Chapter 2: Interactive Notebook for Instructors

Ram Gopal, Dan Philps, and Tillman Weyde

Summer 2022

Table of Contents

We will focus on two important tools – summarization and visualization – which can explain trends, patterns, and exceptions by leveraging our visual skills. In fact the goal of these tools is to use visual representations to explore, make sense of, and communicate with data. It is often said that statistics is about proving what we expect, whereas visualization is about sense making and discovering what we did not expect. We will explore the data visualization packages ggplot and plotly, along with the grammar of graphics principles that aid in the design of effective visualizations to drive insights and understanding.

# Indexing and Subsetting

Indexing and subsetting are ways of viewing specific parts of a dataframe. For example, we use them if we want to see the first four rows of the dataframe df, or the first, third, and sixth columns. The syntax is as follows:

df[rows,columns] df[,1]

Below are some examples using the ghw dataframe.

weight = c(60,72,57,90,95,72)   
height = c(1.75,1.8,1.65,1.9,1.74,1.91)   
gender = c("m","f","m","f","f","m")   
ghw=data.frame(gender,height,weight)   
ghw[1,1]

## [1] "m"

ghw[,1:3]

## gender height weight  
## 1 m 1.75 60  
## 2 f 1.80 72  
## 3 m 1.65 57  
## 4 f 1.90 90  
## 5 f 1.74 95  
## 6 m 1.91 72

ghw[2:5,]

## gender height weight  
## 2 f 1.80 72  
## 3 m 1.65 57  
## 4 f 1.90 90  
## 5 f 1.74 95

ghw[,1]

## [1] "m" "f" "m" "f" "f" "m"

ghw[c(1,3,6),2:3]

## height weight  
## 1 1.75 60  
## 3 1.65 57  
## 6 1.91 72

It is sometimes useful to store the subset in a new variable.

x = ghw[1:3,]   
dim(x)

## [1] 3 3

## Specifying Conditions

student <- read.csv("../../data/student.csv")   
x1 = student[student$daysabs==0,]   
head(x1)

## id gender math prog daysabs  
## 16 1016 0 89 2 0  
## 18 1018 1 35 2 0  
## 22 1022 0 61 2 0  
## 24 1024 0 63 2 0  
## 28 1028 1 21 2 0  
## 31 1031 0 1 2 0

x2 = student[student$daysabs==0 & student$math>60,]   
head(x2)

## id gender math prog daysabs  
## 16 1016 0 89 2 0  
## 22 1022 0 61 2 0  
## 24 1024 0 63 2 0  
## 35 1035 0 68 2 0  
## 71 1071 0 72 2 0  
## 85 1085 0 75 2 0

It is also useful to use formulas to create conditions. We can do this in R by saving a condition as a variable as shown below:

f1 = student$gender==0 & student$prog==3 & student$math>60   
x3 = student[f1,]   
head(x3)

## id gender math prog daysabs  
## 89 1089 0 84 3 4  
## 165 2007 0 71 3 0  
## 167 2009 0 71 3 0  
## 169 2011 0 77 3 2  
## 173 2015 0 65 3 1  
## 175 2017 0 77 3 0

# Summarizing Data

Summaries are a quick and simple way to see our data in an organized fashion. In R, we will be using the whiteside dataframe in the MASS package, and the Arthritis dataframe in the vcd package, which need to be installed.

library(MASS)  
library(vcd)

## Factor Variables

The command we use to create summary tables of factor variables is table().

table(Arthritis$Sex)

##   
## Female Male   
## 59 25

x = table(Arthritis$Treatment,Arthritis$Improved)   
x

##   
## None Some Marked  
## Placebo 29 7 7  
## Treated 13 7 21

If we want proportions in our table, we use the command prop.table().

x = table(Arthritis$Treatment,Arthritis$Sex)  
prop.table(x,margin = 1)

##   
## Female Male  
## Placebo 0.7442 0.2558  
## Treated 0.6585 0.3415

prop.table(x,margin = 2)

##   
## Female Male  
## Placebo 0.5424 0.4400  
## Treated 0.4576 0.5600

Three dimensional factor data requires the use of the command ftable. The purpose of ftable is to combine the data into one table, rather than three, in a manner that is simple to understand.

x = table(Arthritis$Treatment,Arthritis$Sex,Arthritis$Improved)  
ftable(x)

## None Some Marked  
##   
## Placebo Female 19 7 6  
## Male 10 0 1  
## Treated Female 6 5 16  
## Male 7 2 5

## Numeric Variables

To create a table with a factor variable and a numeric variable, we use the following syntax:

aggregate(numeric variable, list(factor variable), simple statistic command).

