Clean Air in the Time of COVID

Use Cases for Google Earth Engine & Sentiment Analysis

Ben Best, PhD

2020-05-25

Contents

Overview

These materials are for a 45 minute teaching demonstration on 2020-05-26 oriented towards students in the new Masters in Environmental Data Science at UCSB.

Motivation

This is an exciting time for the emerging field of Environmental Data Science. Environmental problems are increasingly complex and require advanced technical skills to translate the ever expanding flow of data into actionable insights.

Technologies

The two specific technologies featured in this teaching demonstration highlight some of the most promising aspects of truly :

- **Google Earth Engine** leverages the massive storage and computational capabilities of the Google Cloud to analyze petabytes of the publicly available satellite data. For instance, global climatologies averaging across 40 years of temperature can be calculated and mapped in seconds. This is a truly *big data* platform!
- **TensorFlow** is the machine learning software made open-source by Google. It is the most commonly used software for audio, text and image analysis. More specifically tensorflow allows construction of *convolutional neural networks*, which represent a layering of nodes to enable complex pattern matching. These *deep learning* models have been popularized by their ability to beat the best human at the most complex game of Go, self-drive vehicles and respond to your beck and call through Alexa's voice commands.

Use Case

I was struck by the learning

Prerequisites

R Skill Level: Intermediate - you've got basics of R down.

You will use the `sf` package for vector data along with the `dplyr` package for calculating and manipulating attribute data.

Github

Here is the Github repository of the source files for this website:

- github.com/ecoquants/r3-workshop

Google Drive

- **workshop/**
  - **agenda**
  - **notes**
  - **presentations/**
  - **data/**

Resources

Online books: - R for Data Science - Spatial Data Analysis and Modeling with R

Chapter 1

Setup

1.1 Install software

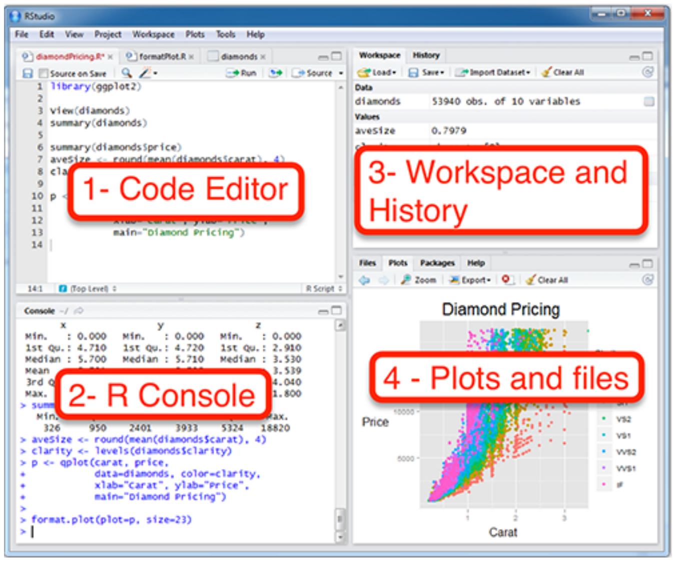
This workshop will require the following software installed on your machine:

- R
- RStudio

Please download the appropriate stable release for your operating system.

1.2 Launch RStudio

RStudio is an integrated development environment (IDE) for editing text in the code editor, running commands in the R console, viewing defined objects in the workspace and past commands in the history, and viewing plots, files and help. Here's a layout of the panes in RStudio, each of which can have several tabs for alternate functionality:


RStudio
File Edit View Project Workspace Plots Tools Help
diamondPricing.R* formatPlot.R diamonds
Source on Save Run Source
1- Code Editor
2- R Console
3- Workspace and History
4 - Plots and files
Workspace History
Load Save Import Dataset Clear All
Data
Values
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
VS2
VS1
VVS2
VVS1
IF
14:1 (Top Level) R Script
Console ~/

```
library(ggplot2)

view(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
cla
 
p <


             main="Diamond Pricing")
```

