Air quality data and meteorological data: accesses, grabbing and cleaning

Hengfang Wang    Zhulin He    Zhengyuan Zhu

Iowa State University
CSSM

10/14/2016
1 Motivation

2 Access for Pollution Data

3 Access for Meteorological Data

4 Grabbing Data Online

5 Data Cleaning
Motivation

- Our project basically contains two parts:
  1. Predict PM$_{2.5}$ based on the meteorological variable via a proposed model (under construction)
  2. Use causal model to link the mortality rate and the above model, the technical details (structural nested mean models) have been covered by Dr. He’s lecture in our first meeting

- Mortality data from Gradiance cannot be accessed right now.
- We need to find air quality data and meteorology data by ourselves.
The size of datasets nowadays is becoming larger and larger, especially environmental data like air quality data and meteorology data, due to long time period and large number of monitors.

However, practically, getting such dataset might be time-consuming if one do not know the access to them.

Also, when there’s no access for such data, we need to write program to grab them on some target webpages regularly.

After grabbing such raw data from specific website, data-cleaning job is often required based on the purpose of project.
1 Motivation

2 Access for Pollution Data

3 Access for Meteorological Data

4 Grabbing Data Online

5 Data Cleaning
Access for Pollution Data

The AirData (from EPA) website gives you access to air quality data collected at outdoor monitors across the United States, Puerto Rico, and the U.S. Virgin Islands. The data comes primarily from the AQS (Air Quality System) database. You can choose from several ways of looking at the data:

1. Download data into a file (or view it on the screen)
2. Output the data into one of AirData's standard reports
3. Create graphical displays using one of the visualization tools
4. Investigate monitor locations using an interactive map

Here I just introduce how to implement 1 and 4 (briefly).
To download such air quality data, generally we need the following steps:

1. Go to https://aqs.epa.gov/api
2. Register for an account
3. Fill in the blanks for what you want and submit
4. Wait for the response of IE or Chrome (If No Notify query type was chosen) or just wait for the response from your e-mail then click the hyperlink (If Notify query type was chosen).
Notice that there’re two types of output we can get: one is raw data and the other is profile.

The raw data is what we usually regard as.

The purpose of ’profile’ service is to give users a ’profile’ of the available data in the system before they construct other queries to retrieve the data. It will returns a list of monitors that match users’ selection criteria and a profile of the data available from those monitors.
In order to maintain the server regularly operated, such service would not allow us require huge data at one time. For all sites in a specific state, the largest time period we can get is one year. For more limitations of data request and instruction to fill in the blanks, please refer to:
https://aqs.epa.gov/aqsweb/documents/gui_query.html

Also, you can write a script by accessing the webpage data in R or other softwares (will not be covered).

Now let’s have a try.
https://aqs.epa.gov/api
As for investigating monitor locations using an interactive map, we can try: https://www.epa.gov/outdoor-air-quality-data/interactive-map-air-quality-monitors

After arriving on the webpage above, we can launch the AirData Map App by clicking on the picture.

You can click on specific site to choose the download option.
1 Motivation

2 Access for Pollution Data

3 Access for Meteorological Data

4 Grabbing Data Online

5 Data Cleaning
We have a more user-friendly website to download meteorological data, i.e., Climate Data Online (CDO), which provides free access to NCDC’s archive of global historical weather and climate data in addition to station history information:

https://www.ncdc.noaa.gov/cdo-web/

Such data include quality controlled daily, monthly, seasonal, and yearly measurements of temperature, precipitation, wind, and degree days and 30-year Climate Normals.
Access for Meteorological Data (Con’t)

Generally, there’re three tools for us to get data in this webpage:

1. Search Tool (Recommended, will show how to use)
2. Mapping Tool (Not so stable as far as I tried)
3. Data Tool (More tools can be explored)
On webpage https://www.ncdc.noaa.gov/cdo-web/search, we need to fill in four fields to locate what data we want:

1. Weather observation type/dataset (e.g., daily summary, precipitation hourly, etc.)
2. Date range
3. Coverage level (Zip Codes, cities, states, etc.)
4. Search term (a location name or identifier, such as CA.)

