STATS 769
Data Technologies Review

Paul Murrell

The University of Auckland

July 29, 2019

Overview

- This section of the course provides a refresh on basic data technologies.
- Our primary computing environment will be R.

File Formats

Flat text files are the simplest data source.

- Data stored in rows, with multiple values per row.
- Delimited files have a delimiter between values.
- Fixed-width files use a constant number of characters per value.
- CSV (comma-separated-value) files use a comma as the delimiter.
- The first row may contain variable names.
- There may be several lines of metadata at the top of the file.
- Any software can read/write the file.

File Formats

An example of a CSV file:

```
publicid,origintime,longitude,latitude,depth
2014p172810,2014-03-05T22:33:37,175.93,-36.91,368.75
2014p172742,2014-03-05T21:57:26,175.89,-40.60,22.46
2014p172698,2014-03-05T21:33:31,175.86,-40.63,27.44
```

File Formats

An example of a fixed-width file with metadata and column headings:

```
         VARIABLE : Mean TS from clear sky composite
         FILENAME : ISCCPMonthly_avg.nc
         FILEPATH : /usr/local/fer_data/data/
         SUBSET   : 24 by 24 points (LONGITUDE-LATIT
         TIME     : 16-JAN-1995 00:00
            113.8W 111.2W 108.8W 106.2W 103.8W 101.2W
              27     28     29     30     31     32
36.2N / 51:  272.7  270.9  270.9  269.7  273.2  275.6
33.8N / 50:  279.5  279.5  275.0  275.6  277.3  279.5
31.2N / 49:  284.7  284.7  281.6  281.6  280.5  282.2
28.8N / 48:  289.3  286.8  286.8  283.7  284.2  286.8
26.2N / 47:  292.2  293.2  287.8  287.8  285.8  288.8
```

File Formats

R can easily read text files and create a data frame.

- `read.csv(file, header, skip)`
- `read.table(file, header, sep, skip)`
- `read.fwf(file, widths, header, skip)`

File Formats

Binary file formats are more complex, but tend to be more efficient in terms of both speed and size.

- Data stored in arbitrarily complex forms.
- Require specific software to read/write.

File Formats

An example of a binary file:

```
2500  :  04 00 40 00 40 00 c1 01 08 00 c1 01 00 00 54 8d 01 00 eb 00 5a 00 0f 00  |  ..@.@.........T.....Z...
2524  :  00 f0 52 00 00 00 00 00 06 f0 18 00 00 00 00 04 00 00 02 00 00 00 01 00  |  ..R.....................
2548  :  00 00 01 00 00 00 01 00 00 00 01 00 00 00 33 00 0b f0 12 00 00 00 bf 00  |  ..............3.........
2572  :  08 00 08 00 81 01 09 00 00 08 c0 01 40 00 00 08 40 00 1e f1 10 00 00 00  |  ............@...@.......
2596  :  0d 00 00 08 0c 00 00 08 17 00 00 08 f7 00 00 10 fc 00 2b 01 23 00 00 00  |  ..................+.#...
2620  :  23 00 00 00 08 00 00 6c 61 74 69 74 75 64 65 07 00 00 58 31 31 33 2e 38  |  #......latitude...X113.8
2644  :  57 07 00 00 58 31 31 31 2e 32 57 07 00 00 58 31 30 38 2e 38 57 07 00 00  |  W...X111.2W...X108.8W...
2668  :  58 31 30 36 2e 32 57 07 00 00 58 31 30 33 2e 38 57 07 00 00 58 31 30 31  |  X106.2W...X103.8W...X101
2692  :  2e 32 57 06 00 00 58 39 38 2e 38 57 06 00 00 58 39 36 2e 32 57 06 00 00  |  .2W...X98.8W...X96.2W...
2716  :  58 39 33 2e 38 57 06 00 00 58 39 31 2e 32 57 05 00 00 33 36 2e 32 4e 05  |  X93.8W...X91.2W...36.2N.
2740  :  00 00 33 33 2e 38 4e 05 00 00 33 31 2e 32 4e 05 00 00 32 38 2e 38 4e 05  |  ..33.8N...31.2N...28.8N.
2764  :  00 00 32 36 2e 32 4e 05 00 00 32 33 2e 38 4e 05 00 00 32 31 2e 32 4e 05  |  ..26.2N...23.8N...21.2N.
2788  :  00 00 31 38 2e 38 4e 05 00 00 31 36 2e 32 4e 05 00 00 31 33 2e 38 4e 05  |  ..18.8N...16.2N...13.8N.
2812  :  00 00 31 31 2e 32 4e 04 00 00 38 2e 38 4e 04 00 00 36 2e 32 4e 04 00 00  |  ..11.2N...8.8N...6.2N...
2836  :  33 2e 38 4e 04 00 00 31 2e 32 4e 04 00 00 31 2e 32 53 04 00 00 33 2e 38  |  3.8N...1.2N...1.2S...3.8
2860  :  53 04 00 00 36 2e 32 53 04 00 00 38 2e 38 53 05 00 00 31 31 2e 32 53 05  |  S...6.2S...8.8S...11.2S.
2884  :  00 00 31 33 2e 38 53 05 00 00 31 36 2e 32 53 05 00 00 31 38 2e 38 53 05  |  ..13.8S...16.2S...18.8S.
2908  :  00 00 32 31 2e 32 53 ff 00 0a 00 23 00 40 04 00 00 0c 00 00 00 63 08 15  |  ..21.2S....#.@.......c..
2932  :  00 63 08 00 00 00 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 02 0a 00  |  .c......................
2956  :  00 00 09 08 10 00 00 06 10 00 bb 0d cc 07 00 00 00 00 06 00 00 00 0c 00  |  ........................
2980  :  02 00 64 00 0f 00 02 00 01 00 11 00 02 00 00 00 10 00 08 00              |  ..d.................
```

