

Intro to R - ggplot2

BIOS 545

Steve Pittard wsp@emory.edu

March 20, 2022

Multiple Graphics Packages

There are Four Major Graphics Packages in R

- Base Graphics - Oldest and Most Well Known
- lattice - Second Oldest and Supports Grouping and Panels
- ggplot - Third Oldest and Supports Grouping and facets (similar to Panels)
- grid - This is a low level library that all the other graphic packages use

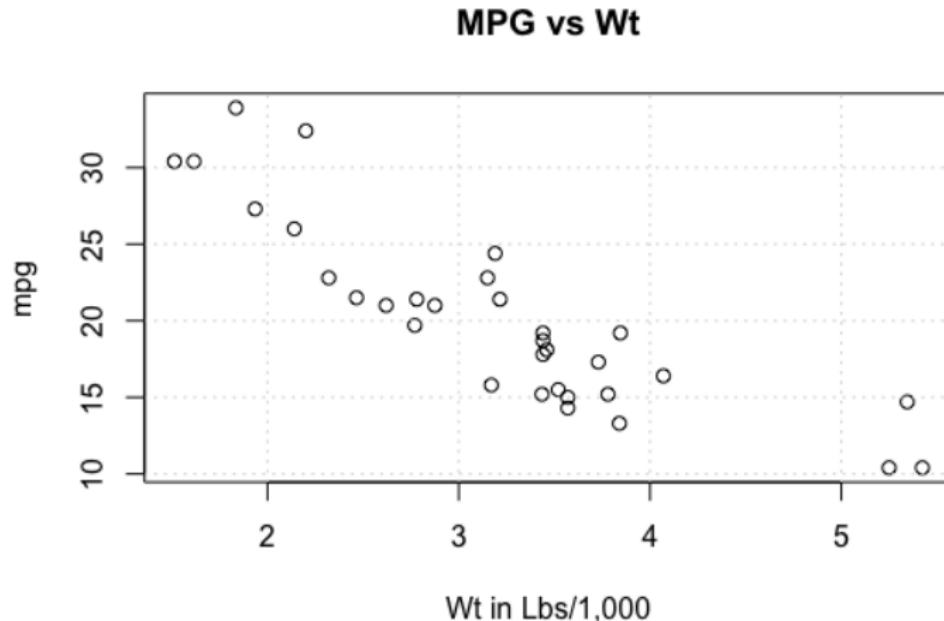
Base Graphics

Base Graphics is the Oldest

- Has both low and high level commands
- You can draw polygons and lines
- Or you can call functions like **host boxplot**, etc
- You can reproduce any graphic you encounter
- Frequently need to build a plot in stages - layer by layer
- Uses “pen on paper” approach
- If you mess up then just start all over

Base Graphics

```
data(mtcars)
xlab <- "Wt in Lbs/1,000"
main <- "MPG vs Wt"
plot(mpg~wt,data=mtcars,main=main,xlab=xlab)
grid()
```



Lattice Graphics

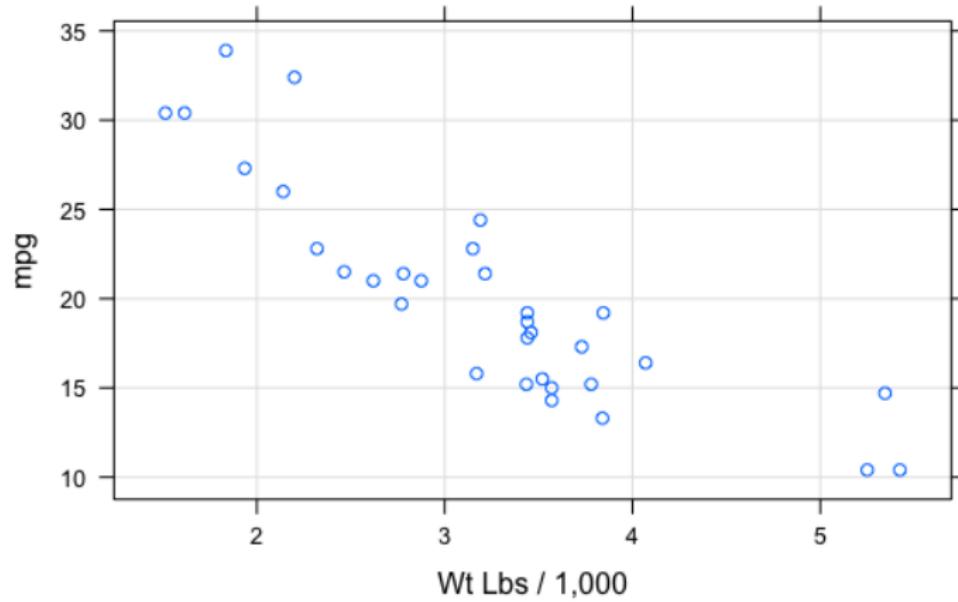
Lattice was written to provide grouping and paneling

- Consistent look and feel
- Great for multivariate data
- Takes care of lots of things for you
- Has a formula interface
- Lots of examples and support on Google
- See
<http://lmdvr.r-forge.r-project.org/figures/figures.html>
- Picks useful defaults for you

Lattice Graphics

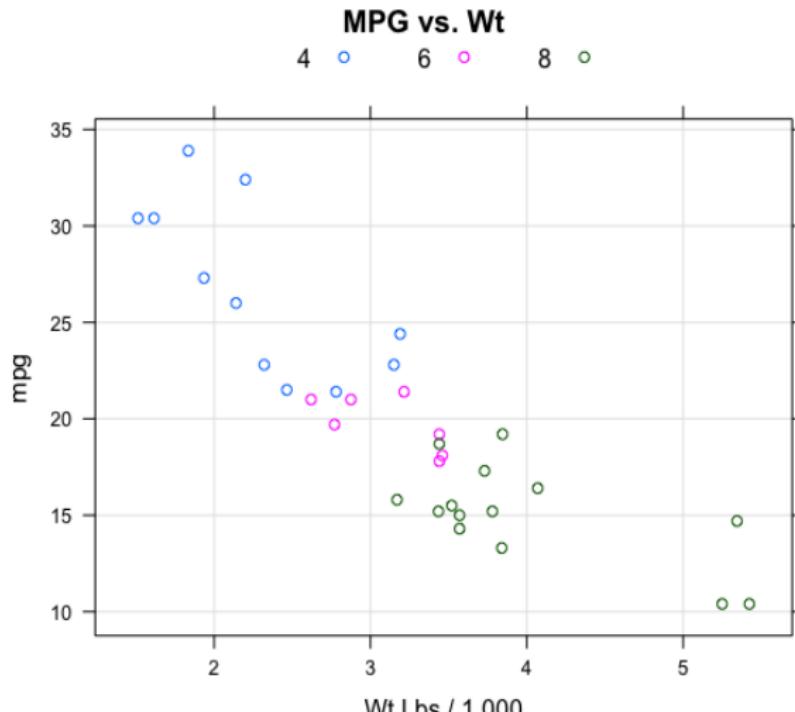
```
library(lattice)
xlab <- "Wt Lbs / 1,000"
main <- "MPG vs. Wt"
xyplot(mpg~wt,data=mtcars,main=main,xlab=xlab,type=c("p", "g"))
```

MPG vs. Wt



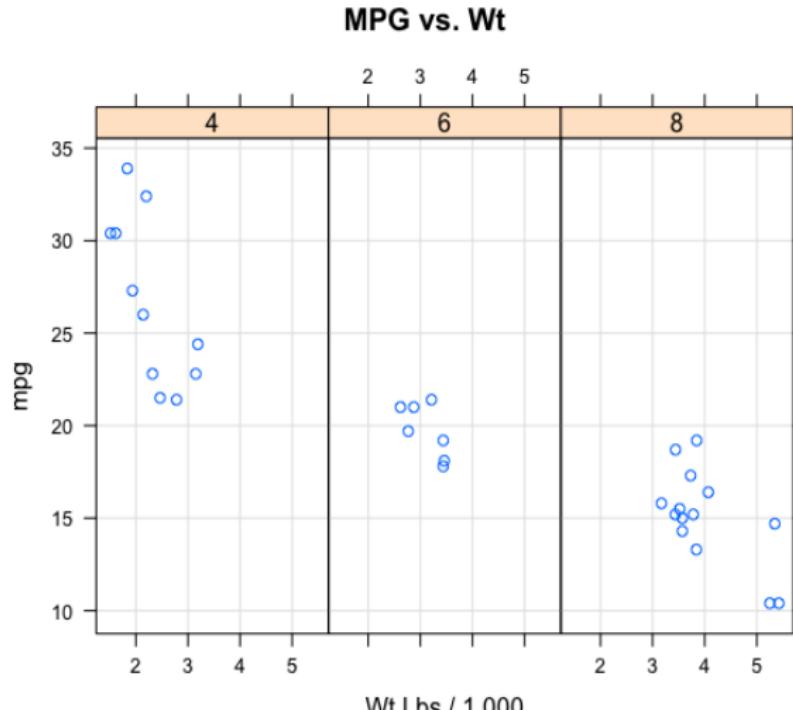
Lattice Graphics - Grouping

```
library(lattice)
xlab <- "Wt Lbs / 1,000"; main <- "MPG vs. Wt"
xyplot(mpg~wt,groups=factor(cyl),data=mtcars,main=main,xlab=xlab,
       type=c("p","g"),auto.key=list(columns=3))
```



Lattice Graphics - Panels

```
library(lattice)
xlab <- "Wt Lbs / 1,000"; main <- "MPG vs. Wt"
xyplot(mpg~wt|factor(cyl),data=mtcars,main=main,xlab=xlab,
       type=c("p","g"),layout=c(3,1))
```



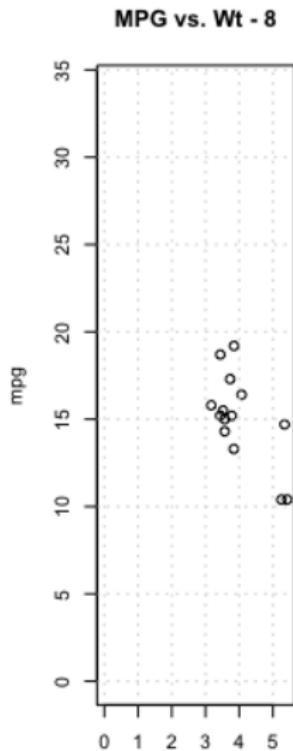
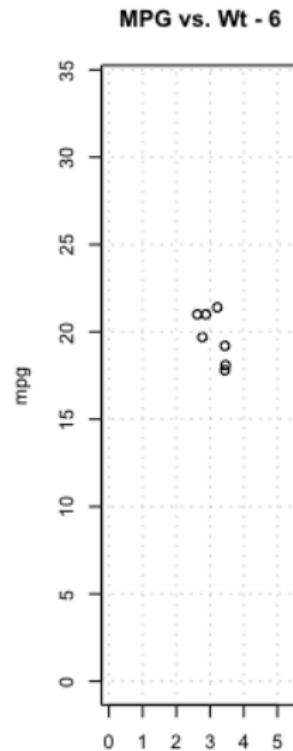
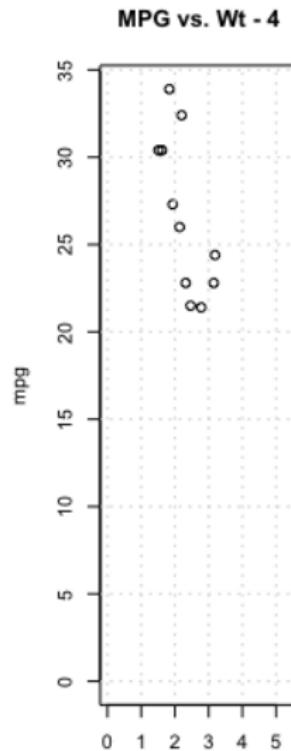
Lattice Graphics - Panels

Doing Panels in Base Graphics is a Manual Process. This can be Powerful but very tedious

```
xlab <- "Wt Lbs / 1,000"; main <- "MPG vs. Wt"
par(mfrow=c(1,3))
maxmpg <- max(mtcars$mpg)
maxwt   <- max(mtcars$wt)
mydf <- split(mtcars,mtcars$cyl)
for (ii in 1:length(mydf)) {
  tmpdf <- mydf[[ii]]
  main <- paste("MPG vs. Wt",names(mydf)[ii],sep=" - ")
  plot(mpg~wt,data=tmpdf,main=main,
        xlim=c(0,maxwt),
        ylim=c(0,maxmpg))
  grid()
}
```

Lattice Graphics - Panels

Doing Panels in Base Graphics is a Manual Process



Two Modes

We usually operate in one of two modes: Exploratory and Publication

- Use exploratory for early attempts at understanding the data
- It does not need to be pretty or elegant
- Look for relationships
- Can add annotations later
- Usually just for yourself and other team members

Two Modes

Independently of the graphics packages we operate in two modes:
Exploratory and Publication

- Publication mode is for, well, publication and sharing
- This is for journals
- Graphs are usually standard types but combined in interesting ways
- Comparisons across panels and facets
- Axes, legends, titles, colors, groups are usually present
- ggplot makes it easy to create good default graphics

Intro

ggplot2 is a relative newcomer to R

- Written according to a "Grammar of Graphics" (Wilkinson, 2005)
- Discuss the visualization of data using an accepted vocabulary
- Flexibly explore the data using a number of visualizations (or combinations thereof)
- No need to "commit" to specific chart types
- Specify plots in abstract terms using the aforementioned grammar
- Rapidly becoming the default R graphics package
- Attempts to leverage the good parts of lattice and Base graphics

See home page for ggplot at <http://ggplot2.org/>

Intro

Some helpful resources:

- Presentation - ggplot2.org/resources/2007-past-present-future.pdf
- Book - ggplot2: Elegant Graphics for Data Analysis (check Amazon)
- Vanderbilt Workshop - ggplot2.org/resources/2007-vanderbilt.pdf
- Mailing List - groups.google.com/forum/?fromgroups#!forum/ggplot2
- Documentation - ggplot2.tidyverse.org/reference/
- R for Data Science Online Book - r4ds.had.co.nz/
- Cheat Sheet -
www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- R Graphics Cookbook - www.cookbook-r.com/Graphs/index.html

tidyverse

ggplot is part of the "tidyverse"

- A collection of R packages that share common philosophies to work well together
- Home page for project is at - <http://tidyverse.org/>
- Main packages are: ggplot2, tibble, tidyverse, readr, purrr, and dplyr
- Can install from within R Studio just like any other package
- The name of the package is simply **tidyverse**

qplot

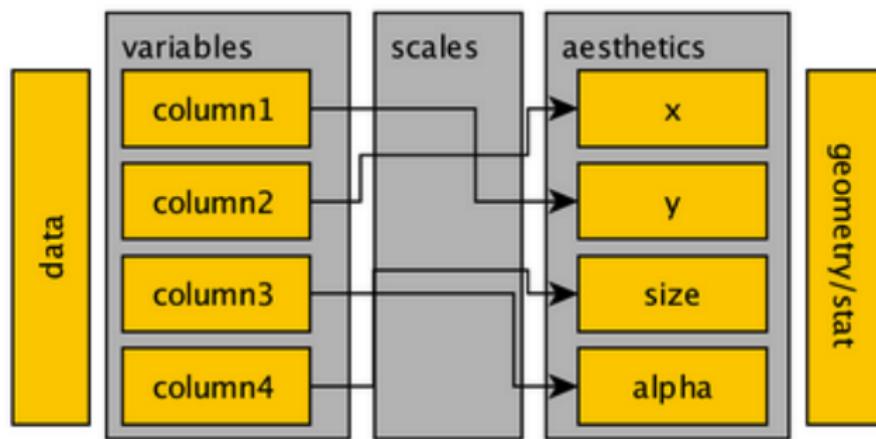
qplot is a command in ggplot that offers a "training wheels" like capability:

- The "q" in **qplot** stands for "quick"
- Is supposed to be an analog of the Base graphics **plot** command
- Useful if you already know Base Graphics really well
- Convenient wrapper for quickly creating a number of different types of plots
- However it's not really demonstrative of the power of grammar of graphics
- Not commonly used in "serious" ggplot projects

Key Ideas

There are some essential terms and concepts for using ggplot:

- Data - The actual data frame under consideration
- Aesthetics - Visual elements mapped to the data (axis, lines, colors, bars, etc)
- Scales - Transformations you might want to apply (e.g. logarithm, polar coordinates)
- Geometries - The shape mapped to the aesthetic(s)



Key Ideas

These ideas come from the **Grammar of Graphics**

- Understanding these ideas will help you define a plot in general terms that can then be implemented using ggplot commands
- Data - The actual data frame under consideration
- Aesthetics - Visual elements mapped to the data (axis, lines, colors, bars, etc)
- Scales - Transformations you might want to apply (e.g. logarithm, polar coordinates)
- Geometries - The shape mapped to the aesthetic(s)

Aesthetics

Here are some of the aesthetics that help make a plot:

- x and y position
- size of the elements
- shape
- color

We use geometries to view the data:

- lines and vsrisations (dashed, segements, etc)
- bars, histograms
- text labels
- points
- <http://ggplot2.tidyverse.org/reference/#section-layer-geoms>

mtcars

We'll first use the built-in mtcars data frame to explore these ideas since mtcars is an easy data frame to understand:

The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973-74 models).

