Unit 6

R for Graphs

Amazing.

Design ..... **Data Collection** ..... **Data Management** ..... **Data Summarization** ..... **Statistical Analysis** ..... **Reporting**

Table of Contents

Follow along!
At the bottom of the next page (page 3), I've listed the packages and datasets used

Learning Objectives

When you have finished this unit, you should be able to use R and **`ggplot2`**, to produce some basic data visualizations, including:

- Single and two discrete variables: bar graph, grouped bar chart;
- Single continuous variable: dot, box and whisker, stem and leaf, histogram;
- Continuous variable, by group: Side-by-Side Dot, Side-by-Side Box;
- Continuous variable, by group: Mean $\pm$ SD (or SE or 95% CI);

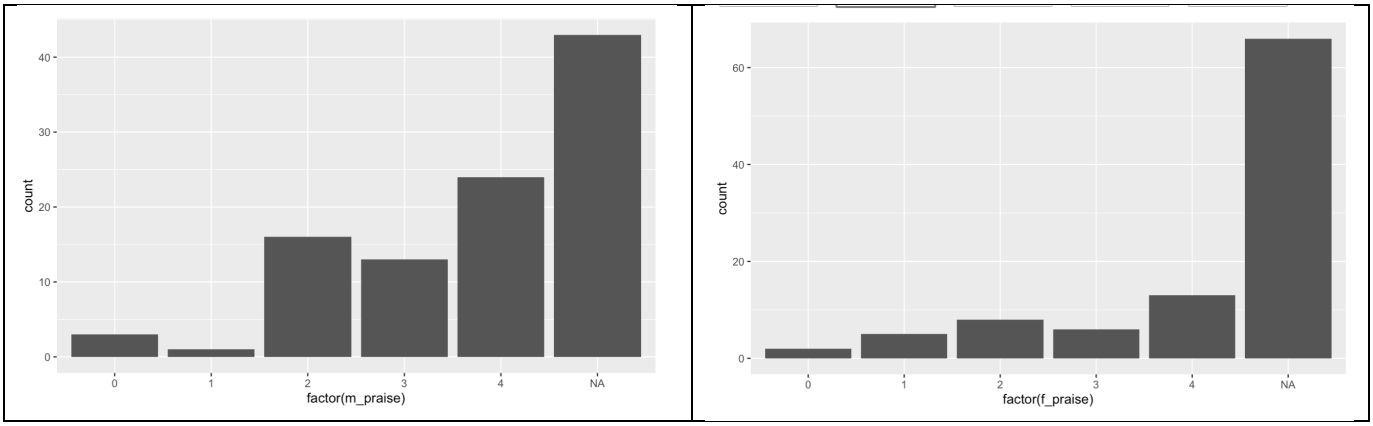
Packages Used in These Notes **Be sure to have done a one-time installation:** **__1. tidyverse (note: ggplot2 is a componenet)** **__2. Hmisc** **__3. summarytools** **__4. aplpack**	**Data Used in These Notes** **Right click to download from the course website** **https://people.umass.edu/~biostat690c/** **__1. framingham.Rdata** **__2. relate100obs.Rdata** **__3. auto.Rdata**

Sample Session

```
#  Attach packages tidyverse and Hmisc
library(tidyverse)
library(Hmisc)

#  Set working directory.  Input data
setwd("~/Desktop")
load(file=”relate100obs.Rdata")

#  Rename variables to be more meaningful
relate100obs <- dplyr::rename(relate100obs,
                m_praise = R3483600,
                f_praise = R3485200,
                age = R3828100)

#  Convert missing value codes to NAs
relate100obs[relate100obs==-1] <- NA
relate100obs[relate100obs==-2] <- NA
relate100obs[relate100obs==-4] <- NA
relate100obs[relate100obs==-5] <- NA

#  Create variable label
var.labels = c(age="Age of R (years)",
               f_praise="f_praise: Father praises R for doing well",
               m_praise="m_praise: Mother praises R for doing well")

#  Attach variable labels
label(relate100obs) = as.list(var.labels[match(names(relate100obs), names(var.labels))])


#  Label variable values
levels(relate100obs$f_praise)<- c("never","rarely","sometimes", "usually", "always")


# Keep only the variables of interest.
relatenew100 <- relate100obs %>%
               select(m_praise, f_praise, age)


#  Save under new name.
save(relatenew100, file = "relatenew100.RData")
```

```
library(ggplot2)

# Single Discrete Variable:  Bar graph
# Bar Graph - NO aesthetics:  m_praise

ggplot(data=relatenew100,aes(x=factor(m_praise))) +
    geom_bar()

# Bar Graph - NO aesthetics:  f_praise
ggplot(data=relatenew100,aes(x=factor(f_praise))) +
    geom_bar()
```

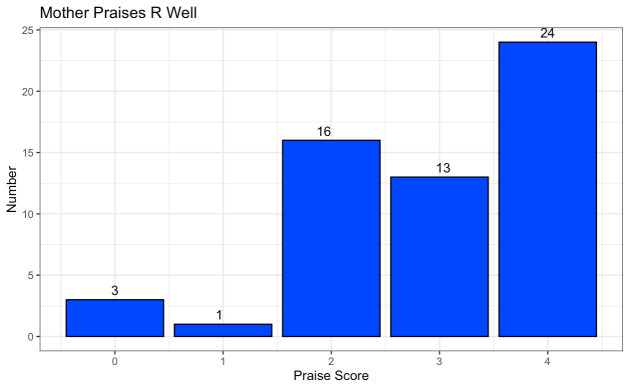
This yields the following

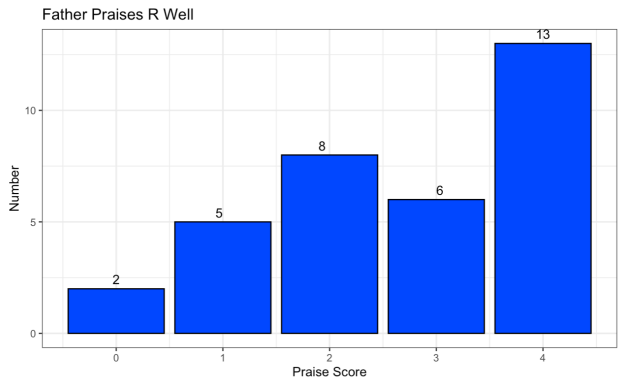


Note – The NA are showing (we might want to exclude those) and we might like some annotations; e.g. – titles, etc.

```
# Bar Graph - WITH aesthetics:  m_praise
ggplot(data=relatenew100, aes(x=m_praise))+
        geom_histogram(stat="count",binwidth = 0.5, color="black", fill="blue",
               na.rm = TRUE)  +
        stat_bin(aes(label=ifelse(..count.. > 0, ..count.., ""),y=..count..),
               geom="text", vjust=-.5,na.rm = TRUE)   +
        ggtitle("Mother Praises R Well")    +
        xlab("Praise Score")   +
        ylab("Number")   +
        theme_bw()

# Bar Graph - WITH aesthetics:  f_praise
ggplot(data=relatenew100, aes(x=f_praise))+
        geom_histogram(stat="count",binwidth = 0.5, color="black", fill="blue",
               na.rm = TRUE)  +
        stat_bin(aes(label=ifelse(..count.. > 0, ..count.., ""),y=..count..),
               geom="text", vjust=-.5,na.rm = TRUE)  +
        ggtitle("Father Praises R Well")  +
        xlab("Praise Score") +
        ylab("Number") +
        theme_bw()
```

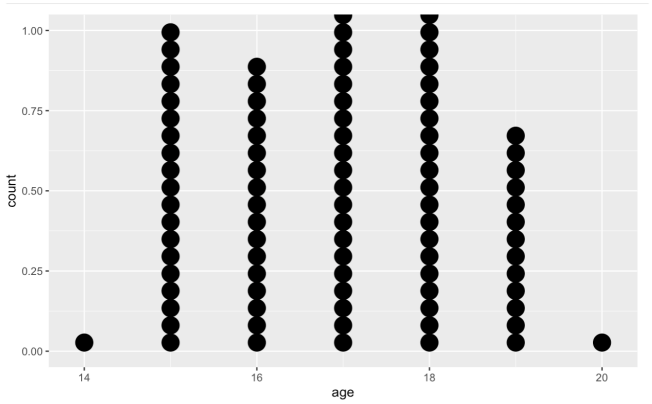
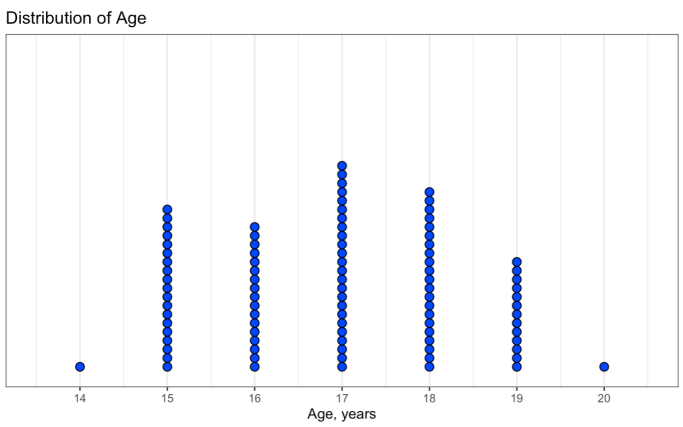
This yields the following





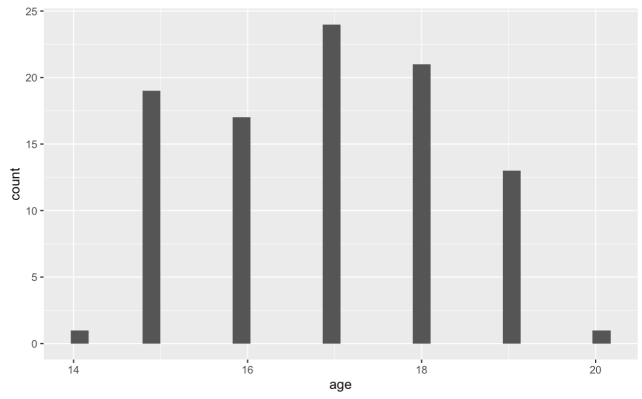
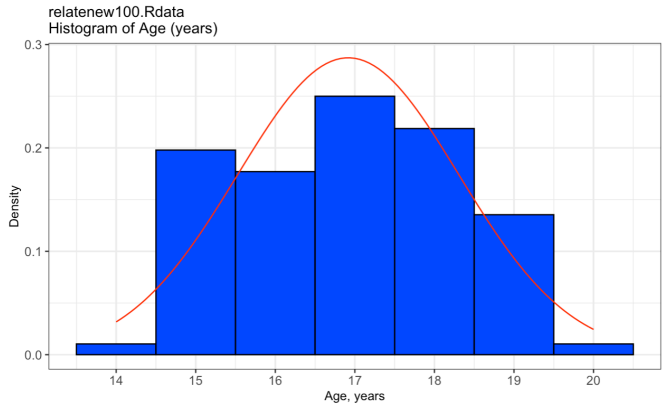
```
library(ggplot2)

# Single Continuous:  Dot Plot

# No Aesthetics
ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot()

# With Aesthetics:  Y axis is meaningless, change, color, change dot, label
ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot(binwidth = 1, color="black", fill="blue", dotsize=0.1) +
    scale_y_continuous(NULL, breaks = NULL)   +
    ggtitle("Distribution of Age")   +
    scale_x_continuous(breaks=seq(14, 20, 1))   +
    xlab("Age, years") +
    theme_bw()
```

This yields the following

No Aesthetics	With Aesthetics
	

```
library(ggplot2)

# Single Continuous:  Histogram

# No Aesthetics
ggplot(data=relatenew100, aes(x = age)) +
    geom_histogram()


# With Aesthetics
ggplot(data=relatenew100, aes(x=age)) +
    geom_histogram(binwidth=1, colour="black", fill="blue", aes(y=..density..)) +
    stat_function(fun=dnorm, colour="red",
                  args=list(mean=mean(relatenew100$age,na.rm=TRUE),
                            sd=sd(relatenew100$age, na.rm=TRUE))) +
    scale_x_continuous(breaks=seq(14, 20, 1))+
    ggtitle("relatenew100.Rdata  \nHistogram of Age (years)") +
    xlab("Age, years") +
    ylab("Density") +
    theme_bw() +
    theme(axis.text=element_text(size=10),
          axis.title=element_text(size=10),
          plot.title=element_text(size=12))
```

This yields the following

No Aesthetics	With Aesthetics
	

1. Introduction to `ggplot2`

1.1 Grammar of `ggplot`

The **grammar of a sentence** has, minimally (required), a noun and verb and also include some extras (adjectives, adverbs, etc!).

Similarly, the **grammar of a ggplot plot** has some minimal (required) components together with one or more (optional) additional specifications.

Grammar of a ggplot graph

Minimal (Required) Components:

`data =`	Tell R what data frame to use
`aes( )`	*Mapping*: What is on X-axis, Y-axis and what (if any) is the 3rd variable
`geom_XXX( )`	Tell R what type of plot to produce (e.g. dot, box, scatterplot, etc.)

