**Coursera Data Science Course 3**

**1. How to Read from MySQL?**

**MySQL**another open source database software  
data is in – databases, tables within databases, fields within tables  
each row is called a record

Tables are linked together like a flow chart

**Connect/List Database  
ucscDb <- dbConnect(MySQL(), user=”genome”, host=”genome-mysql.cse.ucsc.edu”)** // connect **result <- dbGetQuery(ucscDb, “show databases;”); dbDisconnect(ucscDB);** // list, also always remember to disconnect!! ☺  
**result** // prints list

**Example: hg19, human genome**

Connect  
**allTables <- dbListTables(hg19)  
length(allTables)** // tells how many tables there are  
**allTables[1:5]**// gives name of the first five tables

**dbListFields(hg19, “name of the table”)** // this will give you all the “fields” of a table or columns. i.e. matches, mismatches etc  
**dbGetQuery(hg19, “select count(\*) from “name of table””)** // this gives you the amount of rows

**Reading from Data  
affyData <- dbReadTable(hg19, “name of table”)  
head(affyData)**return the entire table

**If we want a specific subset instead..  
query <- dbSendQuery(hg19, “select \* from “name of table” where mismatches between 1 and 3”)** // select all observations where mismatches variable = something between 1 and 3 **affyMis <- fetch(query); quantile(affyMis$misMatches)**

**You can also only pick a smaller data set  
affyMisSmall <- fetch(query,n=10); dbClearResult(query);** //only fetch the top 10 rows. Remember to clear result

**YOU HAVE TO HAVE TO HAVE TO USE: dbDisconnect(hg19)**

**Note: Be mindful of other people’s data and don’t delete or join things.**

**2. Reading HDF5**

**HDF5**-stores large data sets  
-hierarchical data format  
-**groups** containing zero or more data sets and metadata  
 -group header w/ name and attributes  
 -group symbol table with a list of objects  
-**datasets** multidimensional array of data elements with metadata  
 -header: name, datatype, dataspace, and storage layout  
 -data array with the data

**Installing R HDF5 Package  
source(<http://bioconductor.org/biocLite.R>)  
biocLite(“rhdf5”)**

**Library(rhdf5)  
created = h5createFile(“example.h5”)  
created** // should return true

**How to create groups from the file  
created = h5createGroup(“example.h5”, “foo”)** // create a group named foo  
**created = h5createGroup(“example.h5”, “foo/foobaa”)** // create a group named foobaa and it’s a subset from the group foo   
**h5ls(“example.h5”)** // list the groups

**Write to groups  
A = matrix(1:10, nr=5, nc=2)** // creates some random matrix A with 5 rows and 2 columns  
**h5write(A, “example.h5”, “foo/A”)** // writes the matrix A into group foo   
**B = array(seq(0.1,2.0, by=0.1), dim=c(5,2,2))** // creates aa 3D array B  
**attri(B, “scale”) <- “liter”** // can add attributes to B such as units (in our case, liter) **h5write(B, “example.h5”, “foo/foobaa/B”)** //writes the matrix  
**h5ls(“example.h5”)** // lists out the whole table

**Write a data set  
df = data.frame(1L:5L,seq(0,1,length.out=5), c(“ab”, “cde”, “fghi”, “a”, ”s”), stringsAsFactors=FALSE)** //creates a data frame, specifying rows and columns **h5write(df, “example.h5”, “df”)** //writes in df and names it df **h5ls(“example.h5”)**

**Reading data  
readA = h5read(“example.h5”, “foo/A”)** // how to read a data set

**How to write or read large amounts at once  
h5write(c(12,13,14),”example.h5”,”foo/A”, index=list(1:3, 1))** // writing elements 12 13 and 14, and also write it to a specific part of the dataset, therefore using the indices to specify where  
Note that indices can be used to read specific parts of the dataset as well **h5read(“example.h5”, “foo/A”)**