A data frame with 32 observations on 11 variables.

```
[, 1] mpg Miles/(US) gallon
[ , 2] cyl Number of cylinders
[ , 3] disp Displacement (cu.in.)
[ , 4] hp Gross horsepower
[ , 5] drat Rear axle ratio
[ , 6] wt Weight (1000 lbs)
[ , 7] qsec 1/4 mile time
[ , 8] vs V/S
[ , 9] am Transmission (0 = automatic, 1 = manual)
[ ,10] gear Number of forward gears
[ ,11] carb Number of carburetors
```

mtcars

- What are the categories/factors in this data ?
- What are the continuous quantites ?

```
str(mtcars)
```

```
'data.frame': 32 obs. of 11 variables:  
 $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
 $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
 $ disp: num 160 160 108 258 360 ...  
 $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
 $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
 $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
 $ qsec: num 16.5 17 18.6 19.4 17 ...  
 $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
 $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
 $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
 $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

mtcars

- What are the categories/factors in this data ?
- What are the continuous quantitites ?
- Here is a "recipe" you can use with most any data frame

```
sapply(mtcars, function(x) length(unique(x)))  
mpg cyl disp hp drat wt qsec vs am gear carb  
25 3 27 22 22 29 30 2 2 3 6
```

- Variables taking on a limited number of unique values are probably factors
- Variables taking on many different unique values are probably continuous
- We generally seek to summarize and/or compare continuous information in terms of (or across) categories

mtcars

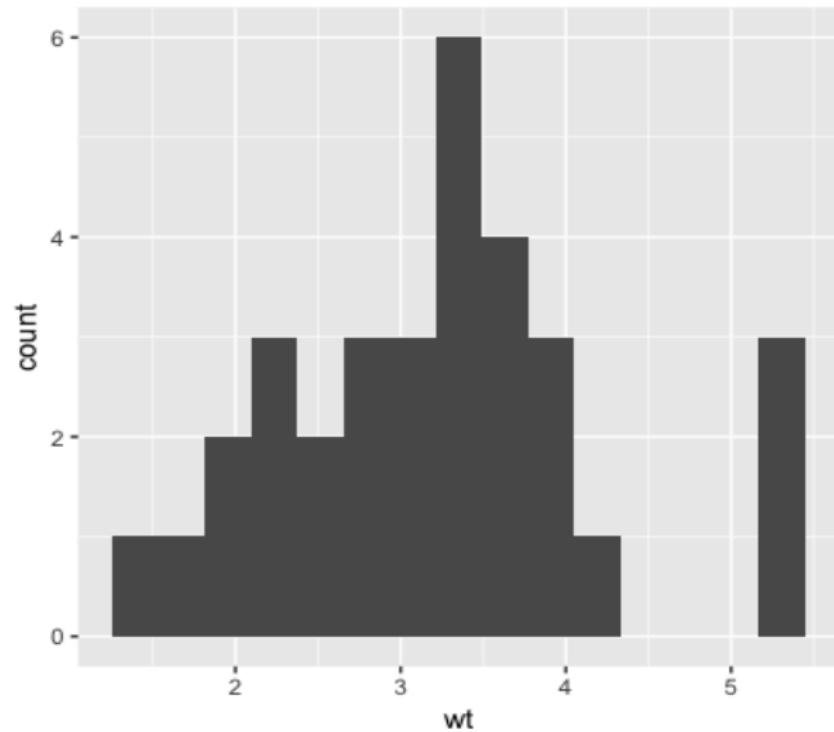
Some driving questions might be:

- What does the distribution of **wt** values look like ?
- Is there a relationship between **mpg** vs **wt** ?
- Does **mpg** appear to be different over individual cylinder groups ?
- Is there a relationship between **mpg** and **wt** that is effected by cylinder group ?
- Does **mpg** appear to be different over different transmission types ?
- What are the counts of transmission types and cylinder groups ?

mtcars

What does the distribution of **wt** values look like ?

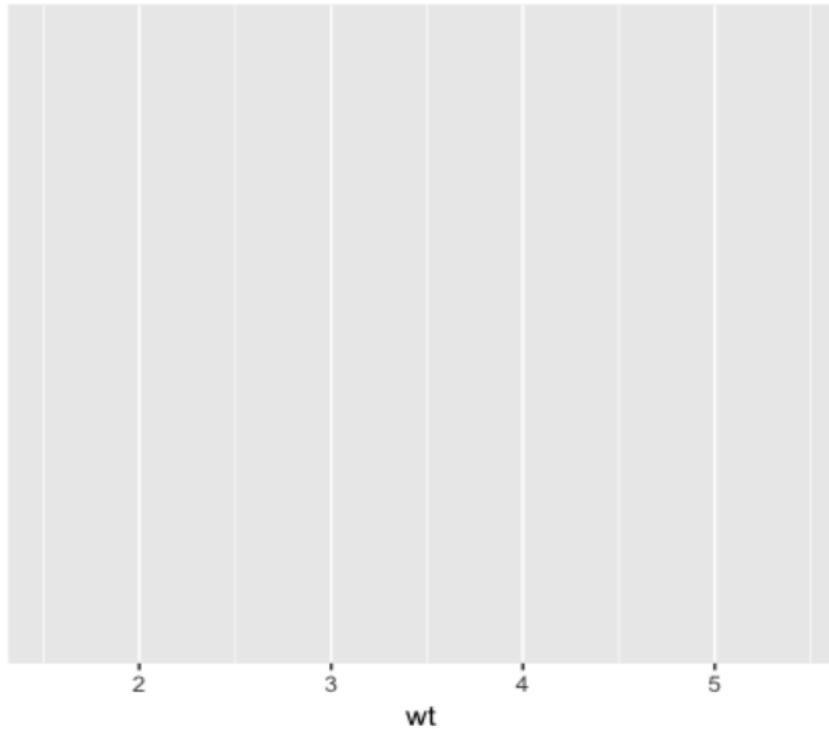
```
ggplot(mtcars,aes(x=wt)) + geom_histogram(bins=15)
```



mtcars

What if we don't supply a **geometry** ?

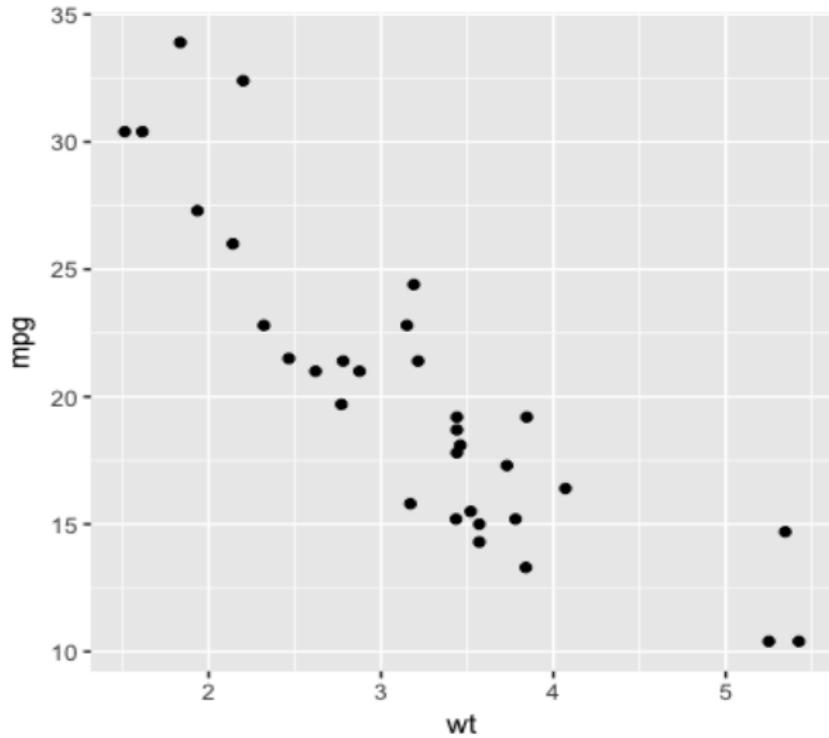
```
ggplot(mtcars,aes(x=wt))
```



mtcars

Is there a relationship between **mpg** vs **wt** ?

```
ggplot(mtcars,aes(x=wt)) + geom_point(aes(y=mpg))
```



mtcars

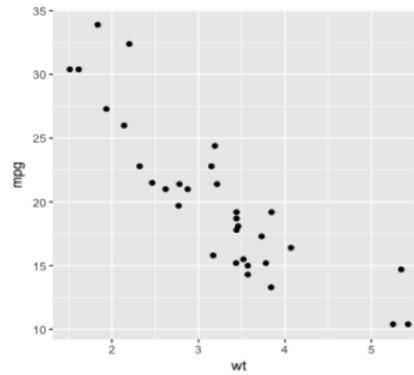
Is there a relationship between **mpg** vs **wt** ?

Note how we added a new geometry on an existing aesthetic mapping then added another aesthetic mapping - we mapped the y-axis to the **mpg** variable

```
ggplot(mtcars,aes(x=wt)) + geom_point(aes(y=mpg))
```

Could have also started over - see how flexible ggplot can be

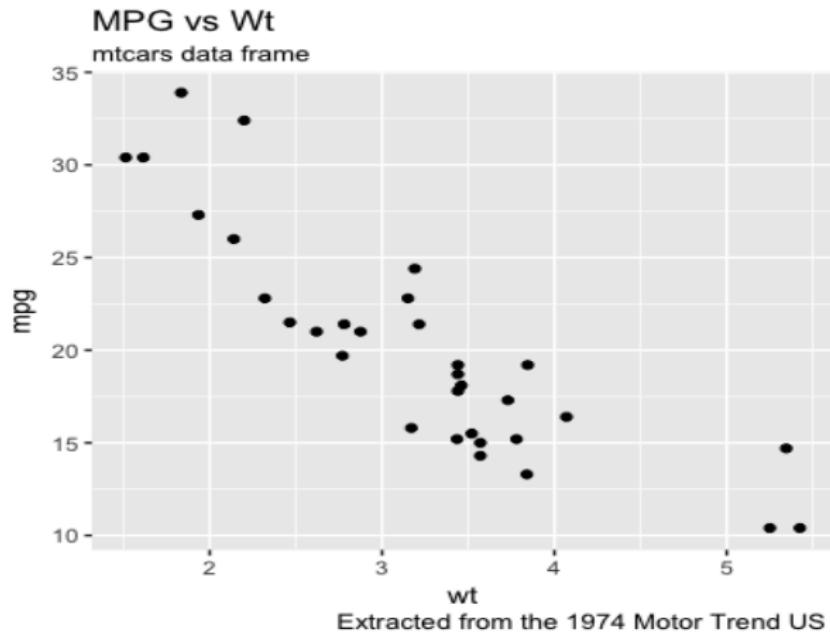
```
ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point()
```



mtcars

Is there a relationship between **mpg** vs **wt** ? We can add titles, labels, and captions

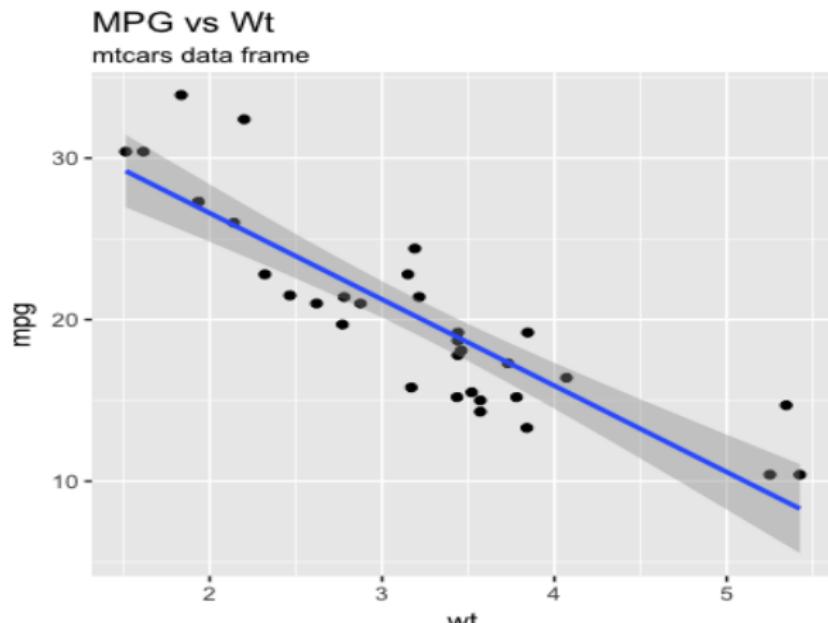
```
ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point() +  
  ggtitle("MPG vs Wt","mtcars data frame") +  
  labs(caption="Extracted from 1974 Motor Trend US")
```



mtcars

We can add another geometry. In this case a regression line with confidence intervals:

```
ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point() +  
  ggtitle("MPG vs Wt","mtcars data frame") +  
  geom_smooth(method="lm")
```



mtcars

- Does **mpg** appear to be different over individual cylinder groups ?

Before answering this question I think that it is a good time to introduce the idea of "grouping":

- For each unique value of a factor or category we can see how it impacts the plot
- We can use color, shapes, and size to accomplish this
- In ggplot we use and "aesthetic mapping" to do this
- Note that the **cyl** variable assumes only three unique values

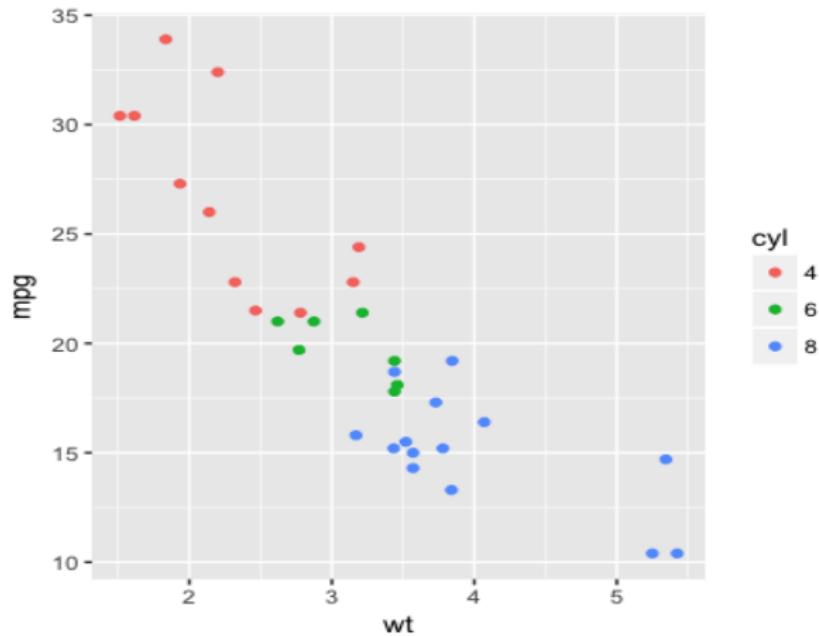
```
unique(mtcars$cyl)  
[1] 6 4 8
```

```
# Let's make cyl and "official" factor  
mtcars$cyl <- factor(mtcars$cyl)
```

mtcars

- Does **mpg** appear to be different over individual cylinder groups ?

