Making Maps with R - Part II
Mapping Locations

Xuemao Zhang
Department of Mathematics
East Stroudsburg University

June 26, 2019

Outline

- Overview
- Approximate Centroids
- Points
- Symbols
- Lines
- Animated Map

Overview

- Region boundaries essentially serve as your backdrop.
- For point-based data, you draw the map, and then you add things on top.
- In this section, you learn how to add dots, lines, and symbols.
- We will use an **airports** data frame with eight variables from the package `nycflights13`.

```
install.packages("nycflights13");
```

```
library(nycflights13);
head(airports);
```

```
## # A tibble: 6 x 8
##   faa   name                          lat   lon   alt    tz dst   tz
##   <chr> <chr>                       <dbl> <dbl> <int> <dbl> <chr> <c
## 1 04G   Lansdowne Airport            41.1 -80.6  1044    -5 A     Am
## 2 06A   Moton Field Municipal A~     32.5 -85.7   264    -6 A     Am
## 3 06C   Schaumburg Regional         42.0 -88.1   801    -6 A     Am
## 4 06N   Randall Airport              41.4 -74.4   523    -5 A     Am
## 5 09J   Jekyll Island Airport        31.1 -81.4    11    -5 A     Am
## 6 0A9   Elizabethton Municipal ~     36.4 -82.2  1593    -5 A     Am
```

Approximate Centroids

- In mathematics, the centroid of a plane figure is the arithmetic mean position of all the points in the figure.

```
library(ggplot2);
usmap = ggplot2::map_data("state");
str(usmap);
```

```
## 'data.frame':    15537 obs. of  6 variables:
##  $ long      : num  -87.5 -87.5 -87.5 -87.5 -87.6 ...
##  $ lat       : num  30.4 30.4 30.4 30.3 30.3 ...
##  $ group     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ order     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ region    : chr  "alabama" "alabama" "alabama" "alabama" ...
##  $ subregion: chr  NA NA NA NA ...
```

Approximate Centroids

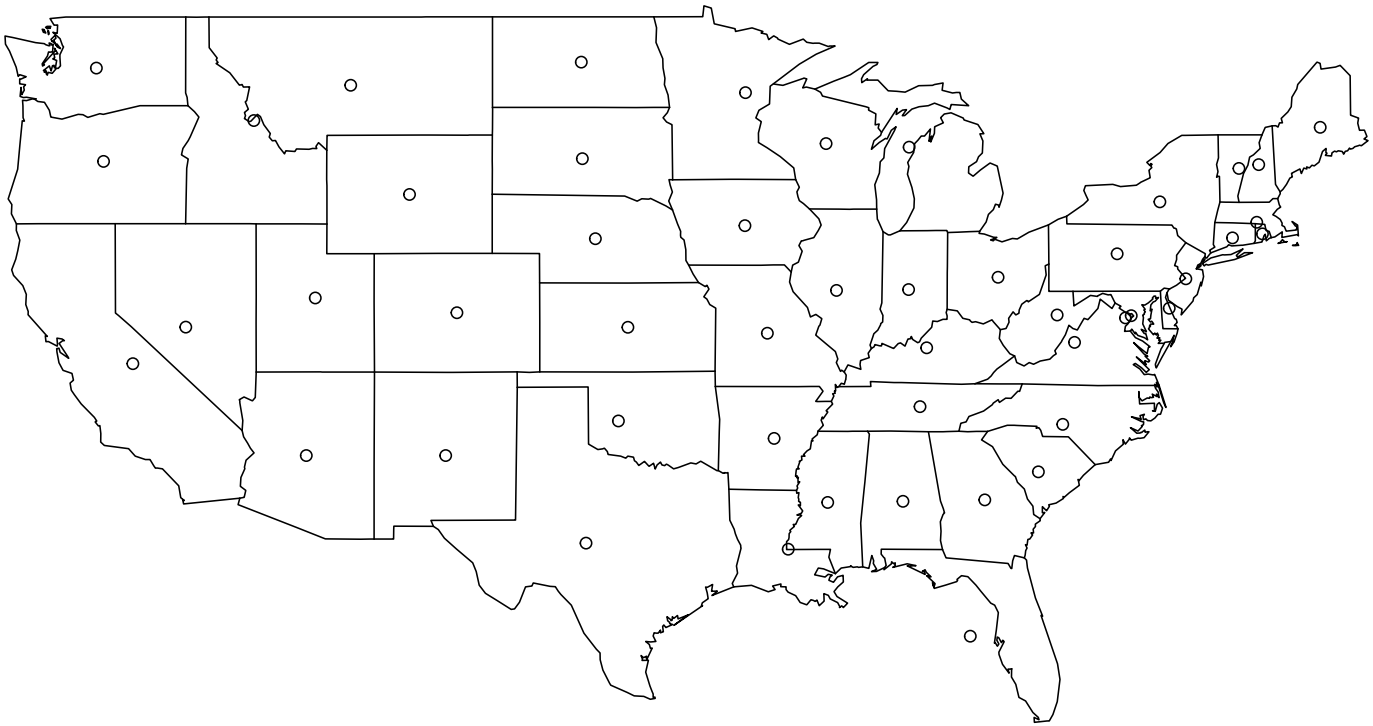
```
library(dplyr);
state_centroids = dplyr::summarize(group_by(usmap, region),
          x = mean(range(long)), y = mean(range(lat)));
names(state_centroids)[1] = "state";
head(state_centroids);
```

```
## # A tibble: 6 x 3
##   state            x     y
##   <chr>        <dbl> <dbl>
## 1 alabama      -86.7  32.6
## 2 arizona     -112.   34.2
## 3 arkansas     -92.1  34.8
## 4 california  -119.   37.3
## 5 colorado    -106.   39.0
## 6 connecticut  -72.8  41.5
```