Additional Specifications:

`aesthetic`	Tell R what you want vis a vis: position, size, color, shape, fill, etc.
`scale`	Customize scale; e.g. – log scale, color scale, size, shape
`stat =`	Tell R to overlay statistical summaries; e.g. – regression line, overlay normal
`facets`	Tell R to do separate graphs by group or to create multi-panel graphs.

1.2 Build Your Plot Layer by Layer

How to Build Your Plot Layer by Layer:

- Be sure to have attached the package ggplot2 or tidyverse. Either will work.
  `library(ggplot2)` or
  `library(tidyverse)`

- *Recommended:*
  As you are learning, create each plot in its own chunk in an R Markdown

Code and execute the minimal components

- Create your first line of code with the first two minimal components `data =` and `aes( )` Put a + at the end of this line.

- Create your second line of code with the third minimal component `geom_xxx( )`
  Highlight and do a <control-enter> to execute the first and second lines
  Make corrections, re-issue and repeat until these lines of code execute successfully
  All set? **Put a + at the end of this line**.
  You are ready to move on to add the 3rd line of code

Code and execute each additional (optional) specification, *one at a time*

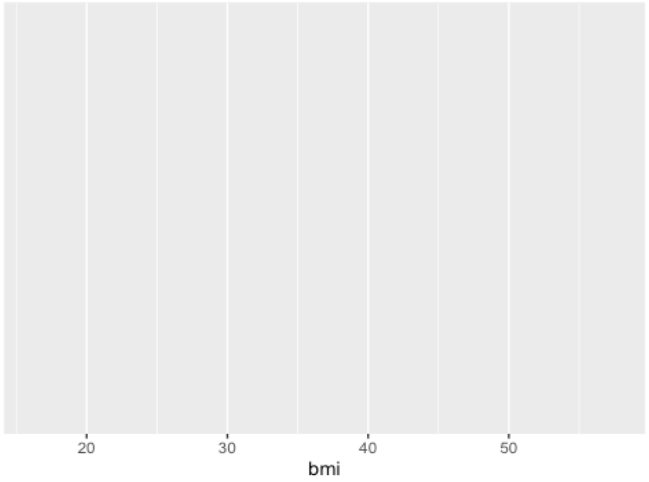
- Create each subsequent specification in its own line **WITHOUT a +** at the end of the line
  Highlight and do a <control-enter> to execute
  Make corrections, re-issue and repeat until this line of code executes successfully
  All set? **Put a + at the end of this line**.
  You are ready to move on to add the next line of code

- and so on ….

IMPORTANT: ***The continuation character "+" must go at the end of the line, NOT the start of the next line!***

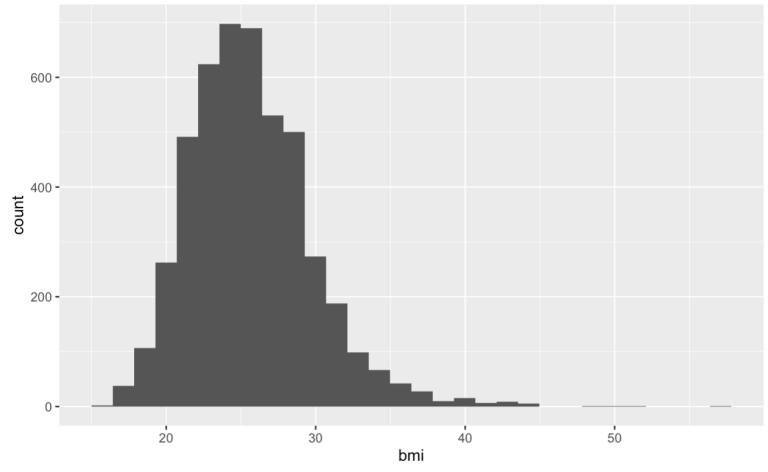
Illustration

```
# Attach ggplot2 or tidyverse
library(ggplot2)

# Minimal Components: data= and aes( ) +
# Tell R which data to use.   Tell R how to map variable to graph
ggplot(data=framinghamdf, aes(x=bmi)) +
```



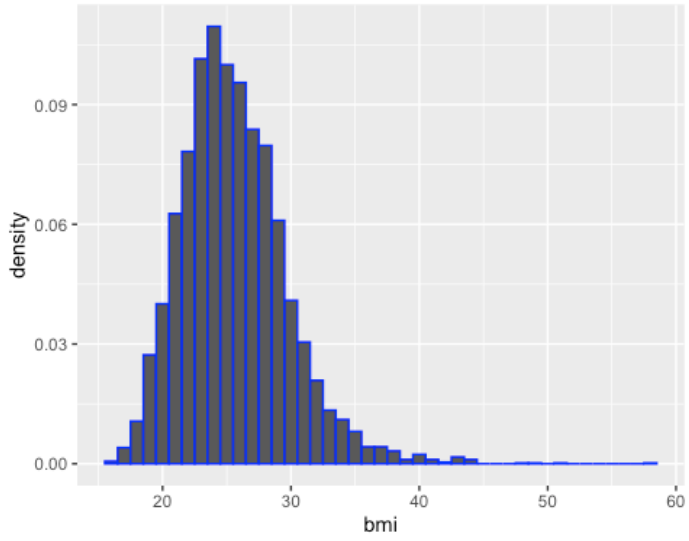
You'll get a blank plot

```
# Minimal Component: geom_XXX( ) +
# Tell R which kind of plot to produce
geom_histogram( ) +
```



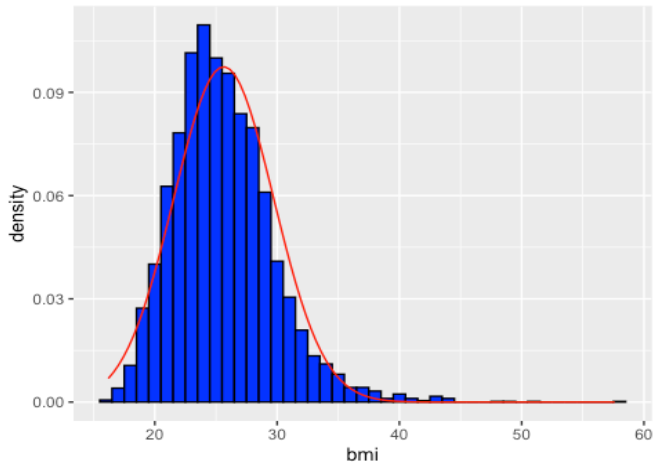
You get a basic plot – no frills.

```
# Additional Specification:    +
# binwidth, color, y-axis
geom_histogram(binwidth=1, colour="blue", aes(y=..density..)) +
```



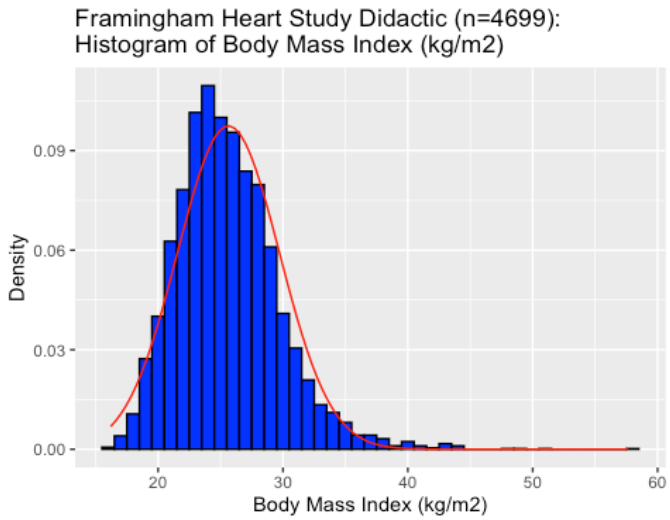
This looks a little better…

```
# Additional Specification:  stat_function( ) +
# Note:  This is actually an argument of the geom_SOMETHING( )
# Here we are telling R overlay a statistical calculation, in particular an overlay nor
mal curve
# Tip:  Be sure to include the option na.rm=TRUE.  Reason:  Calculations will not
# happen if there are missing values
                                                                              +
 stat_function(fun=dnorm, color="red", args=list(mean=mean(framinghamdf$bmi,na.rm=TRUE),
      sd=sd(framinghamdf$bmi,na.rm=TRUE))) +
```

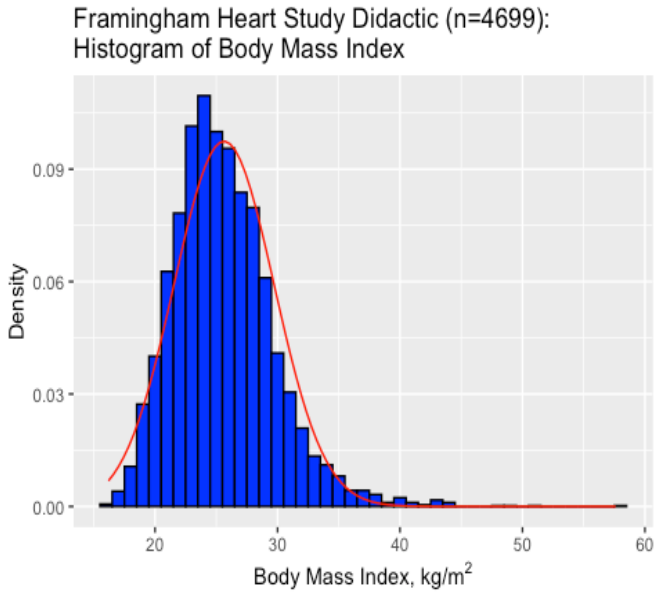


Pretty!

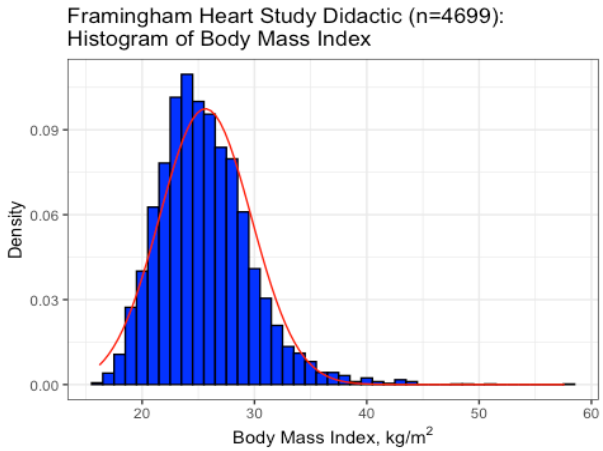
```
# Additional Specification:  ADD TITLE, LABELS, AXIS LIMITS, etc +
ggtitle("Framingham Heart Study Didactic (n=4699):  \nHistogram of Body Mass Index (kg/
m2)") +
    xlab("Body Mass Index (kg/m2)") +
    ylab("Density") +
```



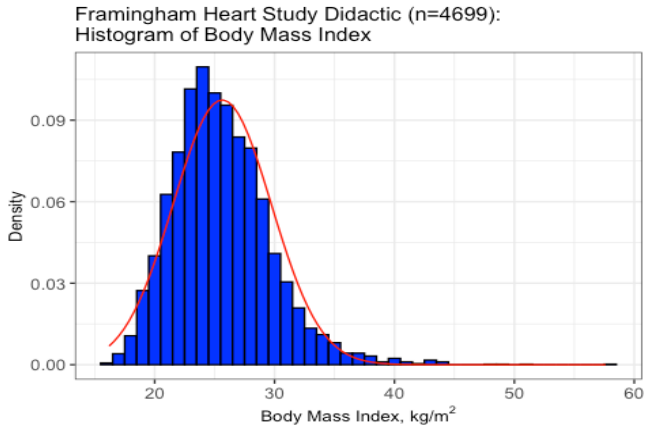

```
# Additional Specification:  Carol decides to superscript for meters squared
ggtitle("Framingham Heart Study Didactic (n=4699):  \nHistogram of Body Mass Index") +
    xlab(expression("Body Mass Index, kg/m"^{2} ))  +
    ylab("Density") +
```



```
# Additional Specification:  theme_xxx( ) +
# Final theme customization to make your graph especially good looking!
# Here we use the theme theme_bw() to get rid of the grey in the plotting area
theme_bw() +
```

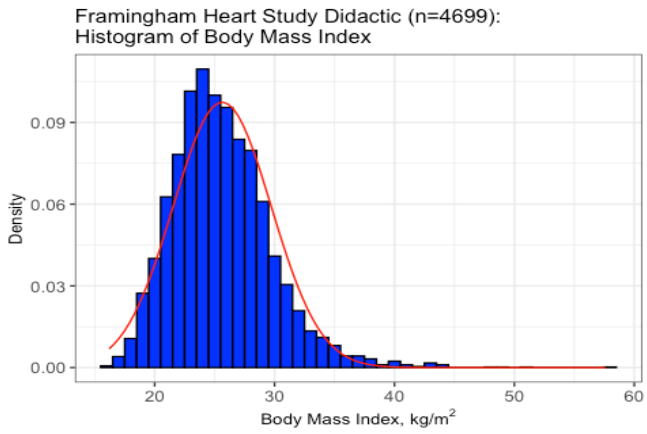



```
# Additional Specification: Fine tune the sizes of the title and axis titles

theme(axis.text=element_text(size=10),
          axis.title=element_text(size=10),
          plot.title=element_text(size=12))
```



```
# Putting it all together
# Illustration:  Histogram with Overlay Normal

ggplot(data=framinghamdf, aes(x=bmi)) +

     geom_histogram(binwidth=1, colour="blue", aes(y=..density..)) +

     stat_function(fun=dnorm, color="red",
         args=list(mean=mean(framinghamdf$bmi,na.rm=TRUE),
         sd=sd(framinghamdf$bmi,na.rm=TRUE))) +

      ggtitle("Framingham Heart Study Didactic (n=4699):  \nHistogram of Body Mass Inde
x (kg/m2)") +

      xlab("Body Mass Index (kg/m2)") +

      ylab("Density") +

      theme_bw() +

      theme(axis.text=element_text(size=10),
          axis.title=element_text(size=10),
          plot.title=element_text(size=12))
```

You get the following…



1.3 How to Save Your Graph

How to Save Your Graph

- Begin your chunk with the following command:

```
library(ggplot2)
```

- Edit your code so that your graph is assigned to an object. For example:

```
p <- ggplot(stuff) +
     morestuff +
     etc
```

- Execute your code so as to create your object `p`

- To save your graph use the command ggsave( )

```
ggsave(file="FILENAME",(p, option, option)
```

Your graph will be saved to your working directory

Example:

```
ggsave(file="mygraph.tiff",p, width=7, height=5, units="in")
```

Extensions Allowed with `ggsave( )`

"eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windows only).