```
ggplot(mtcars,aes(x=wt,y=mpg,color=cyl)) + geom_point()
```

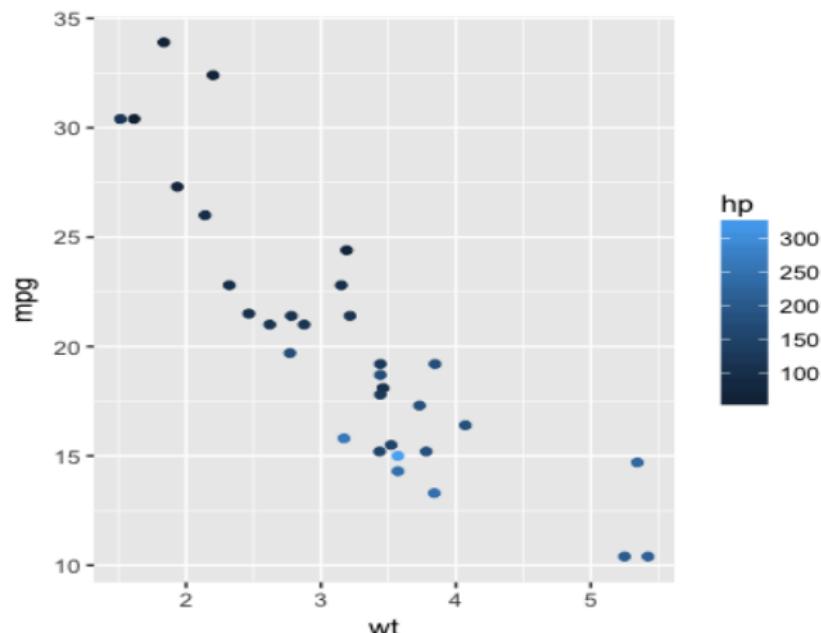


mtcars

- Does **mpg** appear to be different over individual cylinder groups ?

What happens if we use a continuous quantity as a color aesthetic ? Like **hp** ?

```
ggplot(mtcars,aes(x=wt,y=mpg,color=hp)) + geom_point()
```

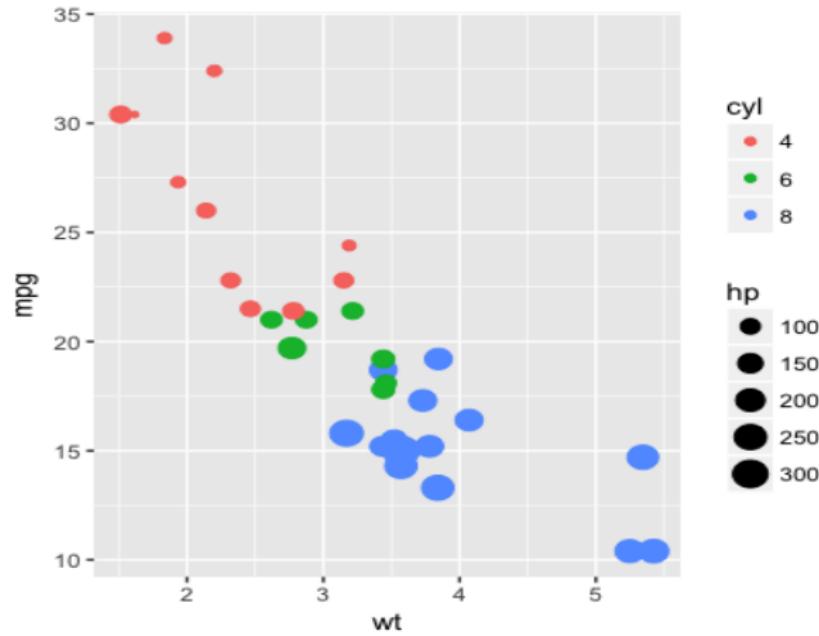


mtcars

- Does **mpg** appear to be different over individual cylinder groups ?

We can use multiple variables for grouping

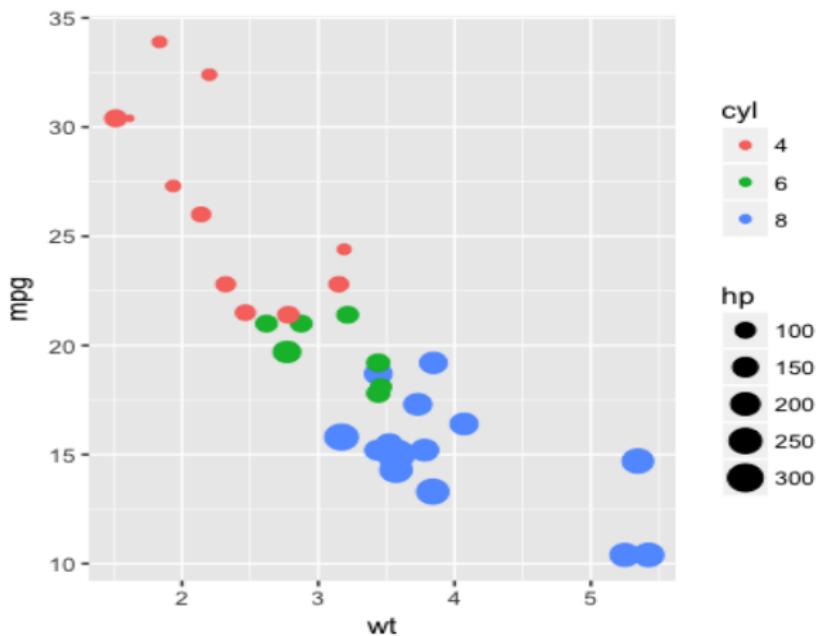
```
ggplot(mtcars,aes(x=wt,y=mpg,size=hp,color=cyl)) + geom_point()
```



mtcars

Maybe better to put aesthetic assignments in the associated **geom** layer.
This gives us flexibility

```
ggplot(mtcars) + geom_point(aes(x=wt,y=mpg,size=hp,color=cyl))
```



mtcars

- Note that there is a difference between **mappings** and **settings** with the former usually being a function of some variable in the data
- **Settings** are for altering appearance in a fixed, “set” way

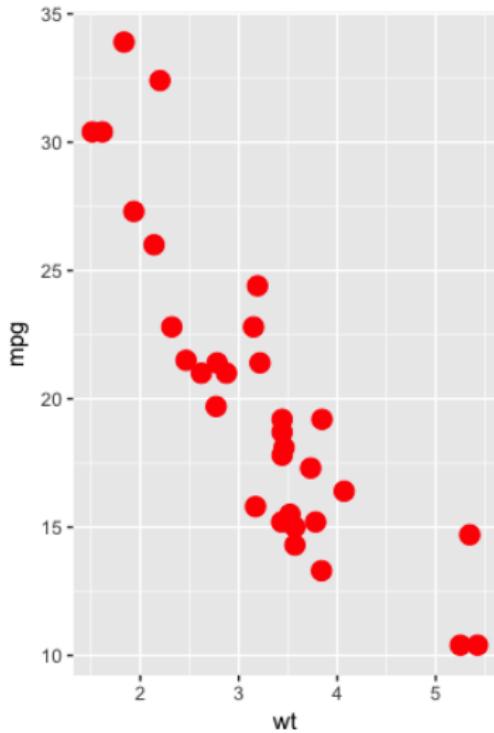
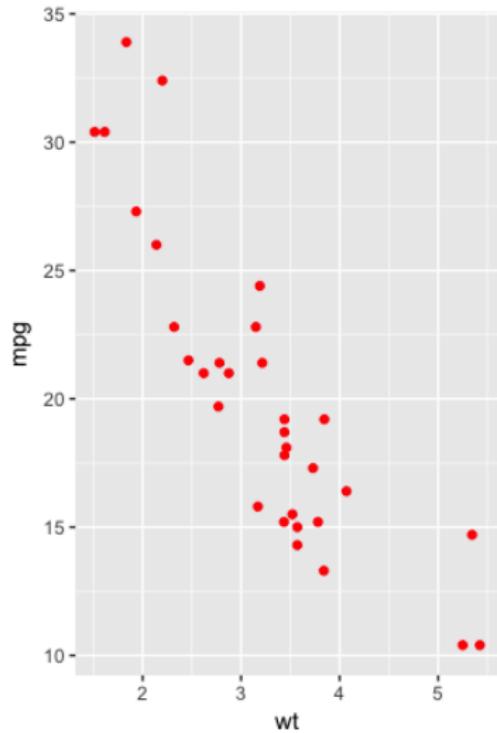
```
library(ggplot2)
library(gridExtra)

p1 <- ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point(color="red")

p2 <- ggplot(mtcars,aes(x=wt,y=mpg)) + geom_point(color="red",size=4)

grid.arrange(p1, p2, nrow=1, ncol=2)
```

mtcars



iamonds

A dataset containing the prices and other attributes of almost 54,000 diamonds. The columns/variables are as follows:

price - price in US dollars (\$326 - \$18,823)

carat - weight of the diamond (0.2 - 5.01)

cut - quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color - diamond colour, from J (worst) to D (best)

clarity - how clear the diamond is

(I1 (worst), SI1, SI2, VS1,
VS2, VVS1, VVS2, IF (best))

x - length in mm (0 - 10.74)

y - width in mm (0 - 8.9)

z - depth in mm (0 - 31.8)

depth - total depth percentage

table - width of top of diamond relative to widest point (43 - 95)

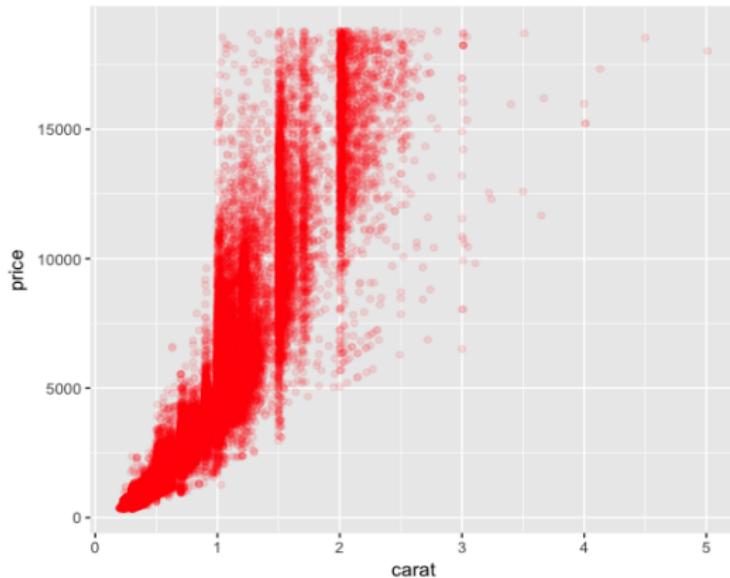
diamonds

```
> str(diamonds)
Classes tbl_df, tbl and data.frame: 53940 obs. of 10 variables:
$ carat   : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
$ cut      : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
$ color    : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
$ clarity  : Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
$ depth    : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
$ table    : num  55 61 65 58 58 57 57 55 61 61 ...
$ price    : int  326 326 327 334 335 336 336 337 337 338 ...
$ x        : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
$ y        : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
$ z        : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

diamonds

Let's look at a more "dense" data set. Using a fixed color setting along with a transparency factor can make obvious certain groups in the data.

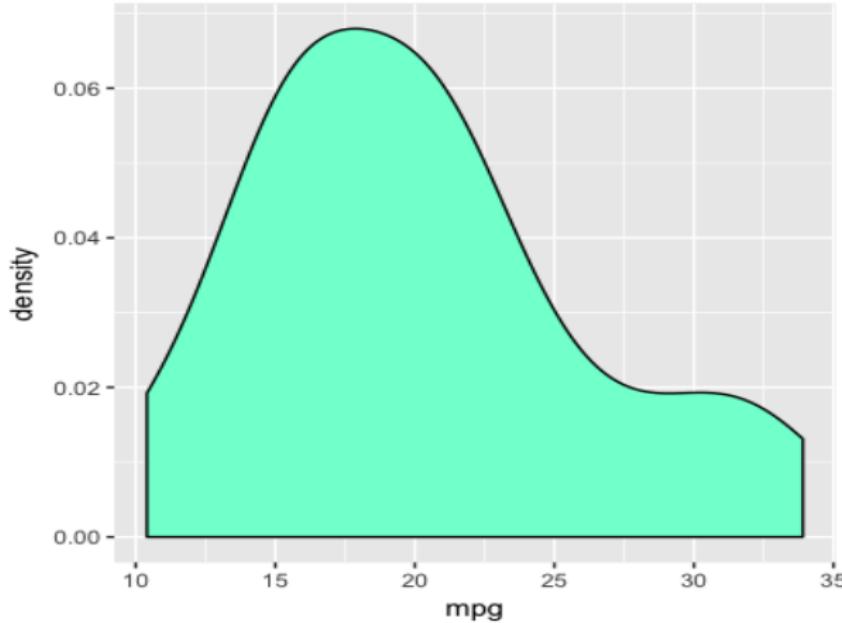
```
data(diamonds)
ggplot(diamonds,aes(x=carat,y=price)) +
  geom_point(color="red",alpha=0.1)
```



mtcars

In general anything you wish to set to a static value should be set **outside** of the **aes** function

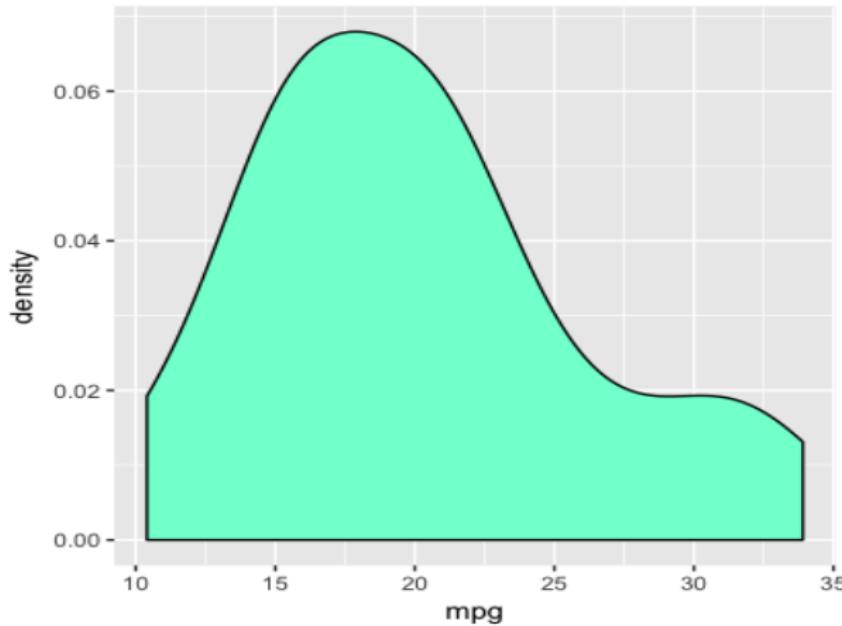
```
ggplot(mtcars) + geom_density(aes(x=mpg), fill="aquamarine")
```



mtcars

Here we create a density plot of MPG. Note with a static color assignment we place it **outside** of the **aes**

