GGPLOT2 Intermediate

Emily Cheng

Aug 18, 2022

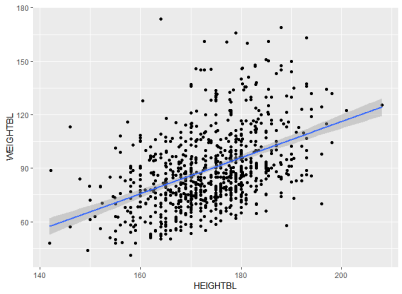
GGPLOT2 Intermediate

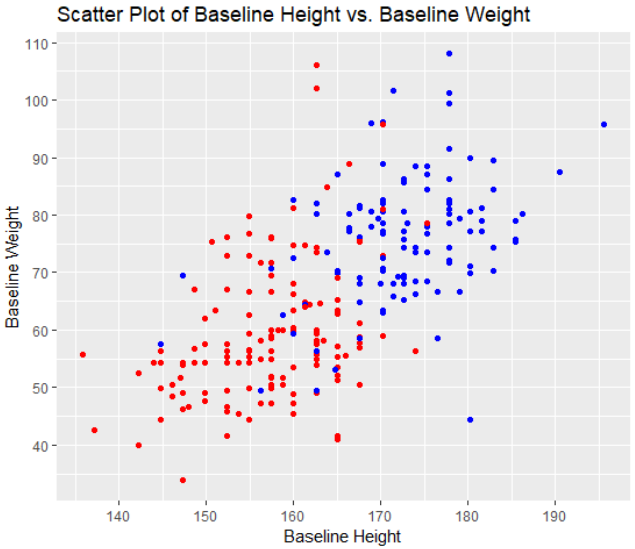
1. Introduction
2. Graphical Layers
3. Building a Basic Plot
4. Aesthetic Mapping
5. Modifying Aesthetics
6. Fixed Attributes
7. Geometries
8. Themes
9. Coordinates, Statistics and Facets
10. Summary

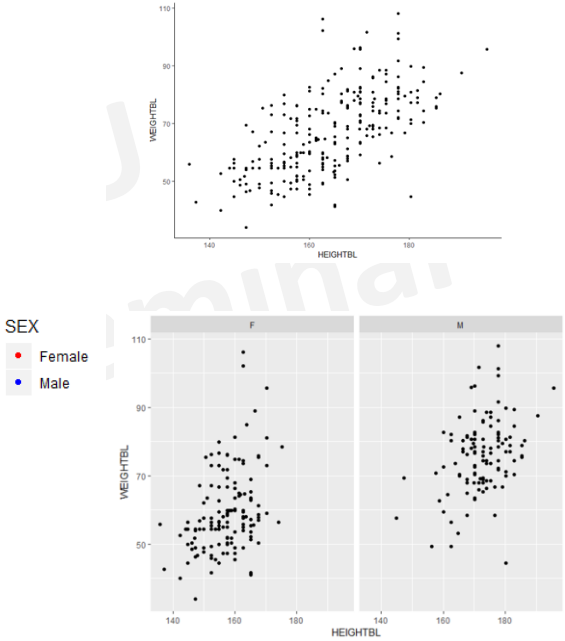
Case 1: Line plot
Case 2: Bar chart
Case 3: Boxplot

1. Introduction

In this lesson, we are going to learn how to create figures using the ggplot2 package, which is part of the tidyverse.