If you want many state-level data for one time, a trick here is to type many states’ abbreviation at the same time such as ”CA, IA”.
Access for Meteorological Data (Con’t)

After clicking on search button, then just add the sites you want into your cart. The final step is to assure what you’ll get:

- Select the Output Format
- Select the Date Range
- Station Detail & Data Flag Options (Station name + Geographic Location)
- Select data types for custom output (meteorological data you’d like to get)
After reviewing what you’ve ordered, submit the order and wait for the response via your e-mail.

It usually take literally 1 ~ 1.5 day (actually 0.5 ~ 1 day) for such data processing.

Also there’re some limitations for the size of data and the number of sites you choose.

Now let’s do this. https://www.ncdc.noaa.gov/cdo-web/
1 Motivation

2 Access for Pollution Data

3 Access for Meteorological Data

4 Grabbing Data Online

5 Data Cleaning
The following materials are some experiences I gained from Dr. Chen’s PM\(_{2.5}\) project. In that project previous grabbing data work has been done by Tao Zou and Haozhe Zhang.

In developing countries, such as China, no such accesses for free data are available.

When there’s no such finely build web GUI, rather than with a simple order for well organized data, we sometimes need to write program grab the data online (if some webpage presents their data for a certain duration, like 1 hour or so).

R, Python and many other softwares can achieve this goal. However, as we’re all familiar with R, I’ll present the idea to grab data and a problem I met.

Basically, we do this job on a machine with Linux system.
Grabbing Data Online (Con’t)

Basically, we need two parts of program:

1. A R program that can access the content on webpage and put the new data into the old ones.

2. A ’monitor’ program that can allow such R program to run every few minutes (depend on the data requirement) — Crontab.

Cron is a unix, solaris utility that allows tasks to be automatically run in the background at regular intervals by the cron daemon.
Here, some packages in R are suggested:

1. "XML": Many approaches for both reading and creating XML (and HTML) documents via HTTP or FTP.

2. "RCurl": provides functions to allow one to compose general HTTP requests and provides convenient functions to fetch URLs.

3. "foreach", "doParallel": facilitate us to grab data parallelly (in case we’ve got a list of urls of bunches of webpages)

4. "rjson": converts R object into JSON (JavaScript Object Notation) objects and vice-versa
There’re two strong functions in Google Chrome that really facilitate us to analyze the data structure of a certain webpage:

1. View → Developer → View Source
2. Right click → Inspect → Network

I’ll present the example using these two functions.
Generally, the code structure of a webpage is like a huge tree. What we need to do is applying some R functions to trace the tree branch at certain nodes from bottom, until we get what we want, like leaves we desire.

Also, doing operation with string in R is a necessary technique in this scenario.

The case I’ll present is not classical as above, because some tutorial can be easily found on the Internet, basically use functions in R package ”XML” to analyze the source code.
Based on my experience, we can roughly follow the following step to build up and maintain program for grabbing data:

1. Figure out what variables you want.
2. Find possible webpages which present such variable for a certain time, better try to get a list of such webpages if you need many sources. In our project, we need many sites.
3. Use Chrome and R to analyse the data structure.
4. Write R program for a loop and test.
5. Initialize the files where to store the data.
6. Write Crontab and test, modify the time interval to a reasonable one.
7. Check the data regularly (such as every day) to ensure the machine operates well.
The case I’ll present took place due to modification of webpage.
The basic issue is that the webpage designer veiled the data we want in their source code.
I tried to present this example in English version, but failed...
Now let’s go
http://www.nmc.cn/publish/forecast/ABJ/beijing.html
Here are some tips for grabbing data in order to save your machine from system halted:

1. Write commands to initialize the cache (buffer memory) every time you run the R program.

2. Similarly, you’d better write commands to initialize the cache (buffer memory) every time you run the Crontab.

3. Set a threshold value for the tolerance time for your R program access the webpage, i.e., reasonably add missing value to the updated data file if some webpages are hard to access for a while.

4. Your machine should be kept in good condition, with stable electricity maintenance.

The above tips are real, on which we did spend lots of time to figure out.
Motivation

Access for Pollution Data

Access for Meteorological Data

Grabbing Data Online

Data Cleaning
Data Cleaning

Basically, what we grabbed from the webpage are raw data.