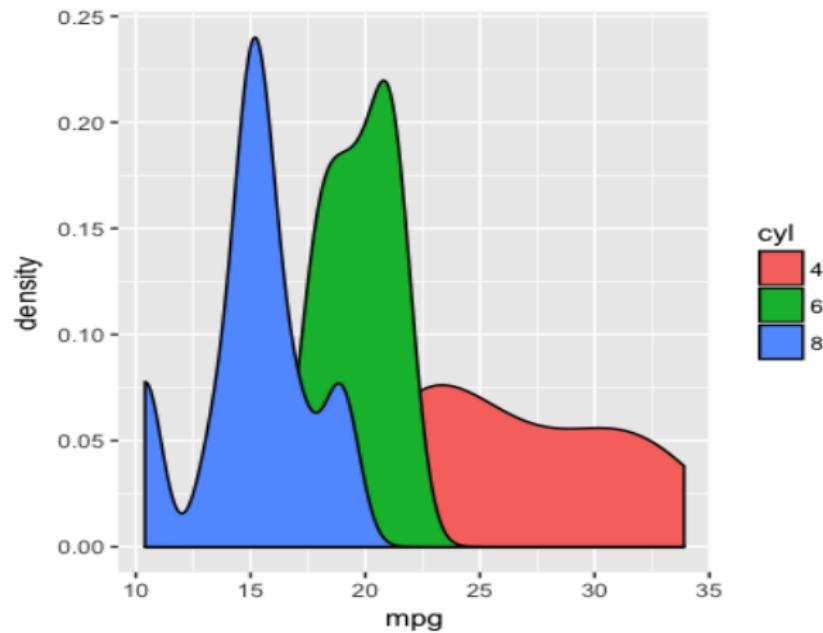
```
ggplot(mtcars) + geom_density(aes(x=mpg), fill="aquamarine")
```



mtcars

Here we create a density plot of MPG. We can group the density by cyl

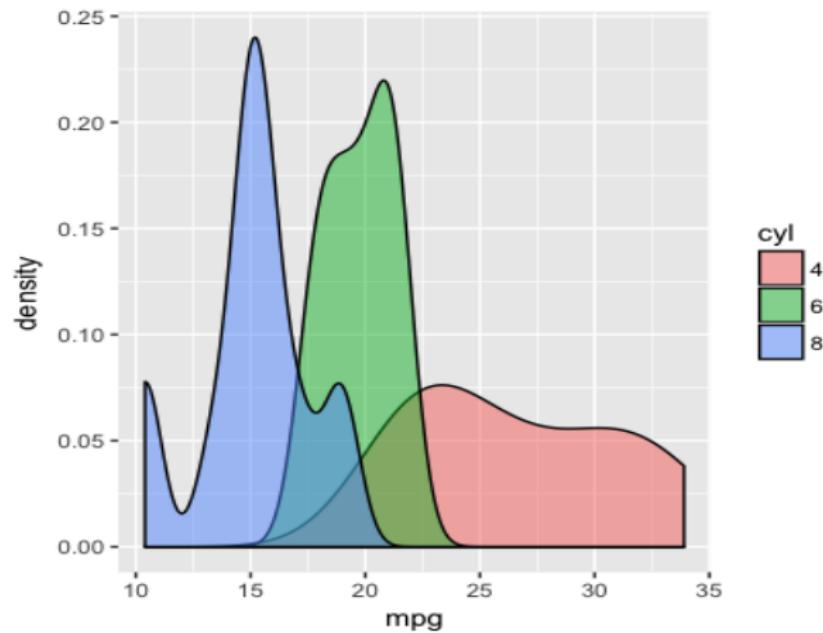
```
ggplot(mtcars) + geom_density(aes(x=mpg, fill=cyl))
```



mtcars

What about looking at a boxplot of **mpg** across cylinder groups ?

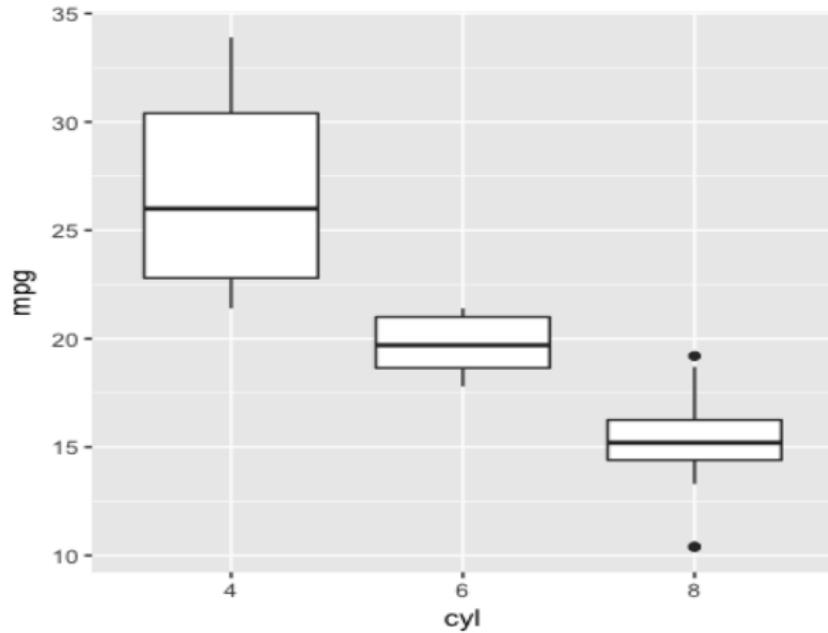
```
ggplot(mtcars) + geom_boxplot(aes(x=cyl,y=mpg),alpha=0.5)
```



mtcars

What about looking at a boxplot of **mpg** across cylinder groups ?

```
ggplot(mtcars) + geom_boxplot(aes(x=cyl,y=mpg))
```



mtcars

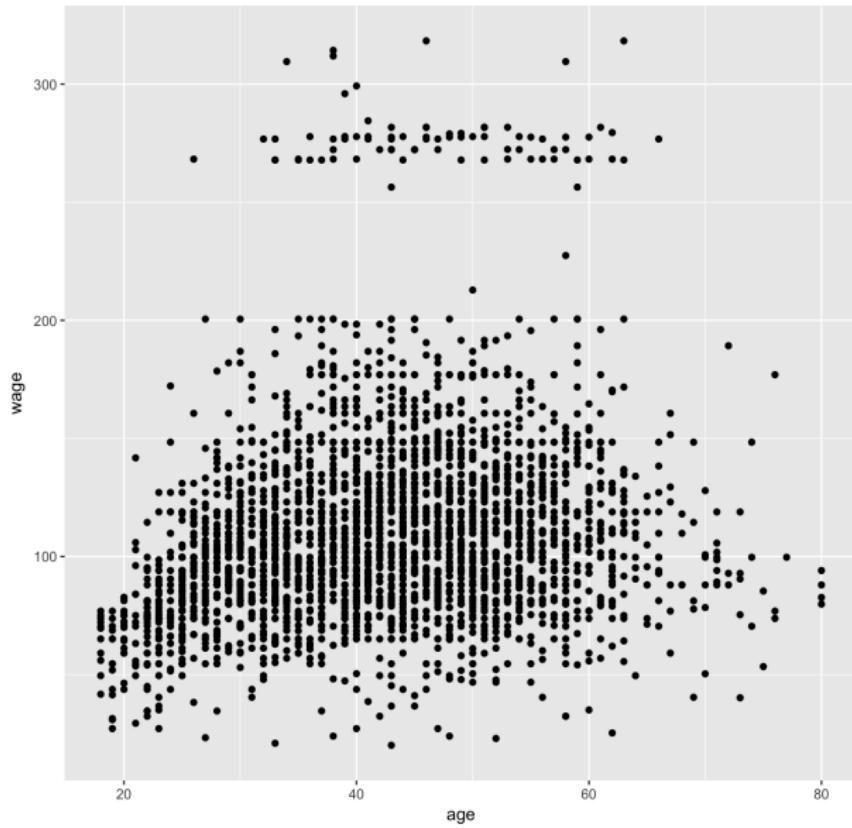
Wage data

```
url <- "https://raw.githubusercontent.com/pittardsp/bios545r_spring_2018/master/SUPPORT/wage.csv"

wages <- read.csv(url)
ggplot(wages,aes(x=age,y=wage)) +
    geom_point() + ggtitle("Wage vs Age")
```

mtcars

Wage vs Age

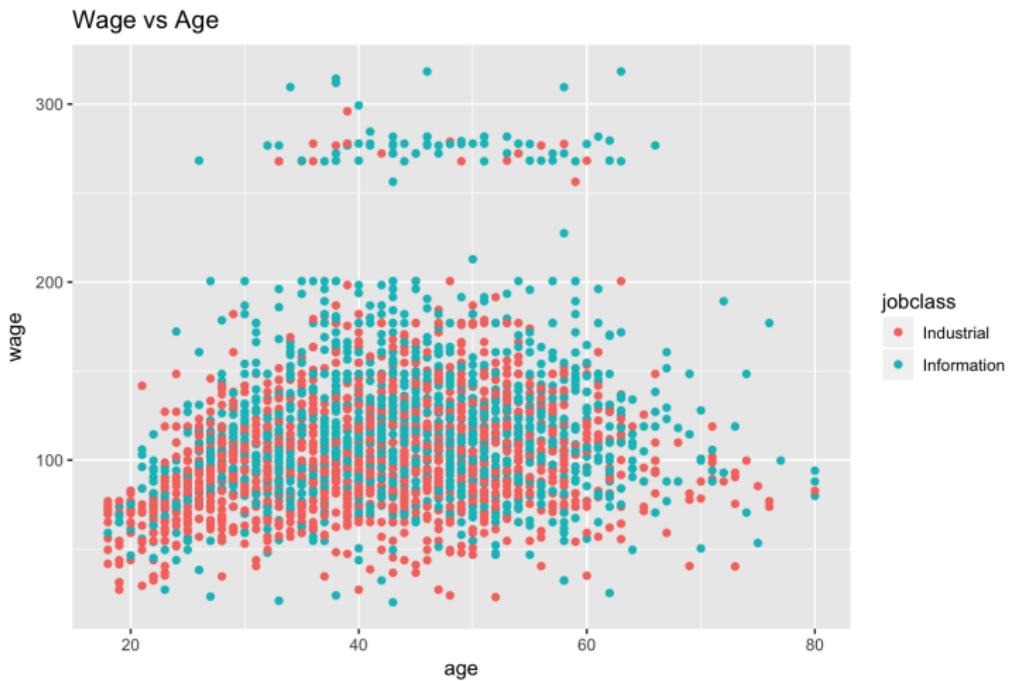


mtcars

Wage data

```
url <- "https://raw.githubusercontent.com/pittardsp/bios545r_spring_2018/master/SUPPORT/wage.csv"  
  
wages <- read.csv(url)  
ggplot(wages,aes(x=age,y=wage,color=jobclass)) +  
  geom_point() + ggtitle("Wage vs Age")
```

mtcars



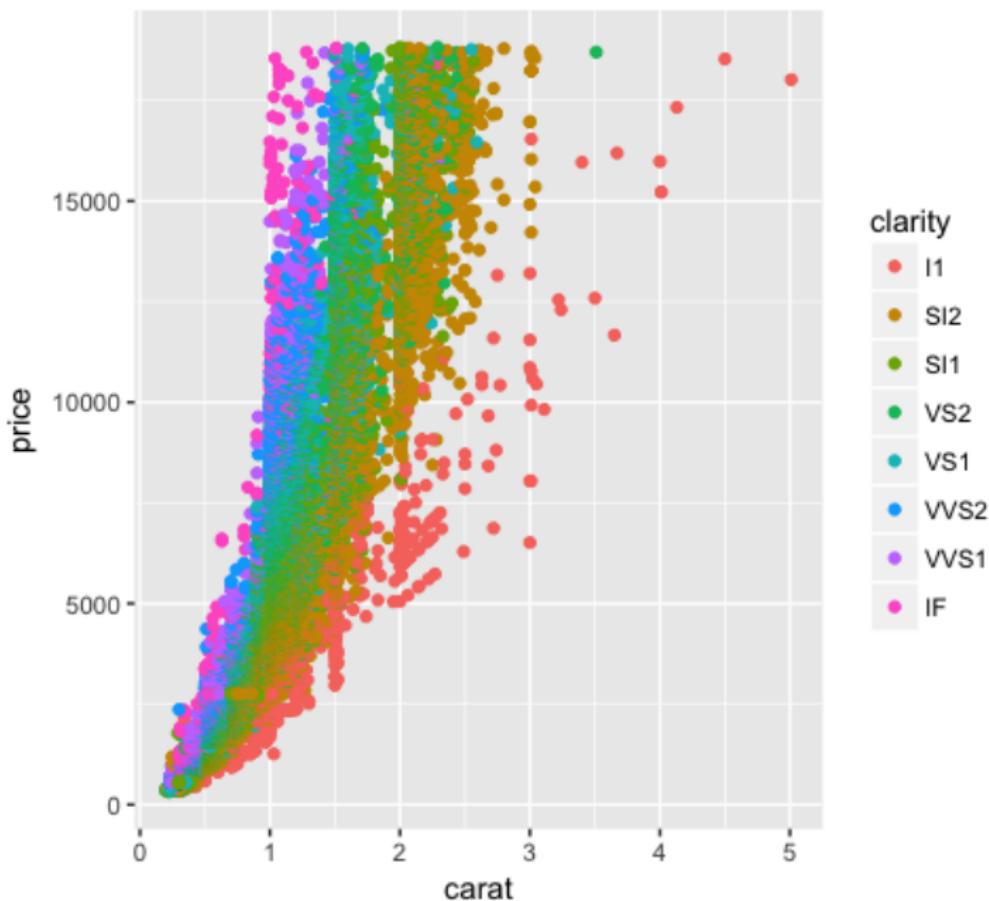
Facets

Facets allow us to display data within panels to facilitate easy comparisons

- In lattice graphics this is called panelling or conditioning
- In Base graphics you have to do panelling or faceting manually - it's a pain
- Consider the following grouping graph:

```
ggplot(diamonds,aes(x=carat,y=price)) +  
    geom_point(aes(color=clarity)) +  
    ggtitle("Price vs Carat Size")
```

Price vs Carat Size

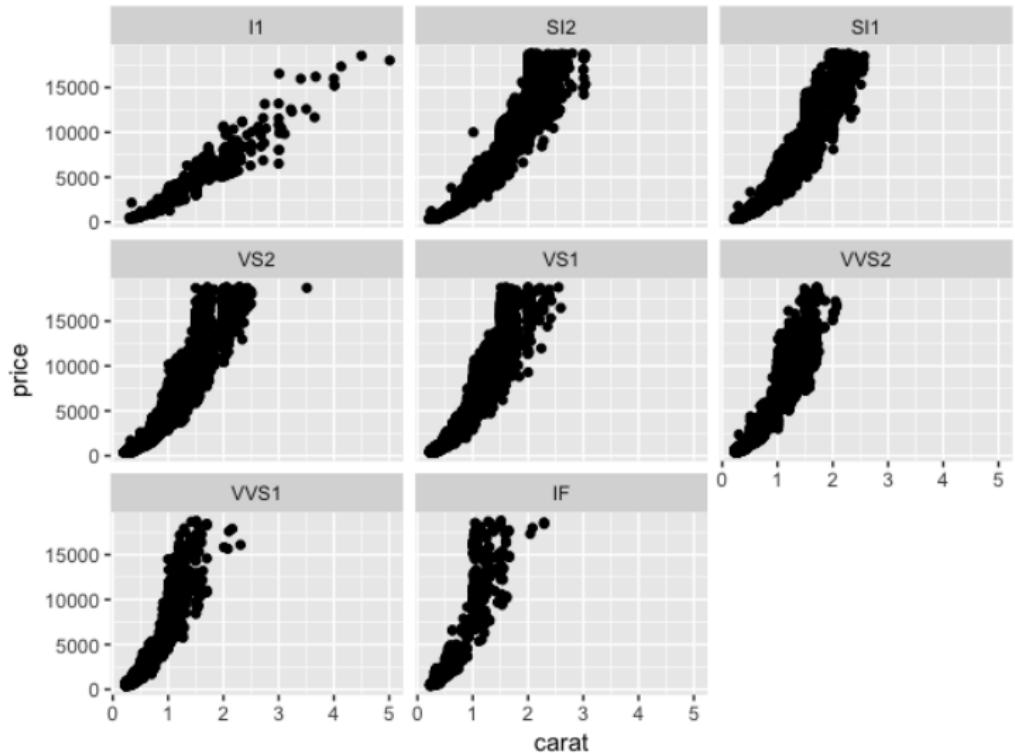


Facets

Here we will plot the same data that was formerly grouped. By splitting the data into separate panels corresponding to each value of the clarity variable we might be able to see interesting trends more easily.

```
ggplot(diamonds,aes(x=carat,y=price)) +  
  geom_point() +  
  facet_wrap(~clarity)
```