z = aggregate(whiteside$Temp,list(whiteside$Insul),mean)   
head(z)

## Group.1 x  
## 1 Before 5.350  
## 2 After 4.463

# Graphing with ggplot2

We will explore graphing with the ggplot2 package. In fact, there is a package called tidyverse which includes ggplot2 along with other really useful packages for data manipulation.

library(tidyverse)  
library(MASS)

Let us take a look at the data frame whiteside, which we used before.

head(whiteside)

## Insul Temp Gas  
## 1 Before -0.8 7.2  
## 2 Before -0.7 6.9  
## 3 Before 0.4 6.4  
## 4 Before 2.5 6.0  
## 5 Before 2.9 5.8  
## 6 Before 3.2 5.8

ggplot2 allows you to create data visualizations. The process involves starting with a basic canvas, and then layering in new aspects of the graphics using +. To create the basic canvas, we use the ggplot() command. Inside the parenthesis we define the dataframe and variables we want on the x and y axes. aes() is used to create the axes. Note that for a single variable we can choose either the x or y axis and ignore the other. The other key command is geom, which stands for geometric object. This is used mainly to create different types of graphs. We will start creating one-dimensional graphs and then move on to two or more dimensions.

## One Variable Plots

### One Factor Variable

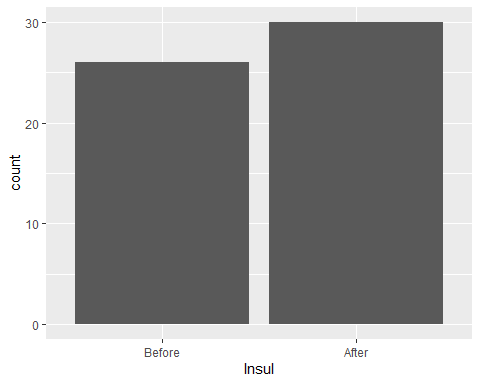
First, let us just create a basic canvas with the whiteside data set, where we want to plot Insul.

ggplot(whiteside,aes(x = Insul))



Since this is factor data, we will plot a simple bar graph.

ggplot(whiteside,aes(x = Insul)) +  
 geom\_bar()



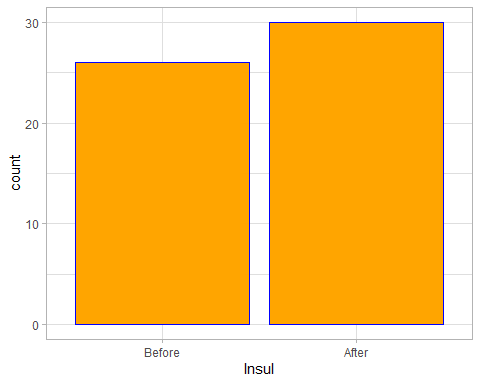
This is the concept of layering; we added a bar graph on top of the basic canvas. Now we can make a few additional adjustments. col makes the outline of the graph the color of your choice, and fill fills the space with the color of your choice.

ggplot(whiteside,aes(x = Insul)) +  
 geom\_bar(col = "blue", fill = "orange")



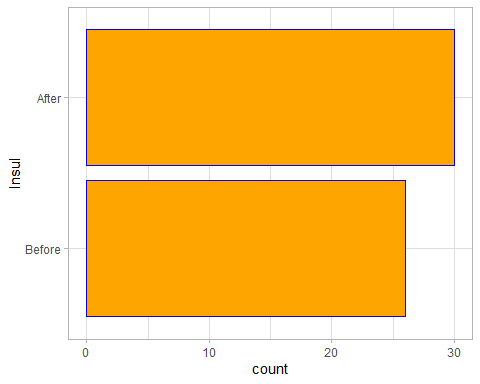
A few more adjustments. Let us change the background theme using theme.

ggplot(whiteside,aes(x = Insul)) +  
 geom\_bar(col = "blue", fill = "orange") +  
 theme\_light()



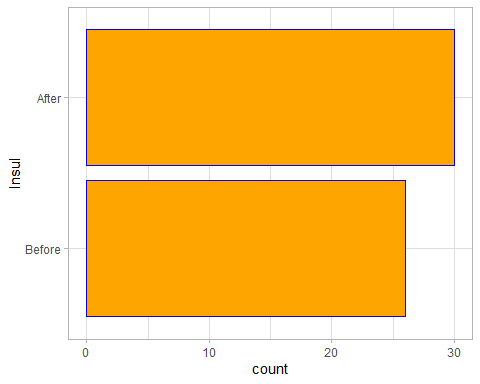
Now, let us flip the coordinates.

