Notes for STAT 5413 - Spatial Statistics

John Tipton

Fall 2020 Semester. Last Modified: 2020-01-14

Contents

Preface

These are the lecture notes for STAT 5413 Fall 2020.

Chapter 1

Day 1

```
library(tidyverse)
```

1.1 Notation

The dimensions of different mathematical objects are very important for the study of spatial statistics. To communicate this, we use the following notation. A scalar random variable is represented by a lowercase alphanumeric letter (x, y, z, etc.), a vector random variable is respresented by a bold lowercase alphanumeric letter ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, etc.), and a matrix random variable is respresented by a bold uppercase alphanumeric letter ($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, etc.). We use a similar notation for parameters as well where scalar parameters are represented by a lowercase Greek letter (μ, α, β, etc.), a vector parameter is respresented by a bold lowercase Greek letter (μ, α, β, etc.), and a matrix random variable is respresented by a bold uppercase Greek letter (Σ, Ψ, Γ, etc.).

1.2 Probability Distributions

We also need notation to explain probability distributions. We use the notation $[y]$ to denote the probability density function $p(y)$ of the random variable y and $[y|x]$ to denote the probability density function $p(y|x)$ of y given x. For example, if y is a Gaussian random variable with mean μ and standard deviation σ we write

$$[y|\mu, \sigma] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(y-\mu)^2\right\}.$$

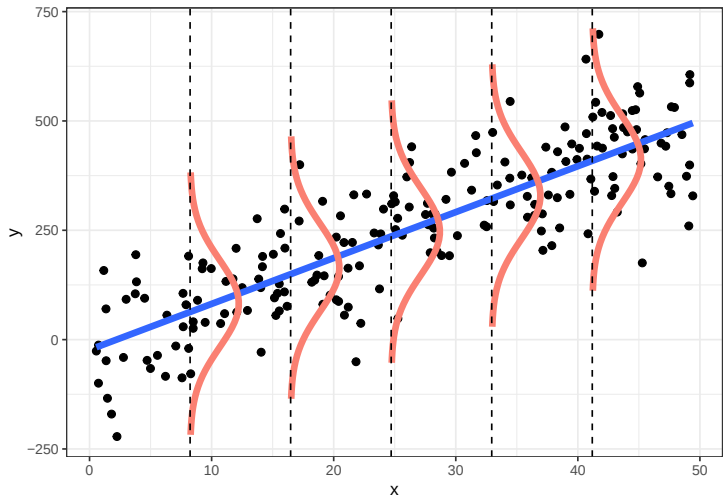
We can also denote that y has a Gaussian (normal) distribution given mean μ and variance σ^2 using the $\sim$ notation

$$y|\mu, \sigma \sim \mathrm{N}(\mu, \sigma^2).$$

1.2.1 Example: linear regression

$$[y_i|\theta] \sim \mathrm{N}(X_i\beta, \sigma^2)$$
$$\theta = (\beta, \sigma^2)$$

```
## Sample data
set.seed(404)
dat <- data.frame(x=(x=runif(200, 0, 50)),
                  y=rnorm(200, 10 * x, 100))

## breaks: where you want to compute densities
breaks <- seq(0, max(dat$x), len=7)[-c(1, 7)]
dat$section <- cut(dat$x, breaks)

## Get the residuals
dat$res <- residuals(lm(y ~ x, data=dat))

## Compute densities for each section, and flip the axes, and add means of sections
## Note: the densities need to be scaled in relation to the section size (2000 here)
ys <- seq(-300, 300, length = 50)
xs <- rep(breaks, each = 50) + 1000 * dnorm(ys, 0, 100)
res <- matrix(0, 50, 5)
for (i in 1:5) {
  res[, i] <- 10 * breaks[i] + ys
}
dens <- data.frame(x = xs, y=c(res),
                   grouping = cut(xs, breaks))
ggplot(dat, aes(x, y)) +
  geom_point(size = 2) +
  geom_smooth(method="lm", fill=NA, lwd=2, se = FALSE) +
  geom_path(data=dens, aes(x, y, group = grouping),
            color="salmon", lwd=2) +
  theme_bw() +
  geom_vline(xintercept=breaks, lty=2)
```



1.3 Hierarchical modeling

- Follow Berliner (1996) framework for hierarchical probability models
- Model encodes our understanding of the scientific process of interest
- Model accounts for as much uncertainty as possible
- Model results in a probability distribution
  - Note: nature may be deterministic – often probabilistic models outperform physical models.
  - Example: model individual rain drops vs. probability/intensity of rain
- Update model with data
- Use the model to generate parameter estimates given data

1.3.1 Bayesian Hierarchical models (BHMs)

- Break the model into components:
  - Data Model.
  - Process Model.
  - Parameter Model.
- Combined, the data model, the process model, and the parameter model define a posterior distribution.

$$[\mathbf{z}, \theta_D, \theta_P|\mathbf{y}] \propto [\mathbf{y}|\theta_D, \mathbf{z}][\mathbf{z}|\theta_P][\theta_D][\theta_P]$$

1.3.2 Empirical Hierarchical models (EHMs)

- Break the model into components:
  - Data Model.
  - Process Model.
  - Parameter estimates (fixed values) are substituted before fitting the model
- Combined, the data model and the process model define a predictive distribution. Thus, numerical evaluation of the predictive distribution is typically required to estimate unceratinty (bootstrap, MLE asymptotics)
  - Note: the predictive distribution is not a posterior distribution because the normalizing constant is not known

$$[\mathbf{z}|\mathbf{y}] \propto [\mathbf{y}|\theta_D, \mathbf{z}][\mathbf{z}|\theta_P]$$

1.3.3 Data Model

$$[\mathbf{y}|\theta_D, \mathbf{z}]$$

- Describes how the data are collected and observed.
- Account for measurement process and uncertainty.
- Model the data in the manner in which they were collected.
- Data $\mathbf{y}$.
  - Noisy.
  - Expensive.
  - Not what you want to make inference on.
- Latent variables $\mathbf{z}$.
  - Think of $\mathbf{z}$ as the ideal data.
  - No measurement error - the exact quantity you want to observe but can't.
- Data model parameters θ_D.

1.3.4 Process Model

$$[\mathbf{z}|\theta_P]$$

- **Where the science happens!**
- Latent process $\mathbf{z}$ is modeled.
- Can be dynamic in space and/or time
- Process parameters θ_P.
- Virtually all interesting scientific questions can be made with inference about $\mathbf{z}$

1.3.5 Parameter (Prior) Model (BMHs only)

$$[\theta_D][\theta_P]$$

- Probability distributions define "reasonable" ranges for parameters.
- Parameter models are useful for a variety of problems:
  - Choosing important variables.
  - Preventing over-fitting (regularization).
  - "Pooling" estimates across categories.

1.3.6 Posterior Distribution

$$[\mathbf{z}, \theta_D, \theta_P|\mathbf{y}] \propto [\mathbf{y}|\theta_D, \mathbf{z}][\mathbf{z}|\theta_P][\theta_D][\theta_P]$$

- Probability distribution over all unknowns in the model.
- Inference is made using the posterior distribution.
- Because the posterior distribution is a probability distribution (BHMs), uncertainty is easy to calculate. This is not true for EHMs.

1.3.7 Scientifically Motivated Statistical Modeling

- Criticize the model
- Does the model fit the data well?
- Do the predictions make sense?
- Are there subsets of the data that don't fit the model well?
- Make inference using the model.
- If the model fits the data, use the model fit for prediction or inference.

Chapter 2

Day 2

```
library(tidyverse)
library(here)
library(sp)
```

2.1 Spatial Data

All data occur at some location is space and time. For know we focus on spatial analyses and will later extend this to spatio-temporal analyses. Let $\mathcal{D}$ represent the spatial domain and let $\mathbf{s}$ be a spatial location. In general, we will let $\mathcal{A} \subset \mathcal{D}$ be a subdomain of the spatial region of $\mathbf{D}$.

Insert Diagram from class here

2.2 Types of spatial data

There are three primary types of spatial data that we are going to consider

2.2.1 Geostatistical data

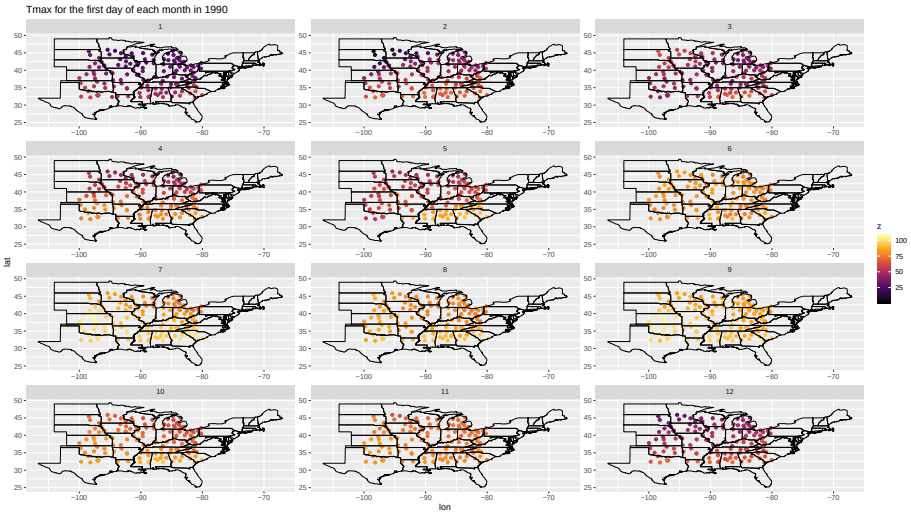
- Occur everywhere
- continuous support
- examples: temperature, precipitation

```
data("NOAA_df_1990", package = "STRbook")
glimpse(NOAA_df_1990)
```

```
## Observations: 730,486
## Variables: 10
## $ julian <int> 726834, 726835, 726836, 726837, 726838, 726839, 726840, 7268...
```

```
## $ year   <int> 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, 1990, ...
## $ month  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ day    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1...
## $ id     <dbl> 3804, 3804, 3804, 3804, 3804, 3804, 3804, 3804, 3804, 3804, ...
## $ z      <dbl> 35, 42, 49, 59, 41, 45, 46, 42, 54, 43, 52, 38, 32, 43, 53, ...
## $ proc   <chr> "Tmax", "Tmax", "Tmax", "Tmax", "Tmax", "Tmax", "Tmax", "Tma...
## $ lat    <dbl> 39.35, 39.35, 39.35, 39.35, 39.35, 39.35, 39.35, 39.35, 39.3...
## $ lon    <dbl> -81.43333, -81.43333, -81.43333, -81.43333, -81.43333, -81.4...
## $ date   <date> 1990-01-01, 1990-01-02, 1990-01-03, 1990-01-04, 1990-01-05,...
```

```r
## Only plot the states with data
states <- map_data("state")
states <- states %>%
    subset(!(region %in% c("washington", "oregon", "california", "nevada", "idaho", "ut

## generate map
NOAA_df_1990 %>%
    subset(year == 1990 & day == 1 & proc == "Tmax") %>%
    ggplot(aes(x = lon, y = lat, color = z)) +
    geom_point() +
    facet_wrap(~ month, scales = "free", nrow = 4) +
    geom_polygon(data = states, aes(x = long, y = lat, group = group),
                 inherit.aes = FALSE, fill = NA, color = "black") +
    scale_color_viridis_c(option = "inferno") +
    ggtitle("Tmax for the first day of each month in 1990")
```



