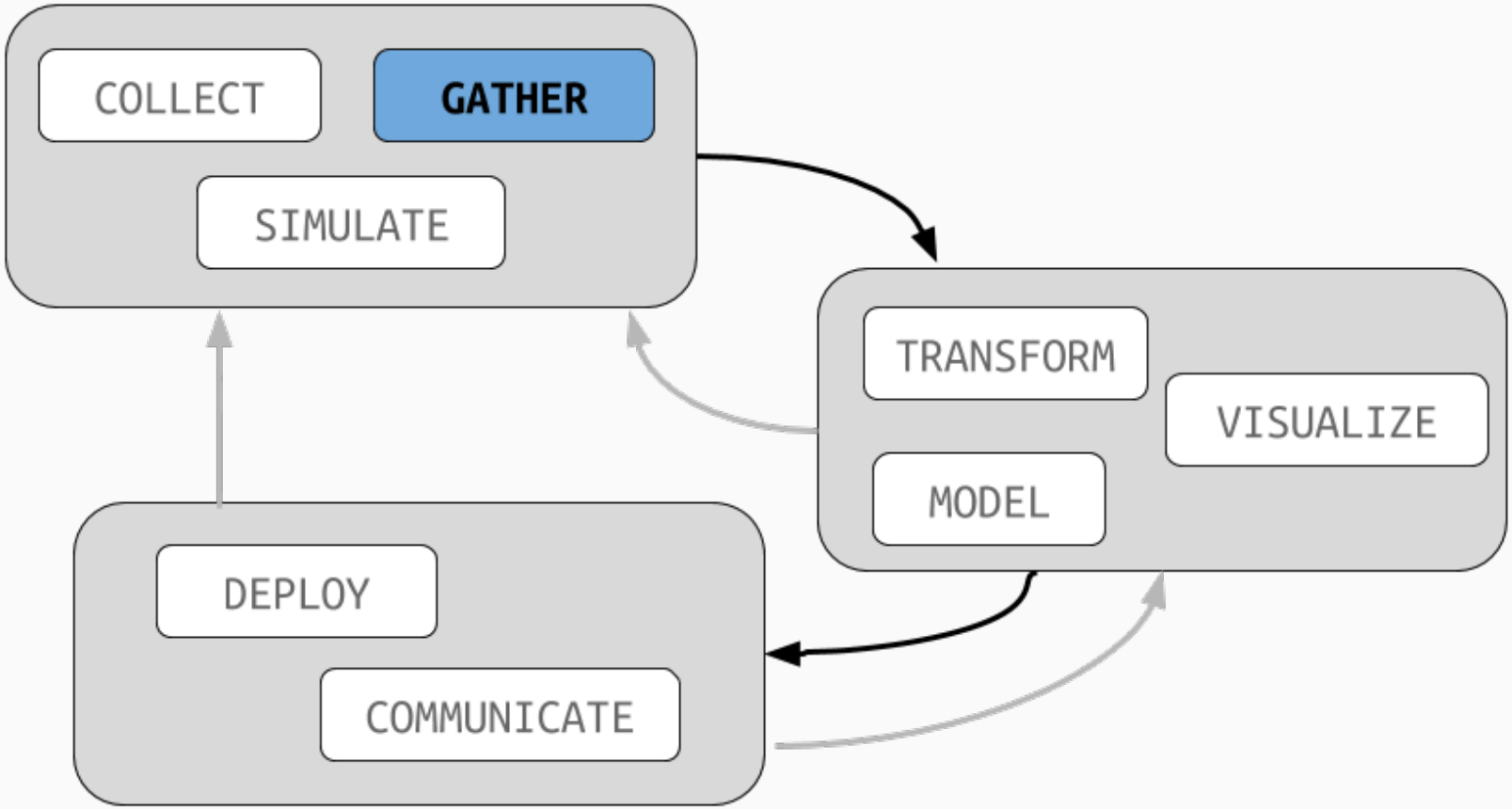
# Lecture 19: Scraping Wikipedia Lists

Taylor Arnold


COLLECT
GATHER
SIMULATE
TRANSFORM
VISUALIZE
MODEL
DEPLOY
COMMUNICATE

# web scraping

Today, we are going to use R to systemically build a dataset from information on Wikipedia.