Options (not complete) allowed with `ggsave( )`

width =
height =
units =
dpi =
scale =
limitsize =

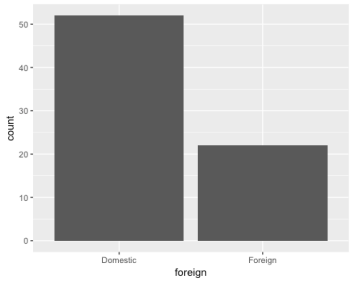
2. Single Variable Graphs – Discrete

2.1 Bar Chart

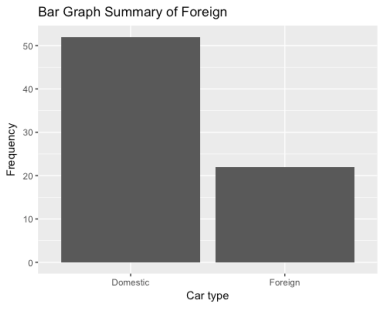
Single Discrete Variable – Bar Chart

```
ggplot(data=dataframe, aes(x=discretevar)) +
    geom_bar(na.rm=T)
```

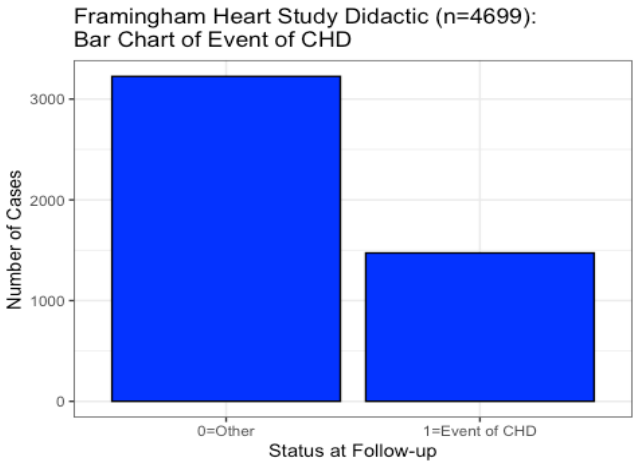
```
# Example 1 – Minimal Bar Graph
# na.rm=T removes missing values prior to plotting (required if there are missing data)
ggplot(data=auto,aes(x=foreign))+
  geom_bar(na.rm=T)
```




```
# Example 2 - Bar Graph with aesthetics
ggplot(data=auto, aes(x=foreign))+
  geom_bar(na.rm=T)+
  labs(x="Car type",y="Frequency", title="Bar Graph Summary of Foreign")
```



```
# Example 3 - Bar Graph with aesthetics
# \n produces carriage return so that title continues on a new line
ggplot(data=framinghamdf,aes(x=factor(chdfate))) +
    geom_bar(color="black", fill="blue",show.legend = FALSE) +
    ggtitle("Framingham Heart Study Didactic (n=4699):  \nBar Chart of Event of CHD") +
    xlab("Status at Follow-up") +
    ylab("Number of Cases") +
    theme(legend.position = "none") +
    theme_bw()
```

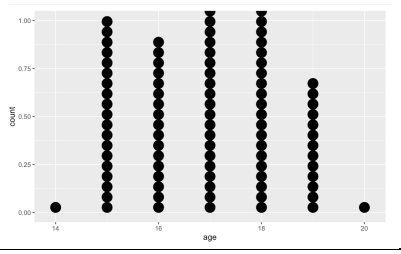


3. Single Variable Graphs - Continuous

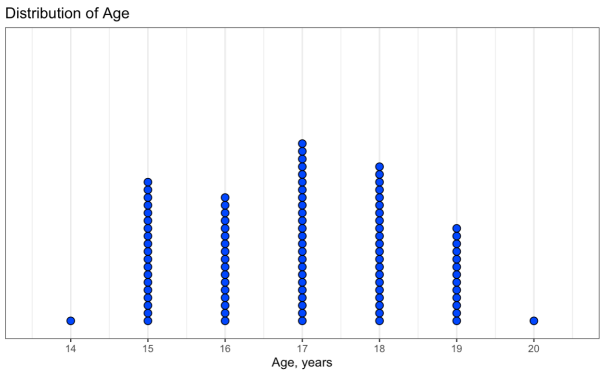
3.1 Dot Plot

```
Single Continuous Variable – Dot Plot

ggplot(data=dataframe, aes(x=continuousvar))+
    geom_dotplot(na.rm=T)
```

```
# Example 1 – Minimal Dot Plot
ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot()
```




```
# Example 2 – Dot Plot with aesthetics
ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot(binwidth = 1, color="black", fill="blue", dotsize=0.1) +
    scale_y_continuous(NULL, breaks = NULL) +
    ggtitle("Distribution of Age")+
    scale_x_continuous(breaks=seq(14, 20, 1))+
    xlab("Age, years") +
    theme_bw()
```

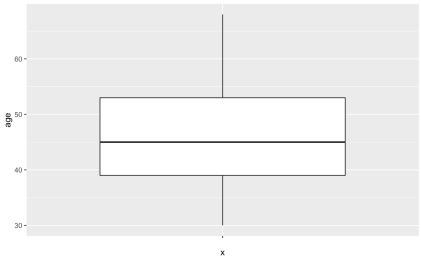


3.2 Box and Whisker Plot

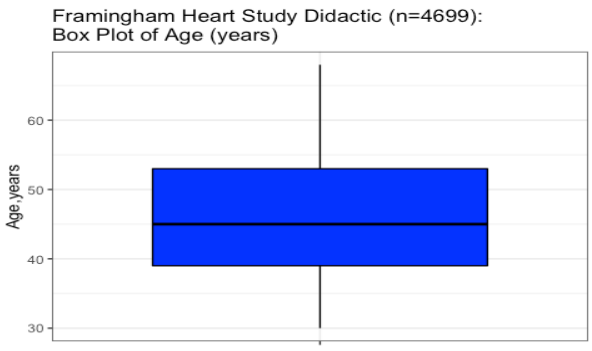
Single Continuous Variable – Box and Whisker Plot

```
ggplot(data=dataframe, aes(x=" ",y=continuousvar))+
    geom_boxplot(na.rm=T)
```

```
# Example 1 – Minimal Box Plot
ggplot(data=framinghamdf, aes(x="", y=age)) +
    geom_boxplot(na.rm=T)
```




```
# Example 2 – Box Plot with Aesthetics
ggplot(data=framinghamdf, aes(x="", y=age)) +
    geom_boxplot(na.rm=T,color="black", fill="blue") +
    xlab("") +
    ylab("Age,years") +
    ggtitle("Framingham Heart Study Didactic (n=4699):  \nBox Plot of Age (years)") +
    theme_bw()
```



3.3 Stem and Leaf

Single Continuous Variable – Stem and Leaf

```
stem(dataframe$continuousvar)
```

Note: ggplot2 does not have a geom_xxx( ) for creating a stem and leaf plot. To obtain a stem and leaf plot, use the function stem( ) in the base package.

```
# Example
stem(auto$mpg)
```

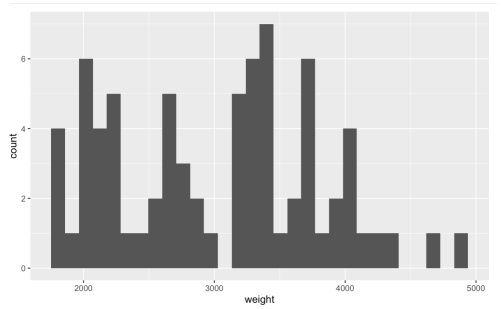
```
 The decimal point is 1 digit(s) to the right of the |

  1 | 22444444
  1 | 55666677778888888888999999999
  2 | 00011111222223334444
  2 | 555556668889
  3 | 0014
  3 | 55
  4 | 1
```

3.4 Histogram

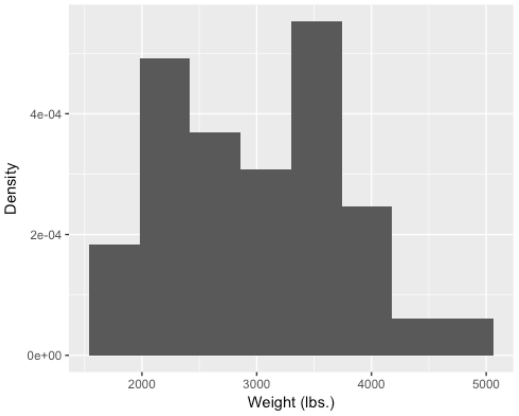
Single Continuous Variable – Histogram

```
ggplot(data=dataframe, aes(x=continuousvar))+
    geom_histogram(na.rm=T)
```

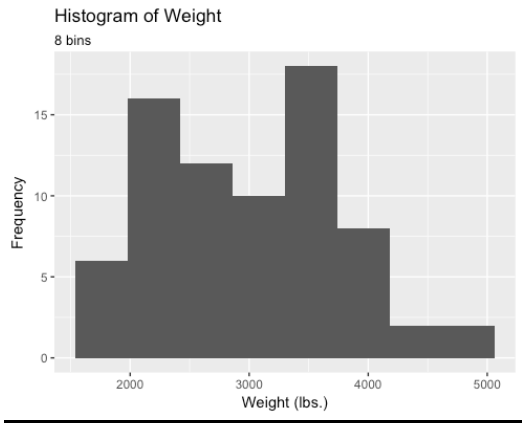
```
# Example 1 – Minimal Histogram
ggplot(data=auto, aes(x=weight))+
  geom_histogram(na.rm=T)
```



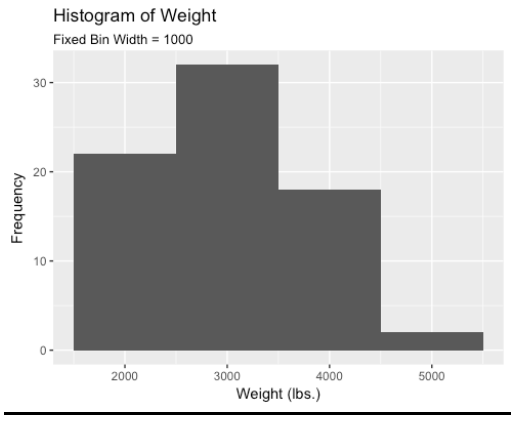

```
#  Example 2 – Histogram with Aesthetics
#  User sets # bins.  Y-axis is also changed to show density (I don't like it, frankly)
ggplot(data=auto, aes(x=weight,y=..density..))  +
  geom_histogram(na.rm=T,bins=8)  +
  labs(x="Weight (lbs.)",y="Density")
```



