Lab K: Leveraging APIs

Due April 17, 9 pm

1 Introduction

This lab is inspired by one from the course STAT679: Computing for Data Science and Statistics taught by Keith Levin at the University of Wisconsin.

In this lab we will be working with an API maintained by NASA for retrieving information about near earth objects (NEOs), asteroids that pass close to Earth. The documentation is available at:

https://api.nasa.gov/

To start, you'll need an API key to access the NASA data. You can get one at https://api.nasa.gov/. You can supply your York email address to sign up.

The Asteroids NeoWs Feed provides users with data about Near Earth Objects. You can access the NeoWs Feed using the Feed API is accessible at:

https://api.nasa.gov/neo/rest/v1/feed

The feed requires that you supply three URL parameters: start_date, end_date and an api_key. The parameters start_date and end_date specify the start and end of a date range, and should be formatted as follows: 'YYYY-MM-DD'. The API key will be the key that you are provided by NASA upon signing up.

Your first task will be to retrieve a JSON (Javascript Object Notation) object from the NASA NeoWs Feed API for January 1st, 2015. You can do this by setting start_date and end_date to '2015-01-01' and adding your API key to the mix. You will use a method called 'GET' to pass these parameters to the NASA API. GET parameters (also called URL parameters or query strings) are name-value pairs, separated by an equals sign =. An example that illustrates the structure of the call you will want to make is below:

https://api.nasa.gov/neo/rest/v1/feed?start_date=2015-01-01&end_date=2015-01-01&api_key=YOUR_API_KEY

Remember to replace YOUR_API_KEY with the API key you receive from NASA. If you place this URL in a browser you should receive a JSON response with the following attributes:

element_count: the number of near earth objects recorded between start_date and end_date.
near_earth_objects: a collection of JSON objects with information about each near earth object recorded between start_date and end_date.
links: these are URLs you can use to access information for the days before and after January 1st, 2015.

If you look closely at the the near_earth_objects attribute, you will see that this contains a collection of JSON objects. The attributes of these include 'estimated_diameter' and 'is_potentially_hazardous_asteroid' and many others.

To make an API call from within a Matlab script, use the method 'webread'. An example of this is in the labK.m file in your starter files. The call will look like this:

data=webread("https://api.nasa.gov/neo/rest/v1/feed?start_date=2015-01-01&end_date=2015-01-01&api_key=YOUR_API_KEY");

The attributes of the JSON object that is returned can be accessed using 'dot' notation, as follows:

data.element_count will give you the number of near earth objects recorded;
data.near_earth_objects will give you a 'struct' object that contains a collection of objects with information each near_earth_objects recorded during the date interval;
data.links will give you links to API calls for data from the day before and the day after.

To access information for a particular near earth object within the 'struct' **data.near_earth_objects**, we can write the following:

NEOs = struct2cell(data.near_earth_objects);
NEOs = NEOs{1};

These lines convert the 'struct' into a 'cell' array and returns the value within that cell. You can then access information about individual NEO objects as follows:

NEOs{1} will provide information about the first recorded near earth object;
NEOs{2} will provide information about the second recorded near earth object;
NEOs{3} will provide information about the third recorded near earth object;

and so on.

2 Programming Task

Your first task is write a function to retrieve all the near Earth objects on a particular day. More specifically, write a function called:

getNEOs(yyyy,mm,dd,API_KEY)

This will take three positive doubles as input: yyyy, mm and dd as well as a string API key, in that order. The parameters yyyy, mm and dd will encode a year, a month and a date, respectively, and the string API_KEY will be an API key. Your function should return the JSON object from the NASA NeoWs Feed for the specified day. *Hint: You can convert numbers to strings using the string method in Matlab, and you can concatenate strings using the '+' operator with strings as the operands.*

Then, write a function called:

hazardousNEOs(data)

that will take a JSON object from a NASA API call (data) and return a logical value (true or false). This function should cycle thru all the NEO objects in the input data (i.e. within the **data.near_earth_objects** collection) and determine if any single one of them is 'potentially hazardous'. If yes, your function should return true, else it should return false.

Finally and OPTIONALLY, write a function called:

hazardousAlert(a, yyyy, mm, dd, API_KEY)

This will take an arduino object (that points to your GROVE kit), a year, month and day (which will all be doubles), and an API_KEY (which will be a string). This function should sound the buzzer on your Grove Kit for a second if, and only if, there was a potentially hazardous NEO recorded on the day in question. You can play any frequency of tone that you like!

3 What to Submit

Submit, via eClass:

1. Your getNEOs.m file
2. Your hazardousNEOs.m file
3. Your hazardousAlert.m file (optional)

You will receive 1/4 mark for a working getNEOs script and 1/4 a mark for a well commented, logical getNEOs script. You will receive 1/4 mark for a

working hazardousNEOs script and 1/4 a mark for a well commented, logical hazardousNEOs script. Your code should follow the function template presented in class; your function should therefore be commented and there should be output when we type 'help' for any function you have written into the console. Your code should pass all of the tests in the file 'labKtest.m' as well as any additional tests that we may write!!

HAVE FUN AND GOOD LUCK!