STAT 19000 Project 5

Topics: Introduction to XML

Motivation: Data comes in many formats, usually in a messy and not pre-processed way. Understanding different formats, and how to obtain information from them is crucial. Getting the data is the first step in analyzing it!

Context: We've been working with a variety of data sources/types. Here we will focus on parsing a format commonly used in web pages, XML.

Scope: Understand XML features, how parse and analyze XML using R.

You can find useful examples that walk you through relevant material here or on scholar: `/class/datamine/data/spring2020/s`
It is highly recommended to read through these to help solve problems.

Use the template found here or on scholar: `/class/datamine/data/spring2020/stat19000project05template.ipynb` to submit your solutions.

Important note: Make sure you have your output calculated and displayed inside your notebook prior to submission.

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms that will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online. You are not required to utilize all the given keywords. You will receive full points as long as your code gives the correct result.

Don't forget the very useful documentation shortcut `?`. To use, simply type `?` in the console, followed by the name of the function you are interested in.

You can also look for package documentation by using `help(package=PACKAGENAME)`.

Sometimes it can be helpful to see the source code of a defined function. To do so, type the function's name without the `()`.

Question 1: understanding XML

1a. *(1 pt)* What is XML and what does it stand for?

Hint: *For (1a) & (1b) the links provided in the examples are a good place to start. A search on the web never hurts as well. Make sure that if you do choose to use any code off of a website,* ***document*** *it. Show where and when you got the code.*

1b. *(1 pt)* Name two differences between XML and HTML.

Question 2: parsing XML in R

Pick your favorite XML package (`XML` or `xml2`) to work with for the following questions. You do *not* need to provide solutions using both packages!

2a. *(2 pt)* Scrape the NASA information located here using either the `xmlParse()` function from the `XML` package or the `read_xml()` function from the `xml2` package. What is the name of the root node? How many children does it have?

Keywords (XML package): *xmlParse, xmlRoot, xmlName, xmlSize*

Keywords (xml2 package): *read_xml, xml_root, xml_name, xml_length*

2b. *(2 pt)* Are the number of children (sub-nodes) constant (the same) for the root's first 10 sub-nodes? What type of information is the NASA xml providing us? In other words, the NASA xml is providing us with a collection of _________ about (or on the subject of), for example, astronomy.

Hint: *Take a look at what information we have for a child of the root, in addition to the root's name.*

Hint: *If you use the `XML` package, using `xmlChildren` is optional to solve this.*

Keywords: *sapply, for loop*

Keywords (XML package): *xmlChildren, xmlName, xmlSize*

Keywords (xml2 package): *xml_children, xml_name, xml_length*

Question 3: exploring NASA

Pick your favorite XML package (`XML` or `xml2`) to work with for the following questions. You do *not* need to provide solutions using both packages!

3a. *(2 pt)* Now that we have a better understanding of the data we are looking at, let's explore it a bit. Make a function that, given a specific node (a child from the root), calculates and returns the number of words in the "title" tag. Leveraging the technique you used to count the words in movie titles from project 02 (1b), may be useful. Use this function (and our suite of apply functions) to calculate the number of words in each title. Print the title that contains the greatest number of words.

Hint: *Begin by figuring out how you would do this for the first child node. Then extend it to a function. Take a look at our examples for more ideas.*

Keywords: *function, strsplit, length, sapply, which.max*

Keywords (XML package): *xmlChildren (from 2b), xmlElementsByTagName, xmlValue*

Keywords (xml2 package): *xml_children (from 2b), xml_find_all, xml_text*

3b. *(2 pt)* What information would you like to know about the NASA XML file? Make a data.frame called `myDF` containing two variables extracted from the NASA XML file. Find an interesting fact related to the variables in your new `myDF` data.frame (similar to what was done in (3a)).

Include your code (with comments) as a part of your solution. Write 1-2 sentences about what you hoped to learn from the data, explain what you chose to extract, and include any other information you think would be interesting or useful to the reader.

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/fCMGyr-V using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .ipynb and .pdf formats. We've updated our instructions to include multiple ways to convert your .ipynb file to a .pdf on scholar. You can find a copy of the instructions on scholar as well: `/class/datamine/data/spring2020/jupyter.pdf`. If for some reason the script does not work, just submit the .ipynb. Make sure you have your output calculated and displayed inside your notebook prior to submission.