Introduction to ggplot2

Manuel Gijón Agudo

27 de septiembre, 2018

Contents

This is just an introduction to the syntax an idiosyncrasy of ggplot2 package. We are going to use a simple scatterplot and a preload dataset.

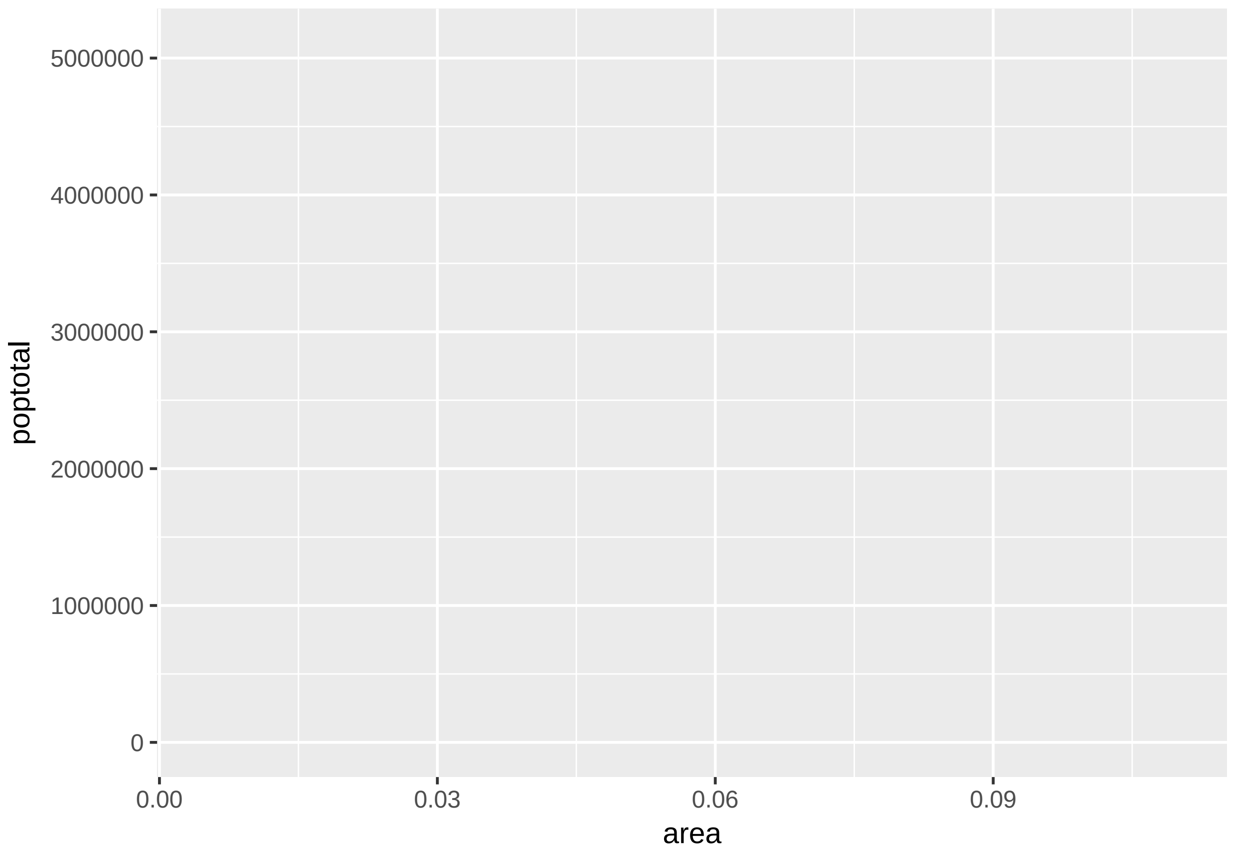
First steps

The first step is to load the library or install it if it si not.

```
#install.packages("ggplot")  # uncommented in case it is not installed
library(ggplot2)
```

First we are going to initialize based on a preloaded dataset.