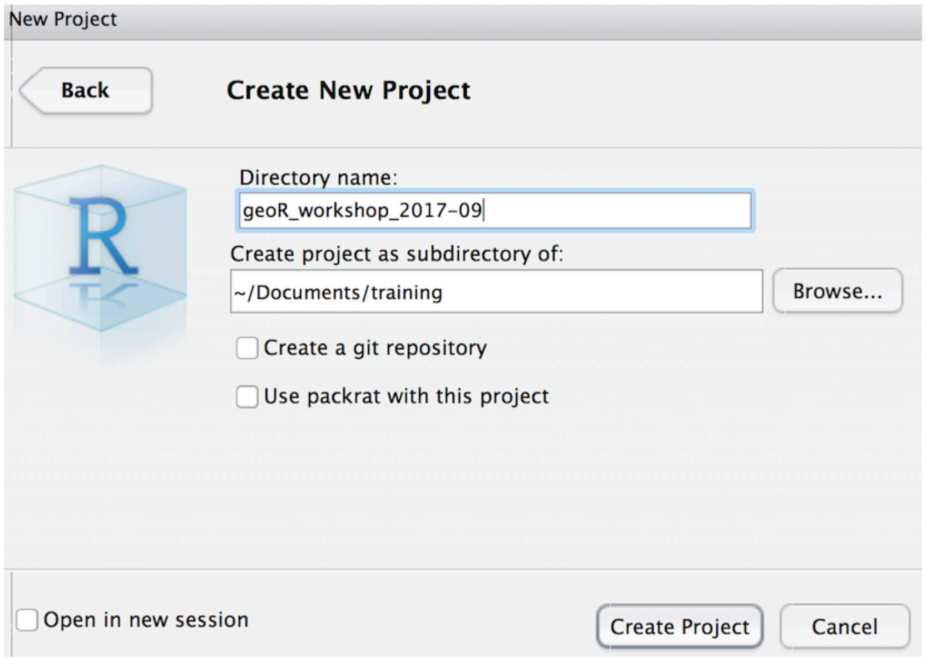
```
      x                y                z
Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
Median : 5.700   Median : 5.710   Median : 3.530
Mean                                       : 3.539
3rd Q                                      4.040
Max.                                       1.800
> summ
   Min.                                      Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(plot=p, size=23)
> 
```

```
diamonds   53940 obs. of 10 variables
aveSize    0.7979
```

Check out the Help > Cheatsheets > RStudio IDE Cheat Sheet.

1.3 Create RStudio project

In RStudio, please create an RStudio project (File > New Project... > New Directory > Empty Project) to organize your code and data for this workshop into a single folder. Feel free to organize this folder wherever makes sense on your laptop. Here's what I'm using:



Directory name: geoR_workshop_2017-09

Create project as subdirectory of: ~/Documents/training

☐ Create a git repository

☐ Use packrat with this project

☐ Open in new session

1.3.1 Create relative path to data folder

In order to work through a common set of code and access data files in a consistent manner, getting properly situated with paths will save many headaches. Paths can be referenced as an "absolute" path starting with your drive letter (eg "C:" on Windows or "/Users" on Mac). Absolute paths are specific to platform (Windows / Linux / Mac) and not portable for use on other machines or other users wishing to organize content anywhere else. A "relative" path, meaning relative to your current working directory, is portable. Try these commands in the console of RStudio:

```r
# get working directory
getwd()

# list files in working directory
list.files('.')

# list files one directory up
list.files('..')

# create a directory
dir.create('data')

# set working directory
```

```r
setwd('data')
setwd('..')
```

So you discovered the absolute path to your current working directory, and list files in the current directory (`.`) or one directory up (`..`). Then you created the "data" directory and set the working directory to be inside of it, then back out of it.

The main reason we created the RStudio project file (filename ends in `.Rproj`) is to have the same working directory every time you return to this project by double-clicking the `*.Rproj` file in your file explorer. So paths defined in the code of files there can be based on that same working driectory and consistently work, even if you zip them up and send to a colleague who places them on an arbitrary location on their computer.