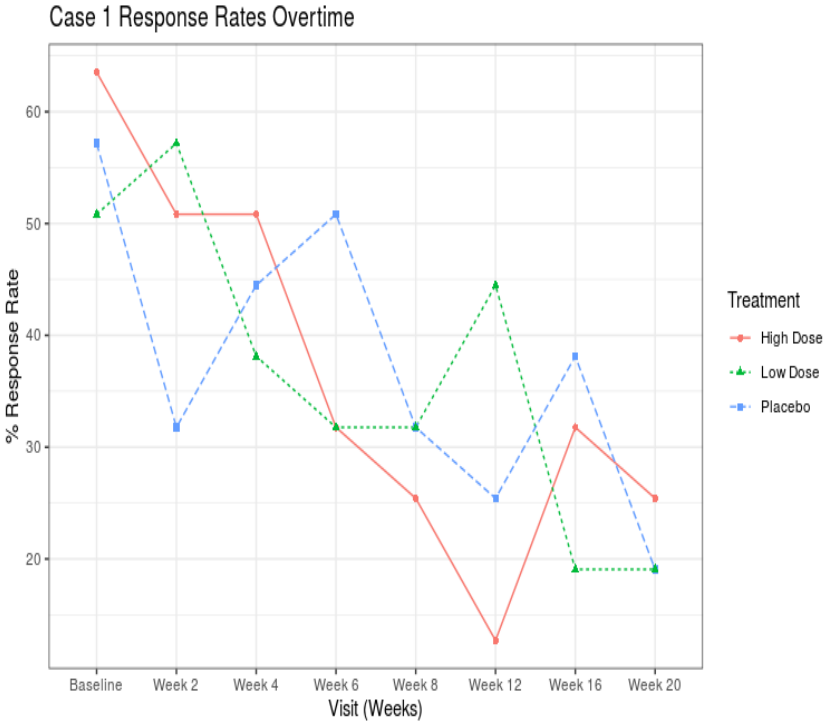


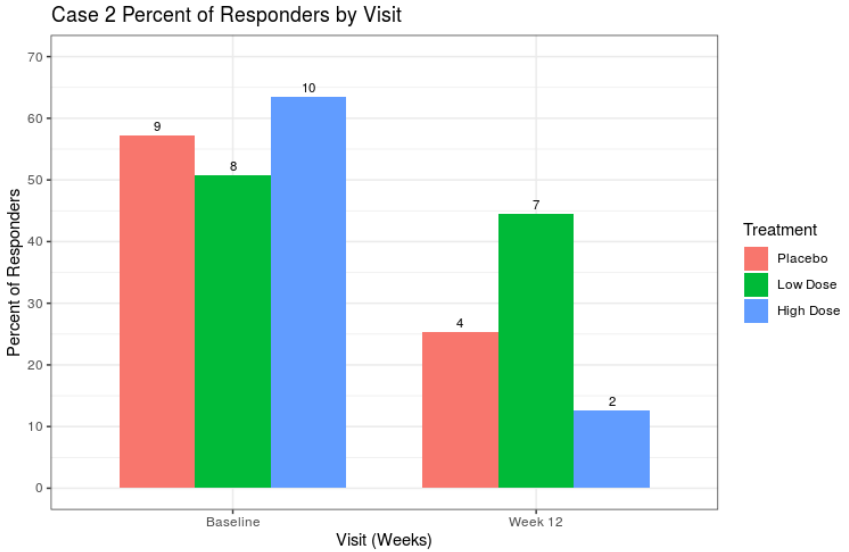


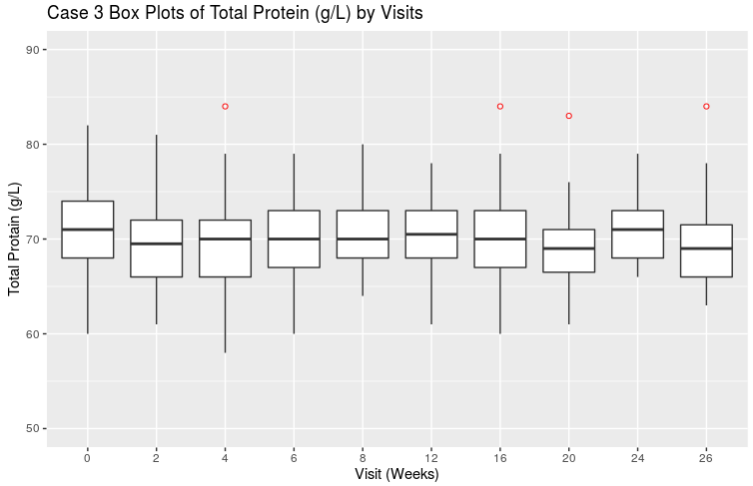
Throughout this chapter, we will use ggplot2 along with dplyr for some simple data manipulation and haven to read in SAS data.

```
library(ggplot2)
library(dplyr)
library(haven)
```

After this session, you will be able to create several common figures...