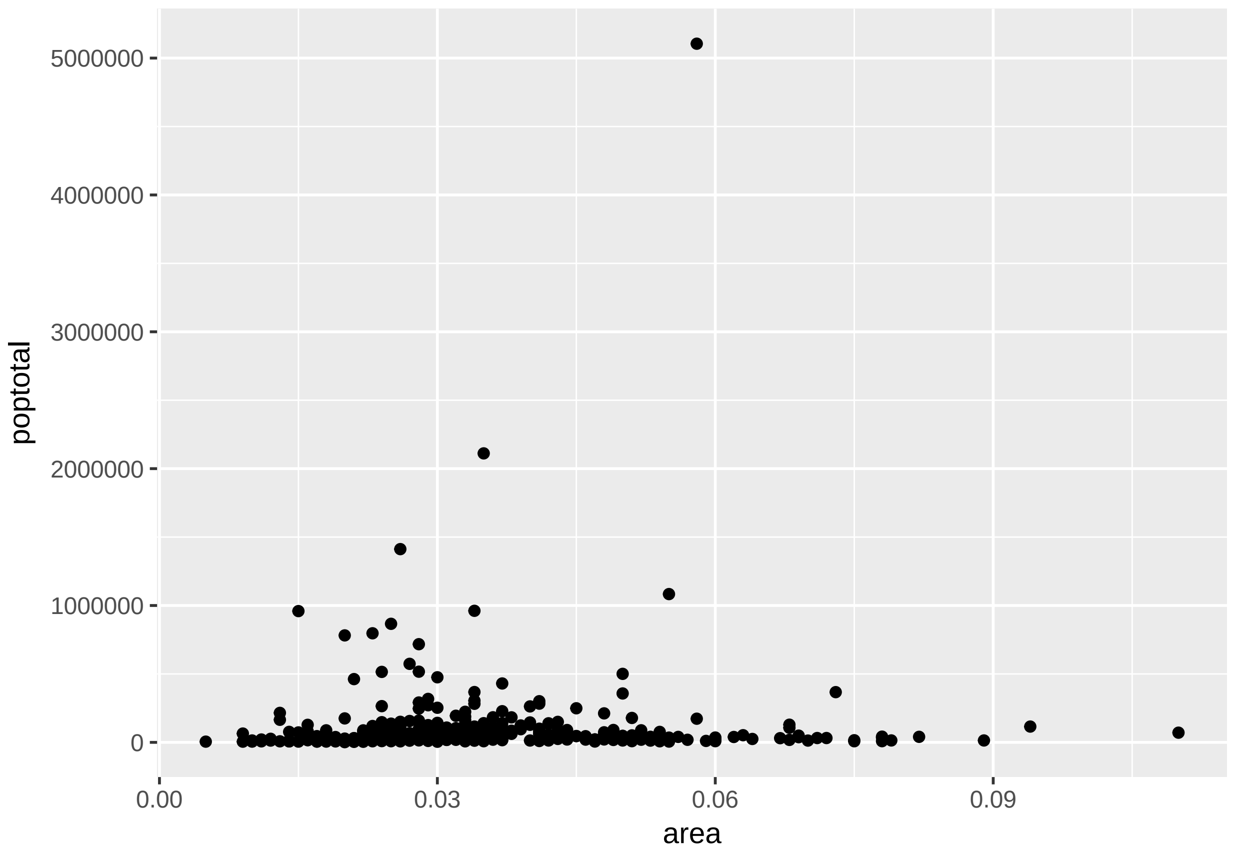
```
# Setup
options(scipen=999)  # turn off scientific notation like 1e+06
data("midwest", package = "ggplot2")  # load the data

# Init ggplot
ggplot(midwest, aes(x = area, y = poptotal))  # area and poptotal are columns in 'midwest'```
```



Note that we are using the function `aes` to specify the axis to ggplot.

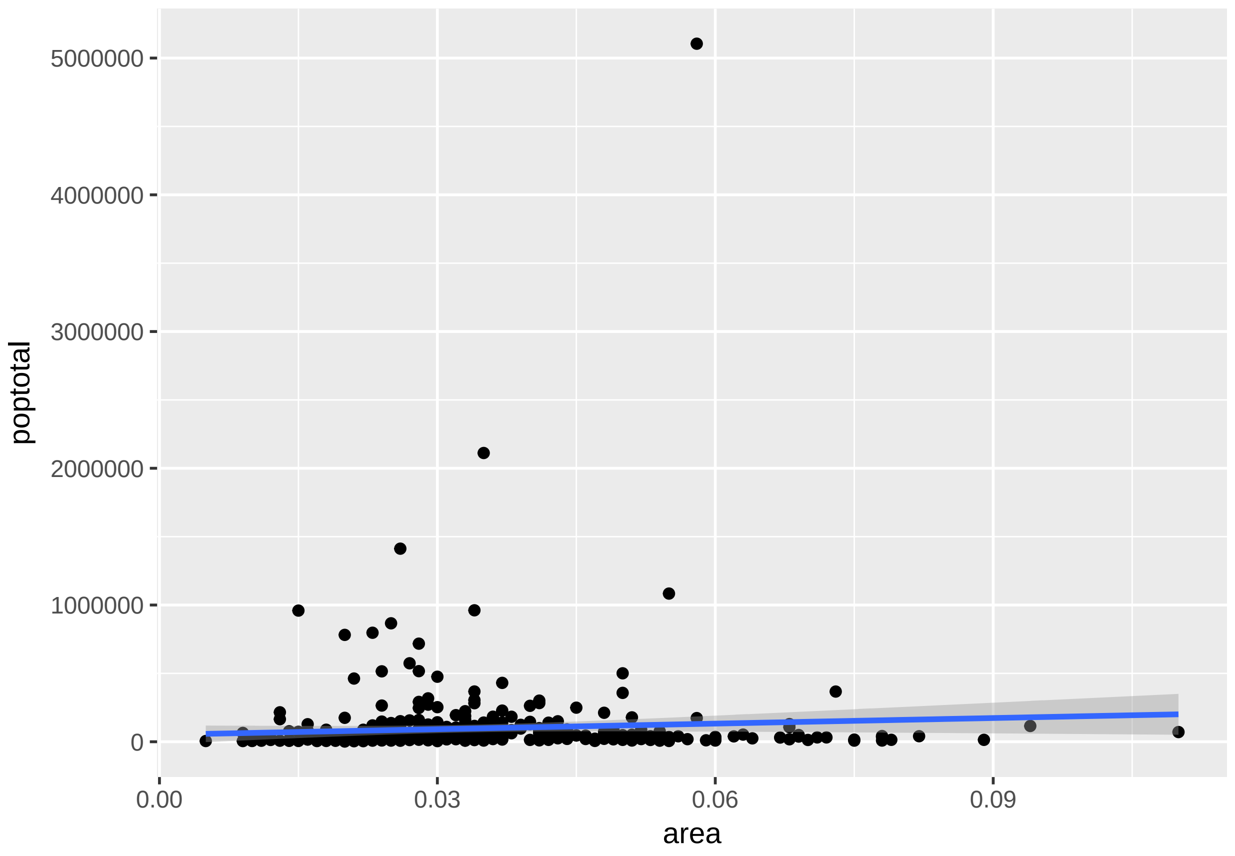
Simple scatterplot

We'll use the function `geom_point()` to add the points to the canvas.