Words of caution about scraping data (note: this is not legal advice):

- many websites prohibit the automatic downloading of material in their terms and conditions
- a small set of people have gotten in serious trouble for scraping websites against their terms of use
- we are only going to apply this to Wikipedia, as they have generously permissive rules about this sort of thing
- if you ever want to replicate this on a large scale, please use the Wikipedia API rather than the tools here

# largest cities

Wikipedia has curated a table of the most populous cities in the world. We can download this list, as in the last class, using the following code:

```
url <- "https://en.wikipedia.org/wiki/List_of_largest_cities"
wpage <- data_frame(line = readLines(url))
```

Make sure to view the site itself as well in a browser.

# extracting cities

Searching through the source HTML, we see that rows in the city table always start with the string `<th scope="row">`. Let's filter our dataset to only include these rows:

```
wpage <- filter(wpage, stri_detect(line, fixed = '<th scope="row">'))
```

## extracting city names

We will also remove all HTML tags on these rows, and store the results as the variable city.

```
wpage <- mutate(wpage, city = stri_replace_all(line, "", regex = "<[^>]
wpage$city
```

```
##   [1] "Shanghai"                "Karachi"
##   [3] "Beijing"                 "Delhi"
##   [5] "Lagos"                   "Dhaka"
##   [7] "Guangzhou"               "Istanbul"
##   [9] "Tokyo"                   "Mumbai"
##  [11] "Moscow"                  "São Paulo"
##  [13] "Shenzhen"                "Suzhou"
##  [15] "Lahore"                  "Cairo"
##  [17] "Kinshasa"                "Jakarta"
##  [19] "Seoul"                   "Mexico City"
##  [21] "Lima"                    "London"
##  [23] "Xi'an"                   "New York City"
##  [25] "Bengaluru"               "Bangkok"
##  [27] "Nanjing"                 "Dongguan"
```

# extracting links

Finally, looking at row of data, notice that there is a link to the Wikipedia page about each city:

```
wpage$line[1]
```

```
## [1] "<th scope=\"row\"><a href=\"/wiki/Shanghai\" title=\"Shanghai\">S
```

# extracting links

We can extract these using the `stri_extract` function:

```
wpage <- mutate(wpage, link = stri_extract(line, regex = "/wiki/[^[\"]]
wpage$link
```

```
##   [1] "/wiki/Shanghai"
##   [2] "/wiki/Karachi"
##   [3] "/wiki/Beijing"
##   [4] "/wiki/Delhi"
##   [5] "/wiki/Lagos"
##   [6] "/wiki/Dhaka"
##   [7] "/wiki/Guangzhou"
##   [8] "/wiki/Istanbul"
##   [9] "/wiki/Tokyo"
##  [10] "/wiki/Mumbai"
##  [11] "/wiki/Moscow"
##  [12] "/wiki/S%C3%A3o_Paulo"
##  [13] "/wiki/Shenzhen"
##  [14] "/wiki/Suzhou"
##  [15] "/wiki/Lahore"
```

## a city page

With this link, we can now download the specific information from a given city. For instance, let's take the 5'th city (Lagos) and download the website from Wikipedia:

```
i <- 5
url <- stri_c("https://en.wikipedia.org", wpage$link[i])
cpage <- data_frame(line = readLines(url))
```

Note the use of the `stri_c` function to combine the base URL with the link.

# city coordinates

One piece of information on most city pages is the latitude and longitude of the city. We can find this by detecting the string "Coordinates:" in the webpage. Here, we save only the first occurrence of the string (here there is only one, but we want to safely abstract this to other pages):

```
coord <- filter(cpage, stri_detect(line,
                           fixed = "Coordinates:"))$line[1]
coord
```

```
## [1] "<td colspan=\"2\" style=\"text-align:center\">Coordinates: <span
```

# city coordinates

Looking at the output, we want to not just remove the HTML tags but to capture a very specific part of the tags. Here we get the easiest form of the coordinates from the "geo" tag:

```
coord <- stri_extract(coord,
                      regex = "<span class=\"geo\">[^<]+</span>")
coord
```

```
## [1] "<span class=\"geo\">6.455027; 3.384082</span>"
```

# city coordinates

We want to now remove the html tags. We could do this by a regular expression, but let's instead do it by taking a substring:

```
coord <- stri_sub(coord, 19, -8)
coord
```

```
## [1] "6.455027; 3.384082"
```

# city coordinates

Next, we want to split the string into two parts based on the ";" symbol. We do this with the `stri_split` function.

```
coord <- stri_split(coord, fixed = ";")[[1]]
coord
```

```
## [1] "6.455027"  " 3.384082"
```

