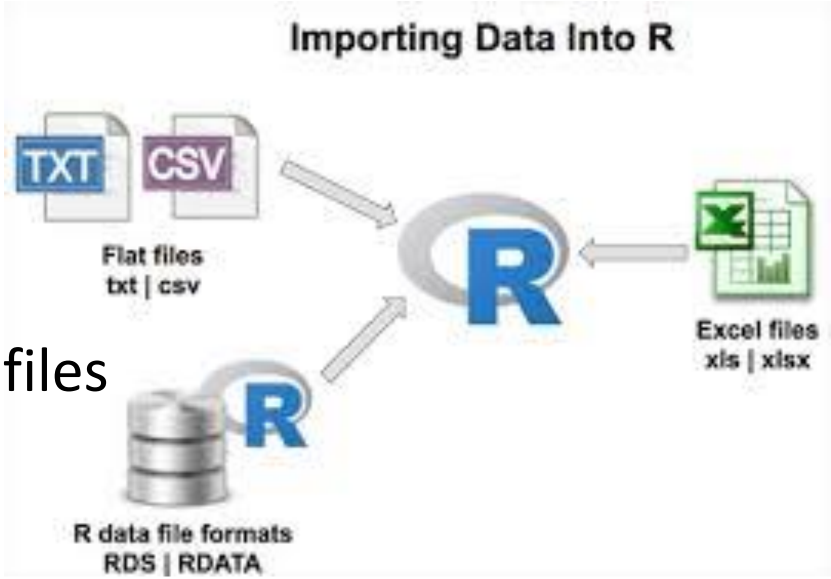
Lesson 03
Data Extraction

Outline



- Getting Data Into R
- Text files
  - Comma Separated Values (CSV) files
  - Other delimited TXT files
- Excel files
- Databases
  - SQLite, MySQL, …
- Web Data
  - HTML tables
  - JSON files
- Other Statistical Programs
  - SAS, SPSS, Minitab, …
- R Binary (RData) files
- R Sample Data

CSV Files – read.csv Function

- Open the already created project
  - Project: `Data_Extract.Rproj`
  - Folder: `IDSC_4110_Files/03_Data_Extract/Lectures`
  - File: `Extract1_Import_CSV.r`
- Reading Loans CSV file
  - Use `getwd()` function to get current working directory
  - Add the forward slash (“/”) and the file name
    - Note the use of forward instead of back slashes in Windows paths
  - Use `read.csv` function to read content into data frame for further analysis

CSV Files – read.table Function

- The all-purpose function for reading text files
  - `read.table` from which others are derived
  - The most useful (and necessary) paramters
    - `file` – path and file name to be read (required)
    - `header` – TRUE for first row of headers, FALSE by default
    - `sep` – “” by default, comma for CSV files, could be tab, semicolon, ...
    - `stringsAsFactors` – when set to TRUE all character columns read as factors, could slow down big files with lots of different char levels
    - `quote` – used to deal with single and double quotes as string delimiters
    - `colClasses` – for specifying data types of each column

CSV Files – Other Reading Functions

- Other functions derived from `read.table`
  - `read.delim` with tab `\t` as default separator
  - `read.csv2` and `read.delim2`
    - For data with comma for decimals like 123,45 euros or yen
- Large files should be read with
  - `read_csv` or other functions from `readr` package
    - Used towards the end of the class with `dplyr` package

Excel Files – readxl Package

- Preferred to export into CSV files
- Possible to read directly using **`readxl`** package
  - Accommodates both `.xls` and `.xlsx` extensions
  - Can specify the worksheet you want to read from
- Read Investments Excel file
  - Install and load **`readxl`** package
  - Create the appropriate path and file name
  - Use `read_excel` function to read content of the second sheet into a `tibble` data frame for further analysis

Databases – Install SQLite

- Download and install SQLite

  https://github.com/pawelsalawa/sqlitestudio/releases

  - Download the `InstallSQLiteStudio` EXE file (Windows) or DMG file (Mac)
  - Run the installation following the instructions
- Additional sites for SQLite and SQLiteStudio
  - SQLite
    - https://www.sqlitetutorial.net/download-install-sqlite/
    - https://www.sqlite.org/download.html
  - SQLiteStudio
    - https://sqlitestudio.pl/

Databases – Sporting Goods

- Run SQLite Studio
  - Database -> Add Database -> New Database File
    - Folder: `03_Data_Extract/Lectures/Data_Extract`
    - File: `SportingGoods`
- Execute the following commands
  - **Connect to SportingGoods database**
  - **Open SQL editor**
  - **Load SQL from file**
    - Folder: `03_Data_Extract/Lectures/Data_Extract`
    - File: `SportingGoods.sql`
  - Run the code with **Execute Query** button

Databases - Connectivity

- Accessed through various drivers
  - ODBC: Open Database Connectivity
- Specific open-source database connectivity packages
  - **`RMySQL`** and **`RPostgreSQL`**
- Databases without a specific package
  - Use **`DBI`** or **`RODBC`** packages
- We will use SQLite specific package
  - Install and load **`RSQLite`** package
  - Specify driver with `dbDriver` function

```
SQLite_driver <- dbDriver("SQLite")
```

  - Establish database connection with `dbConnect` function

```
sports_conn <- dbConnect(SQLite_driver, db_file)
```

  - Disconnect from the database when done

```
dbDisconnect(sports_conn)
```

Databases – Product Query

- List all the tables in the database

```
dbListTables(sports_conn)
```

- List all the fields in the Product table

```
dbListFields(sports_conn, "Product")
```

