The power of R Markdown + GitHub + Overleaf

Your Name Here

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

These code blocks are where I keep all of my notes & working thoughts on figure generation and analyses. When you "Knit" the R Markdown doc, it converts this markdown file to other document formats, such as PDF or HTML. This format is useful *in place of* creating a powerpoint of your figures because you can integrate your code and annotations all in one! And send to others as a single document (pdf/html) or the source .Rmd that created the doc. However, one downside is - if your code takes awhile to process (large dataset/many iterations), it will take that much longer to knit. This can be frustrating when producing a final product. I would recommend avoiding intensive processing in your .Rmd - it will certainly work and can be done, but if you're trying to iterate quickly and refine your .Rmd for sharing, long knitting times can be prohibitive. You can also use the "cache=TRUE" option in a chunk, which works well if you're confident that chunk and its outputs won't need to change. When importing the datasets/analysis products from other scripts/locations, just make sure to document VERY well what script generated it & where the data live.

This first chunk is the setup chunk. Currently, this will not appear in your knitted document because of "include=FALSE" (i.e. do NOT include this chunk in the output pdf/html). You can always remove the "include=FALSE" argument from the brackets to make more transparent what was required for your notebook (i.e. include in the output pdf all packages used and information to set up your .Rmd.

Here we do 4 things in the set up brackets:

1. Designate this is R code (you can run other languages)
2. Load the built-in cars dataset
3. Designate global options set for the entire notebook
4. Designate the include=FALSE, which means: do NOT include this chunk visibly in the knit document

Then, set knitr options: *Specifically, we tell knitr how to create figures and where to write them out (i.e. our repository!)*

Then, we load any necessary packages used in the .Rmd

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Above, I changed the font (bold, italicized), but there are built-in ways to organize your document. Different headings, as seen below, are useful for guiding readers through a more "outlined" document.

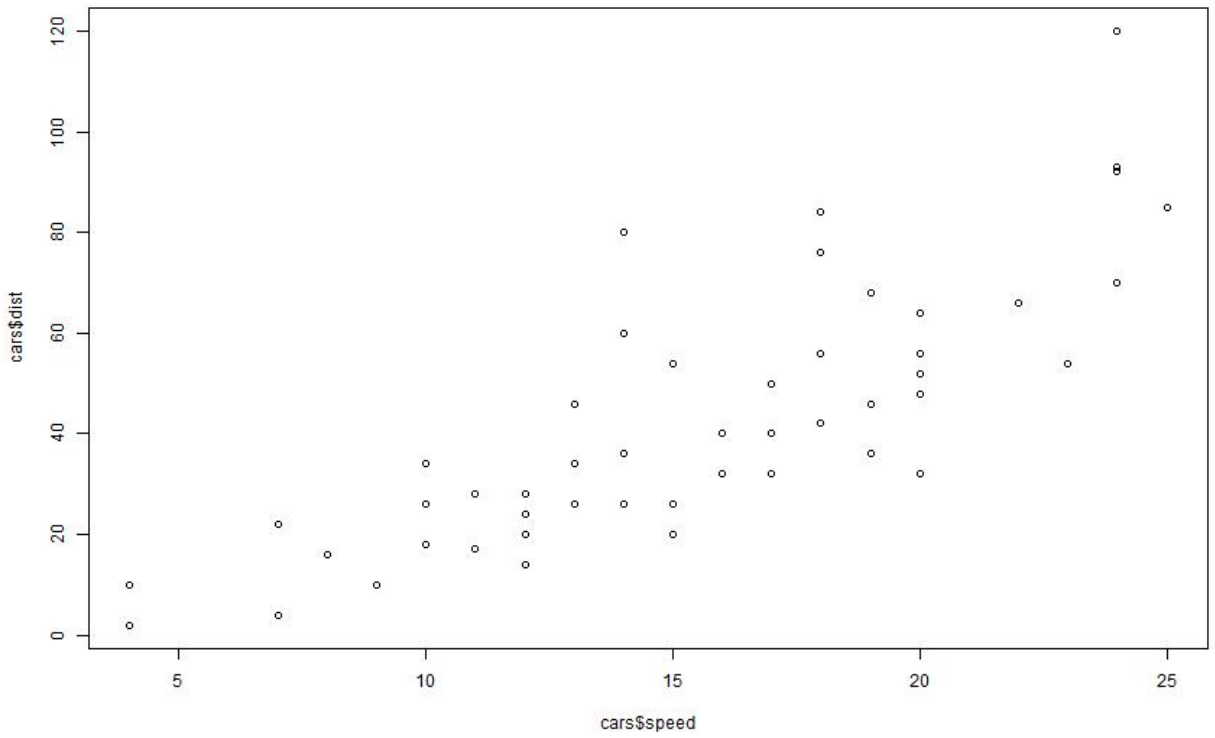
Yay for Plots!

