Contents

This is from the second chapter of learn.r-journalism.com.

JSON stands for JavaScript Object Notation and is the data structure behind many website features like maps.

Let's say theoretically you were interested in compiling a list of all Sinclair Broadcast TV stations and their locations.

You’d first visit their web site.

And then you might find they have a map!
Look at the developer tools in your browser and click over to **Network** you could sort by size and see there’s a **json** file being called by the map.

If you click into the JSON link you’ll see this data structure that includes affiliation, call letters, and latitude and longitude.
Here’s a close up.

It looks like it could be transformed into rectangular data frame so we can analyze it.

```json

```

We’re going to use the `jsonlite`

First, install and load the package.

```r
#install.packages("jsonlite")
library(jsonlite)
```

Then point to where the JSON file is. You can use the URL or the local path to it if you’ve downloaded it. I recommend downloading it as a backup in case the website is restructured.
Use the `fromJSON()` function.

```r
json_url <- "http://sbgi.net/resources/assets/sbgi/MetaverseStationData.json"
```

```
## If the url above doesn't exist anymore uncomment the line below and run it
# json_url <- "data/MetaverseStationData.json"
```

```r
stations <- fromJSON(json_url)
```

Let's look at the structure of what we've imported.

```r
str(stations)
```

```
#' data.frame': 611 obs. of 26 variables:
## $ Call_Letter : chr  "KAAS" "KAAS-2" "KAAS-3" "KAAS-LD" ...
## $ Logo_List : chr  ""  "/resources/assets/sbgi/Logo_List-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_List-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_List-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_List-DEFAULT.jpg" ...
## $ Logo_Map : chr  ""  "/resources/assets/sbgi/Logo_Map-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_Map-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_Map-DEFAULT.jpg"  "/resources/assets/sbgi/Logo_Map-DEFAULT.jpg" ...
## $ Station : chr  "KAAS" "KAAS" "KAAS" "KAAS-LD" ...
## $ Channel : chr  "Primary" "Secondary" "Tertiary" "Primary" ...
## $ DMA : chr  "Wichita - Hutchinson, KS" "Wichita - Hutchinson, KS" "Wichita - Hutchinson, KS" "Wichita - Hutchinson, KS" ...
## $ DMA_Code : chr  "678" "0" "0" "678" ...
## $ DMA_Short : chr  "Wichita\_KS" "Wichita\_KS" "Wichita\_KS" "Wichita\_KS" ...
## $ DMA_Rank : int  67 67 67 67 67 67 31 31 31 195 ...
## $ Station_Status : chr  "O&O" "O&O" "O&O" "O&O" ...
## $ Station_Address : chr  "316 North West Street, Wichita, KS 67203"  "316 North West Street, Wichita, KS 67203"  "316 North West Street, Wichita, KS 67203"  "316 North West Street, Wichita, KS 67203" ...
## $ Station_City : chr  "Wichita" "Wichita" "Wichita" "Wichita" ...
## $ Station_State : chr  "KS" "KS" "KS" "KS" ...
## $ Station_Zip : int  67203 67203 67203 67203 67203 67203 78229 78229 78229 NA ...
## $ Station_Logo : chr  "sbg_noimage" "antenna" "comet" "sbg_noimage" ...
## $ Station_Phone_Number : chr  "316-942-2424" "316-942-2424" "316-942-2424" "316-942-2424" ...
## $ Station_Fax_Number : chr  "316-942-8927" "316-942-8927" "316-942-8927" "316-942-8927" ...
## $ Actual_RF_Channel : chr  "17" "17" "17" "31" ...
## $ News_Schedule_Weekday : chr  "" "" "" "" ...
## $ News_Schedule_Saturday: chr  "" "" "" "" ...
## $ News_Schedule_Sunday : chr  "" "" "" "" ...
## $ Location : chr  "Point (-97.388134 37.68888)" "" "" "" "" "" "" "" "" "" "" ...
```

And how's it now look as a data frame?

```r
View(stations)
```
Alright, this is a great start.

We can proceed to analyzing it and maybe visualizing it ourselves on a map.

But we’ll get to that in later chapters.

Also, it should be noted that JSON is rarely ever this clean.

I forget where this metaphor came from but consider your computer’s folder structure right now. How would you communicate the structure of your folders in a spreadsheet?

Tough, right? But possible when necessary.

So JSON is usually nested and messy. But there are ways of dealing with that.