# city coordinates

Finally, we want to convert these strings into numbers. Recall that `factor` turns numeric data into categorical ones; at the time, I mentioned that rarely can we go in the reverse order. Here we use the `as.numeric` function to do just that:

```
coord <- as.numeric(coord)
lat <- coord[1]
lon <- coord[2]
lat
```

```
## [1] 6.455027
```

```
lon
```

```
## [1] 3.384082
```

# country name

Let's also find the name of the country that each city is in. this is a bit complicated because the row that shows the tag "Country" is actually one away from the row that gives the data:

```
filter(cpage, stri_detect(line,
                fixed = "<th scope=\"row\">Country</th>"))
```

```
## # A tibble: 1 x 1
##                                  line
##                                 <chr>
## 1 "<th scope=\"row\">Country</th>"
```

## country name

To fix this, wrap the `stri_detect` function in the function `which`. This gives the indices where something is True:

```
id <- which(stri_detect(cpage$line,
                fixed = "<th scope=\"row\">Country</th>"))
id
```

```
## [1] 104
```

# country name

Now, we just add 1 to this index (extracting just the first one, in case there are multiple) and grab those lines:

```
country <- cpage$line[id[1] + 1]
country
```

```
## [1] "<td><span class=\"flagicon\"><img alt=\"\" src=\"//upload.wikimed
```

# country name

Cleaning up the results yields the desired information:

```
country <- cpage$line[which(stri_detect(cpage$line,
            fixed = "<th scope=\"row\">Country</th>"))[1] + 1]
country <- stri_replace_all(country, "", regex = "<[^>]+>")
country <- stri_replace_all(country, "", fixed = " ")
country
```

```
## [1] "Nigeria"
```

# automating over cities

We now have code that works on a single city page. Let's add empty attributes to the `wpage` dataset. We'll fill these in in a moment.

```
wpage$lat <- NA
wpage$lon <- NA
wpage$country <- NA
```

# automating over cities

We now use a *for loop* to cycle over all rows of the wpage dataset as follows:

```
for (i in 1:nrow(wpage)) {
  url <- stri_c("https://en.wikipedia.org", wpage$link[i])
  cpage <- data_frame(line = readLines(url))

  # extract stuff

  wpage$lat[i] <- lat
  wpage$lon[i] <- lon
  wpage$country[i] <- country
}
```