Using built-in R data & Base R

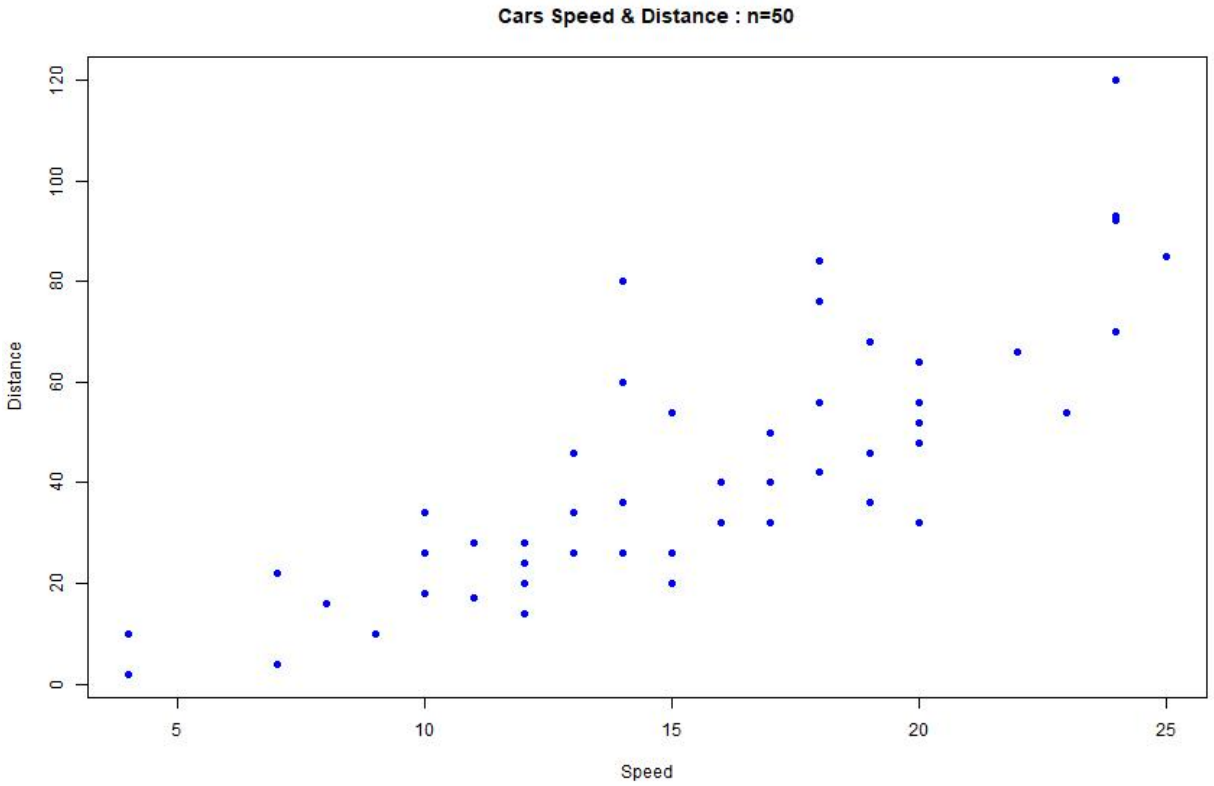
Oooooh aaaaaahhhh, in-line code

There are 50 records in the cars dataset.

```
#plot speed over distance
plot(cars$speed,cars$dist)
```




```
#same plot, but improved! color, aces labels, symbol type specification, and title
plot(cars$speed,cars$dist,col="blue",xlab="Speed",ylab="Distance",pch=16,main="Cars Speed & Distance : r
```




```
#we can write out any dataframe as csv
write.csv(cars,"cars.csv")
```

