

# Interactive Maps for Dummies:

## Lecture 3: Raster Data and Geocomputation

SHI Shuang

The University of Hong Kong  
Visiting Research Fellow at Brown University

Jul. 29, 2020

Prepared for HKBU Zoom Lectures

# Today

- ▶ Raster Data

- ① Read Raster Data
- ② Plot and Summarize Raster Data
- ③ Reclassify Value in a Raster
- ④ Rasters in ggplot2
- ⑤ Masks
- ⑥ Summarize Rasters with Polygons
- ⑦ Summarizing Rasters with Points
  - \* An Application

- ⑧ Read in NetCDFs Raster Data

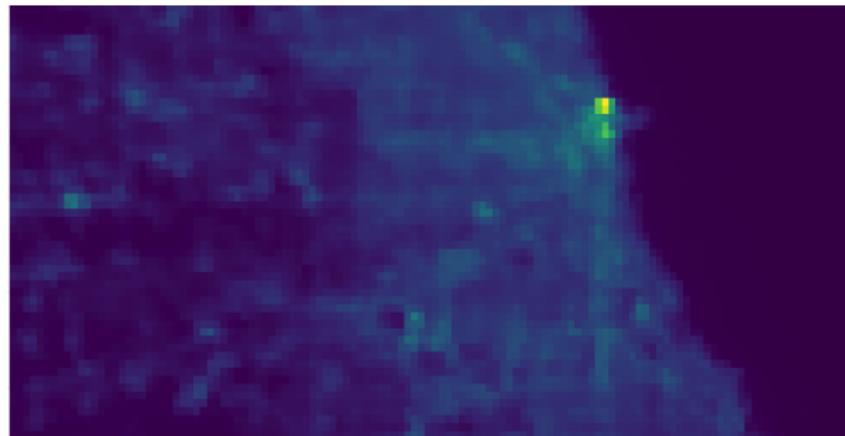
- ▶ Geocoding

- ▶ Voronoi Diagram

- ▶ Geocomputation

# Raster Data

- ▶ Raster data are essentially grids with values assigned to the squares within the grid
  - A major difference between rasters and vectors: vectors are composed of discrete elements throughout space, whereas rasters are defined continuously through space
- ▶ A great source of raster data is from remote sensing images (e.g. pollution data, land use, nightlight etc.)



# Nightlight Raster Data Developed by EOG

- ▶ A great example is the nightlight raster data developed by Earth Observation Group (EOG) under NOAA
  - Economists use this data to develop high-resolution measure on economic activities (Henderson et al., 2012; Pinkovskiy and Sala-i-Martin, 2016)

"Measuring Economic Growth from Outer Space", <https://www.aeaweb.org/articles?id=10.1257/aer.102.2.994>

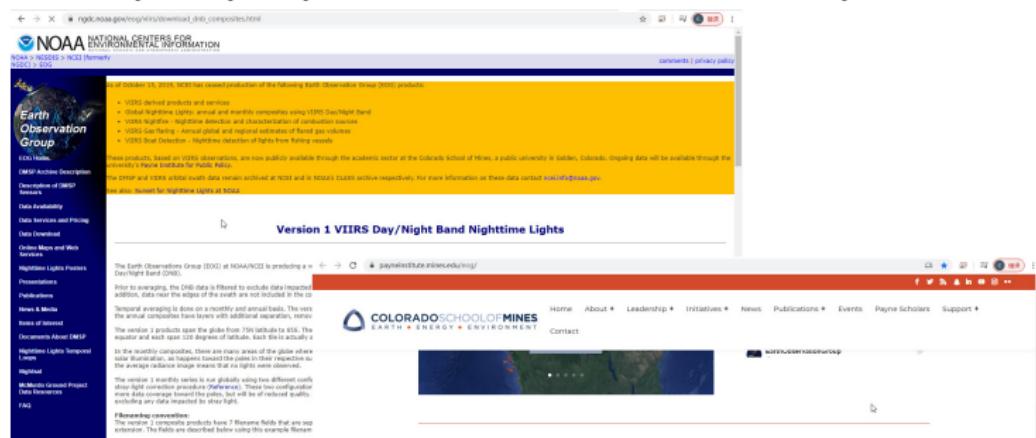
"Lights, Camera...Income! Illuminating the National Accounts-Household Surveys Debate",

<https://academic.oup.com/cje/article-abstract/131/2/579/2607043>



# Where to download?

- ▶ Could be downloaded online from  
<https://www.ngdc.noaa.gov/eog/index.html> (data on and before April 2019) and <https://payneinstitute.mines.edu/eog/> (for later period) for free!
  - Yearly data from DMSP between 1992 and 2013
  - Monthly and yearly data from VIIRS from 2012 to today



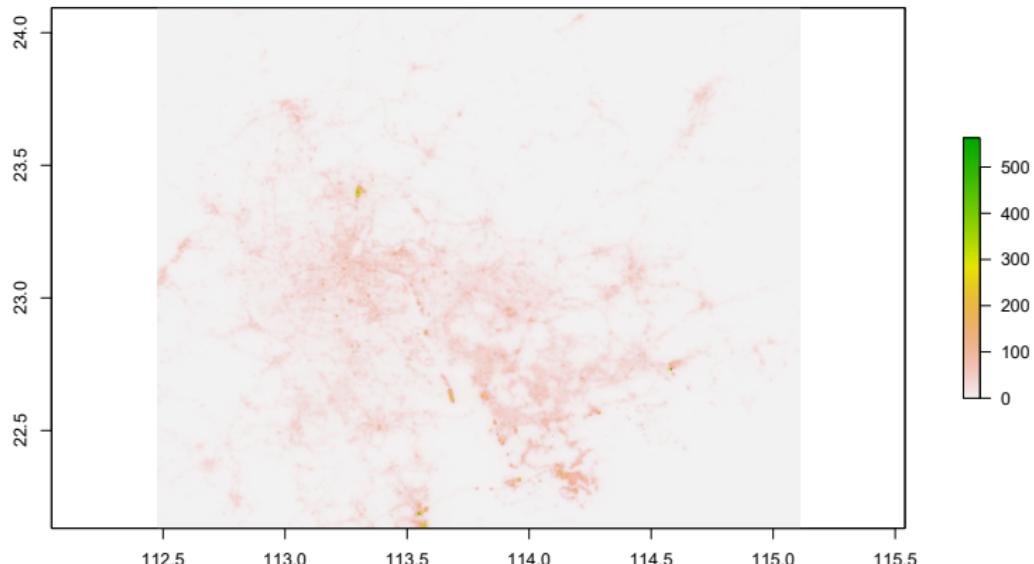
The screenshot shows a web browser displaying the NOAA/NCEI website. The main content is titled "Version 1 VIIRS Day/Night Band Nighttime Lights". It includes a map of the globe showing nighttime light distribution. Below the map, there's a section titled "QUICK DATA ACCESS" with three buttons: "Boat Detection", "Nighttime Light", and "Nightfire". The URL in the address bar is <https://payneinstitute.mines.edu/eog/>.