2.2.2 Areal data

- Occur only over discrete areas

- can be thought of as an integral of a continuous process over a subdomain $\mathcal{A} \in \mathcal{D}$
- examples: cases of a disease by counties, votes in an election by congressional district

```
data("BEA", package = "STRbook")
glimpse(BEA)
```

```
## Observations: 116
## Variables: 5
## $ Description <chr> "Per capita personal income (dollars)", "Per capita per...
## $ NAME10      <fct> "Adair, MO", "Andrew, MO", "Atchison, MO", "Audrain, MO...
## $ X1970       <int> 2723, 3577, 3770, 3678, 3021, 2832, 3263, 2508, 2147, 3...
## $ X1980       <int> 7399, 7937, 5743, 8356, 7210, 7445, 8596, 6125, 5431, 9...
## $ X1990       <int> 12755, 15059, 14748, 15198, 12873, 13530, 13195, 11854,...
```

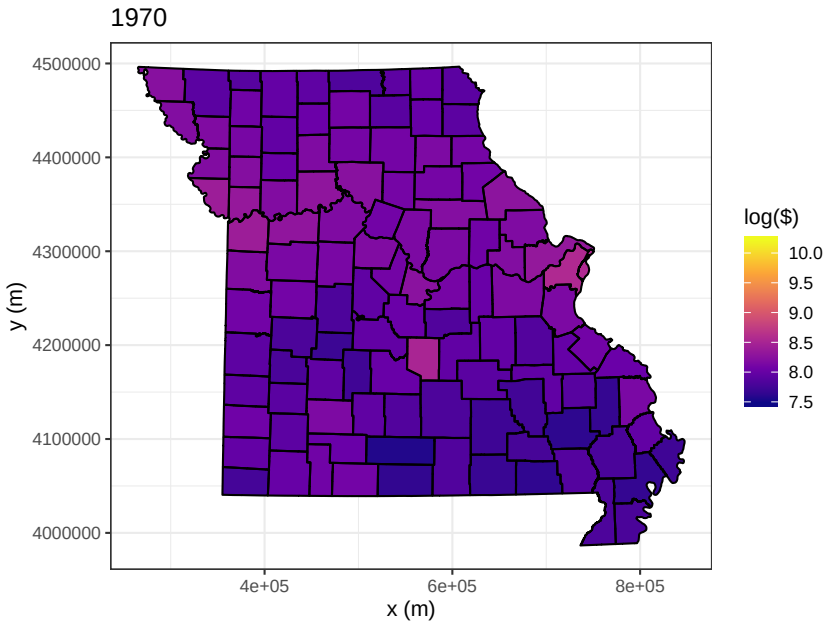
```
data("MOcounties", package = "STRbook")
glimpse(MOcounties)
```

```
## Observations: 214,279
## Variables: 53
## $ long       <dbl> 627911.9, 627921.4, 627923.0, 627947.8, 627956.5, 627994...
## $ lat        <dbl> 4473554, 4473559, 4473560, 4473577, 4473583, 4473612, 44...
## $ order      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1...
## $ hole       <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ...
## $ piece      <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ id         <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "...
## $ group      <fct> 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0...
## $ STATEFP10  <fct> 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, ...
## $ COUNTYFP10 <fct> 045, 045, 045, 045, 045, 045, 045, 045, 045, 045, 045, 0...
## $ COUNTYNS10 <fct> 00758477, 00758477, 00758477, 00758477, 00758477, 007584...
## $ GEOID10    <fct> 29045, 29045, 29045, 29045, 29045, 29045, 29045, 29045, ...
## $ NAME10     <fct> "Clark, MO", "Clark, MO", "Clark, MO", "Clark, MO", "Cla...
## $ NAMELSAD10 <fct> Clark County, Clark County, Clark County, Clark County, ...
## $ LSAD10     <fct> 06, 06, 06, 06, 06, 06, 06, 06, 06, 06, 06, 06, 06, 06, ...
## $ CLASSFP10  <fct> H1, H1, H1, H1, H1, H1, H1, H1, H1, H1, H1, H1, H1, H1, ...
## $ MTFCC10    <fct> G4020, G4020, G4020, G4020, G4020, G4020, G4020, G4020, ...
## $ CSAFP10    <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ CBSAFP10   <fct> 22800, 22800, 22800, 22800, 22800, 22800, 22800, 22800, ...
## $ METDIVFP10 <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ FUNCSTAT10 <fct> A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, ...
## $ ALAND10    <dbl> 1307146971, 1307146971, 1307146971, 1307146971, 13071469...
## $ AWATER10   <dbl> 18473547, 18473547, 18473547, 18473547, 18473547, 184735...
## $ INTPTLAT10 <fct> +40.4072748, +40.4072748, +40.4072748, +40.4072748, +40....
## $ INTPTLON10 <fct> -091.7294720, -091.7294720, -091.7294720, -091.7294720, ...
## $ AREA       <dbl> 1324937990, 1324937990, 1324937990, 1324937990, 13249379...
## $ PERIMETER  <dbl> 161503.6, 161503.6, 161503.6, 161503.6, 161503.6, 161503...
```

```
## $ COUNTY10_  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,...
## $ COUNTY10_I <int> 115, 115, 115, 115, 115, 115, 115, 115, 115, 115, 115, 1...
## $ POP90      <int> 7547, 7547, 7547, 7547, 7547, 7547, 7547, 7547, 7547, 75...
## $ WHITE90    <int> 7528, 7528, 7528, 7528, 7528, 7528, 7528, 7528, 7528, 75...
## $ BLACK90    <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,...
## $ ASIANPI90  <int> 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
## $ AMIND90    <int> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,...
## $ OTHER90    <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ HISP90     <int> 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, ...
## $ POP00      <int> 7416, 7416, 7416, 7416, 7416, 7416, 7416, 7416, 7416, 74...
## $ WHITE00    <int> 7329, 7329, 7329, 7329, 7329, 7329, 7329, 7329, 7329, 73...
## $ BLACK00    <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ ASIAN00    <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ AMIND00    <int> 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, ...
## $ HAWNPI00   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ OTHER00    <int> 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, ...
## $ MULTRA00   <int> 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, ...
## $ HISP00     <int> 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, ...
## $ POP10      <int> 7139, 7139, 7139, 7139, 7139, 7139, 7139, 7139, 7139, 71...
## $ WHITE10    <int> 7011, 7011, 7011, 7011, 7011, 7011, 7011, 7011, 7011, 70...
## $ BLACK10    <int> 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, ...
## $ ASIAN10    <int> 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, ...
## $ AMIND10    <int> 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,...
## $ HAWNPI10   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ OTHER10    <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,...
## $ MULTRA10   <int> 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, ...
## $ HISP10     <int> 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, ...
```

```
MOcounties <- left_join(MOcounties, BEA, by = "NAME10")
```

```
ggplot(MOcounties) +
    geom_polygon(aes(x = long,
                     y = lat, # county boundary
                     group = NAME10, # county group
                     fill = log(X1970))) + # log of income
    geom_path(aes(x = long, y = lat, group = NAME10)) +
    scale_fill_viridis_c(limits = c(7.5, 10.2), option = "plasma", name = "log($)") +
    coord_fixed() +
    ggtitle("1970") +
    xlab("x (m)") +
    ylab("y (m)") +
    theme_bw()
```



2.2.3 Point process data

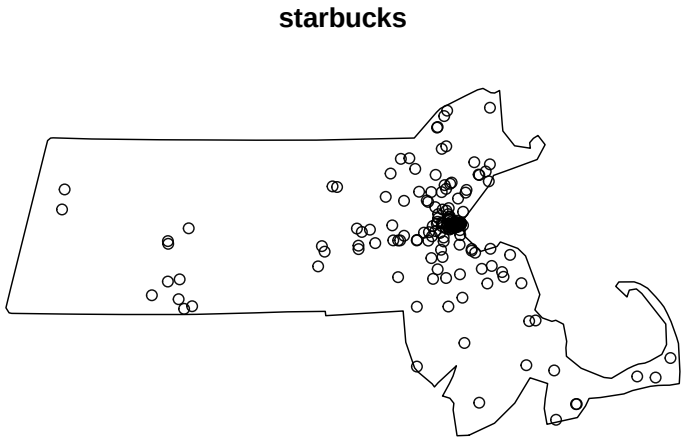
- The count and location of the data are random
- examples: tornados, lightning strikes

```
# uncomment out this line to download the data
# load(url("http://github.com/mgimond/Spatial/raw/master/Data/ppa.RData"))
# save(starbucks, ma, pop, file = here::here("data", "ppa-starbucks.RData"))
load(here::here("data", "ppa-starbucks.RData"))
glimpse(starbucks)
```

```
## List of 5
##  $ window    :List of 4
##   ..$ type  : chr "rectangle"
##   ..$ xrange: num [1:2] 648032 917741
##   ..$ yrange: num [1:2] 4609785 4748107
##   ..$ units :List of 3
##   .. ..$ singular  : chr "unit"
##   .. ..$ plural    : chr "units"
##   .. ..$ multiplier: num 1
##   .. ..- attr(*, "class")= chr "unitname"
##   ..- attr(*, "class")= chr "owin"
##  $ n         : int 171
##  $ x         : num [1:171] 917741 911147 902987 876188 875868 ...
##  $ y         : num [1:171] 4637151 4628510 4628982 4616741 4616719 ...
```

```
##  $ markformat: chr "none"
##  - attr(*, "class")= chr "ppp"
```

```
library(spatstat)
## add the massachusetts polygon
Window(starbucks) <- ma
marks(starbucks) <- NULL
## plot using the plot function from spatstat
plot(starbucks)
```



- Many different file types for spatial data
- Typically data are in "flat files" like comma-seperated value (CSV) files

```
read.csv(here("path", "to", "file.csv"))
```

- "shapefiles" which can be read using *rgdal* or *maptools* packages

```
library(rgdal)
library(maptools)
```

- "NetCDF" files cane be read using *ncdf4* or *RNetCDF*

```
library(ncdf4)
library(RNetCDF)
```

2.3 Textbook package

To install the data from the textbook, go to https://spacetimewithr.org/ and follow the link to the code.

```
# install.packages("devtools")
library(devtools)
install_github("andrewzm/STRbook")
```

Note that this package is relatively large because it contains a decent amount of spatial data.