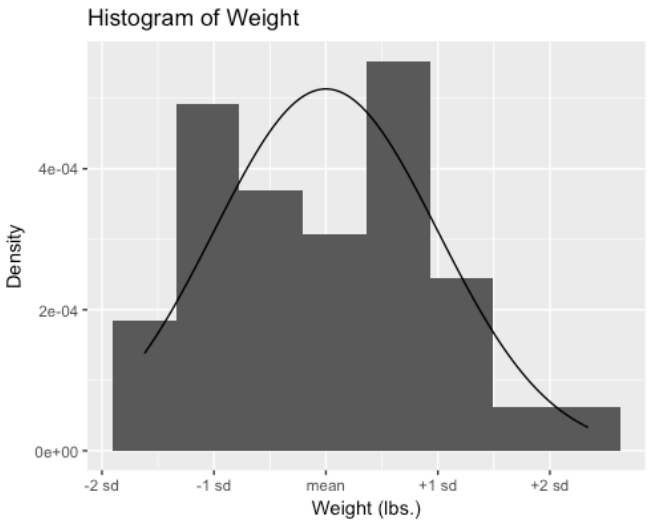
```
# Example 3 – Histogram with Aesthetics
# User sets number of bins
# Y-axis changed to show frequency (I like this better)

ggplot(data=auto, aes(x=weight,y=..count..))   +
  geom_histogram(na.rm=T, bins=8) +
  labs(x="Weight (lbs.)",y="Frequency", title="Histogram of Weight", subtitle="8 bins")
```




```
# Example 4 – Histogram with Aesthetics
#  User sets width of bins
ggplot(data=auto, aes(x=weight,y=..count..))+
  geom_histogram(na.rm=T, binwidth=1000)+
  labs(x="Weight (lbs.)",y="Frequency",title="Histogram of Weight",
          subtitle="Fixed Bin Width = 1000")
```



```
# Example 5 – Histogram with Aesthetic:  tick marks at units of SD and an overlay normal
 mean(auto$weight)
[1] 3019.459
sd(auto$weight)
[1] 777.1936


# *Identify tick marks at multiples of sd
3019.459-(1*777.1936)
2242.2654
3019.459-(2*777.1936)
1465.0718
3019.459-(3*777.1936)
687.8782
3019.459+(1*777.1936)
3796.6526
3019.459+(2*777.1936)
4573.8462
3019.459+(3*777.1936)
5351.0398

ggplot(data=auto) +
  geom_histogram(aes(x=as.numeric(weight),y=..density..), bins=8, na.rm=T) +
  scale_x_continuous(breaks=c(3019.459, 2242.2654,1465.0718,687.8782, 3796.6526,
4573.8462,5351.0398),labels=c("mean", "-1 sd", "-2 sd","-3 sd",
"+1 sd", "+2 sd","+3 sd"))+
  labs(x="Weight (lbs.)",y="Density",title="Histogram of Weight")+
  stat_function(fun = function(x, mean, sd){
      dnorm(x = x, mean = mean, sd = sd)},
    args = c(mean = mean(as.numeric(auto$weight),na.rm = TRUE),
             sd = sd(as.numeric(auto$weight),na.rm = TRUE)))
```



4. Multiple Variable Graphs

4.1 Two Discrete: Grouped Bar Chart

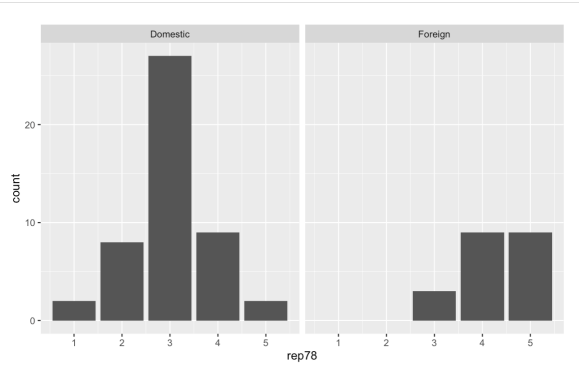
Two Discrete Variables – Grouped Bar Chart

```
ggplot(data=dataframe, aes(x=discretevar1))+
    geom_bar(na.rm=T)+
    facet_wrap(~groupvariable)
```

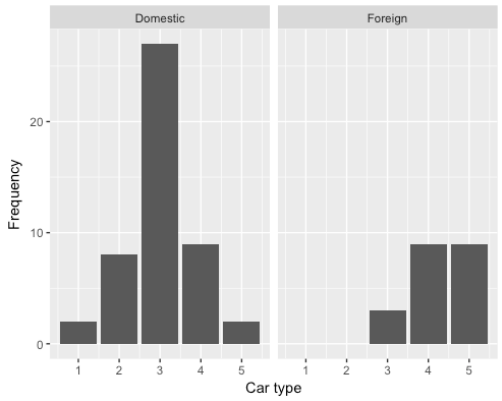
How to change arrangement of panels:

```
    facet_wrap(~groupvariable)                  # Horizontal:  Panels in 1 row, 2 columns
    facet_wrap(~groupvariable), nrow=2)         # Vertical: Panels in 2 rows, 1 column OR
    facet_wrap(~groupvariable), ncol=1)         # Vertical: Panels in 2 rows, 1 column
```

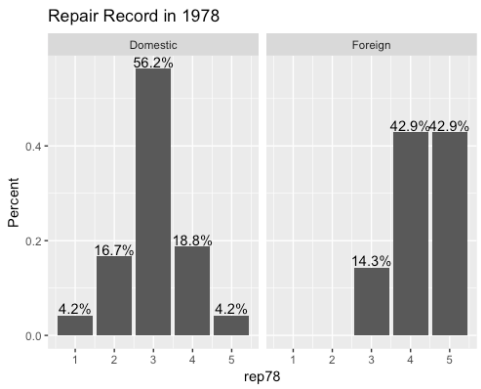
```
# Example 1 – MINIMAL Two Discrete Variables Grouped Bar Chart
ggplot(data=auto,aes(x=rep78))+
  geom_bar(na.rm=T)+
  facet_wrap(~foreign)
```



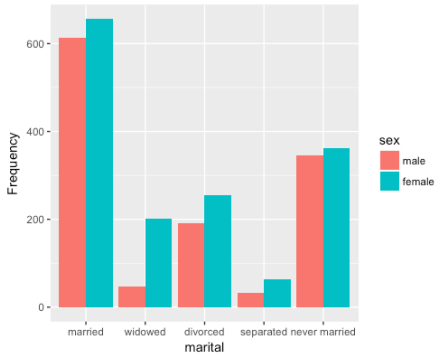
```
# Example 2 – Two Discrete Variables Grouped Bar Chart with Aesthetics
ggplot(data=auto,aes(x=rep78))+
  geom_bar(na.rm=T)+
  facet_wrap(~foreign) +
  labs(x="Car type",y="Frequency", title="Bar Graph Summary of Foreign")
```



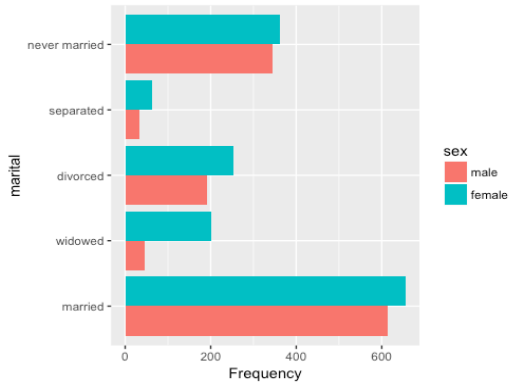

```
# Example 3 – Two Discrete Variables Grouped Bar Chart with Aesthetics
# Y-axis changed to show relative frequencies (%) separately in each group
 ggplot(data=auto,aes(x=rep78))+
    geom_bar(aes(y=..prop..),na.rm=T)+
    labs(y="Percent",title="Repair Record in 1978")+
    facet_wrap(~foreign)+
    geom_text(aes( label = scales::percent(..prop..),y= ..prop.. ),
        stat= "count",
        vjust = -0.2)
```



```
# Example 4 – Two Discrete Variables Grouped Bar Chart with Aesthetics
# 1st variable is x=marital.
# 2nd variable is fill=sex

ggplot(data=descriptive_gss, aes(x=marital,y=..count..,fill=sex))   +
    geom_bar(na.rm=T, position=position_dodge()   +
    labs(y="Frequency")
```




```
# Example 5 – Two Discrete Variables Grouped Bar Chart with Aesthetics:  FLIPPED!
# 1st variable is x=marital.
# 2nd variable is fill=sex

ggplot2::ggplot(data=descriptive_gss, aes(x=marital,y=..count..,fill=sex))+
geom_bar(na.rm=T, position=position_dodge())    +
labs(y="Frequency") +
coord_flip()
```

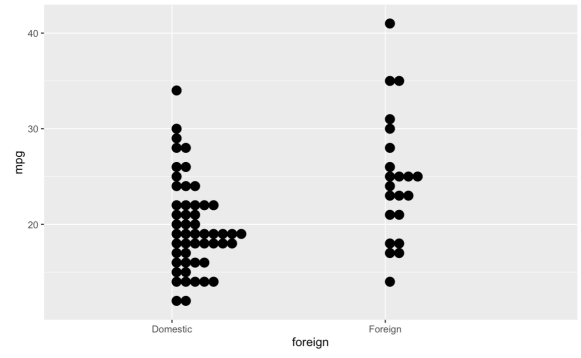


4.2 Continuous, by Group: Side-by-Side Dot Plots

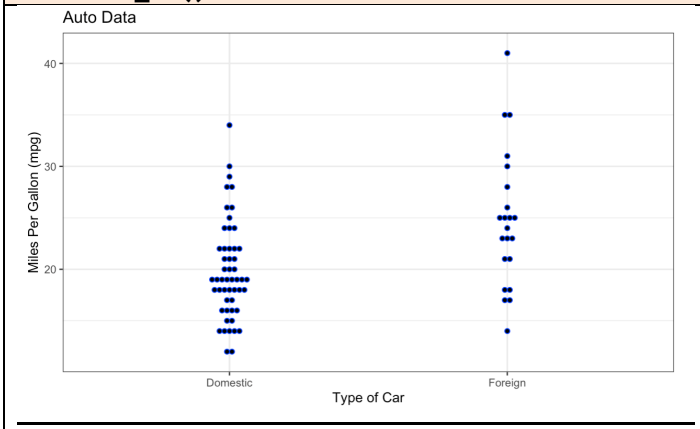
Continuous, by Group: Side-by-Side Dot Plots

```
ggplot(data=dataframe, aes(x=groupvar, y=continuousvar))  +
       geom_dotplot(binaxis="y")
```

```
# Example 1 - Minimal
ggplot(data=auto,aes(x=foreign,y=mpg)) +
 geom_dotplot(binaxis="y")
```




```
# Example 2 – With Aesthetics (hmmmm …. Not sure I like the centering, but you might!)
ggplot(data=auto,aes(x=foreign,y=mpg)) +
   geom_dotplot(binaxis="y", dotsize=0.50, color="blue",
       stackdir="center",  binpositions="all")  +
   xlab("Type of Car")  +
   ylab("Miles Per Gallon (mpg)")  +
   ggtitle("Auto Data")  +
   theme_bw()
```



4.3 Two Continuous: Back-to-Back Stem and Leaf

Note: The following uses the package aplback not ggplot2

```
Two Continuous Variables – Back-to-Back Stem and Leaf

library(aplback)

Two independent groups defined by levels of a grouping variable
a<-subset(dataframe$continuousvar,dataframe$groupvar =="value")
b<-subset(dataframe$ continuousvar, dataframe$groupvar =="value")
aplpack::stem.leaf.backback(a,b)
```

Example

```
library(aplpack)

Domestic<-subset(auto$mpg,auto$foreign=="Domestic")
Foreign<-subset(auto$mpg,auto$foreign=="Foreign")
aplpack::stem.leaf.backback(Domestic,Foreign)
```

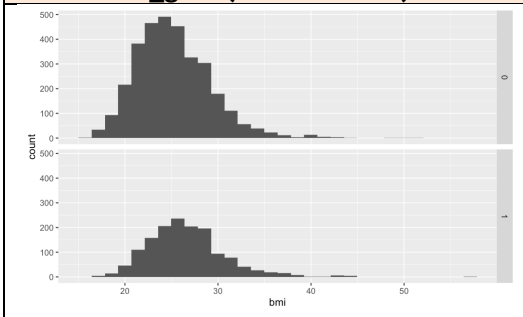
```
+-----------------------------------------------------
  1 | 2: represents 12, leaf unit: 1
               Domestic      Foreign
-------------------------------------------------------
                       | 1* |
    2                22|  t |
    9            5544444|  f |4                 1
   15             776666|  s |77                3
  (15)  999999998888888| 1. |88                5
   22             111000| 2* |11                7
   16              22222|  t |333              10
   11               5444|  f |45555            (5)
    7                 66|  s |6                 7
    5                988| 2. |8                 6
    2                  0| 3* |01                5
                        |  t |
    1                  4|  f |55                3
                        |  s |
                        | 3. |
                        | 4* |1                 1
-------------------------------------------------------
n:                    52      22
-------------------------------------------------------
```

4.4 Continuous, by Group: Side-by-Side Histograms

Continuous, by Group: Side-by-Side Histograms
Separate Panels

```
ggplot(data=framinghamdf, aes(x=bmi)) +
    geom_histogram() +
    facet_grid(chdfate ~ .)
```

```
# Example 1 – Separate Panels:  Minimal
ggplot(data=framinghamdf, aes(x=bmi)) +
    geom_histogram() +
    facet_grid(chdfate ~ .)
```

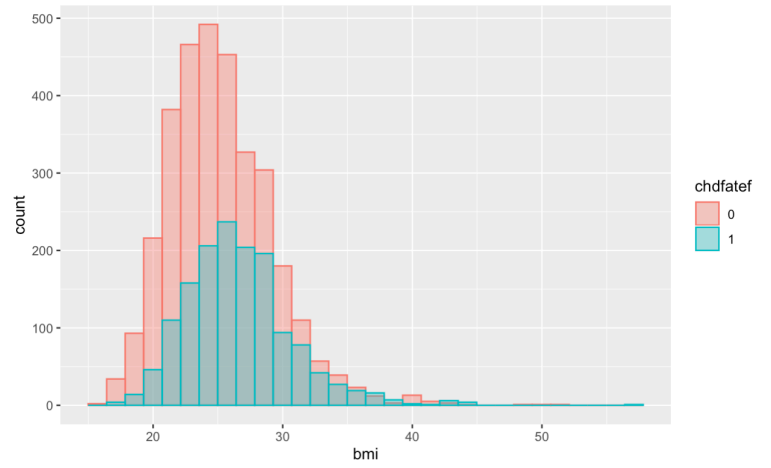



```
# Example 2 – Separate Panels:  With Aesthetics
ggplot(data=framinghamdf, aes(x=bmi)) +
    geom_histogram(binwidth=5, color="blue", fill="grey") +
    facet_grid(chdfate ~ .) +
    scale_color_grey() +scale_fill_grey() +
    ggtitle("Framingham Heart Study Didactic (n=4699):  \nBody Mass Index (kg/m2)") +
    xlab("Body Mass Index (kg/m2)") +
    ylab("Density") +
    theme(axis.title=element_text(size=9),
                plot.title=element_text(size=10)) +
    theme_bw()
```