ggplot(whiteside,aes(x = Insul)) +  
 geom\_bar(col = "blue", fill = "orange") +  
 theme\_light() +  
 coord\_flip()



You can also flip the graph by putting the Insul data on the y axis in the aes() command.

ggplot(whiteside,aes(y = Insul)) +  
 geom\_bar(col = "blue", fill = "orange") +  
 theme\_light()

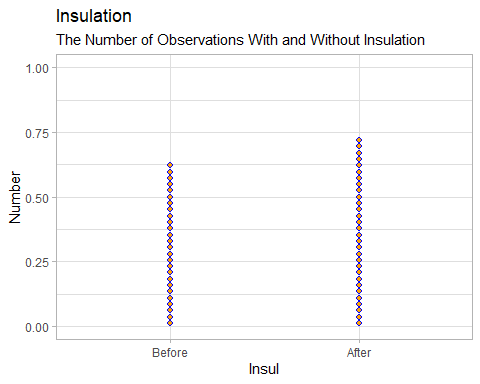


Another useful graph for factor data is a dot plot.

You can add titles and labels using labs().

ggplot(whiteside,aes(x = Insul)) +  
 geom\_dotplot(col = "blue", fill = "orange") +  
 theme\_light() +  
 labs(title = "Insulation",  
 subtitle = "The Number of Observations With and Without Insulation",  
 y = "Number")

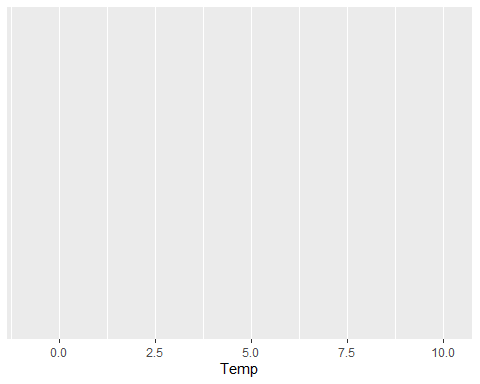
## Bin width defaults to 1/30 of the range of the data. Pick better value with `binwidth`.



This is a very simple graph with a one factor variable. Let us move on to graphs with a single numeric variable.

### One Numeric Variable

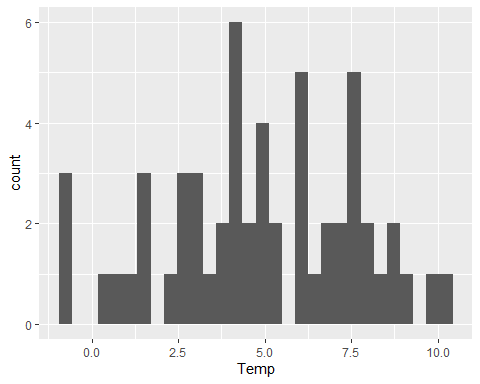
ggplot(whiteside, aes(x = Temp))



With a numeric variable, the simplest plot is a histogram.

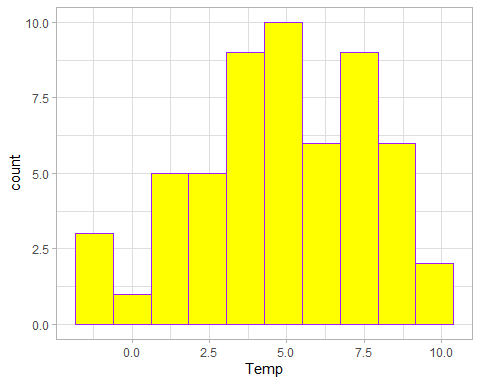
ggplot(whiteside,aes(x = Temp)) +  
 geom\_histogram()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



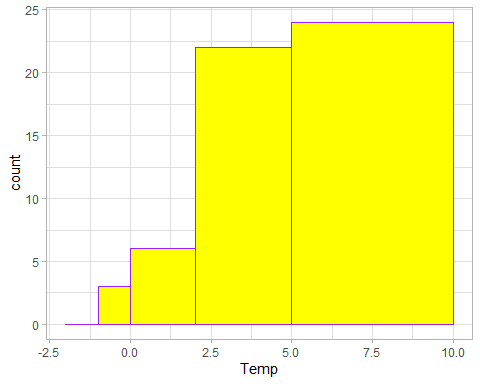
Obviously, we want to change how the numeric variable is binned to create the histogram.

