Working With APIs

Introduction

API is an important source of data. API stands for Application Programming Interface and it is a system that allows applications communicate and share data with eac other. Our goal in this project is to extract solar resource data for New York city using the the data.gov API. This is an API that allows us access to dfferent datasets with information about the United States. The data we will be extracting has the following output fields:

- The Average Direct Normal Irradiance(avg_dni): Direct Normal Irradiance is the amount of solar radiation a surface receives per unit area.
- The Average Tilt at Latitude (avg_lat_tilt): Tilt at Latitude is the amount of radiation a surface receives per unit area that doesn't arrive on a direct path from the sun.
- The Average Global Horizontal Irradiance(avg_ghi): Global Horizontal Irradiance is the total amount of shortwave radiation a surface horizontal to the ground receives from above.

The end result of the project is to have a dataframe of the extracted data that we can easily work with. We are going to take the following steps to achieve this:

- Querying the API using `httr GET` function.
- Extracting the JSON content from the response gotten from the API and converting it to a complex list.
- The complex list will be converted to a dataframe.
- A function will be created that does all of the above tasks.
- Finally we are going to visualise the data.

```
# loading required libraries
library(tidyverse)
library(httr)
library(kableExtra)
```

```
# function to render tibbles as  pdf tables
render_table <- function(table, scale_down=F){
  if(scale_down == T){
    rendered_table <- kbl(table) %>% kable_styling(
      latex_options = c("stripe", "HOLD_position", "scale_down")
    )
  } else{
    rendered_table <- kbl(table) %>% kable_styling(
      latex_options = c("stripe", "HOLD_position")
    )
  }
  return(rendered_table)
}
```

Querying The API

To query the API, we need an API key and we also need to be familliar with the url end point and the parameters that will be required. The API key and the required parameters will be stored in a list.

```
api_key <- Sys.getenv("data.gov_api")
d_url <- "https://developer.nrel.gov/api/solar/solar_resource/v1.json"
params <- list(api_key = api_key,lat= 41,lon = -75)

response <- GET(d_url, query = params) # returns a JSON response from the query
status_code(response) %>% print()
```

```
## [1] 200
```

```
http_type(response) %>% print()
```

```
## [1] "application/json"
```

A status code of 200 means that our query was successful and the http type showed that we got a JSON response.

Extracting JSON Content

```
content <- content(response, "text") # extracts the response JSON
json_list <- jsonlite::fromJSON(content) # converts the JSON object into a complex list

str(json_list)
```

```
## List of 6
##  $ version : chr "1.0.0"
##  $ warnings: list()
##  $ errors  : list()
##  $ metadata:List of 1
##   ..$ sources: chr "Perez-SUNY/NREL, 2012"
##  $ inputs  :List of 2
##   ..$ lat: chr "41"
##   ..$ lon: chr "-75"
##  $ outputs :List of 3
##   ..$ avg_dni     :List of 2
##   .. ..$ annual : num 3.69
##   .. ..$ monthly:List of 12
##   .. .. ..$ jan: num 3.12
##   .. .. ..$ feb: num 3.36
##   .. .. ..$ mar: num 4.1
##   .. .. ..$ apr: num 4.07
##   .. .. ..$ may: num 4.15
##   .. .. ..$ jun: num 4.17
##   .. .. ..$ jul: num 4.6
##   .. .. ..$ aug: num 4.14
##   .. .. ..$ sep: num 4.02
##   .. .. ..$ oct: num 3.26
##   .. .. ..$ nov: num 2.58
##   .. .. ..$ dec: num 2.72
##   ..$ avg_ghi     :List of 2
##   .. ..$ annual : num 3.87
##   .. ..$ monthly:List of 12
##   .. .. ..$ jan: num 1.97
##   .. .. ..$ feb: num 2.69
##   .. .. ..$ mar: num 3.86
##   .. .. ..$ apr: num 4.7
```

```
##   .. .. ..$ may: num 5.45
##   .. .. ..$ jun: num 5.78
##   .. .. ..$ jul: num 5.98
##   .. .. ..$ aug: num 5.14
##   .. .. ..$ sep: num 4.23
##   .. .. ..$ oct: num 2.94
##   .. .. ..$ nov: num 1.99
##   .. .. ..$ dec: num 1.67
##   ..$ avg_lat_tilt:List of 2
##   .. ..$ annual : num 4.52
##   .. ..$ monthly:List of 12
##   .. .. ..$ jan: num 3.55
##   .. .. ..$ feb: num 4.04
##   .. .. ..$ mar: num 4.86
##   .. .. ..$ apr: num 4.97
##   .. .. ..$ may: num 5.18
##   .. .. ..$ jun: num 5.24
##   .. .. ..$ jul: num 5.58
##   .. .. ..$ aug: num 5.24
##   .. .. ..$ sep: num 5
##   .. .. ..$ oct: num 4.11
##   .. .. ..$ nov: num 3.26
##   .. .. ..$ dec: num 3.13
```