1.4 Download data

Please download the following zip files into your newly created "data" folder from above.

- Site layout shapefiles
- Airborne remote sensing data
- Landsat NDVI raster data
- Meteorological data

Then unzip. You should see the following when listing files (and directories) in the data directory.

```r
list.files('data')
```

```
##  [1] "anand-vihar, delhi, delhi, india-air-quality.csv"
##  [2] "aq_delhi_no2.csv"
##  [3] "city_delhi"
##  [4] "city_Delhi.India.geojson"
##  [5] "city_Delhi.India.zip"
##  [6] "city_Delhi+India.geojson"
##  [7] "city_Delhi+India.json"
##  [8] "ee-chart_no2-delhi.csv"
##  [9] "grace-hopper.jpg"
## [10] "loc_DelhiIndia.csv"
## [11] "new-delhi us embassy, india-air-quality.csv"
## [12] "openaq.csv"
## [13] "tweets_airquality_DelhiIndia.rds"
## [14] "waqi-covid19-airqualitydata-2015H1.csv"
## [15] "waqi-covid19-airqualitydata-2016H1.csv"
## [16] "waqi-covid19-airqualitydata-2017H1.csv"
## [17] "waqi-covid19-airqualitydata-2018H1.csv"
## [18] "waqi-covid19-airqualitydata-2019Q1.csv"
## [19] "waqi-covid19-airqualitydata-2019Q2.csv"
## [20] "waqi-covid19-airqualitydata-2019Q3.csv"
## [21] "waqi-covid19-airqualitydata-2019Q4.csv"
## [22] "waqi-covid19-airqualitydata-2020.csv"
```

1.5 Install R Packages

Here's a bit of code to install packages that we'll use throughout the workshop. Please copy and paste this code into your console.

```
# concatenate a vector of package names to install
packages = c(
  # general data science
  'tidyverse',
  # dynamic document creation
  'knitr','rmarkdown',
  # handle spatial data
  'rgdal','raster','sp','sf','geojsonio',
  # spatial data
  'maps',
  # spatial analysis
  'rgeos','geosphere',
  # static plotting & mapping
  'RColorBrewer','ggplot2','rasterVis',
  # interactive plotting & mapping
  'plotly','leaflet','mapview','mapedit')

# loop through packages
for (p in packages){

  # if package not installed
  if (!require(p, character.only=T)){

    # install package
    install.packages(p)
  }

  # load package
  library(p, character.only=T)
}

# report on versions of software & packages
```

```r
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] units_0.6-6          here_0.1             glue_1.4.1
##  [4] jsonlite_1.6.1       mapedit_0.5.0        mapview_2.7.8
##  [7] leaflet_2.0.3        plotly_4.9.1         rasterVis_0.46
## [10] latticeExtra_0.6-28 lattice_0.20-38      RColorBrewer_1.1-2
## [13] geosphere_1.5-10     rgeos_0.5-3          maps_3.3.0
## [16] geojsonio_0.9.2      sf_0.8-0             raster_3.1-5
## [19] rgdal_1.4-8          sp_1.4-2             rmarkdown_2.1
## [22] knitr_1.28           forcats_0.4.0        stringr_1.4.0
## [25] dplyr_0.8.5          purrr_0.3.4          readr_1.3.1
## [28] tidyr_1.1.0          tibble_3.0.1         ggplot2_3.3.0
## [31] tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] leafem_0.1.1       colorspace_1.4-1   ellipsis_0.3.1     class_7.3-15
##  [5] rprojroot_1.3-2    satellite_1.0.2    base64enc_0.1-3    fs_1.3.1
##  [9] httpcode_0.3.0     rstudioapi_0.11    hexbin_1.28.0      fansi_0.4.1
## [13] lubridate_1.7.8    xml2_1.2.2         codetools_0.2-16   broom_0.5.5
## [17] dbplyr_1.4.2       png_0.1-7          shiny_1.4.0.2      compiler_3.5.2
## [21] httr_1.4.1         backports_1.1.7    assertthat_0.2.1   fastmap_1.0.1
## [25] lazyeval_0.2.2     cli_2.0.2          later_1.0.0        htmltools_0.4.0
## [29] tools_3.5.2        gtable_0.3.0       geojson_0.3.2      V8_3.0.2
## [33] Rcpp_1.0.4.6       cellranger_1.1.0   vctrs_0.3.0        crul_0.9.0
## [37] nlme_3.1-142       blogdown_0.17      crosstalk_1.1.0.1  xfun_0.14
## [41] rvest_0.3.5        mime_0.9           lifecycle_0.2.0    jqr_1.1.0
## [45] zoo_1.8-7          scales_1.1.1       hms_0.5.3          promises_1.1.0
## [49] parallel_3.5.2     yaml_2.2.1         curl_4.3           stringi_1.4.6
## [53] maptools_1.0-1     e1071_1.7-3        rlang_0.4.6        pkgconfig_2.0.3
## [57] evaluate_0.14      htmlwidgets_1.5.1  tidyselect_1.1.0   vembedr_0.1.3
## [61] magrittr_1.5       bookdown_0.16      R6_2.4.1           generics_0.0.2
## [65] DBI_1.1.0          pillar_1.4.4       haven_2.2.0        foreign_0.8-72
## [69] withr_2.2.0        modelr_0.1.5       crayon_1.3.4       KernSmooth_2.23-16
## [73] grid_3.5.2         readxl_1.3.1       data.table_1.12.6  reprex_0.3.0
## [77] digest_0.6.25      classInt_0.4-3     webshot_0.5.2      xtable_1.8-4
## [81] httpuv_1.5.2       stats4_3.5.2       munsell_0.5.0      viridisLite_0.3.0
```