# Prepare R

```
2 # General R setup ----  
3 rm(list = ls())  
4 getwd()  
5 setwd("D:/Dropbox/course/ECON4016/Lecture6_2020spring")  
6  
7 # Load new packages  
8 install.packages("pacman")  
9 library(pacman)  
10 p_load(geos, GISTools)  
11 # Load old packages  
12 p_load(lubridate, rgdal, raster, broom, rgeos, GISTools)  
13 p_load(dplyr, sp, ggplot2, ggthemes, fields, magrittr, viridis, ggmap)  
14 # My ggplot2 theme  
15 theme_ed <- theme(  
16   legend.position = "bottom",  
17   panel.background = element_rect(fill = NA),  
18   # panel.border = element_rect(fill = NA, color = "grey75"),  
19   axis.ticks = element_line(color = "grey95", size = 0.3),  
20   panel.grid.major = element_line(color = "grey95", size = 0.3),  
21   panel.grid.minor = element_line(color = "grey95", size = 0.3),  
22   legend.key = element_blank())  
23 # The directory stored the nightlight data  
24 dir_nl <- "E:/VIIRS_Nightlight/"  
25
```

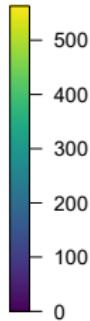
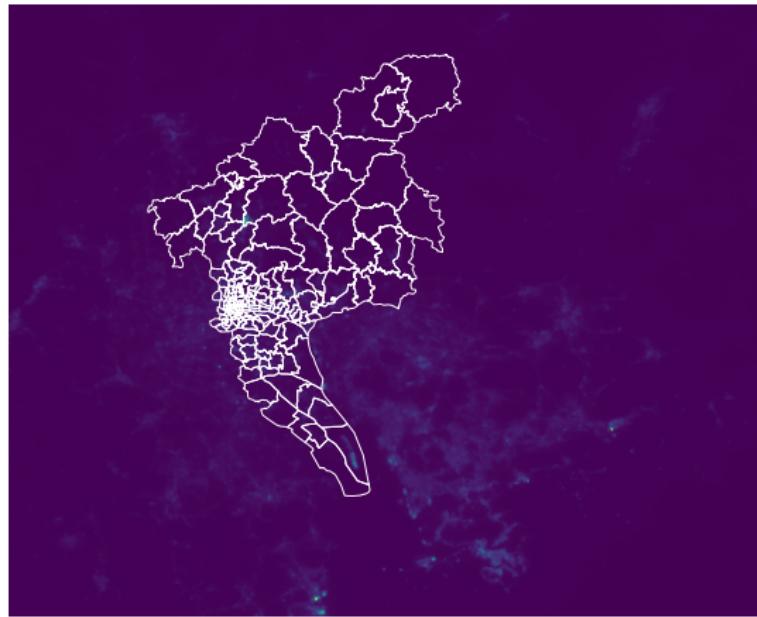
# Read Raster Data

```
27 # 1.1 read raster data  
28 list.files(path=dir_nl, pattern =".tif")  
29 nl_gz_201912 <- raster(paste0(dir_nl, "nl_gz_1912_v3.tif"))  
30 gz_t <- readOGR("gz.shp")  
31 sz_t <- readOGR("sz_t.shp")  
32  
33 # 1.2 plot and summarize a raster  
34 plot(nl_gz_201912)
```



# Plot the Guangzhou Nightlights Raster and the Block Boundary

```
36 library(viridisLite)
37 library(viridis)
38 plot(nl_gz_201912, col = viridis(1e3), axes = F, box = F)
39 lines(gz_t, col = "white", lwd = 0.8)
```



## Summerize the Raster Data

```
> # R's description of the raster
> nl_gz_201912
class      : RasterLayer
dimensions : 471, 631, 297201 (nrow, ncol, ncell)
resolution : 0.004166667, 0.004166667 (x, y)
extent     : 112.4771, 115.1063, 22.13125, 24.09375 (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
source     : E:/VIIRS_Nightlight/nl_gz_1912_v3.tif
names      : nl_gz_1912_v3
values     : -0.14, 562.98 (min, max)

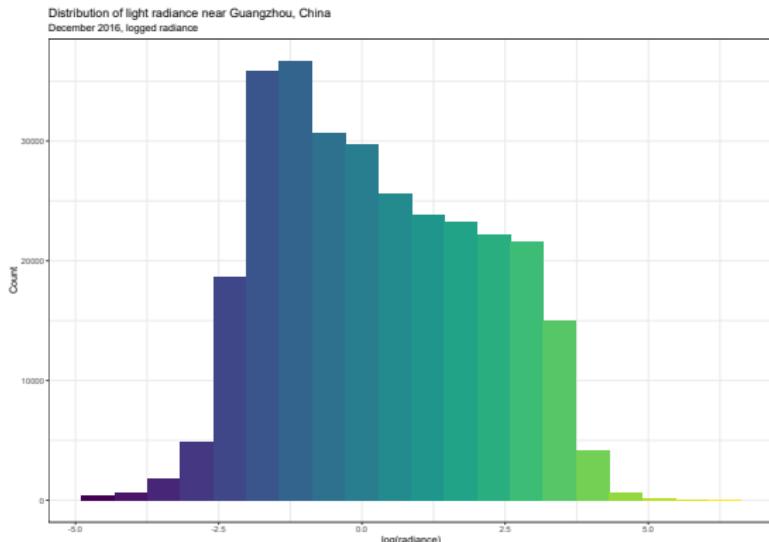
> # R's summary of the raster
> nl_gz_201912 %>% summary()
nl_gz_1912_v3
Min.           -0.11
1st Qu.         0.28
Median          1.05
3rd Qu.         5.77
Max.            562.98
NA's             0.00
Warning message:
In .local(object, ...) :
  summary is an estimate based on a sample of 1e+05 cells (33.65% of all cells)
```

## Summerize the Raster Data (Cont'd)

```
> # The slot names
> nl_gz_201912 %>% slotNames()
[1] "file"      "data"       "legend"     "title"      "extent"     "rotated"
[7] "rotation"   "ncols"     "nrows"     "crs"       "history"    "z"
> # The first 6 values
> nl_gz_201912 %>% getValues() %>% head()
[1] 0.05 0.08 0.02 0.05 0.14 0.18
> # The first six coordinates
> nl_gz_201912 %>% coordinates() %>% head()
      x        y
[1,] 112.4792 24.09167
[2,] 112.4833 24.09167
[3,] 112.4875 24.09167
[4,] 112.4917 24.09167
[5,] 112.4958 24.09167
[6,] 112.5000 24.09167
```