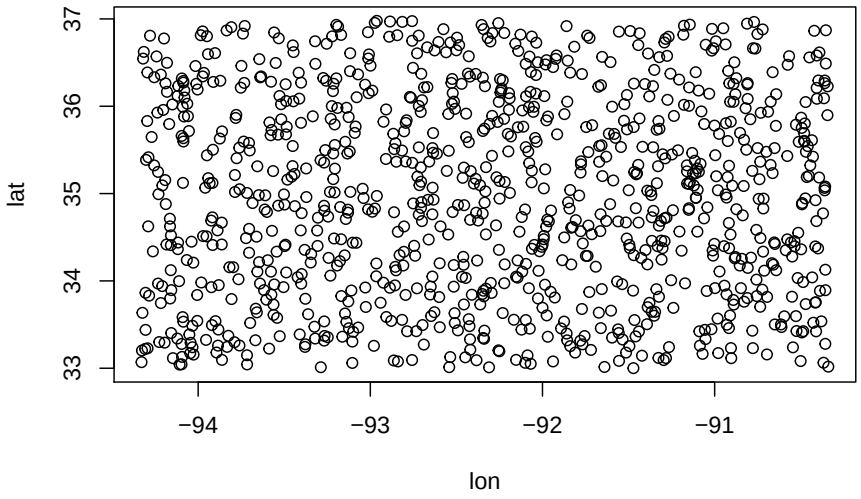
```
library(STRbook)
```

2.4 Spatial Visualization

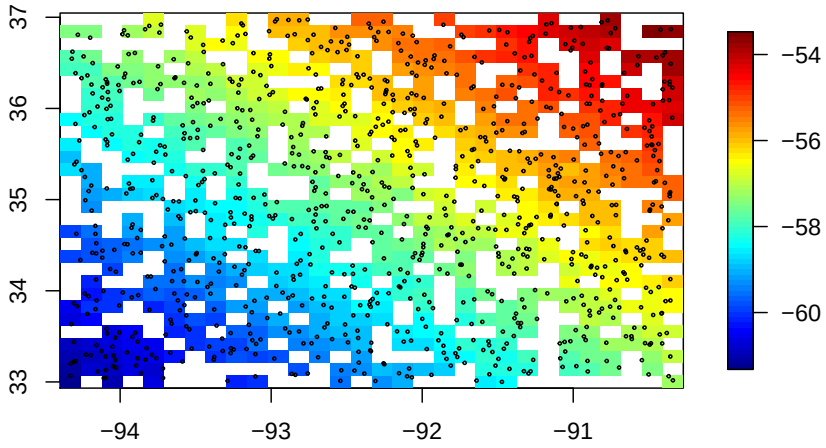
2.4.1 Spatial visualization using *fields*

- Simulate a process with some random locations

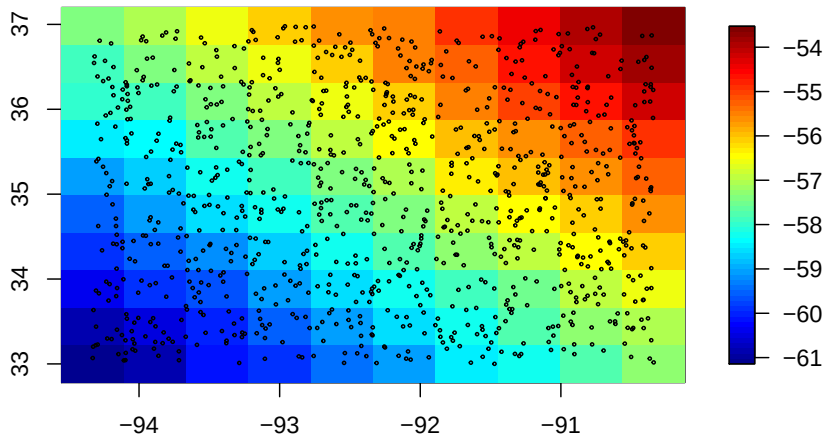
```
library(fields)
## longitude and latitude of approximately the center of Arkansas
lon_lat_center <- c(-92.33, 35.00)

n   <- 1000
## simulate some random locations
lon  <- runif(n, lon_lat_center[1] - 2, lon_lat_center[1] + 2)
lat  <- runif(n, lon_lat_center[2] - 2, lon_lat_center[2] + 2)
y   <- rnorm(n, lat + lon, .1)

plot(lon, lat)
```




```
quilt.plot(lon, lat, y, nx = 30, ny = 30)
points(lon, lat, cex = .3)
```




```
quilt.plot(lon, lat, y, nx = 10, ny = 10)
points(lon, lat, cex = .3)
```



- Simulate a process on a regular grid

```
n <- 50^2
## simulate locations on a grid
lon  <- seq(lon_lat_center[1] - 2, lon_lat_center[1] + 2, length = sqrt(n))
lat  <- seq(lon_lat_center[2] - 2, lon_lat_center[2] + 2, length = sqrt(n))
s <- expand.grid(lon, lat)

head(lon)
```

```
## [1] -94.33000 -94.24837 -94.16673 -94.08510 -94.00347 -93.92184
```

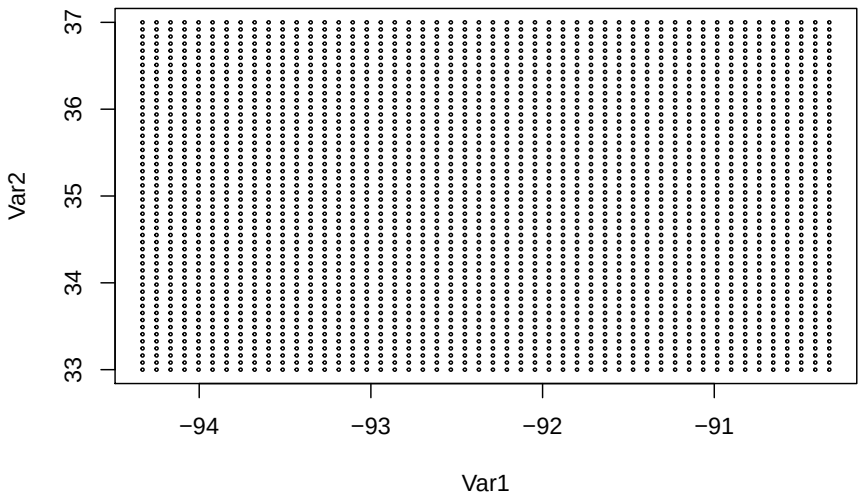
```
head(lat)
```

```
## [1] 33.00000 33.08163 33.16327 33.24490 33.32653 33.40816
```

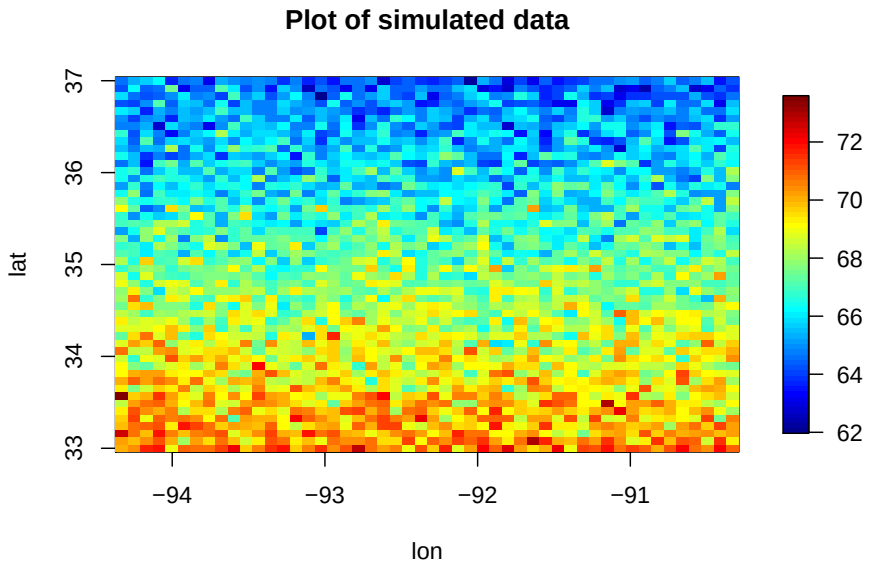
```
head(s)
```

```
##        Var1 Var2
## 1 -94.33000   33
## 2 -94.24837   33
## 3 -94.16673   33
## 4 -94.08510   33
## 5 -94.00347   33
## 6 -93.92184   33
```