ggplot(whiteside,aes(x = Temp)) +  
 geom\_histogram(bins = 10, col = "purple", fill = "yellow") +  
 theme\_light()



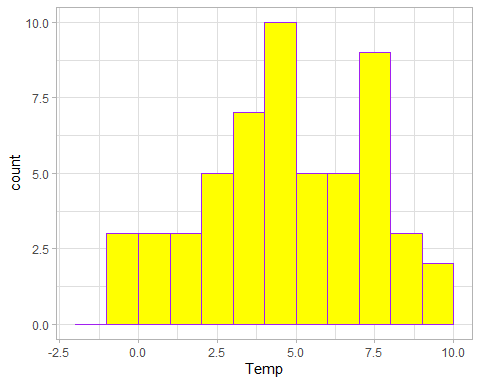
You can also specify the length of each bin using the breaks option.

ggplot(whiteside,aes(x = Temp)) +  
 geom\_histogram(breaks = c(-2,-1,0,2,5,10), col = "purple", fill = "yellow") +  
 theme\_light()



It is probably more reasonable to have meaningful breaks.

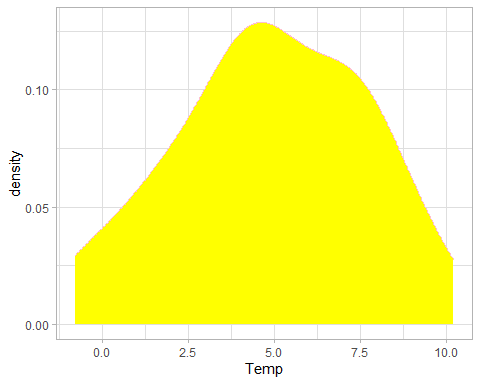
ggplot(whiteside,aes(x = Temp)) +  
 geom\_histogram(breaks = seq(-2,10,by = 1), col = "purple", fill = "yellow") +  
 theme\_light()



Sometimes, histograms don’t give a completely accurate depiction of the data. Imagine if you increase the number of bins, drew a point in the middle of each bin, and connected them with a line. This is how we create a density graph of our data. Density graphs are more useful because they give a more accurate picture of how the data is distributed. With histograms, it is a little confusing because the shape looks different depending on how the bins are defined. You can see this from the previous histograms of the temperature variable.

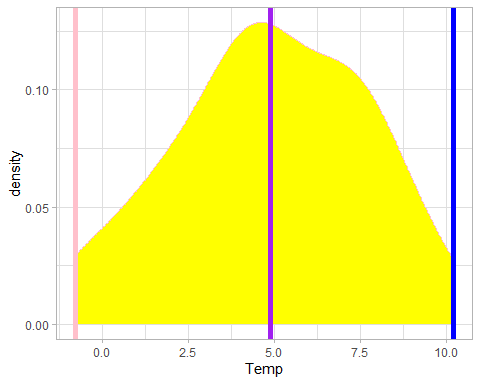
Now, let us create a density graph.

ggplot(whiteside,aes(x = Temp)) +  
 geom\_density(col = "pink", fill = "yellow") +  
 theme\_light()



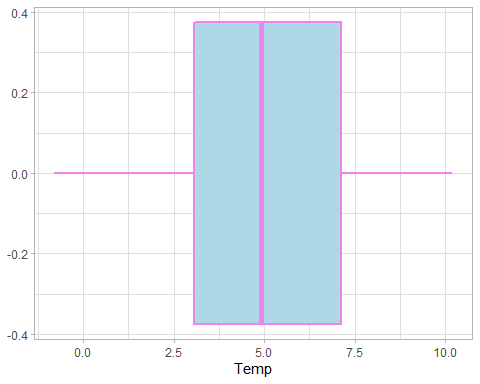
What if you want to create a vertical line at the mean, maximum, and minimum values of the temperature. First we need to compute the mean, max, and min, and then we can draw vertical lines at these values. You can change the color and size of the lines.

meantemp = mean(whiteside$Temp)  
mintemp = min(whiteside$Temp)  
maxtemp = max(whiteside$Temp)  
  
ggplot(whiteside,aes(x = Temp)) +  
 geom\_density(col = "pink", fill = "yellow") +  
 geom\_vline(xintercept = maxtemp, col = "blue", size = 2) +  
 geom\_vline(xintercept = mintemp, col = "pink", size = 2) +  
 geom\_vline(xintercept = meantemp, col = "purple", size = 2) +  
 theme\_light()