# A histogram of the raster's values (logged)

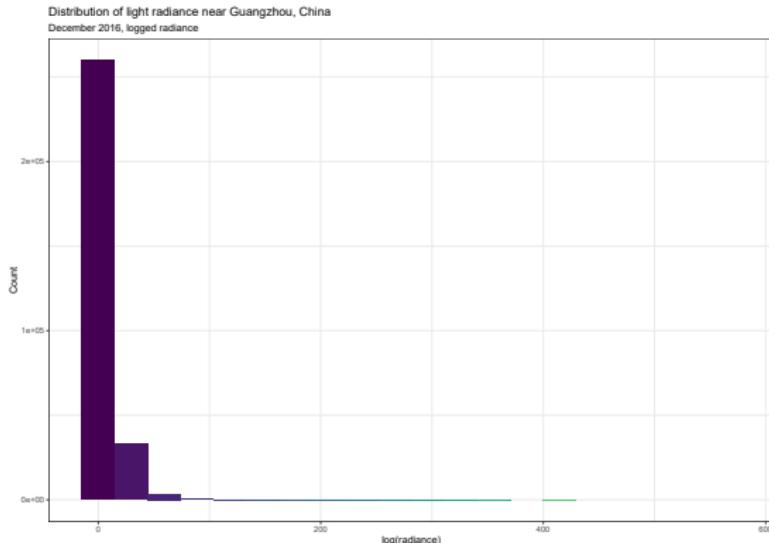
```
53 ## A histogram of the raster's values (logged):  
54 ggplot(data = data.frame(light = getValues(nl_gz_201912)),  
55   aes(x = log(light))) +  
56   geom_histogram(bins = 20, fill = viridis(20)) +  
57   ylab("Count") +  
58   xlab("log(radiance)") +  
59   ggttitle("Distribution of light radiance near Guangzhou, China",  
60             subtitle = "December 2016, logged radiance") +  
61   theme_bw()
```



# What if we do not take log?

- There are some extreme values at both sides

```
63 ggpplot(data = data.frame(light = getValues(nl_gz_201912)),  
64   aes(x = light)) +  
65   geom_histogram(bins = 20, fill = viridis(20)) +  
66   ylab("Count") +  
67   xlab("log(radiance)") +  
68   ggtitle("Distribution of light radiance near Guangzhou, China",  
69     subtitle = "December 2016, logged radiance") +  
70   theme_bw()
```



## Reclassify Value in a Raster

- We need to reclassify the negative and extreme large values in the data using the **reclassify()**
  - We first need to generate a 3-cols matrix with the range and new value information
  - Then apply the **reclassify()** and generate a new raster data

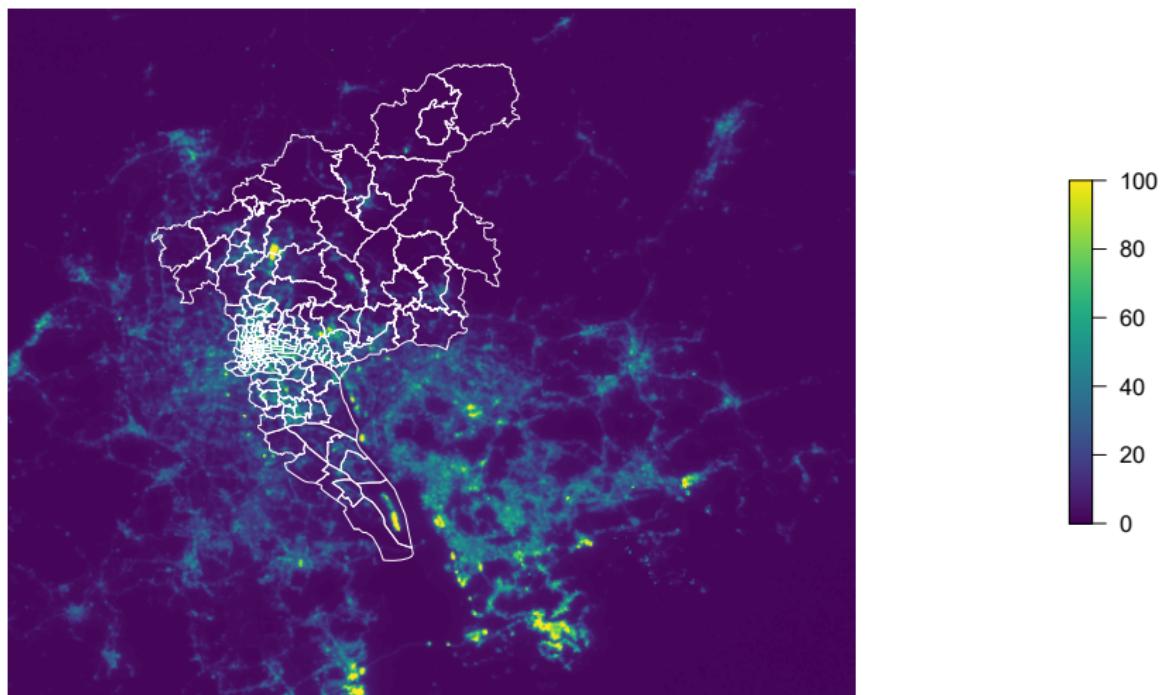
```
## reclassify value in a raster
bl <- c(-0.13, 0:100)
bh <- c(0:100, 600)
nv <- c(0:100, 100)
m <- cbind(bl, bh, nv)
nl_gz_201912_re <- reclassify(nl_gz_201912, m)
```

I

## Replot the Raster

- The variation is much clearer after excluding the extreme values

```
79 plot(nl_gz_201912_re, col = viridis(1e3), axes = F, box = F)  
80 lines(gz_t, col = "white", lwd = 0.8)
```



# Today

## ► Raster Data

- ① Read Raster Data
- ② Plot and Summarize Raster Data
- ③ Reclassify Value in a Raster
- ④ Rasters in ggplot2
- ⑤ Masks
- ⑥ Summarize Rasters with Polygons
- ⑦ Summarizing Rasters with Points
  - \* An Application
- ⑧ Read in NetCDFs Raster Data
- ⑨ Creating Raster Data

## ► Geocoding

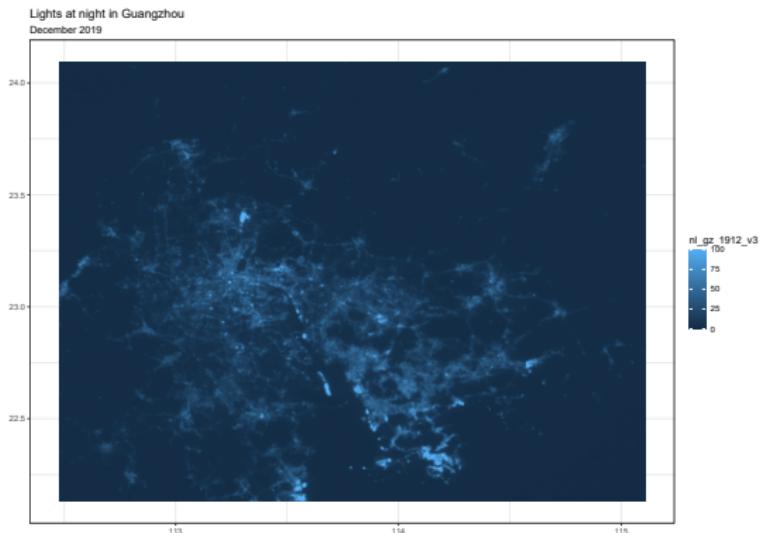
# Plot the Raster using ggplot2

- ▶ The problem is that `ggplot()` wants a data-frame-like object, and the raster is not a data-frame-like object (think about `sf` object)
- ▶ Luckily, we can use `rasterToPoints` to convert the raster to a matrix of coordinates and the corresponding values from a raster
- ▶ And then we can convert the matrix to a data frame

```
82 # 1.4 Rasters in ggplot2
83 ## Convert raster to matrix and then to data frame
84 nl_gz_201912_df <- nl_gz_201912_re %>% rasterToPoints() %>% tbl_df()
85 ## Plot with ggplot|
86 ggplot(data = nl_gz_201912_df,
87         aes(x = x, y = y, fill = nl_gz_1912_v3)) +
88         geom_raster() +
89         ylab("") + xlab("") +
90         ggtitle("Lights at night in Guangzhou",
91                 subtitle = "December 2019") +
92         theme_bw()
```