Now we are saving the graphic like an object in the envairoment of variables in R. We call it `g`, so to plot it we need to use the function `plot` aplied to the object.

```
g <- ggplot(midwest, aes(x = area, y = poptotal)) + geom_point() + geom_smooth(method = "lm")
plot(g)
```



Note: the function `geom_smooth` is used here with the paramether and the values `method = "lm"` to plot a linear regresion that includes the confidence intervals.

Adjusting axis

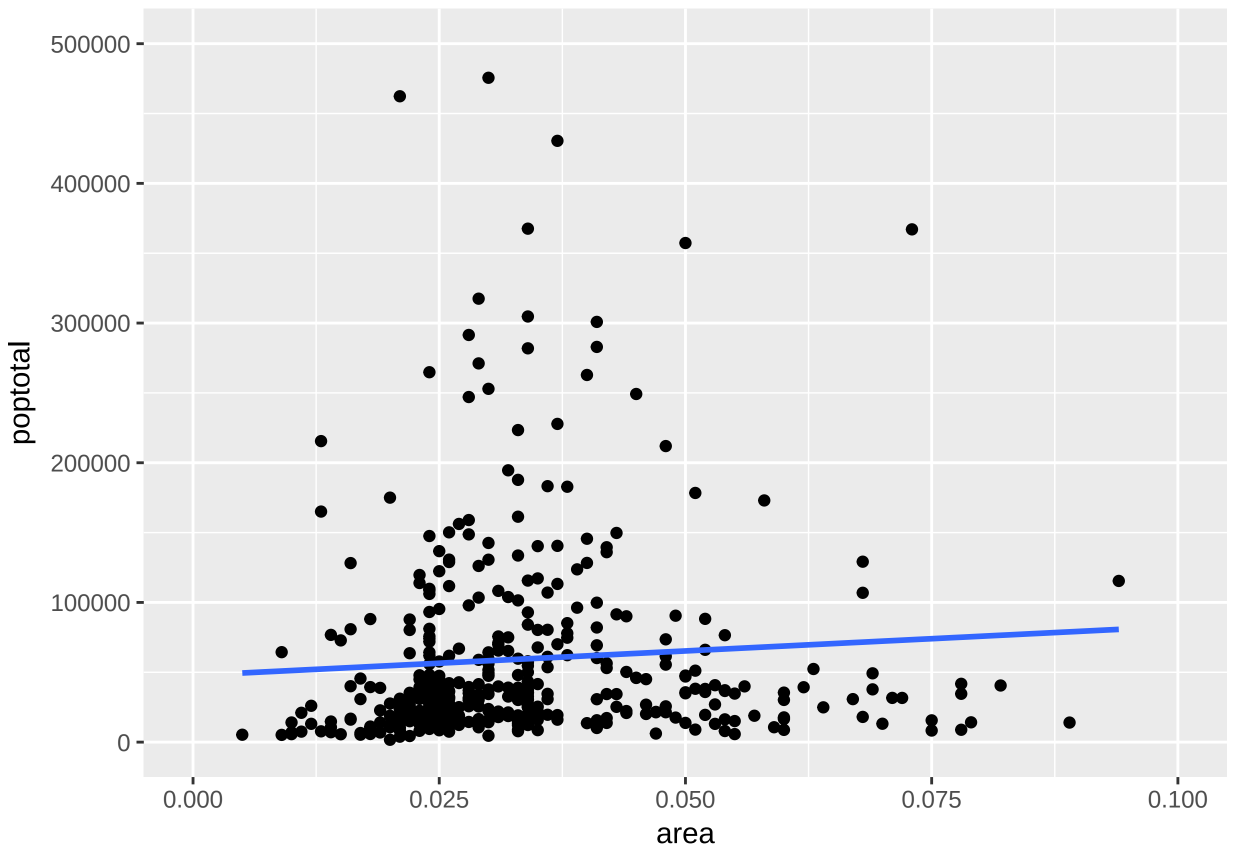
Deleting points outside the range

In this case we delete the point by fixing a range, the points outside will be remove.

```
g <- ggplot(midwest, aes(x = area, y = poptotal)) + geom_point() + geom_smooth(method = "lm", se=FALSE)

# Delete the points outside the limits
g + xlim(c(0, 0.1)) + ylim(c(0, 500000))   # deletes points
```

```
## Warning: Removed 15 rows containing non-finite values (stat_smooth).
```