1.6 Create Rmarkdown file

Rmarkdown is a dynamic document format that allows you to knit chunks of R code with formatted text (aka markdown). We recommend that you use this format for keeping a reproducible research document as you work through the lessons To get started, File > New File > Rmarkdown… and go with default HTML document and give any title you like (default "Untitled" or "test" or "First Rmd" is fine).

Check out the Help > Markdown Quick Reference and Help > Cheatsheets > R Markdown Cheat Sheet.

Here's a 1 minute video on the awesomeness of Rmarkdown:

Chapter 2

Satellite

Objectives

Question

- How have emissions related to air quality changed since COVID-19 lockdowns were put in place?

Study Area: Delhi, India

- We'll use **Delhi, India** as our initial city study area. Prime Minister Modhi issued a nationwide lockdown on 24 March, 2020.

News:

- Air pollution falls by unprecedented levels in major global cities during coronavirus lockdowns - CNN
- India: air pollution in the north has hit a 20-year low, NASA report says - CNN
- Fact Check: Is The COVID-19 Lockdown Decreasing Delhi Air Pollution? - Smart Air Filters

Learning Outcomes

1. Browse Google Earth Engine's data catalogue
2. Map satellite imagery dataset layer
3. Filter by date
4. Average values over time
5. Upload a polygon into assets

6. Use polygon to extract values for study area
7. Average values in space within the polygon
8. Generate a time series chart for satellite data
9. Download data as a text file (csv)

Prerequisites

A **Google Earth Engine account** that is associated with a Google account, such as from Gmail, is required to log into https://code.earthengine.google.com.

If you need a GEE account, please visit https://signup.earthengine.google.com.

You may need to log out and back into your web browser as the preferred Google account to request permissions. This approval process can take days to weeks unfortunately.

