Interactive Maps for Dummies:

Lecture 3: Raster Data and Geocomputation

SHI Shuang

The Univeristy of Hong Kong

Visiting Research Fellow at Brown University

Jul. 29, 2020

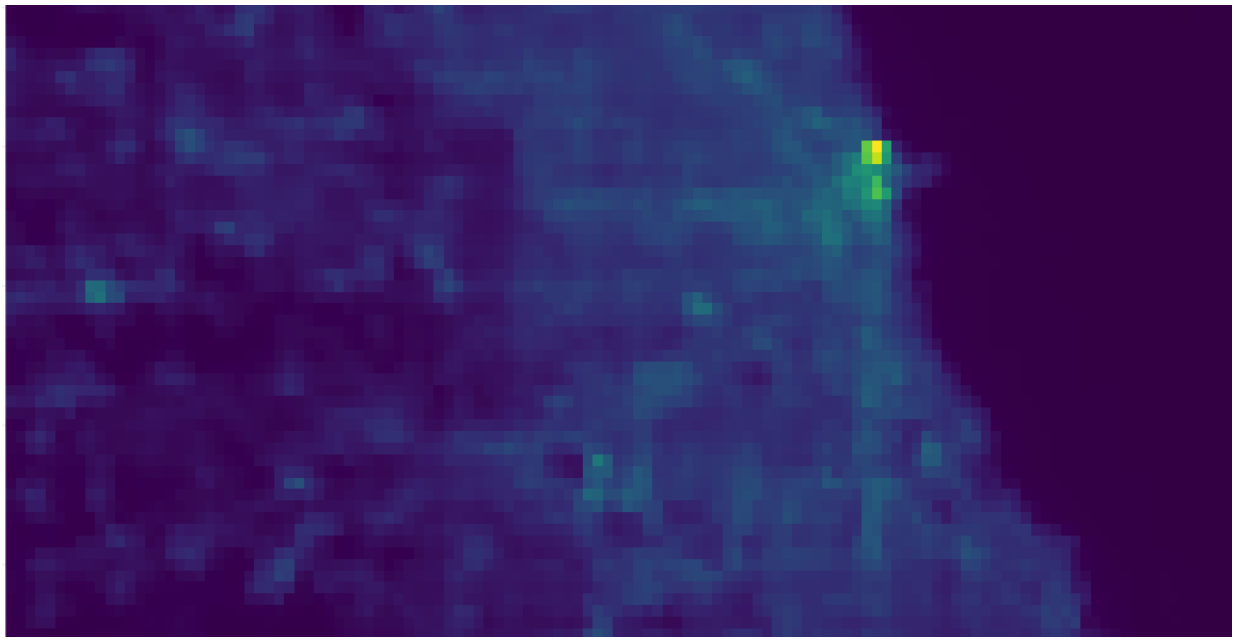
Prepared for HKBU Zoom Lectures

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.)



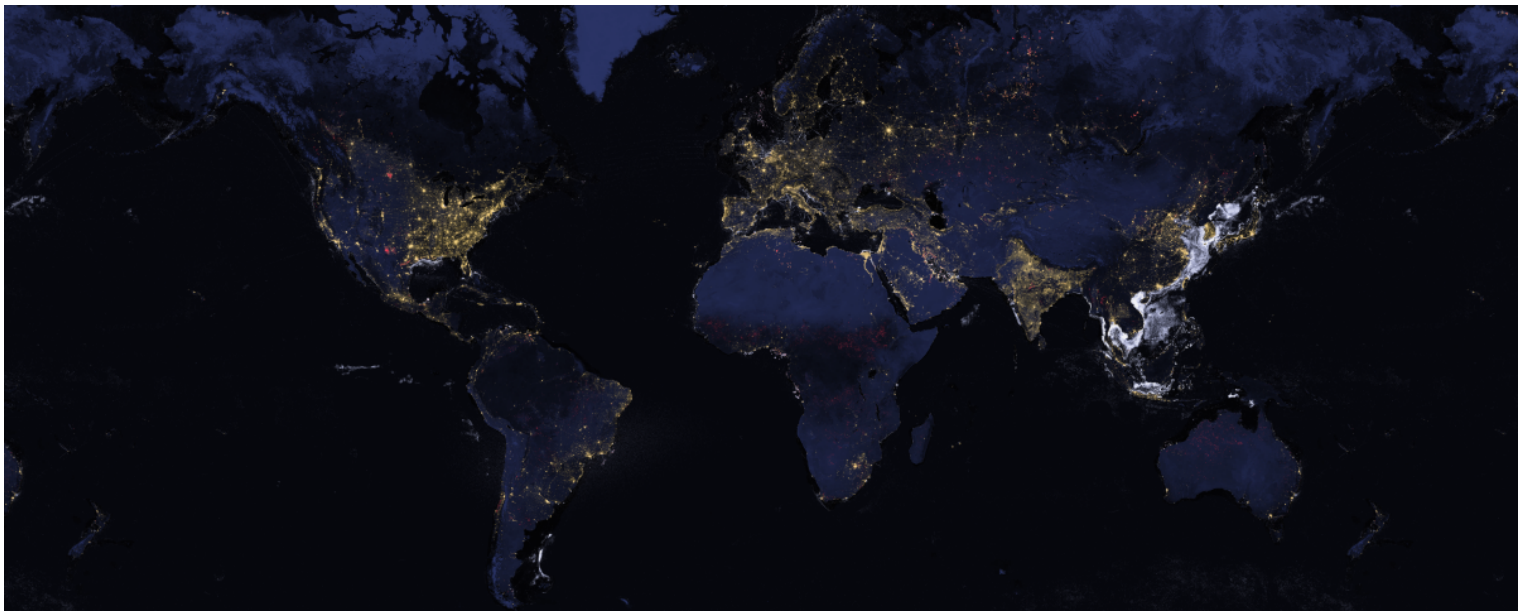
Nightlight Raster Data Developed by EOG

- 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

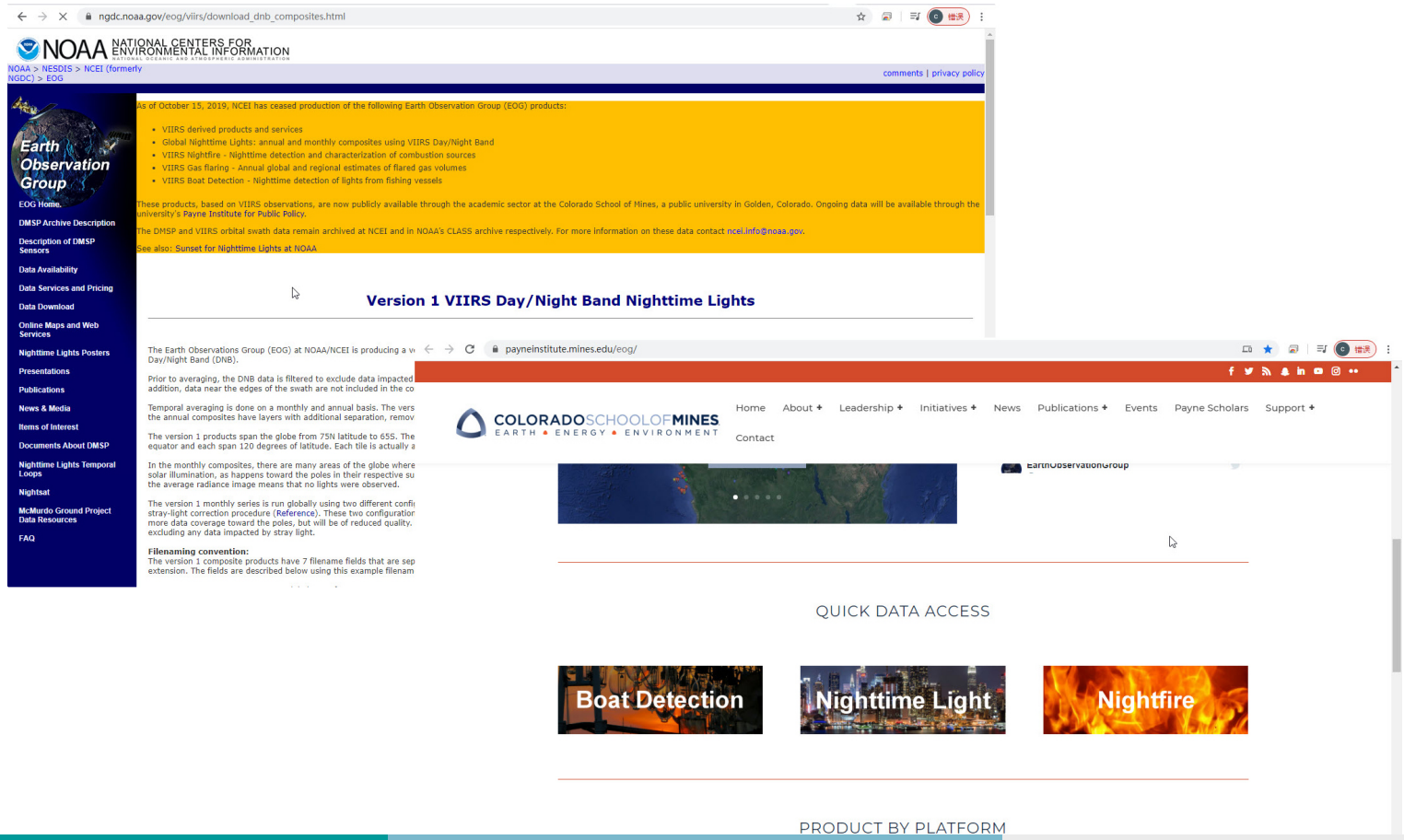
    "Lights, Camera...Income! Illuminating the National Accounts-Household Surveys Debate",

    https://academic.oup.com/qje/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

