Graphics in R

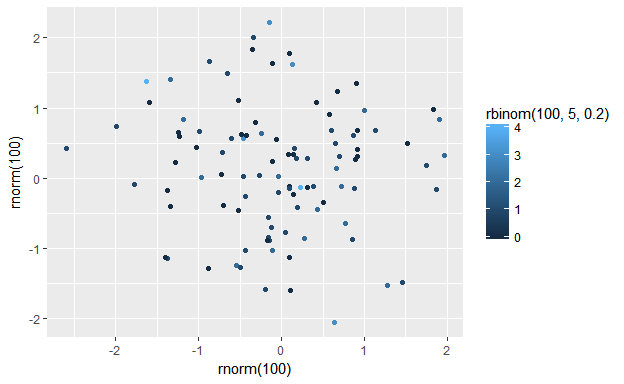
Graphics Objectives:

* Use qplot and ggplot to construct plots in R
* Understand how to create layered graphics
* Apply perceptual principles to create good graphics

# Basic Plots

## Setup

Run the [Setup.R](./code/Setup.R) file.

If everything works correctly, you should see a plot: 

## ggplot2 In a Nutshell

* Package for statistical graphics
* Developed by Hadley Wickham
* Designed to adhere to good graphical practices
* Supports a wide variety plot types
* Constructs plots using the concept of layers
* <http://had.co.nz/ggplot2/> or Hadley's book *ggplot2: Elegant Graphics for Data Analysis}* for reference material

## *qplot*

The *qplot()* function is the basic workhorse of ggplot2

* Produces all plot types available with ggplot2
* Allows for plotting options within the function statement
* Creates an object that can be saved
* Plot layers can be added to modify plot complexity

The *qplot()* function has a basic syntax:

*qplot(variables, plot type, dataset, options)*

* variables: list of variables used for the plot
* plot type: specified with a *geom =* statement
* dataset: specified with a *data =* statement
* options: there are so, so many options!

## Diamonds Data

Objective: Explore the diamonds data set (preloaded along with ggplot2) using qplot for basic plotting.

The data set was scraped from a diamond exchange company data base. It contains the prices and attributes of over 50,000 diamonds.

### Examining the Diamonds Data

What does the data look like?

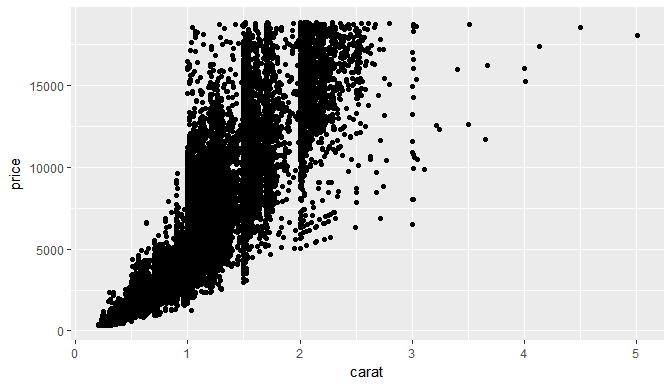
Look at the top few rows of the diamond data frame to find out!

head(diamonds)  
## # A tibble: 6 × 10  
## carat cut color clarity depth table price x y z  
## <dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>  
## 1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43  
## 2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31  
## 3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31  
## 4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63  
## 5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75  
## 6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48

## Basic Scatterplots

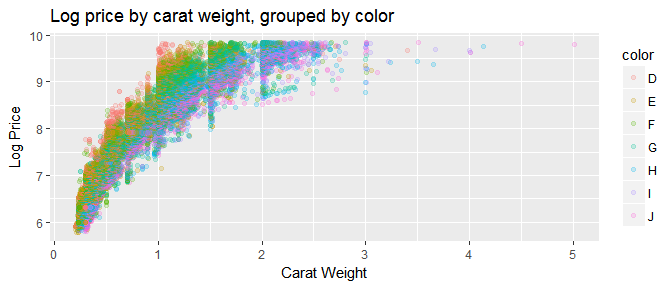
Basic scatter plot of diamond price vs. carat weight

qplot(carat, price, geom = "point", data = diamonds)



Scatter plot of diamond price vs carat weight showing versitility of options in qplot

qplot(carat, log(price), geom = "point", data = diamonds,   
 alpha = I(0.2), color = color,   
 main = "Log price by carat weight, grouped by color") +   
 xlab("Carat Weight") + ylab("Log Price")



##### Your Turn

All of the "Your Turns" for this section will use the tips data set:

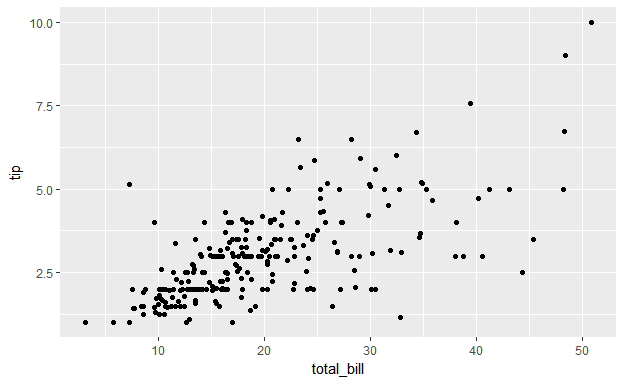
tips <- read.csv("https://bit.ly/2gGoiLR")

1. Use qplot to build a scatterplot of variables tips and total bill
2. Use options within qplot to color points by smokers
3. Clean up axis labels and add main plot title

##### Solutions

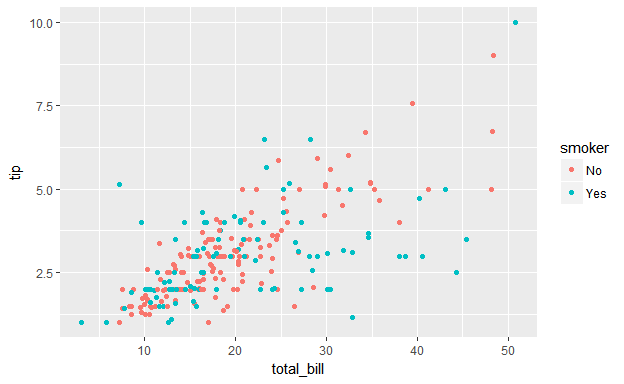
1. Scatterplot of variables tips and total bill

qplot(data = tips, x = total\_bill, y = tip)



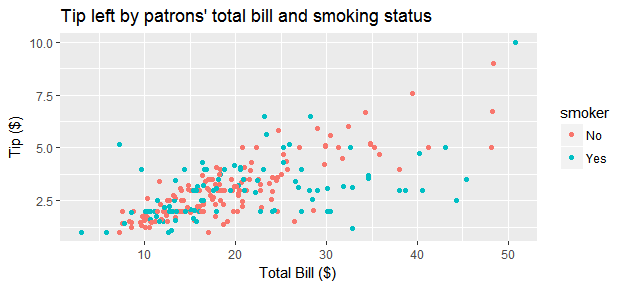
1. Color points by smokers

qplot(data = tips, x = total\_bill, y = tip, color = smoker)



1. Pretty axis lables and title

qplot(data = tips, x = total\_bill, y = tip, color = smoker,  
 xlab = "Total Bill ($)", ylab = "Tip ($)",   
 main = "Tip left by patrons' total bill and smoking status")



# Plotting Map Data

To make a map, load up the states data and take a look:

states <- map\_data("state")  
head(states)  
## long lat group order region subregion  
## 1 -87.46201 30.38968 1 1 alabama <NA>  
## 2 -87.48493 30.37249 1 2 alabama <NA>  
## 3 -87.52503 30.37249 1 3 alabama <NA>  
## 4 -87.53076 30.33239 1 4 alabama <NA>  
## 5 -87.57087 30.32665 1 5 alabama <NA>  
## 6 -87.58806 30.32665 1 6 alabama <NA>

## Basic Map Data

What data is needed in order to plot a basic map?

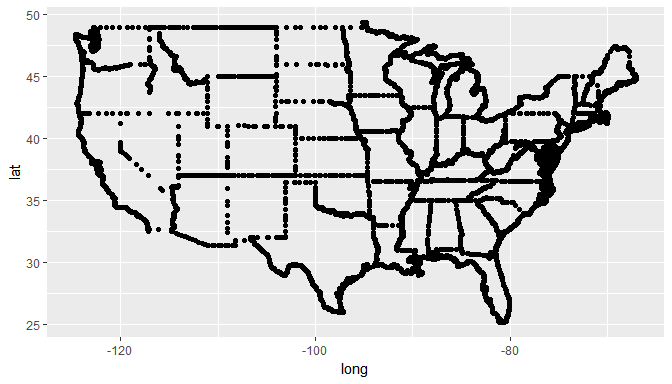
* Latitude/longitude points for all map boundaries
* Which boundary group all lat/long points belong
* The order to connect points within each group

The *states* data has all necessary information

## A Basic Map

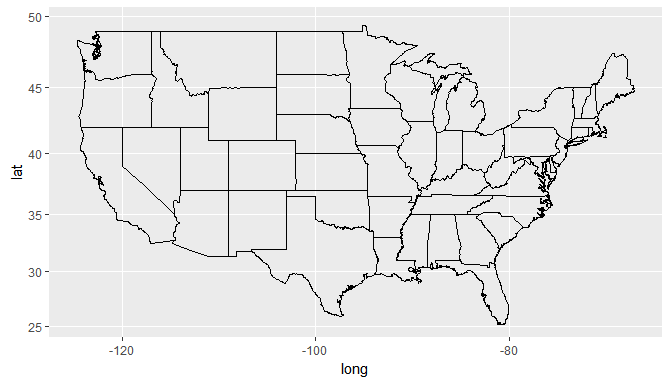
A bunch of latitude longitude points...

qplot(long, lat, geom = "point", data = states)



... that are connected with lines in a very specific order.