2.1 Get City Boundary

The first step is to define our study area. I made a little helper function in R `city2zip()` to:

1. Fetch the administrative boundary from the Nominatim OpenStreetMap API given a city name.
2. Extract the polygon information, convert to shapefile and zip for upload into GEE.

```r
source("functions.R")

city2zip("Delhi, India")
```

```
## $geojson
## [1] "/Users/bbest/github/meds-demo/data/city_Delhi.India.geojson"
##
## $zip
## [1] "/Users/bbest/github/meds-demo/data/city_Delhi.India.zip"
```

Chapter 3

Sentiment

3.1 Twitter Examples

"air quality" - Twitter Search / Twitter

For the first time in decades, Mount Everest was visible from Kathmandu due to improved air quality.

— Tom Patterson (@MtnMapper) May 21, 2020

{{% tweet "1263487780799184897" %}}

3.2 Authenticate Twitter

Obtaining and using access tokens • rtweet:

- Creating a Twitter App
- Authorization methods
  - 2. Access token/secret method
- Authorization in future R sessions

  ```r
  library(rtweet)
  get_token()
  ```

3.3 Enable Google Geocoding

TODO: switch to OpenStreetMap API, for radius too

To obtain an API key and enable services, go to https://cloud.google.com/maps-platform/. [as ben@ecoquants.com]

- Create project "meds-gee"
- Maps APIs and Servi… – Google Maps – meds-gee – Google Cloud Platform
- APIs sidebar. Click Geocoding API. Enable.
- Credentials sidebar. + CREATE CREDENTIALS -> API key
- Saved to `~/private/google_meds-gee_api-key.txt`

```r
# load libraries ----
# use librarian to load libraries, installing if needed
if (!require("librarian")) install.packages("librarian")
library("librarian")

pkgs <- c(
  # utility
  "here","glue","stringr","dplyr","readr",
  # airquality
  "ropensci/ropenaq",
  # spatial
  "ggmap","mapview","leaflet",
  # text
  "rtweet","textdata","tidytext",
  # tensorflow
  "tensorflow","keras","tfhub","rstudio/tfds")
shelf(pkgs)

# variables ----
q           <- '"air quality"'
city        <- "Delhi, India"
city_r      <- "15mi"

q_s         <- str_remove_all(q, '[ ,"]')
city_s      <- str_replace(city, ", ", "+")
city_geo    <- glue("data/city_{city_s}.geojson")
#cities_csv <- glue("data/city_{loc_s}.csv")
tweets_rds <- glue("data/tweets_{q_s}_{city_s}.rds")
#tbl_rds <- glue("data/tweets_{qloc}.csv")
#d_csv  <- "data/tweets_air-quality.csv"
gkey        <- "~/private/google_meds-gee_api-key.txt"

# get location lon/lat from place name
if (!file.exists(loc_csv)){
  register_google(key = readLines(gkey))
  loc_xy <- geocode(loc)
  stopifnot(nrow(loc_xy) == 1)

  write_as_csv(loc_xy, loc_csv)
}
loc_xy  <- read_csv(loc_csv)
loc_xyr <- glue("{loc_xy$lat},{loc_xy$lon},{loc_r}")

if (!file.exists(tbl_rds)){
  tbl <- search_tweets(
    q, n = 1000,
    geocode = loc_xyr)

  saveRDS(tbl, tbl_rds)
}
tbl <- readRDS(tbl_rds)
```

```
s_b <- get_sentiments('bing')
s_a <- get_sentiments('afinn')
s_n <- get_sentiments('nrc')

tbl <- tbl %>%
  mutate(
    text_clean = text %>%
      str_replace_all("[^[:ascii:]]", "_") %>%
      tolower() %>%
      str_replace_all("@[^ ]+", "_usr_") %>%
      str_replace_all("http[^ ]+", "_url_"))
# tbl %>% select(created_at, screen_name, text, text_clean) %>% View()

df <- tbl %>%
  select(status_id, created_at, screen_name, text_clean) %>%
  unnest_tokens(output = word, input = text_clean, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  left_join(s_b, by = "word")
```