Plot Midwest Data

Use ggplot

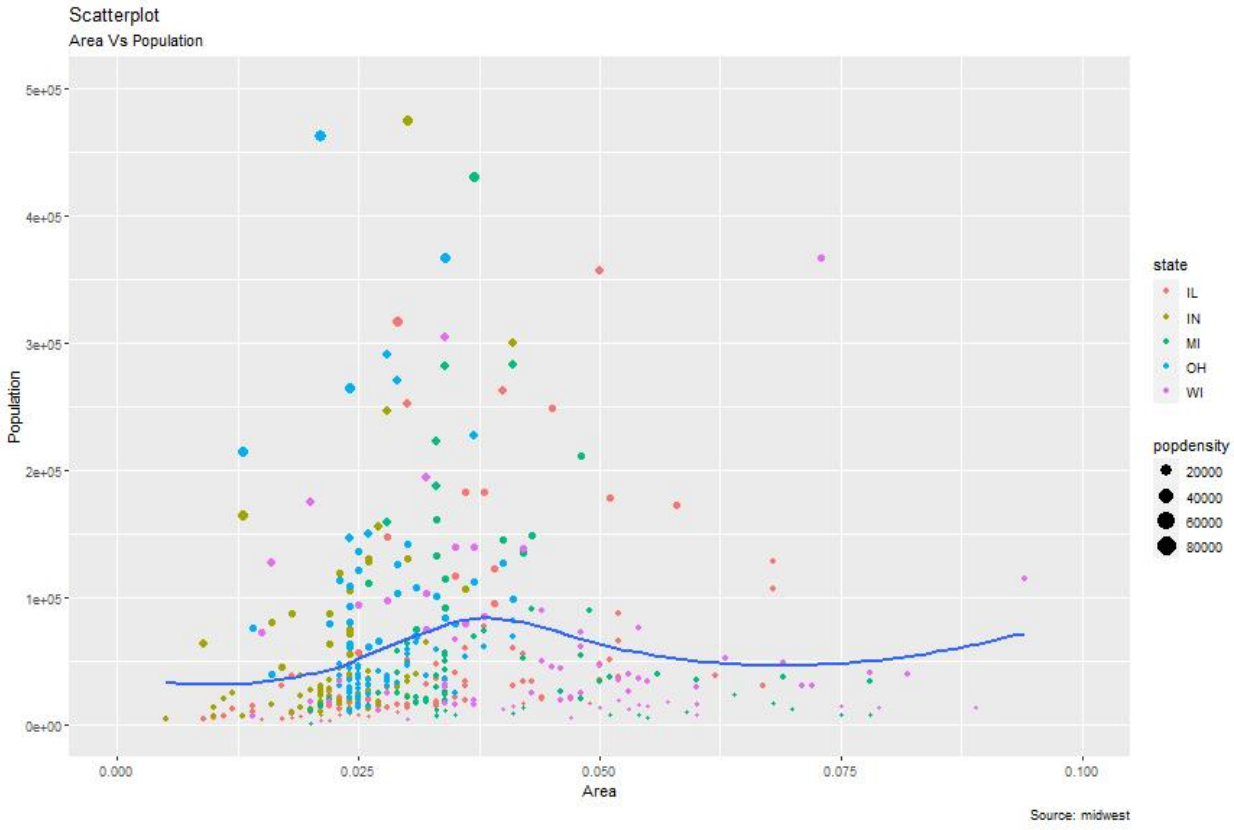
Learn about plot strucure & themes!

Plots examples below and more from this website:

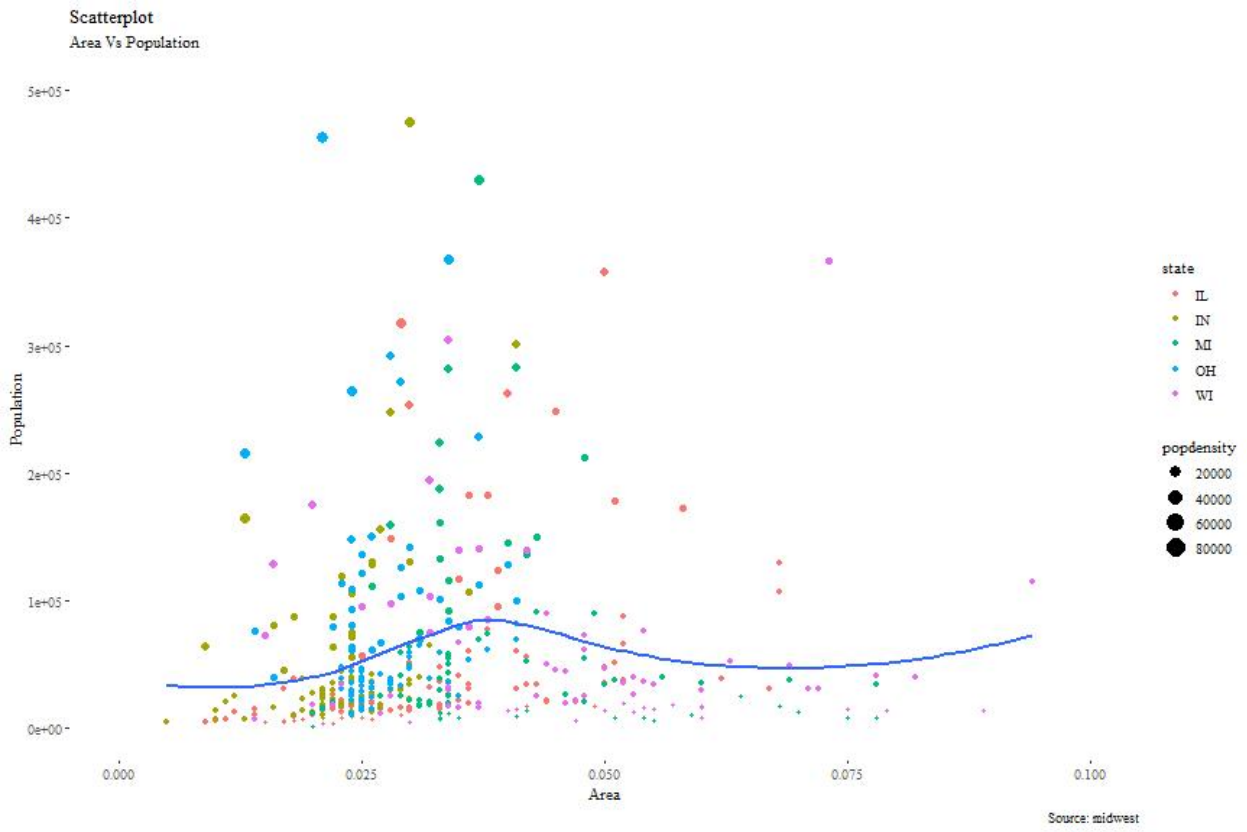
http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html#Bubble%20Plot

```
#we read in another built-in dataset
data("midwest", package = "ggplot2")

#specify as dataframe, this is a step I generally do, but is not always necessary
midwest_df<-as.data.frame(midwest)

# Creating a scatterplot w/ ggplot
gg <- ggplot(midwest_df, aes(x=area, y=poptotal)) + #plot with specifying data as x,y
  geom_point(aes(col=state, size=popdensity)) + #color the scatterplot by state and size the dots by po
  geom_smooth(method="loess", se=F) + #loess = "smooth local regression"
  xlim(c(0, 0.1)) + #x-axis limits
  ylim(c(0, 500000)) + #y-axis limits
```

```
  labs(subtitle="Area Vs Population",  #specify all labels
       y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")

#here we can plots or design elements together to create a new plot
#it doesn't plot the plot, until calling "plot" (below)
gg_themed<- gg +  theme_tufte()

plot(gg)
```




```
plot(gg_themed)
```