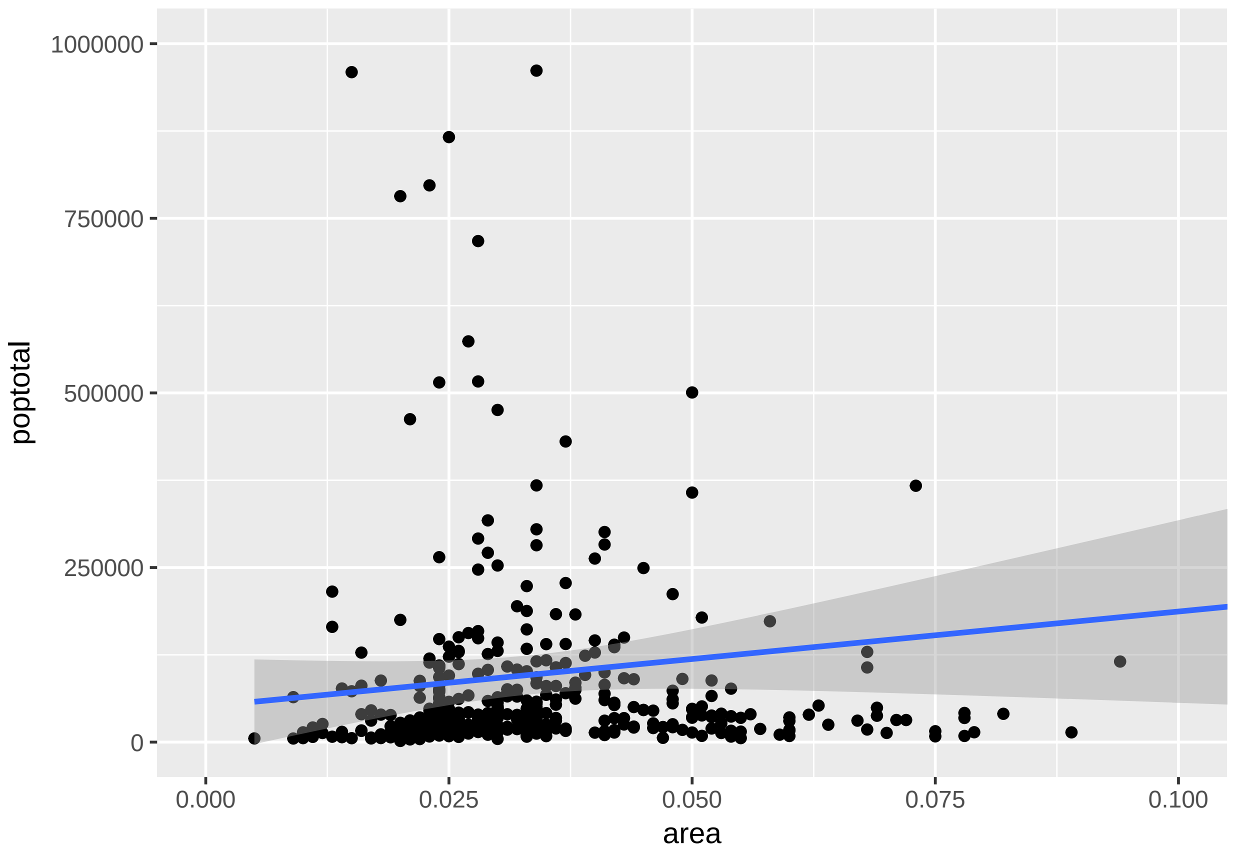
```
## Warning: Removed 15 rows containing missing values (geom_point).
```



Notes: * Be aware that the parameter `se` with the value `FALSE` erase the confidence interval. Realise also that the default value is `TRUE`. * See that changing the total amount of points the regresion line change too.

Zooming in

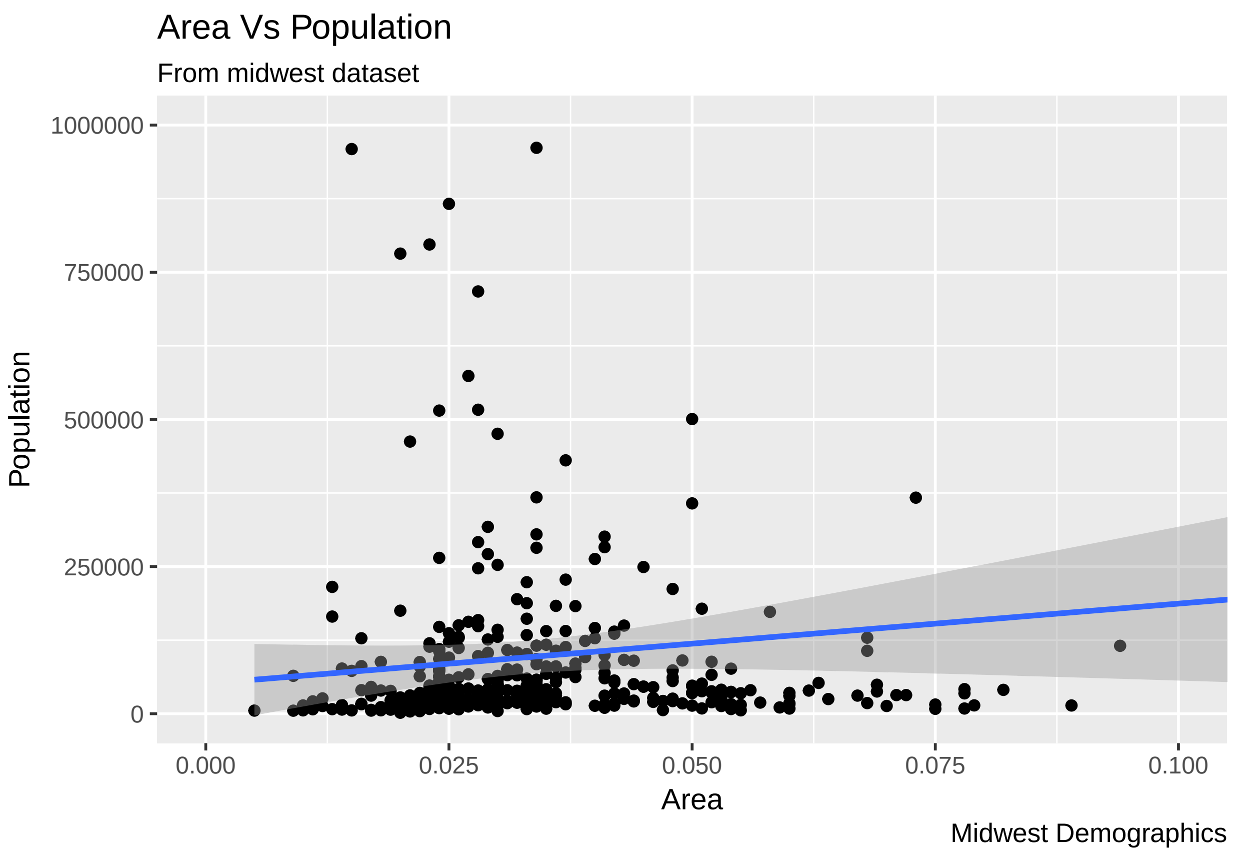
```
g <- ggplot(midwest, aes(x = area, y = poptotal)) + geom_point() + geom_smooth(method = "lm")

g1 <- g + coord_cartesian(xlim = c(0,0.1), ylim = c(0, 1000000))
plot(g1)
```