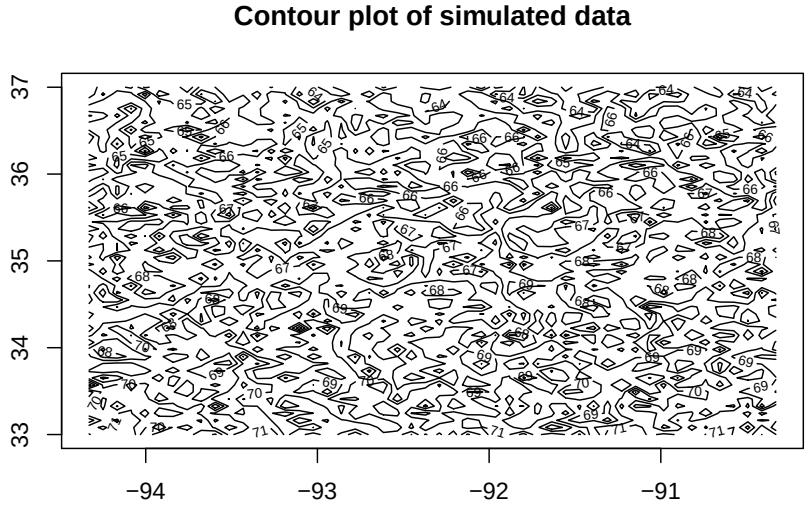
```
plot(s, cex = 0.3)
```



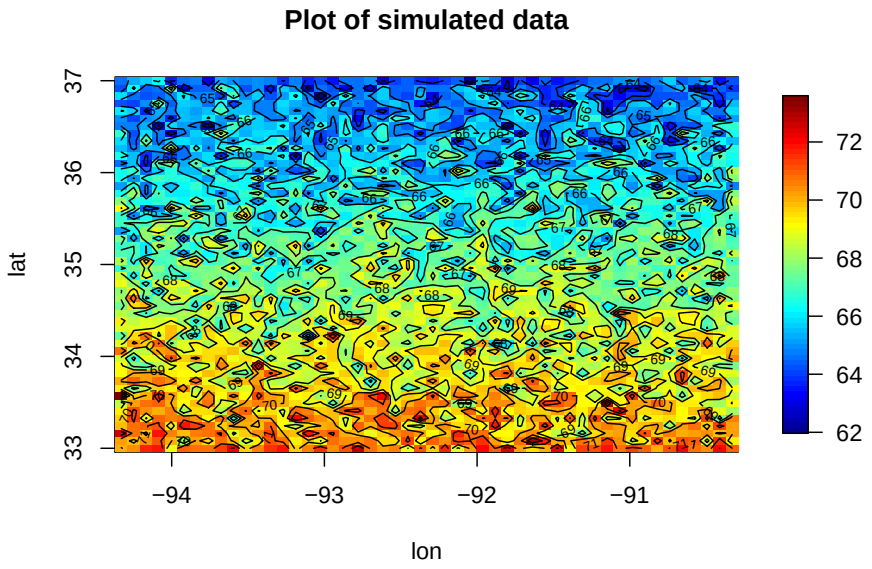

```
## simulate some fake data with a north/south trend
y <- 120 - 1.5 * s[, 2] + matrix(rnorm(n), sqrt(n), sqrt(n))
image.plot(lon, lat, y, main = "Plot of simulated data")
```



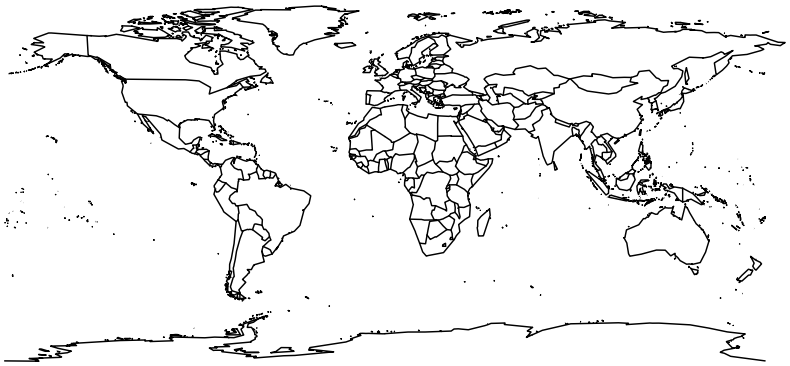

```
contour(lon, lat, y, main = "Contour plot of simulated data")
```



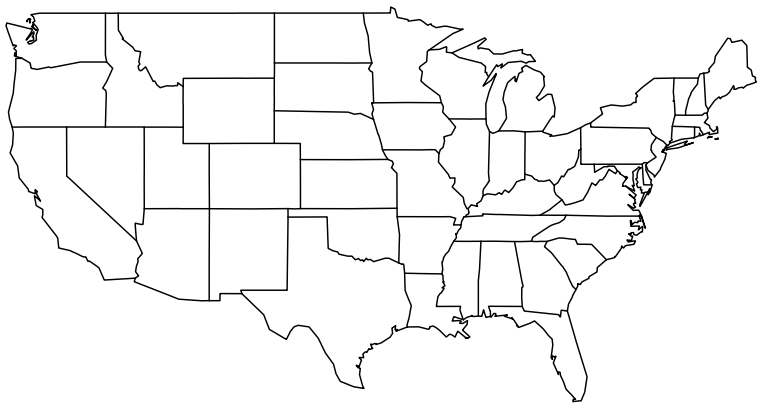

```
image.plot(lon, lat, y, main = "Plot of simulated data")
contour(lon, lat, y, main = "Contour plot of simulated data", add = TRUE,
        nlevels = 10)
```



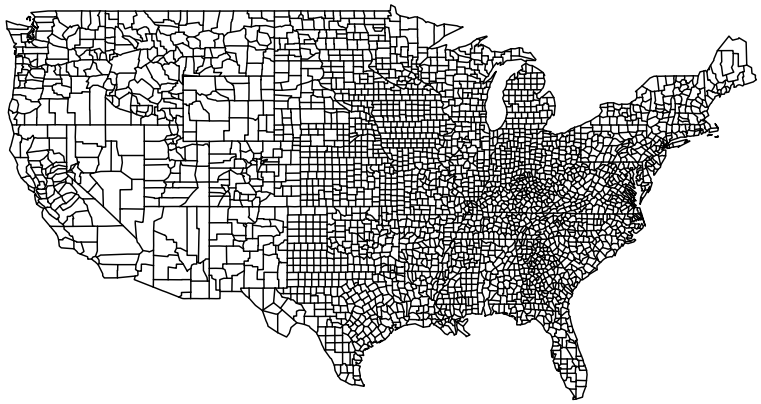

```
## adding in maps
library(maps)
maps::map("world")
```



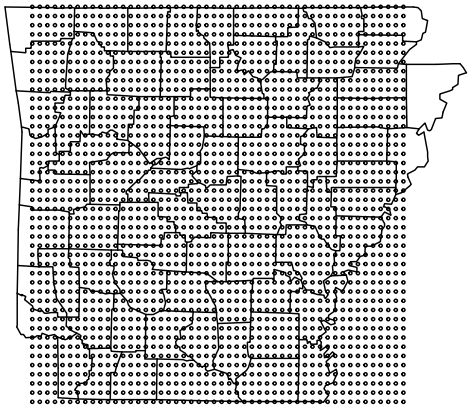
```
maps::map("state")
```



```
maps::map("county")
```



```
maps::map("county", "Arkansas")
points(s, cex = 0.3)
```



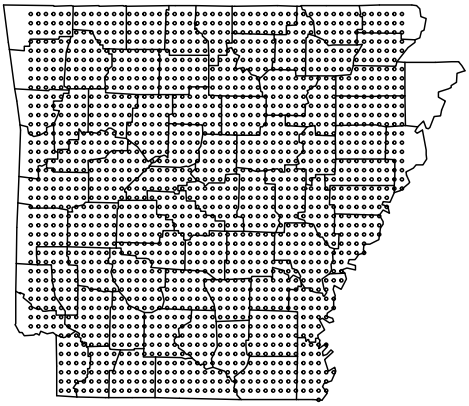
```
state <- map.where("state", x = s[, 1], y = s[, 2])
head(state)
```

```
## [1] "texas"     "texas"     "texas"     "texas"     "louisiana" "louisiana"
```

```
table(state)
```

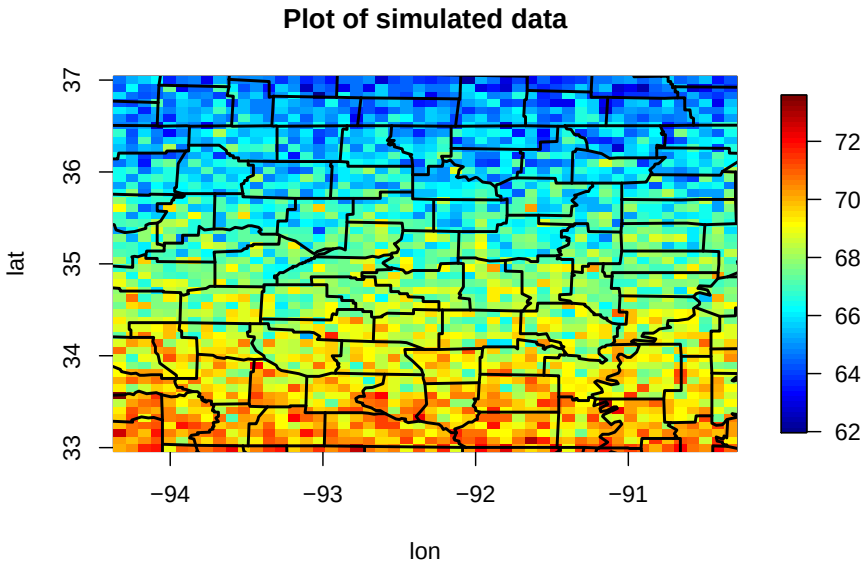
```
## state
##    arkansas   louisiana mississippi    missouri       texas
##        1903          34         180         351          32
```

```
## subset only points in arkansas
dat <- data.frame(
    lon   = s[, 1],
    lat   = s[, 2],
    state = state
)

maps::map("county", "Arkansas")
dat %>%
    subset(state == "arkansas") %>%
    points(cex = 0.3)
```

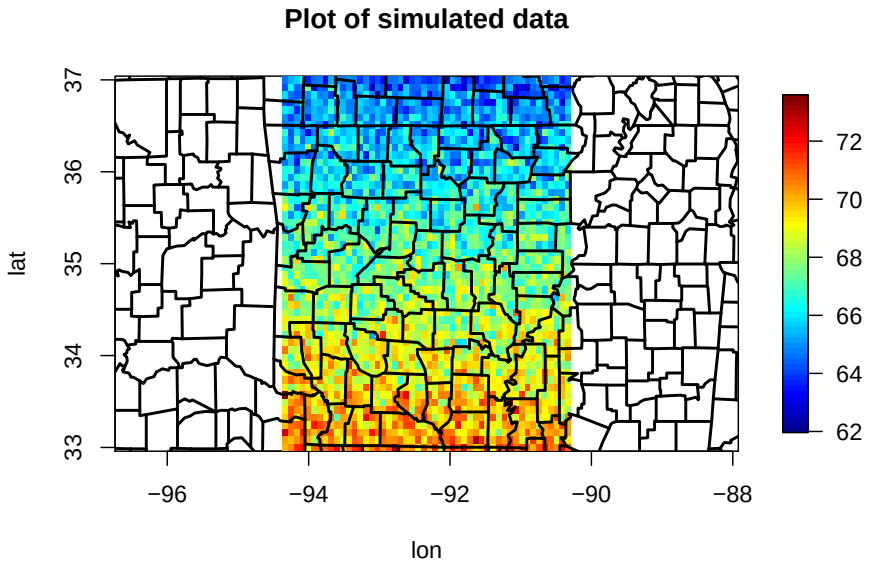


Plot the simulated data with the county boundaries

```
image.plot(lon, lat, y, main = "Plot of simulated data")
maps::map("county", add = TRUE, lwd = 2)
```