- Create a query string to list the product name, product group and retail price

```
sql_str_prod <- "SELECT ProdName, ProdGroup, RetailPrice
FROM Product"
```

- Run the query to retrieve the results into a data frame for further analysis

```
prod_df <- dbGetQuery(sports_conn, sql_str_prod)
```

Databases – Sales View

- Most users have limited access to database data through so-called views or stored queries
- List all the tables in the database again to see the `Sales` view

```
dbListTables(sports_conn)
```

- List all the fields in the Sales view just like you would for a table

```
dbListFields(sports_conn, "Sales")
```

- Create a query string to select everything from the view

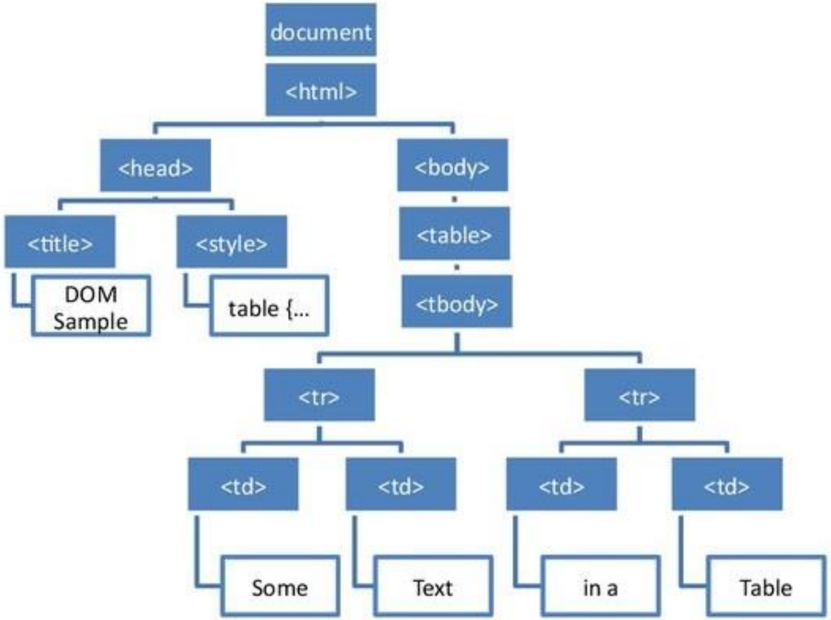
```
sql_str_prod <- "SELECT * FROM Sales"
```

- Run the query to retrieve the results into a data frame for further analysis

```
sales_df <- dbGetQuery(sports_conn, sql_str_sales)
```

Web Data – HTML and DOM

- Web pages written in HTML
- Hypertext **Markup Language**
  - “Marks up” text in different ways (headers, lists, spans, divs and **tables**)
  - Mostly interested in HTML tables
- Browser’s parser creates a document object model (DOM)
  - DOM is a hierarchical representation of the Web page
  - Tags open **<table>** and close **</table>** a portion of DOM tree



HTML Tables – XML Package

- We want to analyze data from a Web page
  - Data not provided in easy to access format, like .csv files
  - Need to “scrape” the data into R data frames
- Using `XML` package
  - `htmlTreeParse` – creates R data structure `HTMLInternalDocument` representing the HTML tree
  - `getNodeSet` – uses the HTML tree structure to create another R data structure `XMLNodeSet` which includes **<table>** nodes among other HTML elements
  - `readHTMLTable` – used to retrieve the data from the <table> node into a R data.frame

HTML Tables – rvest Package

- Using `rvest` package
  - Conceptually identical to `XML` package procedure
  - `read_html` – creates R data structure `xml_document` representing the HTML tree
  - `html_nodes` – uses the HTML tree structure to create another R data structure `xml_nodeset` which includes **<table>** nodes among other HTML elements
  - `html_table` – used to retrieve the data from the <table> node into R data.frame

Web Data – JSON Files

- JSON = JavaScript Object Notation
  - A text-based data format well suited for complex (nested) list-like data structures
  - Used to transmit data between a server and Web application
  - An alternative to XML, easier to read and work with
- Using `jsonlite` package
  - Created JSON files in R using `write_json` function
  - Reading the files back in using `read_json` function
  - Compare and contrast reading into lists vs. vector & data frame structures
  - Need to stay on top of the type of data structure one is working with – it can get pretty messy

Other Statistical Programs

- Proprietary statistical software still has significant market share
  - SPSS, Stata, SAS, …
- Using `foreign` package
  - Reading files with `read.spss`, `read.dta`, `read.ssd`, … functions into data frames
- Using `haven` package
  - Reading files with `read_spss`, `read_stata`, `read_sas`, … functions into tibbles

R Binary RData Files

- Use `save/load` pair of functions to create and restore an `rdata` file
  - `load` function does not need a data frame name because it is restored from the R workspace with the same name it was created with using `save` function
- Alternatively use `saveRDS/readRDS` pair of functions to create and restore an `rds` file
  - `readRDS` function needs a data frame name because it is not maintained with `saveRDS` function

Sample Data in R

- Display a list of pre-loaded R data sets with

```
data()
```

- Most (if not all?) of these are data frames
- Type the name of the data frame to see content

```
mtcars
```

- To learn about the data frame

```
?mtcars
```

- Use data() function to actually build the data frame

```
data(mtcars)
```

Summary

- Reviewed all the major types of R data sources
- Most of the external data comes from either
  - CSV and Excel files
  - Databases (reviewed basics of SQL)
- Another major source is the data embedded in Web pages
  - HTML tables and other markup features
  - JSON files for more complex nested lists
- Other statistical software programs
  - SPSS, Stata, SAS, ….
- R binary data files and pre-loaded sample data