Lab 3E - Scraping web data

Directions: Follow along with the slides and answer the questions in **bold** font in your journal.

## The web as a data source

* The internet contains huge amounts of information.
* Using computers to gathering this information in an automated fashion is referred to as *scraping web data*.
* Scraping data from the web can be difficult because each website displays & stores data differently.
* In this lab, we'll learn how to scrape data in two steps:
* Step 1: Gather information from the web.
* Step 2: Clean it up and turn it into a usable data frame for Lab 3F.

## Our first web scraper

* Copy and paste the link below into a web browser to view the website of data we'd like to *scrape* and analyze.

<http://gh.mobilizingcs.org/ids_labs/extras/webdata/mountains.html>

* **Briefly describe what the data on the website is about.**
  + **Then write down 3 questions you'd be interested in answering by analyzing this data.**

## HTML

* HTML is the code that's used to render every website you've ever visited.
* The following slide shows the HTML code used to create the first two rows of the web data.
  + **How is the data table in HTML different than the data tables we're used to seeing in R, for example, when we use the View() function?**
  + **What do you think the *tags* <TABLE>, <TR>, <TH>, <TD> mean? How does HTML use these *tags* to display the table?**

## 

<TABLE>  
 <TR>  
 <TH>peak</TH>  
 <TH>range</TH>  
 <TH>state</TH>  
 <TH>long</TH>  
 <TH>lat</TH>  
 <TH>elev\_ft</TH>  
 <TH>elev\_m</TH>  
 <TH>prominence\_ft</TH>  
 <TH>prominence\_m</TH>  
 <TH>rank</TH>  
 </TR>  
 <TR>  
 <TD>Denali (Mount McKinley)</TD>  
 <TD>Alaska Range</TD>  
 <TD>Alaska</TD>  
 <TD>-151.0063</TD>  
 <TD>63.0690</TD>  
 <TD>20236</TD>  
 <TD>6168</TD>  
 <TD>20174</TD>  
 <TD>6149</TD>  
 <TD>1</TD>  
 </TR>  
</TABLE>

## Get to scraping!

* Use your browser to go back to the website with the data we're interested in scraping.
* Find the URL address for the site and assign it the name data\_url in R.
  + Then fill in the blanks below to have R scrape *every* web table available on the site:

tables <- readHTMLTable(\_\_\_\_)

## Find our data

* Since readHTMLTable() scrapes *every* table that is on a particular web URL, we need to find out which table has the data we're interested in.
  + For example, wikipedia.org often has articles with 3 or more tables.
  + This means we need to check all 3 tables to find the one we're interested in.
* Use the length() function to find out how many tables of data were scraped in our set of tables.

## Saving tables

* Now that we know how many tables we've scraped, we can go back and scrape individual tables by adding the which argument to the readHTMLTable() function.
  + Use readHTMLTable() to re-scrape the data from the web but this time use the which argument to scrape just the individual table.
  + The which argument should be the integer denoting which table you want scraped.
  + Assign the scraped data the name mtns

## From scraping to cleaning

* Data scraped from the web usually needs to be cleaned.
* **Run the following commands and compare the names of the variables. Do you notice any differences?**

View(mtns)  
names(mtns)

* **Which variables in your data are numerical variables and which are factors (i.e. categorical variables)?**
* Put your data in the str() function to see how R classified each variable.
  + **Which variables are wrong?**

## Fixing variable types

* View the mtns data and notice the order of the variables.
  + Use the order of the variables to fill in the blanks below with either the word "factor", if the variable is *categorical*, or "numeric", if the varible is *numerical*.

var\_types <- c("\_\_\_","\_\_\_","\_\_\_","\_\_\_","\_\_\_",  
 "\_\_\_","\_\_\_","\_\_\_","\_\_\_","\_\_\_")

* Finally, re-scrape the data and include colClasses = var\_types as an argument.
  + Don't forget to save the data as mtns and specify which table to scrape.

## Fixing variable names

* View the mtns data and notice the order of the variables.
  + Then use the order of the variables and the following code template to change the names of the mtns data.
  + Replace each "new\_name" with the actual name of the variable.
  + Make sure to include all of the variable names and order them correctly.

names(mtns) <- c("new\_name", "new\_name",  
 ..., "new\_name")

## Check, save and use!

* After scraping and cleaning the data, the only thing left to do is to save it and use it.
  + Before saving, use the names() and str() functions on last time to make sure the variable names and types are correct.
* Fill in the blanks to save the data and give it a file name

save(\_\_\_\_, file = "\_\_\_\_.Rda")

* **What is the mean and standard deviation of elev\_ft?**
* **Which state has the most mountains in our data?**