## Plot the Raster using ggplot2 (Cont'd)

- ▶ The problem is that **ggplot()** wants a data-frame-like object, and the raster is not a data-frame-like object (think about sf object)
- ▶ Luckily, we can use **rasterToPoints** to convert the raster to a matrix of coordinates and the corresponding values from a raster
- ▶ And then we can convert the matrix to a data frame



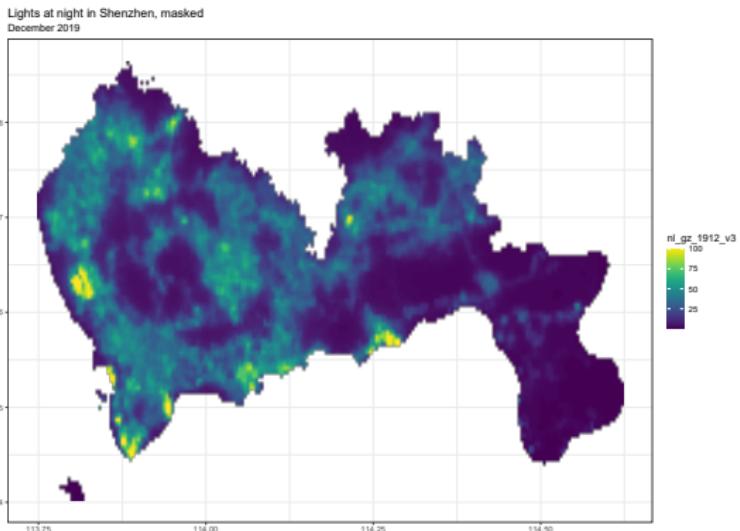
# Masks

- ▶ You may have noticed that our raster includes many locations that are not within Shenzhen city
  - We switch to the example to Shenzhen for the next task
- ▶ One possibility is to use **crop()** to crop the raster to a rectangle covering the city boundary
- ▶ Another possibility is use the **mask()** to get the exact area of our polygon file
  - But first we use the **gUnaryUnion()** from **rgeos** package to dissolve the block map to get the boundary map of Shenzhen city

```
95 # 1.5 Masks
96 library(rgeos)
97 # Take the union of the shenzhen block polygons
98 sz_union <- gUnaryUnion(sz_t)
99 plot(sz_union)
100
101 # Mask night lights raster with shenzhen's outline
102 lights_masked <- mask(x = nl_gz_201912_re, mask = sz_union)
```

# Plot it!

```
104 # Convert raster to matrix and then to data frame  
105 masked_df <- lights_masked %>% rasterToPoints() %>% tbl_df()  
106 # Plot with ggplot  
107 ggplot(data = masked_df,  
108     aes(x = x, y = y, fill = nl_gz_1912_v3)) +  
109     geom_raster() +  
110     ylab("") + xlab("") +  
111     ggtitle("Lights at night in shenzhen, masked",  
112         subtitle = "December 2019") +  
113     scale_fill_viridis(option = "D") +  
114     theme_bw()
```



# Summarize Rasters with Polygons

- ▶ We want to take some sort of summary statistics of our raster, either at the level of a polygon or a point
  - Specifically, let us take the average level of light within each of block in our block district shapefile
- ▶ **extract()** from **raster** package for this task, with the following arguments:
  - x: the raster from which you want to extract values
  - y: the points at which you would like to extract values; in the current context, y is our shapefile, but it could also be lines or points
  - fun: the function we will use to summarize the raster (here: mean)
  - na.rm: do you want to remove missing values (NAs)?
  - df: (optional) do you want a data frame (rather than a matrix)
  - sp: (optional) do you want the outputted results attached to the data slot of y (requires the y is an sp object, i.e., spatial polygons or spatial lines)
- ▶ Because we would end up joining the results to our spatial polygons data frame (`st_t`), I will go ahead and specify `sp = TRUE`

## Summarize Rasters with Polygons (Cont'd)

- ▶ **extract()** is a pretty flexible function, thus you need to make the following decisions before hand
  - What counts as being in a polygon? Must the center of the cell be in the polygon, or is it enough that any part of the cell intersects the polygon?
  - If you require the center of the cell to be within the polygon, what happens when a polygon does not intersect with any cells' centroids?
  - Should all cells count equally, or should we weight by the amount of the cell covered by a polygon?
- ▶ These questions are especially relevant/important when your cells are large relative to your polygons

## Summarize Rasters with Polygons (Cont'd)

```
117 # 1.6 Summarize rasters with polygons
118 # Take averages of lights raster within each street block (jiedao and
119 xiangzhen)
120 gz_block_extract <- raster::extract(
121   x = nl_gz_201912_re,
122   y = gz_t,
123   fun = mean,
124   na.rm = T,
125   sp = T)
```

```
gz_block_extract  Large SpatialPolygonsDataFrame (170 elements, 1...
..@ data : 'data.frame': 170 obs. of 8 variables:
... ..$ town_code : chr [1:170] "440118102" "440118101" "440118106" ...
... ..$ province : chr [1:170] "騷弯箇鏞\x81" "騷弯箇鏞\x81" "騷弯箇鏞\x81...
... ..$ city : chr [1:170] "騷垮窗甯\x82" "騷垮窗甯\x82" "騷垮窗甯\x82" ...
... ..$ county : chr [1:170] "澧怡煙鎮\xba" "澧怡煙鎮\xba" "澧怡煙鎮\xba" ...
... ..$ town : chr [1:170] "鍛虫哔閻\x87" "鋸板 閣\x87" "灏忲ゼ閻\x87" ...
... ..$ town_lng : num [1:170] 114 114 114 114 114 ...
... ..$ town_lat : num [1:170] 23.2 23.1 23.4 23.2 23.5 ...
... ..$ nl_gz_1912_v3: num [1:170] 6.71 19.19 1.59 7.74 1.34 ...
..@ polygons :List of 170
```

# Summarize Rasters with Points

- We can do the same with points

- For this example, let us load in the data of all the locations in Shenzhen where those confirmed coronavirus cases stayed
- The data is downloaded from  
[https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200\\_01503669](https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_01503669)



The screenshot shows the Shenzhen Government Data Open Platform (深圳市人民政府数据开放平台) website. The main navigation bar includes links for Home, Data, Applications, Images, News, Interaction, and User Profile. A search bar at the top has the URL [opendata.sz.gov.cn/data/dataSet/toDataDetails/29200\\_01503669](https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_01503669). Below the search bar is a banner with a blue gradient background featuring icons for government services like Health, Education, and Real Estate.