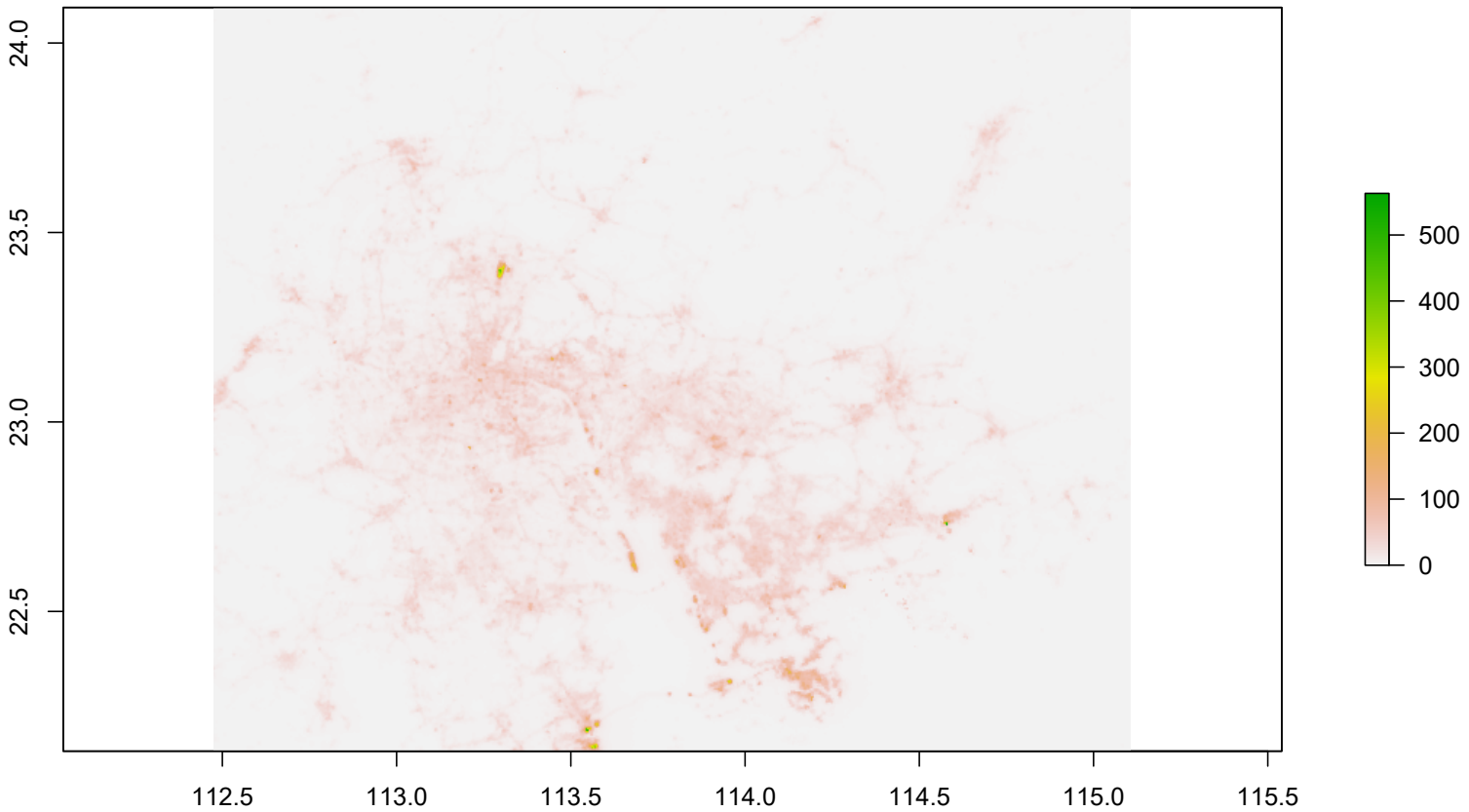


Prepare R

```
# General R setup ----
rm(list = ls())
getwd()
setwd("D:/Dropbox/course/ECON4016/Lecture6_2020Spring")

# Load new packages
install.packages("pacman")
library(pacman)
p_load(rgeos, GISTools)
# Load old packages
p_load(lubridate, rgdal, raster, broom, rgeos, GISTools)
p_load(dplyr, sp, ggplot2, ggthemes, fields, magrittr, viridis, ggmap)
# My ggplot2 theme
theme_ed <- theme(
  legend.position = "bottom",
  panel.background = element_rect(fill = NA),
  # panel.border = element_rect(fill = NA, color = "grey75"),
  axis.ticks = element_line(color = "grey95", size = 0.3),
  panel.grid.major = element_line(color = "grey95", size = 0.3),
  panel.grid.minor = element_line(color = "grey95", size = 0.3),
  legend.key = element_blank())
# The directory stored the nightlight data
dir_nl <- "E:/VIIRS_Nightlight/"
```