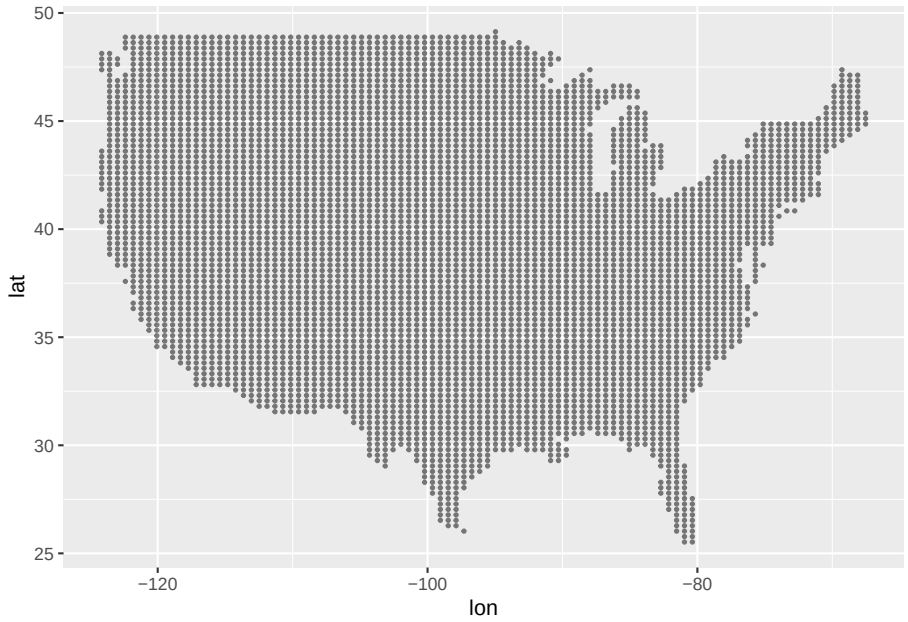
```
## change the aspect ratio
image.plot(lon, lat, y, main = "Plot of simulated data", asp = 1.3)
maps::map("county", add = TRUE, lwd = 2)
```



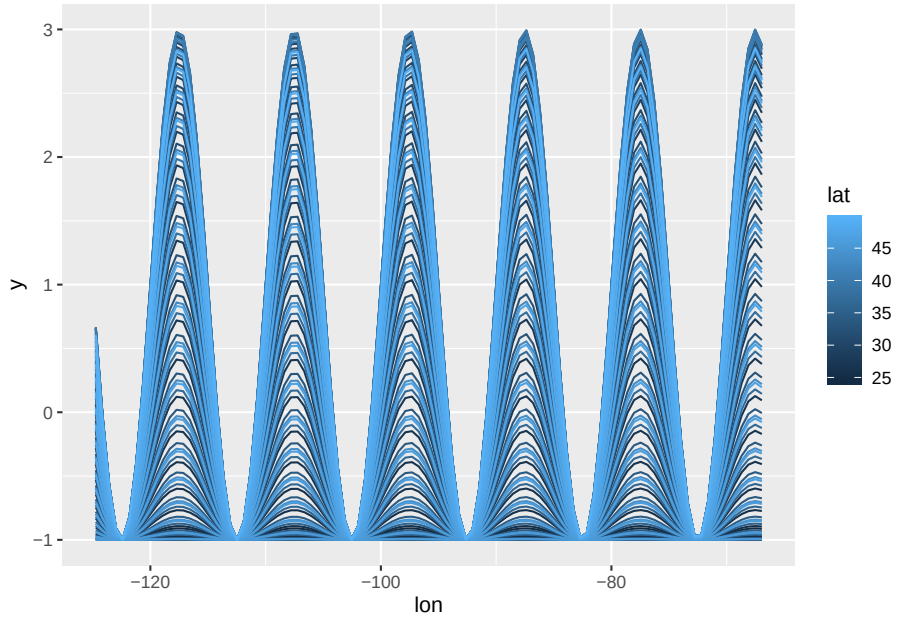
2.4.2 Spatial visualization using *fields*

```
nx <- 100
ny <- 100

library(maps)  # for map.where

# Corner of the USA
corners <- c(-124.733056, -66.947028, 24.520833, 49.384472)

# create grid
grid <- expand.grid(
    seq(corners[1], corners[2], length = nx),
    seq(corners[3], corners[4], length = ny)
)

dat <- data.frame(
    lon  = grid[, 1],
    lat  = grid[, 2],
    inUS = ifelse(is.na(map.where("usa", x = grid[, 1], y = grid[, 2])), FALSE, TRUE)
```

```
)

## Plot only points in the us

dat %>%
    subset(inUS) %>%   ## this selects only the true values
    ggplot(aes(x = lon, y = lat)) +
    geom_point(size = 0.6, alpha = 0.5)
```



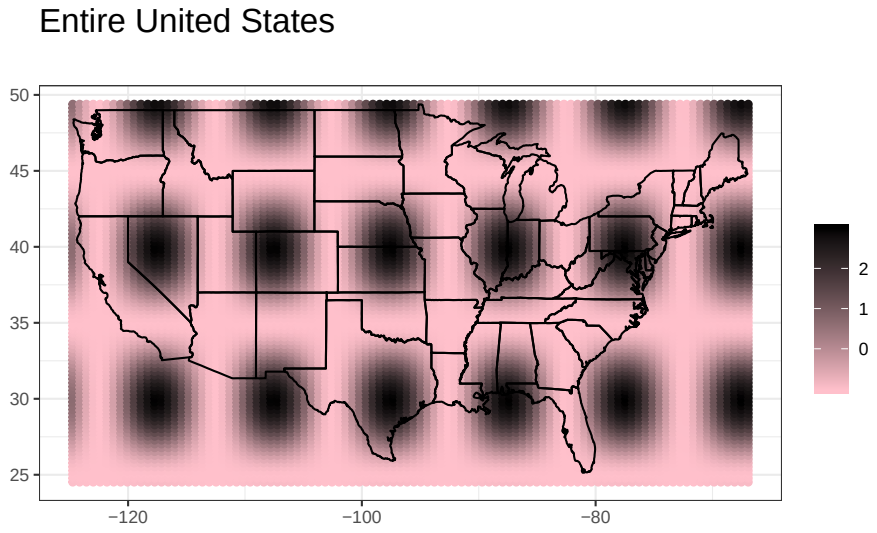

```
## Simulate some data over the grid
dat$y <- sin(2 * pi * dat$lon / 10) + cos(2 *pi * dat$lat / 10) +
    sin(2 * pi * dat$lon / 10) * cos(2 *pi * dat$lat / 10)

## plot each of the responses grouped by latitude
dat %>%
    ggplot(aes(x = lon, y = y, group = lat, color = lat)) +
    geom_line()
```



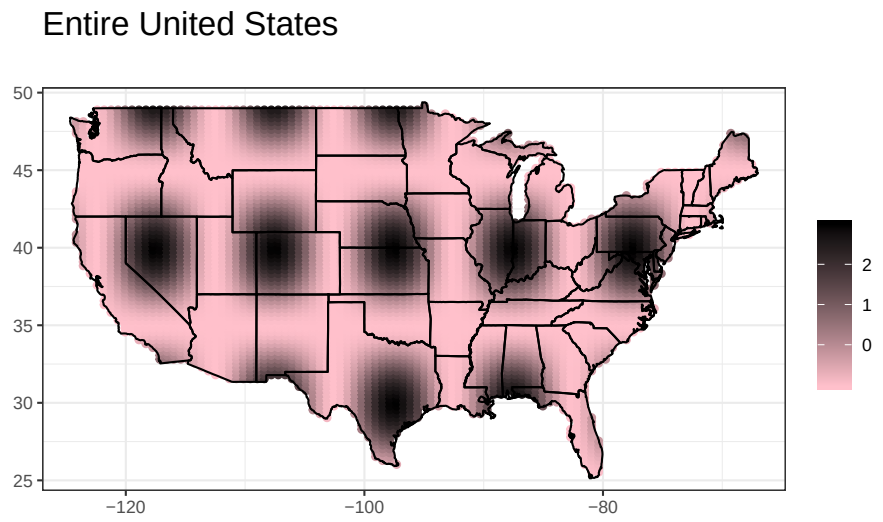

```
## Function to generate maps
map_points <- function (dat,
                        color_low = "white", color_high = "darkred",
                        color_na = gray(0.9), zeroiswhite = FALSE,
                        xlim = NULL, ylim = NULL, zlim = NULL,
                        mainTitle = NULL, legendTitle = "") {
    library(ggplot2)

    ## check if the data.fram dat contains the correct variables
    if (is.null(dat$lon)) { stop('The data.frame dat must contain a "lon" variable') }
    if (is.null(dat$lat)) { stop('The data.frame dat must contain a "lat" variable') }
    if (is.null(dat$y))   { stop('The data.frame dat must contain a "y" variable') }

    # Store the base data of the underlying map
    states <- map_data("state")

    # Set limits for x, y, z if not specified as parameters
    if (is.null(xlim)) { xlim <- range(dat$lon, na.rm = TRUE) }
    if (is.null(ylim)) { ylim <- range(dat$lat, na.rm = TRUE) }
    if (is.null(zlim)) { zlim <- range(dat$y, na.rm = TRUE) }

    # Create the plot
    p <- ggplot(dat, aes(x = lon, y = lat)) +
        theme_bw()
    p <- p + theme(plot.title = element_text(size = rel(1.5)))
```

```
    p <- p + geom_point(aes(colour = y))
    ## add in the map
    p <- p + geom_polygon(data = states, aes(x = long, y = lat, group = group),
                          colour = "black", fill = NA)
    ## a 1.3 coordinate ratio is visually appealing
    p <- p + coord_fixed(ratio = 1.3, xlim = xlim, ylim = ylim)
    p <- p + labs(title = paste(mainTitle, "\n", sep=""), x = "", y = "")
    if(zeroiswhite){
        p <- p + scale_colour_gradient2(
            low      =  color_low,
            high     = color_high,
            na.value = color_na,
            limits   = zlim,
            name     = legendTitle
        )
    }
    if(!zeroiswhite){
        p <- p + scale_colour_gradient(
            low      = color_low,
            high     = color_high,
            na.value = color_na,
            limits   = zlim,
            name     = legendTitle
        )
    }
    return(p)
}
```