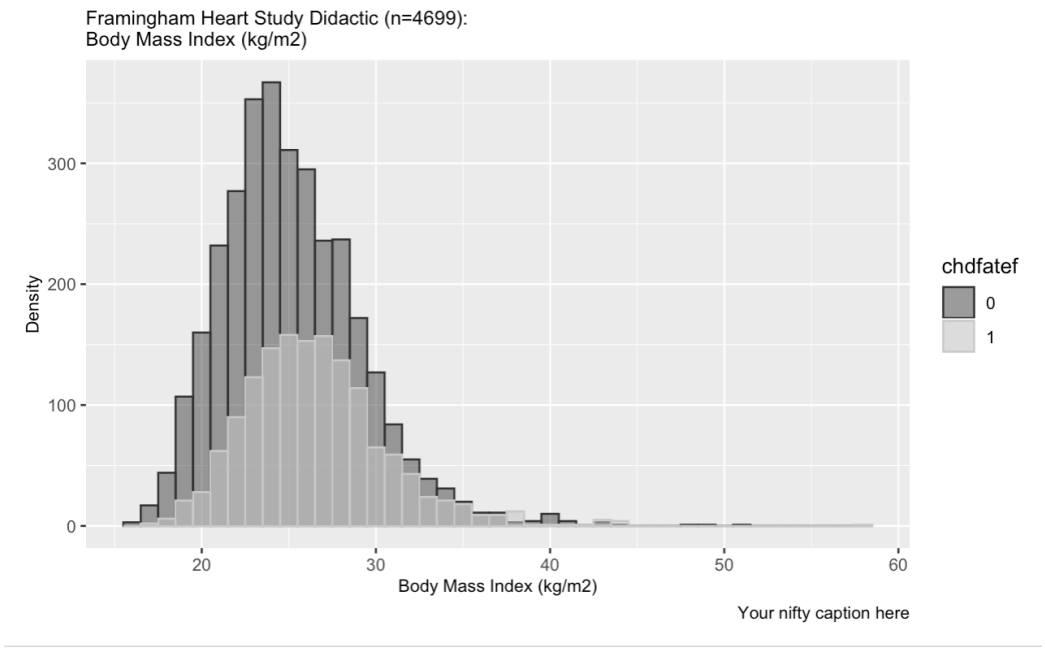
Continuous, by Group: Side-by-Side Histograms
Overlay

```
# Grouping variable must be factor
dataframe$groupvarF <- factor(dataframe$groupvar)

ggplot(data=dataframe, aes(x=continuousvar, fill=groupvarF, color=groupvarF)) +
    geom_histogram(position="identity",alpha=0.4)          # NOTE:  alpha = sets the transparency
```

```
# Example 1 - Overlay:  Minimal
framinghamdf$chdfatef <- factor(framinghamdf$chdfate)
 ggplot(data=framinghamdf, aes(x=bmi, fill=chdfatef, color=chdfatef)) +
    geom_histogram(position="identity",alpha=0.4)
```



```
# Example 2 – Overlay:  With Aesthetics
ggplot(data=framinghamdf, aes(x=bmi)) +
    geom_histogram(binwidth=5, color="blue", fill="grey") +
    facet_grid(chdfate ~ .) +
    scale_color_grey() +scale_fill_grey() +
    ggtitle("Framingham Heart Study Didactic (n=4699):  \nBody Mass Index (kg/m2)") +
    xlab("Body Mass Index (kg/m2)") +
    ylab("Density") +
    theme(axis.title=element_text(size=9),
                plot.title=element_text(size=10)) +
    theme_bw()
```



4.5 Continuous, by Group: Side-by-Side Box and Whisker Plots

Continuous, by Group: Side-by-Side Box and Whisker Plots

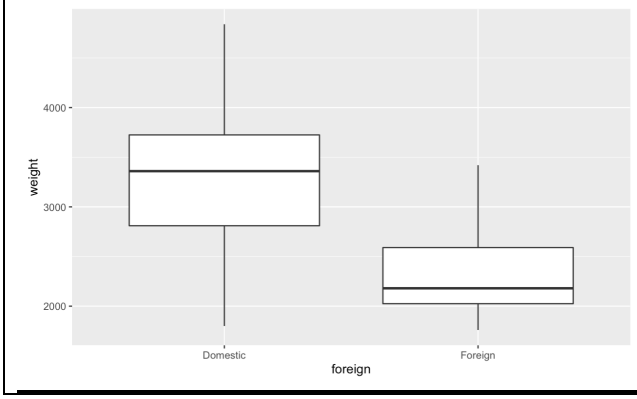
No Color

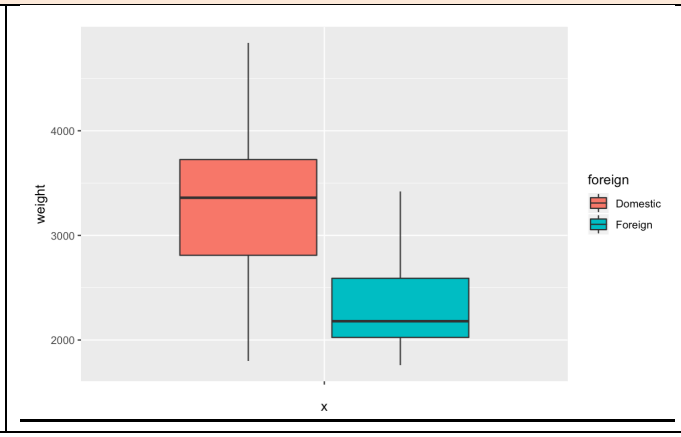
```
ggplot(data= dataframe, aes(x=groupvar,y=continuousvar))  +
  geom_boxplot(na.rm=T)
```

Color

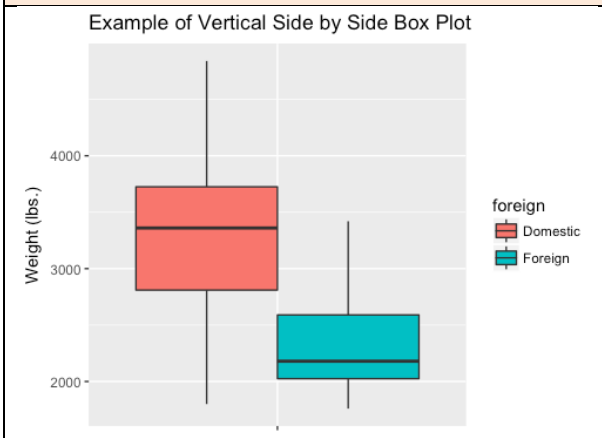
```
ggplot(data=dataframe, aes(x="", y=continuousvar, fill=groupvar))  +
       geom_boxplot(na.rm=T)
```

```
# Example 1 – Minimal
# No Color
ggplot(data=auto, aes(x=foreign,y=weight))  +
  geom_boxplot(na.rm=T)

# Color
ggplot(data=auto, aes(x="",y=weight,fill=foreign))  +
  geom_boxplot(na.rm=T)
```

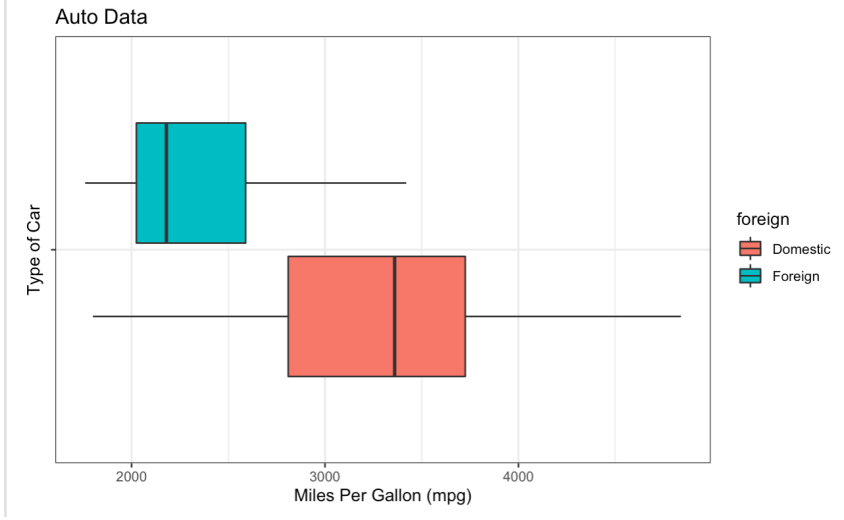




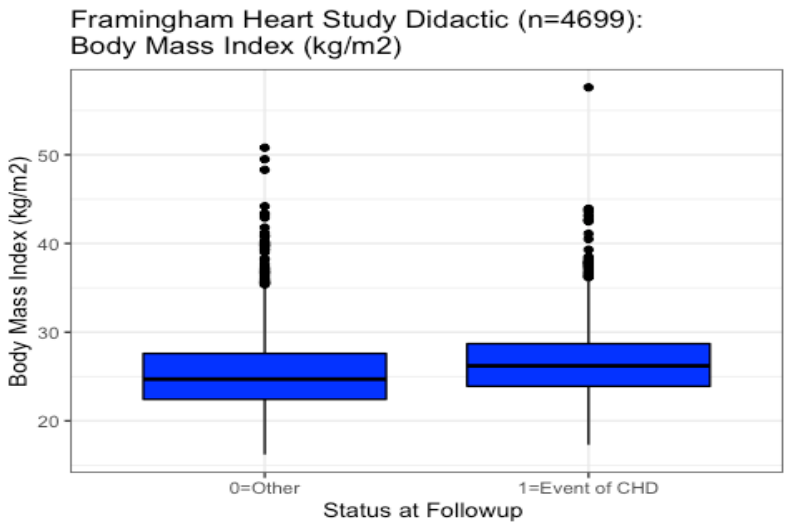
```
# Example 2 – VERTICAL:  With Aesthetics
 ggplot(data=auto, aes(x="",y=weight,fill=foreign))+
  geom_boxplot(na.rm=T)    +
  labs(x="",y="Weight (lbs.)", title="Example of Vertical Side by Side Box Plot")
```




```
# Example 3 – HORIZONTAL: With Aesthetics
ggplot(data=auto, aes(x="",y=weight,fill=foreign))+
  geom_boxplot(na.rm=T) +
   coord_flip() +
   xlab("Type of Car")  +
   ylab("Miles Per Gallon (mpg)")  +
   ggtitle("Auto Data")  +
   theme_bw()
```



```
# Example 4 – VERTICAL:  With colors and fancy titles and axes
ggplot(data=framinghamdf, aes(x=factor(chdfate), y=bmi)) +
    geom_boxplot(color="black", fill="blue") +
    ggtitle("Framingham Heart Study Didactic (n=4699):  \nBody Mass Index (kg/m2)") +
    xlab("Status at Followup ") +
    ylab("Body Mass Index (kg/m2)") +
    theme(legend.position = "none") +
    theme_bw()
```

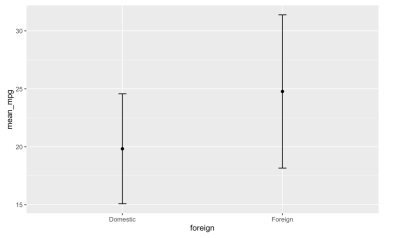


4.6 Continuous, by Group: Mean ± SD (or SE or 95% CI)

```
                Continuous, by Group:  Mean ± SD (or SE or 95% CI)

Preliminaries:  Must obtain summaries first
newdf <- originaldf %>%                      # start with dataframe originaldf THEN
     group_by(groupvar) %>%                  # for each group defined by groupvar THEN
     summarise(                              # obtain the following summaries
     n_yvar = sum(!is.na(yvar)),             # sample size (complete data only) AND
     mean_yvar = mean(yvar, na.rm=TRUE),     # mean (remove missings) AND
     sd_yvar = sd(yvar, na.rm=TRUE),         # standard deviation (remove missings) AND
     se_yvar = sd_yvar/sqrt(n_yvar),         # standard error AND
   ci95_yvar = 1.96*se_yvar)                 # 1.96 * standard error (you may want different)


Plot of Mean ± SD
ggplot(data=newdf, aes(x=groupvar, y=mean_yvar)) +
     geom_errorbar(aes(ymin=mean_yvar - sd_yvar, ymax=mean_yvar + sd_yvar), width=.05)  +
     geom_point()

Plot of Mean ± SE
ggplot(data=newdf, aes(x=groupvar, y=mean_yvar)) +
     geom_errorbar(aes(ymin=mean_yvar - se_yvar, ymax=mean_yvar + se_yvar), width=.05)  +
     geom_point()

Plot of Mean ± 95% CI
ggplot(data=newdf, aes(x=groupvar, y=mean_yvar)) +
     geom_errorbar(aes(ymin=mean_yvar - ci95_yvar, ymax=mean_yvar + ci95_yvar), width=.05)  +
     geom_point()
```

