# post02-jean-ji

*Jean Ji*

*November 30, 2017*

# Learn how to extract and manipulate HTML tables: a demo with the US' natural gas prices data

## Introduction

As our stat133 class comes to an end, I reflect on the Data Analysis Cycle that was introduced to us during the first lecture. To me, a lot of the coding knowledge and R packages that we learned this semester revolved around preparing us to go through the three stages of this cycle. From data preparation to actual analysis, we needed to clean and wrangle our data into accessible formats. Today, I would like to expand on our skills in data preparation and discuss in-depth about HTML table manipulations. This skill is going to help us when we encounter interesting data on websites and would like to analyze or build visualizations for it. In this post, I am going to elaborate on the functions in the "XML" package that was presented in-class. I will also extract HTML tables and demonstrate the ropes for manipulating these tables.

## Background

Before I dive deep into the weeds of HTML table manipulations, I want to review some of the knowledge that we obtained from Dr. Sanchez's lectures on XML. First of all, to appreciate the value of HTML and its power in creating websites, we need to understand the nature of XML languages. XML (eXtensible Markup Language) is defined as a sequence of characters or other symbols inserted at certain places to indicate how the content should be displayed. There are sets of rules for encoding documents in XML, such that the documents are both human-readable and machine-readable. The beauty of XML syntax lies in the duality in this human-machine interface.

An example of hierarchical data in XML

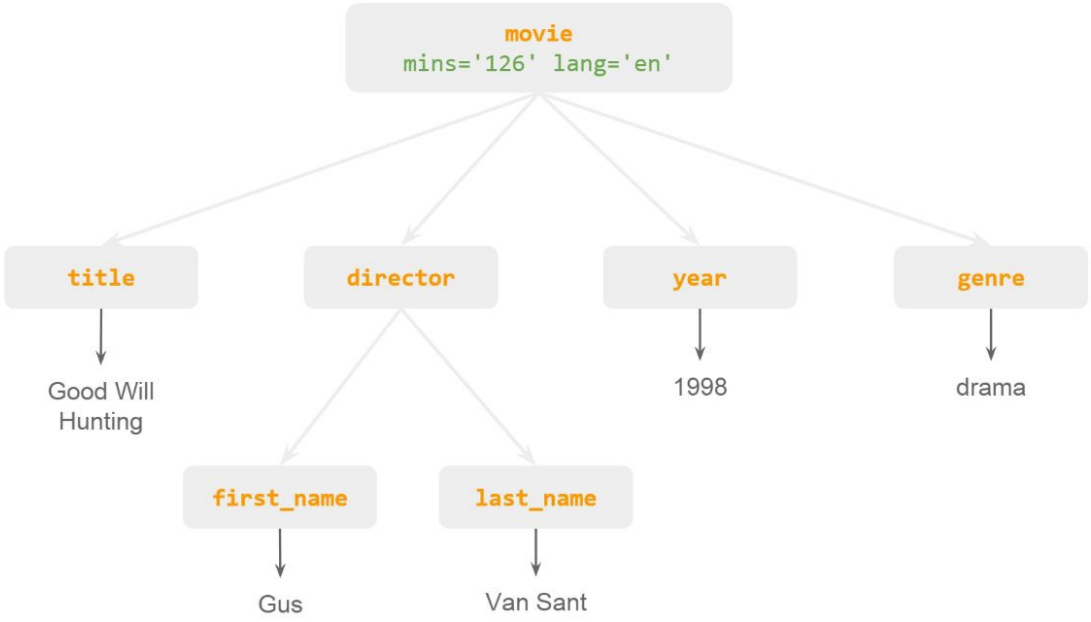
```
<family>
  <parent gender="male" name="John" age="33" />
  <parent gender="female" name="Julia" age="32" />
  <child gender="male" name="Jack" age="6" />
  <child gender="female" name="Jill" age="4" />
  <child gender="male" name="John jnr" age="2" />
</family>
<family>
  <parent gender="male" name="David" age="45" />
  <parent gender="female" name="Debbie" age="42" />
  <child gender="male" name="Donald" age="16" />
  <child gender="female" name="Dianne" age="12" />
</family>
```

Example of XML syntax from lecture

Similar in conventions to the R Markdown language, a well-formed XML document obeys the following syntax rules, including properly-nested elements, self-closing tags, attributes appearing in the start-tags of elements, and the values of attributes being quoted. These rules ensure the functionality of XML which enables people to build websites that contain information in a variety of formats, including tables, images and texts.