Changing titles and labels

This section consists just in present by an example the parameters of the function to manipulate.

```
ggplot(midwest, aes(x = area, y = poptotal)) +
      geom_point() +
      geom_smooth(method = "lm") +
      coord_cartesian(xlim = c(0,0.1), ylim = c(0, 1000000)) +
      labs(title = "Area Vs Population",
           subtitle = "From midwest dataset",
           y = "Population",
           x = "Area",
           caption = "Midwest Demographics")
```

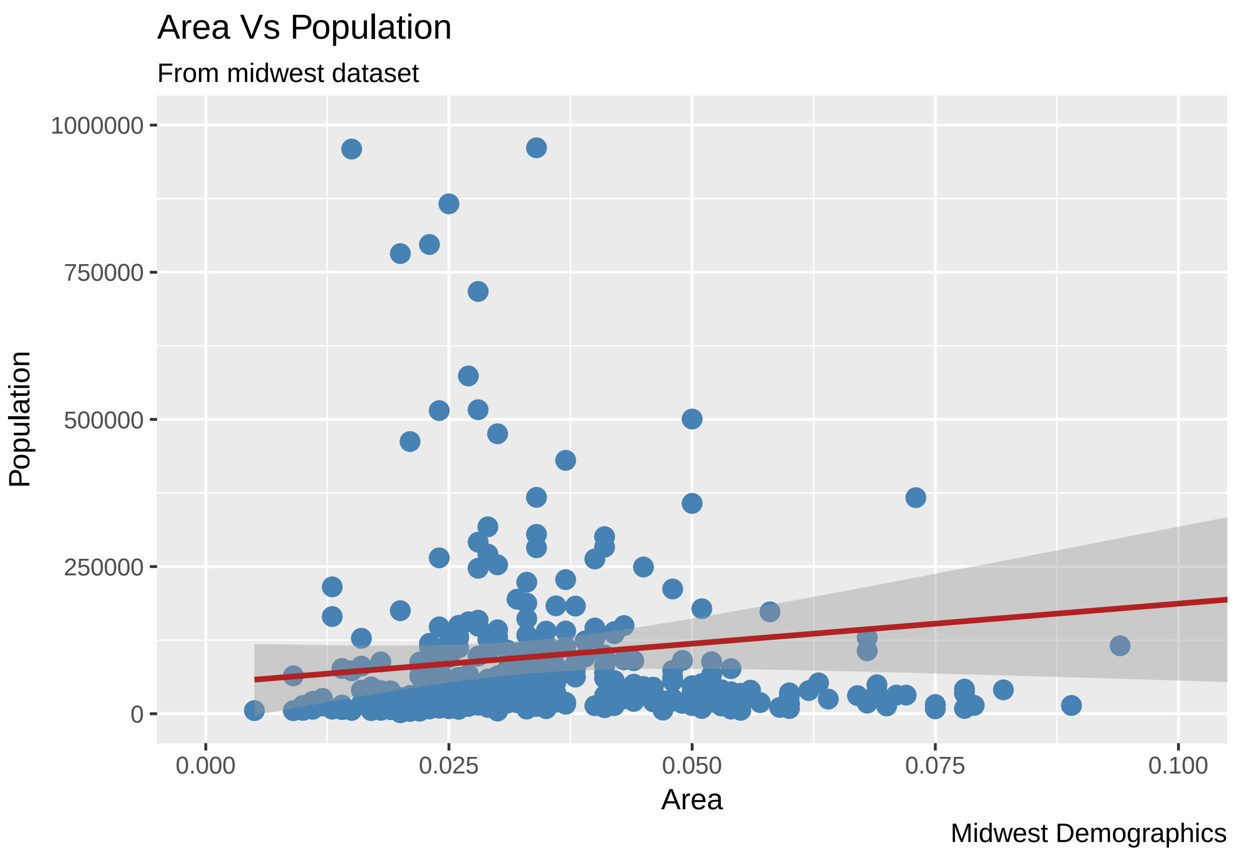


Color and size of the points

Static

In this case we specify the color.