2. Graphical Layers

With ggplot2, we begin a plot with a call to the function ggplot(). Then we can build a graph upon that coordinate system, layer by layer.

The following seven elements, or layers, constitute what is referred to as “the layered grammar of graphics”.

Graphical Elements	
Data	The data frame you want to plot
Aesthetics	The scales onto which we wish to map our data (e.g., x, y).
Geometries	The visual elements used in our data (i.e., how the plot will look)
Coordinates	The space on which data will be plotted
Statistics	Representations of our data to aid understanding
Facets	Plotting small multiples
Themes	All non-data ink

Focus: first three layers (required by ggplot2)

3. Building a Basic Plot

Let's say we want to visually inspect the relationship between baseline height and weight. A scatterplot would be a reasonable way to do this:

1. Start with call to ggplot:

```
ggplot()
```

2. Add data layer:

```
ggplot(data = adsl)
```

USUBJID	TRT01P	AGE	RACE	SEX	BMIBL	HEIGHTBL	WEIGHTBL
01-701-1015	Placebo	63	WHITE	F	25.1	147.3	54.4
01-701-1023	Placebo	64	WHITE	M	30.4	162.6	80.3
01-701-1028	Xanomeline High Dose	71	WHITE	M	31.4	177.8	99.3
01-701-1033	Xanomeline Low Dose	74	WHITE	M	28.8	175.3	88.5
01-701-1034	Xanomeline High Dose	77	WHITE	F	26.1	154.9	62.6
…							

3. Building a Basic Plot

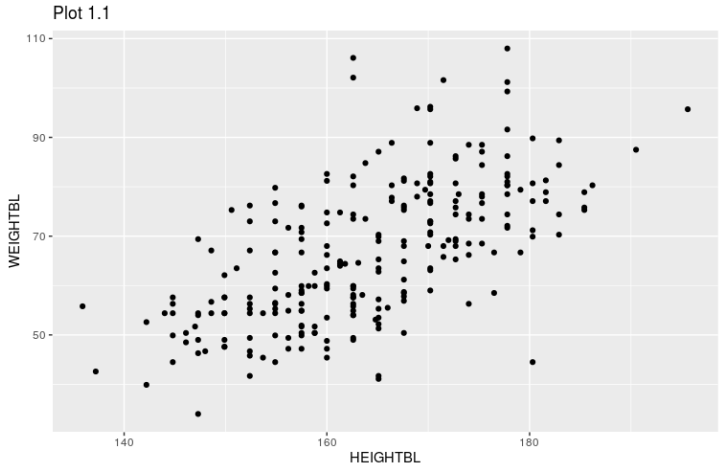
3. Add your aesthetic mappings. Let's map X to baseline height and Y to baseline weight:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL))
```

USUBJID	TRT01P	AGE	RACE	SEX	BMIBL	HEIGHTBL	WEIGHTBL
01-701-1015	Placebo	63	WHITE	F	25.1	147.3	54.4
01-701-1023	Placebo	64	WHITE	M	30.4	162.6	80.3
01-701-1028	Xanomeline High Dose	71	WHITE	M	31.4	177.8	99.3
…							

4. Finally, add a geometry layer. Here we are adding a geom_point (i.e., scatterplot) layer:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point()
```