**3. Reading Data from the Web**

**Webscraping**- programmatically extract data from HTML code of websites  
-it’s a great way to get data but it might be against some terms of service  
-reading too many pages at once can get IP address blocked

**Example:** Google Scholar

**To get data off of the webpage**  
**con = url(“the name of the link”)** // opens connection to URL  
**htmlCode = readLines(con)** //read out the data **close(con)** // remember to close the connection! **htmlCode** // will show the lines read but in a crappy format

**To fix formatting, parse with XML (refer to week 1 for more details)  
library(XML)  
url <- “name of the link”  
html <- htmlTreePrase(url, useInternalNodes=T)  
xpathSApply(html, “//title”, xmlValue)** // will show the title

**xpathSApply(html, “//td[@id=’col-citedby’]”, xmlValue)** // looks at the column of “cited by” specifically

**GET from the httr package** – another approach to get data **library(httr); html2 = GET(url)  
content2 = content(html2,as=”text”)** //extracts content as one big text string **parsedHtml = htmlParse(content2,asText=TRUE)** //parse the text! **xpathSApply(parsedHtml, “//title”, xmlValue”)** // will show the title

**Accessing websites with passwords**  
**pg1 = GET(<http://httpbin.org/basic-auth/user/passwd>)  
pg1**

^This will not work. Will not authenticate.

**pg2 = GET(same link as pg1, authenticate(“user”, “passwd”))  
pg2**

**names(pg2)** // will give you all the names

**Handle  
google = handle(<http://google.com>)  
pg1 = GET(handle=google,path=”/”)  
pg2 = GET(handle=google,path=”search”)**

By setting google as a handle, GET can go from google to anywhere. If the handle is authenticated once, and the cookies will save it to be authenticated forever! ☺

**4. Reading Data from the APIs**

**Create an app**<http://dev.twitter.com/apps>

Create a new application -> there are a bunch of setting such as organizations and OAuth, OAuth allows you to set authorization to different people etc

**Accessing Twitter from R  
myapp = oauth\_app(“twitter”, key=”yourConsumerKeyHere”,secret=”yourConsumerSecretHere”)  
sig = sign\_oauth1.0(myapp, token = “yourTokenHere”, token\_secret = “yourTokenSecretHere”)  
homeTL = GET(“https://api.twitter.com/1.1/statuses/home\_timeline.json”, sig)** //this is the URL that corresponds to twitter API. JSON file is the only supported thing for API. This will get us the page

^This allows us to access data that is privately held by Twitter that can only be available for people with an application.

**Then we will convert to the json object  
json = content(homeTL)** //content function will automatically recognize that this data is JSON data. This will turn the data into a somewhat structured R object **json2 = jsonlite::fromJSON(toJSON(json1))** // this will allow us to turn the JSON object to a data frame for easier reading. Then each row also corresponds to a tweet in the user’s home timeline. **json2[1, 1:4]** // first four columns tell when the tweet was created, ID number of tweet, ID string and the actual text.

**But how do we know which URL to use?**<http://dev.twitter.com/docs/api/1.1/get/search/tweets> will tell you a lot

Go to their API site for some overview

**Documentation is important!**-httr allows GET POST PUT DELETE only if authorized  
-authenticate with username + password  
-oauth is common  
-httr also works with facebook google and github etc

**5. Reading from other sources**

**There are a LOT of useful R packages.** -if you need something, just google “data storage mechanism R package”

**Recall all the ways to interact directly with files**file – open connection to text  
url – open connection to URL  
gzfile – open connection to .gz file  
bzfile – open connection to .bz2 file  
?connections – gives more info  
**REMEMBER TO CLOSE CONNECTIONS**

**Foreign package**useful for loading data from Minitab, S, SAs, SPSS, Stata, Systat  
basic functions include read.foo  
<http://cran.r-project.org/web/packages/foreign/foreign.pdf> has a complete list

Bunch of other database packages as well.

Can also read images!  
jpeg, readbitmap, png, EBImage – check

Can also read GIS data and music data

CHECK CRAN for everything!