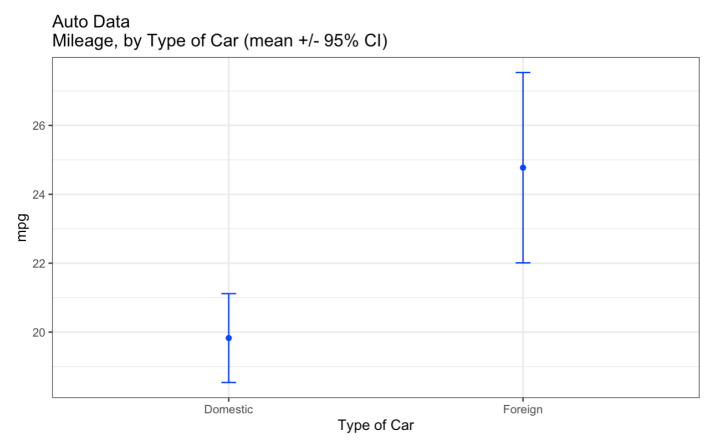
```
# Example 1 – Minimal:  Mean ± SD
```

```
autoplot <- auto %>%                          # dataframe is auto THEN
   group_by(foreign) %>%                      # for each group defined by foreign THEN
   summarise(                                 # obtain the following summaries
        n_mpg = sum(!is.na(mpg)),             # sample size (complete observations only)
        mean_mpg = mean(mpg, na.rm=TRUE),     # mean (remove missings)
        sd_mpg = sd(mpg, na.rm=TRUE),         # standard deviation (remove missings)
        se_mpg = sd_mpg/sqrt(n_mpg),          # standard error
        ci95_mpg = 1.96*se_mpg)               # 1.96 * standard error


ggplot(data=autoplot, aes(x=foreign, y=mean_mpg)) +
    geom_errorbar(aes(ymin=mean_mpg - sd_mpg, ymax=mean_mpg +sd_mpg), width=.05)  +
    geom_point()
```




```
# Example 2 – with Aesthetic:  Mean ± 95% CI
```

```
ggplot(data=autoplot, aes(x=foreign, y=mean_mpg)) +
    geom_errorbar(aes(ymin=mean_mpg - ci95_mpg, ymax=mean_mpg +ci95_mpg), width=.05, color="blue")  +
    geom_point(color="blue") +
    ggtitle("Auto Data \nMileage, by Type of Car (mean +/- 95% CI)") +
    xlab("Type of Car")+
    ylab("mpg") +
    theme_bw()
```

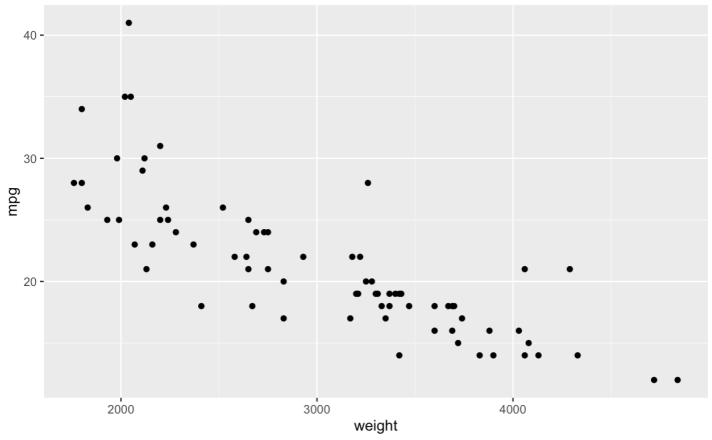


4.7 Two Continuous: XY Plot (Plain and with Overlays)

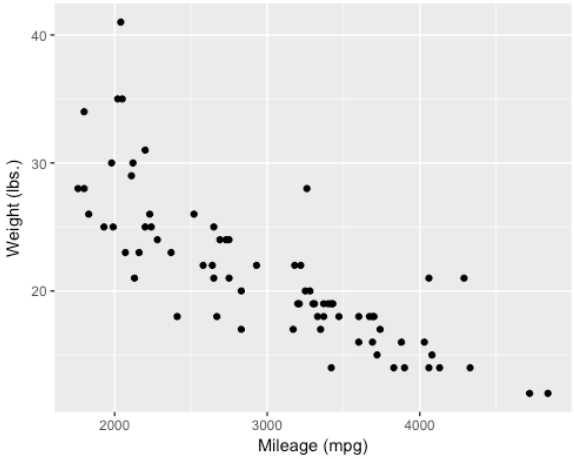
Two Continuous Variables – Plain and with Overlays

```
ggplot(data=dataframe, aes(x=continuousx,y=continuousy))+
    geom_point(na.rm=T)

ggplot2::ggplot(data=auto)+
  geom_point(aes(x=weight,y=mpg),na.rm=T)+
  labs(x="Mileage (mpg)",y="Weight (lbs.)")
```

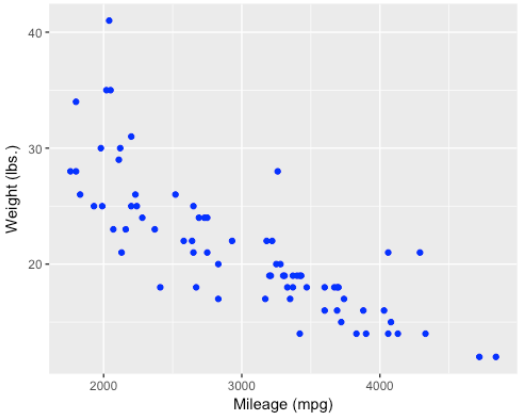
```
# Example 1 – MINIMAL XY Scatterplot
ggplot(data=auto, aes(x=weight,y=mpg))  +
  geom_point(na.rm=T)
```



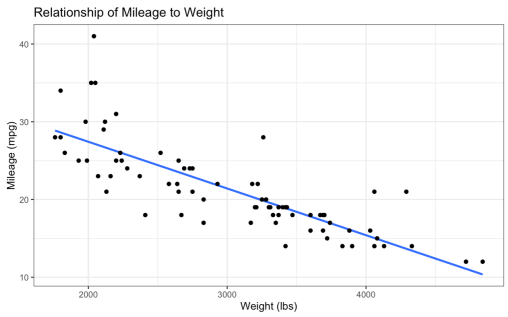
```
# Example 2 – XY Scatterplot with Aesthetic
# Titles and axis labels
ggplot2::ggplot(data=auto, aes(x=weight,y=mpg)) +
  geom_point(na.rm=T) +
  labs(x="Mileage (mpg)",y="Weight (lbs.)")
```



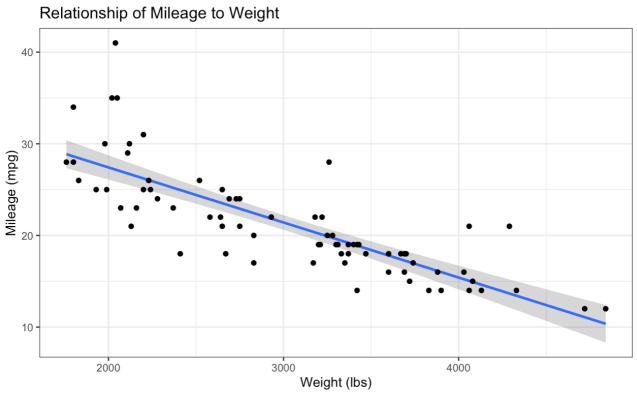

```
# Example 3 – XY Plot with Aesthetic
# Change color of points to blue
ggplot2::ggplot(data=auto, aes(x=weight,y=mpg))  +
  geom_point(na.rm=T,color="blue")  +
  labs(x="Mileage (mpg)",y="Weight (lbs.)")
```



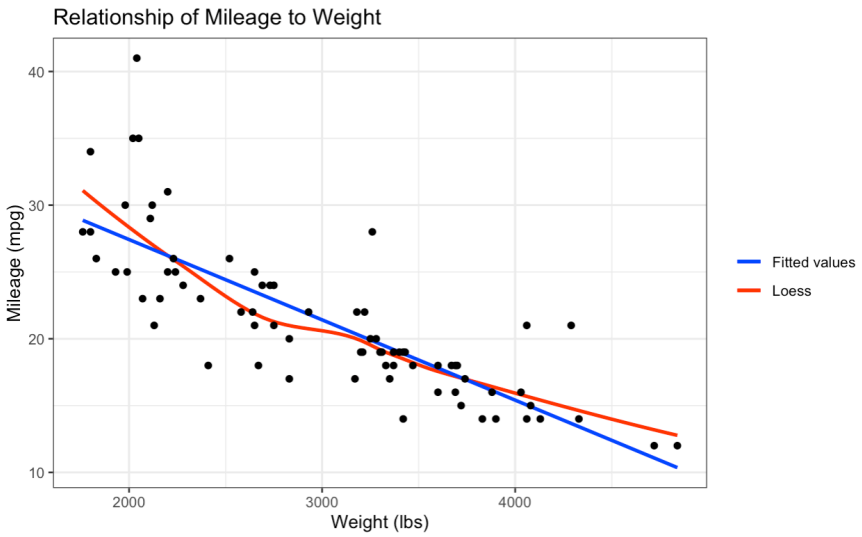
```
# Example 4 – XY Plot with Overlay Fitted Linear Regression
# TIP:  Plot fitted linear regression model first
# Option se=FALSE cancels the default production of 95% CI about the fitted line

ggplot(data=auto, aes(x=weight,y=mpg)) +
 geom_smooth(method=lm, se=FALSE) +
 geom_point() +
 xlab("Weight (lbs)") +
 ylab("Mileage (mpg)") +
 ggtitle("Relationship of Mileage to Weight") +
 theme_bw()
```




```
# Example 5 – XY Plot with Overlay Fitted Linear Regression and 95% Confidence Limits
# TIP:  Plot CI limits and fitted linear regression model first
# Removing option se=FALSE ensures default production of 95% CI

ggplot(data=auto, aes(x=weight,y=mpg)) +
 geom_smooth(method=lm) +
 geom_point() +
 xlab("Weight (lbs)") +
 ylab("Mileage (mpg)") +
 ggtitle("Relationship of Mileage to Weight") +
 theme_bw()
```



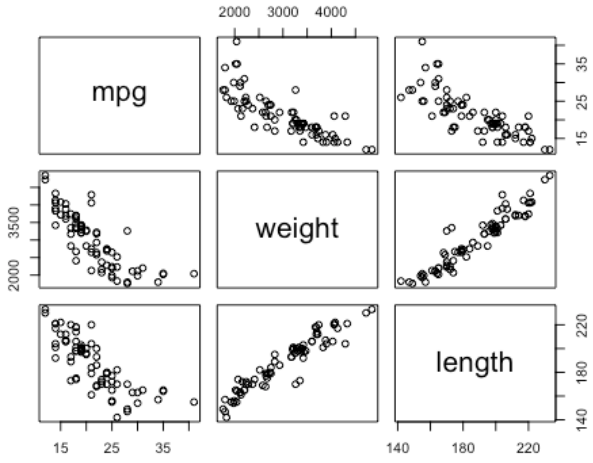
```
# Example 6 – XY Plot with Aesthetic
# LOWESS Smoothing and Linear Fit
ggplot(data=auto, aes(x=weight,y=mpg))  +
   geom_smooth(method = "loess",aes(color="Loess"), se=FALSE)  +
   geom_smooth(method = "lm", aes(color="Fitted values"), se=FALSE)  +
   geom_point()  +
   scale_colour_manual(name="",values=c("blue", "red"))  +
   xlab("Weight (lbs)") +
   ylab("Mileage (mpg)") +
   ggtitle("Relationship of Mileage to Weight") +
   theme_bw()
```



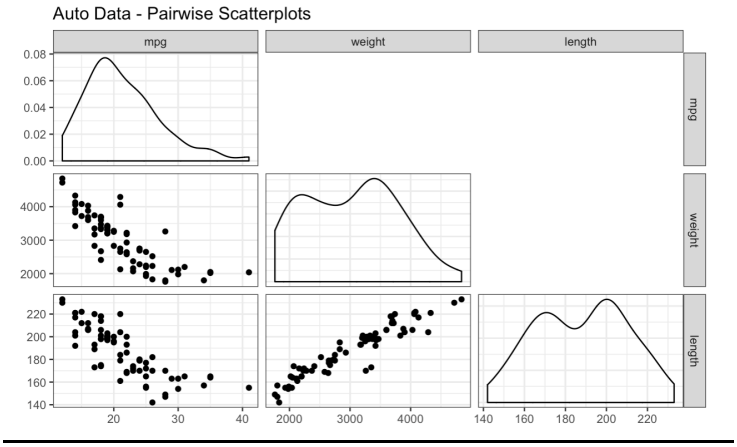
4.8 Many Variables: Matrix Plots

```
                    Many Variables:  Matrix Plots

Using base package – Pairwise scatterplots
myvars <- c("variable1", "variable2", "variable3")
pairs(dataframe[,myvars])

Using GGally package – Pairwise scatterplots & density dlots
library(GGally)
ggpairs(data=dataframe, columns=c("var","var","var"),
       upper="blank",
       title="TITLEHERE")  +
   theme_bw()

Using GGally package – Pairwise scatterplots & correlations
library(GGally)
ggscatmat(data=dataframe, columns=c("var","var","var"))  +
       ggtitle("TITLEHERE")  +
       theme_bw()
```