From the complex named list we have above, the data we need is stored under outputs which contains 3 more lists with the data we require. We can access the data in the lists using the names of the list as keys.

```
# using the list name as key to access data
output_list <- json_list$outputs
avg_dni <- output_list$avg_dni$monthly
avg_ghi <- output_list$avg_ghi$monthly
avg_lat_tilt <- output_list$avg_lat_tilt$monthly

print(avg_dni)
```

```
## $jan
## [1] 3.12
##
## $feb
## [1] 3.36
##
## $mar
## [1] 4.1
##
## $apr
## [1] 4.07
##
## $may
## [1] 4.15
##
## $jun
## [1] 4.17
##
## $jul
## [1] 4.6
```

```
##
## $aug
## [1] 4.14
##
## $sep
## [1] 4.02
##
## $oct
## [1] 3.26
##
## $nov
## [1] 2.58
##
## $dec
## [1] 2.72
```

Creating Dataframe From Lists

We were able to extract 3 lists of all the data we need, to convert these lists to a dataframe we are going to use the `tibble` function. We will also be adding a month column using the `month.abb` function.

```
df <- tibble(month = month.abb,avg_dni = avg_dni,
       avg_ghi = avg_ghi, avg_lat_tilt = avg_lat_tilt)

df %>% print()
```

```
## # A tibble: 12 x 4
##    month avg_dni      avg_ghi      avg_lat_tilt
##    <chr> <named list> <named list> <named list>
##  1 Jan   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  2 Feb   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  3 Mar   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  4 Apr   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  5 May   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  6 Jun   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  7 Jul   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  8 Aug   <dbl [1]>    <dbl [1]>    <dbl [1]>
##  9 Sep   <dbl [1]>    <dbl [1]>    <dbl [1]>
## 10 Oct   <dbl [1]>    <dbl [1]>    <dbl [1]>
## 11 Nov   <dbl [1]>    <dbl [1]>    <dbl [1]>
## 12 Dec   <dbl [1]>    <dbl [1]>    <dbl [1]>
```

The output of the avg_dni, avg_ghi and avg_lat_tilt columns in our dataframe is a list data type. to fix this we simply have to unlist those columns using `dplyr mutate` function.

```
df <- df %>% mutate(
  avg_ghi = unlist(avg_ghi),
  avg_dni = unlist(avg_dni),
  avg_lat_tilt = unlist(avg_lat_tilt)
)

df %>% print()
```

```
## # A tibble: 12 x 4
##    month avg_dni avg_ghi avg_lat_tilt
##    <chr>   <dbl>   <dbl>        <dbl>
```

```
##  1 Jan      3.12    1.97         3.55
##  2 Feb      3.36    2.69         4.04
##  3 Mar      4.1     3.86         4.86
##  4 Apr      4.07    4.7          4.97
##  5 May      4.15    5.45         5.18
##  6 Jun      4.17    5.78         5.24
##  7 Jul      4.6     5.98         5.58
##  8 Aug      4.14    5.14         5.24
##  9 Sep      4.02    4.23         5
## 10 Oct      3.26    2.94         4.11
## 11 Nov      2.58    1.99         3.26
## 12 Dec      2.72    1.67         3.13
```

Creating Function To Query API

The function we are going to create is going to query the API and return a dataframe.

```
nrel_api_json_get_df <- function(endpoint, queries = list()) {

  # Preparing the URL
  url <- modify_url("https://developer.nrel.gov", path = endpoint)

  # API requests
  response <- GET(url, query = queries)

  # Tracking errors
  if ( http_error(response) ){
    print(status_code(response))
    print(http_status(response))
    stop("Something went wrong.", call. = FALSE)
  }

  if (http_type(response) != "application/json") {
    stop("API did not return json", call. = FALSE)
  }

  # Extracting content
  json_text <- content(response, "text")

  # Converting content into Dataframe
  json_list <- jsonlite::fromJSON(json_text)
  output_list <- json_list$outputs
  avg_dni <- output_list$avg_dni$monthly
  avg_ghi <- output_list$avg_ghi$monthly
  avg_lat_tilt <- output_list$avg_lat_tilt$monthly

  df <- tibble(month = month.abb,avg_dni = avg_dni,
       avg_ghi = avg_ghi, avg_lat_tilt = avg_lat_tilt)

  df <- df %>% mutate(
  avg_ghi = unlist(avg_ghi),
  avg_dni = unlist(avg_dni),
  avg_lat_tilt = unlist(avg_lat_tilt)
)
```

```
  # Return the dataframe
  df
}
```

