breaklines=true

Rotten Tomatoes API

Matthew Gromer

April 15th, 2015

Introduction

Rotten Tomatoes is one of the largest and most well-regarded film review sites online, with only one other major challenger, Metacritic. Rotten Tomatoes reviews film with their own unique rating scale, "Fresh", represented by a ripe, red tomato, for good films, and "rotten", a green, splat-mark, for poor films. It garners nearly 26,000,000 unique page-views a month, and is one of the top 250 U.S. websites. It hosts a glut of data on films, actors, critics, as well as having trailers and local theater information.

Rotten Tomatoes, like many other websites, has offered their film information to the general public via their open API. This tutorial will walk you through the process of obtaining access to this API and using it, in this case to determine the average ratings of G- and R-rated current films.

Movies in the US all have guidance ratings, put in place using the Motion Picture Association of America's own system. This system rates a film on a scale, starting with "G", General Audiences, to "PG", Parental Guidance Suggested, to "PG-13", Parents Strongly Cautioned, to "R", Restricted, and ending with "NC-17", Adults Only.

These ratings are given based on the content of the film, as well as general plot and theme; films that are graphically violent, or sexual will be placed much higher on the scale than those that make efforts to reduce harsh visuals. Films that work to create a great sense of fear or terror will also find themselves placed higher on the scale, to warn parents against allowing younger children to view them.

Getting Access to the API

Like many sites that offer an API, Rotten Tomatoes requires a new prospective user to register with them first. To do this, you will need to visit http://developer.rottentomatoes.com/member/ API registration page, and provide some information, specifically, your information, what you intend to use the API for, and how much use you expect it to get. Rotten Tomatoes gathers this information so that they can determine how much load your API key might add to their service.

Once you've completed registration, you'll be sent a confirmation email. Once the link within is accepted, you'll be emailed an API key. An API key, or application programming interface key, is a unique string of characters that's used by a program or interface to identify who or what is calling on it for information. With this key, Rotten Tomatoes' interface will provide you with the information that you ask for. For this example, our key is as follows:

[label=APIKey,caption=API Key] pv8e8c9wkgkuwpsbpk72b8tg

We'll use this key in every interaction with the Rotten Tomatoes API. Make a note of your key, and do not lose it!

Obtaining the Data

Once you've aquired an API key, you can use it, along with a bit of R code, to obtain the information you're looking for. The Rotten Tomatoes API will, unfortunately, only provide 50 "pages" of information, which generally means 50 films/TV shows from a given query. To make a query using R, you'll need to use the R package "jsonlite", and the command "fromJSON".

The API documentation details several calls that can be made of their API, each pulling specific information, and each with various parameters that can be used to customize the call. For example, if you wanted to pull any information Rotten Tomatoes had on Toy Story 3, you would use the following API call:

[label=ToyStory3Info,caption=API Call] http://api.rottentomatoes.com/api/public/v1.0/movies/77067212

When the API key is added to this call, it completes it, creating this:

[label=ToyStory3Info,caption=API Request] http://api.rottentomatoes.com/api/public/v1.0/movies/77067

In the case of our example, we want to pull the ratings of movies that are currently out, alongside their MPAA ratings. To do this, we'll use the "DVD Current Releases" request.

```
> library(jsonlite)
> library(httr)
> library(utils)
> # fromJSON allows us to use the Rotten Tomatoes API call
> reviews <- jsonlite::fromJSON("http://api.rottentomatoes.com/api/public/v1.0/lists/dvds/cu
```

This gets us the data we need, a complete listing of all and the data that Rotten Tomatoes offers on them. It brings them down into a list object, which we'll work with. For our example, we'll want to parse out specific information to work with. We'll do that in the next section.

Scrubbing the Data

Now that we've got our data, we'll "scrub" it; scrubbing is a term for parsing, or formatting the raw data so that it becomes useful to our purposes. In this case,

we want to obtain three specific variables from the data: the MPAA rating, the critics' score, and finally, the audiences' score. These three variables will let us group films by rating, and then determine how both critics and audiences rated them. We'll start by pulling those variables from the list, by column number. The columns can be viewed quite easily by using the "View" command.

```
> # Place ratings & scores from the reviews object into their own object.
> mpaa.rating <- reviews[[2]][[4]]
> scores <- reviews[[2]][[8]]
```