With a single numeric variable, we can also create a boxplot. The code is straightforward.

ggplot(whiteside,aes(x = Temp)) +  
 geom\_boxplot(col = "violet", fill = "lightblue", size = 1) +  
 theme\_light()

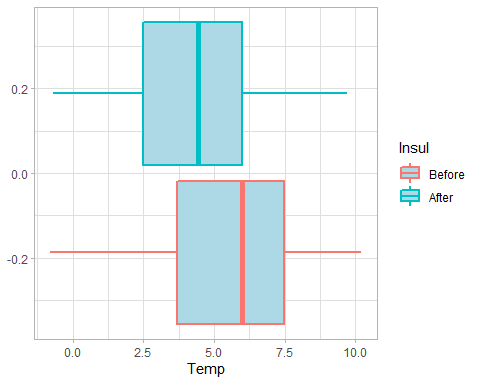


## Two+ Variable Plots

### Numeric + Factor Variable

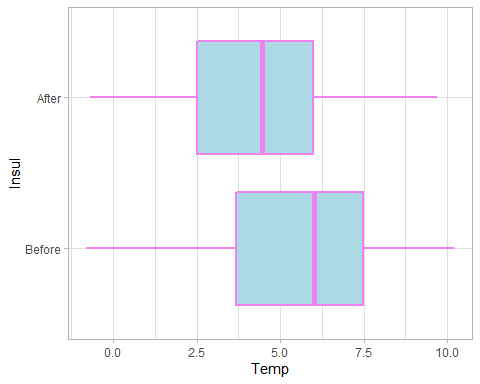
Suppose you want to extend the above box plot by also considering the insulation. In other words, you want two box plots, one before insulation and one after insulation. This is straightforward.

ggplot(whiteside,aes(x = Temp, col = Insul)) +  
 geom\_boxplot(fill = "lightblue", size = 1) +  
 theme\_light()



The above code creates a legend for the factor variable Insul. Another way to create a box plot for before and after insulation is to write y = Insul in the aes() command like so.

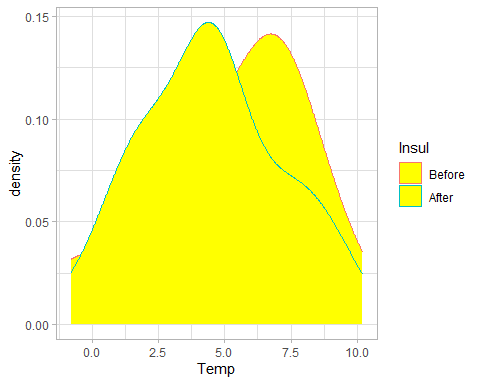
ggplot(whiteside,aes(x = Temp, y = Insul)) +  
 geom\_boxplot(col = "violet", fill = "lightblue", size = 1) +  
 theme\_light()



Instead of creating a legend, this method labels before and after Insul on the y axis.

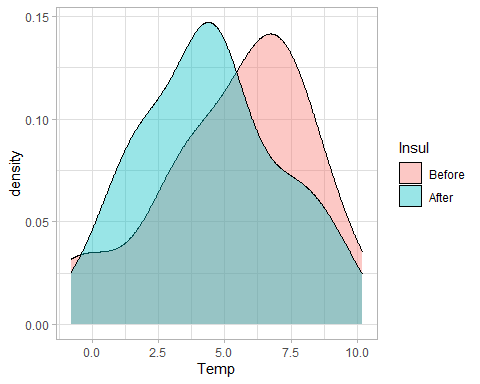
We can use the same logic for a density plot as well.

ggplot(whiteside, aes(x = Temp, col = Insul)) +  
 geom\_density(fill = "yellow") +  
 theme\_light()



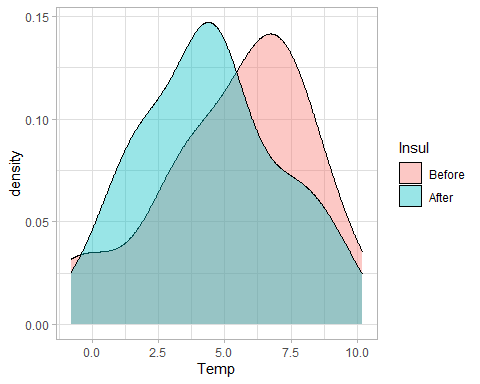
This doesn’t look very good. Here is an alternate to make both graphs clearer.