The main content area displays a dataset titled "深圳市“新型冠状病毒”-确诊患者曾逗留过的场所名单" (List of places where confirmed COVID-19 patients have stayed). It has a rating of 4.9 stars. Key details include:

- 更新日期: 2020-03-06 提供部门: 市政务服务数据管理局 发布时间: 2020-02-01 数据领域: 卫生健康 数据量: 251条 下载量: 1081次 浏览量: 3999次 开放级别: 普通开放
- 数据简介: 本数据是基于深圳市卫生健康委员会官方发布的疫情信息,由深圳市政务服务数据管理局清洗整理形成的结构化数据。本数据包括序号、行政区划、小区名称、发布时间、备注等5个字段。
- 更新频率: 与深圳市卫生健康委员会官方发布的疫情信息更新频率同步,通常为每天。
- 注: 本结构化数据的原始数据来自深圳市卫生健康委员会官方网站,若有出入,请以深圳市卫生健康委员会官方网站公布内容为准。

Below the dataset details, there are tabs for "文件" (File), "元数据" (Metadata), "关联信息" (Relationship Information), "数据项" (Data Items), "数据预览" (Data Preview), "可视化分析" (Visual Analysis), "调用API" (Call API), and "数据纠错" (Data Correction). At the bottom, there are download options for "XLSX", "XML", "JSON", "CSV", and "RDF", along with search and filter buttons for "开始日期" (Start Date) and "结束日期" (End Date), and buttons for "全选" (Select All) and "下载" (Download). A "数据申请" (Data Application) button is also present.

## Summarize Rasters with Points (Cont'd)

- ▶ The data contains xy coordinates for all these locations, from which we can convert into the point shapefile
- ▶ The procedure is a little more complicated in **sp**

```
126 # 1.7 Summarizing rasters with points
127 # Create a point shapefile from xy coordinate table using SpatialPointsDataFrame()
128 ## read in the csv file
129 sz_cases <- read.csv("sz_cases.csv")
130 # rearrange the order of cols in the dataframe
131 sz_cases <- sz_cases[, c(1:4, 6, 5)]
132 str(sz_cases) # we have longitude/latitude information inside
133 ## create object containing the CRS arguments
134 crs_sz <- crs(sz_t)
135 crs_sz
136 ## create a pointshapefile
137 sz_cases_sp <- SpatialPointsDataFrame(sz_cases[, 5:6],
138                                         sz_cases,
139                                         proj4string = crs_sz)
140 plot(sz_t, add=T)
141 plot(sz_cases_sp, add=T)
142
143 sz_pts_extract <- raster::extract(
144   x = nl_gz_201912_re,
145   y = sz_cases_sp,
146   fun = mean,
147   na.rm = T,
148   sp = T)
```

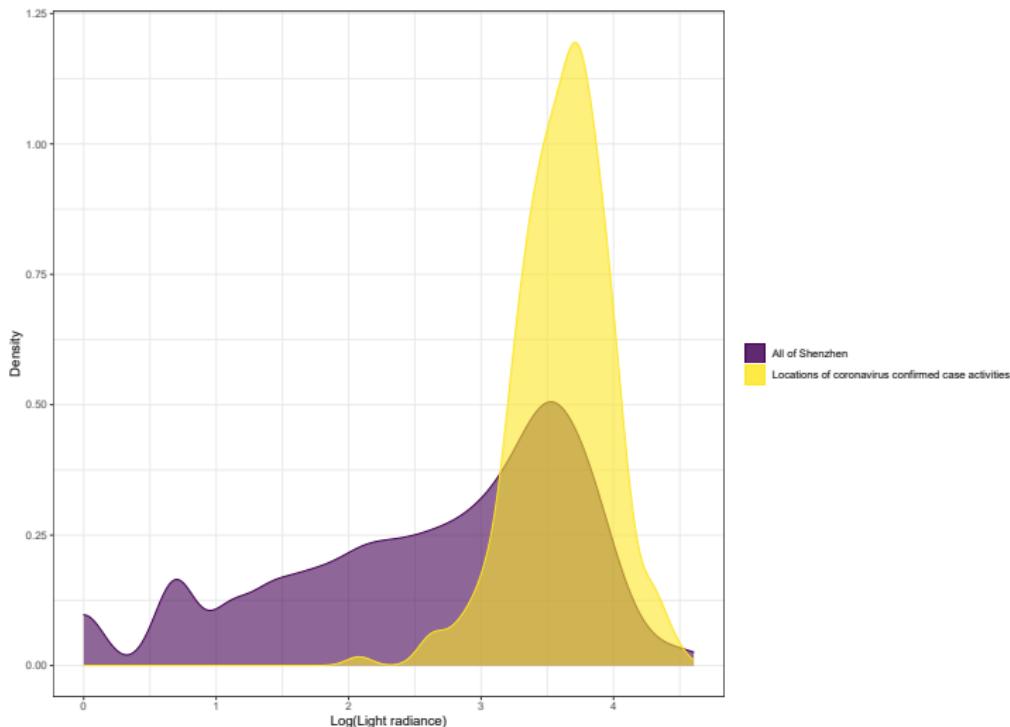
# A Simple Application

- We can then compare the (approximate) densities of (logged) light for (1) all of shenzhen, (2) coronavirus locations

```
150 # 1.7.2 Compare the (approximate) densities of (logged) light for (1) all of  
shenzhen, (2) coronavirus locations  
151 library(dplyr)  
152 # Convert sz_pts_extract to a data frame  
153 sz_pts_extract <- tbl_df(sz_pts_extract)  
154 # Density plots  
155 ggplot() +  
156 geom_density(  
157   data = masked_df,  
158   aes(x = log(nl_gz_1912_v3), color = "All of Shenzhen", fill = "All of  
Shenzhen"),  
159   alpha = 0.6) +  
160 geom_density(  
161   data = sz_pts_extract,  
162   aes(x = log(nl_gz_1912_v3), color = "Locations of coronavirus confirmed  
case activities", fill = "Locations of coronavirus confirmed case activities"  
),  
163   alpha = 0.6) +  
164 ylab("Density") +  
165 xlab("Log(Light radiance)") +  
166 scale_color_viridis("", discrete = T) +  
167 scale_fill_viridis("", discrete = T) +  
168 theme_bw()
```

## A Simple Application (Cont'd)

- We can then compare the (approximate) densities of (logged) light for (1) all of shenzhen, (2) coronavirus locations



# A Simple Application (Cont'd)

- ▶ Try to add the POI data for comparison

## What is POI Data?

Point of interest (POI) data is information about the function and geographical location of a place that people might find interesting. Points of interest are all around us, and are often simple things like parks, shops, bars, gas stations, and hotels.

A point of interest can be either permanent, like a heritage site or monument, or temporary like a small shop or pop-up attraction. Regardless of how long they're around for, finding and monitoring POIs has never been easier, thanks to the online map on every one of our smartphones. This has led to greater customer and business demand for quality information about POIs..

Because POIs are in such high demand and are now so easy to find, they are frequented by masses of people. This makes them are absolute hotspots for gathering data, which can then be used to learn more about the location and the people visiting it. POI data is attractive to a wide range of users and is commonly implemented when evaluating locations and understanding behaviour patterns.