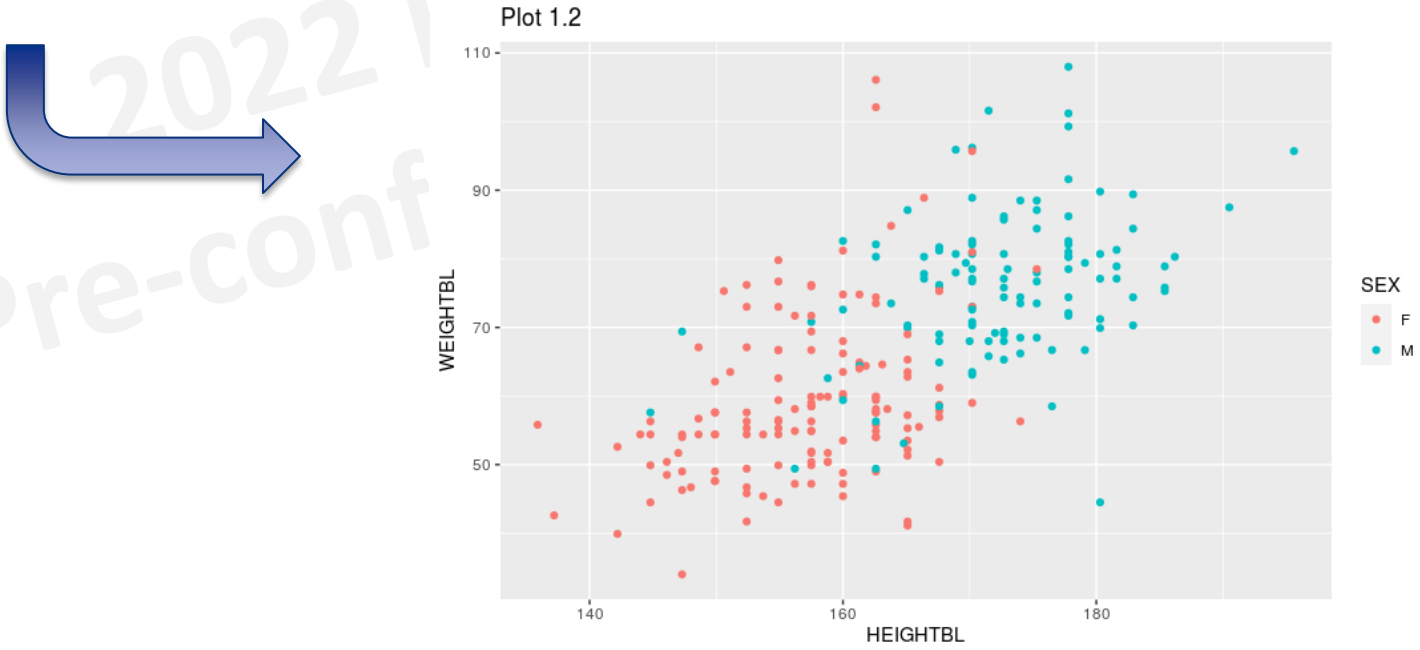


4. Aesthetic Mapping

We can use some other aesthetics, beyond just X and Y, to make the plots more informative (and less dull).

Let's see if the plot is more informative by mapping color to gender:

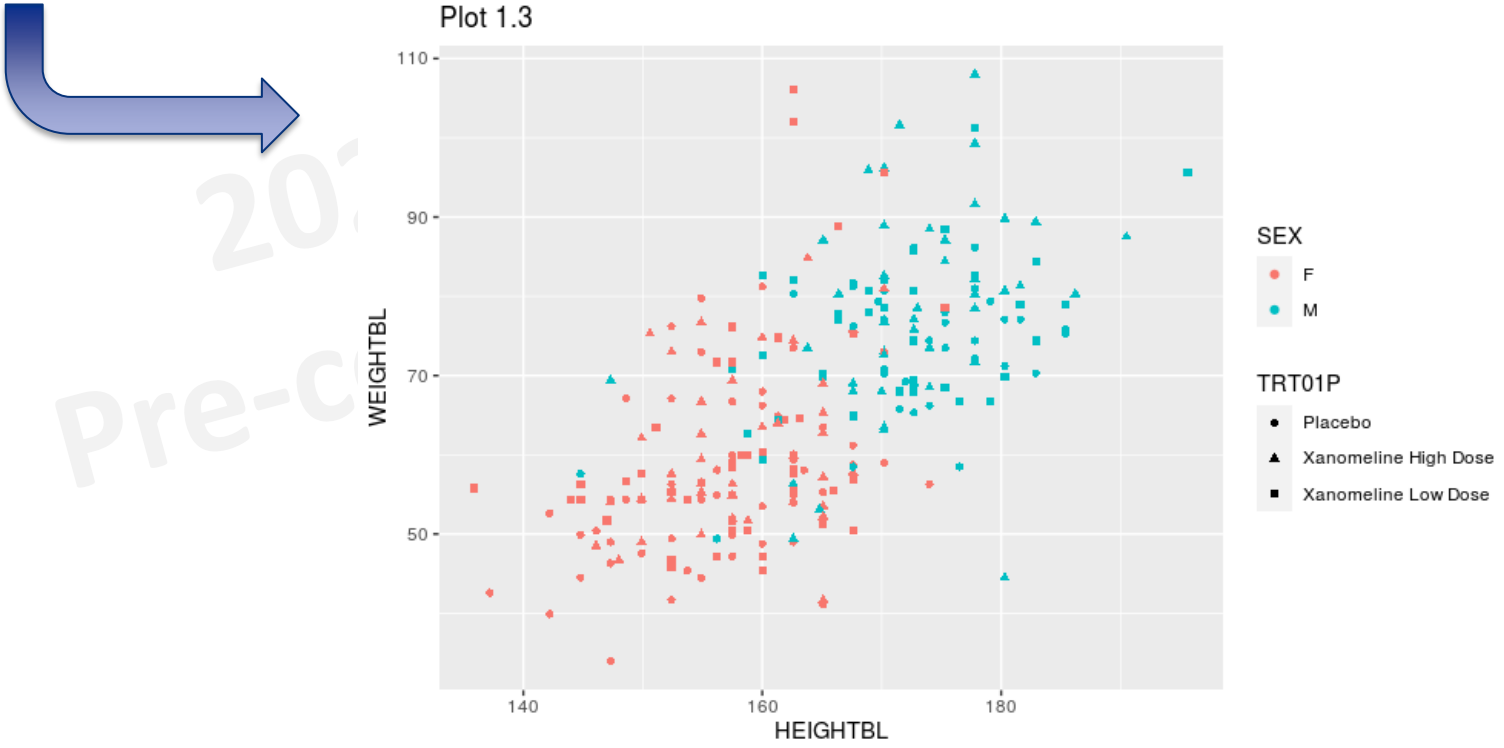
```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL, color = SEX)) +
  geom_point()
```



4. Aesthetic Mapping

How about adding treatment group? Let's map shape to that variable:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL, color=SEX, shape = TRT01P)) +
  geom_point()
```



4. Aesthetic Mapping

To map any aesthetic to a variable, simply associate the name of the aesthetic to the variable inside aes(). ggplot2 will automatically assign a unique level of the aesthetic to each unique value of the variable. This is known as scaling. ggplot2 will also, by default, add a legend that explains the mapping.

Typical Aesthetic Mappings include:

Aesthetic	Description
x	X axis position
y	Y axis position
fill	Fill color
color	Color of points; outlines of other geoms
size	Area, radius of points, thickness of lines
alpha	Transparency
linetype	Line dash pattern
labels	Text on a plot or axis
shape	Shape

5. Modifying Aesthetics

Once you've done your aesthetic mappings, ggplot2 will automatically add scales for you behind the scenes (the highlighted text is running in the background). We can also explicitly write out these functions.

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point() +
  scale_x_continuous() +
  scale_y_continuous ()
```