3.4 city osm poly

```
source("functions.R")

if (!file.exists(city_geo))
  city2geo(city_s, city_geo)
city_ply <- read_sf(city_geo)
```

```r
write_sf(city_ply, "data/city_delhi/delhi.shp")


# https://developers.google.com/earth-engine/charts_image_series
mapview(city_ply)
```

3.5 openaq

- API: https://docs.openaq.org/#api-Measurements-GetMeasurements
- ropenaq: Accesses Air Quality Data from the Open Data Platform OpenAQ

```r
library(ropenaq)
library(tidyr)

cities_table <- aq_cities() %>%
  filter(str_detect(city,"Delhi"))

aq_cities() %>%
  filter(str_detect(city,"Delhi"))

city_locs <- aq_locations(country = "IN", city = "Delhi") %>%
  filter(no2, pm25) %>%
  arrange(desc(count))
View(city_locs)


loc_d <- city_locs %>%
  select(-count) %>%
  unnest(countsByMeasurement) %>%
  filter(
    parameter == "no2",
    date(lastUpdated) > now() %>% date() - days(1)) %>%
  arrange(desc(count)) %>%
  slice(1)

loc <- loc_d$location
loc <- loc_d$location %>% str_replace_all(" ", "+")

city_loc_d <- city_locs %>%
  filter(location == loc)

city_loc_d %>% View()
```

```
city_locs %>%
  filter(location == loc) %>%
  View()

m_no2 <- aq_measurements(country = "IN", city = "Delhi", parameter = "no2")
range(as_date(m_no2$dateUTC)) # "2019-12-12" "2020-05-23"
attr(m_no2, "meta") # limit: 10,000;  found: 1,162,354

m_no2 <- aq_measurements(
  latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "10",
  parameter = "no2", page = 1) %>%
  bind_rows(
    aq_measurements(
      latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "100",
      parameter = "no2", page = 2)) %>%
  bind_rows(
    aq_measurements(
      latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "100",
      parameter = "no2", page = 3)) %>%
  bind_rows(
    aq_measurements(
      latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "100",
      parameter = "no2", page = 4))
range(as_date(m_no2$dateUTC)) # "2019-12-12" "2020-05-23"
attr(m_no2, "meta") # limit: 10,000;  found: 35,069

m1 <- aq_measurements(
      latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "100",
      parameter = "no2", page = 1)
range(as_date(m1$dateUTC)) # "2020-02-24" "2020-05-23"
attr(m1, "meta") # limit: 10,000;  found: 35,069

m2 <- aq_measurements(
      latitude  = loc_d$latitude, longitude = loc_d$longitude, radius    = "100",
      parameter = "no2", page = 2)
range(as_date(m2$dateUTC)) #
attr(m2, "meta") # limit: 10,000;  found:

location <- https://api.openaq.org/v1/locations

q_loc <- loc_d$location %>% str_replace_all(" ", "+")
q_loc
fromJSON(glue("https://api.openaq.org/v1/locations?location={q_loc}"))
openaq
https://api.openaq.org/v1/measurements?location=Shadipur%2C%2BDelhi%2B-%2BCPCB&parameter=no2
```

```
https://api.openaq.org/v1/locations/IN-88
https://api.openaq.org/v1/locations?location=Shadipur,+Delhi+-+CPCB
https://api.openaq.org/v1/measurements?location=Shadipur,+Delhi+-+CPCB&parameter=no2

library(httr)

m_url <- "https://api.openaq.org/v1/measurements"

json_url <- glue("https://api.openaq.org/v1/measurements?location={q_loc}&parameter=no
x <- GET(
    m_url,
    query = list(
      location  = loc_d$location,
      parameter = "no2",
      format    = "json",
      limit     = 1))
meta <- content(x, "text") %>% fromJSON() %>% .$meta

n_max   <- 10000
n_pages <- ceiling(meta$found/n_max)
for (i in 1:pages){ # i = 2
  date_beg <- loc_d$firstUpdated %>% as_date() %>% as.character()
  date_end <- loc_d$firstUpdated + days(60)
  date_end <- as_date(date_end) %>% as.character()

  date_beg <- date("2018-01-01")
  date_end <- date_beg + months(1)

  x <- GET(
    m_url,
    query = list(
      #location  = loc_d$location,
      coordinates = with(city_ply, glue("{round(lat,2)},{round(lon,2)}")),
      #radius      = 2500, # default
      radius      = 1000000, # default
      parameter   = "no2",
      format      = "csv",
      #format    = "json",
      date_from   = date_beg,
      date_to     = date_end,
      #limit       = n_max,
      limit       = 5,
      page        = i))
  x$url
  #if (http_type(x) != "application/json")
  #  stop("API did not return json", call. = F)
  #cat(content(x, "text")) # {"statusCode":404,"error":"Not Found"}
  # no2 unit: µg/m³

  #y <- fromJSON(content(x, "text"))

  y <- read_csv(content(x, "text"))
  y

  y %>%
    select(dtime_utc = utc, no2_mgm3 = value) %>%
    write_csv("data/aq_delhi_no2.csv", append = ifelse(i != 1, T, F))
  # 6636
}


d <- read_csv(url)

 %>%   .$results
tbl <- res$results %>% as_tibble()
names(tbl)
tbl %>%
  select(parameter, date_utc = tbl$date$)
names(res$results)
m_no2_a <- aq_measurements(
  country = "IN", city = "Delhi", parameter = "no2",
  attribution = T, averaging_period = T, source_name = T)

aq_locations(country = "IN", city = "Delhi")
m_no2_loc <- aq_measurements(
  #country = "IN", city = "Delhi", location = "Shadipur, New Delhi - CPCB",
  #country = "IN", city = "Delhi", location = loc,
  #location = "DTU, Delhi - CPCB",
  location = "DTU,+Delhi+-+CPCB",
  #country = "IN", city = "Delhi", location = "IN-88",
  parameter = "no2")
range(as_date(m_no2_loc$dateUTC)) #
attr(m_no2_loc, "meta") # limit: 10,000;  found:

aq_measurements

tmp <- aq_measurements(country="IN", city="Delhi", location="Anand+Vihar", parameter="pm25")

coordinates, radius
date_from
```

```
date_to

m_pm25 <- aq_measurements(country = "IN", city = "Delhi", parameter = "pm25")
attr(m, "meta")
# limit: 10,000;  found: 1,141,521

range(as_date(m_pm25$dateUTC)) # "2019-12-12" "2020-05-23"
#, limit = 10, page = 1)
kable(results_table)
```