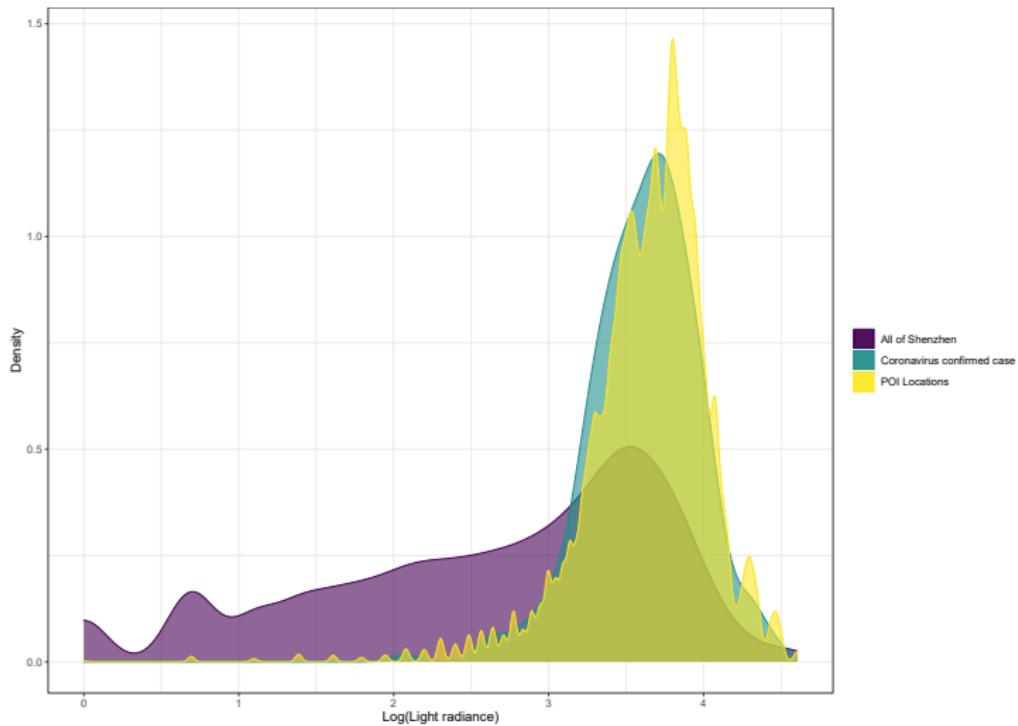
- ▶ Sources: [wikipedia.cn](https://wikipedia.cn) and <https://datarade.ai/data-categories/point-of-interest-poi-data/guide>

## A Simple Application (Cont'd)

- ▶ Load in the baidu POI data for Guangdong province in 2019
- ▶ Some pre-processing before the next step:
  - Convert the longitude and latitude variables from string to numeric class
  - Remove those observations (rows) with missing longitude or latitude
  - Detect and Extract those points outside Shenzhen city using `over()`

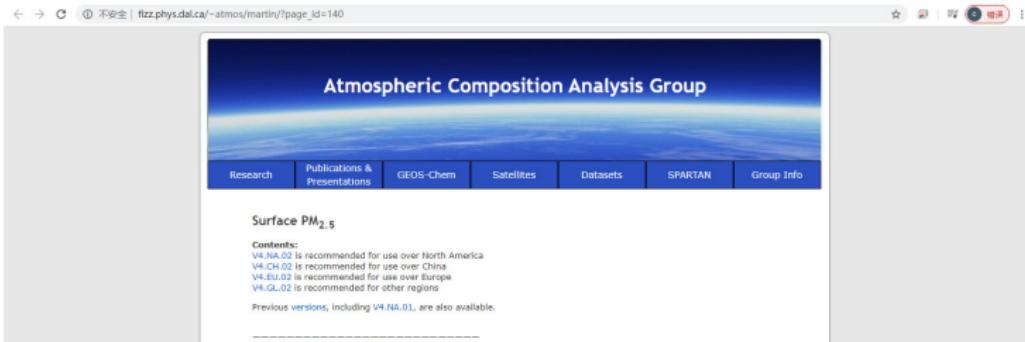
```
170 ## read in the guangdong poi csv file
171 gd_poi <- read.csv("gd_poi.csv")
172 str(gd_poi) # we have longitude/latitude information inside
173 ## create a pointshapefile
174 gd_poi_sp <- SpatialPointsDataFrame(gd_poi[, 1:2],
175                                     gd_poi,
176                                     proj4string = crs_sz)
177 ## Take the union of the shenzhen polygons
178 sz_union <- gUnaryUnion(sz_t)
179 ## Check which crime points are inside of Shenzhen
180 test_points <- over(gd_poi_sp, sz_union)
181 # From 1 and NA to T and F (F means not in Shenzhen)
182 test_points <- !is.na(test_points)
183 sz_poi_sp <- gd_poi_sp[which(test_points == T), ]
184
185 sz_poi_extract <- raster::extract(
186   x = nl_gz_201912_re,
187   y = sz_poi_sp,
188   fun = mean,
189   na.rm = T,
190   sp = T)
```

## A Simple Application (Cont'd)



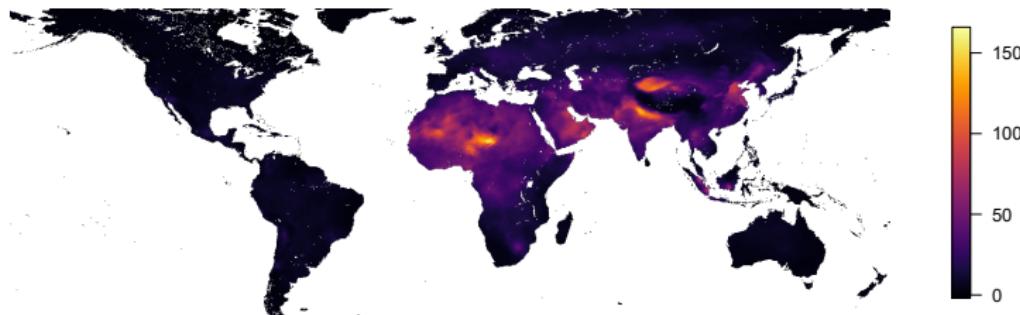
# Read in NetCDFs Raster Data

- ▶ If you explore more on the world of geospatial data, sooner or later you will encounter netCDF data (file extension: .nc)
  - NetCDF is a file type developed by NASA to contain raster data—gridded latitudes and longitudes paired with values, which is common to more (physical) scientific datasets
- ▶ The **ncdf4** package has a bunch of tools for working with (reading, creating, and manipulating) netCDFs
- ▶ For demonstration today, we will use two .nc downloaded from Atmospheric Composition Analysis Group at Dalhousie University, contain estimates for annual mean PM<sub>2.5</sub> levels for 2010 and 2016