Points

- Adding points to a map drawn by the `maps::map` function with the default projection can be added using the function `points()`, treating longitude as your x-coordinate and latitude as your y-coordinate.

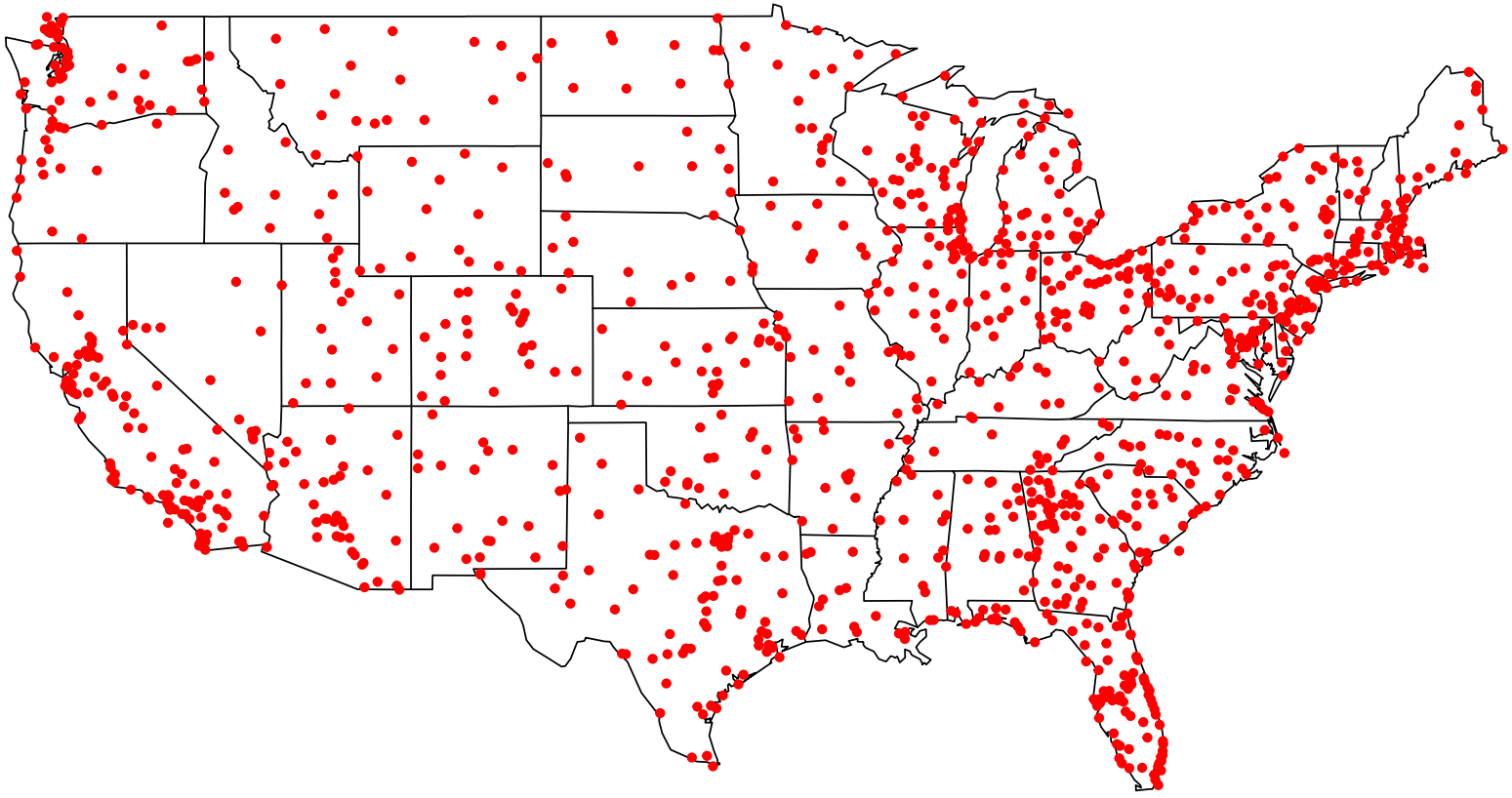
```
library(maps); map("state");
with(state_centroids, points(x, y)); # with() function evaluates
```