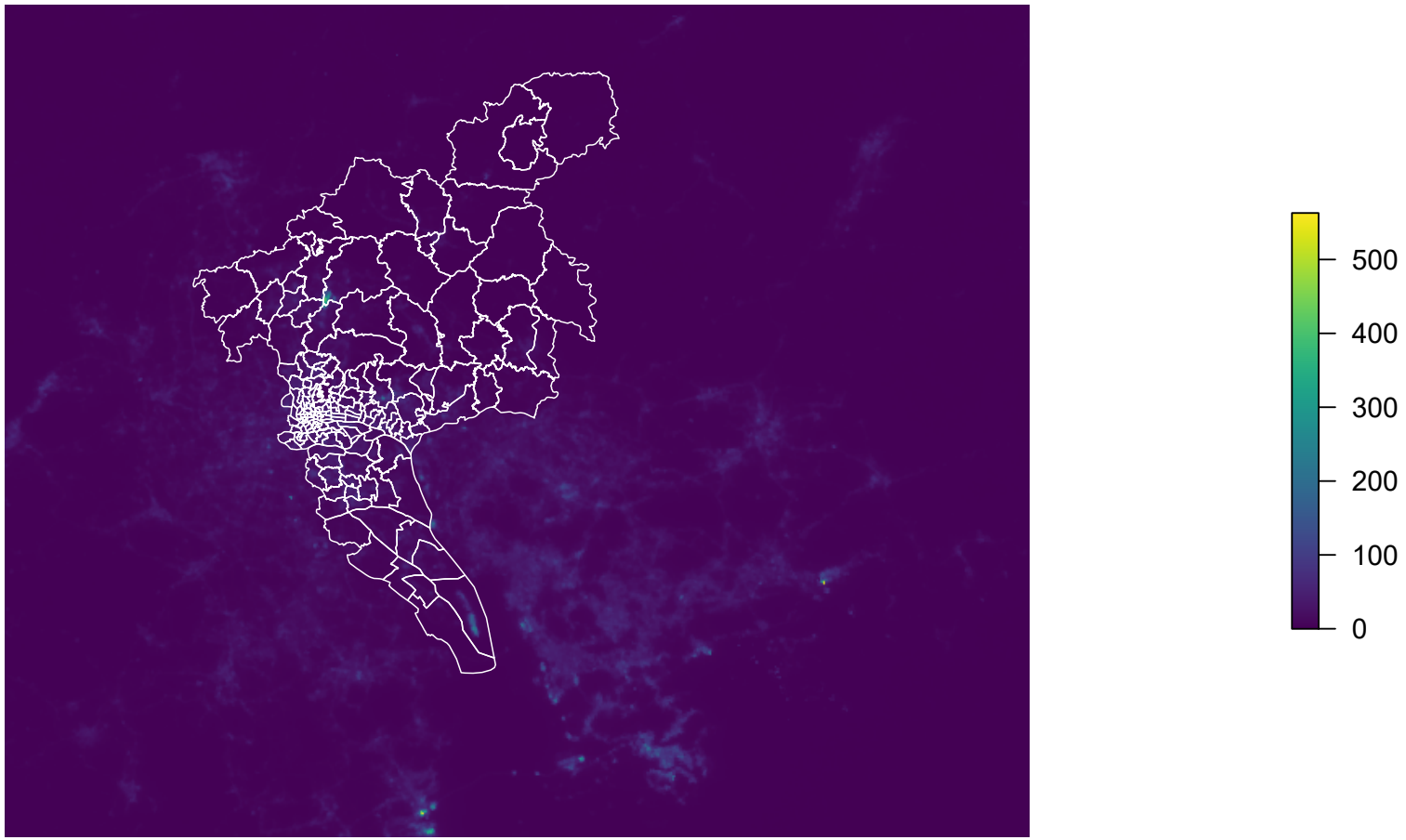
Read Raster Data

```
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

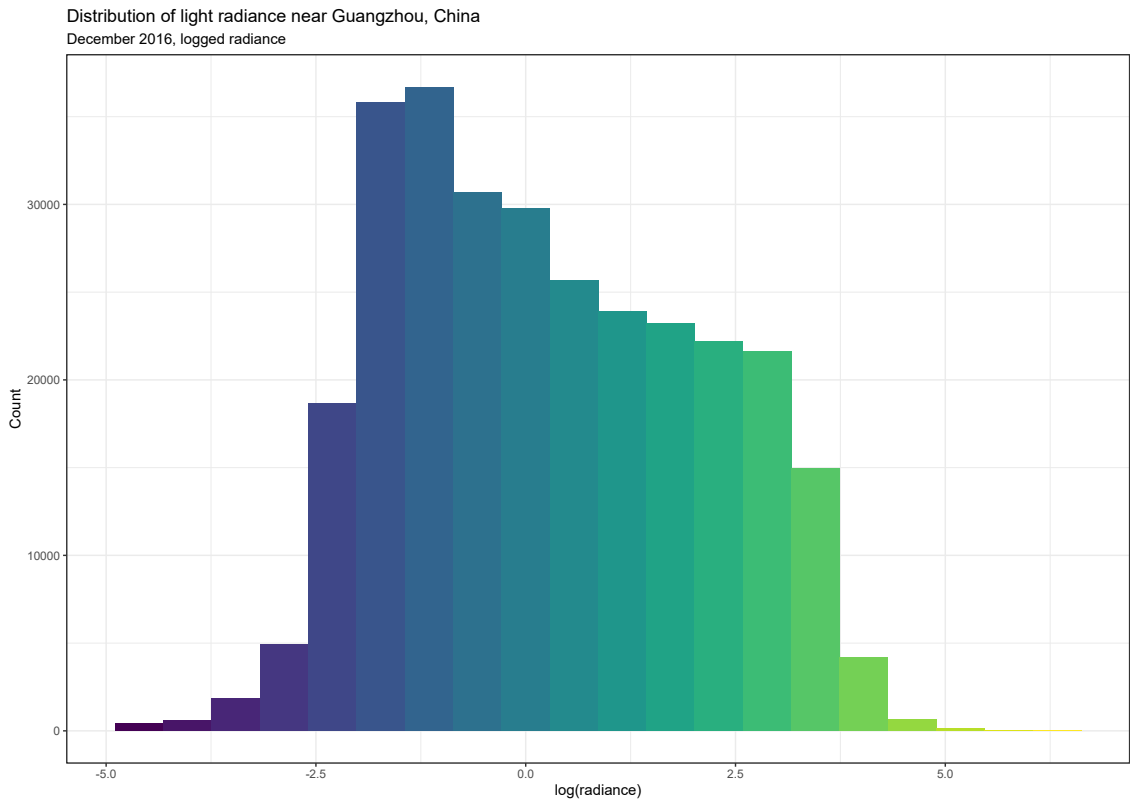
```
> # R's description of the raster
> nl_gz_201912
class       : RasterLayer
dimensions  : 471, 631, 297201  (nrow, ncol, ncell)
resolution  : 0.004166667, 0.004166667  (x, y)
extent      : 112.4771, 115.1063, 22.13125, 24.09375  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs +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)

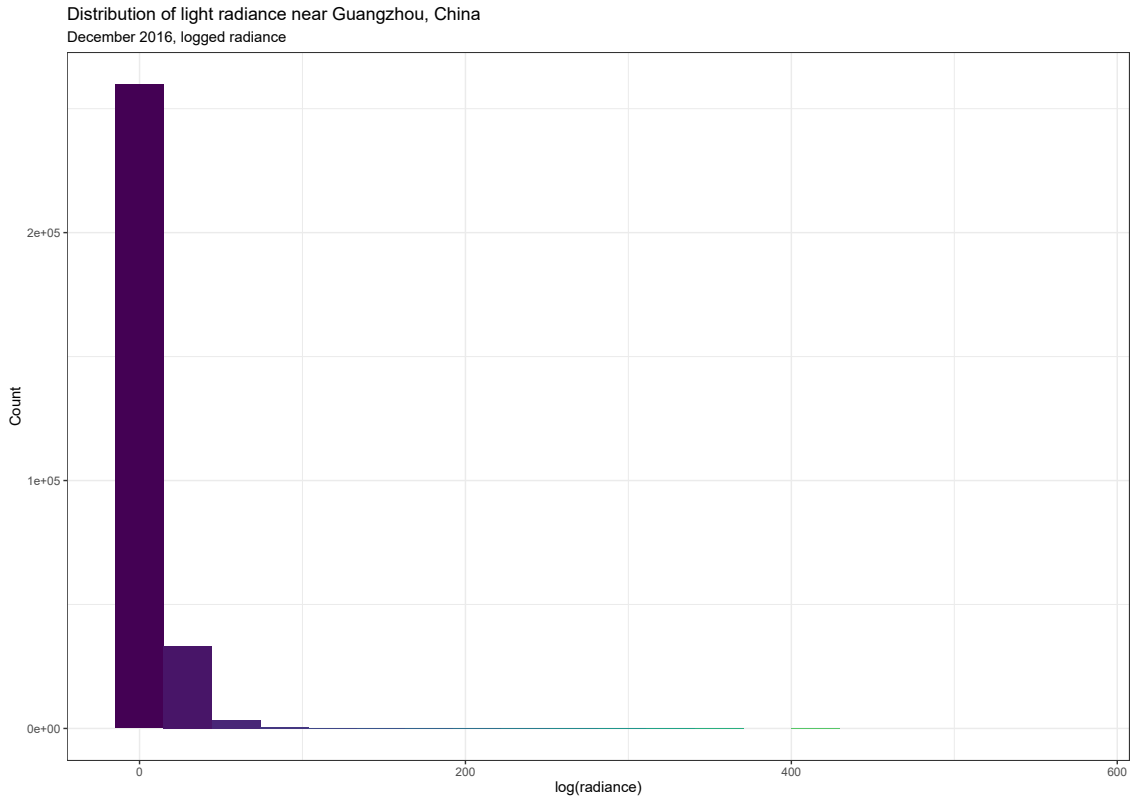
```
53  ## A histogram of the raster's values (logged):
54  ggplot(data = data.frame(light = getvalues(nl_gz_201912)),
55         aes(x = log(light))) +
56    geom_histogram(bins = 20, fill = viridis(20)) +
57    ylab("Count") +
58    xlab("log(radiance)") +
59    ggtitle("Distribution of light radiance 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