Facets

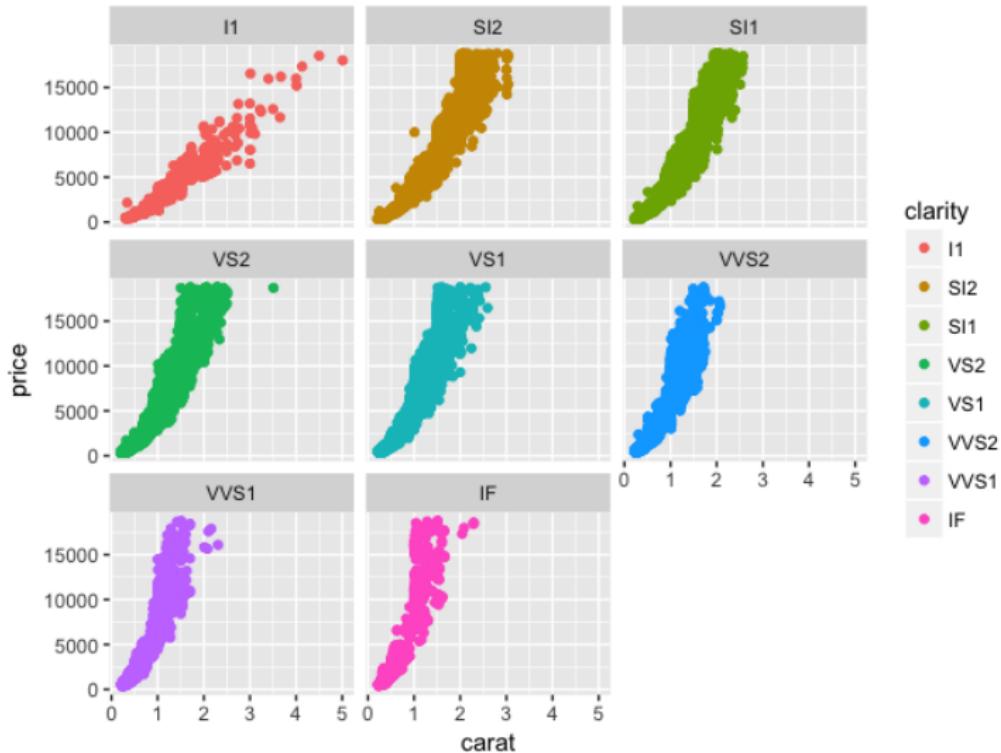


Facets

Here we will plot the same data that was formerly grouped. However we will still preserve the grouping colors. While this isn't essential it might be helpful

```
ggplot(diamonds,aes(x=carat,y=price)) +  
  geom_point(aes(color=clarity)) +  
  facet_wrap(~clarity)
```

Facets



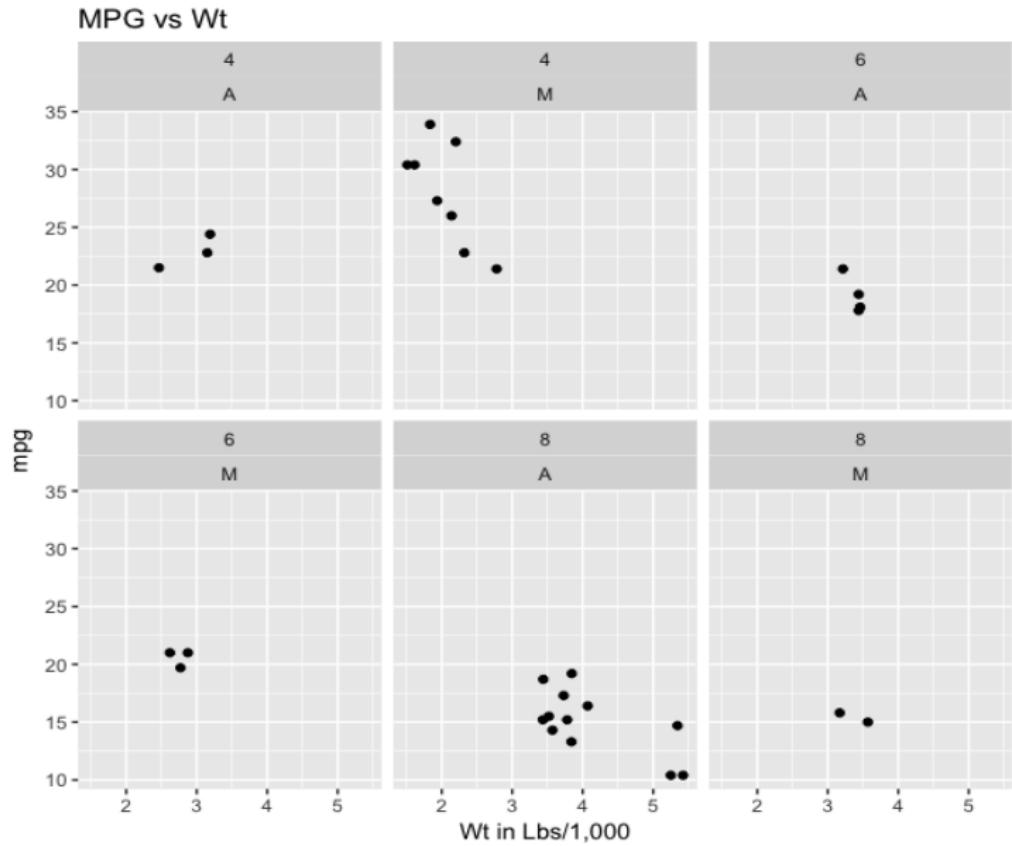
Facets

Let's go back to mtcars

- plot mpg vs wt for all combinations of cylinder group and Transmission Type
- Let's make these variables into factors
- What combination gets the best MPG ?

```
mtcars <- transform(mtcars, am=factor(am,labels=c("A", "M")),  
                      cyl=factor(cyl))  
  
ggplot(mtcars,aes(x=wt,y=mpg)) +  
    geom_point() +  
    facet_wrap(~cyl+am) +  
    ggtitle("MPG vs Wt") +  
    xlab("Wt in Lbs/1,000")
```

Facets



Count and Table

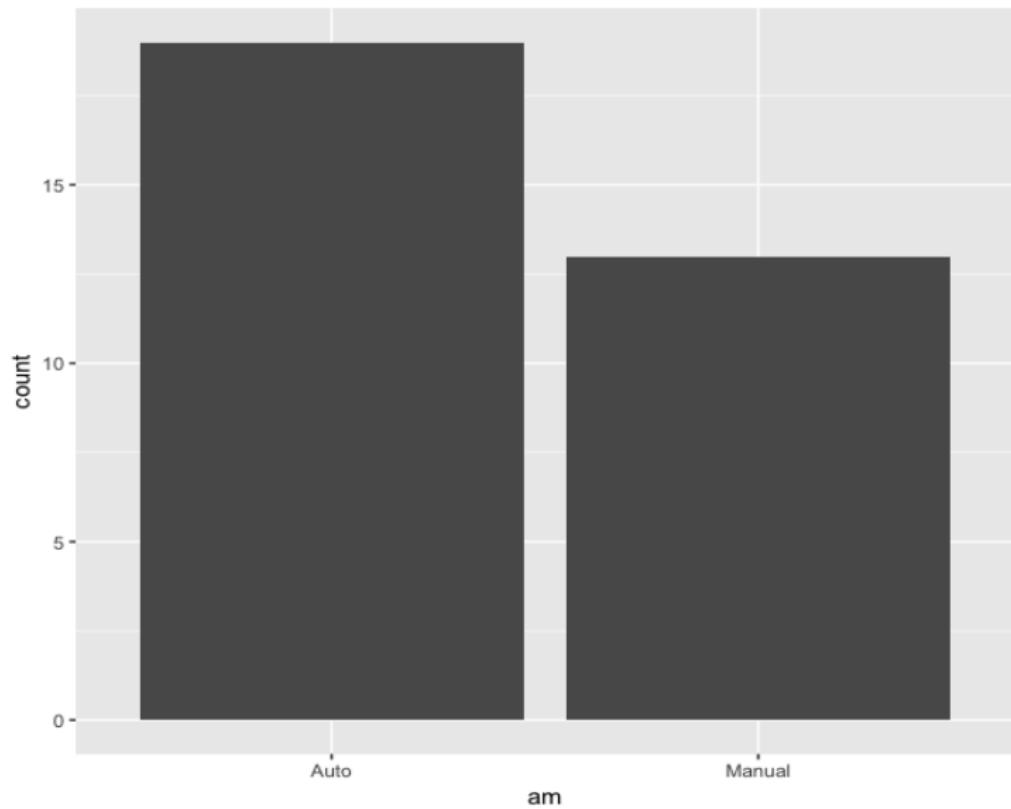
- What about tabular data ?
- ggplot ususally prefers data frames as input
- aggregation commands in R produce tables
- If you have the dataframe used to produce the table then use that
- Consider the following:

```
# A barplot of cars by Transmission Type
```

```
mtcars$am <- factor(mtcars$am, labels=c("Auto","Manual"))
ggplot(mtcars,aes(x=am)) + geom_bar() +
  ggtitle("Distribution of Transmission Types")
```

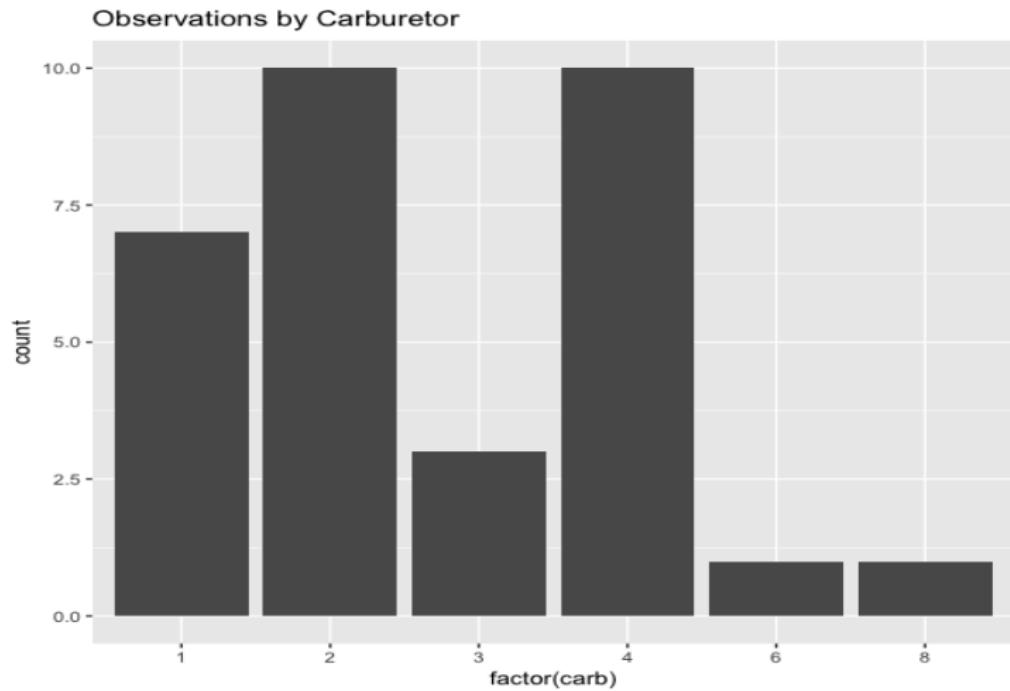
Count and Tabular Data

Distribution of Transmission Types



Count and Tabular Data

```
ggplot(mtcars,aes(x=factor(carb))) + geom_bar() +  
  ggttitle("Observations by Carburetor")
```



Count and Tabular Data

What if you are starting with a table ?

```
(ctab <- table(carb=mtcars$carb))
carb
 1  2  3  4  6  8
7 10  3 10  1  1

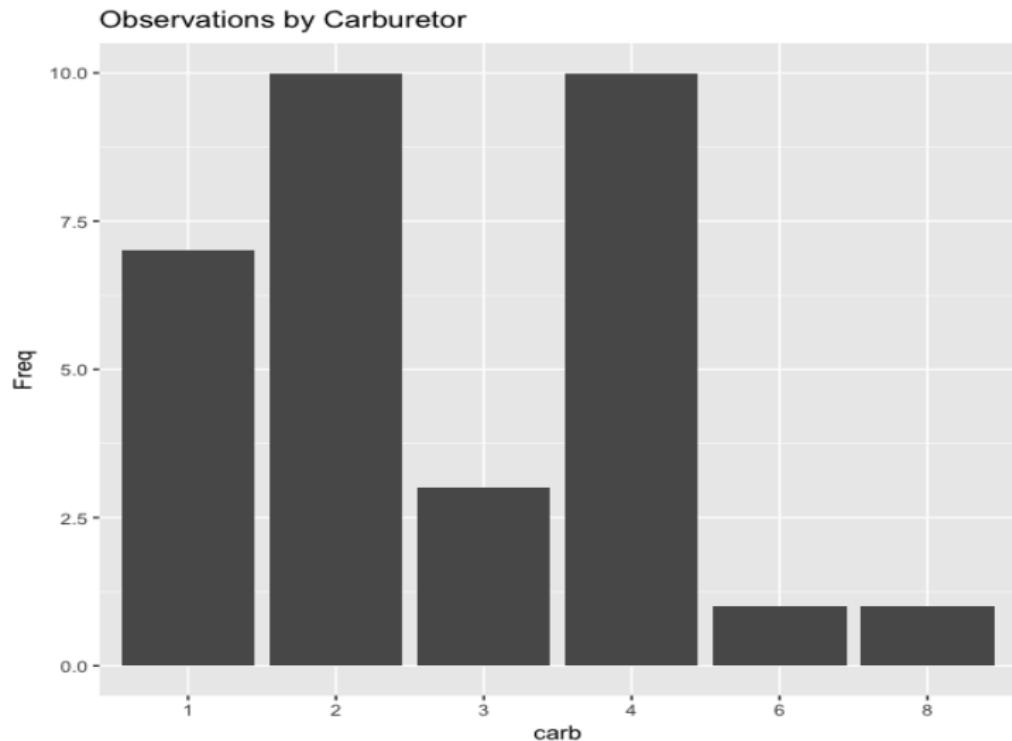
# Remember that ggplot wants a data frame

(df <- as.data.frame(ctab))

ggplot(df,aes(x=carb,y=Freq)) + geom_bar(stat="identity") +
  ggttitle("Observations by Carburetor")
```

Count and Tabular Data

What if you are starting with a table ?



