6.1.9 Multiple Plots on One Page

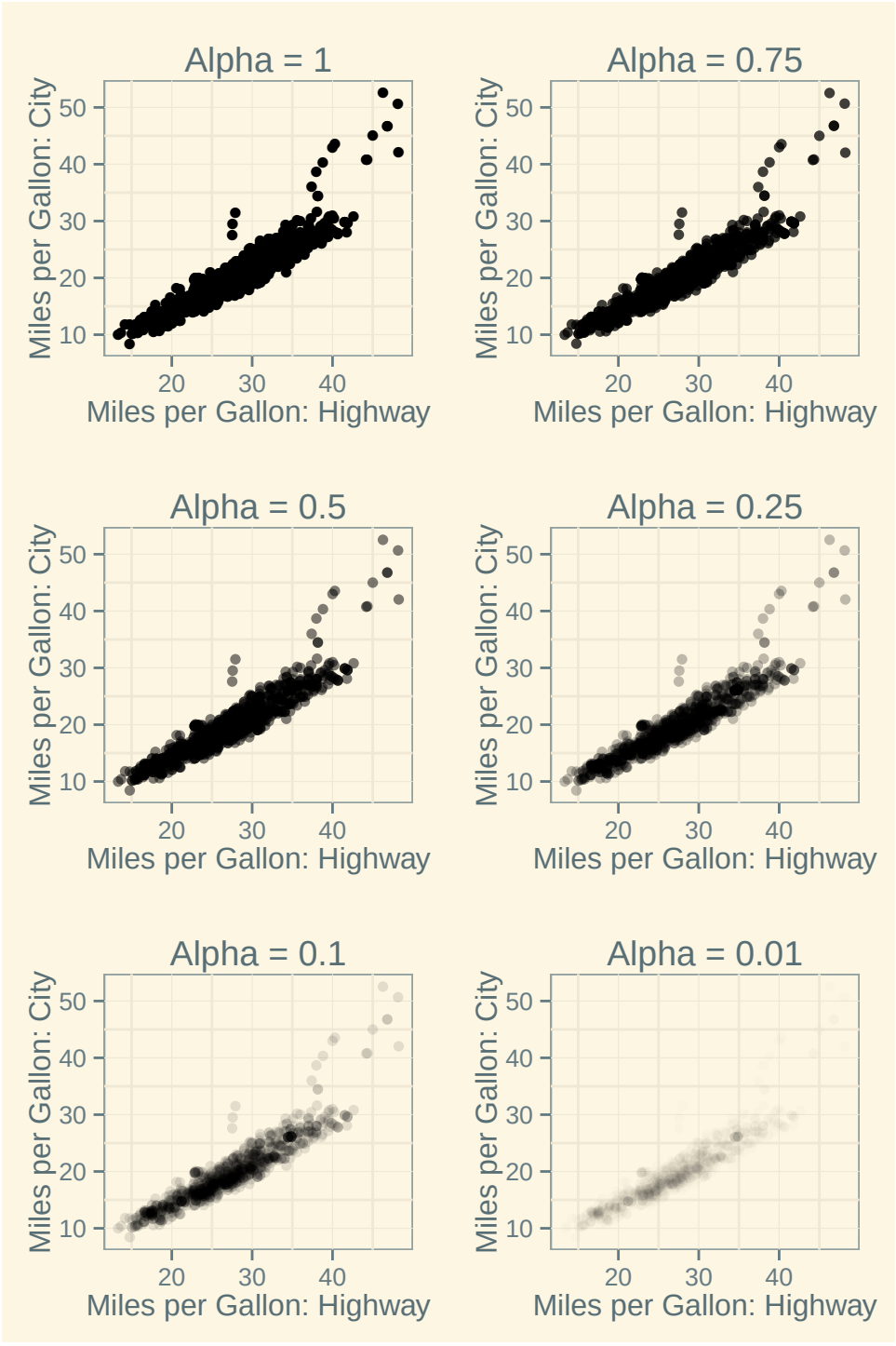
It is often the case that you will want to add multiple plots to a page. This can easily be done by initiating a device that is split up into a grid. To do this you will need to install a package called `grid` and then load it via the `library()` function.