Suppose we grab the data every 15 minutes, and we want some sort of hourly data. Therefore, two scenarios appear:

1. The data happens to be the hourly data if we are lucky enough.
2. The data updates for a certain frequency (e.g., every 20 minutes), what we need to do additionally is to assign these data into hourly data.

Such is life, in most of the cases, we’re unlucky.
To be more specific, I list a small piece of raw data we grabbed.

<table>
<thead>
<tr>
<th>GrabbingTime</th>
<th>Temperature</th>
<th>Pressure</th>
<th>Rainfall</th>
<th>UpdateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-04-25 11:15:52</td>
<td>25.30</td>
<td>1015</td>
<td>0.00</td>
<td>2015-04-25 10:55</td>
</tr>
<tr>
<td>2015-04-25 11:30:51</td>
<td>26.00</td>
<td>1014</td>
<td>0.00</td>
<td>2015-04-25 11:10</td>
</tr>
<tr>
<td>2015-04-25 11:45:52</td>
<td>26.40</td>
<td>1014</td>
<td>0.00</td>
<td>2015-04-25 11:25</td>
</tr>
<tr>
<td>2015-04-25 12:00:51</td>
<td>26.90</td>
<td>1014</td>
<td>0.00</td>
<td>2015-04-25 11:40</td>
</tr>
<tr>
<td>2015-04-25 12:15:51</td>
<td>27.30</td>
<td>1014</td>
<td>0.00</td>
<td>2015-04-25 11:50</td>
</tr>
<tr>
<td>2015-04-25 12:30:49</td>
<td>28.10</td>
<td>1014</td>
<td>0.00</td>
<td>2015-04-25 12:10</td>
</tr>
</tbody>
</table>

Considering the data we want to use, we need to 'summary' the observations from 2nd row to 5th row above to the time 2015-04-25 12:00.
To make such hourly summary more sense, we need to figure out what variables mean at each updated time.

In this example, we have the corresponding meaning for each variable:

1. Temperature: the temperature at the updated time (Celsius)
2. Pressure: the pressure at the updated time (hPa)
3. Rain fall: the cumulative rain fall from the previous hour (i.e., from 2015-04-25 11:00) (mm)
Though bias might happen, the following method seems to be reasonable:

1. Temperature: take the average of previous hour observation.
2. Pressure: take the average of previous hour observation.
3. Rainfall: find the closest point to the target time in the previous hour.

Essentially, what we need to do is handling subscripts of the data within the previous hour (which can be located by year, month, day, and hour).
As a newbie, I did such data cleaning job with the following idea:

1. Generate a new data frame with only integral hour, preserving blanks (in R is NA) for other variables
2. Use 'for' loop to get the relative close time's subscripts in raw data with respect to the current time point we’re manipulating (at a specific loop)
3. Then fill in the blanks one-by-one.

Time for running R program with the idea above would be around 8 hours for one site, which is unrealistic for the progress of project. For example, in Hebei, we got around 53 weather sites, after applying parallel computing with 3 cores, we still need 6 days to clean the data.
Suggested by Xiaojun Mao and Fan Cao, I used R package 'dplyr'.

It was developed by Hadley Wickham, a former PhD student in our department, who also developed package 'lubridate', 'ggplot2', etc.

We can use the R function in that package, which will invokes C program, making computation much faster.
After fetching a tutorial for 'dplyr', I found there are many useful functions:

1. "group_by": group a tbl (a data structure, data.frame can also be applied) by one or more variables.
2. "mutate": adds new variables and preserves existing.
3. "filter": return rows with matching conditions. (will not be introduced)
4. ...

Wang, He & Zhu (ISU)
A combination of "group_by(.data, ...) %>% mutate(Indicator = f)" can be very powerful, where "f" is a function created with the variable within the group which generated by "group_by" function.

After get the object generated from above, we can use "apply" function to fetch such indicator for each group.

Program after modification only need 25 seconds for a weather site.

Now see the example R code.
Summary

- We introduced access for AirData (from EPA) and meteorological data (from Climate Data Online) in US
- Skeleton for grabbing data online with regular frequency
- A useful package for data cleaning: "dplyr"
References

- EPA: https://www.epa.gov/outdoor-air-quality-data
- Climate Data Online: https://www.ncdc.noaa.gov/cdo-web/
- R package "dplyr" Tutorial: https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html