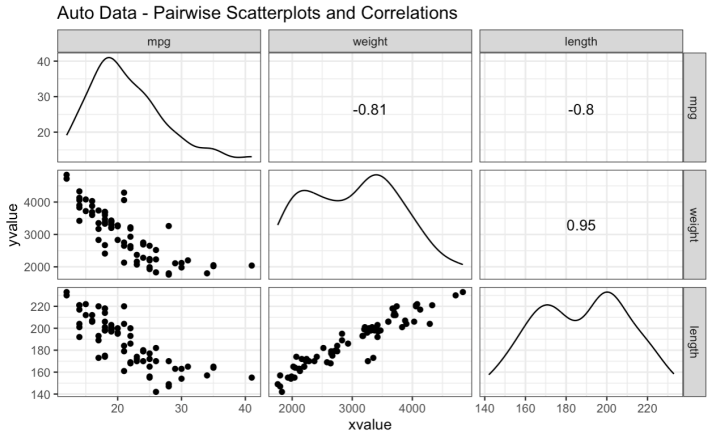
```
# Example 1
myvars <- c("mpg", "weight", "length")
pairs(auto[,myvars])
```



```
# Example 2 – Using package GGally
# Pairwise Scatterplots and Density Plots

library(GGally)
ggpairs(data=auto, columns=c("mpg","weight","length"),
        upper="blank",
        title="Auto Data - Pairwise Scatterplots")  +
   theme_bw()
```




```
# Example 3 – Using package GGally
# Pairwise Scatterplots and Pairwise Correlations

library(GGally)
ggscatmat(data=auto, columns=c("mpg", "weight", "length"))  +
 ggtitle("Auto Data - Pairwise Scatterplots and Correlations")  +
 theme_bw()
```



Appendices

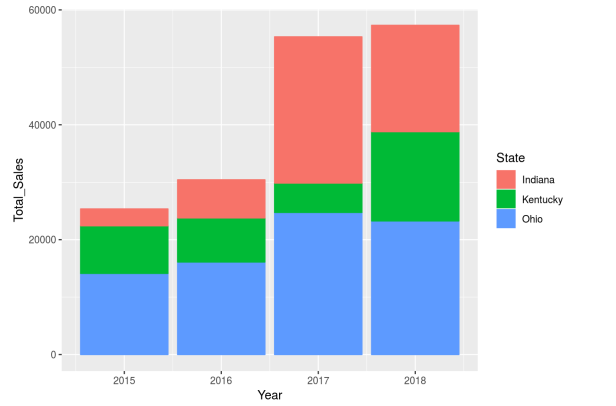
A.1 Choose Your Theme

```
theme_xxx( )
```

Themes available with package `ggplot2`

`theme_gray( )` *this is the default*

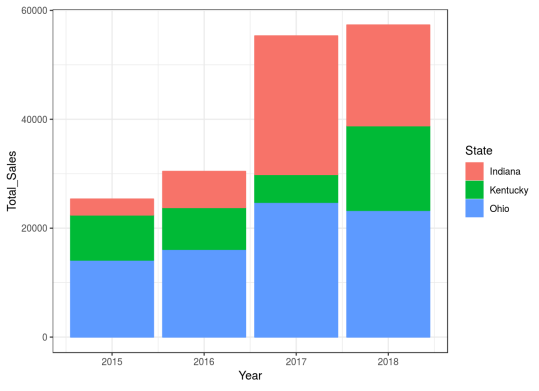
```
plot + geom_col() +
  theme_gray()
```



`theme_bw( )`

theme_bw()

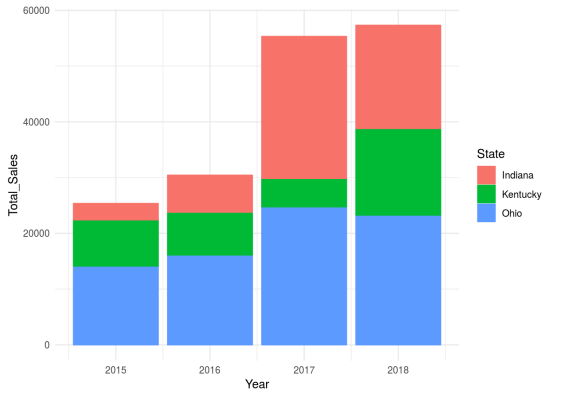
```
# ggplot 2 default themes
# https://ggplot2.tidyverse.org/reference/ggtheme.html
# theme_bw()
plot + geom_col()+
  theme_bw()
```



theme_minimal( )

theme_minimal()

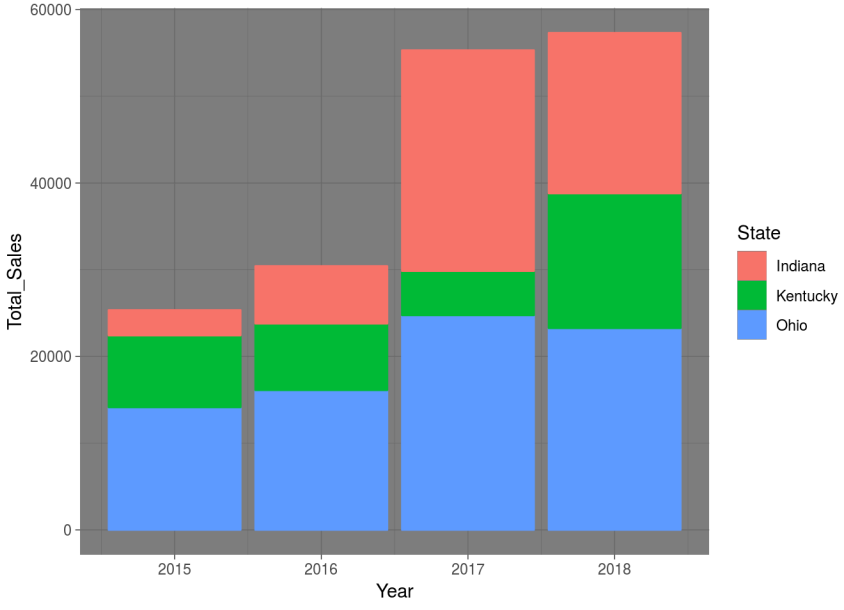
```
# theme_minimal()
plot + geom_col() +
  theme_minimal()
```



theme_dark( )

theme_dark()

```
# theme_dark()
plot + geom_col() +
  theme_dark()
```

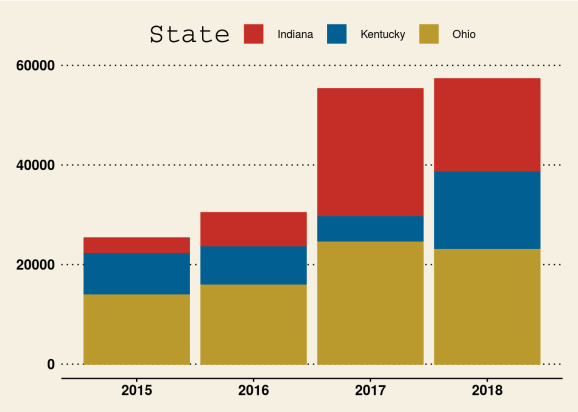


Themes available with package ggthemes

theme_wsj( )

theme_wsj()

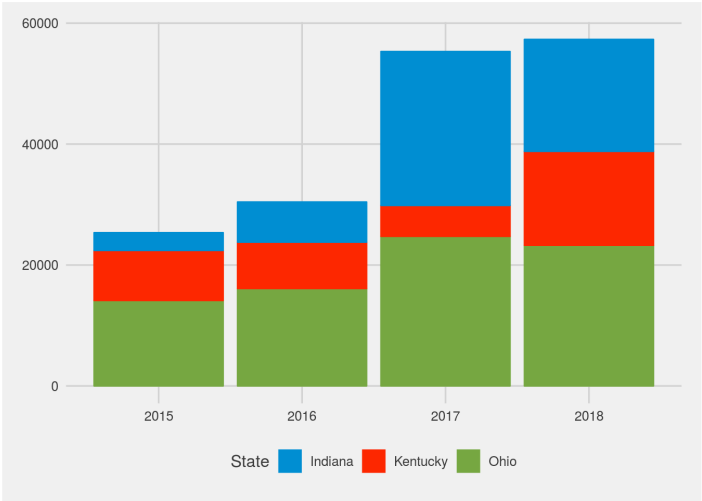
```
# theme_wsj()
library(ggthemes)
plot + geom_col() +
  theme_wsj() +
  scale_fill_manual(values = wsj_pal(palette = "colors6")(3)) +
  scale_color_manual(values = wsj_pal(palette = "colors6")(3))
```



theme_fivethirtyeight( )

theme_fivethirtyeight()

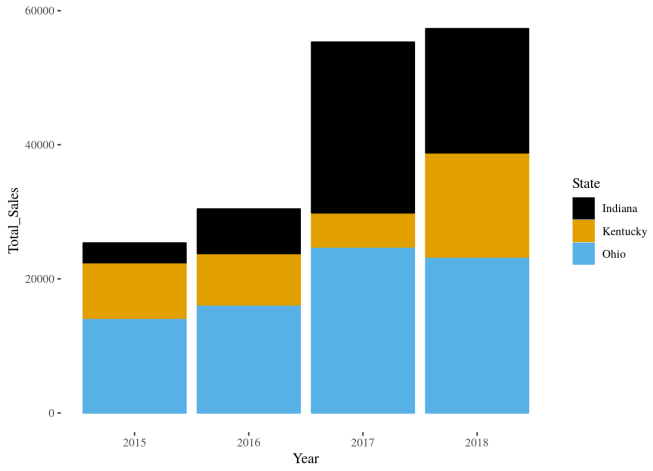
```
plot + geom_col() +
  theme_fivethirtyeight() +
  scale_fill_manual(values = fivethirtyeight_pal()(3)) +
  scale_color_manual(values = fivethirtyeight_pal()(3))
```



theme_tufte( )

theme_tufte()

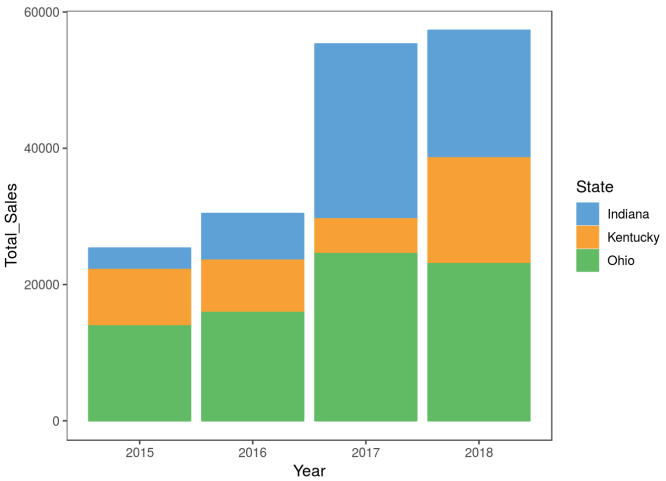
```
plot + geom_col() +
  theme_tufte() +
  scale_fill_manual(values = colorblind_pal()(3)) +
  scale_color_manual(values = colorblind_pal()(3))
```



theme_few( )

theme_few()

```
plot + geom_col() +
  theme_few() +
  scale_fill_manual(values = few_pal()(3)) +
  scale_color_manual(values = few_pal()(3))
```

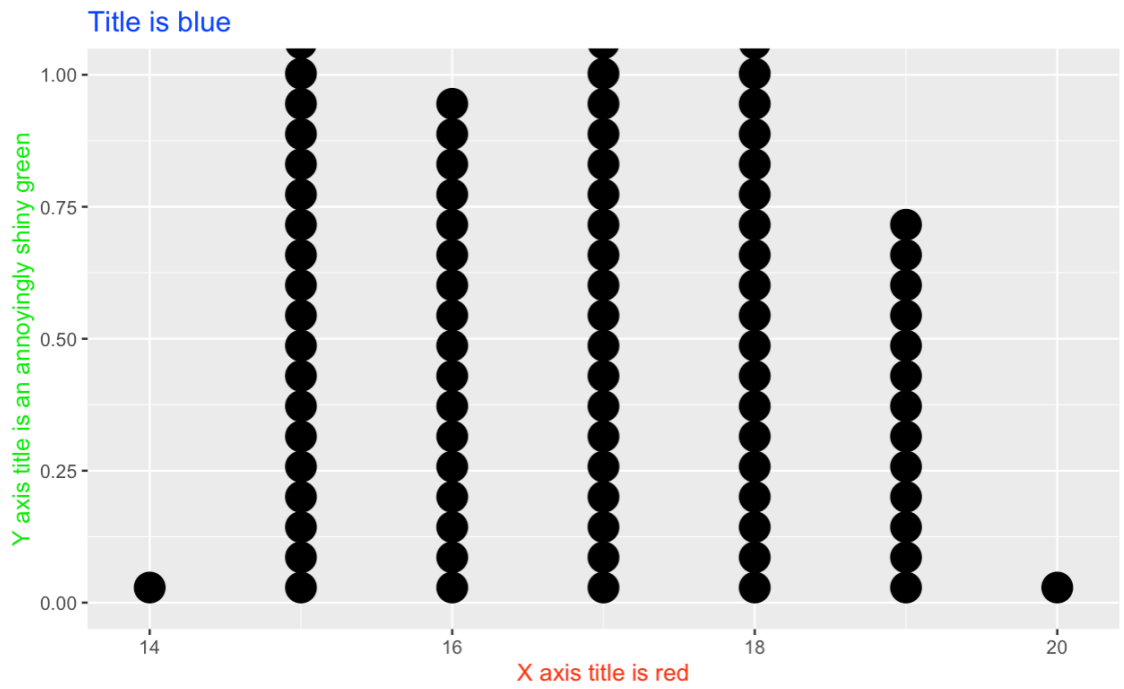


For additional information, see.
http://www.sthda.com/english/wiki/ggplot2-themes-and-background-colors-the-3-elements