ggplot(whiteside, aes(x = Temp)) +  
 geom\_density(aes(fill = Insul), alpha = .4) +  
 theme\_light()



This works a little better, because we set alpha = .4, which improves the transparency of the graph. An alternate code which accomplishes the same task is:

ggplot(whiteside, aes(x = Temp, fill = Insul)) +  
 geom\_density(alpha = .4) +  
 theme\_light()



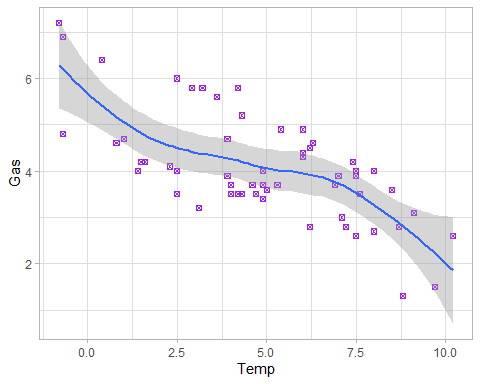
When you have one factor and one numeric variable, you can do a density or box plot, as we just demonstrated.

### Two+ Numeric Variables

Now, let us plot temperature against gas consumption.

ggplot(whiteside, aes(x = Temp, y = Gas)) +  
 theme\_light() +   
 geom\_point(pch = 7, col = "purple") +  
 geom\_smooth()

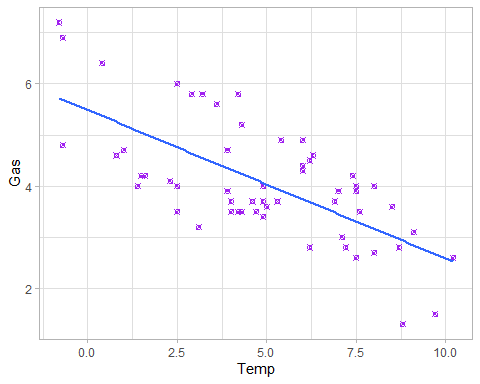
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



geom\_point gives you a scatterplot of the data. As before, you can change the design of the points using pch and the color using col. However, fill will not work. geom\_smooth gives you the best fit curve between x and y. The grey region is the 95% confidence interval. You can make a few adjustments as below.

ggplot(whiteside, aes(x = Temp, y = Gas)) +  
 theme\_light() +   
 geom\_point(pch = 13, col = "purple") +  
 geom\_smooth(method = lm, se = FALSE)

## `geom\_smooth()` using formula 'y ~ x'

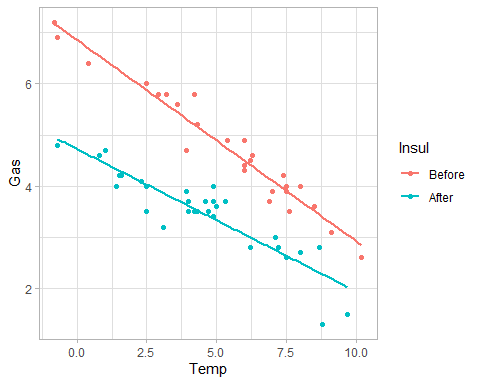


method = lm gives you a linear model of the relationship between x and y. se = FALSE removes the confidence interval around the smoothing function.

Now, we can also add a third dimension if it is a factor variable. In this example, suppose we want to plot the relationship between temperature and gas consumption both before and after insulation. One simple way to do this is to use the color option to add the third dimension.

ggplot(whiteside, aes(x = Temp, y = Gas, col = Insul)) +  
 theme\_light() +   
 geom\_point() +  
 geom\_smooth(method = lm, se = FALSE)

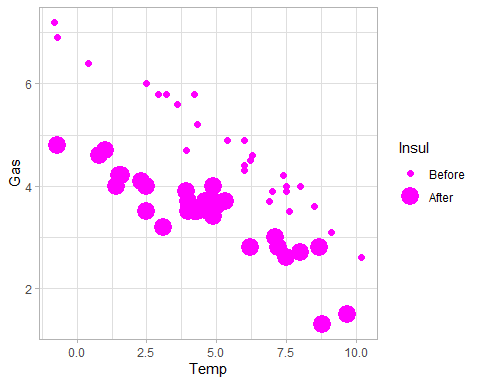
## `geom\_smooth()` using formula 'y ~ x'



As you can see, the graph clearly illustrates the gas consumption is lower after insulation, but the gap between the two narrows as the temperature increases.