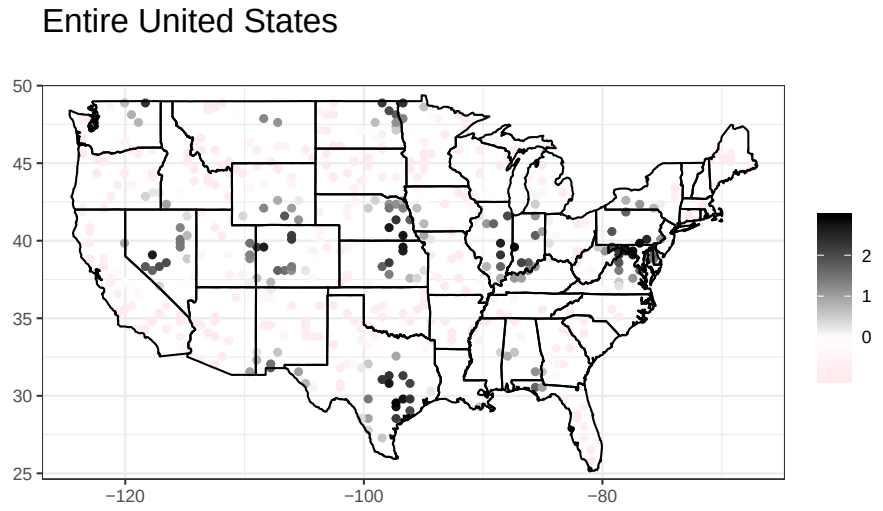
```
## Let's make some plots
dat %>%
    map_points(
        color_low  = "pink",
        color_high = "black",
        mainTitle  = "Entire United States"
    )
```



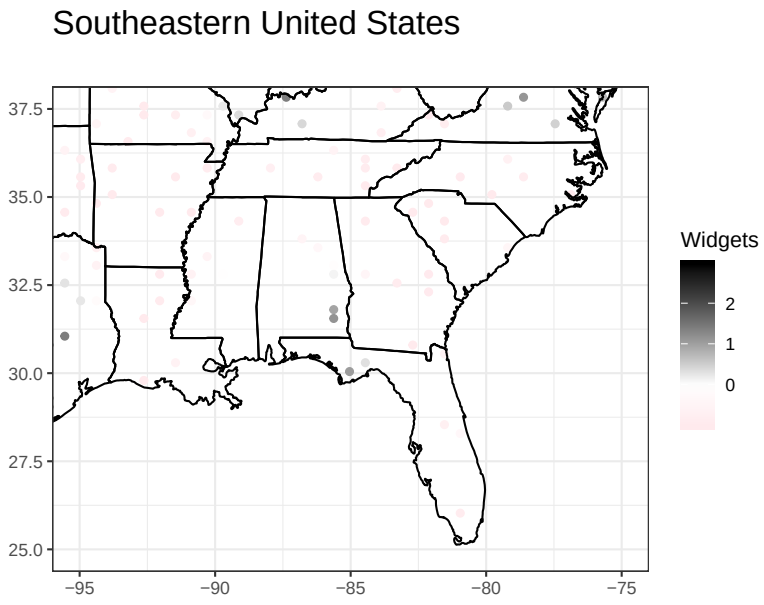

```
## Subset only the US
dat %>%
    subset(inUS) %>%
    map_points(
        color_low  = "pink",
        color_high = "black",
        mainTitle  = "Entire United States"
    )
```




```
## plot only a subset of points
dat %>%
    subset(inUS) %>%
    ## sample 500 points at random
    sample_n(500) %>%
    map_points(
        color_low   = "pink",
        color_high  = "black",
        zeroiswhite = TRUE,
        mainTitle   = "Entire United States"
    )
```




```
## Truncate the southeastern US
dat %>%
    subset(inUS) %>%
    ## sample 500 points at random
    sample_n(500) %>%
    map_points(
        color_low   = "pink",
        color_high  = "black",
        zeroiswhite = TRUE,
        xlim        = c(-95, -75),
        ylim        = c(25, 37.5),
        mainTitle   = "Southeastern United States",
        legendTitle = "Widgets"
    )
```



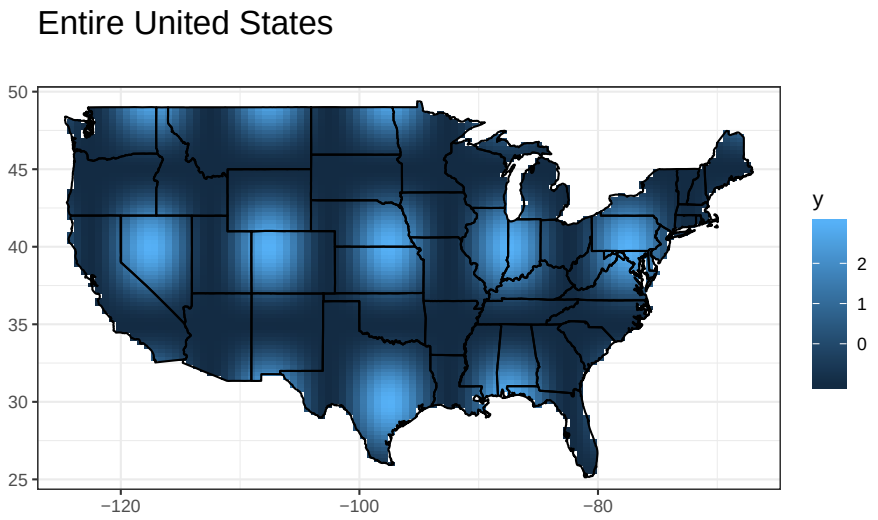
Heatmaps can also be used for plotting. In general, there are two ggplot geoms that are useful for spatial data: *geom_tile* is good for irregularly spaced data, *geom_raster* is best for regularly spaced data as it is faster to process.

```
## Function to generate maps
map_heat <- function (dat,
                      color_low = "white", color_high = "darkred",
                      color_na = gray(0.9), zeroiswhite = FALSE,
                      xlim = NULL, ylim = NULL, zlim = NULL,
                      mainTitle = NULL, legendTitle = "",
                      geom = "raster") {
    library(ggplot2)

    ## check if the data.fram dat contains the correct variables
    if (is.null(dat$lon)) { stop('The data.frame dat must contain a "lon" variable') }
    if (is.null(dat$lat)) { stop('The data.frame dat must contain a "lat" variable') }
    if (is.null(dat$y))   { stop('The data.frame dat must contain a "y" variable') }
    if (!(geom %in% c("raster", "tile"))) { stop('The only options for geom are "raster" or "tile

    # Store the base data of the underlying map
    states <- map_data("state")

    # Set limits for x, y, z if not specified as parameters
    if (is.null(xlim)) { xlim <- range(dat$lon, na.rm = TRUE) }
    if (is.null(ylim)) { ylim <- range(dat$lat, na.rm = TRUE) }
```

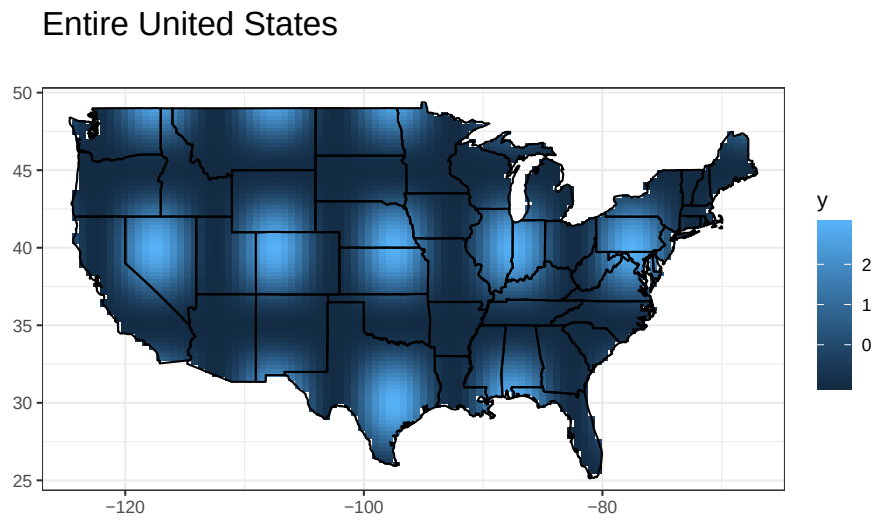
```
    if (is.null(zlim)) { zlim <- range(dat$y, na.rm = TRUE) }

    # Create the plot
    p <- ggplot(dat, aes(x = lon, y = lat)) +
        theme_bw()
    p <- p + theme(plot.title = element_text(size = rel(1.5)))
    if (geom == "raster") {
        p <- p + geom_raster(aes(fill = y))
    }
    if (geom == "tile") {
        p <- p + geom_tile(aes(fill = y))
    }
    ## add in the map
    p <- p + geom_polygon(data = states, aes(x = long, y = lat, group = group),
                          colour = "black", fill = NA)
    ## a 1.3 coordinate ratio is visually appealing
    p <- p + coord_fixed(ratio = 1.3, xlim = xlim, ylim = ylim)
    p <- p + labs(title = paste(mainTitle, "\n", sep=""), x = "", y = "")
    if(zeroiswhite){
        p <- p + scale_colour_gradient2(
            low      =  color_low,
            high     = color_high,
            na.value = color_na,
            limits   = zlim,
            name     = legendTitle
        )
    }
    if(!zeroiswhite){
        p <- p + scale_colour_gradient(
            low      = color_low,
            high     = color_high,
            na.value = color_na,
            limits   = zlim,
            name     = legendTitle
        )
    }
    return(p)
}
```

```
## Subset only the US
dat %>%
    subset(inUS) %>%
    map_heat(
        color_low  = "blue",
        color_high = "yellow",
```

```
        mainTitle  = "Entire United States",
        geom = "raster"
    )
```