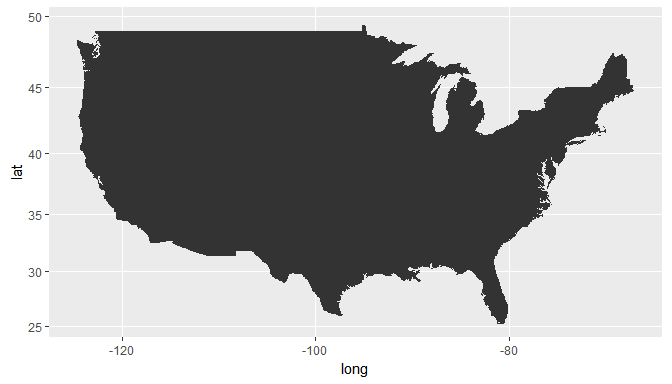
qplot(long, lat, geom = "path", data = states, group = group) +   
 coord\_map()



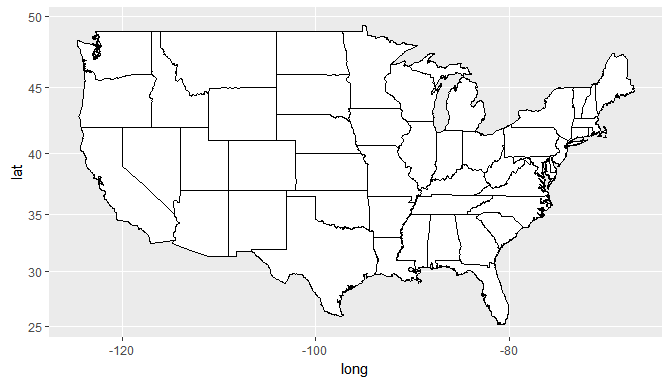
## Polygon vs Path

Polygons are shapes that can be filled. Paths are lines that have no fill color. Polygons are often more appropriate for showing additional information on a map.

qplot(long, lat, geom = "polygon",   
 data = states, group = group) +   
 coord\_map()



# Fancier map  
qplot(long, lat, geom = "polygon",   
 fill = I("white"), color = I("black"),  
 data = states, group = group) +   
 coord\_map()



## Incorporating Information

* Add other geographic information by adding geometric layers to the plot
* Add non-geopgraphic information by altering the fill color for each state
* Use geom = "polygon" to treat states as solid shapes
* Show numeric information with color shade/intensity
* Show categorical information using color hue

### Categorical Data

If a categorical variable is assigned as the fill color then qplot will assign different hues for each category.

Load in a state regions dataset:

statereg <- read.csv("https://bit.ly/2i0AFHK")  
head(statereg)  
## State StateGroups  
## 1 california West  
## 2 nevada West  
## 3 oregon West  
## 4 washington West  
## 5 idaho West  
## 6 montana West

### Joining Data

join or merge the original states data with new info

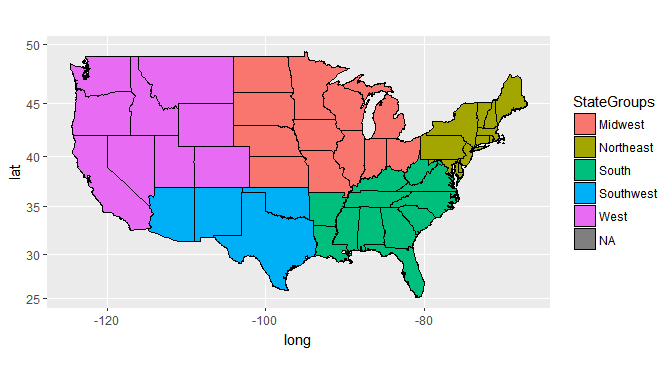
The left\_join function is used for merging\*\*:

library(dplyr)  
states.class.map <- left\_join(states, statereg, by = c("region" = "State"))  
head(states.class.map)  
## long lat group order region subregion StateGroups  
## 1 -87.46201 30.38968 1 1 alabama <NA> South  
## 2 -87.48493 30.37249 1 2 alabama <NA> South  
## 3 -87.52503 30.37249 1 3 alabama <NA> South  
## 4 -87.53076 30.33239 1 4 alabama <NA> South  
## 5 -87.57087 30.32665 1 5 alabama <NA> South  
## 6 -87.58806 30.32665 1 6 alabama <NA> South

\*\* More on this later

### Plotting the Result

qplot(long, lat, geom = "polygon", data = states.class.map,   
 group = group, fill = StateGroups, color = I("black")) +   
 coord\_map()



## Numerical Data & Maps

* Behavioral Risk Factor Surveillance System
* 2008 telephone survey run by the Center for Disease Control (CDC)
* Ask a variety of questions related to health and wellness
* Cleaned data with state aggregated values posted on website

### BRFSS Data Aggregated by State

states.stats <- read.csv("https://bit.ly/2gT95Hc")  
  
## state.name avg.wt avg.qlrest2 avg.ht avg.bmi avg.drnk  
## 1 alabama 180.7247 9.051282 168.0310 29.00222 2.333333  
## 2 alaska 189.2756 8.380952 172.0992 28.90572 2.323529  
## 3 arizona 169.6867 5.770492 168.2616 27.04900 2.406897  
## 4 arkansas 177.3663 8.226619 168.7958 28.02310 2.312500  
## 5 california 170.0464 6.847751 168.1314 27.23330 2.170000  
## 6 colorado 167.1702 8.134715 169.6110 26.16552 1.970501

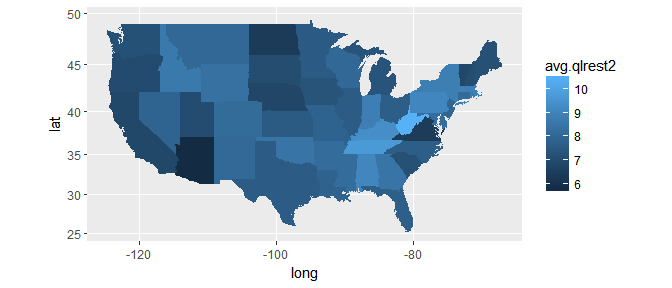
#### Join the data again

states.map <- left\_join(states, states.stats, by = c("region" = "state.name"))  
head(states.map)  
## long lat group order region subregion avg.wt avg.qlrest2  
## 1 -87.46201 30.38968 1 1 alabama <NA> 180.7247 9.051282  
## 2 -87.48493 30.37249 1 2 alabama <NA> 180.7247 9.051282  
## 3 -87.52503 30.37249 1 3 alabama <NA> 180.7247 9.051282  
## 4 -87.53076 30.33239 1 4 alabama <NA> 180.7247 9.051282  
## 5 -87.57087 30.32665 1 5 alabama <NA> 180.7247 9.051282  
## 6 -87.58806 30.32665 1 6 alabama <NA> 180.7247 9.051282  
## avg.ht avg.bmi avg.drnk  
## 1 168.031 29.00222 2.333333  
## 2 168.031 29.00222 2.333333  
## 3 168.031 29.00222 2.333333  
## 4 168.031 29.00222 2.333333  
## 5 168.031 29.00222 2.333333  
## 6 168.031 29.00222 2.333333

#### Shade and Intensity

Average # of days in the last 30 days of insufficient sleep

qplot(long, lat, geom = "polygon", data = states.map,   
 group = group, fill = avg.qlrest2) + coord\_map()



### BRFSS Data by Gender and State

states.sex.stats <- read.csv("https://srvanderplas.github.io/NPPD-Analytics-Workshop/02.Graphics/data/states.sex.stats.csv")  
states.sex.stats <- read.csv("https://bit.ly/2hiKFIb")  
head(states.sex.stats)  
## state.name SEX avg.wt avg.qlrest2 avg.ht avg.bmi avg.drnk sex  
## 1 alabama 1 198.8936 8.648936 177.5729 28.50714 3.033333 Male  
## 2 alabama 2 173.0315 9.224771 163.9956 29.21280 2.041667 Female  
## 3 alaska 1 203.3919 7.236111 178.3896 28.91494 2.487179 Male  
## 4 alaska 2 169.5660 9.907407 163.1296 28.89286 2.103448 Female  
## 5 arizona 1 191.3739 5.163793 177.1724 27.63152 2.814286 Male  
## 6 arizona 2 156.2054 6.142857 162.7043 26.67683 2.026667 Female

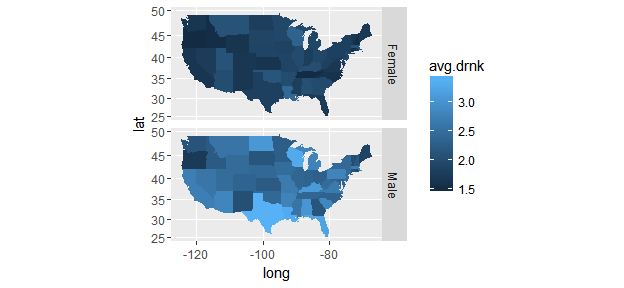
Join the data again

states.sex.map <- left\_join(states, states.sex.stats, by = c("region" = "state.name"))  
head(states.sex.map)  
## long lat group order region subregion SEX avg.wt  
## 1 -87.46201 30.38968 1 1 alabama <NA> 1 198.8936  
## 2 -87.46201 30.38968 1 1 alabama <NA> 2 173.0315  
## 3 -87.48493 30.37249 1 2 alabama <NA> 1 198.8936  
## 4 -87.48493 30.37249 1 2 alabama <NA> 2 173.0315  
## 5 -87.52503 30.37249 1 3 alabama <NA> 1 198.8936  
## 6 -87.52503 30.37249 1 3 alabama <NA> 2 173.0315  
## avg.qlrest2 avg.ht avg.bmi avg.drnk sex  
## 1 8.648936 177.5729 28.50714 3.033333 Male  
## 2 9.224771 163.9956 29.21280 2.041667 Female  
## 3 8.648936 177.5729 28.50714 3.033333 Male  
## 4 9.224771 163.9956 29.21280 2.041667 Female  
## 5 8.648936 177.5729 28.50714 3.033333 Male  
## 6 9.224771 163.9956 29.21280 2.041667 Female

#### Adding Information

Average # of alcoholic drinks per day by state and gender

qplot(long, lat, geom = "polygon", data = states.sex.map,   
 group = group, fill = avg.drnk) + coord\_map() +   
 facet\_grid(sex ~ .)