File Formats

An example of a binary file:

```
=========
4000  :  00 01 00 01 00 0f 00 1b aa 01 00 03 02 0e 00 01 00 02 00 0f 00  |  .....................
=========temperature
4021  :  66 66 66 66 66 ee 70 40                                          |  270.9
=========
4029  :  03 02 0e 00 01 00 03 00 0f 00                                    |  ..........
=========temperature
4039  :  66 66 66 66 66 ee 70 40                                          |  270.9
=========
4047  :  bd 00 12 00 01 00 04 00 0f 00 6b a5 01 00 0f 00 e3 aa 01 00     |  ..........k.........
=========
4067  :  05 00 03 02 0e 00 01 00 06 00 0f 00                              |  ............
=========temperature
4079  :  9a 99 99 99 99 39 71 40                                          |  275.6
=========
4087  :  bd 00 12 00 01 00 07 00 0f 00 4b b1 01 00 0f 00 4b b1 01 00     |  ..........K.....K...
=========
4107  :  08 00 03 02 0e 00 01 00 09 00 0f 00                              |  ............
=========temperature
4119  :  66 66 66 66 66 6e 71 40                                          |  278.9
=========
4127  :  03 02 0e 00 01 00 0a 00 0f 00                                    |  ..........
=========temperature
4137  :  66 66 66 66 66 6e 71 40                                          |  278.9
```

File Formats

Each binary format tends to need a specific package and the results are not necessarily as simple as a data frame:

- **foreign** for a variety of statistical software binary formats (SPSS, SAS, etc)
- **RNetCDF** for netCDF files
- **hdf5** for HDF5 files

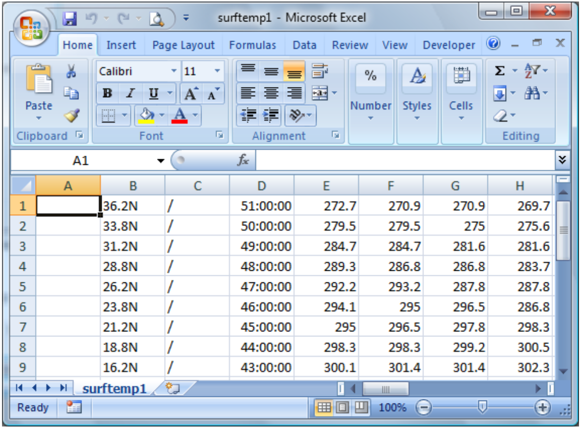
Spreadsheets

Spreadsheets are not an ideal data format, but are very common.