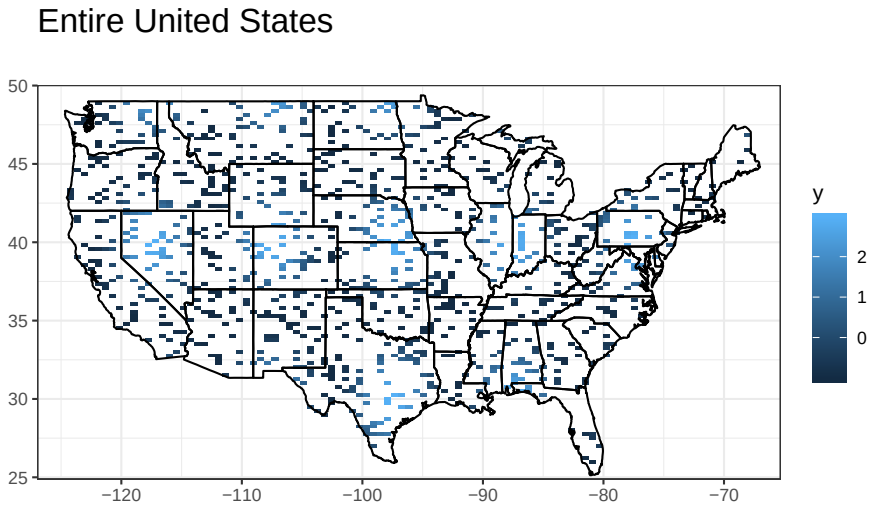

```
## Subset only the US
dat %>%
    subset(inUS) %>%
    map_heat(
        color_low  = "blue",
        color_high = "yellow",
        mainTitle  = "Entire United States",
        geom = "tile"
    )
```



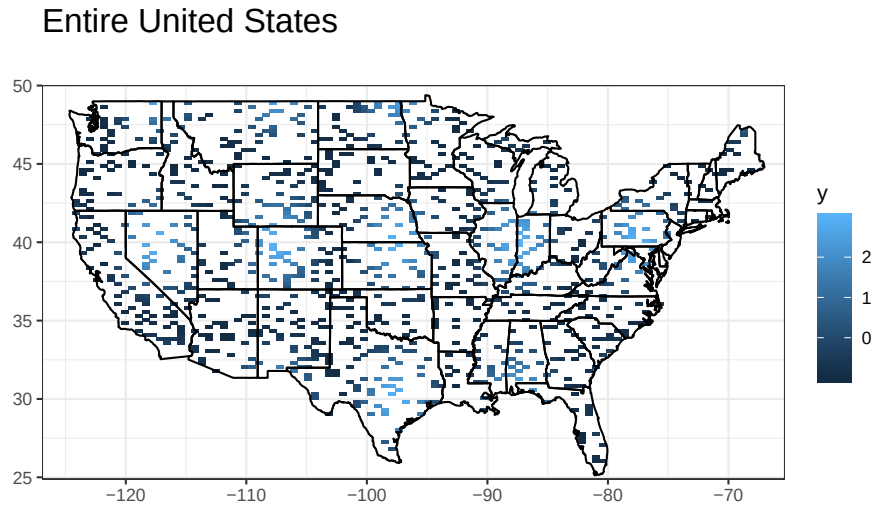

```
## Subsample the data
dat %>%
    subset(inUS) %>%
    sample_n(1000) %>%
    map_heat(
        color_low  = "blue",
        color_high = "green",
        mainTitle  = "Entire United States",
        geom = "raster"
    )
```

```
## Warning in f(...): Raster pixels are placed at uneven horizontal intervals and
## will be shifted. Consider using geom_tile() instead.
```

```
## Warning in f(...): Raster pixels are placed at uneven vertical intervals and
## will be shifted. Consider using geom_tile() instead.
```




```
## Subsample the data
dat %>%
    subset(inUS) %>%
    sample_n(1000) %>%
    map_heat(
        color_low  = "pink",
        color_high = "black",
        mainTitle  = "Entire United States",
        geom = "tile"
    )
```



- Plotting spatial data using google maps

```
## longitude and latitude of approximately the center of Arkansas
arkansas_center <- c(-92.33, 35.00)

library(maps)
library(ggplot2)
library(ggmap)

lon <- arkansas_center[1] + seq(-2, 2, length = 10)
lat <- arkansas_center[2] + seq(-2, 2, length = 10)
s   <- expand.grid(lon, lat)

head(lon)
```

```
## [1] -94.33000 -93.88556 -93.44111 -92.99667 -92.55222 -92.10778
```

```
head(lat)
```

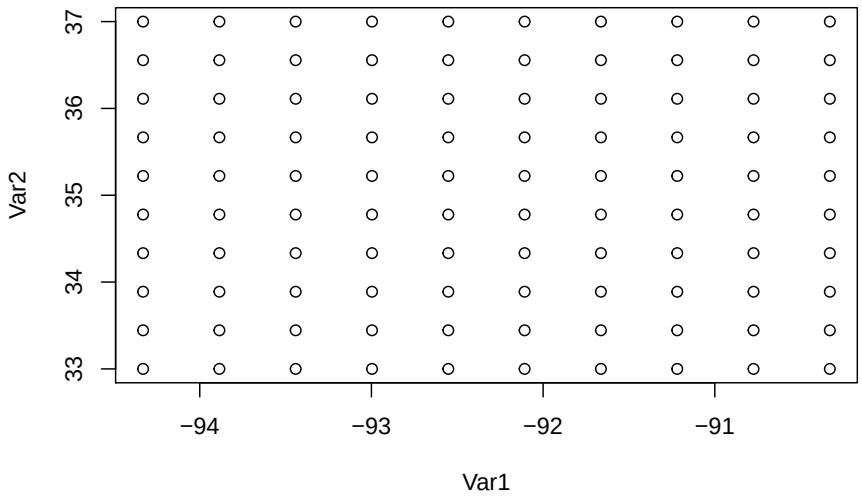
```
## [1] 33.00000 33.44444 33.88889 34.33333 34.77778 35.22222
```

```
str(s)
```

```
## 'data.frame':    100 obs. of  2 variables:
##  $ Var1: num  -94.3 -93.9 -93.4 -93 -92.6 ...
##  $ Var2: num  33 33 33 33 33 33 33 33 33 33 ...
##  - attr(*, "out.attrs")=List of 2
##   ..$ dim     : int  10 10
```

```
##   ..$ dimnames:List of 2
##   .. ..$ Var1: chr  "Var1=-94.33000" "Var1=-93.88556" "Var1=-93.44111" "Var1=-92.99667" ...
##   .. ..$ Var2: chr  "Var2=33.00000" "Var2=33.44444" "Var2=33.88889" "Var2=34.33333" ...
```

```
plot(s)
points(arkansas_center, pch = 19, col = 2)
```



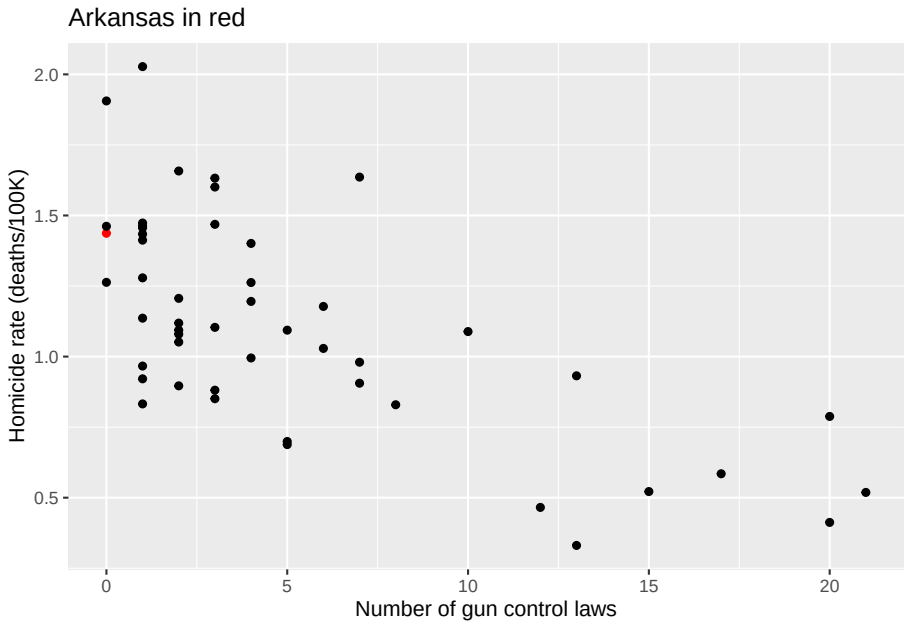

```
dat <- data.frame(lon = lon, lat = lat)
```

Using Google maps requires registration of a key. See https://www.littlemissdata.com/blog/maps for details.

- Plotting areal data The example is from https://www4.stat.ncsu.edu/~reich/SpatialStats/code/Guns.pdf taken from https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(15)01026-0/fulltext

```
## process the guns data
# load(here::here("data", "guns.RData"))
# dat <- data.frame(
#     deaths_2010             = Y,
#     population              = N,
#     deaths_per_10000        = Z[, 1],
#     firearm_quartile        = Z[, 2],
#     unemployment_quartile   = Z[, 3],
#     non_firearm_homocide    = Z[, 4],
#     firearm_export_quartile = Z[, 5],
#     numlaws                 = apply(X, 1, sum),
```

```
#     region                  = region
# )
# save(dat, file = here::here("guns_processed.RData"))
load(here::here("guns_processed.RData"))

## mutate a death rate
dat <- dat %>%
    mutate(rate = 10000 * deaths_2010 / population)

# names(Y)[1:5]
# region  <- tolower(names(Y))
# region[1:5]
# rate    <- 10000*Y/N
# numlaws <- rowSums(X)
# crime   <- data.frame(Y=Y,N=N,rate=rate,X=X,numlaws,region=region)
#
# crime[1:5,]

dat %>%
    ggplot(aes(x = numlaws, y = rate, color = region == "arkansas")) +
    geom_point() +
    scale_color_manual(values = c("black", "red")) +
    xlab("Number of gun control laws") +
    ylab("Homicide rate (deaths/100K)") +
    ggtitle("Arkansas in red") +
    theme(legend.position = "none")
```




```
lm(rate ~ numlaws, data = dat) %>% summary()
```

```
##
## Call:
## lm(formula = rate ~ numlaws, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.46715 -0.16720 -0.02576  0.16171  0.72809
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.344693   0.055145  24.385  < 2e-16 ***
## numlaws     -0.045276   0.007302  -6.201 1.24e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2867 on 48 degrees of freedom
## Multiple R-squared:  0.4448, Adjusted R-squared:  0.4332
## F-statistic: 38.45 on 1 and 48 DF,  p-value: 1.236e-07
```

```
us <- map_data("state")
head(us)
```