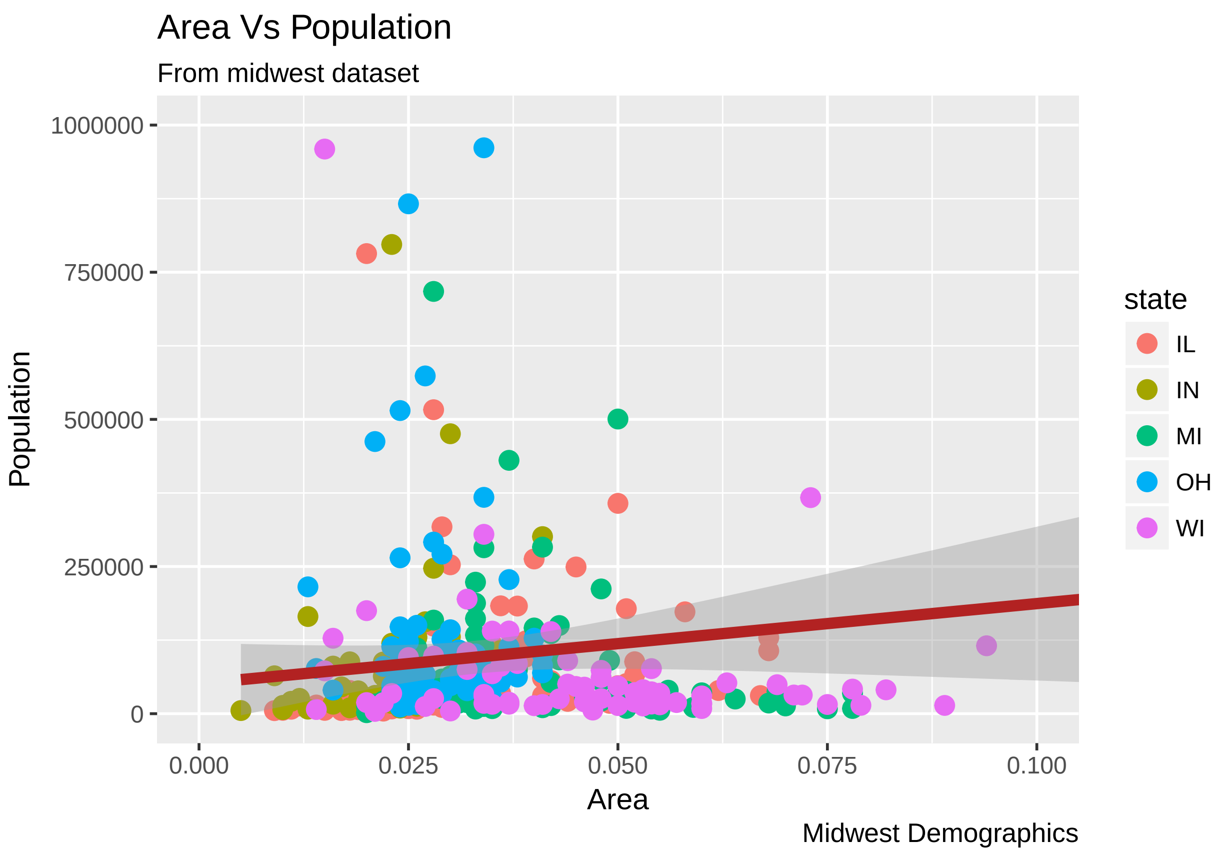
```
ggplot(midwest, aes(x = area, y = poptotal)) +
      geom_point(col = "steelblue", size = 3) +    # Set static color and size for points
      geom_smooth(method = "lm", col = "firebrick") +  # change the color of line
      coord_cartesian(xlim = c(0, 0.1), ylim = c(0, 1000000)) +
      labs(title = "Area Vs Population",
           subtitle = "From midwest dataset",
           y = "Population",
           x = "Area",
           caption = "Midwest Demographics")
```



Categories

We specify a category of colors, ie. a collection of colors.

```r
gg <- ggplot(midwest, aes(x = area, y = poptotal)) +
            geom_point(aes(col = state), size = 3) +  # Set color to vary based on state categories.
            geom_smooth(method = "lm", col = "firebrick", size = 2) +
            coord_cartesian(xlim = c(0, 0.1), ylim = c(0, 1000000)) +
            labs(title = "Area Vs Population",
                 subtitle = "From midwest dataset",
                 y = "Population",
                 x = "Area", caption = "Midwest Demographics")
plot(gg)
```



Another paletes of color

With this package we would have more palets to use.

```
#trinstall.packages("RColorBrewer")    # uncommented in case it is not installed
library(RColorBrewer)
head(brewer.pal.info, 10)
```