3.6 aqicn.org

Downloaded from: - Air Quality Historical Data Platform

aqicn.org/city/delhi - Delhi Air Pollution: Real-time Air Quality Index; daily avg of this station: - Anand Vihar, Delhi sources: - Delhi Pollution Control Commitee (Government of NCT of Delhi) - CPCB - India Central Pollution Control Board

TODO:

- Beijing Air Pollution: Real-time Air Quality Index
- COVID-19 Worldwide Air Quality data
  - 380 major cities in the world, from January 2020 until now.

The CSV data sets can be downloaded programatically: The url is

https://aqicn.org/data-platform/covid19/report/14286-3d64290c/period

Where period is any of 2019Q1, 2019Q2, 2019Q3, 2019Q4, 2018H1, 2017H1, 2016H1, 2015H1.

For instance using curl: curl –compressed -o waqi-covid-2020.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2020 curl –compressed -o waqi-covid-2019Q1.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2019Q1 curl –compressed -o waqi-covid-2019Q2.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2019Q2 curl –compressed -o waqi-covid-2019Q3.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2019Q3 curl –compressed -o waqi-covid-2019Q4.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2019Q4 curl –compressed -o waqi-covid-2018H1.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2018H1 curl –compressed -o waqi-covid-2017H1.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2017H1 curl –compressed -o waqi-covid-2016H1.csv https://aqicn.org/data-platform/covid19/report/14286-3d64290c/2016H1 curl –compressed -o waqi-covid-2015H1.csv https://aqicn.org/data-platform/covid19/report/14286-