Speaking of a diversity of formats to present information, HTML (Hyper Text Markup Language) is a special markup language in the XML family which allows website developers to build website contents that often include HTML tables. Because of the valuable information that these tables contain, data analysts extract them from their HTML format and manipulate them in data-processing software, such as R.

However, one can ask the question, "how does this work?", "what truly goes on behind a simple R command to read a HTML table?" The secret process that happens behind the software console is called parsing. A parser is a software component that takes input data, in this case would be a website URL, and builds a data structure for the input, which is an XML tree. Below is an example of a simple XML tree, where we can see the hierarchy of each node and their relationships with each other.




Example of an XML tree from lecture

From the XML tree, we can locate nodes and extract information from them; this reflects the real power of parsing. Similar to navigating the file paths and accessing different folders in our homework's designated file structure, we can do the same thing to these nodes in a XML tree. XPath is an arugment in a family of "html_()" functions of the XML package that uses path expressions to select nodes in XML documents based on specified node names, contents, and their relationships with other nodes. With the ability to extract the precise information that we need, we can read HTML tables into R by identifying their node and extracting them.

In the following sections, I am going to speak more about the "XML" package in R and the functions that can help us extract HTML tables for data analyses.

# Examples

To illustrate how to read in a HTML table, I decided to use the natural gas prices data from the United States' Energy Information Administration's (EIA) website. The prices of natural gas are crucial to determining the wholesale prices of electricity, as we currently rely on natural gas power plants to provide the baseload of our electricity consumption as well as when our peaking demands.

EIA has an Open Data platform where researchers can access different types of data and import them into their software for analyses. In my example, I am going to use the annual average prices of natural gas in the United States from 1997 and 2016. The "readHTMLtable ()" function conveniently extracts data from HTML tables via reading the URL and storing each table as a data frame. The "which" argument allows me to subset the dataframe that I desire. Other handy arguments of "readHTMLtable ()" include specifying column names, classes, and rows to skip. After reading the HTML table into R, the next steps involve processing the table including renaming the variables. Below is my code chunk for loading the HTML table and the sources of my data.

```
#Downolad the html to my working directory
url <- "https://www.eia.gov/opendata/qb.php?category=461217&sdid=NG.N3035US3.A"
download.file(url, destfile = 'natural-gas-price-summary.html')

#Import the HTML table
library(XML)
table1 <- readHTMLTable('natural-gas-price-summary.html', which = 1,
                        header = c("Series Name", "Period","Frequency","Value","Units"))
```

Code chunk for loading the HTML table and creating a data frame

After converting the table into an R object we can manipulate it in order to understand the trends in natural gas prices over this 20-year period. I chose to visualize the data with a scatterplot and a bar plot. The scatterplot shows the year and the price as the x- and y-axis, respectively.

```
#Ploting the time-series price data of natural gas
library(ggplot2)
ggplot(data = table1, aes(x = V2, y = V4)) + geom_col(fill = "red") + labs(x = "Year", y = "Prices")
ggsave(filename = "bargraph.jpg")
```