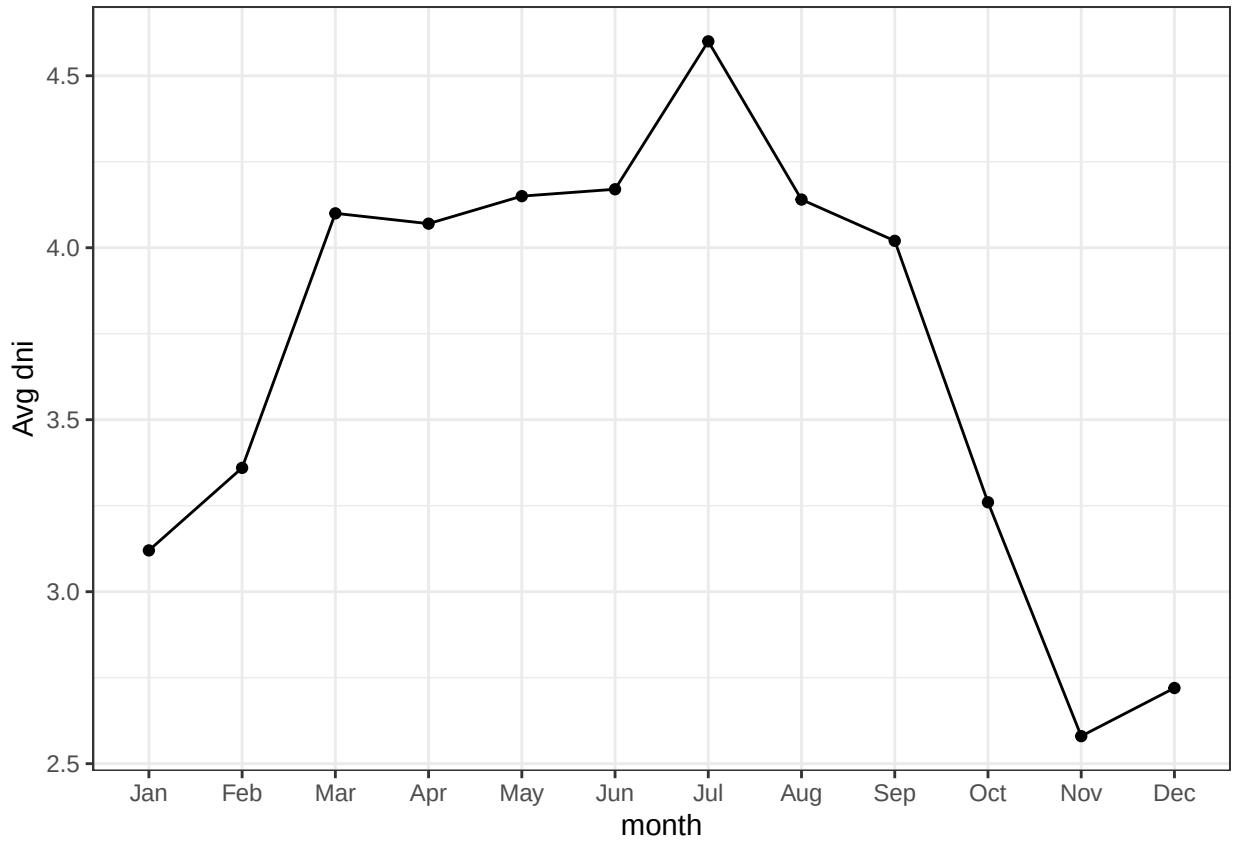
```
solar_resource_df <- nrel_api_json_get_df(endpoint = "api/solar/solar_resource/v1.json", queries = para

solar_resource_df %>% render_table()
```

month	avg_dni	avg_ghi	avg_lat_tilt
Jan	3.12	1.97	3.55
Feb	3.36	2.69	4.04
Mar	4.10	3.86	4.86
Apr	4.07	4.70	4.97
May	4.15	5.45	5.18
Jun	4.17	5.78	5.24
Jul	4.60	5.98	5.58
Aug	4.14	5.14	5.24
Sep	4.02	4.23	5.00
Oct	3.26	2.94	4.11
Nov	2.58	1.99	3.26
Dec	2.72	1.67	3.13

Visualising The Data.

```
solar_resource_df %>% mutate(
  month = factor(month, levels=month.abb))  %>% ggplot(
    aes(x= month, y = avg_dni, group = 1)
  ) +
  geom_line() +
  geom_point() +
  labs(y = "Avg dni") +
  theme_bw()
```



From the plot we can tell that the average direct normal irradiance(avg_dni) is seasonal. It is higher during the spring and summer months and lower during the fall and winter months.

Conclusion

This project showed a step by step approach to working with API as a data source. The goal was to extract data on New York Solar resource and we achieved that goal by:

- First taking a step by step approach From querying the API to creating a dataframe we can easily work it.
- Creating a function that follows the afformentioned steps.
- Lastly we visualised the data to get some insights.