3d64290c/2015H1 To download the list of stations with latitude, longitude and EPA feed: curl –compressed -o airquality-covid19-cities.json https://aqicn.org/data-platform/covid19/airquality-covid19-cities.json

```
library(dygraphs)
library(lubridate)
library(xts)

#d <- read_csv("data/new-delhi us embassy, india-air-quality.csv") %>%
d <- read_csv("data/anand-vihar, delhi, delhi, india-air-quality.csv") %>%
  mutate(
    date = as_date(date))
d
# date         pm25  pm10    o3   no2   so2    co
#View(d)
range(d$date)
#range(d$jday)
d <- d %>%
  mutate(
    jday = yday(date))

# calculate climatology
d_c <- d %>%
  group_by(jday) %>%
  summarise(no2_clim = mean(no2, na.rm = T)) %>%
  ungroup()

# show in time
d_y <- expand_grid(
  year = min(year(d$date)):year(now()),
  jday = 1:365) %>%
  mutate(
    date = date(glue("{year}-01-01")) + days(jday - 1)) %>%
  left_join(d_c, by = "jday") %>%
  filter(date <= date(now()))

d_y2 <- d_y %>%
  select(date, no2_clim) %>%
  left_join(
    d %>%
      select(date, no2),
    by = "date")

#x <- xts(x = d$pm25, order.by = d$date)
x <- xts(x = select(d_y2, -date), order.by = pull(d_y2, date))
dygraph(x) %>%
  dyRangeSelector()
```

Next steps: weekly / monthly average

- RPubs - Beijing PM 2.5 Line Charts 2017/01 to 2018/03 ## search full archive

Setup dev env:

- Dev environments — Twitter Developers

```
beg <- "2020-05-20"
end <- "2020-05-23"

tbl <- search_fullarchive(
  q,
  n = 100,
  fromDate = beg,
  toDate = end,
  env_name = "research")

  # safedir = NULL,
  # parse = TRUE,
  # token = NULL
  # point_radius:[lon lat radius])

table(as.Date(tbl$created_at))
ts_plot(tbl)


    fromDate, toDate,
/search/fullarchive/:label.json
```

TODO:

1. get sentiment per tweet by tallying negative & positive
2. join back to tweets
3. get over time

3.7 TensorFlow

- datasets to train on:
  - Sentiment140
  - `dictionary.txt` in Sentiment analysis for text with Deep Learning - Towards Data Science from https://nlp.stanford.edu/sentiment/

  - Sentiment analysis on Twitter using Word2vec and Keras | Ahmed BESBES: ”The dataset can be downloaded from this link. It’s a csv file that contains **1.6 million rows**. Each row has amongst other things the text of the tweet and the corresponding sentiment.

- covid-twitter-bert | TensorFlow Hub
- Sentiment140 dataset with 1.6 million tweets | Kaggle
- http://help.sentiment140.com/ Sentiment140 allows you to discover the sentiment of a brand, product, or topic on Twitter.
  - http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip

3.8 tfhub

RStudio AI Blog: tfhub: R interface to TensorFlow Hub

you need to install the Python packages once:

```r
tensorflow::install_tensorflow()
keras::install_keras()
tfhub::install_tfhub()
tfds::install_tfds()
reticulate::py_config()
```

```r
library(tfhub)
library(keras)

layer_mobilenet <- layer_hub(
  handle = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4")

input <- layer_input(shape = c(224, 224, 3))
output <- layer_mobilenet(input)
model <- keras_model(input, output)
summary(model)
```

`{rv}, eval = Fimg <- image_load("data/grace-hopper.jpg", target_size = c(224,224)) %>%    image_to_array() img <- img/255 dim(img) <- c(1, dim(img)) pred <- predict(model, img) imagenet_decode_predictions(pred[,-1,drop=FALSE])[[1]]`
“