##### Your Turn

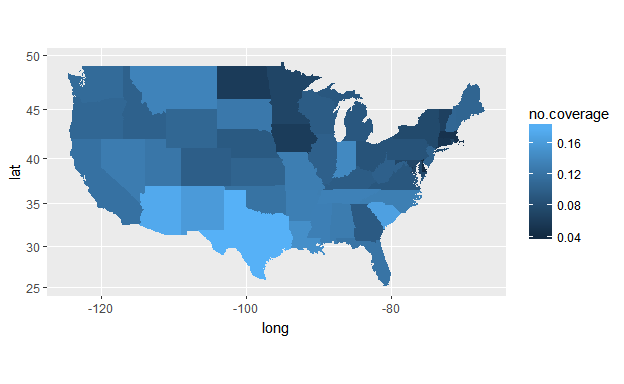
* Use left\_join to combine child healthcare data with maps information.  
  You can load in the child healthcare data with:

states.health.stats <- read.csv("https://bit.ly/2hRBMq0")

* Use qplot to create a map of child healthcare undercoverage rate by state

##### Solutions

library(maps)  
library(dplyr)  
states <- map\_data("state")  
states.health.map <- left\_join(states, states.health.stats,   
 by = c("region" = "state.name"))  
  
# Use qplot to create a map of child healthcare undercoverage rate by state  
  
qplot(data = states.health.map, x = long, y = lat, geom = 'polygon',  
 group = group, fill = no.coverage) + coord\_map()



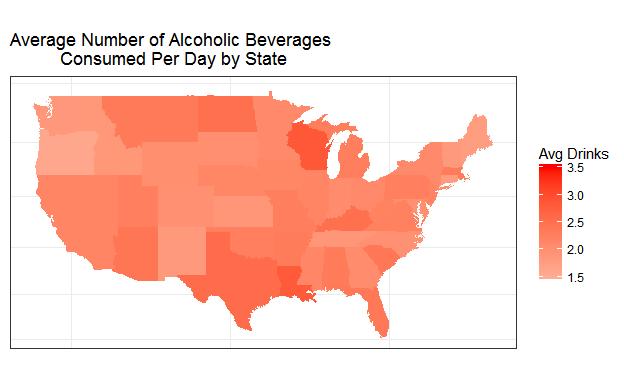
###### End Solutions

## Cleaning Up Maps

Use ggplot2 options to clean up the map!

* Add Titles + ggtitle(...)
* Use a plain white background + theme\_bw()
* Familiar geography may eliminate need for latitude and longitude axes + theme(...)
* Customize color gradient + scale\_fill\_gradient2(...)
* Keep aspect ratios correct + coord\_map()

qplot(long, lat, geom = "polygon", data = states.map,   
 group = group, fill = avg.drnk) +   
 coord\_map() + theme\_bw() +  
 scale\_fill\_gradient2(  
 name = "Avg Drinks",  
 limits = c(1.5, 3.5),   
 low = "lightgray", high = "red") +   
 theme(axis.ticks = element\_blank(),  
 axis.text = element\_blank(),  
 axis.title = element\_blank()) +  
 ggtitle("Average Number of Alcoholic Beverages   
 Consumed Per Day by State")

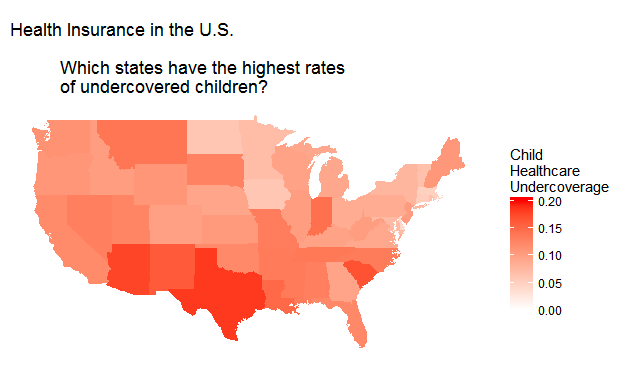


##### Your Turn

Use options to polish the look of the map of child healthcare undercoverage rate by state!

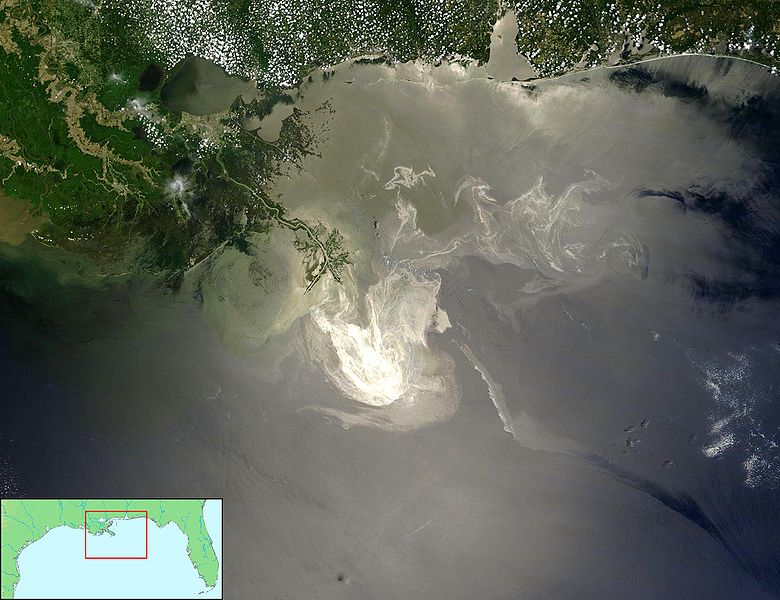
##### Solutions

qplot(data = states.health.map, x = long, y = lat,   
 geom = 'polygon', group = group, fill = no.coverage) +   
 coord\_map() +   
 scale\_fill\_gradient2(  
 name = "Child\nHealthcare\nUndercoverage",  
 limits = c(0, .2),   
 low = 'white', high = 'red') +   
 ggtitle("Health Insurance in the U.S.\n  
 Which states have the highest rates   
 of undercovered children?") +  
 theme\_minimal() +   
 theme(panel.grid = element\_blank(),   
 axis.text = element\_blank(),  
 axis.title = element\_blank())



# Plotting Using Layers

## Deepwater Horizon Oil Spill



### Datasets

NOAA Data: - National Oceanic and Administration - Temperature and Salinity Data in the Gulf of Mexico - Measured using Floats, Gliders and Boats

US Fisheries and Wildlife Data:

* Animal Sightings on the Gulf Coast
* Birds, Turtles and Mammals
* Status: Oil Covered or Not

Both data sets have geographic coordinates for every observation

### Loading NOAA Data

NOAA data is a .rdata file. Read it in:

1. Download the data from <http://heike.github.io/rwrks/02-r-graphics/data/noaa.rdata>
2. Run the getwd() command to find your current working directory
3. Place noaa.rdata in the directory from step 2.
4. Run the command below:

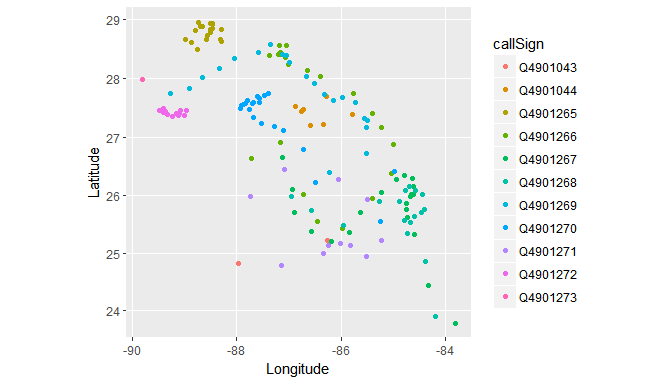
load("noaa.rdata")

### Floats

Take a peek at the top of the floats NOAA data:

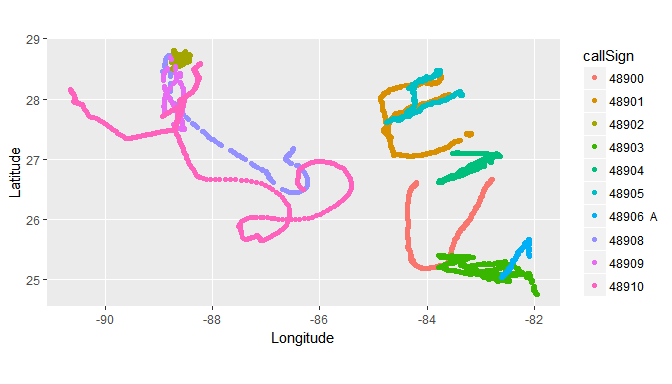
head(floats, n = 2)[,1:5]  
## callSign Date\_Time JulianDay Time\_QC Latitude  
## 1 Q4901043 7/12/2010 2455390 1 24.823  
## 2 Q4901043 7/12/2010 2455390 1 24.823  
head(floats, n = 2)[,6:10]  
## Longitude Position\_QC Depth Depth\_QC Temperature  
## 1 -87.964 1 2 1 29.83  
## 2 -87.964 1 4 1 29.65  
head(floats, n = 2)[,11:14]  
## Temperature\_QC Salinity Salinity\_QC Type  
## 1 1 36.59 1 Float  
## 2 1 36.58 1 Float

qplot(Longitude, Latitude, color = callSign, data = floats) +   
 coord\_map()



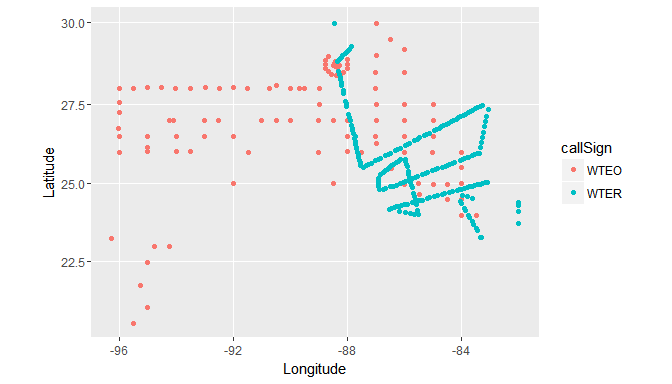
### Gliders

qplot(Longitude, Latitude, color = callSign, data = gliders) +   
 coord\_map()



