON and MongoDB.
- XML and XPath and BaseX.

# NoSQL

- Relational databases are based on strict rules (tables, column types, primary keys, foreign keys) and have a standard query language (SQL).
- NoSQL databases (Not only SQL ?) deliberately break the rules of relational databases.
- They do this to obtain better speed and flexibility (for some applications).
- Examples include: graph databases, object-oriented databases, key-value databases, and document-oriented databases.

# Document-oriented databases

- As with relational databases, our interest is mainly in gaining access to the data and querying the database to extract a subset.
- We need to know basic concepts about document-oriented databases (how the data are structured; what documents look like) and we need to know how to query a document-oriented database (replacements for SQL).

# Document-oriented databases

- A collection of "documents".
- All information about an object stored in a single document (rather than spread across multiple tables).
- No requirement that all documents have same structure.
- Example document formats are JSON and XML.

# JSON

- JSON Data Types: `null`, `true`, `false`, number, string.
- JSON Data Containers: ordered (and unnamed) arrays `[ ]`, named (and unordered) arrays `{ }`.
- Can mix types within arrays.

# JSON in R

- The **jsonlite** package has `fromJSON()` (and `toJSON()`).
- The result is the simplest structure possible, in the order: vector, matrix, data frame, list.
- Nested JSON objects can produce nested data frames; the `flatten()` function turns these into normal 2-D data frames.
- Use `prettify()` to eyeball raw JSON and `str()` to explore nested list structures.

# MongoDB

- A document-oriented database based on JSON documents.
- R interface provided by the **mongolite** package.
- Connect via `m <- mongo(...)`

# MongoDB

- MongoDB allows queries via a `find` operation that uses documents to specify selections (with a JSON-like syntax)
- A query, via `m$find()`, has the form:

  `find(query, fields, sort, limit)`

  `find` selects documents from a single collection (like `SELECT`).
- `query` identifies documents to select (like `WHERE` in `SELECT`).
- `fields` selects fields from document (like column specification in `SELECT`); 0 means exclude, 1 means include.
- `sort` specifies result order (like `ORDER BY` in `SELECT`); 1 means ascending, -1 means descending.
- `limit` specifies how many results to return (like `LIMIT` in `SELECT`)

# MongoDB

- Aggregation (`GROUP BY`) is also possible (via `m$aggregate()`)
- The first argument is an "aggregation pipeline" consisting of "aggregation stages".
- `$match` filters documents.
- `$group` groups documents.
- `$sort` and `$limit` do what you would expect.

# XML in R

- The **xml2** package has `read_xml()`
- The result is NOT a data frame.
- Extract elements of interest using `xml_find_all()` and XPath expressions
- Use `xml_text()` to extract text content from elements.
- Use `xml_attr()` to extract attribute values from elements.
- **NOTE** that all content and attribute values are **character** values.

# XPath

Language for expressing subsets of an XML document.

- An XPath expression consists of some combination of **location paths** of the form ...

  `axisname::nodetest[predicate]`

  ... but usually just ...

  `nodetest`

  ... or ...

  `nodetest[predicate]`

# XPath

An XPath expression is formed by combining several location paths, separated by a forward slash, `/`.

- If the expression **begins** with a forward slash, matching starts from the document root node.

  ```
  /a/b/c
  ```

- A double forward slash, `//`, is short for `/descendant-or-self::node()/`.

  ```
  /a//c
  ```

- A vertical bar, `|`, means "or" (the result is the union of two expressions).

  ```
  /a/b | //c
  ```

# XPath

The nodetest is commonly just the name of an element or `@name` to match an attribute.

- It can also be a wildcard, `*`, which matches any element, or `@*`, which matches any attribute.

```
/a/b
/a/@id
/a/*/c
```

# XPath

The predicate is like a subsetting expression.

- It can be a simple integer.

```
/a/b[1]
```

- It can be a comparison.

```
/a/b[@year > 2000]
/a[@lang = 'en']
```

- It can be a special function.

```
/a/b[last() - 1]
/a/b[contains(@id, 'paul')]
```

# XPath

The axis is relative to the current node in the XML document. The default axis is `child`, which means to search children of the current node (this is what happens if we do not specify an axis in a location path).

- The `following-sibling` axis can force the search to look at siblings of the current node.

  ```
  /a/b/following-sibling::*
  ```

- The `parent` or `ancestor` axes can force the search to look back up the hierarchy of XML nodes.

  ```
  /a/b/c/ancestor::a
  ```

- These can be useful within a predicate.

  ```
  //c[ancestor::b@lang = 'en']
  ```

# BaseX

- A document-oriented database based on XML documents.
- BaseX provides an implementation of a query language for XML documents called XQuery.

# XQuery

- The simplest XQuery expression is just text
- The next simplest XQuery expression is just an XPath expression
- XQuery can include one or more FLWOR expressions
  - FLWOR is an acronym for "For, Let, Where, Order by, Return"
  - FLWOR is the XQuery equivalent of `SELECT`
  - The `for` clause selects elements, like the `FROM` clause
  - The `where` clause filters results, like the `WHERE` clause
  - The `order by` clause defines the result order, like `ORDER BY`
  - The `return` clause specifies the result, like the column specification in `SELECT`
- The return can include a mixture of literal XML and FLWOR expressions, but the FLWOR expressions must be "enclosed" in `{ }`.

# Reading

- 'jsonlite' mapping between R objects and JSON
  http://cran.r-project.org/web/packages/jsonlite/vignettes/json-mapping.pdf
- MongoDB Cheat Sheet
  `https://www.opentechguides.com/how-to/article/mongodb/118/mongodb-cheatsheat.html`
- w3schools XPath tutorial
  http://www.w3schools.com/xpath/
- BaseX manual
  `http://docs.basex.org/wiki/Main_Page`
- XQuery examples
  `http://www.datypic.com/books/xquery/examples.html`