- Data stored in rows and columns.
- There may be multiple sheets in one workbook.
- There may be formatting for humans, which can make processing more difficult.
- Can be easily converted to flat text file (if there is only one sheet).
- Require specific software to read/write.

Spreadsheets

An example of a spreadsheet:



	A	B	C	D	E	F	G	H
1		36.2N	/	51:00:00	272.7	270.9	270.9	269.7
2		33.8N	/	50:00:00	279.5	279.5	275	275.6
3		31.2N	/	49:00:00	284.7	284.7	281.6	281.6
4		28.8N	/	48:00:00	289.3	286.8	286.8	283.7
5		26.2N	/	47:00:00	292.2	293.2	287.8	287.8
6		23.8N	/	46:00:00	294.1	295	296.5	286.8
7		21.2N	/	45:00:00	295	296.5	297.8	298.3
8		18.8N	/	44:00:00	298.3	298.3	299.2	300.5
9		16.2N	/	43:00:00	300.1	301.4	301.4	302.3

Spreadsheets

One approach with R is to convert the spreadsheet to a flat file and then use previous functions.
Otherwise, there are packages that can read spreadsheets directly, typically as data frames:

- **gdata** has `read.xls()`.
- **xlsReadWrite** (Windows only) also has `read.xls()`.
- **xlsx** has `read.xlsx()`.
- **RODBC** provides detailed access to individual spreadsheet cells.

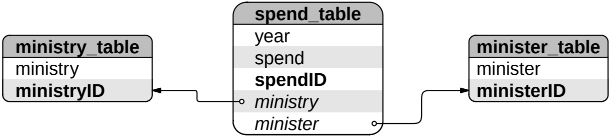
Relational Databases

Databases provide a more sophisticated storage option, especially for large and/or complex data.

- The data are spread across multiple tables.
- Table columns are typed.
- Each table has a primary key.
- Tables are linked by foreign keys.
- Access can be limited to authorised users.
- Access can be concurrent for multiple users.
- Require very specific software to read/write.
- Standard interface via SQL.

Relational Databases

An example of a database:



Relational Databases

An example of a database:

```
spend_table
+------+-------+---------+----------+----------+
| year | spend | spendID | ministry | minister |
+------+-------+---------+----------+----------+
| 1972 |   335 |       1 |        1 |        2 |
| 1973 |   377 |       2 |        1 |        2 |
| 1974 |   440 |       3 |        1 |        2 |
| 1975 |   527 |       4 |        1 |        2 |
| 1976 |   627 |       5 |        1 |        1 |
| 1977 |   699 |       6 |        1 |        1 |
| 1978 |   808 |       7 |        1 |        1 |
| 1972 |   292 |       8 |        2 |        5 |
| 1973 |   343 |       9 |        2 |        3 |
| 1974 |   401 |      10 |        2 |        6 |
| 1975 |   492 |      11 |        2 |        4 |
| 1976 |   606 |      12 |        2 |        4 |
| 1977 |   689 |      13 |        2 |        4 |
| 1978 |   809 |      14 |        2 |        4 |
+------+-------+---------+----------+----------+
```

```
ministry_table
+-----------+------------+
| ministry  | ministryID |
+-----------+------------+
| Education |          1 |
| Health    |          2 |
| Social    |          3 |
+-----------+------------+
```

```
minister_table
+---------------------+------------+
| minister            | ministerID |
+---------------------+------------+
| Les Gandar          |          1 |
| Phil Amos           |          2 |
| Bob Tizard          |          3 |
| Frank Gill          |          4 |
| Lance Adams-Schneide |          5 |
| Tom McGuigan        |          6 |
+---------------------+------------+
```

Relational Databases

SQL can be used to access information in a database:

```
SELECT year, spend, mr.minister, my.ministry
  FROM spend_table st
    INNER JOIN minister_table mr
      ON st.minister = mr.ministerID
    INNER JOIN ministry_table my
      ON st.ministry = my.ministryID
  WHERE year = 1978;
```

```
+------+-------+-------------+-----------+
| year | spend | minister    | ministry  |
+------+-------+-------------+-----------+
| 1978 |   808 | Les Gandar  | Education |
| 1978 |   809 | Frank Gill  | Health    |
+------+-------+-------------+-----------+
```

Relational Databases

There are R packages for querying databases and getting the result as a data frame:

- **ROracle**, **RMySQL**, and **RSQLite** provide `dbConnect()` and `dbGetQuery()`.