### Boats

qplot(Longitude, Latitude, color = callSign, data = boats) +   
 coord\_map()



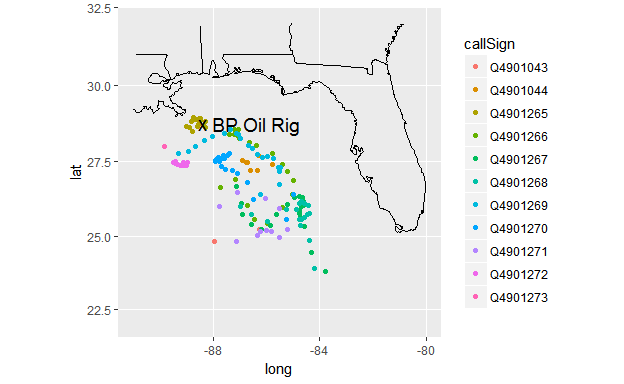
## Layering

The NOAA data has the same context - a common time and common place

* Want to aggregate information from different sources onto a common plot
* Start with a common background the lat/long grid
* Superimpose data onto the grid in layers using ggplot2

#### Preview

ggplot() +  
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = floats, aes(x = Longitude, y = Latitude, color = callSign)) +   
 geom\_point(aes(x, y), shape = "x", size = 5, data = rig) +   
 geom\_text(aes(x, y), label = "BP Oil Rig",   
 size = 5, data = rig, hjust = -0.1) +   
 xlim(c(-91, -80)) + ylim(c(22,32)) + coord\_map()



## What is a Plot?

* Most maps (and many plots) have multiple layers of data.
* The layers may be from the same or different datasets.
* ggplot2 makes it easy to add layers to a plot.

Plots have: - A default dataset - A coordinate system - layers of geometric objects (geoms) - A set of aesthetic mappings (taking information from the data and converting into an attribute of the plot) - A scale for each aesthetic - A facetting specification (multiple plots based on subsetting the data)

### Example: Floats Decomposed

**Data**: floats, states

**Mappings**:

|  |  |
| --- | --- |
| aesthetic | mapping |
| x | Longitude |
| y | Latitude |
| color | CallSign |

**Scales**:

|  |  |
| --- | --- |
| aesthetic | scale |
| x | continuous |
| y | continuous |
| color | discrete |

**Geoms**: Points (floats), lines (states)

**Facetting**: None

## qplot vs ggplot

qplot() stands for "quickplot":

* Automatically chooses default settings to make life easier
* Less control over plot construction

ggplot() stands for "grammar of graphics plot"

* Contructs the plot using components listed in previous slides

There are two ways to construct the same plot for float locations:

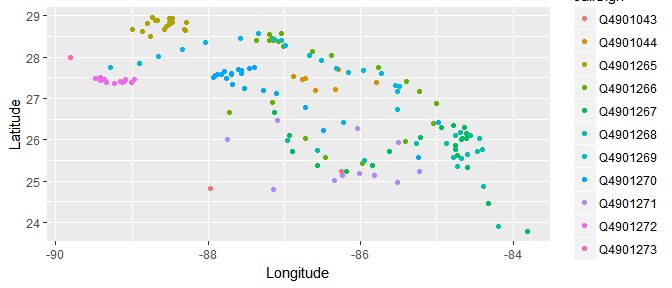
qplot(Longitude, Latitude, color = callSign, data = floats)

Or:

ggplot(data = floats,   
 aes(x = Longitude, y = Latitude, color = callSign)) +  
 geom\_point() +   
 scale\_x\_continuous() +   
 scale\_y\_continuous() +   
 scale\_color\_discrete()

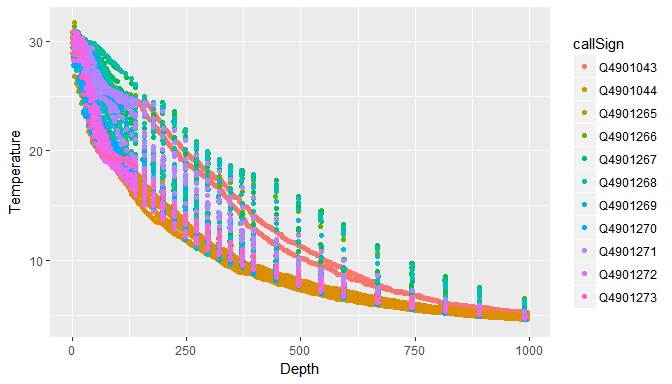
It is not necessary to specify everything when using ggplot. The function will automatically pick default scales if none are specified.

ggplot(data = floats,   
 aes(x = Longitude, y = Latitude, color = callSign)) +  
 geom\_point()



##### Your Turn

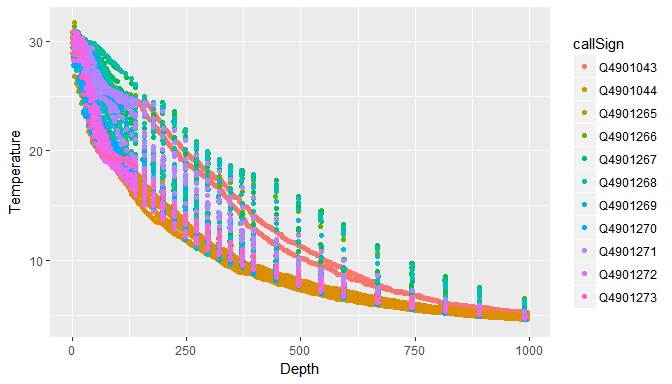
Find the ggplot() statement that creates this plot:



Hint: look at the Floats data for variable ideas

##### Solutions

ggplot(aes(x = Depth, y = Temperature, color = callSign),   
 data = floats) +   
 geom\_point()



###### End Solutions

## What is a Layer?

A layer added ggplot() can be a geom...

* The type of geometric object
* The statistic mapped to that object
* The data set from which to obtain the statistic

... or a position adjustment to the scales

* Changing the axes scale
* Changing the color gradient

### Examples

|  |  |  |
| --- | --- | --- |
| Plot | Geom | Stat |
| Scatterplot | point | identity |
| Histogram | bar | bin count |
| Smoother | line + ribbon | smoother function |
| Binned Scatterplot | rectange + color | 2d bin count |

More geoms described at <http://docs.ggplot2.org/current/>

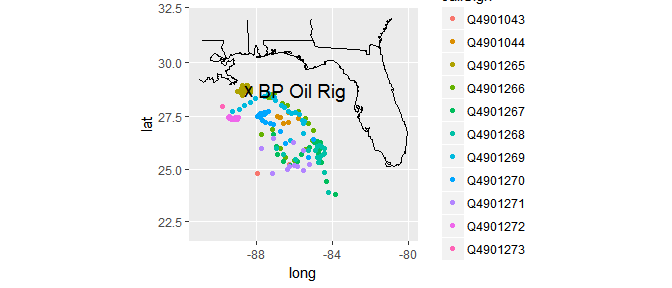
## Piecing Things Together

Build a map using NOAA data

* Coordinate system (mapping Long-Lat to X-Y)
* Add layer of state outlines
* Add layer of points for float locations
* Add layers for Oil Rig marker and label
* Adjust the range of x and y scales

#### The Result

ggplot() +  
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = floats, aes(x = Longitude, y = Latitude, color = callSign)) +   
 geom\_point(aes(x, y), shape = "x", size = 5, data = rig) +   
 geom\_text(aes(x, y), label = "BP Oil Rig", size = 5, data = rig, hjust = -0.1) +   
 xlim(c(-91, -80)) +   
 ylim(c(22, 32)) + coord\_map()



##### Your Turn

animal <- read.csv("https://bit.ly/2hNlTUl")

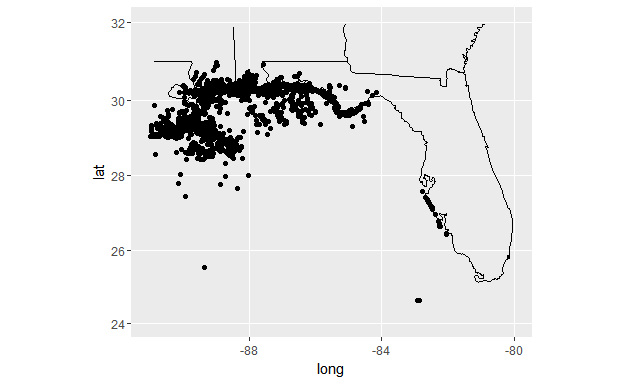
1. Read in the animal.csv data:  
   (Data of animal sightings around the Deepwater Site)
2. Plot the location of animal sightings on a map of the region
3. On this plot, try to color points by class of animal and/or status of animal
4. **Advanced**: Is there a way to indicate time?

library(lubridate)  
animal$month <- month(as.Date(animal$Date\_))

##### Solutions

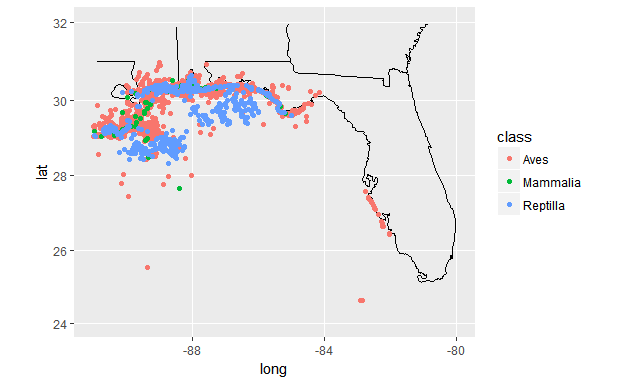
1. Plot the location of animal sightings on a map

ggplot() +   
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = animal, aes(x = Longitude, y = Latitude)) +   
 xlim(c(-91, -80)) + ylim(c(24,32)) + coord\_map()