When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

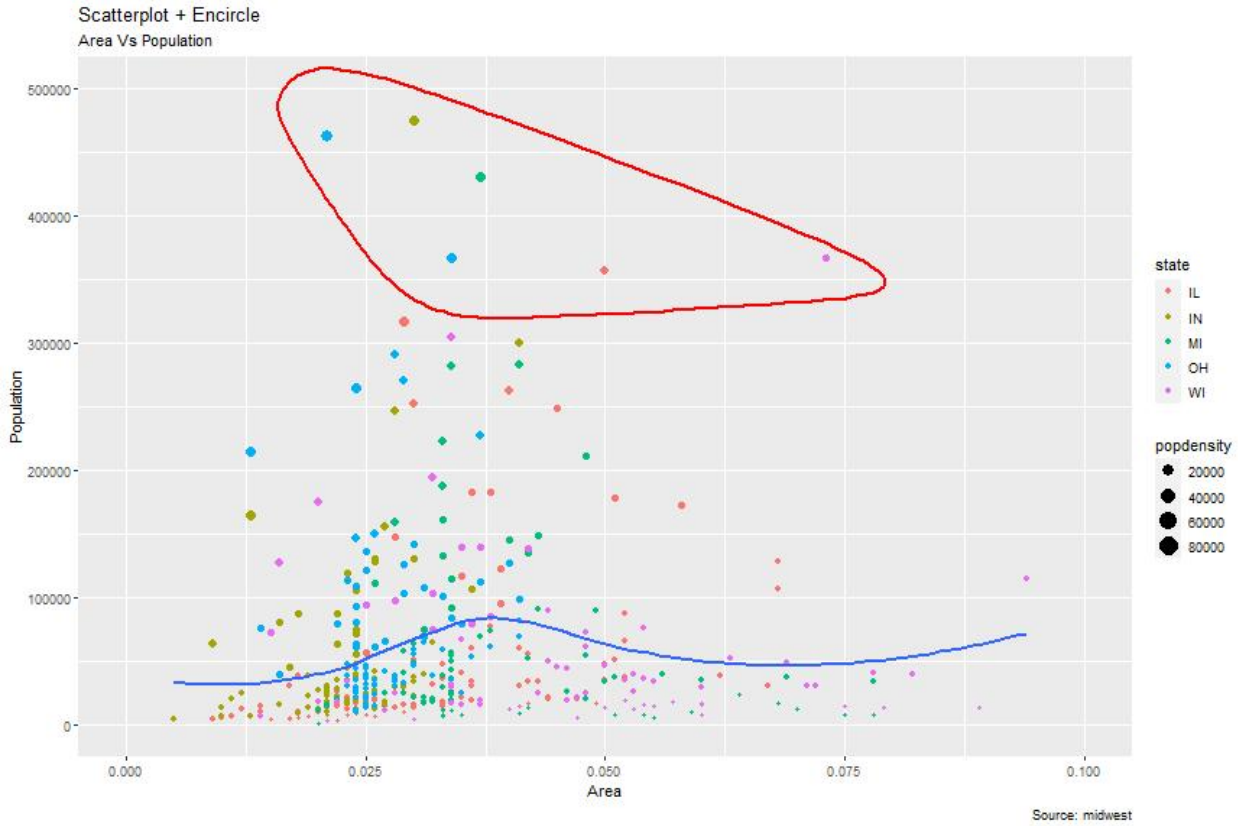
Some additional functionality with ggplot & Markdown

Can create a variety of graphics

Again, this is a great place to leave notes,questions, or explanations about the figures in the chunk below. We all know we should write better figure captions or document our thoughts more often before we share a plot or have to explain it. This is a great place to write while it's fresh on your mind!

```
#specifies when to use scientific notation
#if you want to read more, here's a useful explanation: https://stackoverflow.com/questions/25946047/ho
options(scipen = 999)
midwest_select <- midwest_df[midwest_df$poptotal > 350000 & #creating catagories for the data
                             midwest_df$poptotal <= 500000 &
                             midwest_df$area > 0.01 &
                             midwest_df$area < 0.1, ]

# Plot
ggplot(midwest_df, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +   # draw points
  geom_smooth(method="loess", se=F) +
  xlim(c(0, 0.1)) +
  ylim(c(0, 500000)) +   # draw smoothing line
  geom_encircle(aes(x=area, y=poptotal),
```

```
                data=midwest_select,
                color="red",
                size=2,
                expand=0.08) +   # encircle
  labs(subtitle="Area Vs Population",
       y="Population",
       x="Area",
       title="Scatterplot + Encircle",
       caption="Source: midwest")
```



To prove ggplot makes more than scatterplots

Annotations in an .Rmd do NOT take the place of commenting in line! Comment. Comment. Comment.

```
# Data Prep
data("mtcars")  # load data

mtcars_df<-as.data.frame(mtcars)

mtcars_df$`car name` <- rownames(mtcars_df)  # create new column for car names
mtcars_df$mpg_z <- round((mtcars_df$mpg - mean(mtcars_df$mpg))/sd(mtcars_df$mpg), 2)  # compute normali
mtcars_df$mpg_type <- ifelse(mtcars_df$mpg_z < 0, "below", "above")  # above / below avg flag
mtcars_df <- mtcars_df[order(mtcars_df$mpg_z), ]  # sort
mtcars_df$`car name` <- factor(mtcars_df$`car name`, levels = mtcars_df$`car name`)  # convert to facto
```

```
# Diverging Barcharts
ggplot(mtcars_df, aes(x='car name', y=mpg_z, label=mpg_z)) +
  geom_bar(stat='identity', aes(fill=mpg_type), width=.5)  +
  scale_fill_manual(name="Mileage",
                    labels = c("Above Average", "Below Average"),
                    values = c("above"="#00ba38", "below"="#f8766d")) +
  labs(subtitle="Normalised mileage from 'mtcars'",
       title= "Diverging Bars") +
  coord_flip()
```