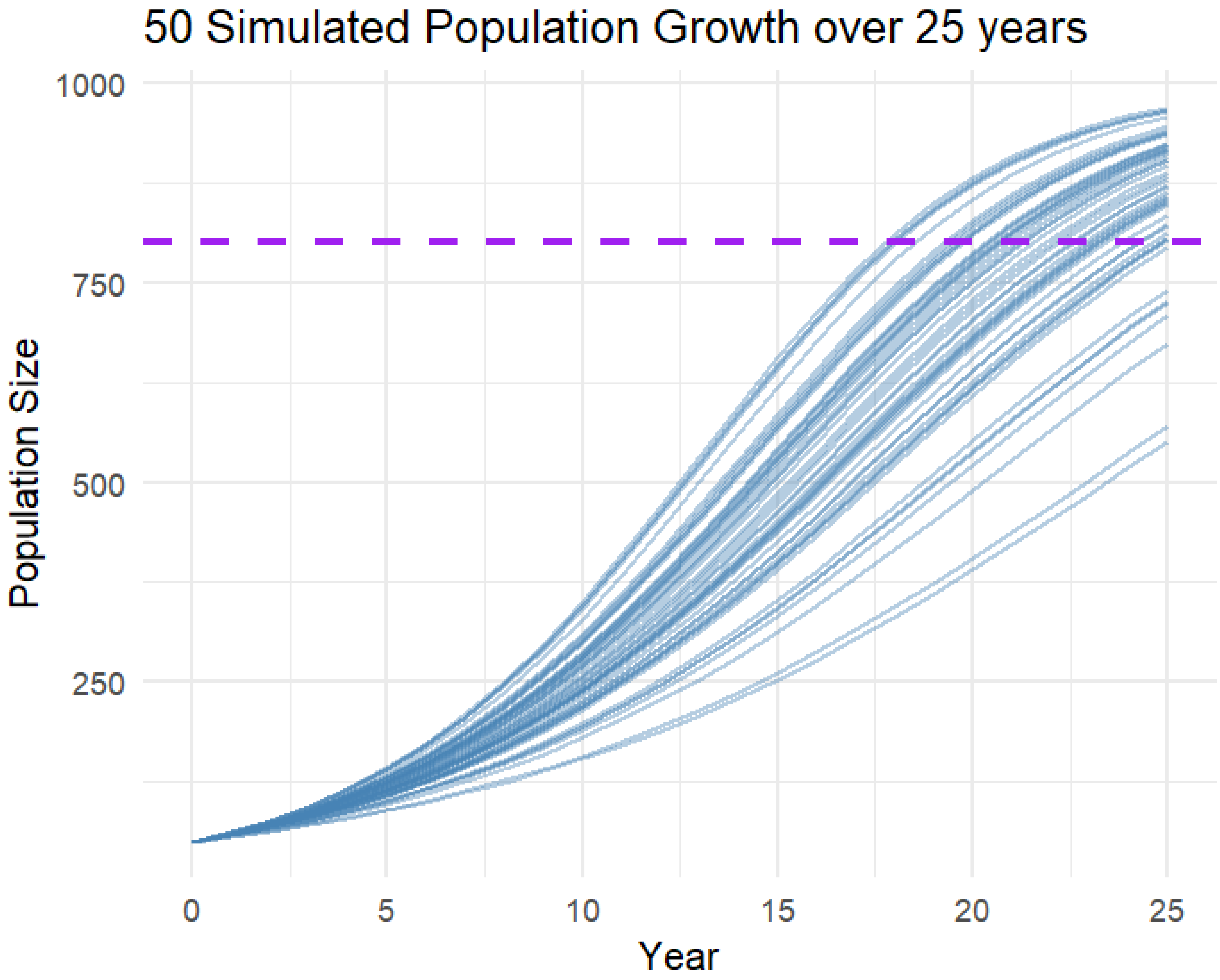
Week 7
Data exploration and visualization II – maps

Today

- Review homework
- Rasters, polygons, and points
- In-class exercise

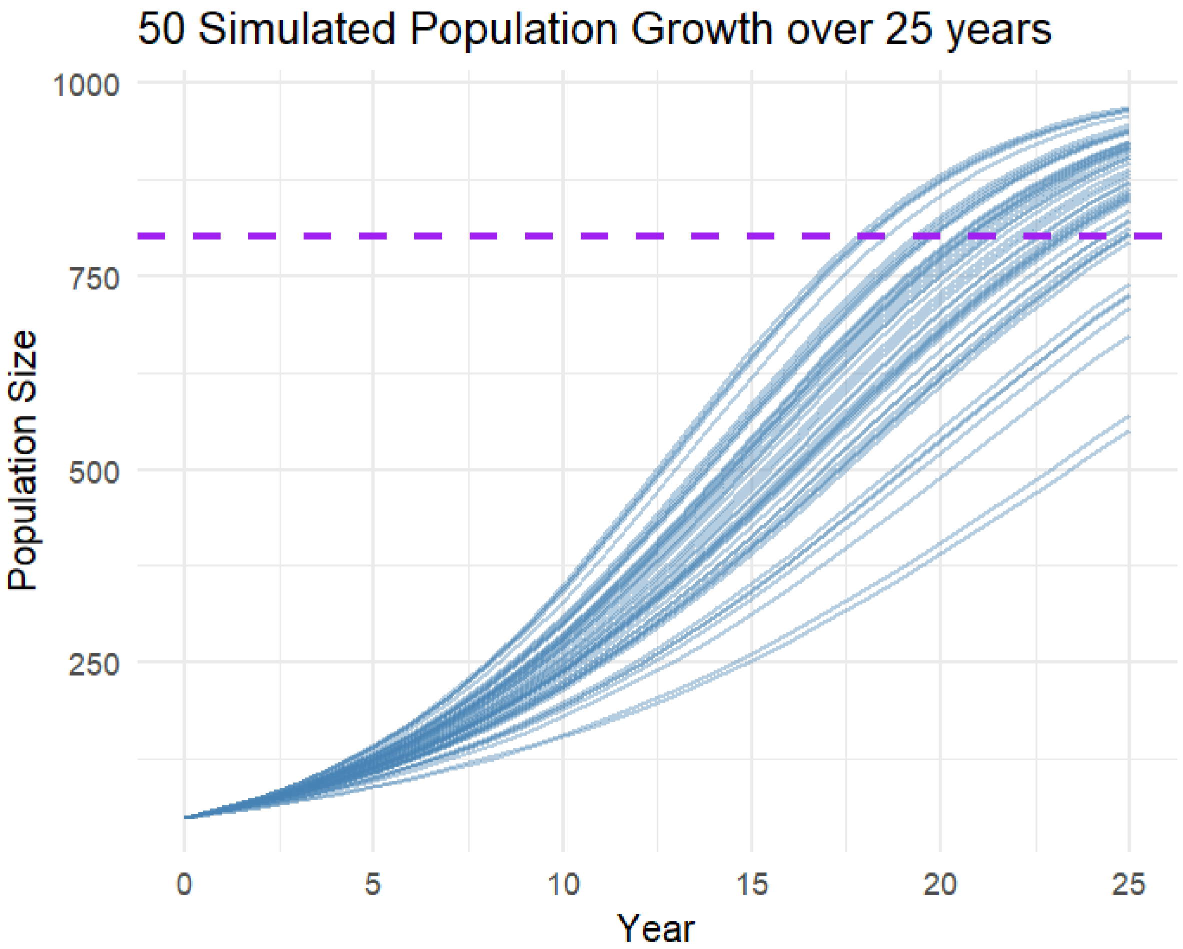
ggplot HW - transparency

```
geom_line(alpha = 0.4, color =
"steelblue")
```



