Week 2

Week 2 of Getting and Cleaning Data: Extracting Data From Databases and the Web

- Welcome to Week 2 of Getting and Cleaning Data!
- The primary goal is to introduce you to the most common data storage systems and the appropriate tools to extract data from web or from databases like MySQL.
- Remember that the Course Project is open and ongoing. It is due BEFORE 11:30 PM UTC on the Sunday at the end of Week 3, but please don't put it off until the last minute.
- To access Course Project instructions and submission interface, click the Course Project link in the left navigation bar.
- With the skills you learn this week you should be able to start on the basic data extraction that will form the beginnings of your project.

The httr package

Question 1

Question

- Register an application with the Github API here

  `https://github.com/settings/applications.`

- Access the API to get information on your instructors repositories
- (hint: this is the url you want "https://api.github.com/users/jtleek/repos").
- Use this data to find the time that the datasharing repo was created.
- What time was it created?
- This tutorial may be useful

  `(https://github.com/hadley/httr/blob/master/demo/oauth2-github.r).`

- You may also need to run the code in the base R package and not R studio.

Options

(i) 2012-06-20T18:39:06Z

(ii) 2014-03-05T16:11:46Z

(iii) 2014-01-04T21:06:44Z

(iv) 2013-11-07T13:25:07Z

The Basics of Structured Query Language (SQL)

The sqldf package

Question 2

The **sqldf** package allows for execution of SQL commands on `R` data frames.

We will use the **sqldf** package to practice the queries we might send with the `dbSendQuery` command in RMySQL.

Download the American Community Survey data and load it into an R object called `acs`.

`https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06pid.csv`

Which of the following commands will select only the data for the probability weights `pwgtp1` with ages less than 50?

(i) `sqldf("select * from acs where AGEP < 50 and pwgtp1")`

(ii) `sqldf("select * from acs")`

(iii) `sqldf("select * from acs where AGEP < 50")`

(iv) `sqldf("select pwgtp1 from acs where AGEP < 50")`

```
library(sqldf)

## Data saved in local directory as "ss06pid.csv"

acs <- read.csv("./ss06pid.csv", header=T, sep=",")

names(acs)
```

Question 3

Using the same data frame you created in the previous problem, what is the equivalent function to `unique(acs$AGEP)`

(i) `sqldf("select unique AGEP from acs")`

(ii) `sqldf("select distinct pwgtp1 from acs")`

(iii) `sqldf("select AGEP where unique from acs")`

(iv) `sqldf("select distinct AGEP from acs")`

R commands for working with Text

Here are a small selection of useful commands for working with text

The nchar() function

The nchar() command returns the number of characters in the argument.

```
> string=c("kevin")
> nchar(string)
[1] 5
> nchar(1001)
[1] 4
```

The grep() function

The grep() command returns the location of a string that contains a specified substring, from a character vector. If you specify the additional argument "value=T", it will return that string.

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
>
> class(names(iris))
[1] "character"
> grep("Petal",names(iris))
[1] 3 4
>
> grep("Petal",names(iris),value=T)
[1] "Petal.Length" "Petal.Width"
>
```

The `paste()` function

This command creates a string of specified components.
The default is to have whitespace between each component. This can be removed with the additional argument `sep=""`.

```
> x=5
> paste("file",x,".csv")
[1] "file 5 .csv"
>
> paste("file",x,".csv",sep="")
[1] "file5.csv"
```

```
> filenm(2)
[1] "file 2 .csv"
```

The `gsub()` function

The `R` command `gsub()` is used to replace a character or piece of text with another in a specified string

```
> string=c("kevin")
> gsub("k","s",string)
[1] "sevin"
```

The `sprintf()` function

This command returns a character vector containing a formatted combination of text and variable values. The structure of the command is `sprintf(format, input)`.

```
> sprintf("%f", pi)
[1] "3.141593"
> sprintf("%.3f", pi)   # 3decimal places
[1] "3.142"
> sprintf("%1.0f", pi)   # no decimal places
[1] "3"
> sprintf("%5.1f", pi)  # 5 characters with whitespace
[1] "  3.1"
> sprintf("%05.1f", pi)  #5 characters no whitespace
[1] "003.1"
> sprintf("%+f", pi)
[1] "+3.141593"
```

To express asingle or double digit character integer as three character number

```
> x = 4
> sprintf("%03d", x)
[1] "004"
>
> x=40
> sprintf("%03d", x)
[1] "040"
```

For character data (i.e. strings)

```
> sprintf("%s %d", "test", 1:3)
[1] "test 1" "test 2" "test 3"
```

N.B *s* for string and *d* for integers.

The `readLines()` function

This command is used to read some or all text lines from a connection (i.e. an internet connection or Database connection

The `list.files()` function

This command is used to produce a list of files in a specified directory.

```
getwd() # working directory
list.files(getwd()) # List all files in working directory
```

Question 4

How many characters are in the 10th, 20th, 30th and 100th lines of HTML from this page:

`http://biostat.jhsph.edu/~jleek/contact.html`

(Hint: the `nchar()` function in `R` may be helpful)

(i) 43 99 8 6

(ii) 45 31 7 31

(iii) 43 99 7 25

(iv) 45 31 7 25

(v) 45 0 2 2

(vi) 45 31 2 25

(vii) 45 92 7 2

Question 5

Read this data set into R and report the sum of the numbers in the fourth column.

https://d396qusza40orc.cloudfront.net/getdata%2Fwksst8110.for

Original source of the data:

http://www.cpc.ncep.noaa.gov/data/indices/wksst8110.for

(Hint this is a fixed width file (fwf) format)

(i) 32426.7

(ii) 35824.9

(iii) 222243.1

(iv) 36.5

(v) 28893.3

(vi) 101.83

Find out about fixed width files

```
help(read.fwf)
```

```
## Data Saved in Local Directory
## as "DSS3wk5q5.for"

data <- read.csv("./DSS3wk5q5.for", header = TRUE)
file_name <- "./DSS3wk5q5.for"
df <- read.fwf(file=file_name,
    widths=c(-1,9,-5,4,4,-5,4,4,-5,4,4,-5,4,4), skip=4)

## Carry Out usual Data Frame Inspection Procedures
```