```
ggplot(data = data.frame(light = getValues(nl_gz_201912)),
       aes(x = light)) +
  geom_histogram(bins = 20, fill = viridis(20)) +
  ylab("Count") +
  xlab("log(radiance)") +
  ggtitle("Distribution of light radiance near Guangzhou, China",
          subtitle = "December 2016, logged radiance") +
  theme_bw()
```



Reclassify Value in a Raster

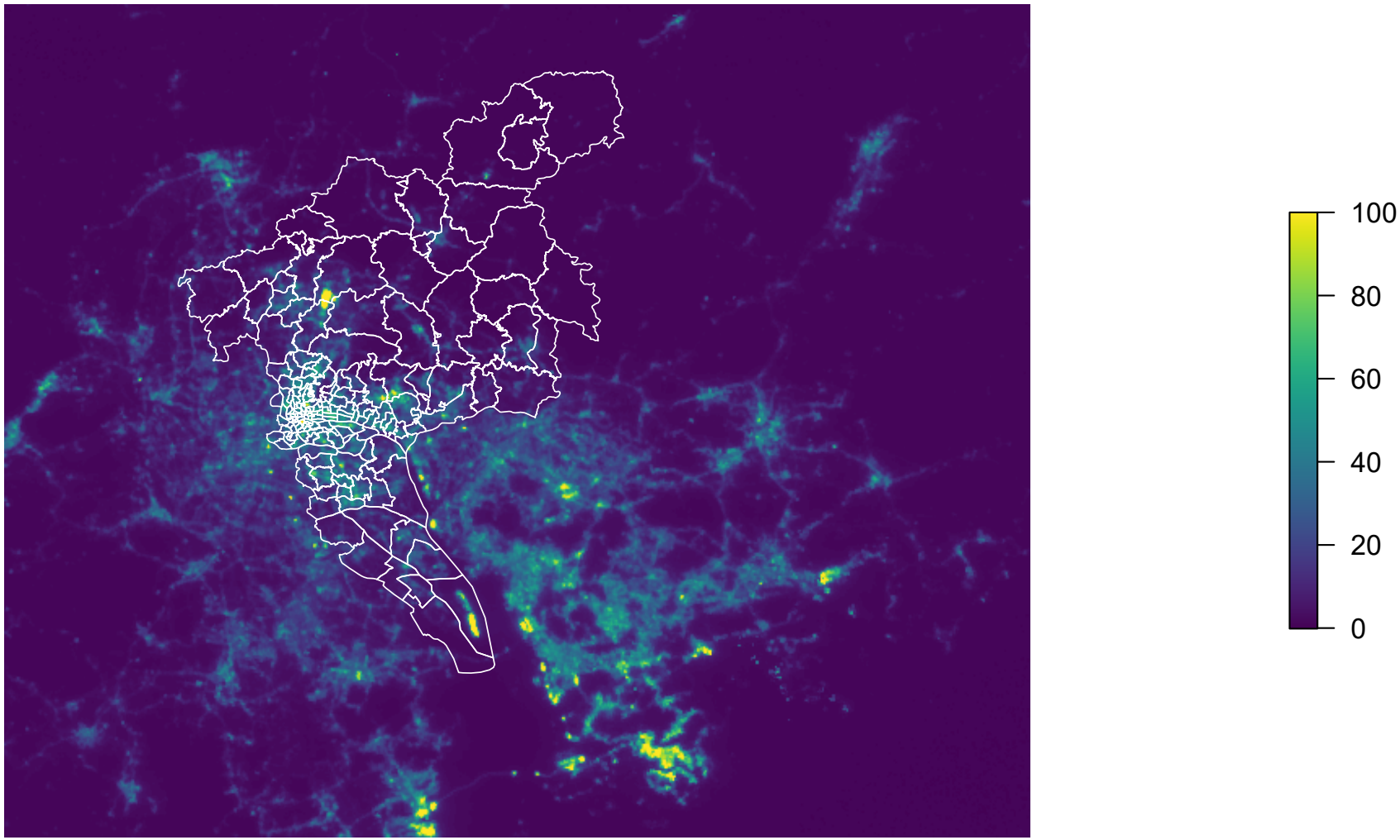
- We need to reclassify the negative and extreme large values in the data using the **reclassify()**
  - We first need to generate a 3-cols matrix with the range and new value information
  - Then apply the **reclassify()** and generate a new raster data

```
## 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)
```

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

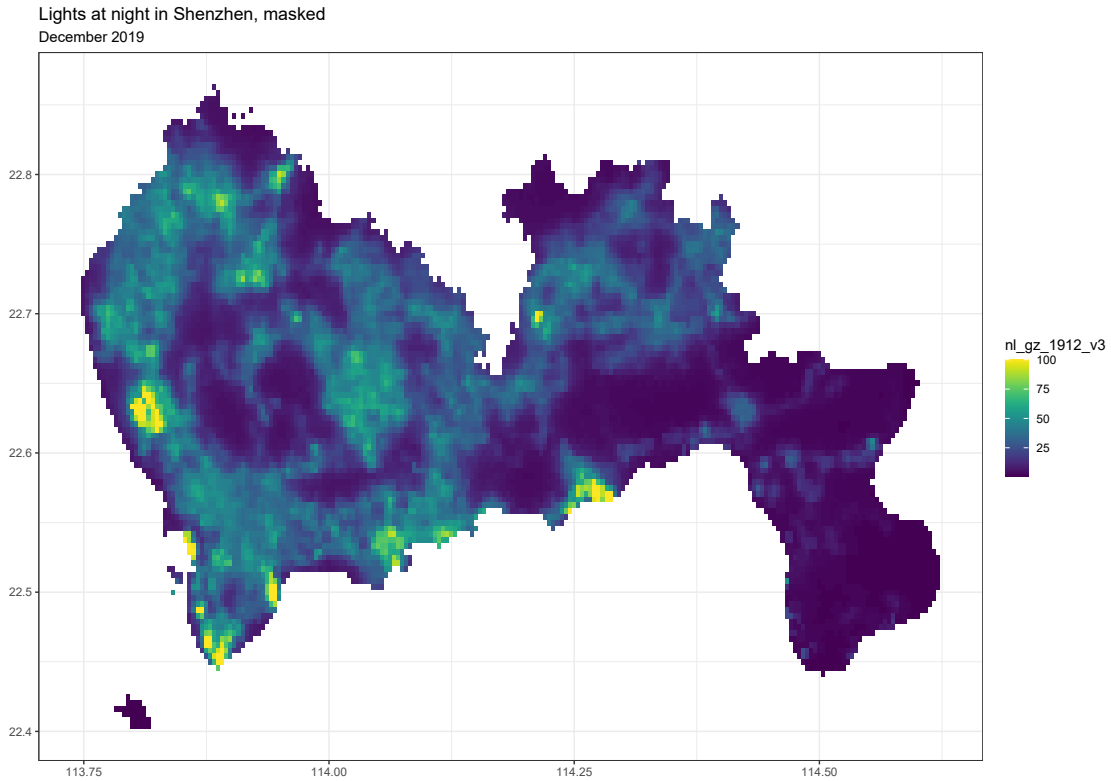
Masks

- 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 the 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
    xiangzhen)
119 gz_block_extract <- raster::extract(
120   x = nl_gz_201912_re,
121   y = gz_t,
122   fun = mean,
123   na.rm = T,
124   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] "褰夸箕鍩\x81" "褰夸箕鍩\x81" "褰夸箕鍩\x81...
  .. ..$ city : chr [1:170] "褰垮窅甯\x82" "褰垮窅甯\x82" "褰垮窅甯\x82" "...
  .. ..$ county : chr [1:170] "澧炲煄鍖\xba" "澧炲煄鍖\xba" "澧炲煄鍖\xba" ...
  .. ..$ town : chr [1:170] "鏌虫哗闀\x87" "鑷板  闀\x87" "灏忚ぜ闀\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 coranavirus cases stayed
  - The data is downloaded from https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_01503669

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 compliated in **sp**