ggplot HW – reference lines

```
geom_hline(yintercept = target
, linetype = "dashed", color =
"purple", size = 1)
```



Hat tip: Samantha!

ggplot HW – wide to long format

Wide format

Sim1	Sim2	Sim3	…	Sim50
50	50	50		50

Long format

Year	Simulation_number	Population_size
1	1	50
2	1	63
…		
25	1	857
1	2	50

Hat tip: Lonnie!

ggplot HW – wide to long format

```
sim_df <- as.data.frame(sim_mat)
sim_df$year <- 0:50
sim_df_long <- pivot_longer(sim_df, cols = sim_df$year,
names_to = "simnum", values_to = "pop_size")
```

Wide format

Sim1	Sim2	Sim3	…	Sim50
50	50	50		50

Long format

Year	Simulation_number	Population_size
1	1	50
2	1	63
…		
25	1	857
1	2	50

Hat tip: Lonnie!

ggplot HW – generating random numbers from a distribution

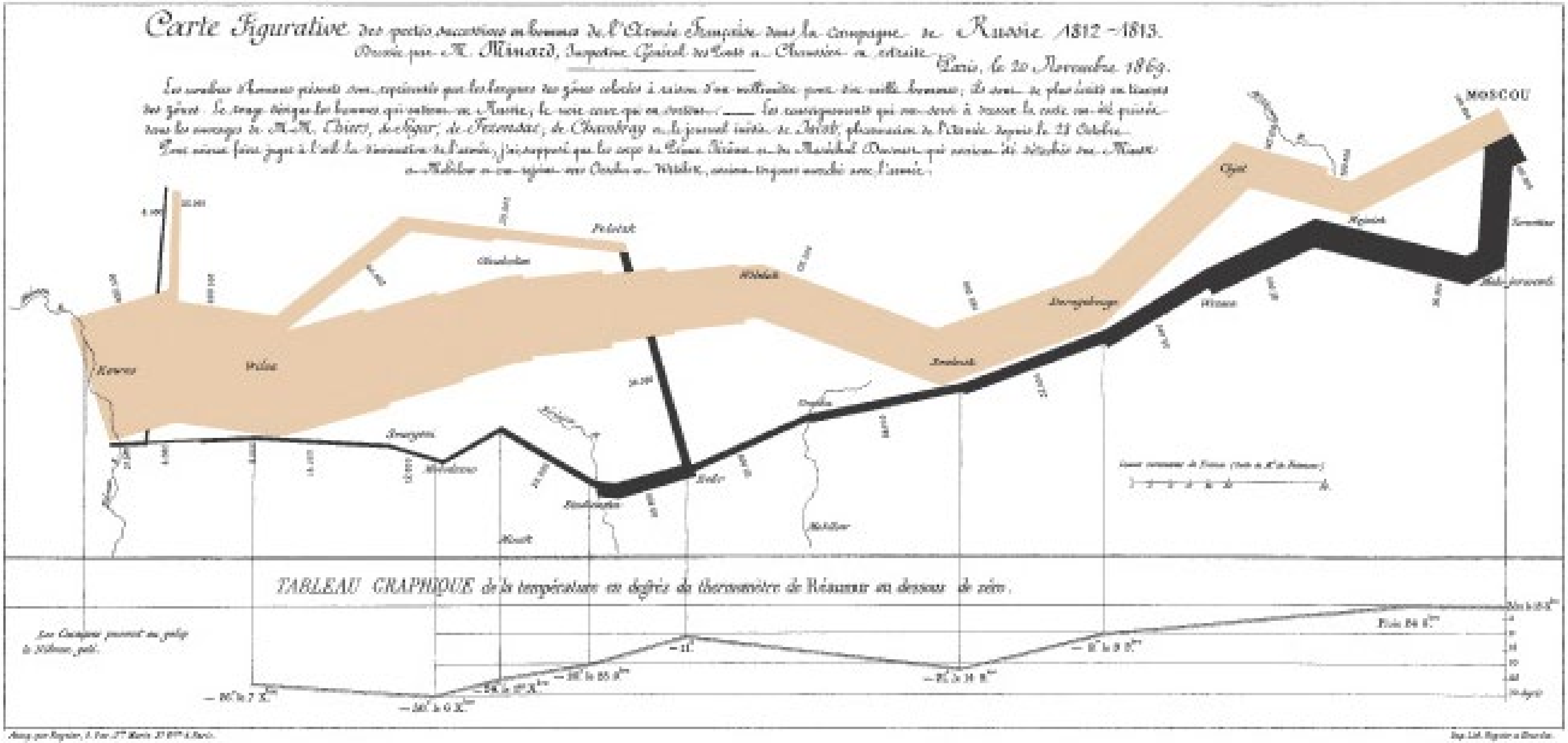
```
#use rnorm to make all the random values to use for r
r_vals = rnorm(50, mean = 0.2, sd = 0.03)
```