Count and Tabular Data

What if you are starting with a table ? How to order the bars ?

```
(ctab <- table(carb=mtcars$carb))
```

carb

1	2	3	4	6	8
7	10	3	10	1	1

```
# Remember that ggplot wants a data frame. We also use a "stat" of  
# "identity" since we are using a pre-existing table that already has  
# count information.
```

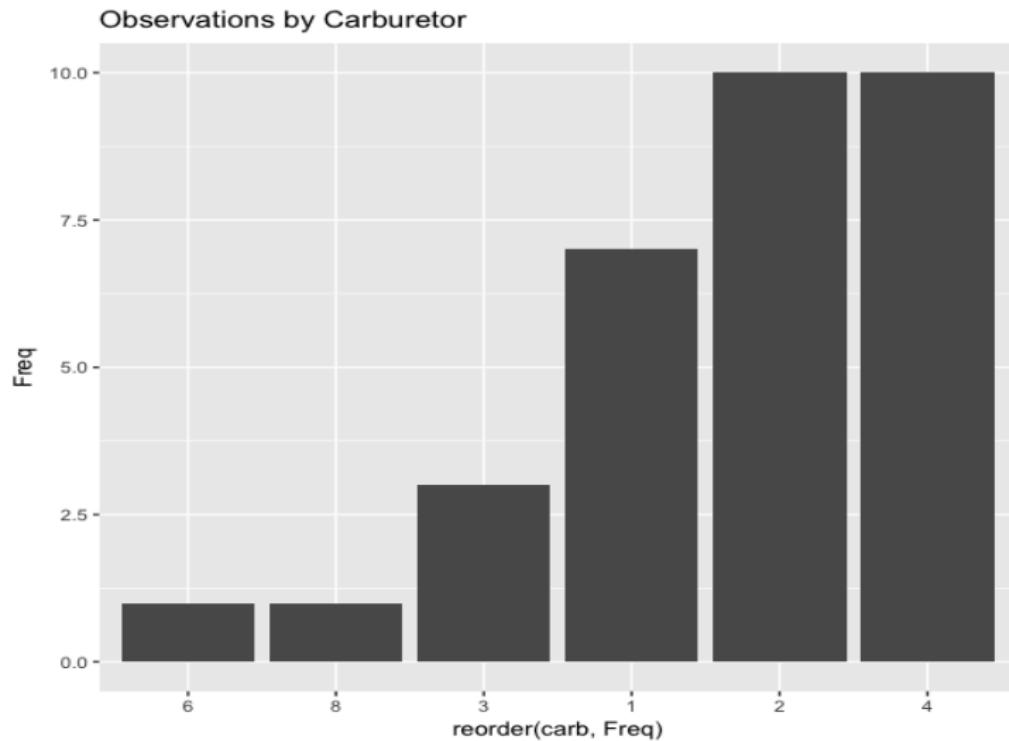
```
# geom_bar uses "stat_count" by default which does the counting for you.  
# But we already have the counts in our table !
```

```
(df <- as.data.frame(ctab))
```

```
ggplot(df,aes(x=reorder(carb,Freq),y=Freq)) + geom_bar(stat="identity") +  
  ggtitle("Observations by Carburetor")
```

Count and Tabular Data

What if you are starting with a table ? How to order the bars ?



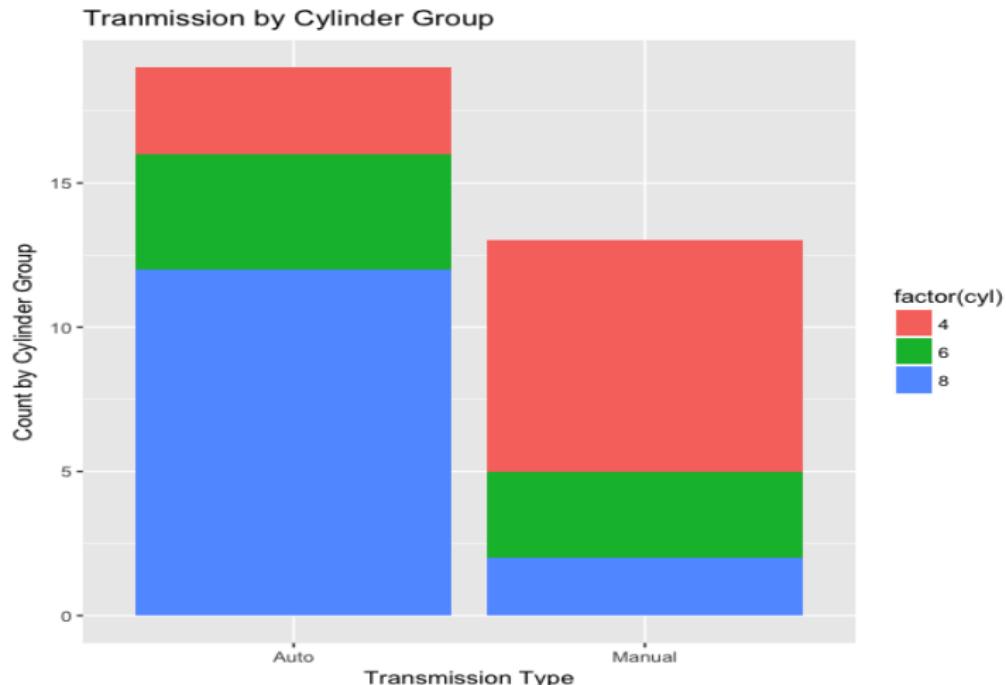
Count and Tabular Data

- If you want to look at a 2-way count table then you can use the **fill** aesthetic.
- This is one of those things that actually winds up being easier in Base graphics
- Although once you learn the “grammar of graphics” this becomes easier
- This relates mostly to table and count data

```
ggplot(mtcars,aes(x=am)) + geom_bar(aes(fill=factor(cyl))) +  
  ggtitle("Transmission by Cylinder Group") +  
  xlab("Transmission Type") +  
  ylab("Count by Cylinder Group")
```

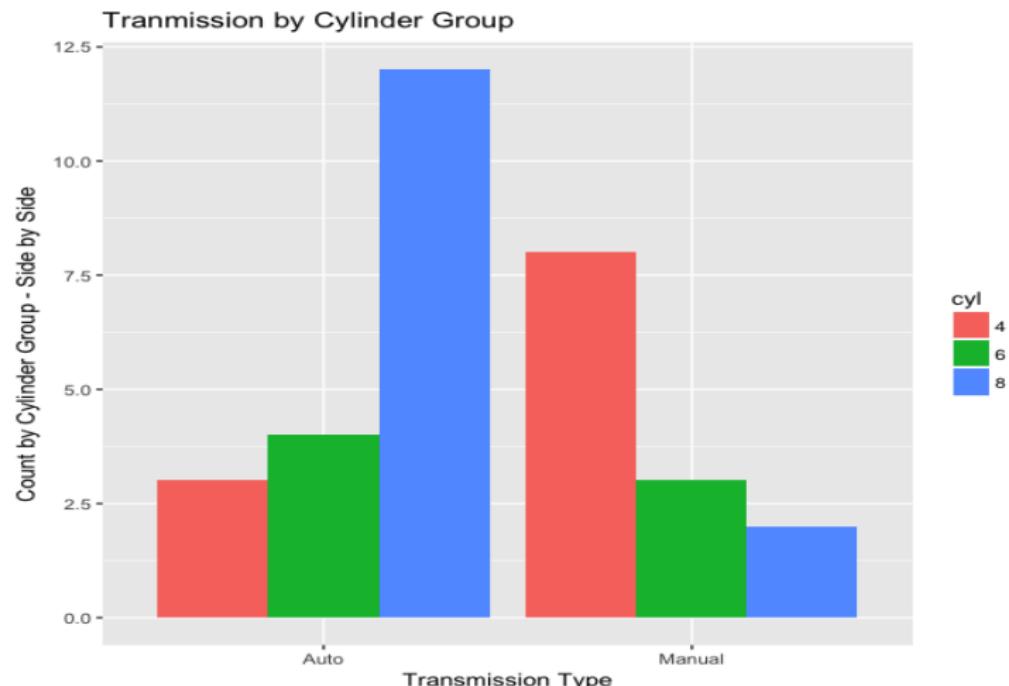
Count and Tabular Data

```
ggplot(mtcars,aes(x=am)) + geom_bar(aes(fill=factor(cyl))) +  
  ggtitle("Transmission by Cylinder Group") +  
  xlab("Transmission Type") +  
  ylab("Count by Cylinder Group")
```



Count and Tabular Data

```
mtcars <- transform(mtcars, am=factor(am,labels=c("Auto","Manual")), cyl=factor(cyl))  
ggplot(mtcars,aes(x=am)) + geom_bar(aes(fill=cyl),position="dodge") +  
  ggtitle("Transmission by Cylinder Group") +  
  xlab("Transmission Type") +  
  ylab("Count by Cylinder Group - Side by Side")
```



Count and Tabular Data

But what if you have just the table without data from whence it came ?

```
sometable
```

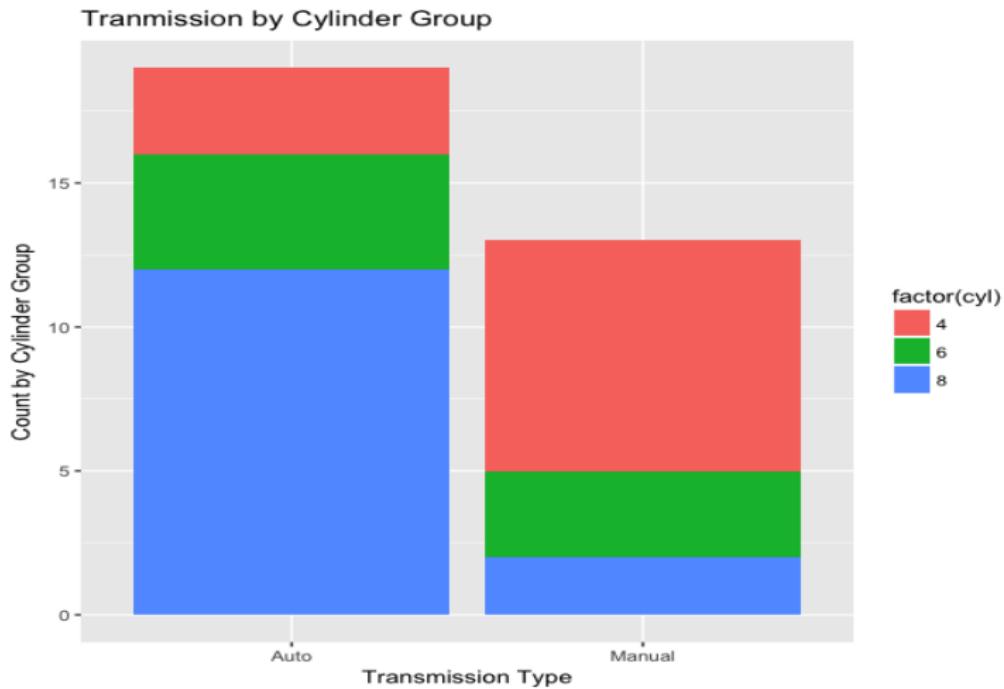
	cyl
--	-----

transmission	4	6	8
Auto	3	4	12
Manual	8	3	2

```
df <- as.data.frame(sometable)
```

```
ggplot(df,aes(transmission,Freq,fill=cyl)) +  
    geom_bar(stat="identity") +  
    ggtitle("Transmission by Cylinder Group") +  
    xlab("Transmission Type") +  
    ylab("Count by Cylinder Group")
```

Count and Tabular Data

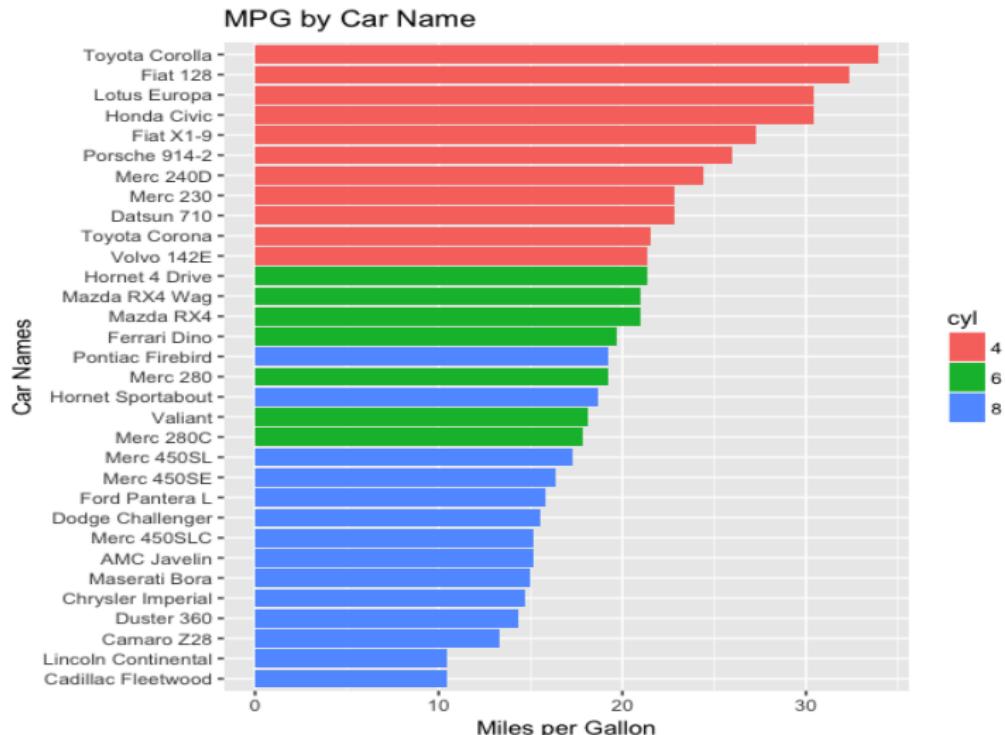


Count and Tabular Data

Here is how to list the cars in mtcars by MPG from best to worst

```
# Make Cylinder into a factor  
  
mtcars$cyl <- factor(mtcars$cyl,labels=seq(4,8,2))  
  
# Use reorder to arrange the data from high MPG to low  
  
ggplot(mtcars, aes(x = reorder(row.names(mtcars), mpg),  
                    y = mpg)) +  
  geom_bar(aes(fill = cyl),stat = "identity") +  
  coord_flip() +  
  xlab("Car Names") + ylab("Miles per Gallon") +  
  ggtitle("MPG by Car Name")
```

Count and Tabular Data



ggmaps - Maps

The following material adopted from Data Camp course on Spatial Data

ggmaps - Maps

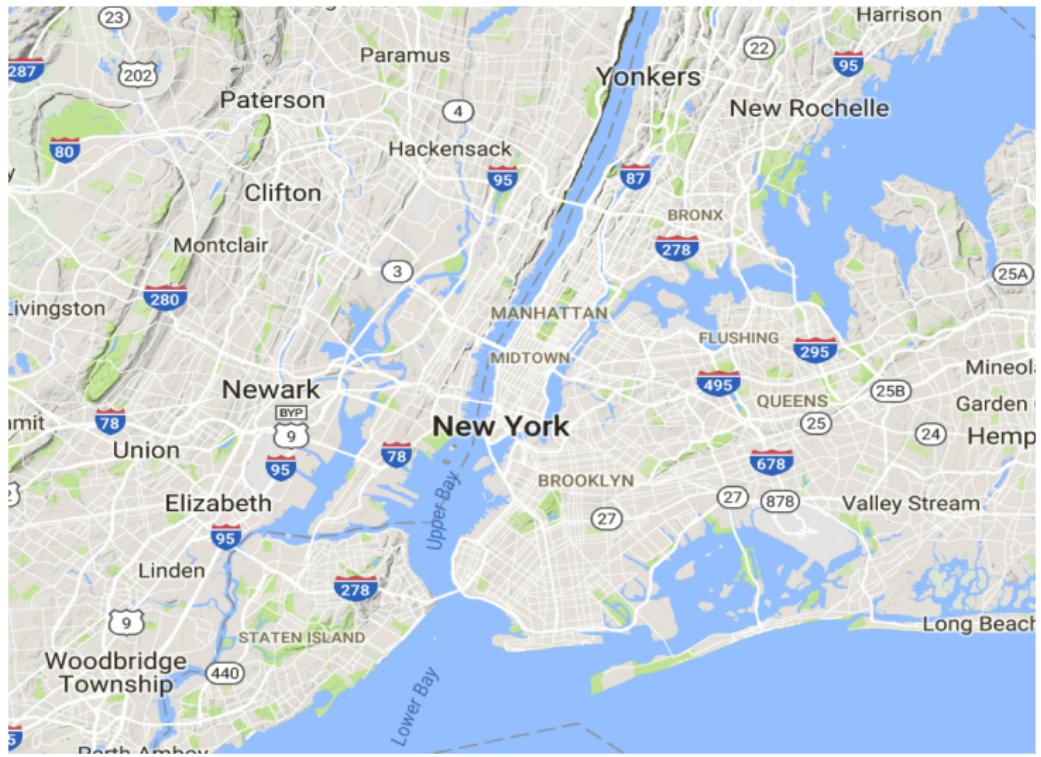
ggplot has a companion package called ggmap which helps us with maps.
It conforms to the philosophy adhered to in the ggplot package

```
# Get the Lat / Lon pair
nyc <- c(lon = -74.0059, lat = 40.7128)

# Get the Map and set a zoom leve
nyc_map <- get_map(location = nyc, zoom = 10)

# Actually display the map
ggmap(nyc_map)
```

ggmap



ggmaps - Maps

ggplot has a companion package called ggmap which helps us with maps. It conforms to the philosophy adhered to in the ggplot package

```
atlanta <- geocode("1510 Clifton Rd Atlanta, GA 30322")
```

```
# Get map at zoom level 5: map_5
```

```
map_5 <- get_map(atlanta, zoom = 5, scale = 1)
```

```
# Plot map at zoom level 5
```

```
ggmap(map_5)
```