Points

- Example: Plot all airports in the **airports** data.

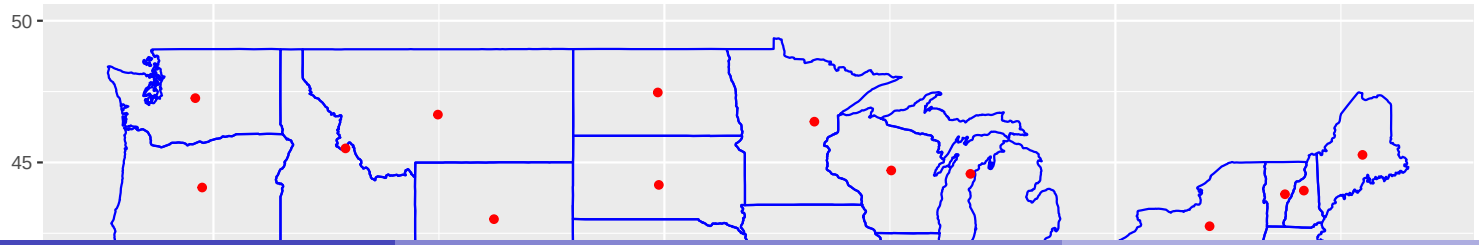
```
map("state");
with(airports, points(lon, lat,pch=20,col='red'));
```



Points

- When using other projections with map, any points or lines added to the plot need to have their coordinates projected as well.
- Using `coord_quickmap()` in ggplot2 seems to be comparable to the `map` default.
- `ggplot2::coord_map()` can be used to specify projections.
- ggplot2::geom_polygon() can be used to draw the boundaries.

```
library(ggplot2);
ggplot(usmap)+
  geom_polygon(aes(long, lat, group = group), fill = NA,
               color = "blue")+
  geom_point(aes(x, y), data = state_centroids, color = "red")+
  coord_quickmap();
```