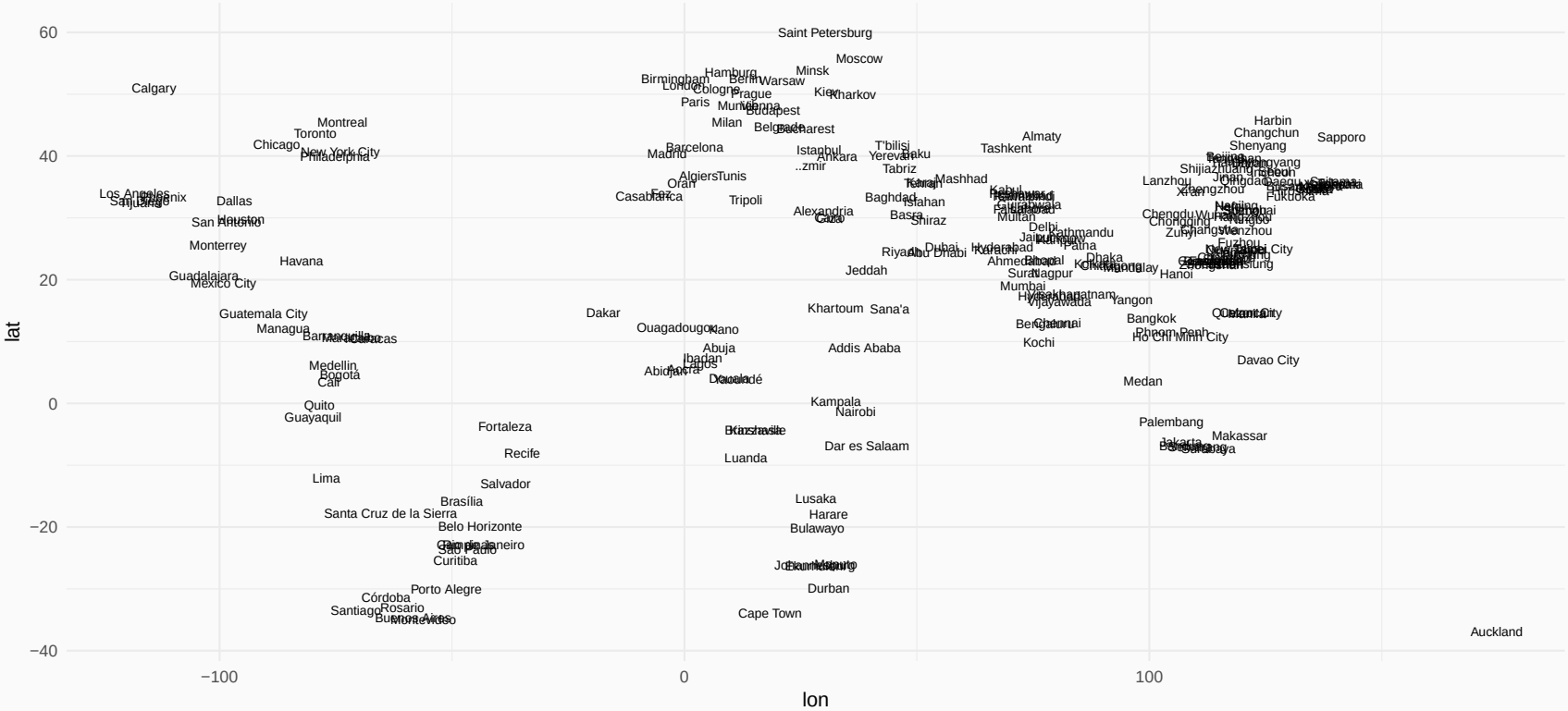
## automating over cities

```
for (i in 1:nrow(wpage)) {
  url <- stri_c("https://en.wikipedia.org", wpage$link[i])
  cpage <- data_frame(line = readLines(url))

  coord <- filter(cpage,
      stri_detect(line, fixed = "Coordinates:"))$line[1]
  coord <- stri_extract(coord,
      regex = "<span class=\"geo\">[^<]+</span>")
  coord <- stri_sub(coord, 19, -8)
  coord <- stri_split(coord, fixed = ";")[[1]]
  coord <- as.numeric(coord)
  wpage$lat[i] <- coord[1]
  wpage$lon[i] <- coord[2]

  country <- cpage$line[which(stri_detect(cpage$line,
      fixed = "<th scope=\"row\">Country</th>"))[1] + 1]
  country <- stri_replace_all(country, "", regex = "<[^>]+>")
  country <- stri_replace_all(country, "", fixed = " ")
```