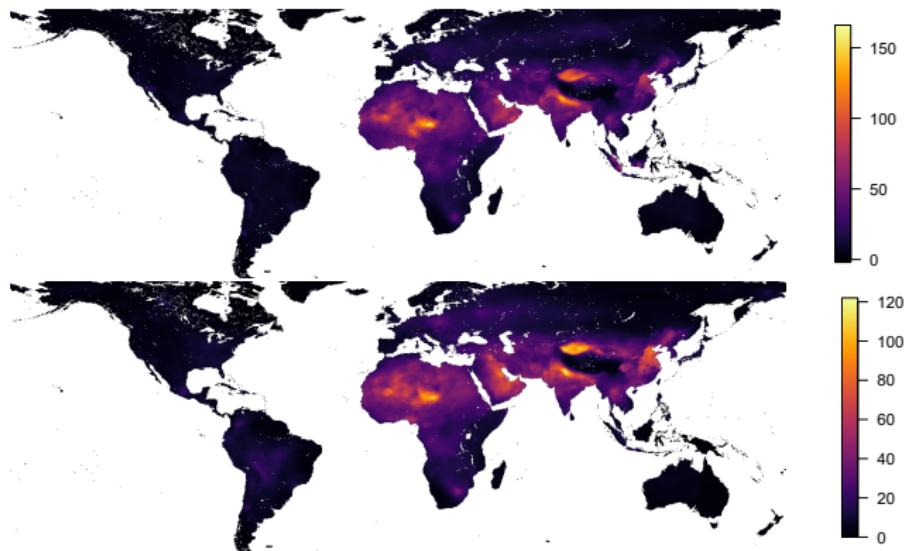
## Read in NetCDFs Raster Data (Cont'd)

```
218 # 1.8 read in NetCDFs data
219 ## Download the pm2.5 raster data from http://fizz.phys.dal.ca/~atmos/martin/?page\_id=140
220 ## install.packages("ncdf4")
221 library(ncdf4)
222 pm_2016 <- raster("GlobalGWRwUni_PM25_GL_201601_201612-RH35_Median.nc")
223 pm_2010 <- raster("GlobalGWRwUni_PM25_GL_201001_201012-RH35_Median.nc")
224 pm_2016
225 pm_2010
226 plot(pm_2016, col = inferno(1e3), axes = F, box = F)
227 plot(pm_2010, col = inferno(1e3), axes = F, box = F)
228
229 plot(pm_2016 - pm_2010, col = inferno(1e3), axes = F, box = F)
230
231 ## Min and max values of PM difference (2016 - 2010)
232 diff_min <- getValues(pm_2016 - pm_2010) %>% min(na.rm = T)
233 diff_max <- getValues(pm_2016 - pm_2010) %>% max(na.rm = T)
```



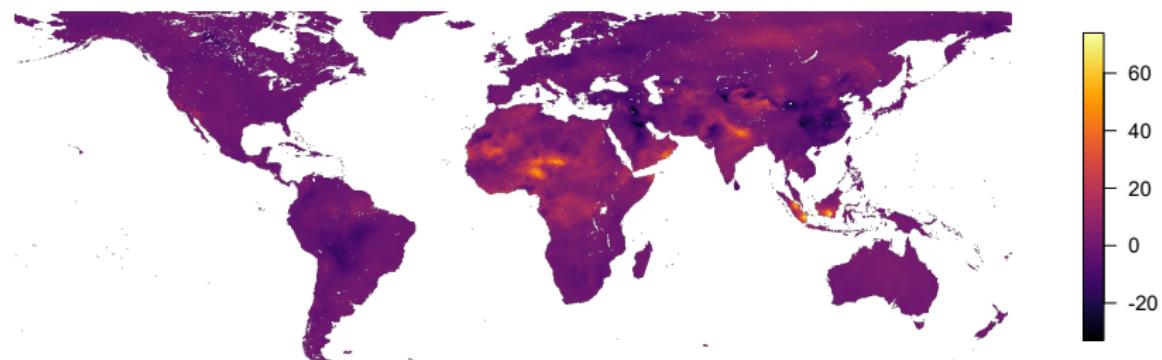
# Compare across Maps

- ▶ It is hard to compare across two maps, since the color scales are different
- ▶ We can improve either by 1) create a constant color scale using the `scale_*``_gradient` function in `ggplot2`, or 2) perform mathematic operations using subtraction (-)



## Compare across Maps (Cont'd)

```
229 | plot(pm_2016 - pm_2010, col = inferno(1e3), axes = F, box = F)
```



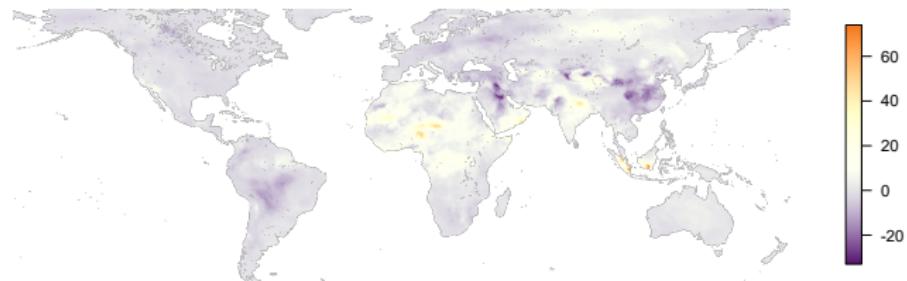
## Skill to Emphasize Contrast in Raster Plot

- ▶ The color scale is still not super informative: zero is stuck in a arbitrary dark purple region
- ▶ We can improve by assigning different colors to the minimum and maximum values and another color scale to other points in between
  - Use the **designer.colors()** function from the **fields** package to generate color scales

```
239 ## Min and max values of PM difference (2016 - 2010)
240 diff_min <- getvalues(pm_2016 - pm_2010) %>% min(na.rm = T)
241 diff_max <- getvalues(pm_2016 - pm_2010) %>% max(na.rm = T)
242
243 ## Create the color scale
244 library(maps)
245 library(fields)
246 c_scale <- designer.colors(
247   n = 500,
248   col = c(inferno(100)[25], "grey92", inferno(100)[70]),
249   x = c(diff_min, 0, diff_max),
250   alpha = 1
251 )
252 ## Plot again
253 plot(pm_2016 - pm_2010, col = c_scale, axes = F, box = F)
```

## Skill to Emphasize Contrast in Raster Plot (Cont'd)

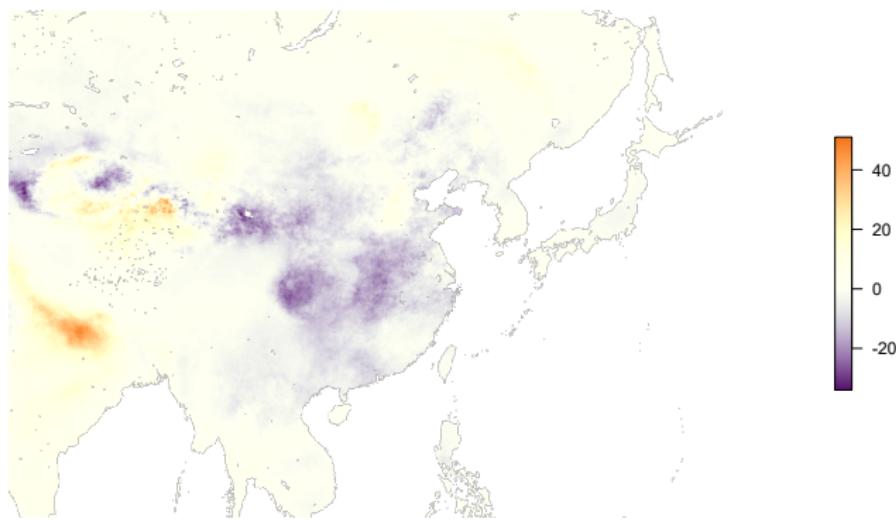
- ▶ The color scale is still not super informative: zero is stuck in a arbitrary dark purple region
- ▶ We can improve by assigning different colors to the minimum and maximum values and another color scale to other points in between
  - Use the `designer.colors()` function from the **fields** package to generate color scales