ggplot HW – vectorized operations (no need for nested for-loop)

```
#Create an empty matrix to store generated values for each random r
N_random <- matrix(NA, nrow = years + 1, ncol = length(r_random))

#Set initial population size for each random r (all start at 50)
#It tells row 1 in N_random matrix to equal N0 which was defined as 50
N_random[1, ] <- N0

#Create for loop that gives time series for each generated r value
for (t in 1:years) {
N_random[t + 1, ] <- N_random[t, ] + r_random * N_random[t, ] * (1 - N_random[t, ] /
K)}
```

Hat tip: Maddy!

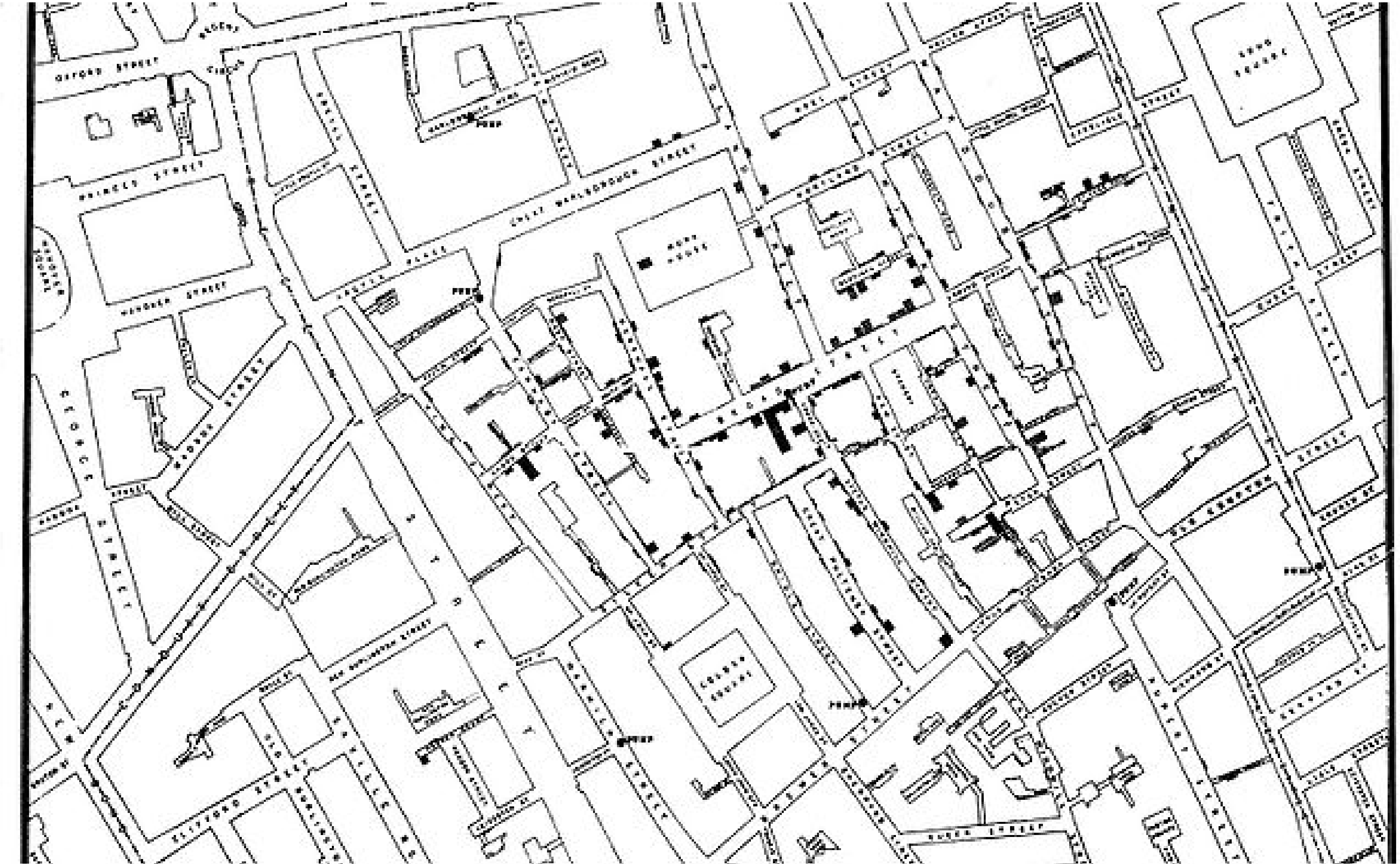
Mapping and (visual) spatial analysis


Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812–1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.
Paris, le 20 Novembre 1869.
MOSCOU
Kowno
Wilna
Polotzk
Witebsk
Smolensk
Dorogobouge
Chjat
Mojaisk
Tarantino
Malo-jarosewli
Wiasma
Orscha
Bobr
Studienska
Minsk
Mohilow
Molodeczno
Smorgoni
TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.
Les Cosaques passent au galop le Niémen gelé.

Charles Minard: Figures should “speak to the mind”

Tobler's first law of geography:
"Everything is related to everything else, but near things are more related than distant things."