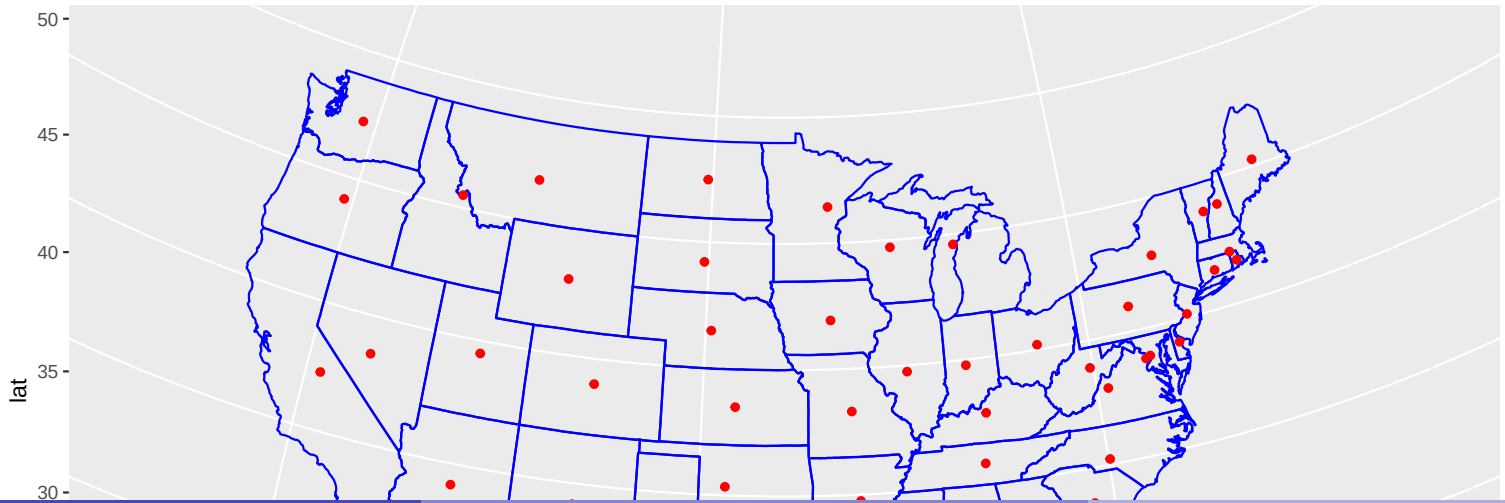


Points

- An example of projected map.

```
library(ggplot2);
ggplot(usmap)+
  geom_polygon(aes(long, lat, group = group), fill = NA,
               color = "blue")+
  geom_point(aes(x, y), data = state_centroids, color = "red")+
  coord_quickmap()+
  coord_map("bonne", parameters=45);
```

```
## Coordinate system already present. Adding new coordinate system,
```