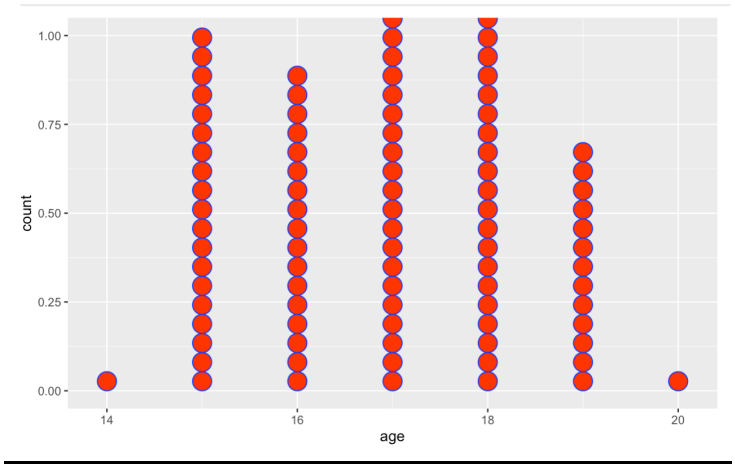
A.2 Choose Your Color

```
color="NAMEOFCOLOR"          # Set color of text and border of plotting character
fill="NAMEOFCOLOR"           # Set color of interior of plotting character
```

```
# Example – Set colors of title, X-axis title, and Y-axis title

ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot() +
    ggtitle("Title is blue") +
    xlab("X axis title is red") +
    ylab("Y axis title is an annoyingly shiny green") +
    theme(plot.title = element_text(color="blue"),        # Title color
          axis.title.x=element_text(color="red"),         # X-axis title color
          axis.title.y=element_text(color="green"))       # Y-axis title color
```



```
# Example – Set border (color=) and interior (fill=) of plotting character

ggplot(data=relatenew100, aes(x = age)) +
    geom_dotplot(color="blue", fill="red")     # Border is color="XXX", fill is fill="XXX"
```



The following colors work just fine:

"red", "blue", "green", "gray"

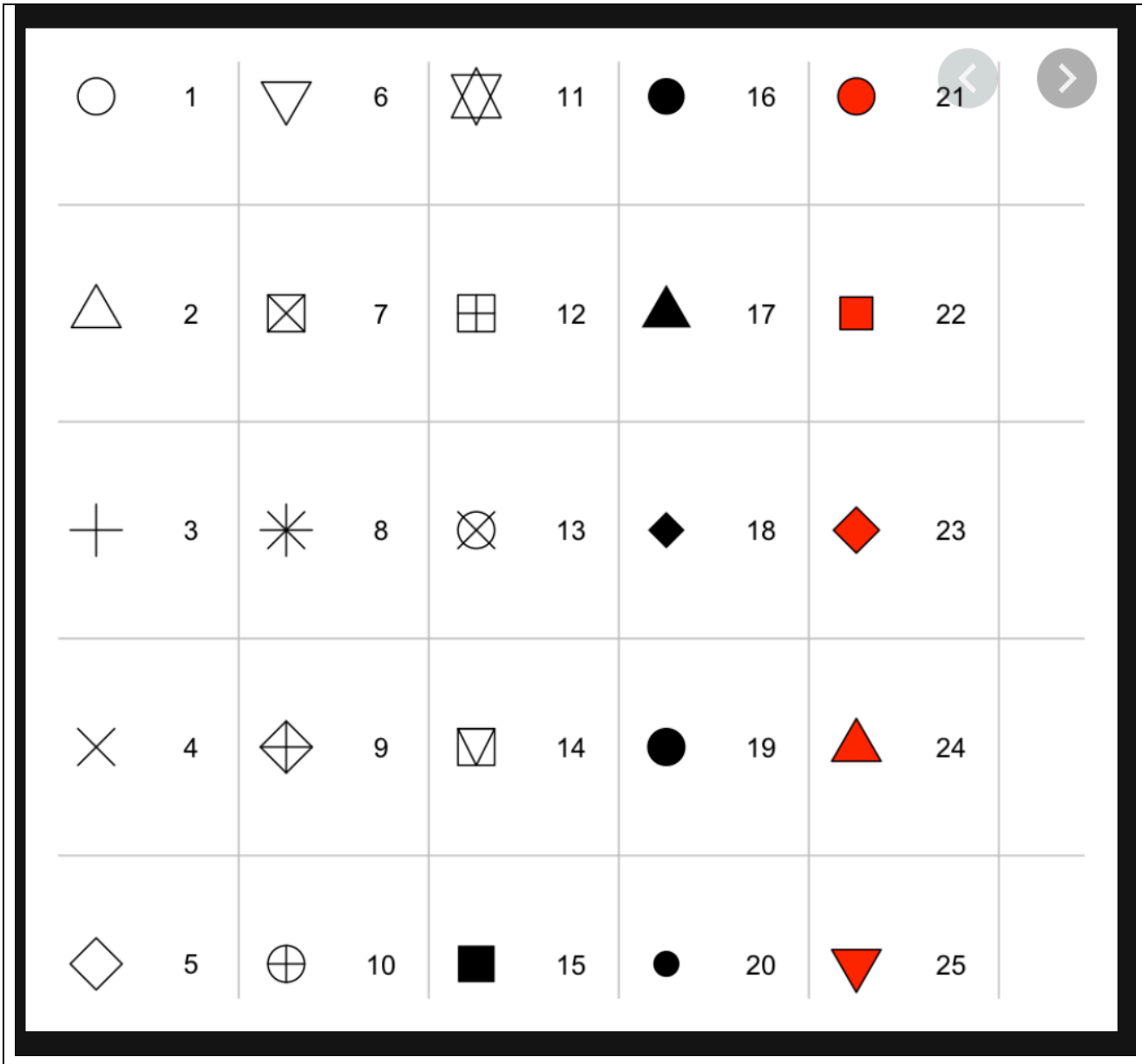
For more detail, see.
https://www.r-graph-gallery.com/ggplot2-color.html

Want more? Here are 8 pages of colors!
http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Good to know: R provides color palettes for color-blind
https://riptutorial.com/r/example/28354/colorblind-friendly-palettes

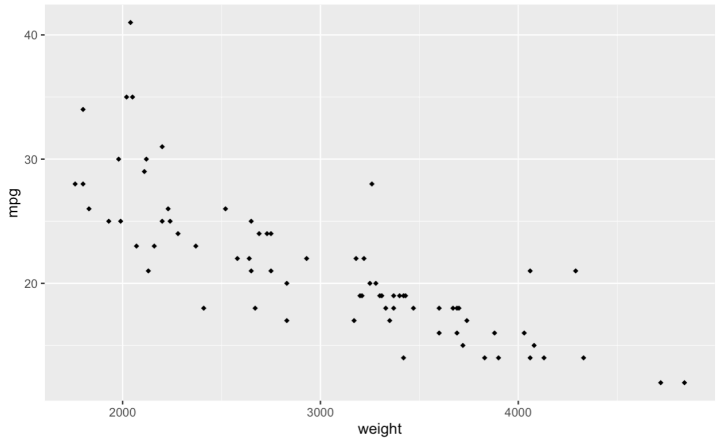
A.3 Choose Your Plotting Character

```
shape=###                                   # Note: No quotes around a number
```

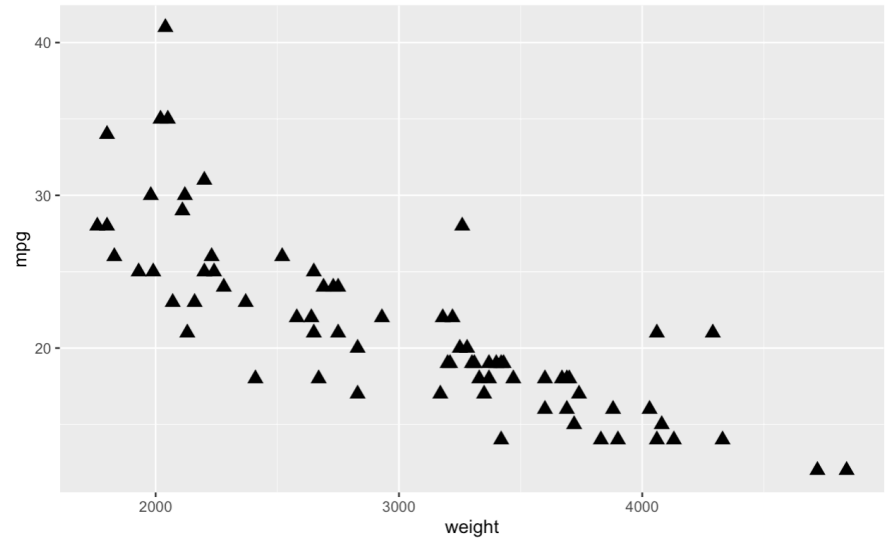


Source: jofrhwld.github.io

```
# Example 1 – Set shape of plotting character

ggplot2::ggplot(data=auto, aes(x=weight,y=mpg))  +
     geom_point(na.rm=T,shape=18)                          # shape=18 produces diamond
```

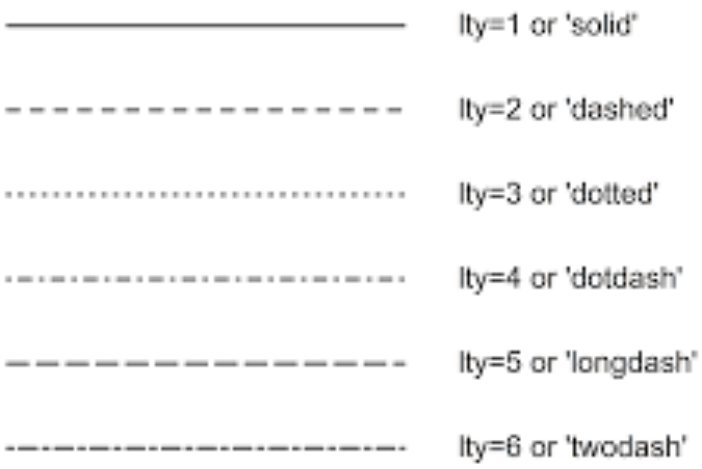



```
# Example 2 – Set shape of plotting character and set size of plotting character

ggplot2::ggplot(data=auto, aes(x=weight,y=mpg))  +
     geom_point(na.rm=T,shape=17,size=3)          # shape=17 produces triangle, size changed
```

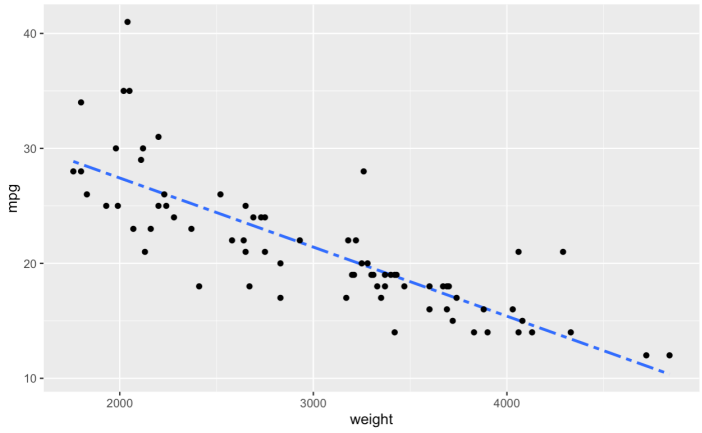


A.4 Choose Your Line Type

```
lty=###                    # Note: No quotes around number

lty="NAME"                 # Note: Quotes if you supply name
```




```
# Example 1 – Specifying twodash line with lty = ###  NO QUOTES

ggplot(data=auto, aes(x=weight,y=mpg)) +
   geom_smooth(method=lm, se=FALSE, lty=6) +     # se=FALSE gets rid of confidence interval
   geom_point()
```



```
# Example 2 – Specifying twodash line with lty = "twodash"  in QUOTES

ggplot(data=auto, aes(x=weight,y=mpg)) +
   geom_smooth(method=lm, se=FALSE, lty="twodash") +  # se=FALSE gets rid of conf interval
   geom_point()
```