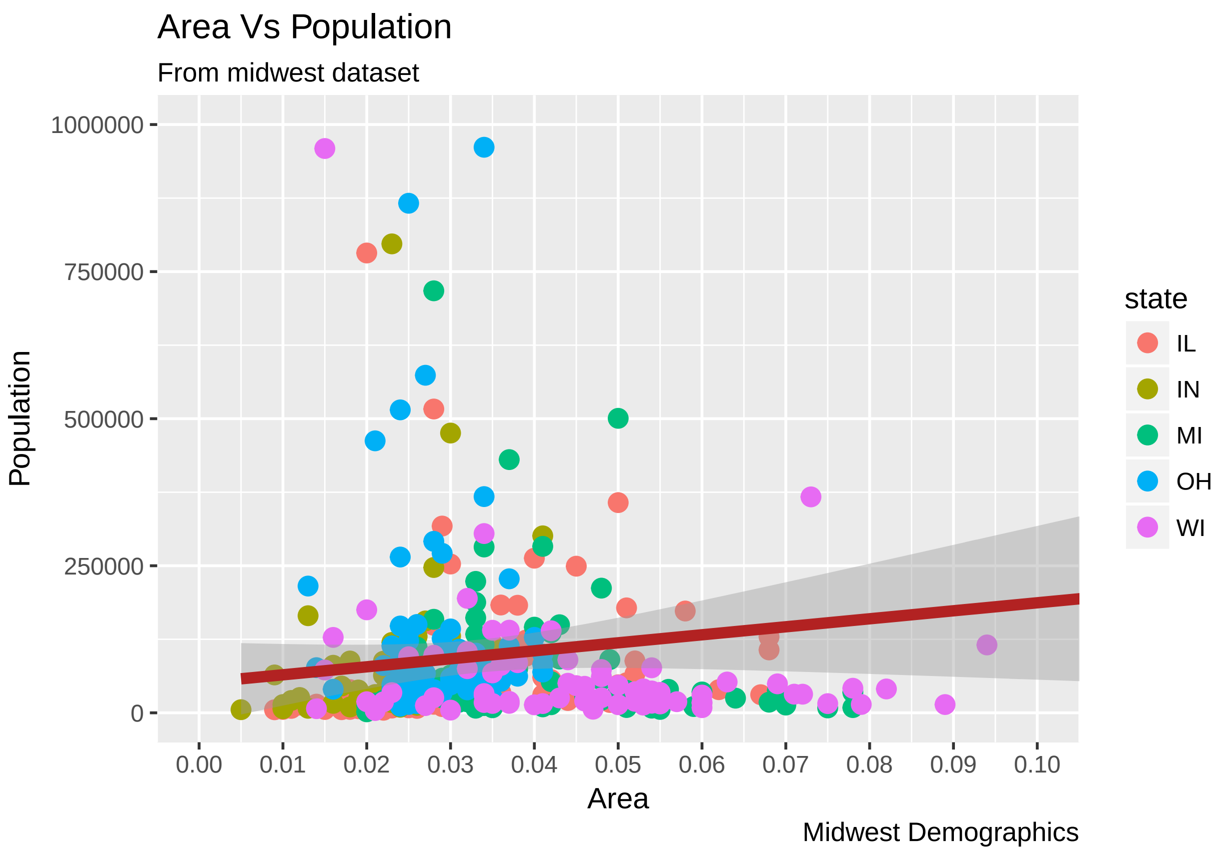
```
##          maxcolors category colorblind
## BrBG            11      div       TRUE
## PiYG            11      div       TRUE
## PRGn            11      div       TRUE
## PuOr            11      div       TRUE
## RdBu            11      div       TRUE
## RdGy            11      div      FALSE
## RdYlBu          11      div       TRUE
## RdYlGn          11      div      FALSE
## Spectral        11      div      FALSE
## Accent           8     qual      FALSE
```

Note: see that the `head`function is just to show us the first (10 in this case) elements in a collections of elements saved in a data.frame object.

Change the X axis texts and ticks location

- *Step 1*: Set the breaks

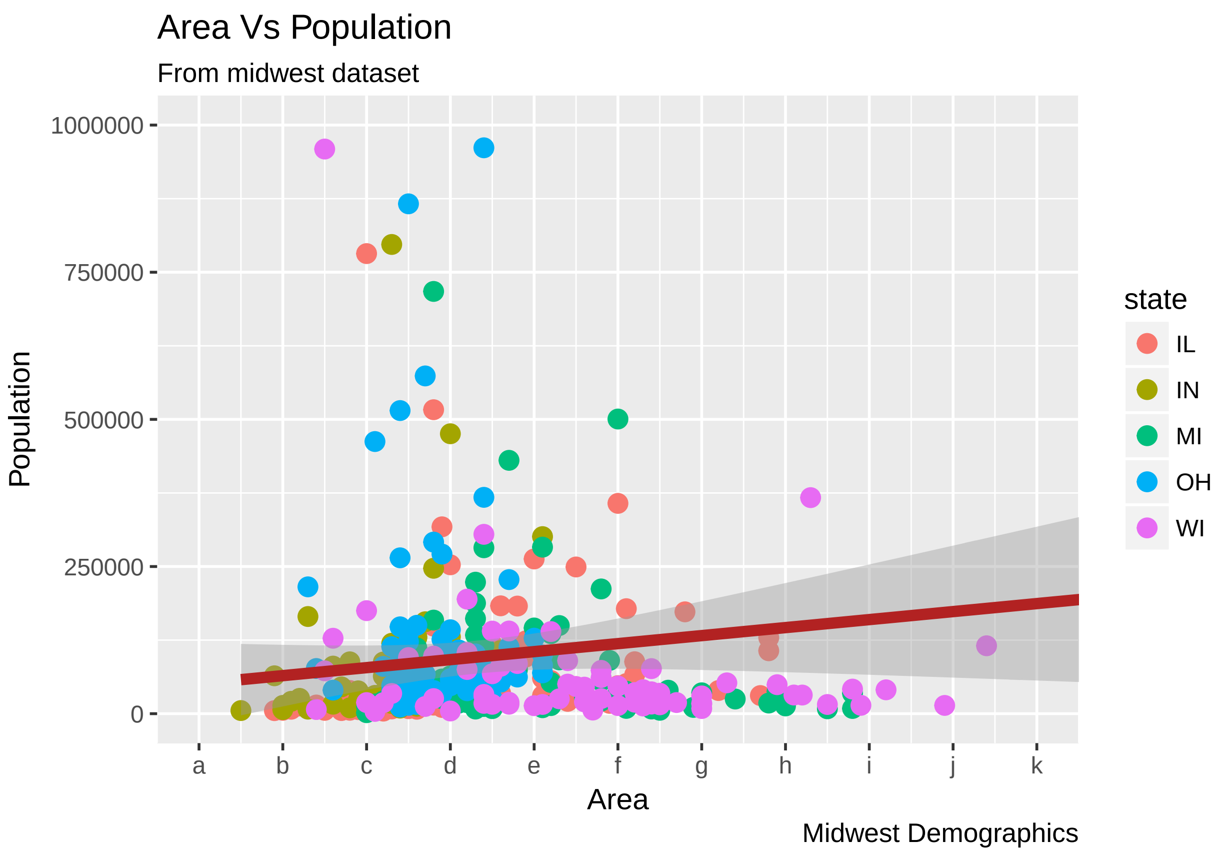
```r
# Base plot
gg <- ggplot(midwest, aes(x = area, y = poptotal)) +
        geom_point(aes(col = state), size = 3) +  # Set color to vary based on state categories.
        geom_smooth(method = "lm", col = "firebrick", size = 2) +
        coord_cartesian(xlim = c(0, 0.1), ylim = c(0, 1000000)) +
        labs(title = "Area Vs Population",
             subtitle = "From midwest dataset",
             y = "Population",
             x = "Area",
             caption = "Midwest Demographics")

# Change breaks
gg + scale_x_continuous(breaks = seq(0, 0.1, 0.01))
```



