New York Solar Resource Data

Eda AYDIN

13 05 2021

Contents

Introduction

Using APIs gives use access to incredible amount of data only available online. In this exercise, we want to extract New York City Solar Data. Such data can, allow us to determine on average the most productive periods of the year for solar panel deployment.

Finding the Suitable Endpoint and Parameters to Query the API

```
# Storing my api key in a variable
the_key  = "S2uaBTmPv1jAZdoZuh8kKzavKXsvKsjN4bQLB73r"

# Identify the API url
url <- "https://developer.nrel.gov/api/solar/solar_resource/v1.json"
```

```
# Specifying the necessary parameters to request the New York City solar data
parameters_list <- list(api_key= the_key, lat = 41, lon = -75)
```

Extracting the New York Solar Resource Data

```
# Loading the `httr` package
library(httr)
```

```
## Warning: package 'httr' was built under R version 4.0.5
```

```
# Using the `GET()` function to request the data from the API with `url` and `parameters_list`
response <- GET(url, query = parameters_list)

# Tracking errors
## Displaying the status code with the `status_code()` function
status <- status_code(response)
status
```

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

```
## Displaying the API response format
response_type <- http_type(response)
response_type
```

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

```
# Extracting the API response content as text
content <- content(response, "text")

# Displaying this content to check how it looks visually.
print(content)
```

```
## [1] "{\"version\":\"1.0.0\",\"warnings\":[],\"errors\":[],\"metadata\":{\"sources\":[\"Perez-SUNY/NRI
```

Parsing the JSON into R Object

```
# Parsing the "json_text" to a R object using the "jsonlite::fromJSON()" function
json_lists <- jsonlite::fromJSON(content)

# Displaying the structure of the R object using the "str()" function
str(json_lists)
```

```
## 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
```

How to Create a Dataframe from a Complex List

Building Dataframe from a Complex List

```
# Extracting the outputs data
outputs_list <- json_lists$outputs

# Extracting the monthly vector ("monthly") from the ("avg_dni") list in the outputs data
avg_dni <- outputs_list$avg_dni$monthly

# Extracting the monthly vector ("monthly") fron the ("avg_ghi") list in the outputs data
avg_ghi <- outputs_list$avg_ghi$monthly

# Extracting the monthly vector ("monthly") from the ("avg_lat_tilt") list in the outputs data
avg_lat_tilt <- outputs_list$avg_lat_tilt$monthly

# Combining the monthly vectors into a dataframe using the "tibble::tibble()" function
## Adding the "month" column containing month abbreviations: "Jan","Feb", ... "Dec"
dataframe <- tibble::tibble("month" = month.abb,
                            "avg_dni" = avg_dni,
                            "avg_ghi" = avg_ghi,
                            "avg_lat_tilt" = avg_lat_tilt)

# Dipslaying the dataframe
dataframe
```

```
## # 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]>
```

Extracting Dataframe from a Complex List:

```
# Extracting the outputs list
outputs_list <- json_lists$outputs

# Simplifying the outputs list
simplified_outputs_list <- unlist(outputs_list)
```

```
# Restructuring the simplified list into a matrix of 13 rows (the annual value and 12 months values)
data_matrix <- matrix(data = simplified_outputs_list,
                      nrow= 13)

# Removing the annual value from the data matrix
data_matrix <- data_matrix[-1,]

# Converting the matrix into a dataframe using the as.data.frame()
another_dataframe <- as.data.frame(data_matrix)

# Displaying the dataframe
another_dataframe
```

```
##      V1   V2   V3
## 1  3.12 1.97 3.55
## 2  3.36 2.69 4.04
## 3  4.10 3.86 4.86
## 4  4.07 4.70 4.97
## 5  4.15 5.45 5.18
## 6  4.17 5.78 5.24
## 7  4.60 5.98 5.58
## 8  4.14 5.14 5.24
## 9  4.02 4.23 5.00
## 10 3.26 2.94 4.11
## 11 2.58 1.99 3.26
## 12 2.72 1.67 3.13
```

Putting all together

```
library(httr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
the_key = "S2uaBTmPv1jAZdoZuh8kKzavKXsvKsjN4bQLB73r"

# Creating the custom "nrel_api_json_get_df()" function.
```

```r
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
  table_lst <- jsonlite::fromJSON(json_text)

  dataframe <- tibble::tibble("month" = month.abb,
                              "avg_dni" = as.numeric(table_lst$outputs$avg_dni$monthly),
                              "avg_ghi" = as.numeric(table_lst$outputs$avg_ghi$monthly),
                              "avg_lat_tilt" = as.numeric(table_lst$outputs$avg_lat_tilt$monthly))

  # Returning the dataframe
  dataframe
}

# Providing the "api/solar/solar_resource/v1.json" as the endpoint parameter
# Providing the parameters_list variable as queries parameter
solar_resource_df <- nrel_api_json_get_df("api/solar/solar_resource/v1.json", parameters_list)

# Printing the output dataframe
solar_resource_df
```