## Skill to Emphasize Contrast in Raster Plot (Cont'd)

- We can zoom in to China using `crop()`

```
255 ## zoom in china  
256 crop(pm_2016 - pm_2010, extent(75, 150, 10, 55)) %>%  
257   plot(col = c_scale, axes = F, box = F)
```



## Raster Stack

- ▶ We can load multiple raster files into a raster stack
- ▶ Particularly useful when we are performing operations on a bunch of rasters

```
236 ## Load multiple raster files into a raster stack
237 pm_stack <- stack(
238   "GlobalGWRwUni_PM25_GL_201601_201612-RH35_Median.nc",
239   "GlobalGWRwUni_PM25_GL_201001_201012-RH35_Median.nc")
240 # Check
241 pm_stack
```

I

```
class      : RasterStack
dimensions : 12470, 36000, 448920000, 2  (nrow, ncol, ncell, nlayers)
resolution : 0.01, 0.01  (x, y)
extent     : -180, 180, -54.85, 69.85  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0
names      : PM25.1, PM25.2
```

# Geocoding

- ▶ Geocoding refers finding the geographic coordinates for a location (or locations) (e.g. taking addresses and finding their latitudes and longitudes)
  - There are many map services/APIs that allow you to geocode (e.g. google map service)

## Compare Geocoding Providers

	Geocodio	SmartyStreets	Google	Bing	Here	Mapbox
Free Lookups	2,500/day	250/month	avg. 1,333/day ?	10,000/month ?	avg. 1,111/day ?	100,000/month
Pay-As-You Go	\$0.10-0.50 per 1,000	✗	\$4-5 per 1,000 n/a	n/a ?	n/a ?	\$0.45-0.75 per 1,000
Unlimited Subscription	✓ \$1,000/month	✓ \$1,000/month	✗	✗	✗	✗
No Subscription Required	✓	✓	✓	✗	✗	✓
No Attribution Requirements or Usage Restrictions	✓	✗ ?	✗ ?	✗ ?	✗ ?	✗ ?
CSV File Upload	✓	✓	✗	✗	✗	✗
API	✓	✓	✓	✓	✓	✓
Bulk Geocoding	✓	✓	✗	✓	✓	✓
Reverse Geocoding	✓	✗	✓	✓	✓	✓
Congressional	✓	✓				

# Geocoding (Cont'd)

- Remember the **ggmap** package? We can use the **geocode()** for the task
  - Do not forget to apply for your free API key from <https://console.cloud.google.com/>

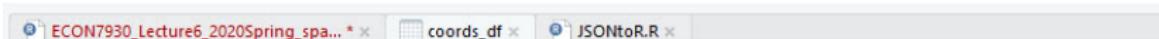
```
293 # 2 Geocoding
294 library(ggmap)
295 # Geocode with Google using geocode()
296 geo_google <- geocode(
297   "香港浸會大學逸夫校園永隆銀行商學大樓",
298   output = "more",
299   source = "google")
300
301 # get a google api to download the basemap from google
302 register_google(key = "AizaSyDsh0yKm-aUByFwfSdEgGkqIUrZMBL7BU")
303 ## Again, this one is fake, apply and use your own API from https://console.cloud.google.com/
```

	lon	lat	type	loctype	address	north	south	east	west
1	114.1816	22.33778	establishment	geometric_center	hong kong baptist university shaw campus, kowloon ts...	22.33913	22.33644	114.1829	114.1802

# Geocoding using Baidu Map Service

- We can also use the Baidu Map API for geocoding
  - To install the **baidugeo** package from github  
(`devtools::install_github("ChrisMuir/baidugeo")`)
  - API key can be applied through <https://lbsyun.baidu.com/>

```
306 ## geocoding using baidu map service
307 devtools::install_github("ChrisMuir/baidugeo")
308 library(baidugeo)
309 bmap_set_key("jNKbY2xrghowxxxxxxxxFgvqseOrk")
310 ## Again, this one is fake, apply and use your own API from https://lbsyun.baidu.com/
311
312 bmap_rate_limit_info()
313 bmap_set_daily_rate_limit(20000)
314
315 locs <- c(
316   "中百超市有限公司长堤街二分店",
317   "浙江省杭州市余杭区径山镇小古城村",
318   "成都高粱红餐饮管理有限公司"
319 )
320
321 coords_df <- bmap_get_coords(locs)
322 coords_df
```



The screenshot shows the RStudio interface with three tabs open: 'ECON7930\_Lecture6\_2020Spring\_sp...', 'coords\_df', and 'JSONtoR.R'. Below the tabs, there is a 'Filter' button. The 'coords\_df' tab is active, displaying a data frame with the following columns: location, lon, lat, status, precise, confidence, comprehension, and level. The data is as follows:

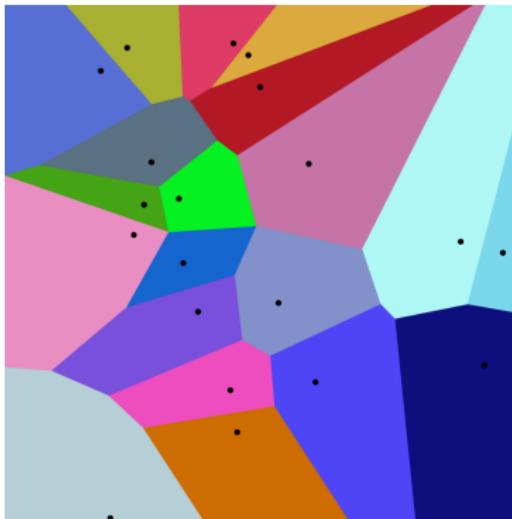
	location	lon	lat	status	precise	confidence	comprehension	level
1	中百超市有限公司长堤街二分店	106.7977	29.72313	0	0	75	0	购物
2	浙江省杭州市余杭区径山镇小古城村	119.8827	30.39242	0	0	50	100	村庄
3	成都高粱红餐饮管理有限公司	104.0931	30.65378	0	1	80	100	公司企业

# Reverse Geocoding

- ▶ We can reverse engineer the process to get address from coordinates (so-called reverse geocoding) usign **bmap\_get\_location()**

# Voronoi Diagram

- ▶ We frequently need to find the nearest hospital, surgery or supermarket. A map divided into cells, each cell covering the region closest to a particular centre, can assist us in our quest
- ▶ Another practical problem is to choose a location for a new service, such as a school, which is as far as possible from existing schools while still serving the maximum number of families



# Voronoi Diagram with Train Stations as Seeds



# Geocomputation

where to set up new infrastructure for cycling?

- ▶ Cycling can be used:
  - replace short distance car travel (identify these routes)
  - take people to nearby train stations
- ▶ Geocomputation help us find these routes