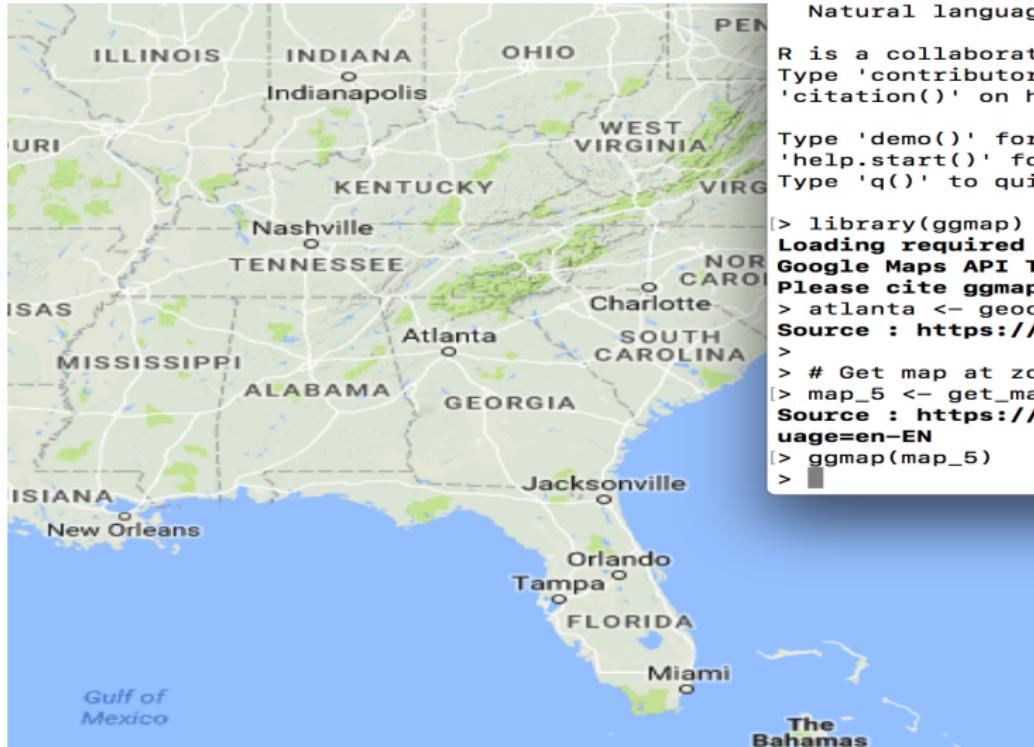
```
# Get map at zoom level 13: atlanta_map
```

```
atlanta_map <- get_map(atlanta, zoom = 13, scale = 1)
```

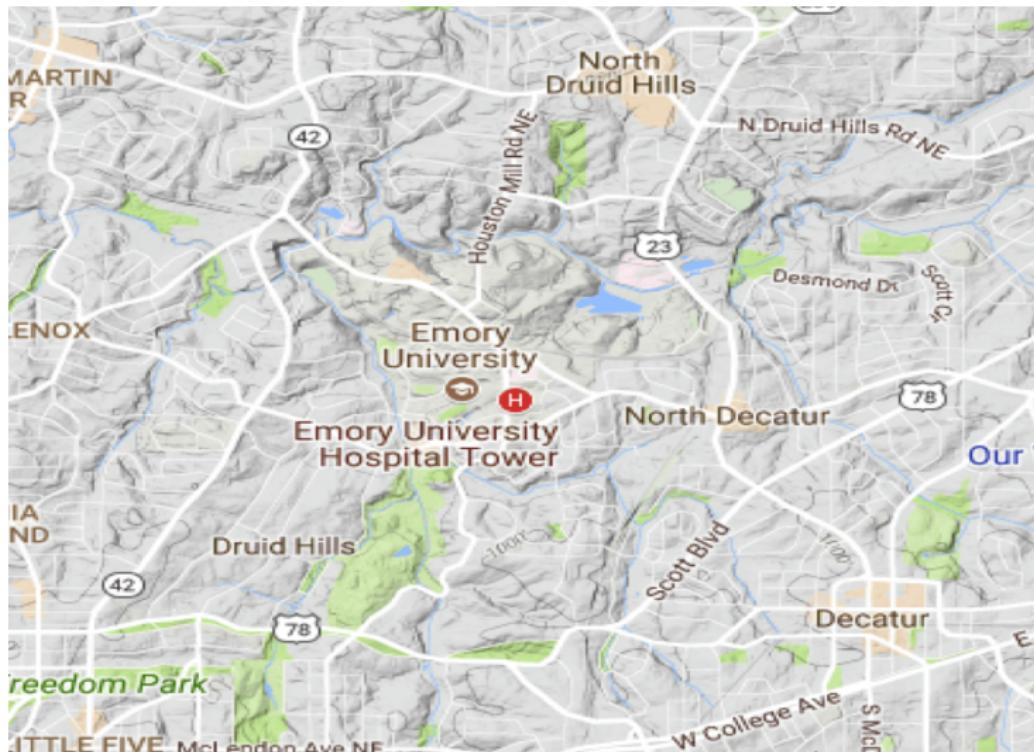
```
# Plot map at zoom level 13
```

```
ggmap(atlanta_map)
```

ggmap



ggmap



ggmaps - Maps

ggplot has a companion package called ggmap which helps us with maps.
It conforms to the philosophy adhered to in the ggplot package

```
corvallis <- c(lon = -123.2620, lat = 44.5646)
```

```
# Get map at zoom level 5: map_5
```

```
map_5 <- get_map(corvallis, zoom = 5, scale = 1)
```

```
# Plot map at zoom level 5
```

```
ggmap(map_5)
```

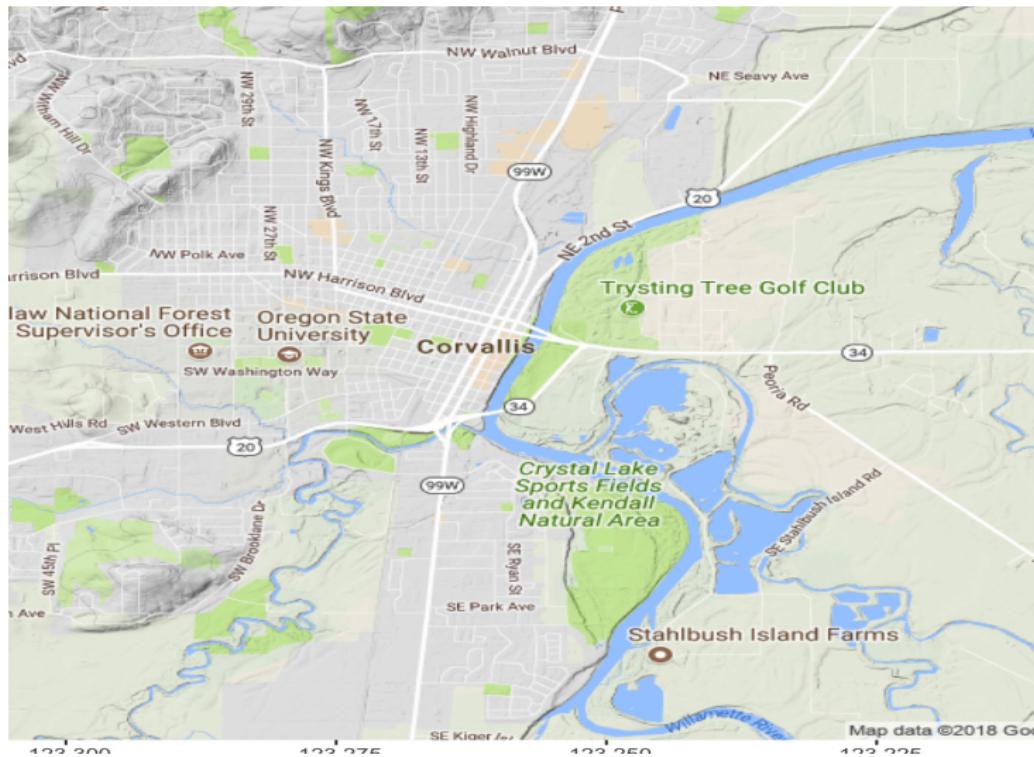
```
# Get map at zoom level 13: corvallis_map
```

```
corvallis_map <- get_map(corvallis, zoom = 13, scale = 1)
```

```
# Plot map at zoom level 13
```

```
ggmap(corvallis_map)
```

ggmap



ggmaps - Maps

We can put points on top of this map:

```
url <- "https://assets.datacamp.com/production/course_1816/datasets/01_corv_sales.rds", "corv_sales.rds"

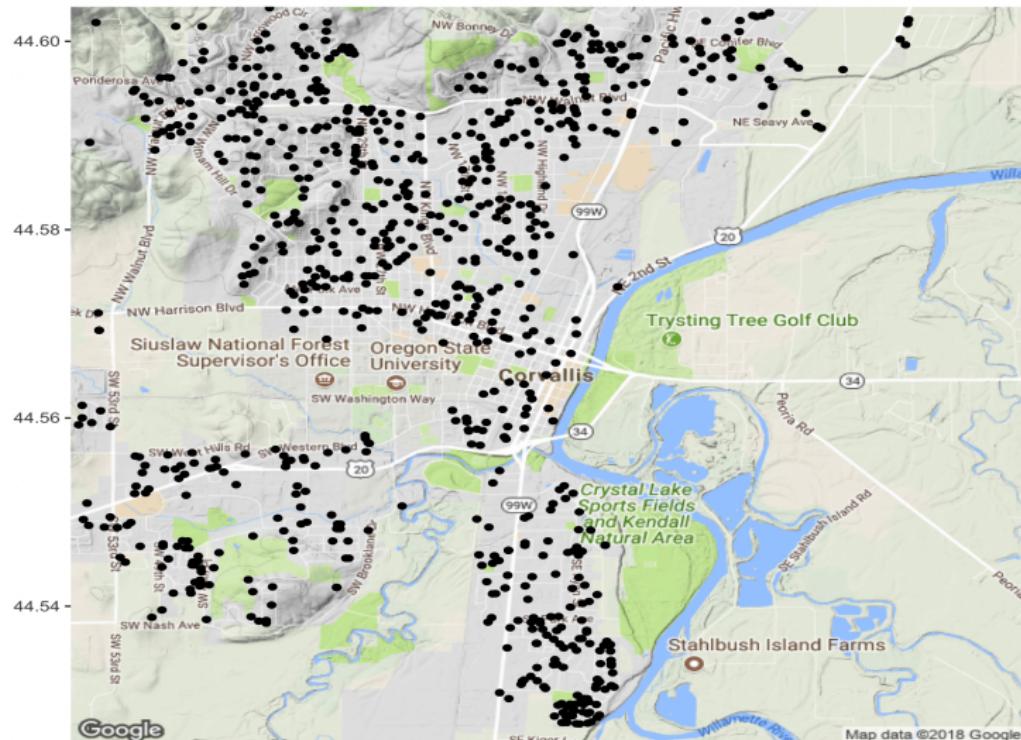
if (!file.exists("corv_sales.rds")) {
  download.file(url,"corv_sales.rds")
}

sales <- readRDS("corv_sales.rds")
head(sales)

# Swap out call to ggplot() with call to ggmap()

ggmap(corvallis_map) +
  geom_point(aes(lon, lat), data = sales)
```

ggmap



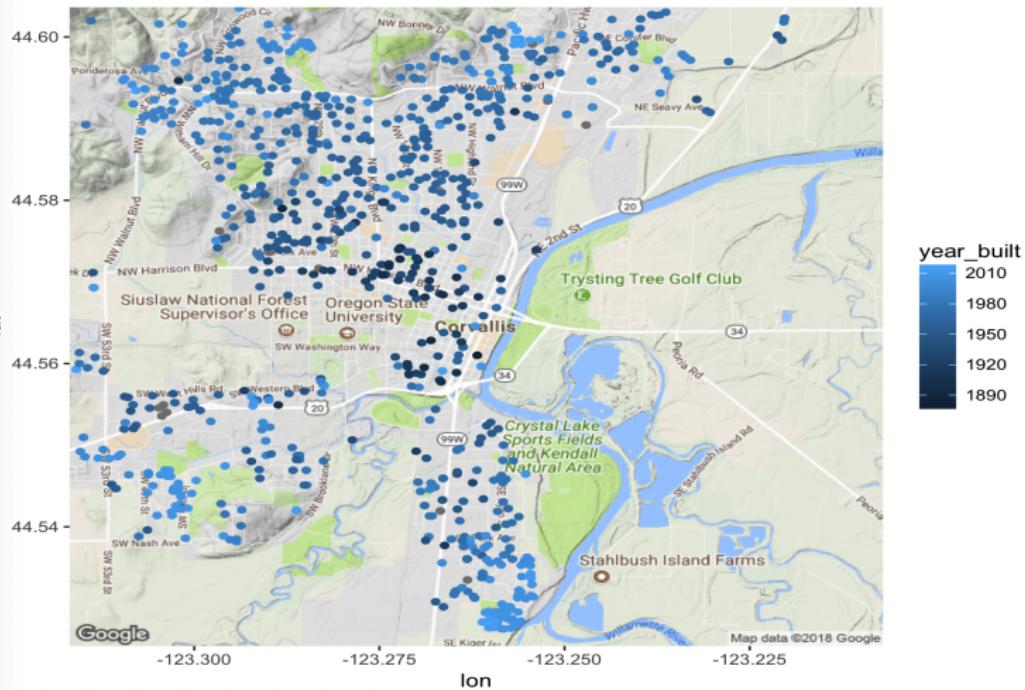
ggmaps - Maps

- Map the color of the points to year built.
- Map the size of the points to bedrooms.
- Map the color of the points to price per squarefoot
- That is the price / finished_squarefeet).
- Are there areas with better "value" than others?