Inferring process from pattern



John Snow’s map of cholera cases in London (1854)

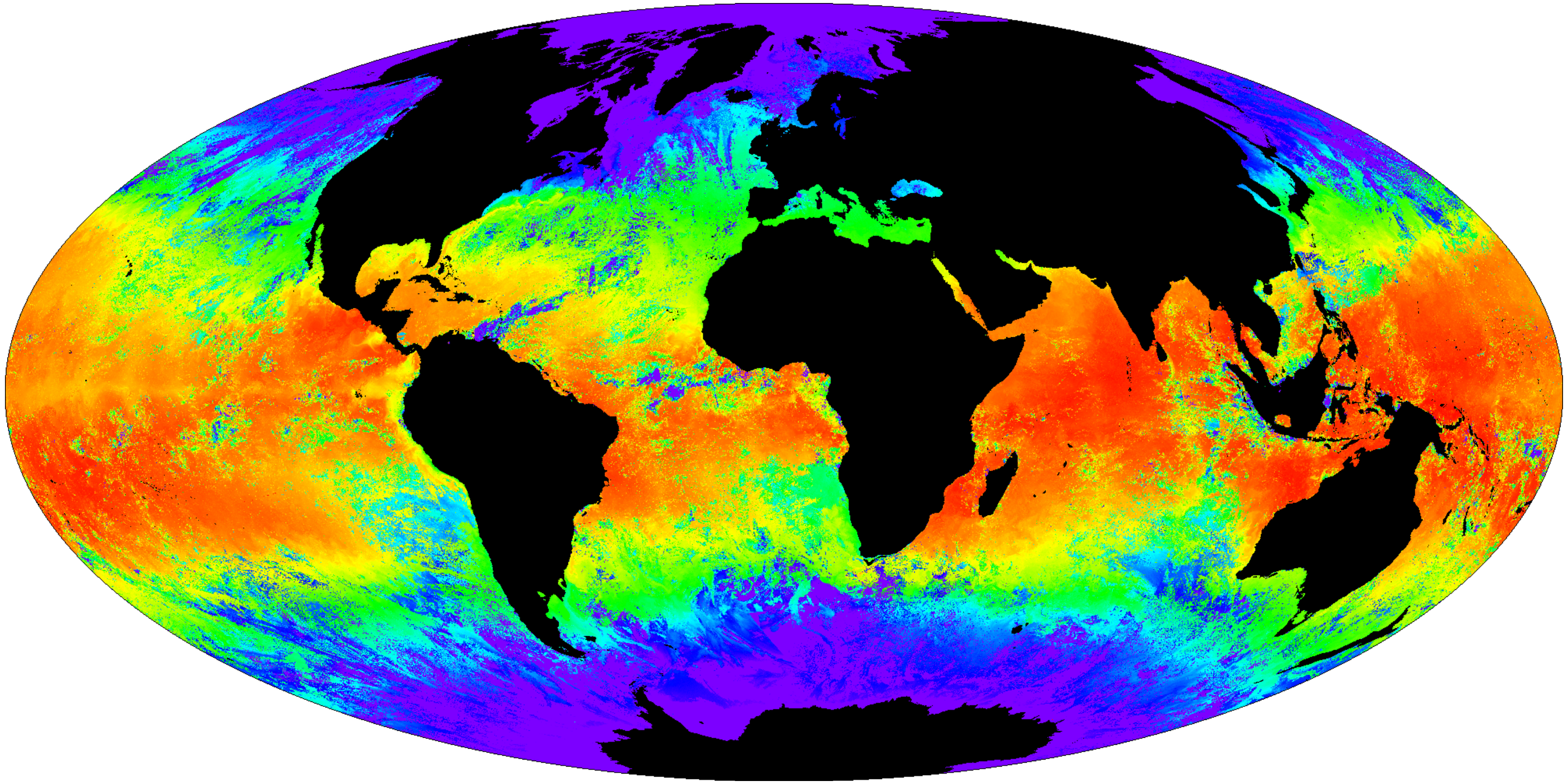
Spatial dependence – exogenous and endogenous processes

