# NY Solar Resource Data

Emmanuel Messori

12/09/2021

The solar resource data is available on the US open data website. To retrieve it we need an access key, freely available on demand.

In particolar we'll work with these 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.

## API connection

```
api_key <-"di7U1kMqN1orvvSS2efbg8tmprjRPQU2bvPnd0bd"
```

The next step is to determine the URL and the parameters to query the API. More info on the official documentation.

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

#NY latitude
lat <- 41
#NY longitutude
lon <- -75

params <- list(api_key = api_key, lat = lat, lon = lon)
```

## Retrieving the data

We will define a function to retrieve the information through the API:

```
library(httr)

solar_data_api <- function(u = url, q = params) {

  response <- GET(u, query = q)
```

```
  print(status_code(response))

  print(http_type(response))

  if (http_error(response)) {
    print(http_status(response))
    stop("Something wrong", .call = FALSE)
  }

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

  data <- jsonlite::fromJSON(json_content)

  data
}
```

Now we can retrieve the data :

```
data <- solar_data_api()
```

```
## [1] 200
## [1] "application/json"
```

```
str(data)
```

```
## 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
```

The resulting data is a nested list of lists. The 6 main lists contain information about the API, the call and the output.

## Processing the API response

We are interested in the information contained in the nested list `outputs`. This list contains in turn three lists of data:

```
str(data$output)
```

```
## 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
```

We will focus on the monthly values. Now we have to convert these lists into a dataframe.

```
values <- data$outputs
avg_dni <- unlist(values$avg_dni$monthly)
avg_ghi <- unlist(values$avg_ghi$monthly)
avg_lat_tilt <- unlist(values$avg_lat_tilt$monthly)
month <- month.abb

ny_sd <- dplyr::bind_cols(month = month, avg_dni = avg_dni, avg_ghi = avg_ghi,
                 avg_lat_tilt = avg_lat_tilt)


ny_sd
```

```
## # 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
```

Another approach of treating the raw data consist of simplifying the complex list and restructuring it into the desired output :

```
#1. Unlist the data
values <- unlist(values)

#2. Transform the resulting vector into a matrix of 13 rows (12 monthly values + 1 annual value)
mat <- matrix(values, nrow = 13)

#3. Convert the matrix into a dataframe
solar_data <- as.data.frame(mat)
colnames(solar_data) <- c("avg_dni", "avg_ghi", "avg_lat_tilt")

#4. Removing the annual values:
solar_data <- solar_data[-1,]
rownames(solar_data)<-NULL
```

## Visualising the data

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
theme_set(theme_bw())

ny_sd %>% pivot_longer(cols = 2:4, names_to = "measure") %>% ggplot(aes(factor(month, levels = month.abl
```