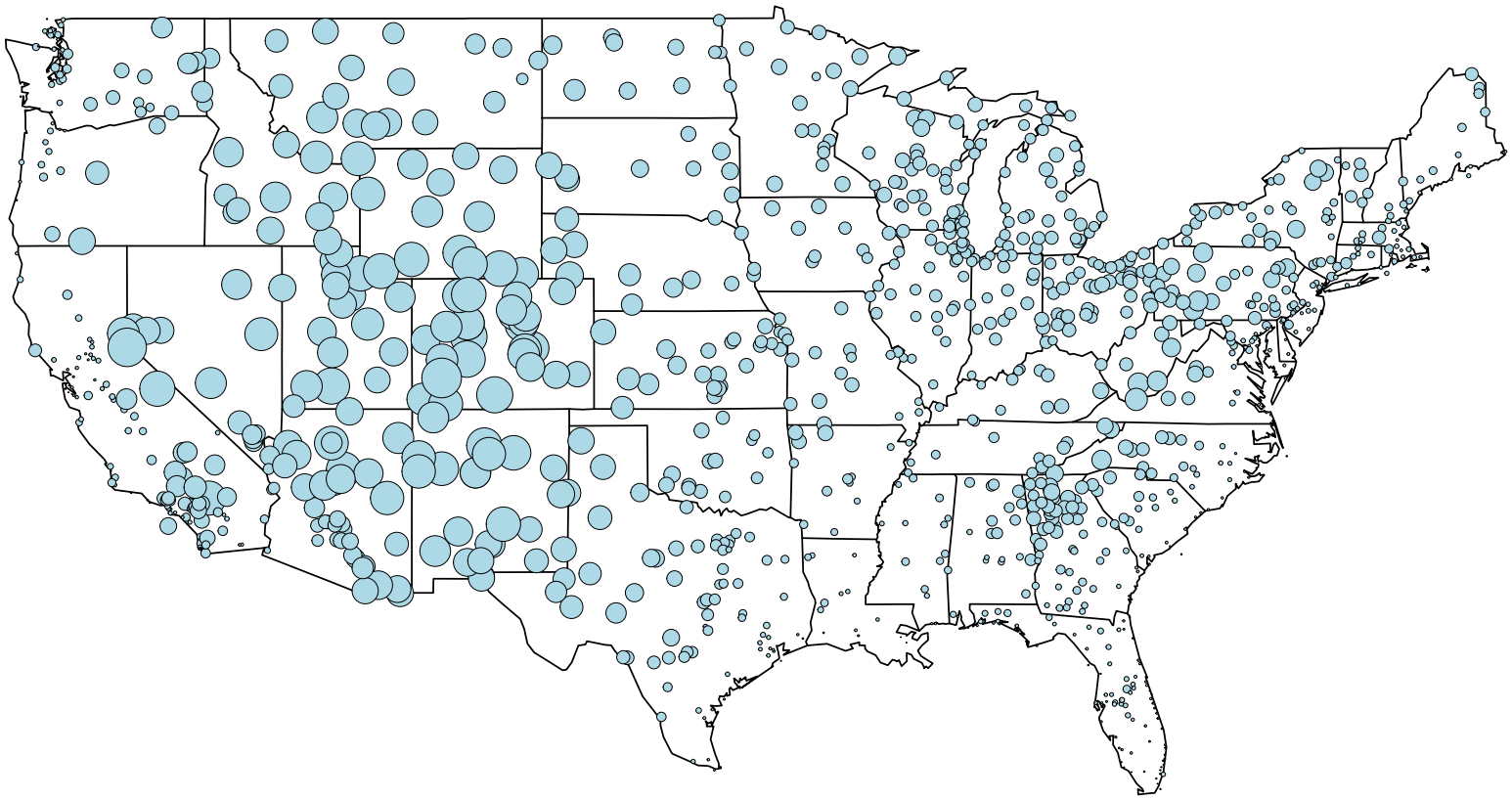


Symbols

- Points only represent location.
- Symbols (scaled shapes) both represent location and a second metric that corresponds to the location.
- The function `symbols()` can be used to draw symbols.
- In the following example,
  - Set `add` to `TRUE`, so that the circles are added to the current map instead of drawing a new plot.
  - Set `inches` to `FALSE` so that you can size circles by the current coordinate system instead of by inches.
  - Set `bg` to whatever `color` you want.
  - `circles` is a vector giving the radii of the circles.
  - Circle size is set to the absolute value of `alt` (altitude) of each airport.
  - The 0.008 multiplier sizes all the circles to fit how you want on the screen.

Symbols

```
map("state");
with(airports, symbols(lon, lat, circles=.008*sqrt(abs(alt)),
     add=TRUE,inches=FALSE,bg="lightblue",lwd=0.5));
```



Lines

- Lines on a map can be used to show roads/running routes.
- The `lines()` function does the drawing when you feed it coordinates.
- We encode a line with multiple coordinates in order instead of a single coordinate. We go from point A to point B to point C, and so on.
- Maybe you want to draw a line from LAX to ABE. Subset the two locations:

```
lax = airports[airports$faa == "LAX",];
abe = airports[airports$faa == "ABE",];
```