```
126 # 1.7 Summarizing rasters with points
127 # Create a point shapefile from xy coordinate table using SpatialPointsDataFra
    me()
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)
```

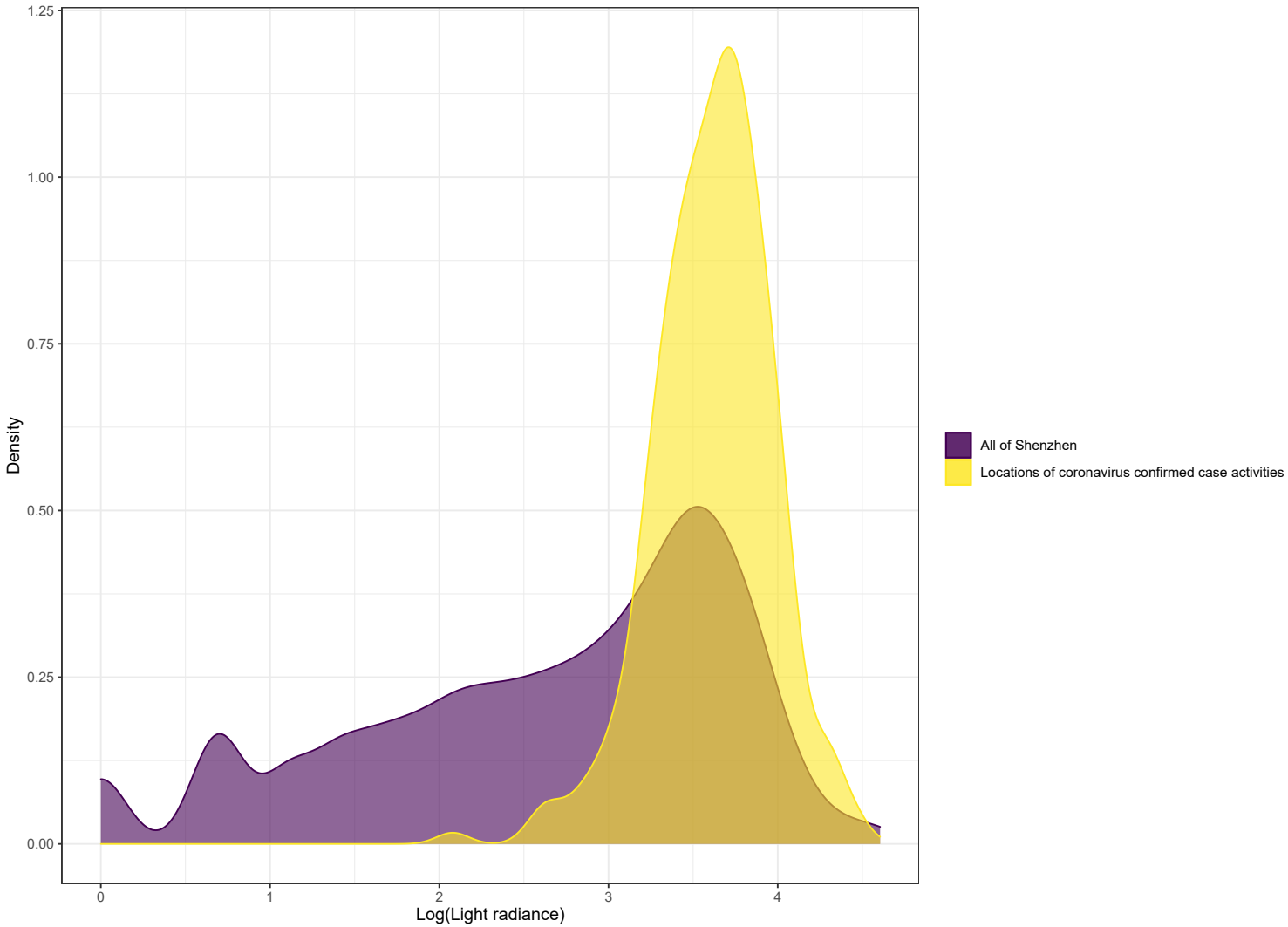
A Simple Application

- 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.

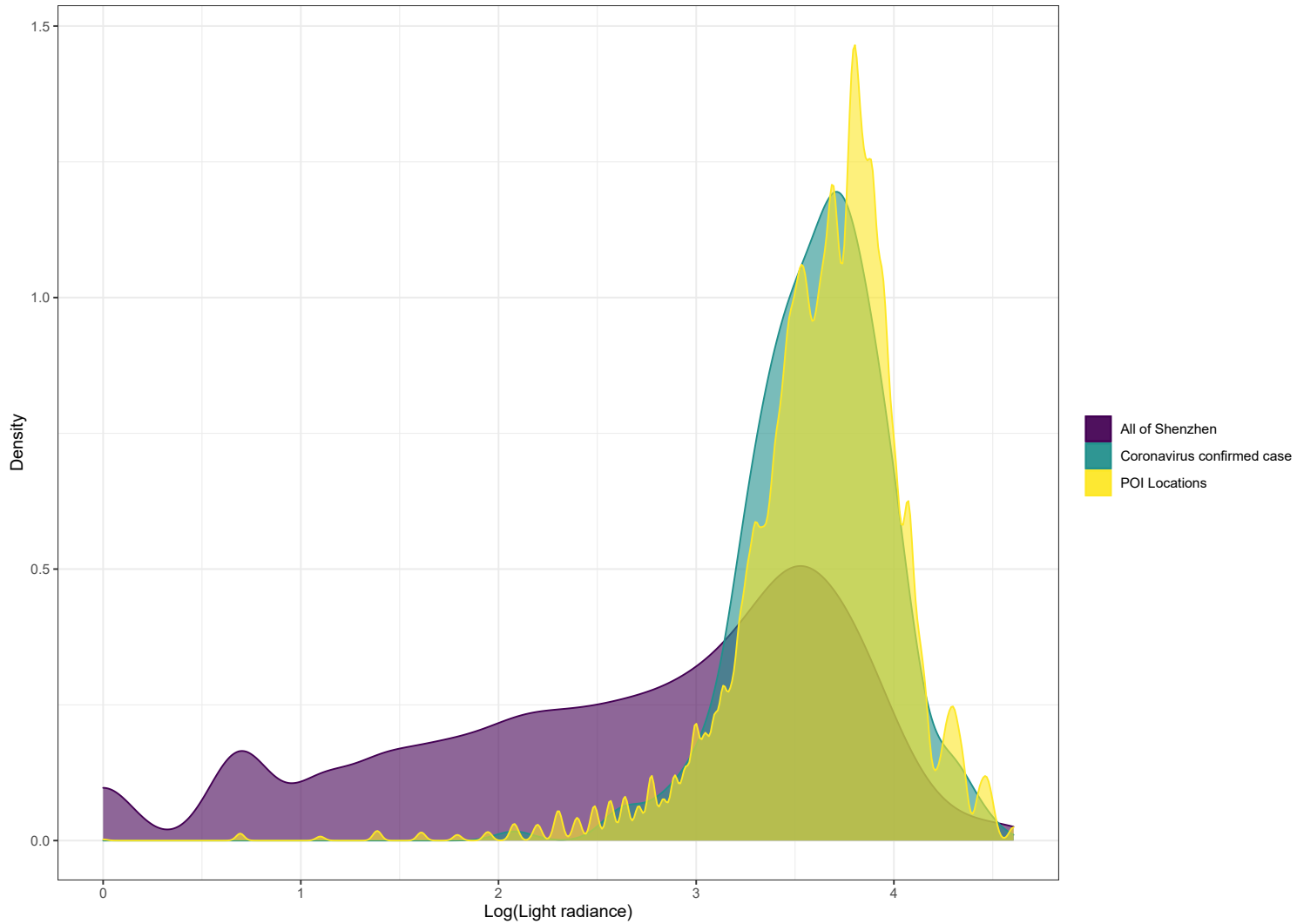
- Sources: wikipedia.cn and https://datarade.ai/data-categories/point-of-interest-poi-data/guide

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()**

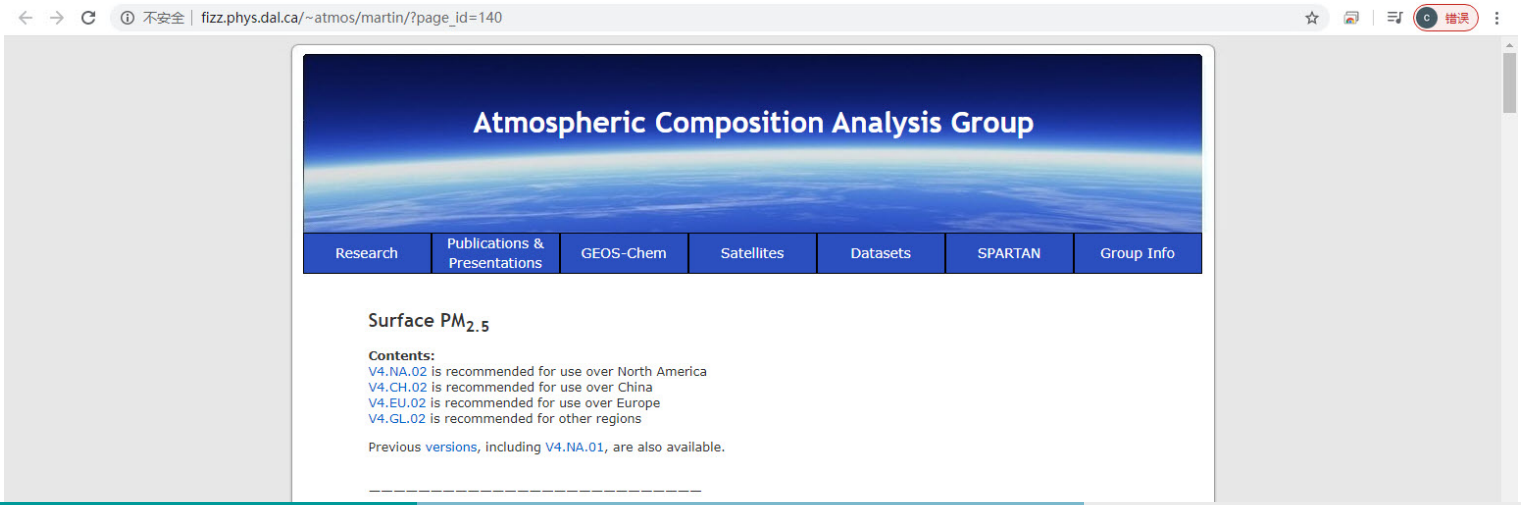
```
## read in the guangdong poi csv file
gd_poi <- read.csv("gd_poi.csv")
str(gd_poi) # we have longitude/latitude information inside
## create a pointshapefile
gd_poi_sp <- SpatialPointsDataFrame(gd_poi[, 1:2],
                                    gd_poi,
                                    proj4string = crs_sz)
## Take the union of the shenzhen polygons
sz_union <- gUnaryUnion(sz_t)
## Check which crime points are inside of Shenzhen
test_points <- over(gd_poi_sp, sz_union)
# From 1 and NA to T and F (F means not in Shenzhen)
test_points <- !is.na(test_points)
sz_poi_sp <- gd_poi_sp[which(test_points == T), ]