Below, I created four separate scatter plots. They are all identical except that I am varying the transparency of the points with the `alpha` argument. This is a useful argument to highlight overplotting.

```
> A <- ggplot(data = FE2013) +
    geom_point(aes(x = FEhighway, y = FEcity), alpha = 1) +
    labs(title = "Alpha = 1", x = "Miles per Gallon: Highway"
       , y = "Miles per Gallon: City")

> B <- ggplot(data = FE2013) +
    geom_point(aes(x = FEhighway, y = FEcity), alpha = 0.5) +
    labs(title = "Alpha = 0.5", x = "Miles per Gallon:
       Highway", y = "Miles per Gallon: City")

> C <- ggplot(data = FE2013) +
    geom_point(aes(x = FEhighway, y = FEcity), alpha = 0.3) +
    labs(title = "Alpha = 0.3", x = "Miles per Gallon:
       Highway", y = "Miles per Gallon: City")

> D <- ggplot(data = FE2013) +
    geom_point(aes(x = FEhighway, y = FEcity), alpha = 0.1) +
    labs(title = "Alpha = 0.1", x = "Miles per Gallon:
       Highway", y = "Miles per Gallon: City")
```

To plot A, B, C, D, E, and F into a 3 by 2 grid, load the grid package and enter the following code:

```
> library(grid) # you may have to install the package first
> grid.newpage()
> pushViewport(viewport(layout = grid.layout(3, 2)))
> Layout <- function(x, y) viewport(layout.pos.row = x,
   layout.pos.col = y)
> print(A, vp = Layout(1, 1))
> print(B, vp = Layout(1, 2))
> print(C, vp = Layout(2, 1))
> print(D, vp = Layout(2, 2))
> print(E, vp = Layout(3, 1))
> print(F, vp = Layout(3, 2))
```


Alpha = 1
Alpha = 0.75
Alpha = 0.5
Alpha = 0.25
Alpha = 0.1
Alpha = 0.01
Miles per Gallon: City
Miles per Gallon: Highway
10
20
30
40
50

6.1.10 Recap: The Makings of a ggplot

A quick step by step summary of how to create a plot in ggplot:

1. Create a ggplot object with the ggplot() function. You will need to specify the data you will be using via the data argument.

   → `ggplot(data = MyDataset)`

2. Add a geom layer to the plot and specify the aesthetic mapping with aes

   → `geom_point(aes(x = Variable1, y = Variable2))`

   → `geom_boxplot(aes(x = Variable1))`

   → `geom_smooth(aes(x = Variable1, y = Variable2))`

3. You are done. Unless you want to mess with the defaults

4. Changing the default axis and plot labels:

   → `labs(title = "MyTitle", x = "X-Axis Label", y = "Y-Axis Label")`

5. Changing the range of the plot

   → `coord_cartesian(xlim = c(xmin,xmax), ylim = c (ymin,ymax))`

6.1.11 Common Aesthetics

All geoms require a set of aesthetic mappings (unless you already specified a default in the ggplot() function itself). The most common ones are x and y. For example to plot a scatterplot the geom_point() function needs to know which variable you want to map to the X-axis and which to the Y-axis. Below is a list of common additional ones that you can often specify within the aes() argument, or outside the aes() argument if you don't want things to be mapped to the legend. Always consult the documentation for a list of aesthetics each geom requires or understands: http://docs.ggplot2.org/current/.

Aesthetic	Description
x	for mapping a variable to the x-axis
y	for mapping a variable to the y-axis
color	for mapping a variable or constant to a color → `aes(color = Variable1)` or `aes(color = "red")`
linetype	for mapping a variable or constant to a linetype → `aes(linetype = Variable1)` or `aes(linetype = 2)`
shape	for mapping a variable or constant to the shape of points (squares, triangles, etc) → `aes(shape = Variable1)` or `aes(shape = 3)`
size	for mapping a variable or constant to the size or width of points or lines → `aes(size = Variable1)` or `aes(size = 5)`
alpha	for mapping a variable or constant to the transparency of lines or points, etc → `aes(alpha = Variable1)` or `aes(alpha = 0.5)`
fill	for mapping a variable or constant to the fill color of an area → `aes(fill = Variable1)` or `aes(fill = "green")`