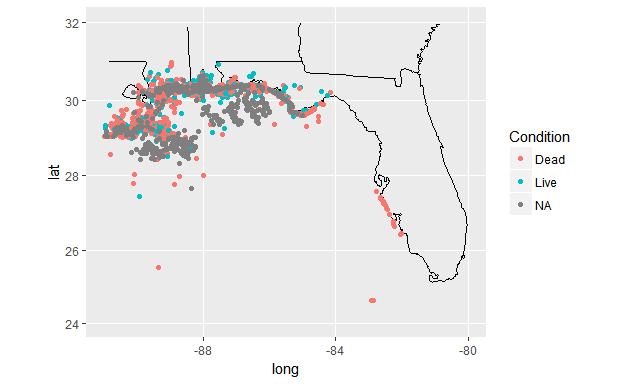


1. On this plot, try to color points by class of animal and/or status of animal

ggplot() +   
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = animal, aes(x = Longitude, y = Latitude,   
 color = class)) +   
 xlim(c(-91, -80)) + ylim(c(24,32)) + coord\_map()

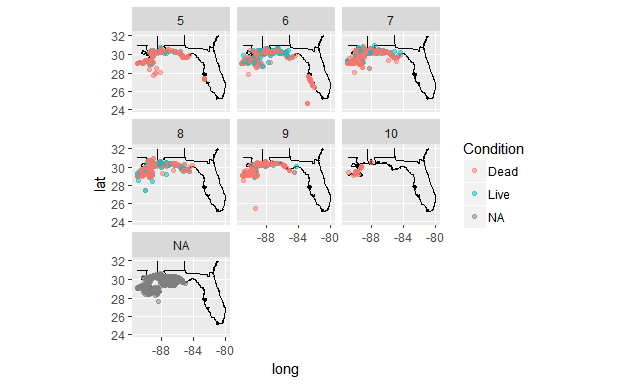


ggplot() +   
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = animal, aes(x = Longitude, y = Latitude,   
 color = Condition)) +   
 xlim(c(-91, -80)) + ylim(c(24,32)) + coord\_map()



1. **Advanced**: Is there a way to indicate time?

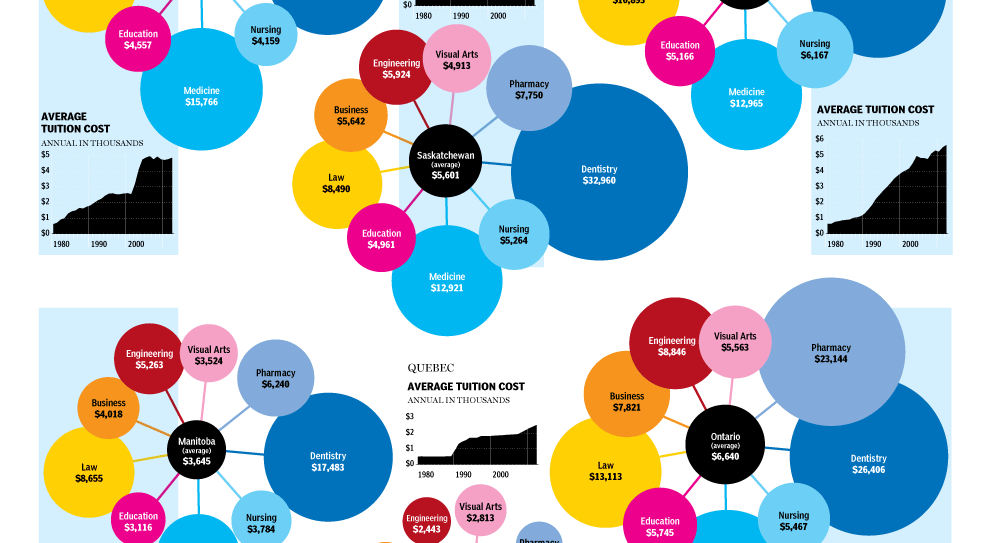
ggplot() +   
 geom\_path(data = states, aes(x = long, y = lat, group = group)) +   
 geom\_point(data = animal, aes(x = Longitude, y = Latitude,   
 color = Condition), alpha = .5) +  
 xlim(c(-91, -80)) + ylim(c(24,32)) +  
 facet\_wrap(~month) + coord\_map()



# Perception

The motivation for this section is to establish why some plots are easier to read than others.

The plot below shows a portion of a figure which was supposed to demonstrate how much it cost to get various degrees in different provinces of Canada.



It is difficult to compare the different provinces on average or across programs, and it is extremely difficult to draw sweeping conclusions about the expense of obtaining an education. The plot is confusing and not particularly useful.

## Good Graphics

Graphics consist of:

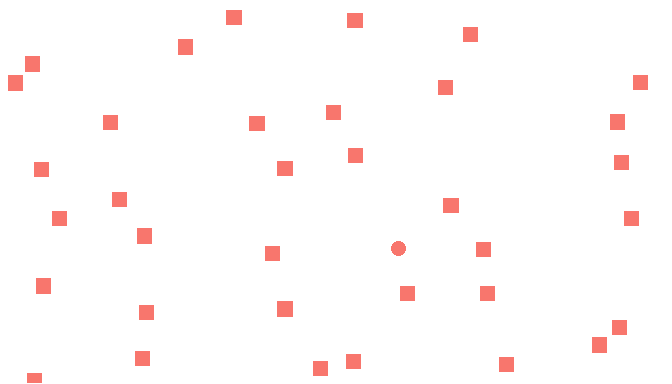
* Structure (boxplot, scatterplot, etc.)
* Aesthetics: features such as
  + color
  + shape
  + size  
    that map other characteristics to structural features

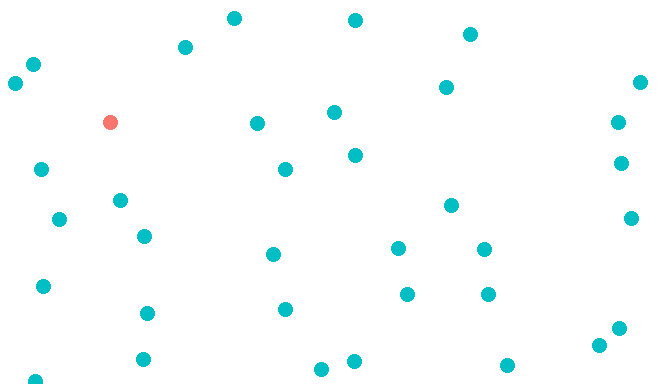
Both the structure and aesthetics should help viewers interpret the information.

In order to understand how to create good graphics, it is necessary to briefly discuss the human perceptual system.

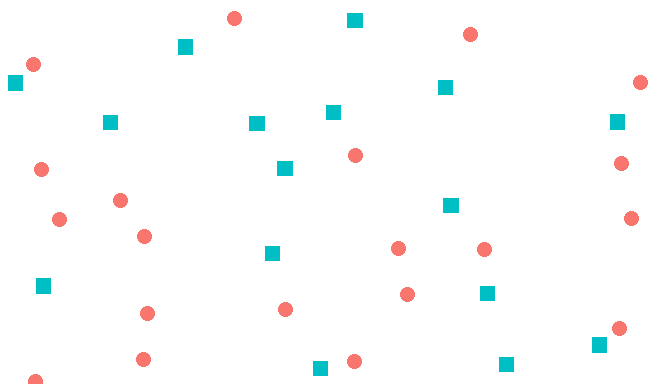
### Pre-Attentive Features

* Things that "jump out" in less than 250 ms
* Color, form, movement, spatial localization

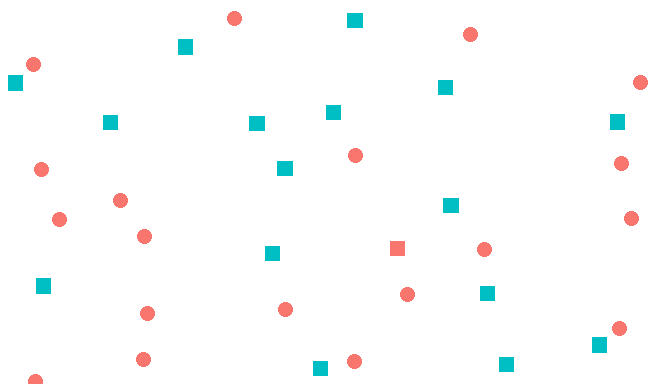




Color is stronger (faster) than shape.



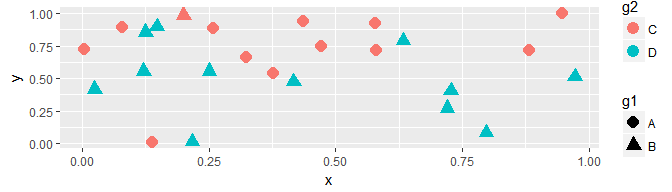
Combinations of pre-attentive features are usually not pre-attentive due to **interference**



##### Your Turn

Find ways to improve the following graphic:

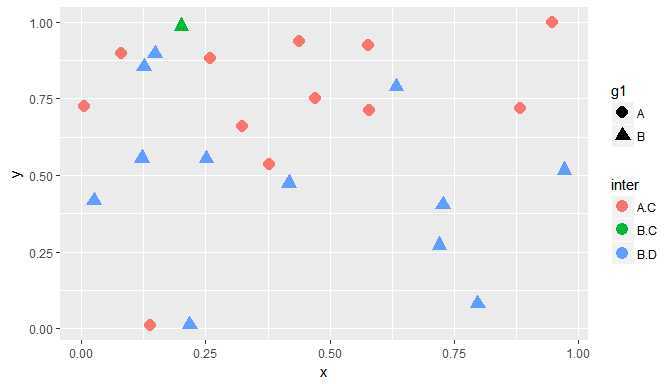
frame <- read.csv("https://bit.ly/2i3Q4Gf")  
qplot(x, y, data = frame, shape = g1, colour = g2, size = I(4))



* Make sure the "oddball" stands out while keeping the information on the groups
* Hint: interaction combines factor variables

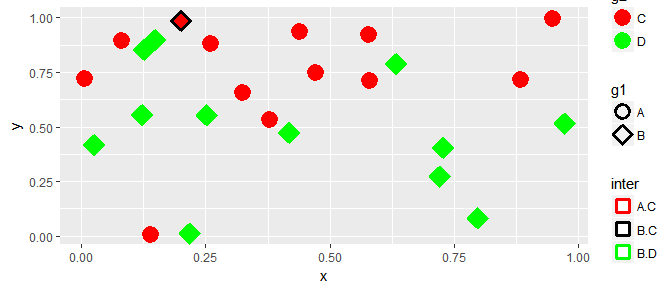
##### Solutions

# Make sure the "oddball" stands out while keeping the   
# information on the groups  
frame$inter <- interaction(frame$g1, frame$g2)  
ggplot(frame, aes(x, y)) +   
 geom\_point(aes(shape = g1, color = inter), size = I(4))



Another (more elegant, but more complicated) solution:

# Make sure the "oddball" stands out while keeping the   
# information on the groups  
frame$inter <- interaction(frame$g1, frame$g2)  
ggplot(frame, aes(x, y)) +   
 geom\_point(aes(shape = g1, fill = g2, color = inter), size = I(4), stroke = I(2)) +   
 scale\_shape\_manual(values = c(21,23)) +   
 scale\_fill\_manual(values = c("red", "green")) +   
 scale\_colour\_manual(values = c("red", "black", "green")) +   
 guides(fill = guide\_legend(override.aes = list(color = c("red", "green"))),  
 colour = guide\_legend(override.aes = list(fill = "white", shape = 22)))



###### End Solutions

## Accuracy of Perception

Human visual perception is very accurate for some qualities, but accuracy is not 100%. The following hierarchy lists judgment types from most accurate to least accurate.