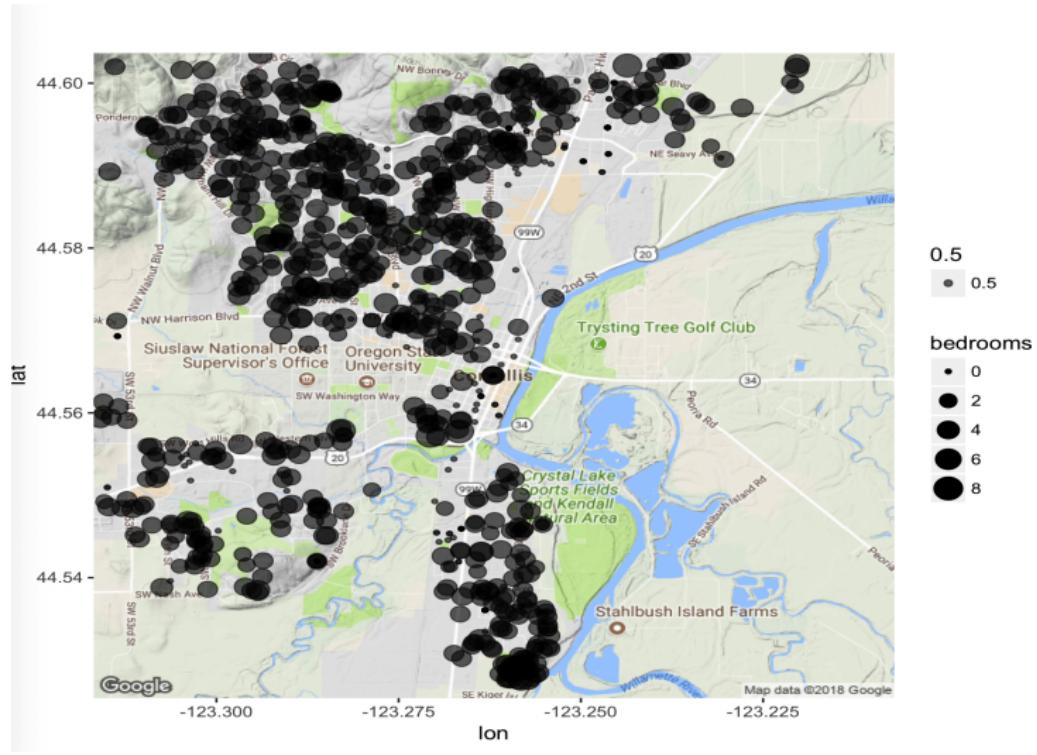
ggmaps - Maps

```
# Map color to year_built  
ggmap(corvallis_map) +  
  geom_point(aes(lon, lat, color=year_built), data = sales)  
  
# Map size to bedrooms  
ggmap(corvallis_map) +  
  geom_point(aes(lon, lat, size=bedrooms, alpha=.5), data = sales)  
  
# Map color to price / finished_squarefeet  
ggmap(corvallis_map) +  
  geom_point(aes(lon, lat, color=price/finished_squarefeet),  
             data = sales)
```

ggmap



ggmap



ggmaps - Maps

There are many different map types as well as sources

```
corvallis <- c(lon = -123.2620, lat = 44.5646)
```

```
# Add a maptype argument to get a satellite map
```

```
corvallis_map_sat <- get_map(corvallis, zoom = 13,  
                               maptype = "satellite")
```

```
# Edit to display satellite map
```

```
ggmap(corvallis_map_sat) +  
  geom_point(aes(lon, lat, color = year_built), data = sales)
```

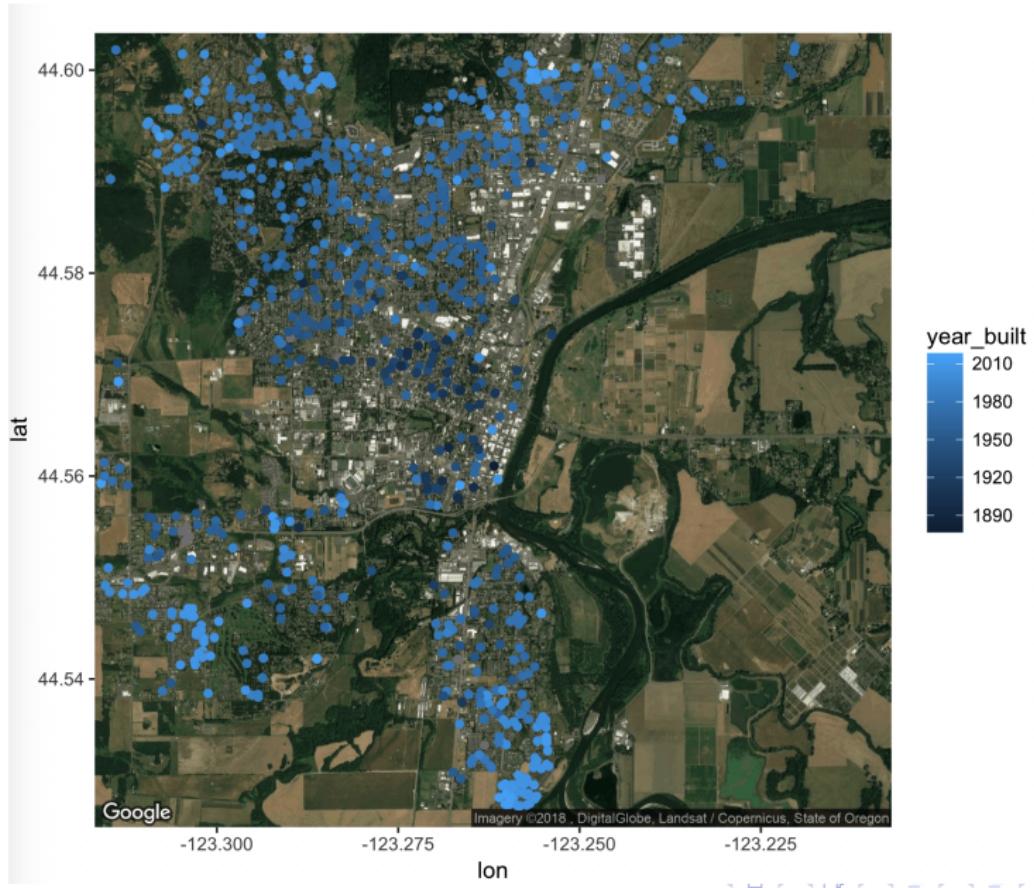
```
# Add source and maptype to get toner map from Stamen Maps
```

```
corvallis_map_bw <- get_map(corvallis, zoom = 13, source = "stamen",  
                           maptype = "toner")
```

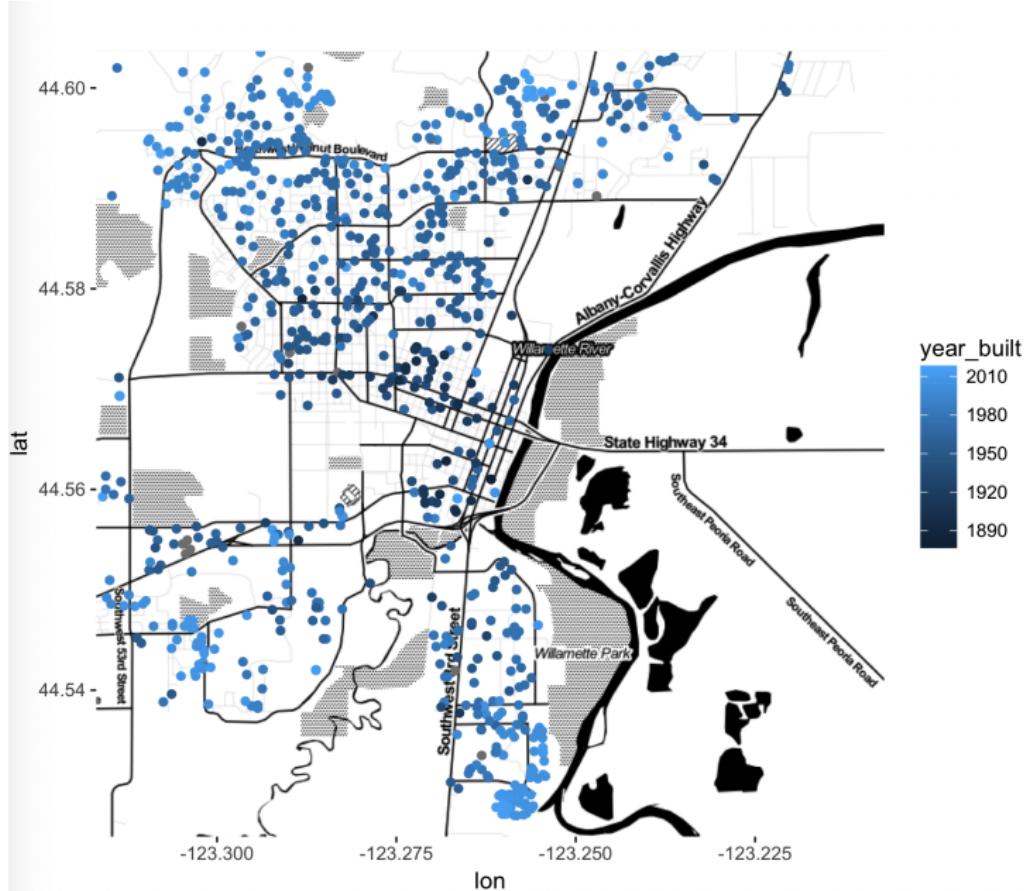
```
# Edit to display toner map
```

```
ggmap(corvallis_map_bw) +  
  geom_point(aes(lon, lat, color = year_built), data = sales)
```

ggmap



ggmap



ggmaps - Maps

Let's look at the San Francisco restaurant data

```
u <- "https://raw.githubusercontent.com/pittardsp/info550_spring_2018/master/SUPPORT/SFFoodProgram_Complete_Data.zip"

download.file(u,"SFFoodProgram_Complete_Data.zip")
system.file("unzip SFFoodProgram_Complete_Data.zip")
businesses <- read.csv("businesses_plus.csv",sep=",")
inspections <- read.csv("inspections_plus.csv",sep=",")

# Let's get the top 40 worst businesses in terms of
# inspection scores

top100worst <- inspections %>% arrange((Score)) %>% head(.,100)

# Join this with the businesses table to get addresses

top100worst <- left_join(top100worst,businesses) %>%
  select(name,latitude,longitude,Score)

top100worst <- top100worst[complete.cases(top100worst),]
```

ggmaps - Maps

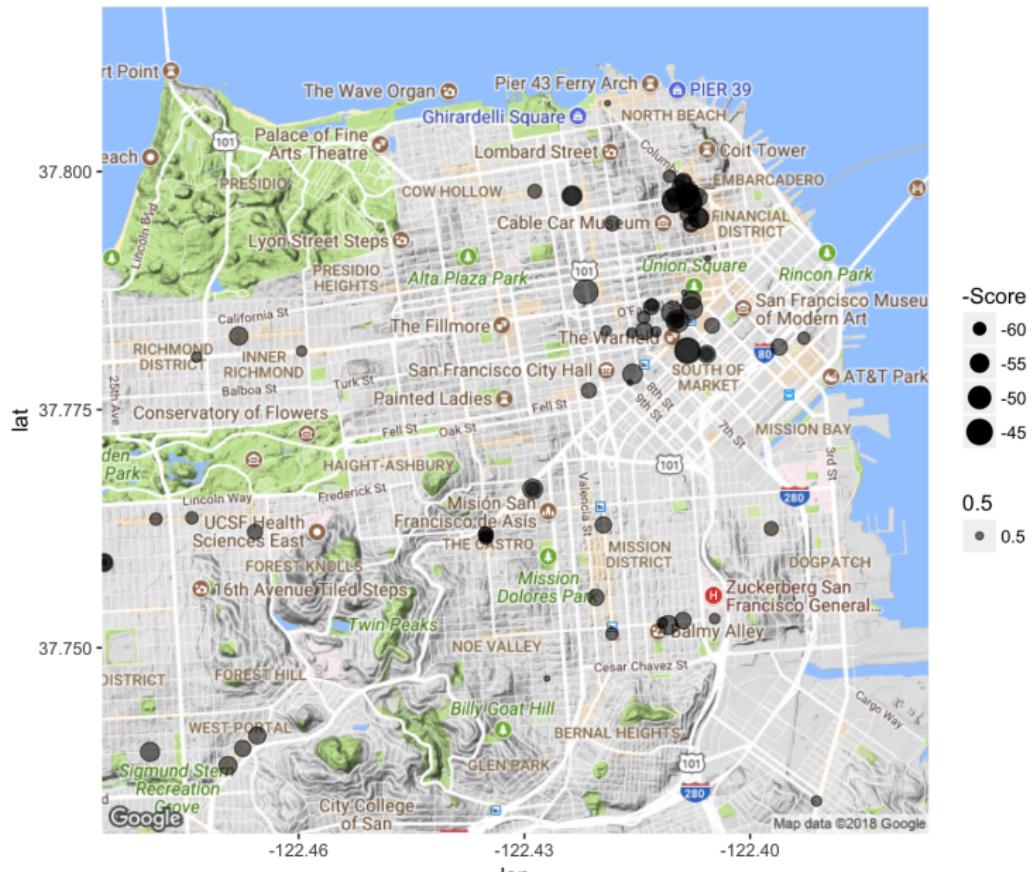
Let's look at the San Francisco restaurant data

```
sf <- c(lon = -122.431297, lat = 37.773972)

# Get map at zoom level 5: map_5
map_5 <- get_map(sf, zoom = 13, scale = 1)

ggmap(map_5) + geom_point(aes(longitude, latitude,
                               alpha=0.5, size=-Score),
                           data = top100worst)
```

ggmap



ggmaps - Maps

Let's look at the San Francisco restaurant data

```
map_5 <- get_map(sf, zoom = 13, scale = 1)

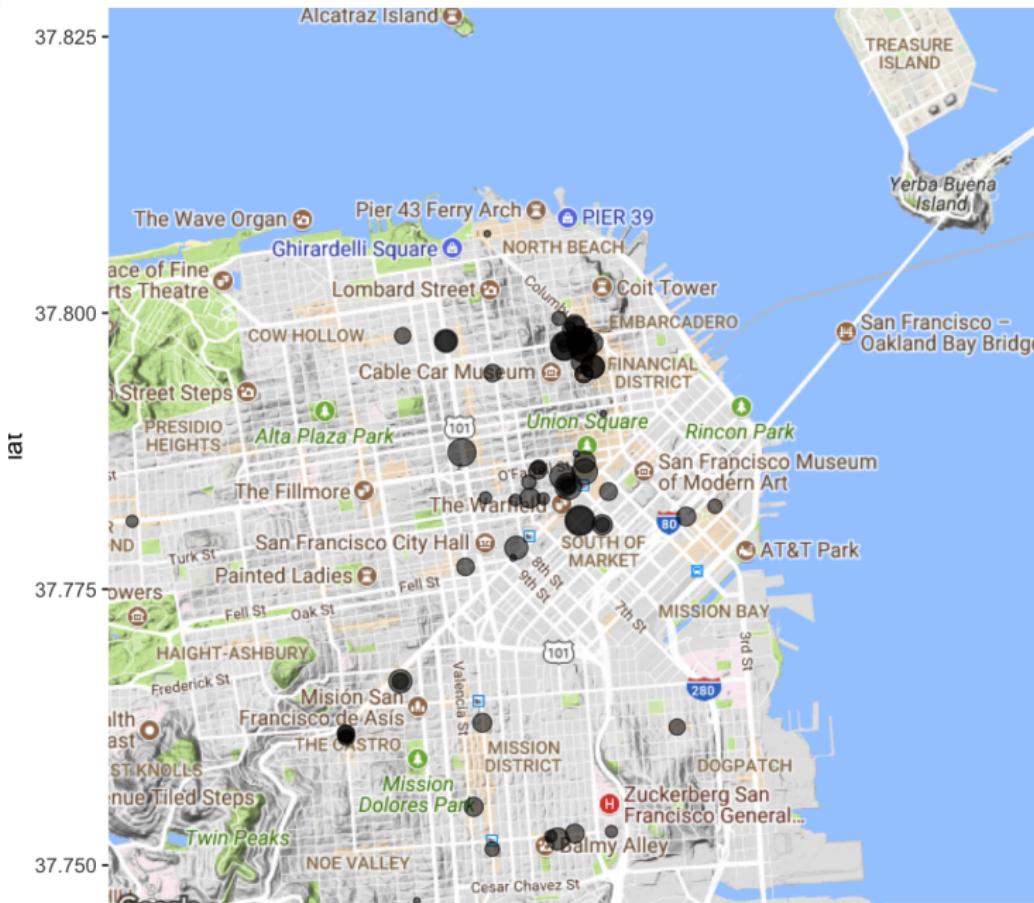
ggmap(map_5) + geom_point(aes(longitude, latitude,
                               alpha=0.5,size=-Score),
                           data = top100worst)
```

```
sf <- c(lon = -122.4075, lat = 37.7880)
```

```
# Get map at zoom level 5: map_5
map_5 <- get_map(sf, zoom = 13, scale = 1)
```

```
ggmap(map_5) + geom_point(aes(longitude, latitude,
                               alpha=0.2,size=-Score),
                           data = top100worst)
```

ggmap



ggmaps - Maps

Let's look at some of the Dialysis Data

```
loc_df <- data.frame(lat=lat,lon=lon,  
                      Mortality_Rate=dia$"Mortality.Rate..Facility.")  
  
map <- get_map(location='united states', zoom=4, maptype = "terrain"  
                 source='google',color='color')  
  
ggmap(map) + geom_point(  
  aes(x=lon, y=lat, color=Mortality_Rate),  
  data=loc_df, alpha=.5, na.rm = T) +  
  scale_color_gradient(low="beige", high="red")
```

ggmap