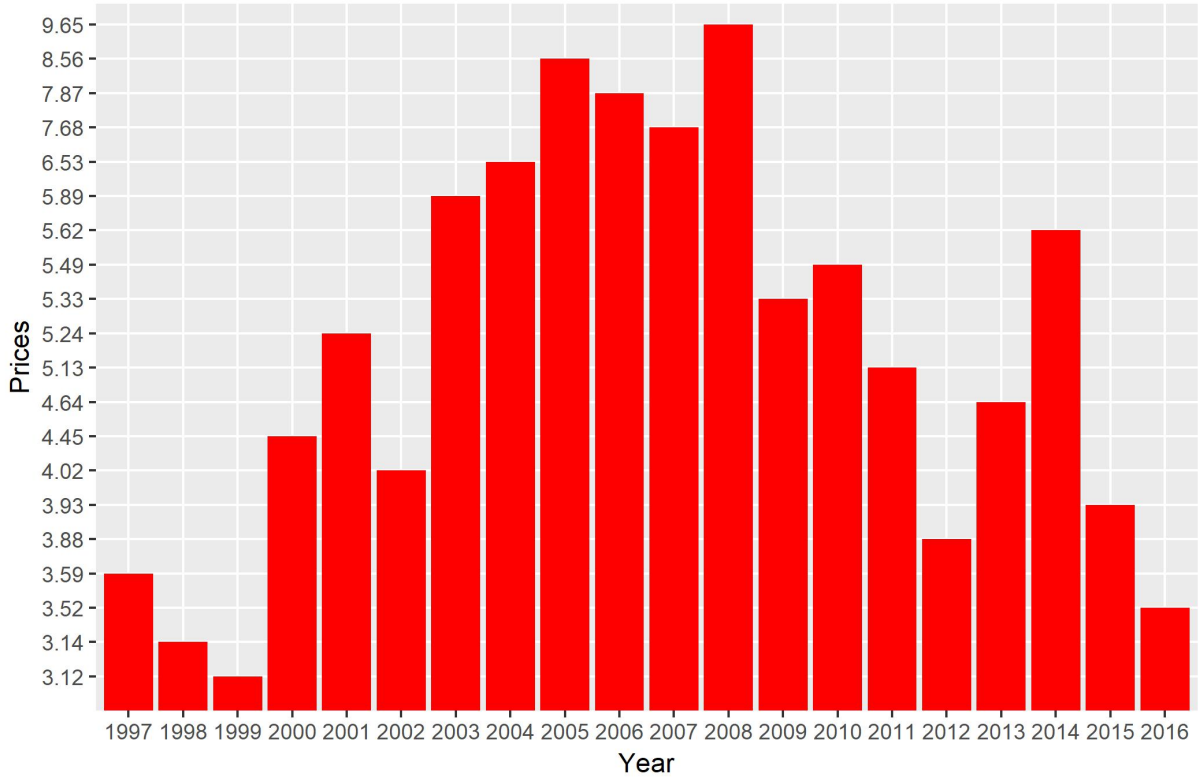
Code chunk for creating the barplot

```
#Visualizing the trend of the natural gas prices data
ggplot(data = table1, aes(x = V2, y = V4)) + geom_point(color = "blue") + labs(x = 'Year', y = 'Prices')
ggsave(filename = "scatterplot.jpg")
```

Code chunk for creating the scatterplot

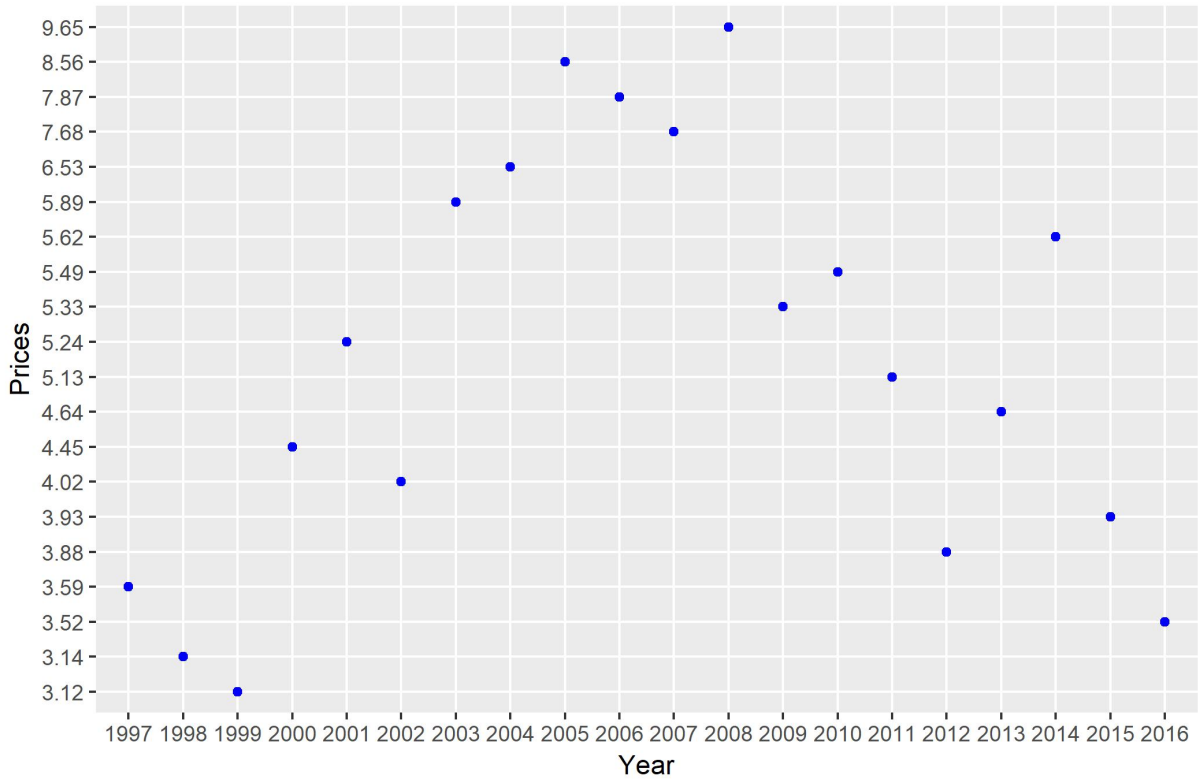
The bar plot visualized the natural gas prices per year and allows me to easily identify the years when natural gas