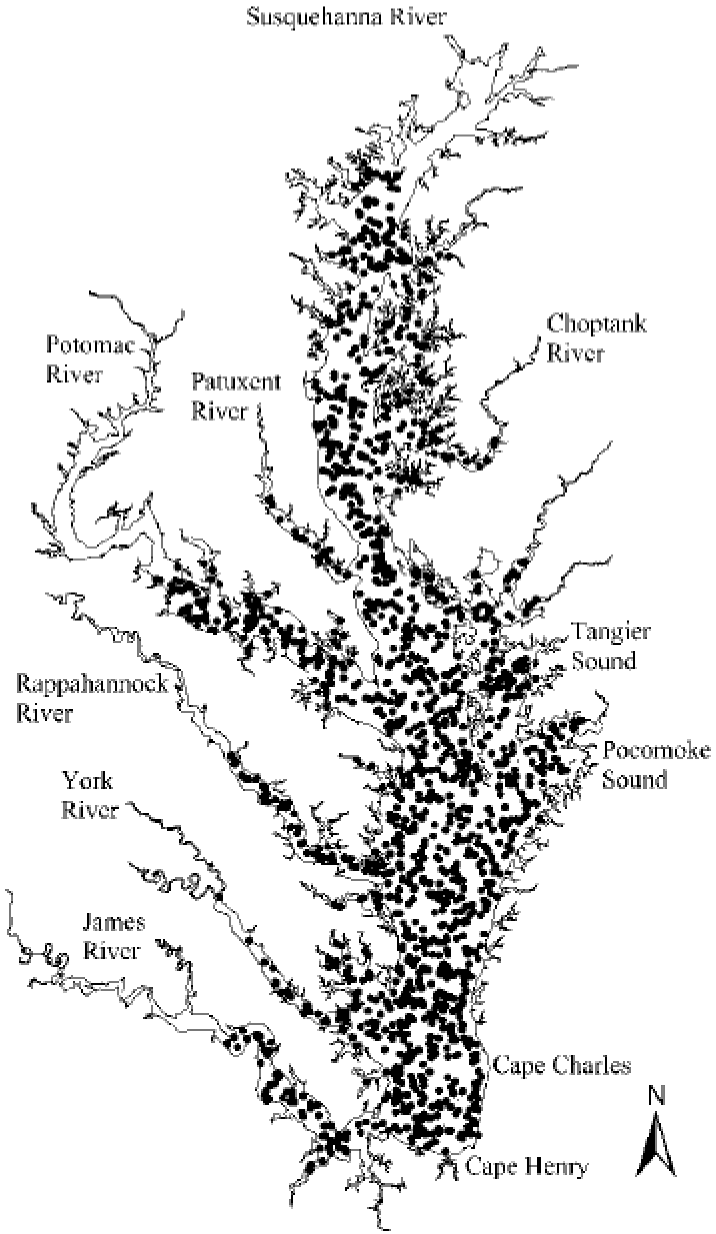
Three classes of spatial data

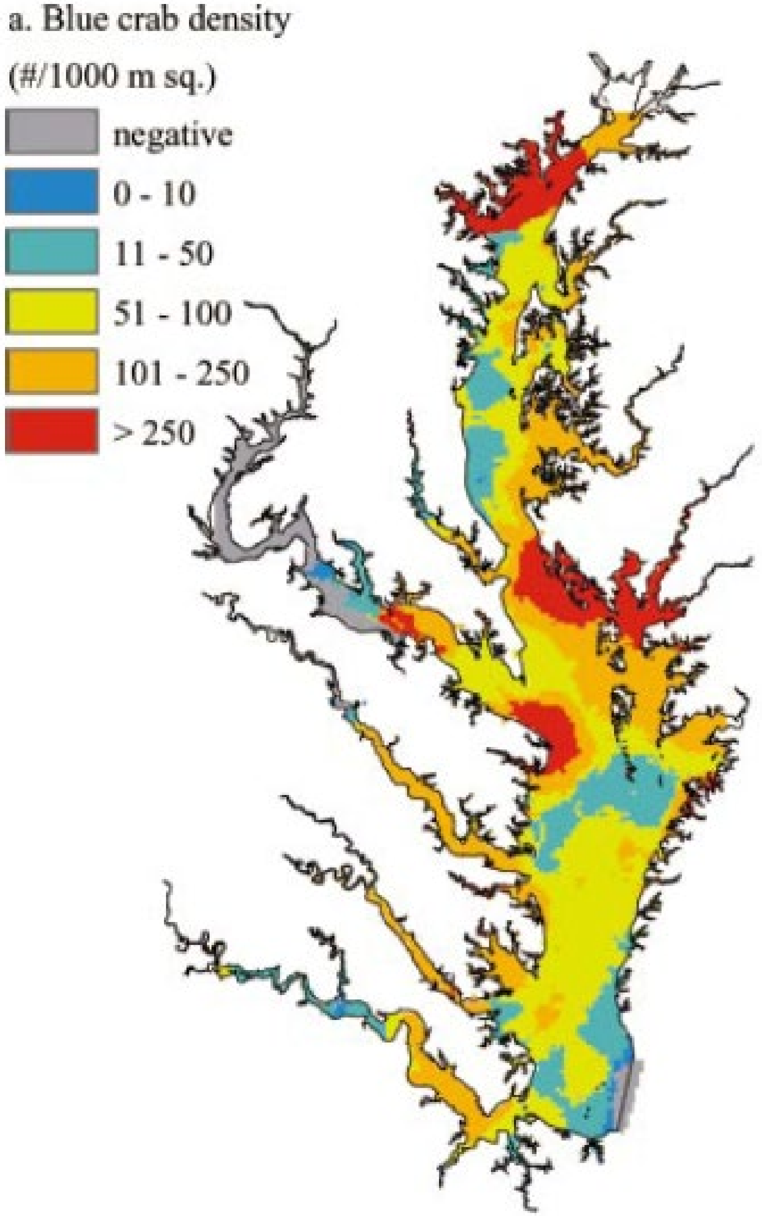
- Points
- Gridded or raster
- Lattice or polygon

Spatial point pattern
(spatial event data)

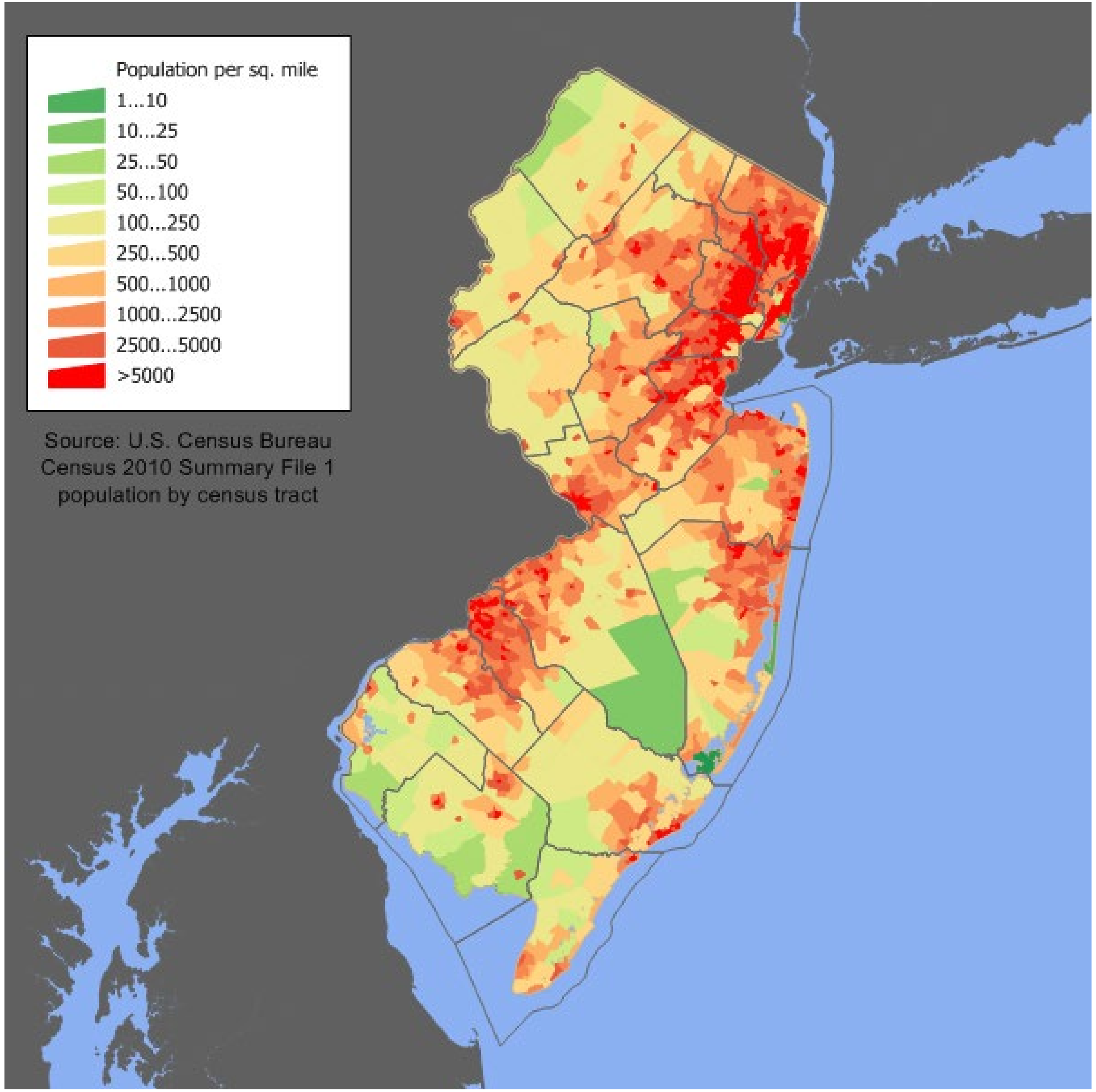
Gridded representation of a spatially continuous process




