Assignment 2

edelsonc

August 28, 2016

This exercise will deal with extensible markdown files (xmls). Xml files are very similar to html files in that they have a tag structure that encasulates information. However, as opposed to html, xml allows users to create their own tags, hence deriving its name of extensible. Because of this, a lot of information can be gleaned from xml files.

Downloading xmls

For this exercise, an xml file from Reed college was used. This was downloaded using the now familar download command:

```r
# the url will be saved and used to both download the xml and recorded the data
# providence
xml_url <- "http://www.cs.washington.edu/research/xmldatasets/data/courses/reed.xml"
xml_file <- "~/Desktop/Data_Science/EDA/assignment_2/reed.xml"
download.file(xml_url, xml_file)

# Data providence is recorded and saved in a separate text file for later reference
ReedCoursesProvidence <- <- paste("Downloaded from", xml_url, "At", Sys.time(), sep = " ")
write(ReedCoursesProvidence, "~/Desktop/Data_Science/EDA/assignment_2/ReedProvidence.txt")
```

Now the data is ready to be parsed.

Parsing the xml

Since xml is such a prevelant file type, unsurprisingly there exists a library for dealing with it named XML. XML comes with a number of useful functions, such as the `xmlToList`, which we can use to convert our xml file into an R list.

```r
# load the XML library to be used to convert out new xml file to a list
library("XML)
xml_data <- xmlToList(xml_file)
```

However, this has a problem. This list contains a number of entries which are `NULL`. `NULL` values are automatically dropped from vectors or matrices, making converting our new list into a matrix or dataframe difficult. To avoid this, we'll instead use the `xmlToDataFrame` function, which handles these tricky cases for us

```r
xml_dataframe <- xmlToDataFrame(xml_file)
```

Counting Values

Counting fields is a common problem in most programming languages. Unsuprisingly, `R` has a number of useful functions to count fields. The first of these is `length`. It returns the length of a given vector or list.

```r
length(xml_data)  # returns 703
length(xml_dataframe$title) # also returns 703
```

If we want to know how many unique instances of a field there are, there is a simple function to allow that, `unique`. `unique` returns a list or vector where it has removed all replicate enteries. Thus we can use `length` and `unique` together in order to learn how many distinct courses are offered at Reed College

```r
# using the unique command with the length first creates a list of titles only
# containing unique values and then returns the length of the list
num_courses <- length(unique(xml_dataframe$title))  # returns 394
```

Counting the number of courses with a `NULL` instructor is a little more difficult. When `xmlToDataFrame` was applied to our reed.xml file, it took empty fields (ie `""`) and directly stuck them into the dataframe as is. Therefore, to learn how many courses has no instructor listed, we must count the number of instructors listed as `""`.

A simplistic way of accomplishing this is to use the `sapply` function. This function takes a vector or a list and applied a function to every element of the list. Using it, we can check if an element is `""` and then use the builtin `sum` to count how many times it occured

```r
# use the sapply function to return a boolean vector
isnull <- sapply(xml_dataframe$instructor, is.element, el="")

# applying sum to a boolean returns the number of TRUEs since TRUE=1
empty_instructor <- sum(isnull)  # returns 15
```

Finally, the number of distinct instuctors can be aquired in the same way as distinct courses, making sure to subtract 1 for the empty field

```r
num_instructors <- length(unique(xml_dataframe$instructor)) - 1  # 135
```