Now, it is worth noting that this list is not simply a two-dimensional construct; it is a multi-dimensional one. Our variable, scores, now contains a data frame, with all the relevant rating information in it. We only need numeric values for our exercise, so we'll take them and build a new data frame.

```
> # Break apart the multi-dimensional scores field into individual objects.
> critics.score <- scores[,2]
> audiences.score <- scores[,4]
> # Combine the three objects back into a new data frame.
> reviews <- data.frame(mpaa.rating, critics.score, audiences.score)
```

Now that we have the specific variables we need in a friendly format, we can subset them further to include only G- and R-rated films. This will be done using the subset method:

```
> # Parse out unwanted films.
> reviews <- subset(reviews, mpaa.rating == "G" | mpaa.rating == "R")
```

At this point, we have only the data we need; the ratings of G- and R-rated films. From here, we can begin to interpret and explore the data.

```
   mpaa.rating critics.score audiences.score
1            R            95              86
3            R             8              54
4            R            84              74
5            R            78              72
6            R            75              70
7            R            93              77
8            R            69              50
10           R            93              79
13           R            53              50
15           R            82              52
```

Exploring the Data

Now that we have the data we need, we can begin to examine and work at it. We'll start with some basic tests.

```
> class(reviews)

[1] "data.frame"

> str(reviews)

'data.frame':        24 obs. of  3 variables:
 $ mpaa.rating    : Factor w/ 5 levels "G","PG","PG-13",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ critics.score  : int  95 8 84 78 75 93 69 93 53 82 ...
 $ audiences.score: int  86 54 74 72 70 77 50 79 50 52 ...

> summary(reviews)

  mpaa.rating critics.score   audiences.score
 G      : 1   Min.   : 6.00   Min.   :28.00
 PG     : 0   1st Qu.:63.00   1st Qu.:52.00
 PG-13  : 0   Median :81.00   Median :68.00
 R      :23   Mean   :71.33   Mean   :64.83
 Unrated: 0   3rd Qu.:90.00   3rd Qu.:74.75
              Max.   :97.00   Max.   :95.00
```

We can see that we've a single data frame object, reviews, with three columns, mpaa.rating, critics.score, and audiences.score, and 24 rows. The mpaa.rating column is a factor that uses string data, with four levels: "G", "PG", "PG-13", and "R", with no minimum. There are 23 R-rated films, and 1 G-rated film, which is an unfortunate ratio, but due to limitations in the API, it is the most we're going to come up with until more G-rated movies make it big.

Both critics.score and audiences.score are ints, but they deviate from there. The column critics.score has a minimum of 6, a median of 81, and a max of 97. The column audiences.score have a minimum of 28, a mean of 64.83, and a maximum of 95.

Graphing the Data

It's at this point that we fully flesh out our question; which have better ratings, G-rated or R-rated films. To do this, we will take the average of both critics' and audiences' scores, for both G- and R-rated films, and compare them.

```
> noGFilms <- length(reviews[which(reviews$mpaa.rating=="G"),2])
> noRFilms <- length(reviews[which(reviews$mpaa.rating=="R"),2])
> audiencesGScore <- reviews[which(reviews$mpaa.rating=="G"),3]
> audiencesRScores <- reviews[which(reviews$mpaa.rating=="R"),3]
> criticsGScores <- reviews[which(reviews$mpaa.rating=="G"),2]
> criticsRScores <- reviews[which(reviews$mpaa.rating=="R"),2]
> aveAudienceGScore <- round(sum(audiencesGScore) / noGFilms, digits=2)
> aveCriticGScore <- round(sum(criticsGScores) / noGFilms, digits=2)
> aveAudienceRScore <- round(sum(audiencesRScores) / noRFilms, digits=2)
```

```
> aveCriticRScore <- round(sum(criticsRScores) / noRFilms, digits=2)
>
```

From the data here, we can see that the average critic's review of G-rated movies as a whole is 80%, and R-rated movies as a whole, 70.96%. The average audience's review of G-rated movies as a whole is 70%, and R-rated movies as a whole, 64.61%. This data can be clearly seen below:

```
  Critic Audience
G  80.00    70.00
R  70.96    64.61
```

Figure 1: Figure 1.1 is a bar graph showing the MPAA rating along the X axis, with the score along the Y, grouped by the "rater", or group that did the rating.

Figure 2: Figure 1.2 is a line graph showing the MPAA rating along the X axis, with the score along the Y, grouped by the "rater", or group that did the rating.

Both graphs show rather clearly that the overall ratings of G-rated movies are slightly higher than that of R-rated ones.