Lines

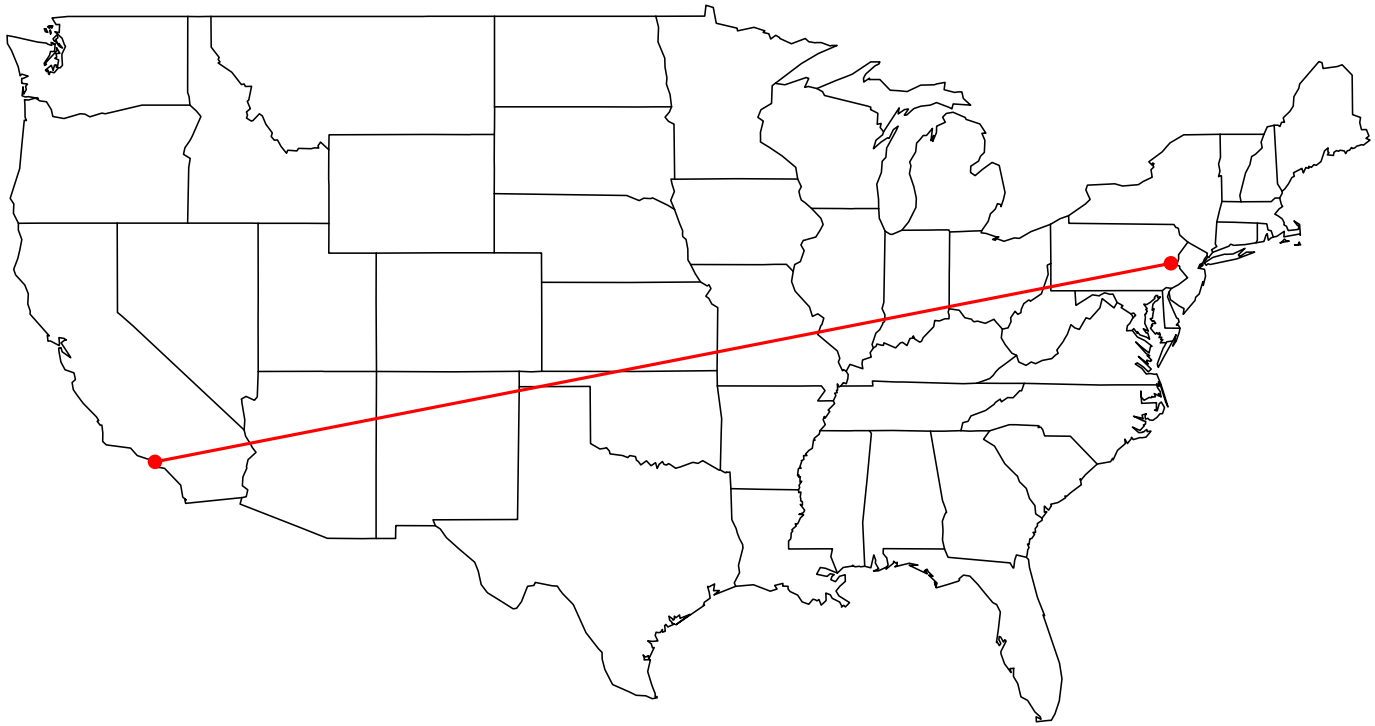
- Draw the map and the two points.

```
map("state");
points(c(lax$lon, abe$lon), c(lax$lat, abe$lat),
       col="red", lwd=2, pch=19);
```

Lines

- Do the same with lines():

```
map("state");
points(c(lax$lon, abe$lon), c(lax$lat, abe$lat), col="red",
       lwd=2, pch=19);
lines(c(lax$lon, abe$lon), c(lax$lat, abe$lat), col="red", lwd=2);
```



Animated Map

- Time series maps and small multiples can go a long way, animation can make your data feel more real and relatable.
- We study the growth of Wal-Mart stores from 1962 to 2010.

```
#load the data first
walmarts=read.csv("F:/DataCamp/data/walmarts_geocoded.csv",
                  sep=",", header=TRUE);
head(walmarts);
```

```
##   store_num year month day      lat       lng   store
## 1         1 1962     7   1 36.33445 -94.17890 walmart
## 2         2 1964     8   1 36.25059 -93.11949 walmart
## 3         4 1965     8   1 36.18320 -94.51260 walmart
## 4         7 1967    10   1 34.83613 -92.23114 walmart
## 5         8 1967    10   1 35.16881 -92.72411 walmart
## 6         9 1968     3   1 36.89540 -89.59512 walmart
```

Animated Map

- We check the years availablein the data.

```
Years=sort(unique(walmarts$year));
print(Years);
```

```
##  [1] 1962 1964 1965 1967 1968 1969 1970 1971 1972 1973 1974 1975
## [15] 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
## [29] 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003
## [43] 2006 2007 2008 2009 2010
```

Animated Map

- To plot the distribution of Wal-Mart stores for each year, we define a function of year.

```
PlotWalmarts= function(x)
{
walmarts_year=walmarts[which(walmarts$year<=x),];
#subset data in the year <= x
map("state");    # Draw map
title(paste("Distribution of Wal-Mart stores, Year",x));
with(walmarts_year, points(x=lng,y=lat,
          pch=21,col='blue',bg="blue",cex=0.3));
}
```

Animated Map

```
library(magick);
img=image_graph(width=600,height=400,res=96);
for( i in Years)  PlotWalmarts(i);
dev.off();
```

```
## pdf
##   2
```

```
LocationsAnimation = image_animate(img, fps = 2);
#show(LocationsAnimation);
image_write(LocationsAnimation, "LocationsAnimation.gif");
```

- You can open the animated map LocationsAnimation.gif in any browser.

Questions?