1. Position (common scale)  
   e.g. bar chart, scatter plot, line graph
2. Position (non-aligned scale)  
   e.g. stacked bar chart
3. Length, Direction, Angle, Slope
4. Area
5. Volume, Density, Curvature
6. Shading, Color Saturation, Color Hue

### Example

Using the previous list, which is a more accurate way to display the same data:

1. A pie chart
2. A bar chart

Answer: A bar chart is more accurate. It displays numerical information on a common, aligned scale, where a pie chart requires the reader to compare numerical information based on area or angle size.

Complicated plots may have more than one or two important variables. When showing multivariate information, it is generally a good idea to prioritize the most important variables, placing them such that the reader can easily compare the values from those variables.

If you have observations of height and weight over time, and height is more important than weight, how would you construct your plot?

|  |  |
| --- | --- |
| aesthetic | variable |
| x |  |
| y |  |
| color |  |

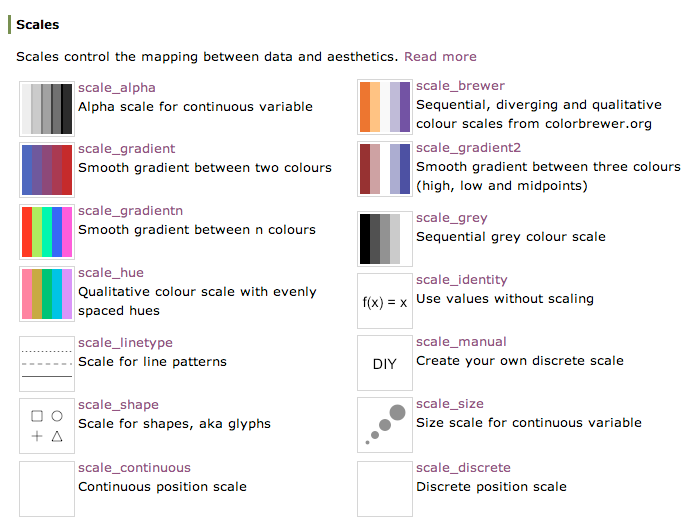
Answer:

|  |  |
| --- | --- |
| aesthetic | variable |
| x | time |
| y | height |
| color | weight |

Since height is more important to show, it should be displayed using a position scale. Weight is less important, so it should be shown using a color (or shape) scale.

### Aesthetics in ggplot2

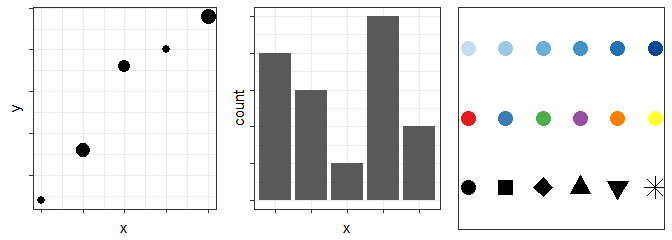
Main parameters: alpha, shape, color, size



### Ordering Variables

Some aesthetics are clearly ordered: size, position, and area all have natural orderings. Other aesthetics do not have natural orderings and should not be used when trying to visually illustrate magnitude.

* Position:  
  higher is larger (y), items to the right are larger (x)
* Size, Area  
  bigger size or more area indicates a larger value
* Color: not always ordered.  
  More contrast = larger, but hue does not indicate order in any natural way.  
  Example: orange is not intuitively greater than green. Light orange, however, is intuitively "less" than dark orange.
* Shape: Unordered  
  A square is not obviously larger/bigger than a circle of the same area.

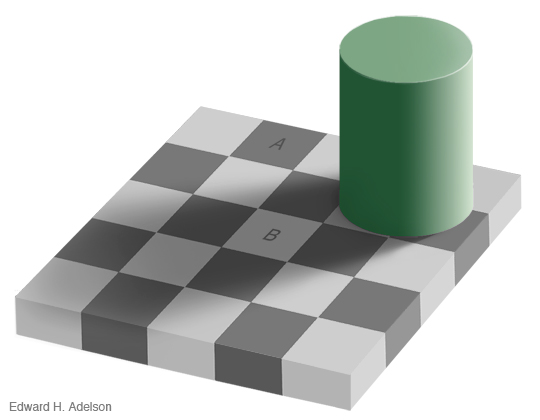


## Color

* Hue: shade of color (red, orange, yellow...)
* Intensity: amount of color
* Both color and hue are pre-attentive.  
  Bigger contrast corresponds to faster detection.

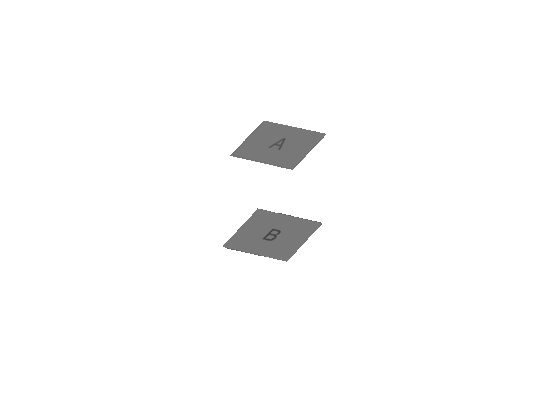
### Perception of Color

Color is context-sensitive:



A and B are the same intensity and hue, but appear to be different.

When the context is removed, it is clear that A and B are the same color.



The human perceptual system has been optimized for perceiving natural scenes. When producing "unnatural" graphics showing data, it is sometimes necessary to make design accommodations so that the graphic can be easily perceived and understood.

The following guidelines may be useful when selecting color schemes.

### Gradients

#### Qualitative schemes

When using a qualitative color scheme for a categorical variable, no more than 7 colors should be used. This is because working memory holds "7 +/- 2" values (between 5 and 9 bits of information). Using more than 7 colors requires the reader to hold more than 7 pieces of information in working memory at a time, leading to headaches and confusion. In addition, it can be hard to distinguish colors when many similar colors are present in a scale.



#### Quantitative schemes

Gradients can be used to represent quantitative information. When using numerical variables (continuous or discrete), best practice is to use a color gradient with only one hue for positive values. These scales move from light to dark with only one color.

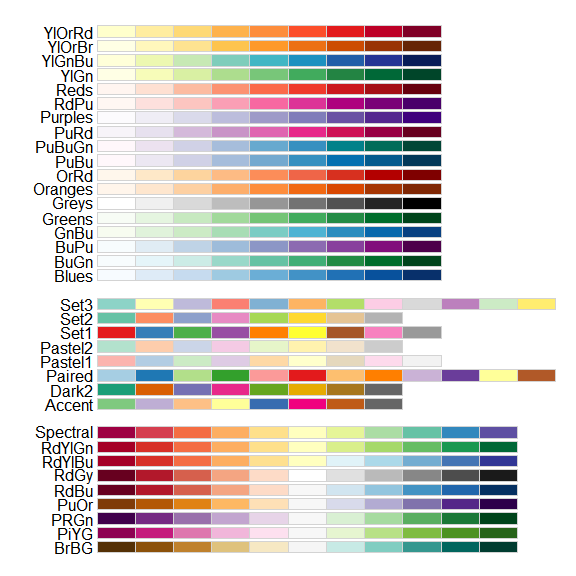


Sometimes, it is useful to emphasize differences in quantitative variables. For instance, it is common on maps to show values which are above and below "average". In these situations, it is permissible to use two colors - one to represent positive values, and one to represent negative values. The gradient should move through a light, neutral color (white, or in some cases light yellow), corresponding to 0 or the average value. 

When working with small objects or thin lines, scales need more contrast than when working with larger areas.

#### RColorBrewer

R package based on Cynthia Brewer's color schemes (<http://www.colorbrewer2.org>). The package contains a number of gradient-style color schemes that integrate nicely with ggplot2.



These schemes are designed for use in maps, but can often be used as-is (or slightly modified) for other types of data displays. <http://www.colorbrewer2.org> provides information about which schemes are colorblind friendly, can be easily photocopied, etc.

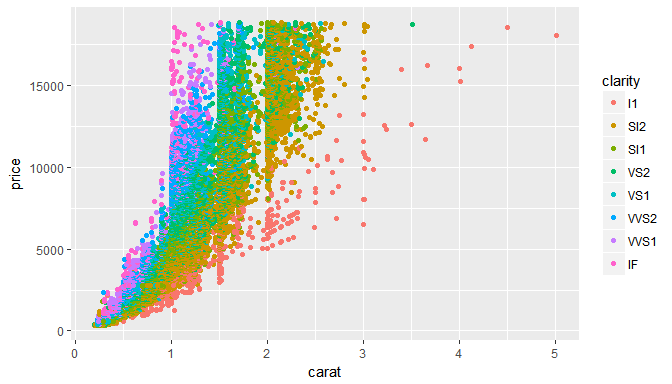
### Color in ggplot2

* Factor variable:
  + scale\_colour\_discrete
  + scale\_colour\_brewer(palette = ...)
* Continuous variable:
  + scale\_colour\_gradient (define low, high values)
  + scale\_colour\_gradient2 (define low, mid, and high values)
  + scale\_colour\_distiller (interpolates a discrete palette for use with continuous data)  
    This allows use of RColorBrewer schemes with continuous variables.
  + Equivalents for fill: scale\_fill\_...