prices are high or low. From this graph I saw that the average annual price of natural gas was the highest at $9.65 per cubic foot in 2008, and the lowest at $3.12 per cubic foot in 1999. Furthermore, the recent annual average natural gas price is $3.52 per cubic foot, which is comparable to the lowest price in two decades. These findings become apparent to myself and to the audience through visual representation in a bar graph.




Bar graph of natural gas prices data

I have also represented the data in a scatter plot format to delineate the nature of one average annual price.




Scatter plot of natural gas prices data

# Discussion

While the display of the average annual natural gas prices was informative of the trend of prices throughout the last two decades, the generalization of the volatile nature of these prices into one average annual price is misleading. The limitation of this dataset emphasizes that the purpose of this post is to demonstrate the techniques of reading a HTML table into R and manipulating the data frame to create visual representation of the data. If I were to investigate the trends of monthly natural gas prices in the past two decades, I could apply the same coding techniques on a larger dataset and provide a more comprehensive analysis that reveals the volatility of natural gas prices.

# Conclusion

To recap the essences of this post, I first reviewed how XML and HTML store a variety of information including tables. Secondly, I presented a suite of techniques from R's "XML" package to extract HTML tables from websites and to manipulate these data frames. In adhering to the reproducibility guideline of this post requirements, I decided to extract natural gas prices data from the Energy Information Administration's publicly-available data source. The dataset that I selected provided average annual of natural gas prices in the United States for the past two decades. The bar graph and the scatter plot of this dataset allowed for comparisons of prices amongst these years, and allowed the reader to easily identify the years with high and low prices.

As we learned from the graphs, the natural gas prices are currently low, which has a significant influence on keeping many natural gas power plants in the electric generation business right now. As our electric generation mix becomes more renewable with wind energy and solar energy becoming cheaper, natural gas prices continue to drive the profit margins of these natural gas power plants. In order to analyze natural gas prices data in an efficient manner, it is extremely useful to learn how to extract this information from EIA's website via R.

# References


1) intro-to-strings Tutorial, URL: https://github.com/ucb-stat133/stat133-fall-2017/blob/master/tutorials/12-intro-to-strings.Rmd

2) Stack Overflow: scraping HTML tables into R data frames, URL: https://stackoverflow.com/questions/1395528/scraping-html-tables-into-r-data-frames-using-the-xml-package

3) Scraping HTML tables by using XML, URL: http://bradleyboehmke.github.io/2015/12/scraping-html-tables.html

4) Web scraping: simple tables, URL: http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/styled-4/styled-6/code-13/

5) Getting the data from the web with R, URL: https://github.com/gastonstat/tutorial-R-web-data/blob/master/03-xml-basics/03-xml-basics.pdf

6) Parsing XML and HTML contents, URL: https://github.com/gastonstat/tutorial-R-web-data/blob/master/04-parsing-xml/04-parsing-xml.pdf

7) EIA Average Annual Natural Gas Prices, URL: https://www.eia.gov/opendata/qb.php?category=461217&sdid=NG.N3035US3.A