Susquehanna River
Potomac River
Patuxent River
Choptank River
Tangier Sound
Rappahannock River
Pocomoke Sound
York River
James River
Cape Charles
N
Cape Henry

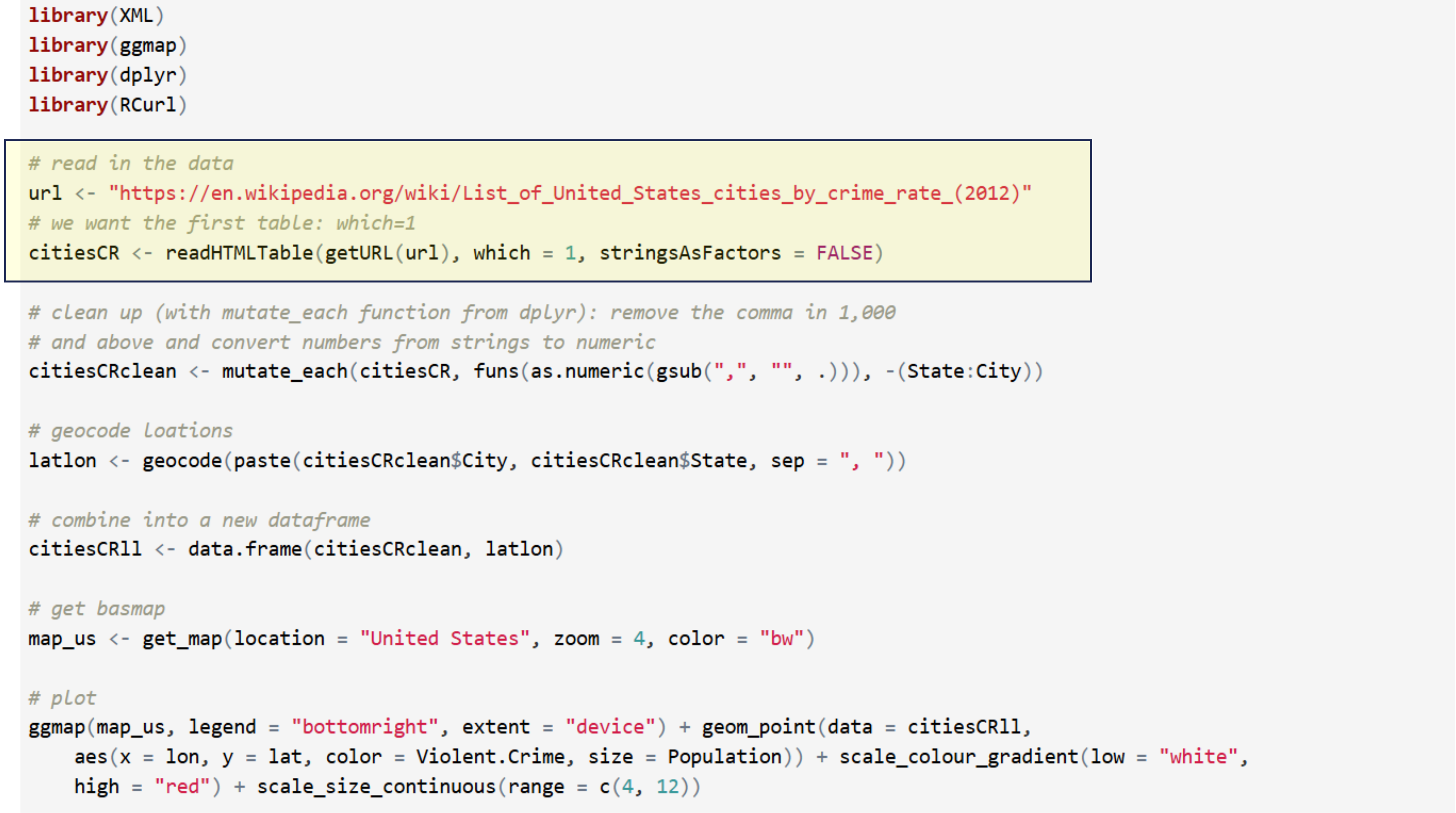

a. Blue crab density
(#/1000 m sq.)
negative
0 - 10
11 - 50
51 - 100
101 - 250
> 250

Lattice data
(regular and irregular lattices)



