Getting and Cleaning Data - Week 2

Carmelo Ramirez

19/10/2020

READING FROM SQL

Connecting and Listing Databases

```
ucscDB <- dbConnect(MySQL(), user = "genome", host = "genome-mysql.cse.ucsc.edu")
result <- dbGetQuery(ucscDB, "show databases;")
dbDisconnect(ucscDB)
```

Connecting to Specific Database

```
hg19 <- dbConnect(MySQL(), user = "genome", db = "hg19", host = "genome-mysql.cse.ucsc.edu")
allTables <- dbListTables(hg19)
length(allTables)
```

Get dimensions of a specific table

```
dbListFields(hg19, "affyU133Plus2")
dbGetQuery(hg19, "select count(*) from affyU133Plus2")
```

Read from table

```
affyData <- dbReadTable(hg19, "affyU133Plus2")
head(affyData)
```

Select a specific Subset

```
query <- dbSendQuery(hg19, "select * from affyU133Plus2 where misMatches between 1 and 3")
affyMis <- fetch(query)
dbClearResult(query)
```

READING HDF5

- Used to store large data sets
- Supports storing a range of data types
- Hierarchical data format
- *groups* containing zero or more data sets and metadata
  - Have a *group header* with group name and list of attributes
  - Have a group *symbol table* with a list of objects in group
- *datasets* multidimensional array of data elements with metadata
  - Have a *header* with name, datatype, dataspace, and storage layout
  - Have a *data array* with data

Create H5 Files

```r
library(rhdf5)

created = h5createdFile("example.h5")
```

Create H5 Groups

```r
created = h5createGroup("example.h5", "foo")
created = h5createGroup("example.h5", "baa")
created = h5createGroup("example.h5". "foo/foobaa")
h5ls("example.h5")
```

Write to Groups

```r
A = matrix(1:10, nrow = 5, ncol = 2)
h5write(A, "example5.h5", "foo/A")
B = array(seq(0.1,2.0, by = 0.1), dim = c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, "example.h5", "foo/foobaa/B")
h5ls("example.h5")
```

Write to Data Frames

```r
df = data.frame(1L:5L, seq(0,1, length.out = 5),
                c("ab", "cde", "fghi", "a", "s", stringsAsFactors= FALSE))
h5write(df, "example.h5", "df")
h5ls("example.h5")
```

Reading Data

```r
# Reading Data
readA = h5read("example.h5", "foo/A")
readB = h5read("example.h5", "foo/foobaa/B")
readF = h5read("example.h5", "df")
```

Writing and Reading in Chunks

```r
h5write(c(12,13,14), "example.h5", "foo/A", index = list(1:3, 1))
h5read("example.h5", "foo/A")
```

READING FROM THE WEB

Web Scrapping

Web Scrapping: Programatically extracting data from the HTML code of websites.

- It can be a great way to get data
- Many websites have information you may want to programatically read
- In some cases this is against the term of service for the website
- Attempting to read too many pages too quickly can get your IP address blocked

Getting data from webpages - readLines()

```r
con = url("http://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en")
htmlCode = readLines(con)
close(con)
```

Parsing with XML

```r
library(XML)
url <- "http://scholar.google.com/citations?user=HI-I6C0AAAAJ&hl=en"
html <- htmlTreeParse(url, useInternalNodes = TRUE)
xpathSApply(html, "//title", xmlValue)
xpathSApply(html, "//td[@id='col-citedby']", xmlValue)
```

GET from the httr package

```r
library(httr)
html = GET(url)
content = content(html, as="text")
parsedHtml = htmlParse(content, asText=TRUE)
xpathSApply(parsedHtml, "//titles", xmlValue)
```

Accessing websites with passwords

```r
page = GET(url, authenticate("user", "password"))
```

Using Handles

```r
google = handle("http://google.com")
page1 = GET(handle = google, path = "/")
page2 = GET(handle = google, path = "search")
```

READING FROM API

Accessing Twitter API from R

```r
myapp = oatuh_app("twitter", key = "yourConsumerKeyHere",
                  secret = "yourConsumerSecret")
sig = sign_ouath1.0(myapp, token = "yourTokenHere",
                    token_secret = "yourTokenSecretHere")
homeTL = GET("https://api.twitter.com/1.1/statuses/home_timeline.json", sig)
```

Converting the JSON object

```r
json1 = content(homeTL)
json2 = jsonlite::fromJSON(toJSON(json1))
json2[1, 1:4]
```