sz_poi_extract <- raster::extract(
  x = nl_gz_201912_re,
  y = sz_poi_sp,
  fun = mean,
  na.rm = T,
  sp = T)
```

A Simple Application (Cont'd)



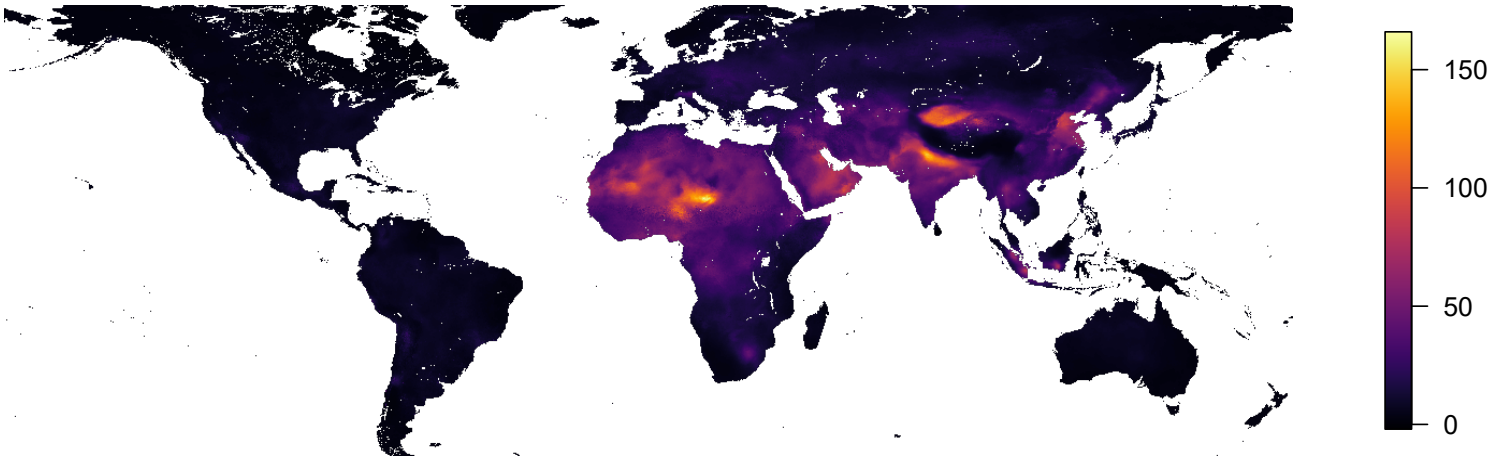
Read in NetCDFs Raster Data

- 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 demostration today, we will use two .nc downloaded from Atomspheric Composition Analysis Group at Dalhousie Univeristy, contain estimates for annual mean PM2.5 levels for 2010 and 2016



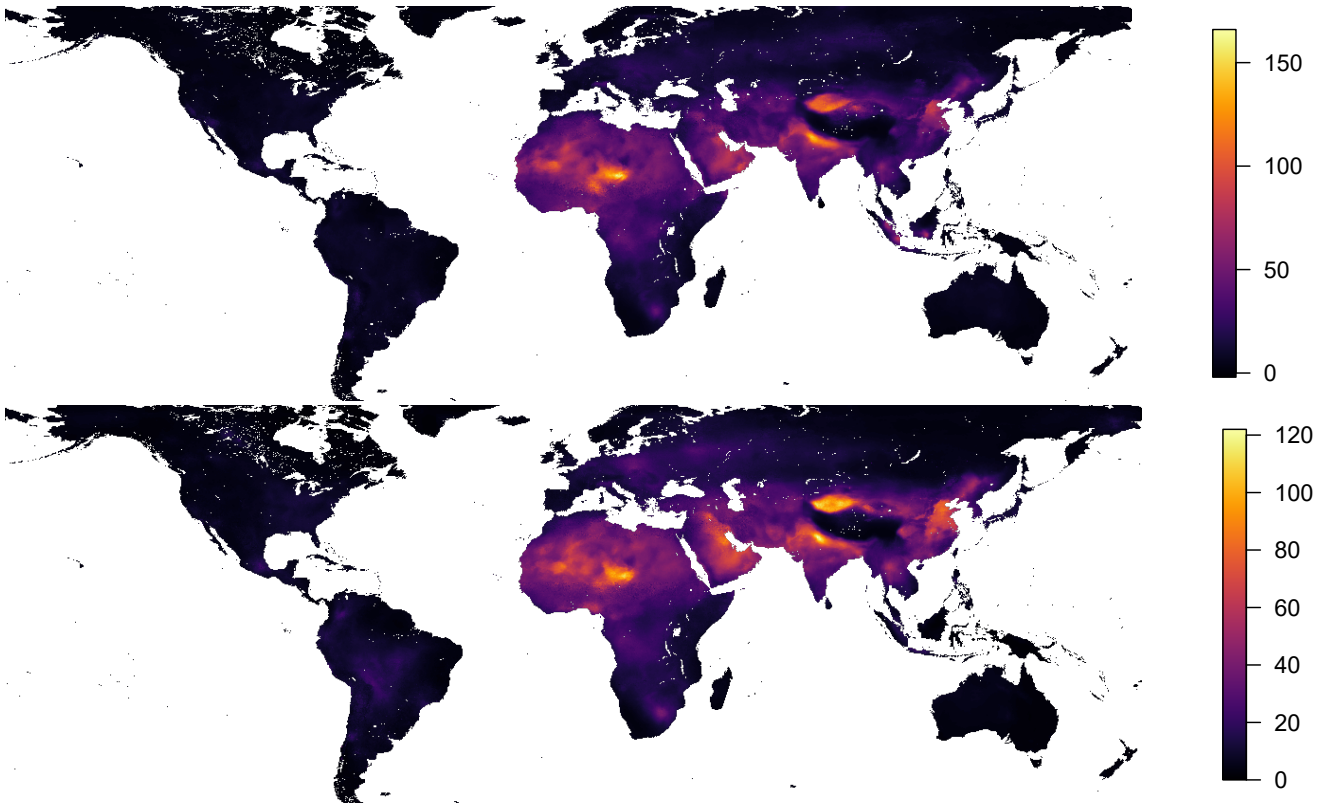
Read in NetCDFs Raster Data (Cont'd)