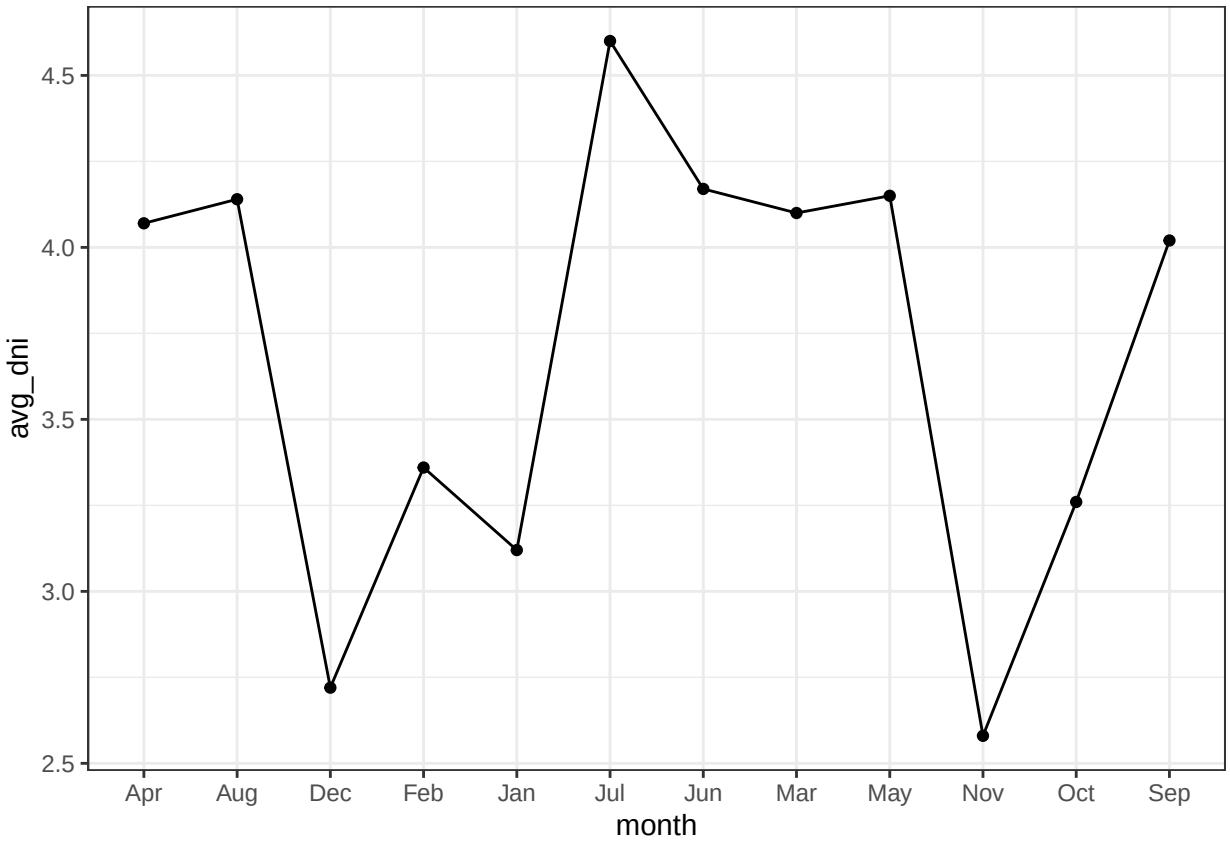
```
## # A tibble: 12 x 4
##    month avg_dni avg_ghi avg_lat_tilt
##    <chr>   <dbl>   <dbl>        <dbl>
##  1 Jan      3.12    1.97         3.55
##  2 Feb      3.36    2.69         4.04
##  3 Mar      4.1     3.86         4.86
##  4 Apr      4.07    4.7          4.97
##  5 May      4.15    5.45         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
```

Visualizing New York City Solar Resource Data

```
# Loading the ggplot2 and dplyr packages
library(ggplot2)
library(dplyr)

# Using the ggplot() function to plot the avg_dni from solar_resource_df
ggplot(data= solar_resource_df,
       aes(x = month,
           y = avg_dni,
           group = 1)) + geom_line() + geom_point() + theme_bw()
```




```
# Converting the "month" column into a factor using the following command
solar_resource_df <- solar_resource_df %>%
  mutate(month = factor(month, levels = month.abb))

# Replotting the "avg_dni" value for each month
ggplot(data= solar_resource_df,
       aes(x = month,
```

```
          y = avg_dni,
          group = 1)) + geom_line() + geom_point() + theme_bw()
```