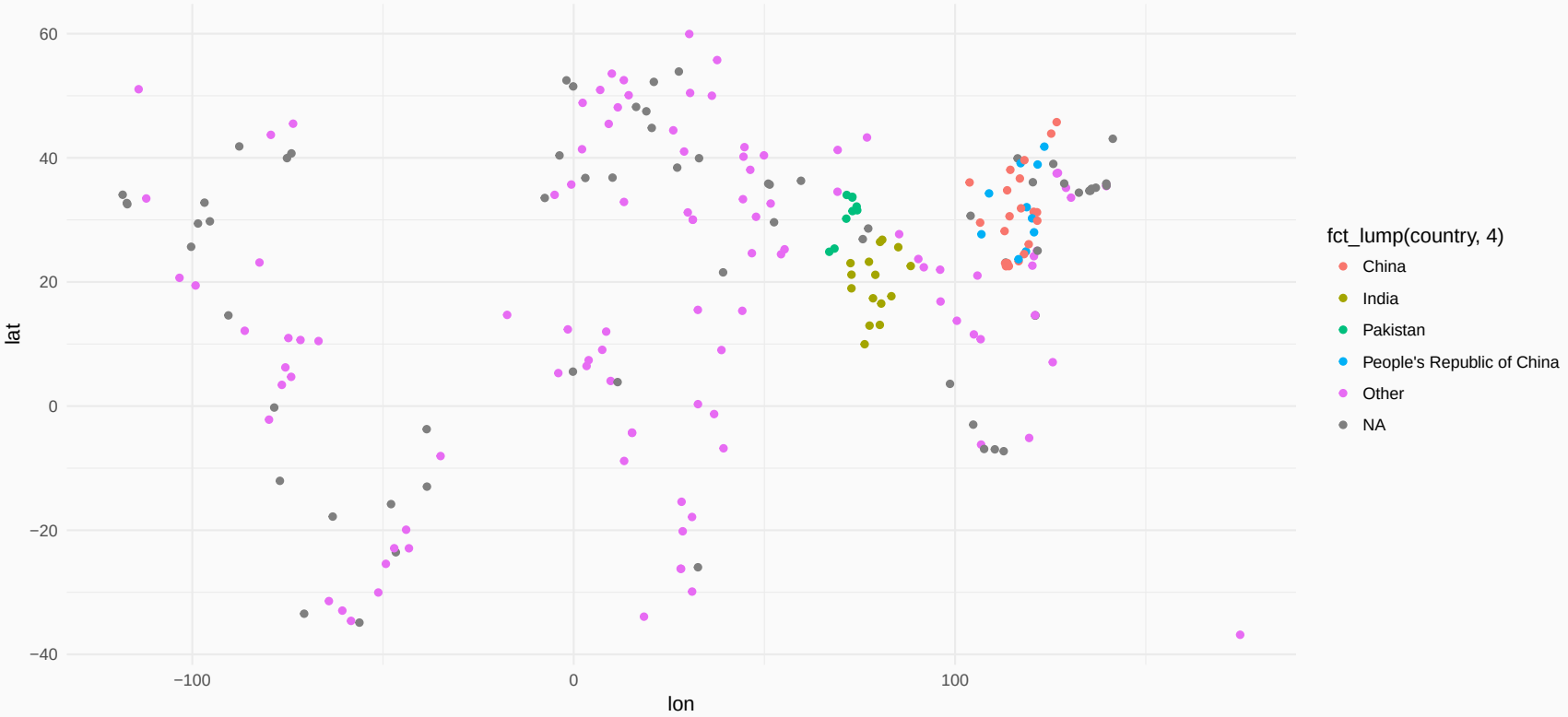
We can now plot the data as follows:

```
qplot(lon, lat, data = wpage, label = city, geom = "text") +
  theme_minimal()
```

# results


lat
lon
60
40
20
0
−20
−40
−100
0
100
Saint Petersburg
Moscow
Minsk
Kharkov
Calgary
Montreal
Toronto
Chicago
New York City
Philadelphia
Los Angeles
Dallas
Houston
San Antonio
Monterrey
Havana
Guadalajara
Mexico City
Guatemala City
Managua
Medellin
Bogotá
Cali
Quito
Guayaquil
Lima
Fortaleza
Recife
Salvador
Brasília
Santa Cruz de la Sierra
Belo Horizonte
Curitiba
Porto Alegre
Córdoba
Santiago
Rosario
Birmingham
Hamburg
Berlin
Warsaw
London
Cologne
Prague
Kiev
Paris
Vienna
Budapest
Milan
Belgrade
Bucharest
Barcelona
Madrid
Istanbul
Ankara
Yerevan
Baku
T'bilisi
Tashkent
Almaty
Tabriz
Tehran
Mashhad
Isfahan
Baghdad
Basra
Shiraz
Tripoli
Algiers
Tunis
Oran
Casablanca
Fez
Alexandria
Cairo
Riyadh
Dubai
Abu Dhabi
Jeddah
Dakar
Ouagadougou
Kano
Abuja
Ibadan
Lagos
Abidjan
Accra
Khartoum
Sana'a
Addis Ababa
Kampala
Nairobi
Dar es Salaam
Luanda
Lusaka
Harare
Bulawayo
Durban
Cape Town
Harbin
Changchun
Sapporo
Shenyang
Lanzhou
Xi'an
Zhengzhou
Fukuoka
Chengdu
Chongqing
Zunyi
Delhi
Kathmandu
Patna
Dhaka
Surat
Nagpur
Mumbai
Hanoi
Yangon
Bangkok
Phnom Penh
Ho Chi Minh City
Kochi
Medan
Palembang
Jakarta
Makassar
Davao City
Auckland

# results


60
40
20
0
−20
−40
lat
−100
0
100
lon
fct_lump(country, 4)
China
India
Pakistan
People's Republic of China
Other
NA

Or tabulate by country:

```
count(wpage, country, sort = TRUE)
```

```
## # A tibble: 69 x 2
##                       country     n
##                         <chr> <int>
## 1                        <NA>    69
## 2                       China    21
## 3                       India    16
## 4 People's Republic of China    10
## 5                    Pakistan     9
## 6                      Russia     7
## # ... with 63 more rows
```