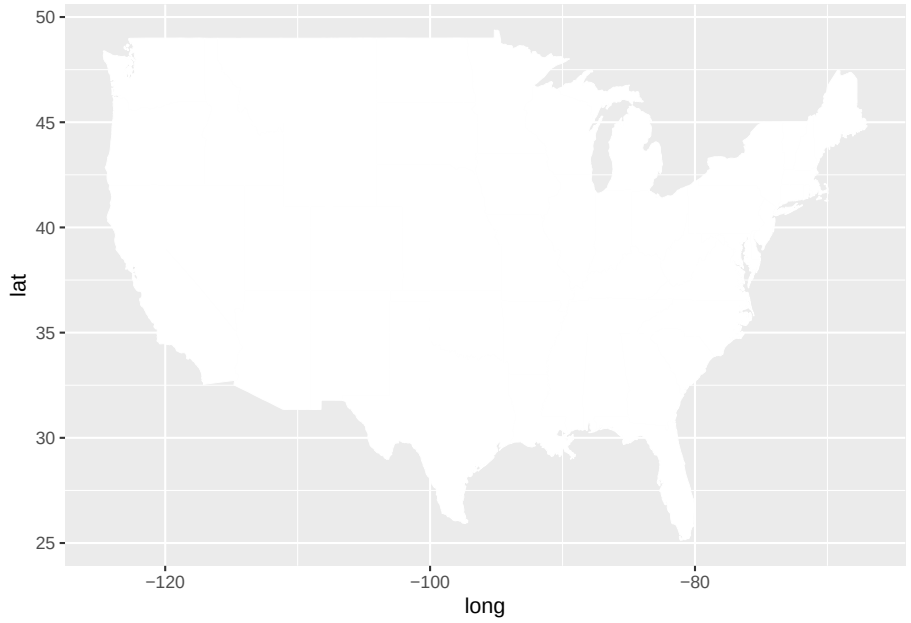
```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
```

```
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

```
gg <- ggplot()
gg <- gg + geom_map(data = us, map = us,
                    aes(x = long, y = lat, map_id = region),
                    fill = "#ffffff", color = "#ffffff", size = 0.15)
```

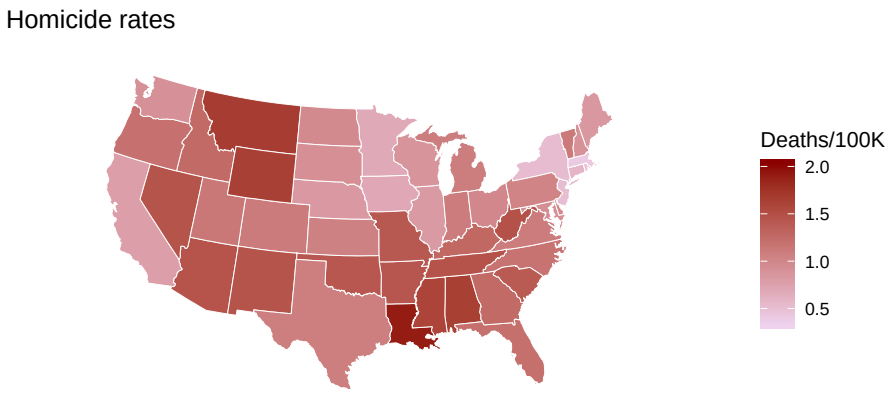
```
## Warning: Ignoring unknown aesthetics: x, y
```

```
gg
```




```
gg <- gg + geom_map(
    data = dat,
    map = us,
    aes(fill = rate, map_id = region),
    color = "#ffffff", size = 0.15
)

gg <- gg + scale_fill_continuous(
    low  = 'thistle2',
    high = 'darkred',
    guide= 'colorbar',
    name = "Deaths/100K"
```

```
)
gg <- gg + labs(x = NULL, y = NULL, title = "Homicide rates")
gg <- gg + coord_map("albers", lat0 = 39, lat1 = 45)
gg <- gg + theme(panel.border = element_blank())
gg <- gg + theme(panel.background = element_blank())
gg <- gg + theme(axis.ticks = element_blank())
gg <- gg + theme(axis.text = element_blank())
gg
```



The map looks right according to

http://www.deathpenaltyinfo.org/murder-rates-nationally-and-state#MRord

2.4.3 In Class Activity:

From Lab 2.1 on the textbook site

```
## Wikle, C. K., Zammit-Mangion, A., and Cressie, N. (2019),
## Spatio-Temporal Statistics with R, Boca Raton, FL: Chapman & Hall/CRC
## Copyright (c) 2019 Wikle, Zammit-Mangion, Cressie
##
## This program is free software; you can redistribute it and/or
## modify it under the terms of the GNU General Public License
## as published by the Free Software Foundation; either version 2
## of the License, or (at your option) any later version.
##
```

```
## This program is distributed in the hope that it will be useful,
## but WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.

library("dplyr")
library("tidyr")
library("STRbook")

## ------------------------------------------------------------------------
locs <- read.table(system.file("extdata", "Stationinfo.dat",
                               package = "STRbook"),
                   col.names = c("id", "lat", "lon"))
Times <- read.table(system.file("extdata", "Times_1990.dat",
                                package = "STRbook"),
                    col.names = c("julian", "year", "month", "day"))
Tmax <- read.table(system.file("extdata", "Tmax_1990.dat",
                               package = "STRbook"))

## ------------------------------------------------------------------------
names(Tmax) <- locs$id

## ------------------------------------------------------------------------
Tmax <- cbind(Times, Tmax)
head(names(Tmax), 10)

## ------------------------------------------------------------------------
Tmax_long <- gather(Tmax, id, z, -julian, -year, -month, -day)
head(Tmax_long)

## ------------------------------------------------------------------------
Tmax_long$id <- as.integer(Tmax_long$id)

## ------------------------------------------------------------
nrow(Tmax_long)
Tmax_long <- filter(Tmax_long, !(z <= -9998))
nrow(Tmax_long)

## ------------------------------------------------------------------------
Tmax_long <- mutate(Tmax_long, proc = "Tmax")
head(Tmax_long)

## ------------------------------------------------------------------------
data(Tmin_long, package = "STRbook")
data(TDP_long, package = "STRbook")
```

```
data(Precip_long, package = "STRbook")

## -------------------------------------------------------------------------
NOAA_df_1990 <- rbind(Tmax_long, Tmin_long, TDP_long, Precip_long)

## -------------------------------------------------------------------------
summ <- group_by(NOAA_df_1990, year, proc) %>%  # groupings
    summarise(mean_proc = mean(z))           # operation

## -------------------------------------------------------------------------
NOAA_precip <- filter(NOAA_df_1990, proc == "Precip" & month == 6)
summ <- group_by(NOAA_precip, year, id) %>%
    summarise(days_no_precip = sum(z == 0))
head(summ)

## -------------------------------------------------------------------------
median(summ$days_no_precip)

## -----------------------------------------------------------------
grps <- group_by(NOAA_precip, year, id)
summ <- summarise(grps, days_no_precip = sum(z == 0))

## -------------------------------------------------------------------------
NOAA_df_sorted <- arrange(NOAA_df_1990, julian, id)

## -------------------------------------------------------------------------
df1 <- select(NOAA_df_1990, julian, z)
df2 <- select(NOAA_df_1990, -julian)

## -------------------------------------------------------------------------
NOAA_df_1990 <- left_join(NOAA_df_1990, locs, by = "id")

## -------------------------------------------------------------------------
Tmax_long_sel <- select(Tmax_long, julian, id, z)
Tmax_wide <- spread(Tmax_long_sel, id, z)
dim(Tmax_wide)

## -------------------------------------------------------------------------
M <- select(Tmax_wide, -julian) %>% as.matrix()

## -----------------------------------------------------------------
library("sp")
library("spacetime")

## -------------------------------------------------------------------------
```

```
NOAA_df_1990$date <- with(NOAA_df_1990,
                          paste(year, month, day, sep = "-"))
head(NOAA_df_1990$date, 4)   # show first four elements

## ------------------------------------------------------------------------
NOAA_df_1990$date <- as.Date(NOAA_df_1990$date)
class(NOAA_df_1990$date)

## ------------------------------------------------------------------------
Tmax_long2 <- filter(NOAA_df_1990, proc == "Tmax")
STObj <- stConstruct(x = Tmax_long2,           # data set
                     space = c("lon", "lat"),  # spatial fields
                     time = "date")            # time field
class(STObj)

## ------------------------------------------------------------------------
spat_part <- SpatialPoints(coords = Tmax_long2[, c("lon", "lat")])
temp_part <- Tmax_long2$date
STObj2 <- STIDF(sp = spat_part,
                time = temp_part,
                data = select(Tmax_long2, -date, -lon, -lat))
class(STObj2)

## ------------------------------------------------------------------------
spat_part <- SpatialPoints(coords = locs[, c("lon", "lat")])
temp_part <- with(Times,
                  paste(year, month, day, sep = "-"))
temp_part <- as.Date(temp_part)

## ------------------------------------------------------------------------
Tmax_long3 <- gather(Tmax, id, z, -julian, -year, -month, -day)

## ------------------------------------------------------------------------
Tmax_long3$id <- as.integer(Tmax_long3$id)
Tmax_long3 <- arrange(Tmax_long3,julian,id)

## ------------------------------------------------------------------------
all(unique(Tmax_long3$id) == locs$id)

## ------------------------------------------------------------------------
STObj3 <- STFDF(sp = spat_part,
                time = temp_part,
                data = Tmax_long3)
class(STObj3)
```

```
## ------------------------------------------------------------------------
proj4string(STObj3) <- CRS("+proj=longlat +ellps=WGS84")

## ------------------------------------------------------------------------
STObj3$z[STObj3$z == -9999] <- NA
```

Chapter 3

Day 3

Chapter 4

Day 4

Chapter 5

Day 5

Chapter 6

Day 6

Chapter 7

Day 5

Chapter 8

Day 6

Chapter 9

Day 4

Chapter 10

Day 10

Chapter 11

Day 11

Chapter 12

Day 12

Chapter 13

Day 13

Chapter 14

Day 14

Chapter 15

Day 15

Chapter 16

Day 16

Chapter 17

Day 17

Chapter 18

Day 18

Chapter 19

Day 19

Chapter 20

Day 20

Chapter 21

Day 21

Chapter 22

Day 22

Chapter 23

Day 23

Chapter 24

Day 24

Chapter 25

Day 25

Chapter 26

Day 26

Chapter 27

Day 27

Chapter 28

Day 28

Chapter 29

Day 29

Chapter 30

Day 30

Chapter 31

Day 31

Chapter 32

Day 32

Chapter 33

Day 33

Chapter 34

Day 34

Chapter 35

Day 35

Chapter 36

Day 36

Chapter 37

Day 37

Chapter 38

Day 38

Chapter 39

Day 39

Chapter 40

Day 40

Chapter 41

Day 41

Chapter 42

Day 42

Chapter 43

Day 43

Bibliography

Berliner, L. M. (1996). Hierarchical Bayesian time series models. In *Maximum Entropy and Bayesian Methods*, pages 15–22. Springer.