Note: the function `breaks` create an array with numbers from 0 to 0.1 whit a step size of 0.01.

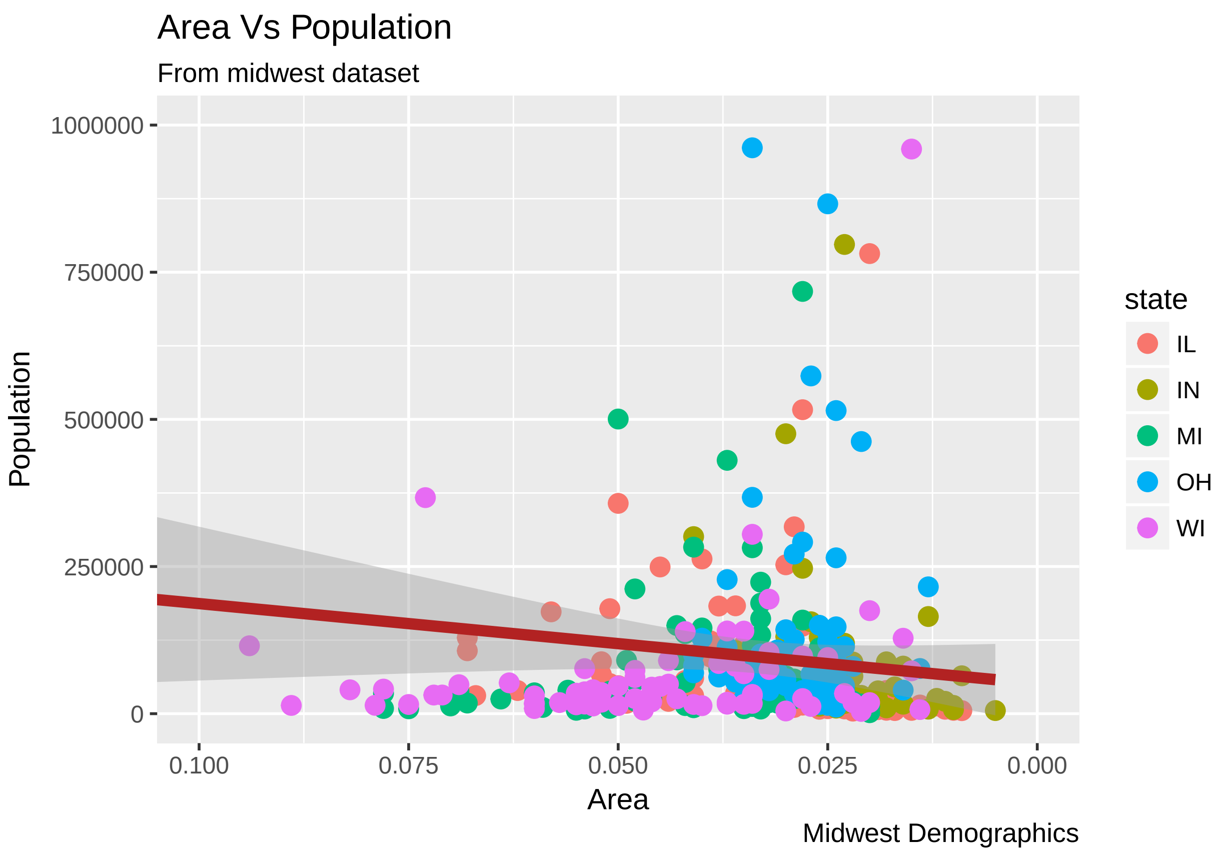
- *Step 2*: Change the labels

```r
gg + scale_x_continuous(breaks = seq(0, 0.1, 0.01), labels = letters[1:11])
```



Note: we can reverse the edges using the next function.

```
# Reverse X Axis Scale
gg + scale_x_reverse()
```



Source: http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html