##### Your Turn

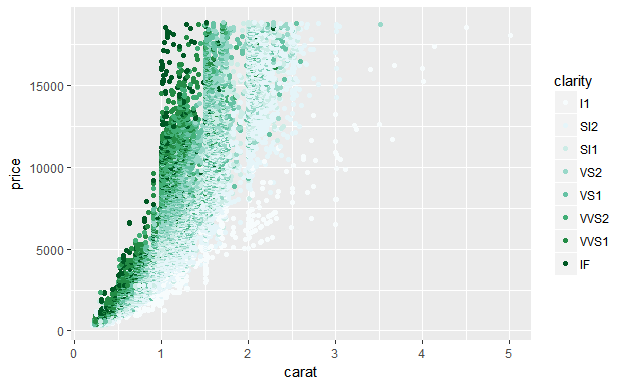
* In the diamonds data, cut is ordinal, while price and carat are continuous
* Find a graphic that gives an overview of these three variables while respecting their types
* Hint: Start with the following code

qplot(carat, price, colour = clarity, data = diamonds)



##### Solutions

qplot(carat, price, colour = clarity, data = diamonds) +   
 scale\_colour\_brewer(palette = "BuGn")



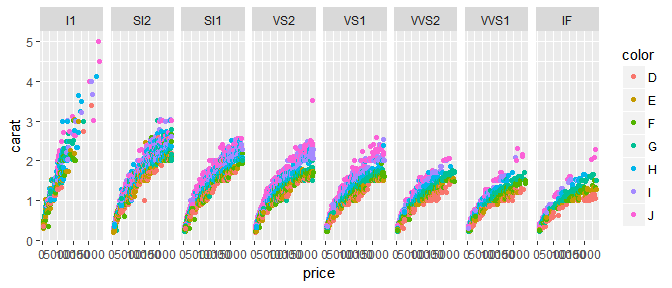
###### End Solutions

## Facetting

Facets are a way to extract subsets of data and place them side-by-side in graphics

* qplot Syntax: facets = row ~ col Use . if there is no variable for either row or column (i.e. facets = . ~ col)
* ggplot Syntax: + facet\_wrap(~ variable) or + facet\_grid(row ~ col)

qplot(price, carat, data = diamonds, color = color,   
 facets = . ~ clarity)



##### Your Turn

The movies dataset contains information from IMDB.com including ratings, genre, length in minutes, and year of release.

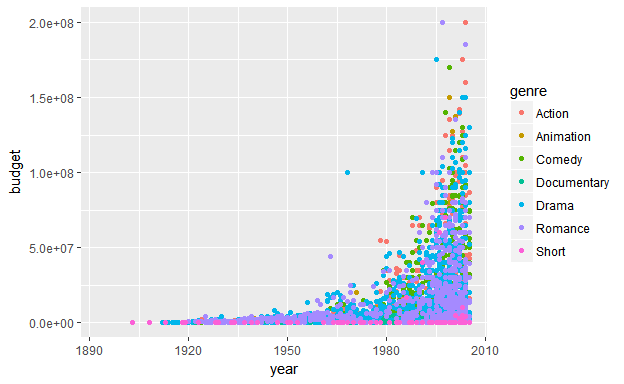
movies <- read.csv("https://bit.ly/2hqhCoM")

* Explore the differences in length, rating, etc. in movie genres over time
* Hint: use facetting!

##### Solutions

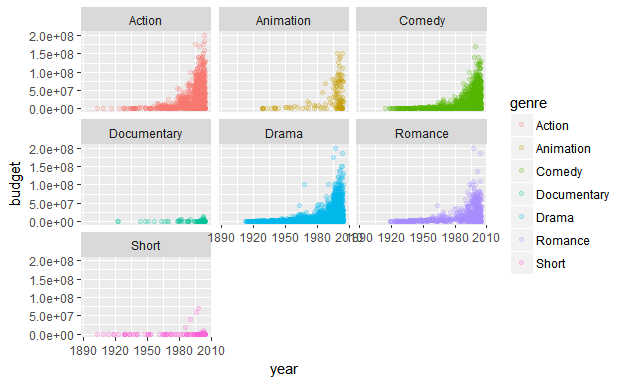
Start by exploring year, budget, and genre.

ggplot(movies, aes(x = year, y = budget,   
 group = genre, color = genre)) +   
 geom\_point()



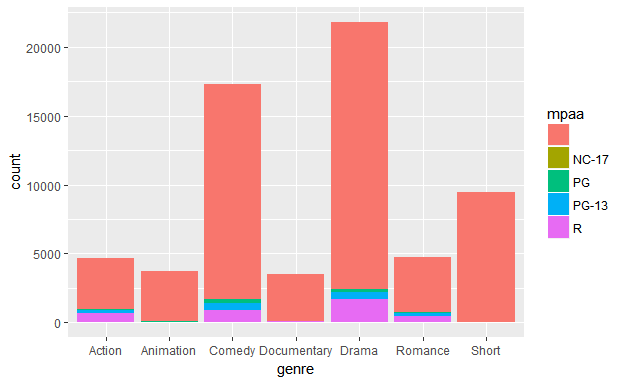
There's a lot of "mush" - it's hard to see individual points.

ggplot(movies, aes(x = year, y = budget,   
 group = genre, color = genre)) +   
 geom\_point(alpha = I(.2)) +   
 facet\_wrap(~genre)



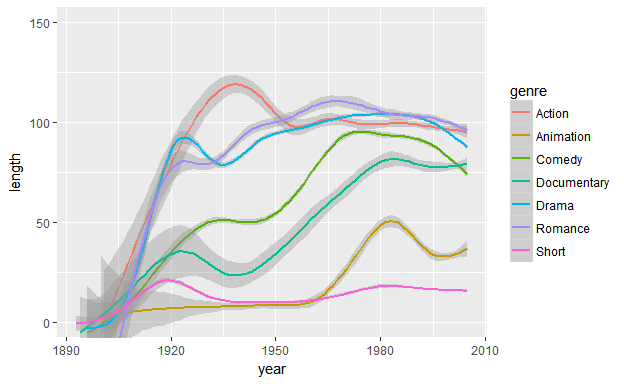
How many observations from each genre, by rating?

ggplot(movies, aes(x = genre, fill = mpaa)) + geom\_bar()



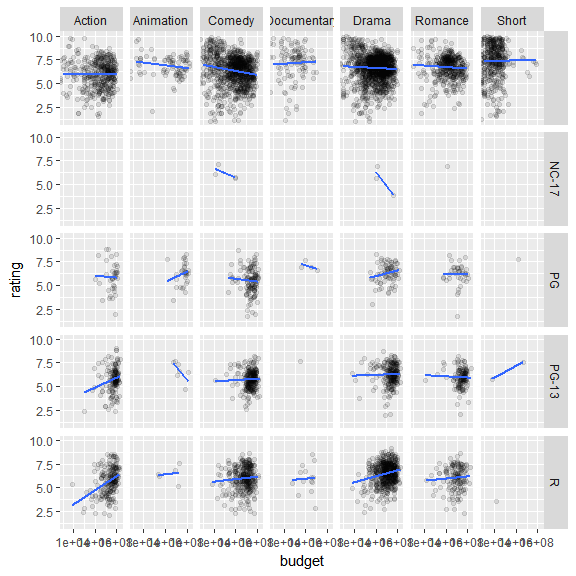
How does movie length change (on average) over time, by genre?

ggplot(movies, aes(x = year, y = length,   
 group = genre, color = genre)) +  
 geom\_smooth(fullrange = F) +   
 coord\_cartesian(ylim = c(0, 150))  
## `geom\_smooth()` using method = 'gam'



What is the relationship between budget and rating, by genre and MPAA classification?

ggplot(movies, aes(x = budget, y = rating, group = genre)) +   
 geom\_point(alpha = .1) +  
 facet\_grid(mpaa ~ genre) +   
 geom\_smooth(method = "lm", se = F) +   
 scale\_x\_log10()



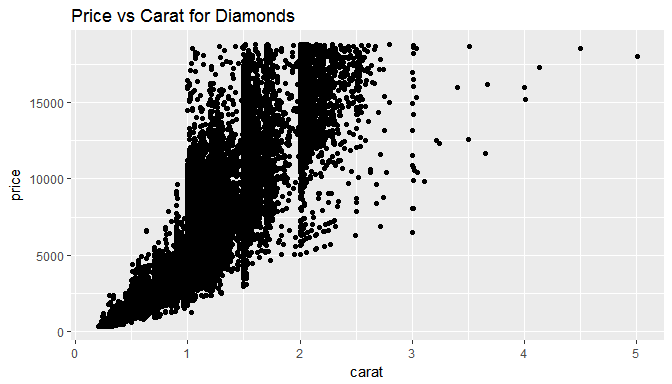
# Polishing Plots

This section focuses on the details of plots - background colors, appearance, fonts, etc.

ggplot allows you to change these details to create highly customized plots.