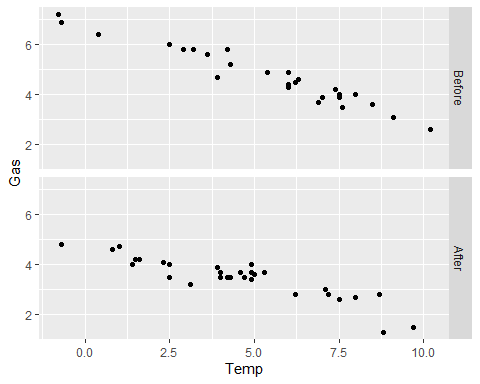
You can also use the size option to add another dimension.

ggplot(whiteside, aes(x = Temp, y = Gas, size = Insul)) +  
 theme\_light() +   
 geom\_point(col = "magenta")

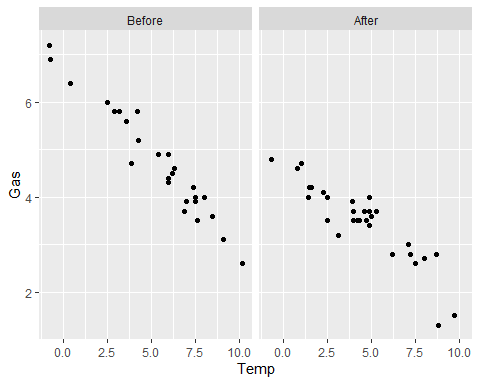


You can also use facets for the third dimension using the facet\_grid command. Facets are a way to create multiple subplots. We use the ~ in the facet function to control how the subplots are displayed. . is the replacement for other elements of the plot. We will try this a few different ways to show the results.

ggplot(whiteside, aes(x = Temp, y = Gas)) +  
 geom\_point() +  
 facet\_grid(Insul~.)



ggplot(whiteside, aes(x = Temp, y = Gas)) +  
 geom\_point() +  
 facet\_grid(.~Insul)



Notice that the variable to the left of the tilde makes up the rows and the variable to the right makes up the columns of the subplots.

### Two+ Factor Variables

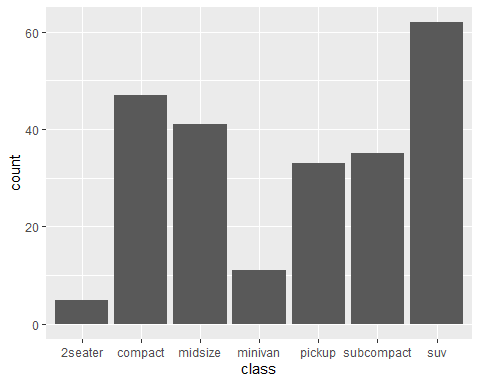
For graphing two factor variables, we will use the mpg data frame, as whiteside does not have two factor variables.

head(mpg)

## # A tibble: 6 × 11  
## manufacturer model displ year cyl trans drv cty hwy fl class   
## <chr> <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>   
## 1 audi a4 1.8 1999 4 auto(l5) f 18 29 p compa…  
## 2 audi a4 1.8 1999 4 manual(m5) f 21 29 p compa…  
## 3 audi a4 2 2008 4 manual(m6) f 20 31 p compa…  
## 4 audi a4 2 2008 4 auto(av) f 21 30 p compa…  
## 5 audi a4 2.8 1999 6 auto(l5) f 16 26 p compa…  
## 6 audi a4 2.8 1999 6 manual(m5) f 18 26 p compa…

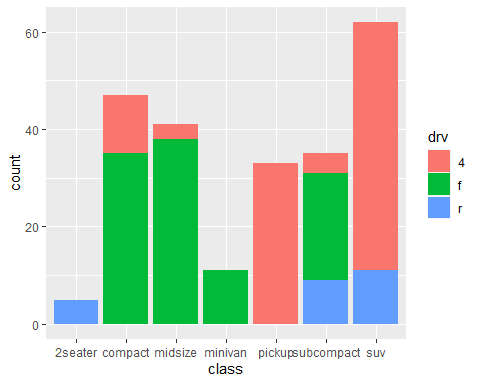
Let us first plot the class variable.

ggplot(mpg, aes(x = class)) +  
 geom\_bar()