Source: U.S. Census Bureau
Census 2010 Summary File 1
population by census tract

Mapping points

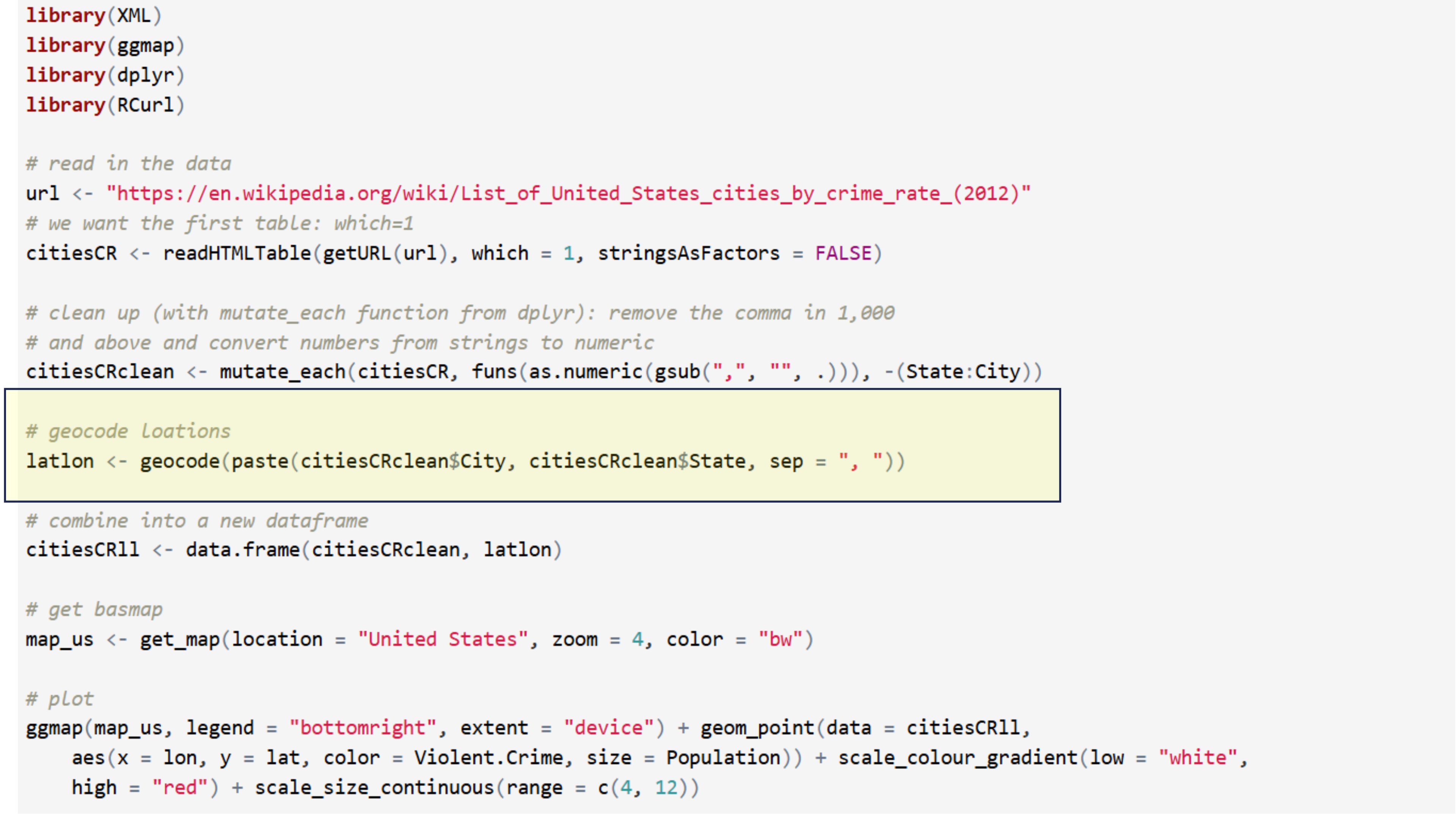


```
library(XML)
library(ggmap)
library(dplyr)
library(RCurl)

# read in the data
url <- "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate_(2012)"
# we want the first table: which=1
citiesCR <- readHTMLTable(getURL(url), which = 1, stringsAsFactors = FALSE)

# clean up (with mutate_each function from dplyr): remove the comma in 1,000
# and above and convert numbers from strings to numeric
citiesCRclean <- mutate_each(citiesCR, funs(as.numeric(gsub(",", "", .))), -(State:City))

# geocode loations
latlon <- geocode(paste(citiesCRclean$City, citiesCRclean$State, sep = ", "))

# combine into a new dataframe
citiesCRll <- data.frame(citiesCRclean, latlon)

# get basmap
map_us <- get_map(location = "United States", zoom = 4, color = "bw")

# plot
ggmap(map_us, legend = "bottomright", extent = "device") + geom_point(data = citiesCRll,
    aes(x = lon, y = lat, color = Violent.Crime, size = Population)) + scale_colour_gradient(low = "white",
    high = "red") + scale_size_continuous(range = c(4, 12))
```

Mapping points

- library(rvest)
- # Read and clean table
- url <- "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate_(2012)"
- citiesCR <- read_html(url) %>% html_table(fill = TRUE)
- citiesCR <- clean_names(citiesCR[[1]])