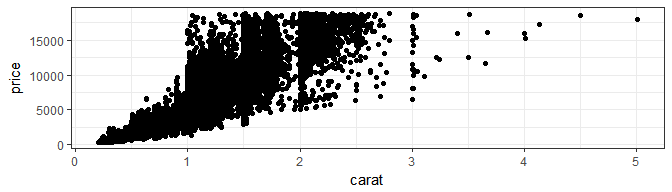
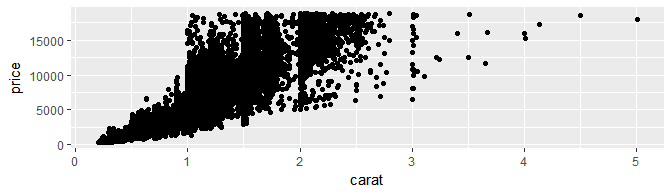
## Plot Title

qplot(carat, price, data = diamonds) +  
 ggtitle("Price vs Carat for Diamonds")



## Built-In Themes

qplot(carat, price, data = diamonds)  
qplot(carat, price, data = diamonds) + theme\_bw()



theme\_set specifies a default theme for all plots:

theme\_set(theme\_bw())

It is also possible to view the options for each theme:

theme\_bw()  
## List of 57  
## $ line :List of 6  
## ..$ colour : chr "black"  
## ..$ size : num 0.5  
## ..$ linetype : num 1  
## ..$ lineend : chr "butt"  
## ..$ arrow : logi FALSE  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_line" "element"  
## $ rect :List of 5  
## ..$ fill : chr "white"  
## ..$ colour : chr "black"  
## ..$ size : num 0.5  
## ..$ linetype : num 1  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ text :List of 11  
## ..$ family : chr ""  
## ..$ face : chr "plain"  
## ..$ colour : chr "black"  
## ..$ size : num 11  
## ..$ hjust : num 0.5  
## ..$ vjust : num 0.5  
## ..$ angle : num 0  
## ..$ lineheight : num 0.9  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : logi FALSE  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.title.x :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 1  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 5.5 0 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.title.x.top :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 0  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 5.5 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.title.y :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 1  
## ..$ angle : num 90  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 5.5 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.title.y.right :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 0  
## ..$ angle : num -90  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 0 5.5  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : chr "grey30"  
## ..$ size :Class 'rel' num 0.8  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.x :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 1  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 2.2 0 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.x.top :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : num 0  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 2.2 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.y :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : num 1  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 2.2 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.y.right :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : num 0  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 0 2.2  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.ticks :List of 6  
## ..$ colour : chr "grey20"  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ lineend : NULL  
## ..$ arrow : logi FALSE  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_line" "element"  
## $ axis.ticks.length :Class 'unit' atomic [1:1] 2.75  
## .. ..- attr(\*, "valid.unit")= int 8  
## .. ..- attr(\*, "unit")= chr "pt"  
## $ axis.line : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ axis.line.x : NULL  
## $ axis.line.y : NULL  
## $ legend.background :List of 5  
## ..$ fill : NULL  
## ..$ colour : logi NA  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ legend.margin :Classes 'margin', 'unit' atomic [1:4] 0.2 0.2 0.2 0.2  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## $ legend.spacing :Class 'unit' atomic [1:1] 0.4  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## $ legend.spacing.x : NULL  
## $ legend.spacing.y : NULL  
## $ legend.key :List of 5  
## ..$ fill : chr "white"  
## ..$ colour : logi NA  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ legend.key.size :Class 'unit' atomic [1:1] 1.2  
## .. ..- attr(\*, "valid.unit")= int 3  
## .. ..- attr(\*, "unit")= chr "lines"  
## $ legend.key.height : NULL  
## $ legend.key.width : NULL  
## $ legend.text :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size :Class 'rel' num 0.8  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ legend.text.align : NULL  
## $ legend.title :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : num 0  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ legend.title.align : NULL  
## $ legend.position : chr "right"  
## $ legend.direction : NULL  
## $ legend.justification : chr "center"  
## $ legend.box : NULL  
## $ legend.box.margin :Classes 'margin', 'unit' atomic [1:4] 0 0 0 0  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## $ legend.box.background: list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ legend.box.spacing :Class 'unit' atomic [1:1] 0.4  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## $ panel.background :List of 5  
## ..$ fill : chr "white"  
## ..$ colour : logi NA  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ panel.border :List of 5  
## ..$ fill : logi NA  
## ..$ colour : chr "grey20"  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ panel.spacing :Class 'unit' atomic [1:1] 5.5  
## .. ..- attr(\*, "valid.unit")= int 8  
## .. ..- attr(\*, "unit")= chr "pt"  
## $ panel.spacing.x : NULL  
## $ panel.spacing.y : NULL  
## $ panel.grid.major :List of 6  
## ..$ colour : chr "grey92"  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ lineend : NULL  
## ..$ arrow : logi FALSE  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_line" "element"  
## $ panel.grid.minor :List of 6  
## ..$ colour : chr "grey92"  
## ..$ size : num 0.25  
## ..$ linetype : NULL  
## ..$ lineend : NULL  
## ..$ arrow : logi FALSE  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_line" "element"  
## $ panel.ontop : logi FALSE  
## $ plot.background :List of 5  
## ..$ fill : NULL  
## ..$ colour : chr "white"  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ plot.title :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size :Class 'rel' num 1.2  
## ..$ hjust : num 0  
## ..$ vjust : num 1  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 6.6 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ plot.subtitle :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size :Class 'rel' num 0.9  
## ..$ hjust : num 0  
## ..$ vjust : num 1  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 0 4.95 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ plot.caption :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size :Class 'rel' num 0.9  
## ..$ hjust : num 1  
## ..$ vjust : num 1  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 4.95 0 0 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ plot.margin :Classes 'margin', 'unit' atomic [1:4] 5.5 5.5 5.5 5.5  
## .. ..- attr(\*, "valid.unit")= int 8  
## .. ..- attr(\*, "unit")= chr "pt"  
## $ strip.background :List of 5  
## ..$ fill : chr "grey85"  
## ..$ colour : chr "grey20"  
## ..$ size : NULL  
## ..$ linetype : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_rect" "element"  
## $ strip.placement : chr "inside"  
## $ strip.text :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : chr "grey10"  
## ..$ size :Class 'rel' num 0.8  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ strip.text.x :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 5.5 0 5.5 0  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ strip.text.y :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : num -90  
## ..$ lineheight : NULL  
## ..$ margin :Classes 'margin', 'unit' atomic [1:4] 0 5.5 0 5.5  
## .. .. ..- attr(\*, "valid.unit")= int 8  
## .. .. ..- attr(\*, "unit")= chr "pt"  
## ..$ debug : NULL  
## ..$ inherit.blank: logi TRUE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ strip.switch.pad.grid:Class 'unit' atomic [1:1] 0.1  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## $ strip.switch.pad.wrap:Class 'unit' atomic [1:1] 0.1  
## .. ..- attr(\*, "valid.unit")= int 1  
## .. ..- attr(\*, "unit")= chr "cm"  
## - attr(\*, "class")= chr [1:2] "theme" "gg"  
## - attr(\*, "complete")= logi TRUE  
## - attr(\*, "validate")= logi TRUE

## Customizing Theme Elements

It is possible to create a new theme or modify an existing one.

Themes are made up of elements, which fall into four broad categories:

* element\_line
* element\_text
* element\_rect
* element\_blank

Customizing each of these elements provides a lot of control over plot appearance.

### Modifying Elements

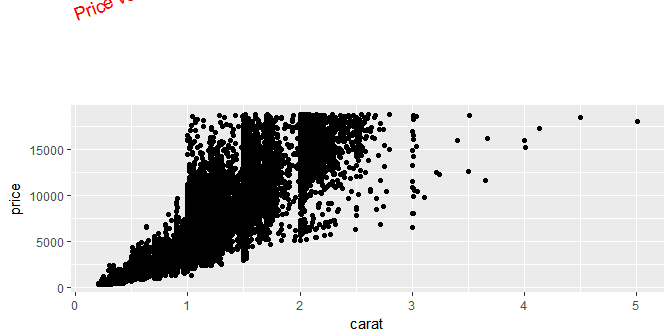
Here are some of the many plot elements that can be modified using theme():

* **Axis**: axis.line, axis.text.x, axis.text.y, axis.ticks, axis.title.x, axis.title.y
* **Legend**: legend.background, legend.key, legend.text, legend.title
* **Panel**: panel.background, panel.border, panel.grid.major, panel.grid.minor
* **Strip**: strip.background, strip.text.x, strip.text.y

## Modifying a plot

In many cases, it is desireable to modify an already existing plot that has been stored in an object.

p <- qplot(carat, price, data = diamonds) +   
 ggtitle("Price vs Carat for Diamonds")  
p + theme(plot.title = element\_text(colour = "red", angle = 20))

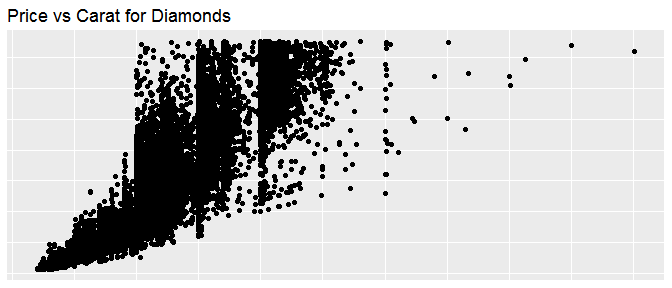


Use this power wisely: not all modifications are perceptually optimal.

## Removing Axes

It's also possible to remove all axes (helpful for maps):

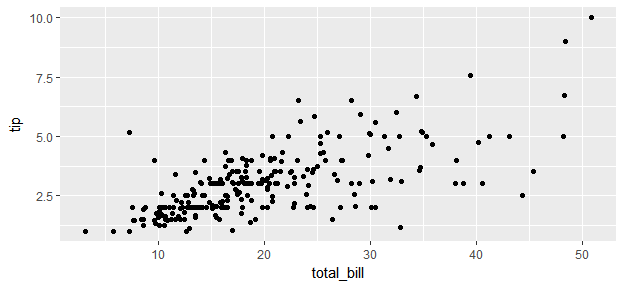
p + theme(  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 axis.title.x = element\_blank(),  
 axis.title.y = element\_blank(),  
 axis.ticks.length = unit(0, "cm")  
)



## Saving your Work

The ggsave() function will save the last plot produced:

qplot(total\_bill, tip, data = tips)  
  
ggsave("tips.png")  
ggsave("tips.pdf")  
ggsave("tips.png", width = 6, height = 6)



It is also possible to tell it which plot to save explicitly:

dplot <- qplot(total\_bill, tip, data = tips)  
ggsave("tips.png", plot = dplot, dpi = 72)

##### Your Turn

1. Save a pdf of a scatterplot of price vs carat
2. Open up the pdf in Adobe Acrobat (or another PDF Reader)
3. Save a png of the same scatterplot

##### Solutions

qplot(price, carat, data = diamonds)  
  
ggsave("diamonds.pdf")  
## Saving 6.5 x 4 in image  
  
ggsave("diamonds.png")  
## Saving 6.5 x 4 in image

Here are the links to the saved plots:

<diamonds.pdf>

<diamonds.png>