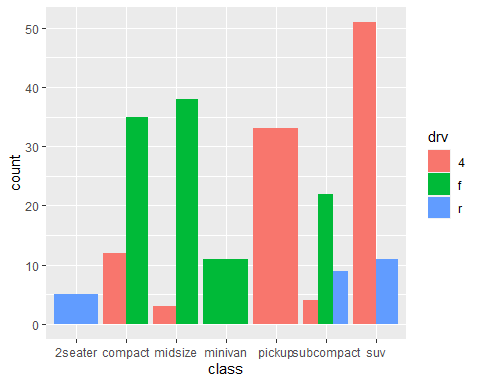
Now, suppose we also want to add drv to this plot.

ggplot(mpg, aes(x = class)) +  
 geom\_bar(aes(fill = drv))



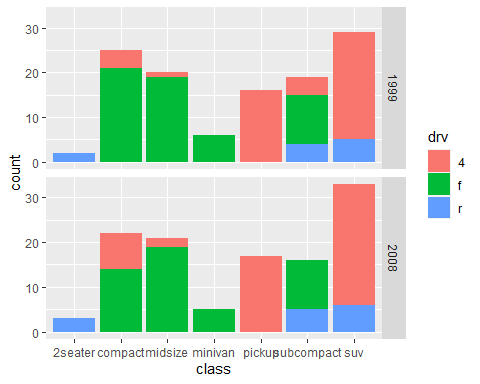
We use fill to add the second dimension. Here is an alternate view.

ggplot(mpg, aes(x = class)) +  
 geom\_bar(aes(fill = drv), position = "dodge")



We can also facet this.

ggplot(mpg, aes(x = class)) +  
 geom\_bar(aes(fill = drv)) +  
 facet\_grid(year~.)



Now we’ve added three dimensions to the figure.

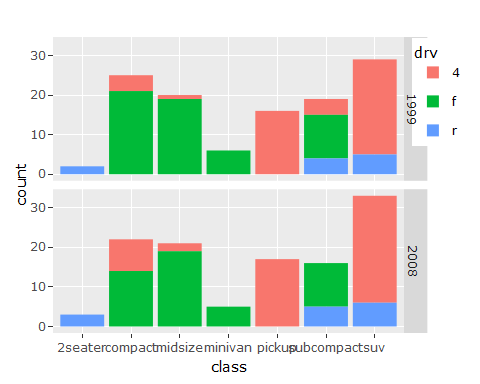
## Interactivity With Plotly

You can add some interactivity with the plotly package. Let us first load it.

library(plotly)

Once this package is loaded, you can add some interactivity for any ggplot. For example,

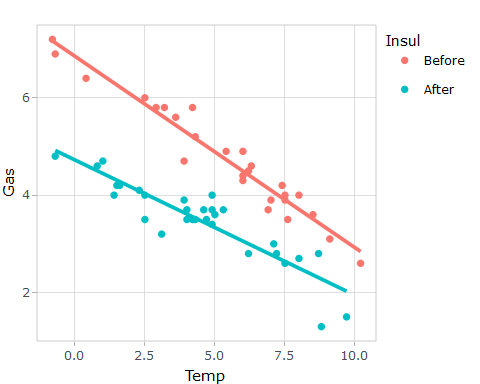
g1 = ggplot(mpg, aes(x = class)) +  
 geom\_bar(aes(fill = drv)) +  
 facet\_grid(year~.)   
ggplotly(g1)



Here is another example.

g2 = ggplot(whiteside, aes(x = Temp, y = Gas, col = Insul)) +  
 theme\_light() +   
 geom\_point() +  
 geom\_smooth(method = lm, se = FALSE)  
ggplotly(g2)

## `geom\_smooth()` using formula 'y ~ x'



# Final Remarks

Instructor: It would be beneficial to students to refer to the ggplot2 cheatsheet.

We have touched upon some of the basics with ggplot2 and plotly. There is a lot more you can do with it. Research it further and do check out some of the links below. One important thing to remember is that the plots you create be visually appealing, concise, and most importantly informative. For example, the various visualizations we did with the whiteside data frame tell us some useful information as follows:

1. There are more observations after insulation in comparison to before.
2. The median temperature after insulation is higher than before.
3. The gas consumption lowered after insulation, but the difference narrows as the temperature increases.

Good visualizations provide you good insights.

# Useful Links

[link] <https://dereksonderegger.github.io/570L/9-graphing-using-ggplot2.html>  
[link] <https://r4ds.had.co.nz/data-visualisation.html>  
[link] <https://uc-r.github.io/ggplot_intro>  
[link] <https://www.r-graph-gallery.com/>