Mapping points

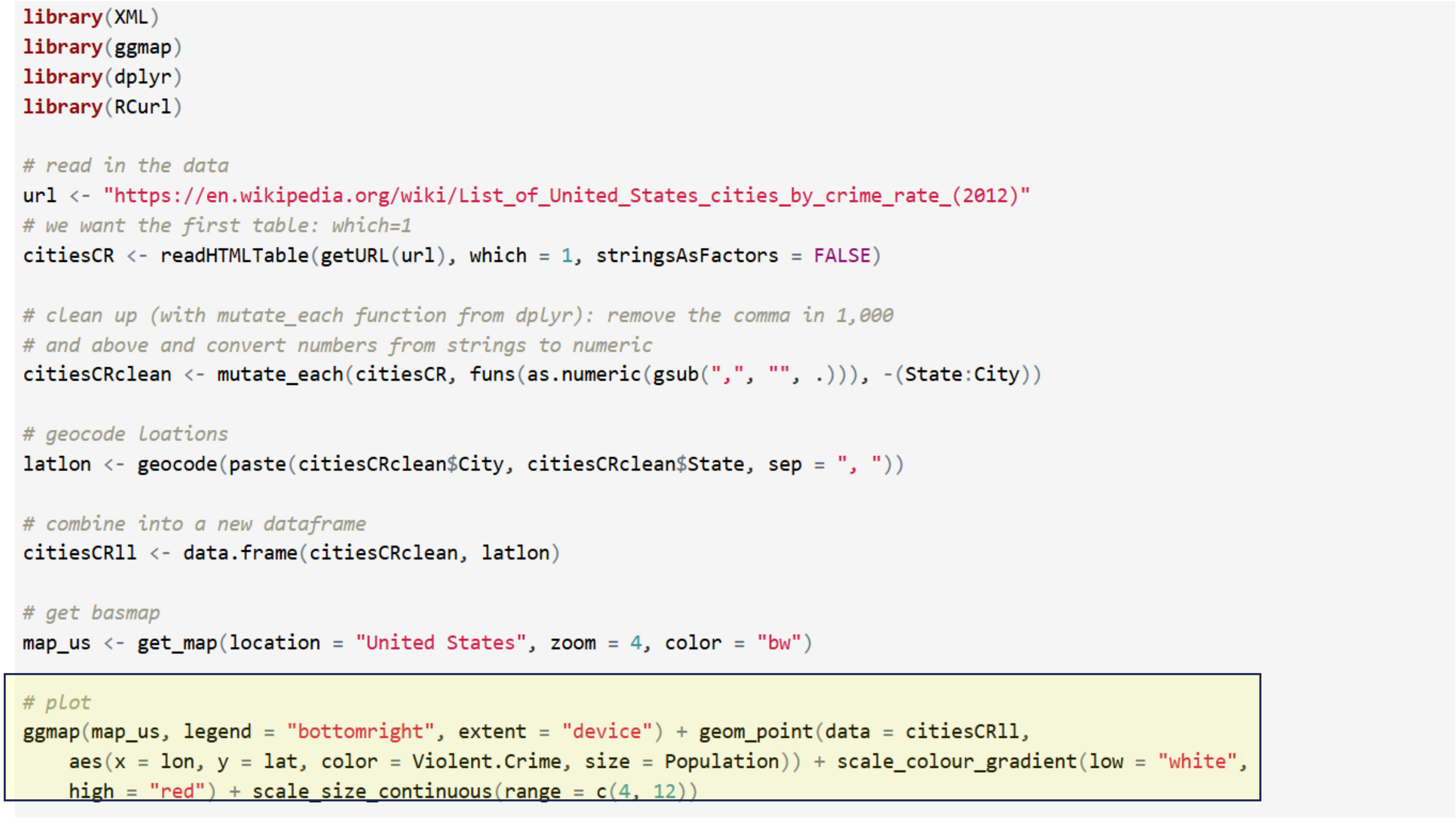


```r
library(XML)
library(ggmap)
library(dplyr)
library(RCurl)

# read in the data
url <- "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate_(2012)"
# we want the first table: which=1
citiesCR <- readHTMLTable(getURL(url), which = 1, stringsAsFactors = FALSE)

# clean up (with mutate_each function from dplyr): remove the comma in 1,000
# and above and convert numbers from strings to numeric
citiesCRclean <- mutate_each(citiesCR, funs(as.numeric(gsub(",", "", .))), -(State:City))

# geocode loations
latlon <- geocode(paste(citiesCRclean$City, citiesCRclean$State, sep = ", "))

# combine into a new dataframe
citiesCRll <- data.frame(citiesCRclean, latlon)

# get basmap
map_us <- get_map(location = "United States", zoom = 4, color = "bw")

# plot
ggmap(map_us, legend = "bottomright", extent = "device") + geom_point(data = citiesCRll,
    aes(x = lon, y = lat, color = Violent.Crime, size = Population)) + scale_colour_gradient(low = "white",
    high = "red") + scale_size_continuous(range = c(4, 12))
```

Mapping points

- library(tidygeocoder)
- # Prepare and geocode
- citiesCR <- citiesCR %>% mutate(location = paste(city, state, sep = ", ")) %>% geocode(location, method = "osm", lat = latitude, long = longitude)

Mapping points



```r
library(XML)
library(ggmap)
library(dplyr)
library(RCurl)

# read in the data
url <- "https://en.wikipedia.org/wiki/List_of_United_States_cities_by_crime_rate_(2012)"
# we want the first table: which=1
citiesCR <- readHTMLTable(getURL(url), which = 1, stringsAsFactors = FALSE)

# clean up (with mutate_each function from dplyr): remove the comma in 1,000
# and above and convert numbers from strings to numeric
citiesCRclean <- mutate_each(citiesCR, funs(as.numeric(gsub(",", "", .))), -(State:City))

# geocode loations
latlon <- geocode(paste(citiesCRclean$City, citiesCRclean$State, sep = ", "))

# combine into a new dataframe
citiesCRll <- data.frame(citiesCRclean, latlon)

# get basmap
map_us <- get_map(location = "United States", zoom = 4, color = "bw")

# plot
ggmap(map_us, legend = "bottomright", extent = "device") + geom_point(data = citiesCRll,
    aes(x = lon, y = lat, color = Violent.Crime, size = Population)) + scale_colour_gradient(low = "white",
    high = "red") + scale_size_continuous(range = c(4, 12))
```

Mapping points

- library(ggplot2)
- library(maps)
- # Get a basic US map
- us_map <- map_data("state")
- # Plot
- ggplot() +
- geom_polygon(data = us_map, aes(x = long, y = lat, group = group), fill = "gray95", color = "gray70")

Exercise

Use ChatGPT to help debug/update the example for lattice data (example 1) from the reading: https://www.rpubs.com/cengel248/97543

Hints:

Work section by section until you get errors

Provide ChatGPT with the error messages and the code that generated them

ChatGPT can usually figure out what you’re trying to do from the code, but sometimes it makes mistakes