HTML and XML

HTML and XML are useful to know for several reasons: they are another potential data source format and HTML is a useful document format (for reports).

- Text files consisting of content and markup.
- Markup consists of elements and attributes.

HTML and XML

An example HTML file:

```
<html>
<head>
  <title>Electorial Donations 2014</title>
</head>
<body>
  <h1>Electoral Donations</h1>
  <p>
    Source:
    <a href="http://www.elections.org.nz/">
      The Electoral Commission of New Zealand</a>.
  </p>
</body>
</html>
```

HTML and XML

An example XML file:

```
<?xml version="1.0"?>
<ElectoralDonations>
  <party id="P2" name="National">
    <candidate id="C1" name="Amy" surname="ADAMS">
      <donation id="d1" amount="15000.00"/>
      <donation id="d2" amount="10000.00"/>
    </candidate>
  </party>
</ElectoralDonations>
```

HTML and XML

R has packages for working with HTML and XML documents. The result may be a data frame or it may be something insanely complicated:

- **XML** has `readHTMLTable()` and `xmlParse()`

We will be going into more depth with processing HTML and XML later in the course.

Data Structures

R has a small set of standard data structures:

- Vectors are 1-dimensional and have a type (character, numeric, logical).
- Matrices are 2-dimensional and have a type.
- Data frames are 2-dimensional and each column has a type.
- Lists are recursive; each component of a list can be any data structure.
- Data frame are lists where each component has to be the same length.
- Use `str()` to get a low-level view of a data structure.

Subsetting

- Single square brackets, `x[index]`, can be used to extract a range of values from `x`; the result is usually the same class as `x`.
- Double square brackets, `x[[index]]` can be used to extract a single component from a list.
- The `index` can be numeric, logical, or character.
- The syntax `x$name` is short for `x[["name"]]`.

Control Flow

R has standard conditional and loop constructs:

```
if (condition) {
  ...
} else {
  ...
}

for (i in values) {
  ...
}

while (condition) {
  ...
}
```

Writing Functions

We can define new R functions:

```
f <- function(x, y=0) {
   ...
}
```

- arguments can have default values.
- the last expression in the body of the function provides the return value for the function.

Data Processing

R has tools for manipulating data structures:

- Summary functions: `min()`, `max()`, `sum()`, `range()`, `mean()`.
- Generating sequences: `seq()`, `rep()`, `c()`.
- Tables of counts: `table()`, `ftable()`, `xtabs()`.
- Combining structures: `cbind()`, `rbind()`, `merge()`.
- Apply functions: `apply()`, `sapply()`, `lapply()`.
- Aggregation functions: `aggregate()`, `tapply()`, `by()`.
- Split-apply-recombine: `split()`, `lapply()`, `do.call()`.

Reshaping

- Long format.
- Wide format.
- The **reshape2** package.

Text Processing

R has tools for manipulating text:

- Search (and replace) text: `grep()`, `regexpr()`, `gsub()`.
- Break text into pieces: `strsplit()`.
- Combine text: `paste()`.
- Most functions make use of regular expressions.

Dates

- Format dates.
- Generate sequences of dates.
- Perform arithmetic on dates.

Debugging

- Use `traceback()` to show the call stack after an error.
- Use `browser()` to interrupt execution and inspect objects.
- Use `debug()` to interrupt execution when a function is called.
- Use `trace()` to interrupt execution within a function call.
- Use `recover()` to browse any currently active function calls.
- Use `options(warn)` to turn warnings into errors.
- Use `options(error)` to call `recover()` after an error.

Resources

- "Introduction to Data Technologies"
  `https://www.stat.auckland.ac.nz/~paul/ItDT/`
- Ross Ihaka's "An R Programming Quick Reference"
  (on Canvas)
- Duncan Murdoch's "Debugging in R"
  `https://web.archive.org/web/20170706215053/http://www.stats.uwo.ca:80/faculty/murdoch/software/debuggingR/`