```
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)
```



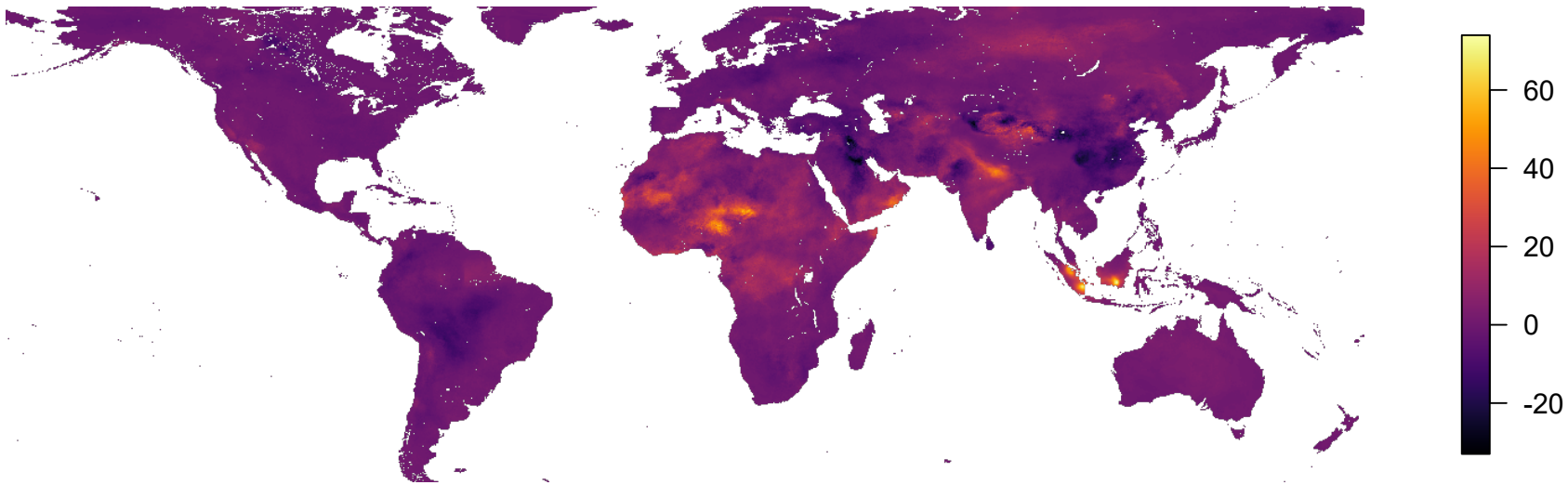
Compare across Maps

- 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)
```



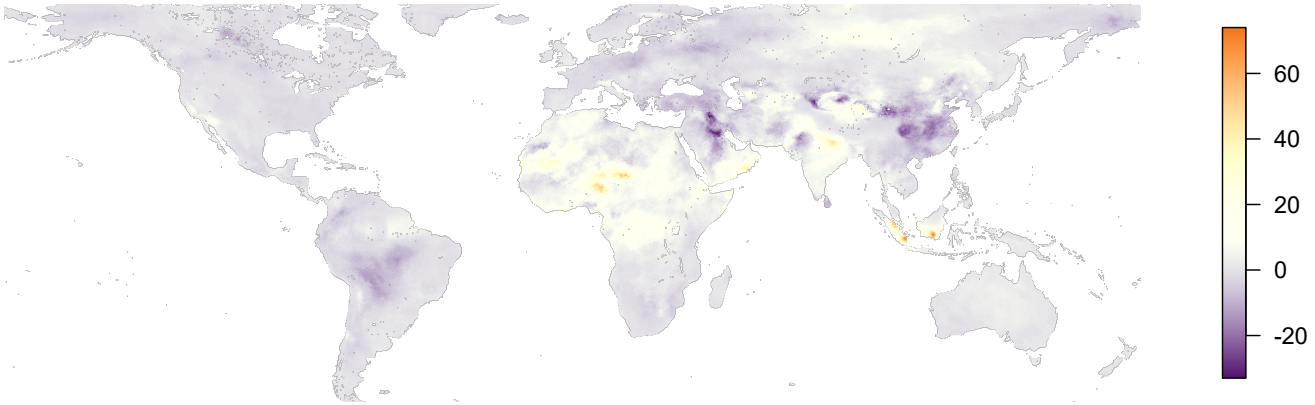
Skill to Emphasize Contrast in Raster Plot

- 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)

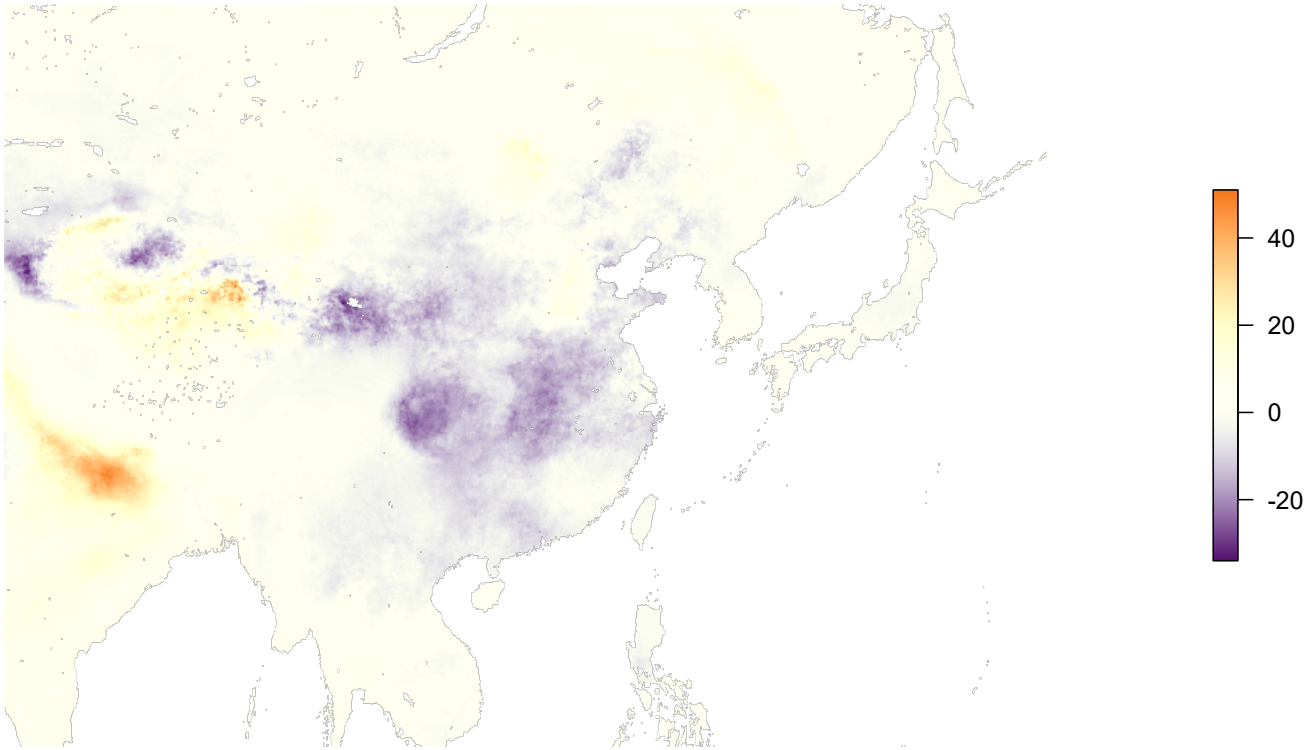
- The color scale is still not super informative: zero is stuck in a arbitrary dark purple region
- We can improve by assigning different colors to the minimum and maximum values and another color scale to other points in between
  - Use the **designer.colors()** function from the **fields** package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

```
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 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 serice)

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	$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-aUByFWfSdEgGGkqIUrZMBL7BU")
303  ## Again, this one is fake, apply and use your own API from https://console
     .cloud.google.com/
```

ECON7930_Lecture6_2020Spring_spa... | gz_block_extract | coords_df | geo_google | JSONtoR.R | sz_cases | ECON7930_Lect

Filter

	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 Servie

- 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("jNKbY2zXrgHowxxxxxxxFgVqseOrK")
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   "成都高梁红餐饮管理有限公司"
319 )
320
321 coords_df <- bmap_get_coords(locs)
322 coords_df
```

ECON7930_Lecture6_2020Spring_spa... * | coords_df | JSONtoR.R

Filter

	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	成都高梁红餐饮管理有限公司	104.0931	30.65378	0	1	80	100	公司企业

Reverse Geocoding

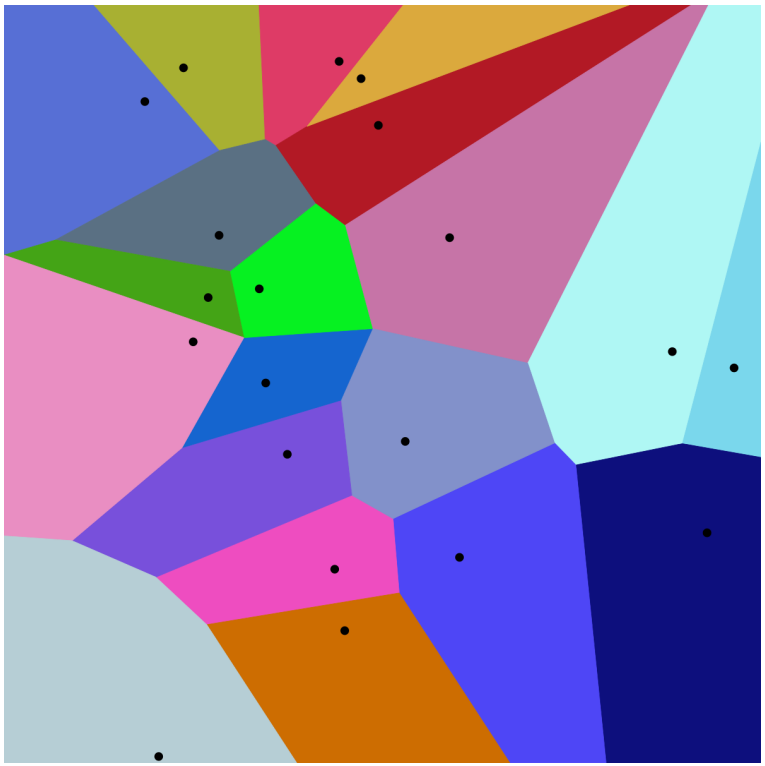
- We can reverse engineer the process to get address from coordinates (so-called reverse geocoding) usign **bmap_get_location()**

```
324 ## Reverse Geocoding
325 addrs_df <- bmap_get_location(coords_df$lat, coords_df$lon)
326 addrs_df
```

```
> addrs_df
  input_lon input_lat return_lon return_lat status              formatted_address                business country
1  106.7977  29.72313   106.7977   29.72313      0                重庆市渝北区福祥路                    龙兴     中国
2  119.8827  30.39242   119.8827   30.39242      0          浙江省杭州市余杭区彭长线                    <NA>     中国
3  104.0931  30.65378   104.0931   30.65378      0 四川省成都市锦江区均隆街1-3号 兰桂坊,东门大桥,东大街     中国
  country_code country_code_iso country_code_iso2 province   city city_level district town ad_code street street_number
1            0              CHN                CN   重庆市 重庆市          2   渝北区 <NA>  500112 福祥路          <NA>
2            0              CHN                CN   浙江省 杭州市          2   余杭区 <NA>  330110 彭长线          <NA>
3            0              CHN                CN   四川省 成都市          2   锦江区 <NA>  510104 均隆街         1-3号
  direction distance sematic_desc city_code
1      <NA>       NA         <NA>       132
2      <NA>       NA         <NA>       179
3      西南       88         <NA>        75
```

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

Geocompuation

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

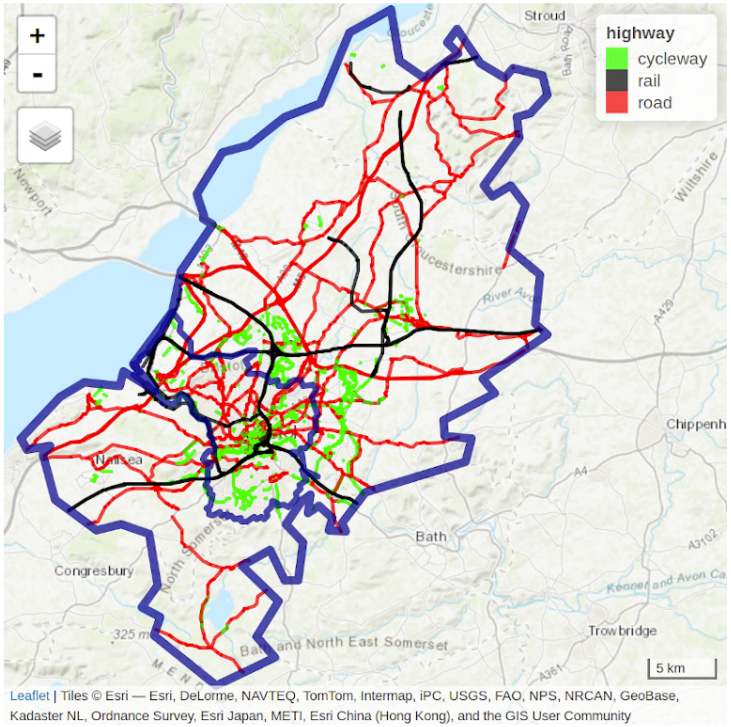


FIGURE 12.1: Bristol's transport network represented by colored lines for active (green), public (railways, black) and private motor (red) modes of travel. Blue border lines represent the inner city boundary and the larger Travel To Work Area (TTWA).

Geocompuation

where to set up new infrastructure for cycling?

- replace frequent short distance car trips
  - desire lines in figure below representing trip patterns in Bristol
  - we only want route desire lines along which a high (300+) number of car trips take place that are up to 5 km in distance

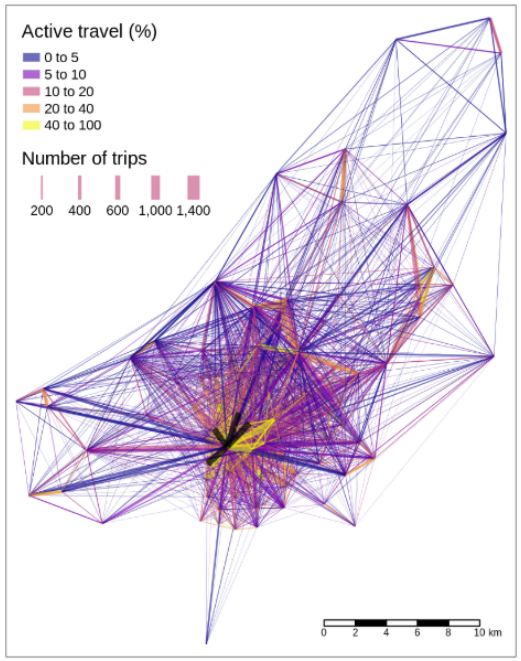


FIGURE 12.3: Desire lines representing trip patterns in Bristol, with width representing number of trips and color representing the percentage of trips made by active modes (walking and cycling). The four black lines represent the interzonal OD pairs in Table 7.1.

Geocompuation

where to set up new infrastructure for cycling?

- link home and train station
  - station nodes (red dots) used as intermediary points that convert straight desire

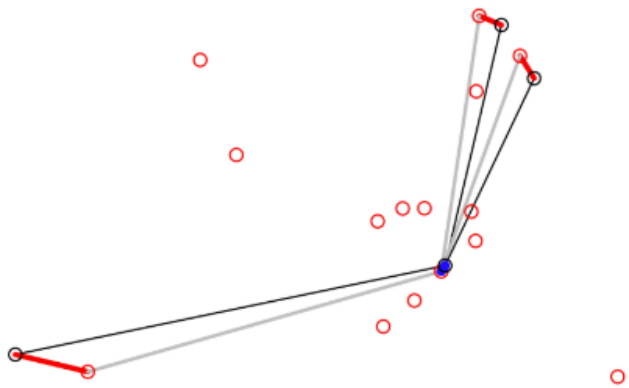


FIGURE 12.4: Station nodes (red dots) used as intermediary points that convert straight desire lines with high rail usage (black) into three legs: to the origin station (red) via public transport (gray) and to the destination (a very short blue line).

Geocompuation

where to set up new infrastructure for cycling?

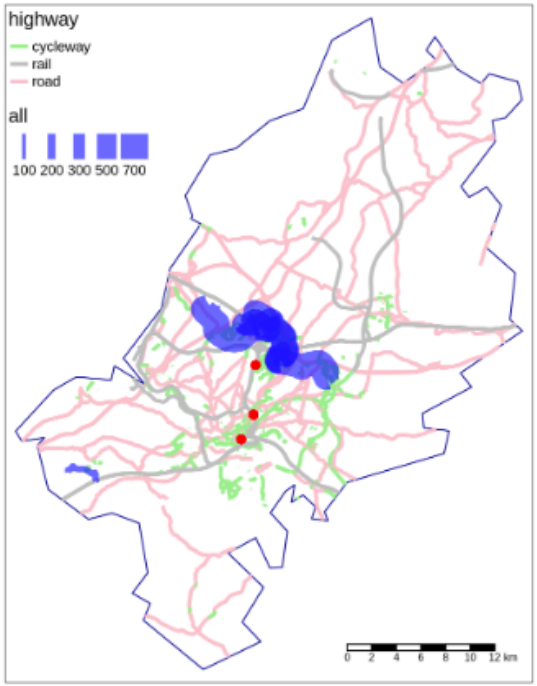


FIGURE 12.6: Potential routes along which to prioritise cycle infrastructure in Bristol, based on access key rail stations (red dots) and routes with many short car journeys (north of Bristol surrounding Stoke Bradley). Line thickness is proportional to number of trips.