Notice the general naming scheme for the scales:

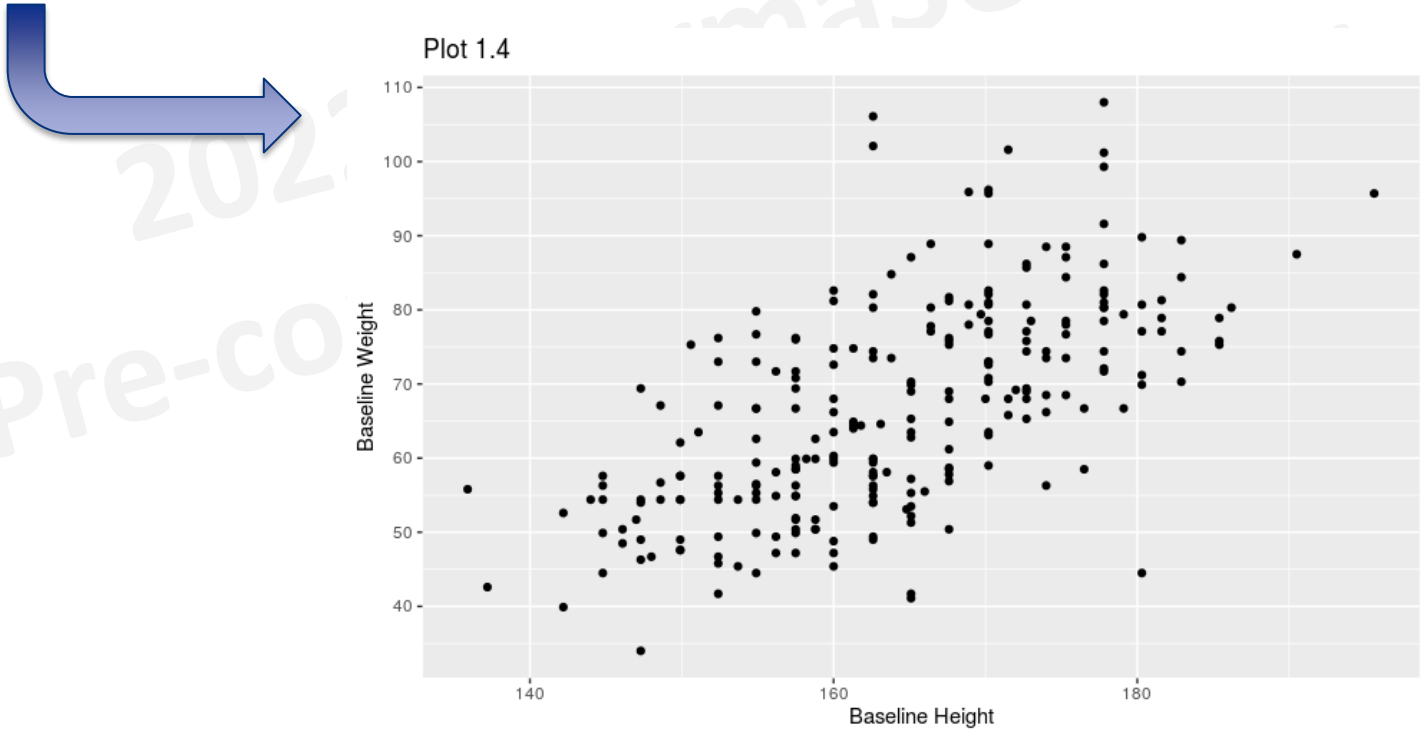
- First, the word "scale",
- then "_",
- then the name of the aesthetic,
- then "_" and then the name of the scale (i.e., continuous, discrete, datetime, date)

These default scales will generally provide you with a good visual representation of the data. If however, you wish to override these defaults, you have the ability to do so…

5. Modifying Aesthetics

We can update the Axes:

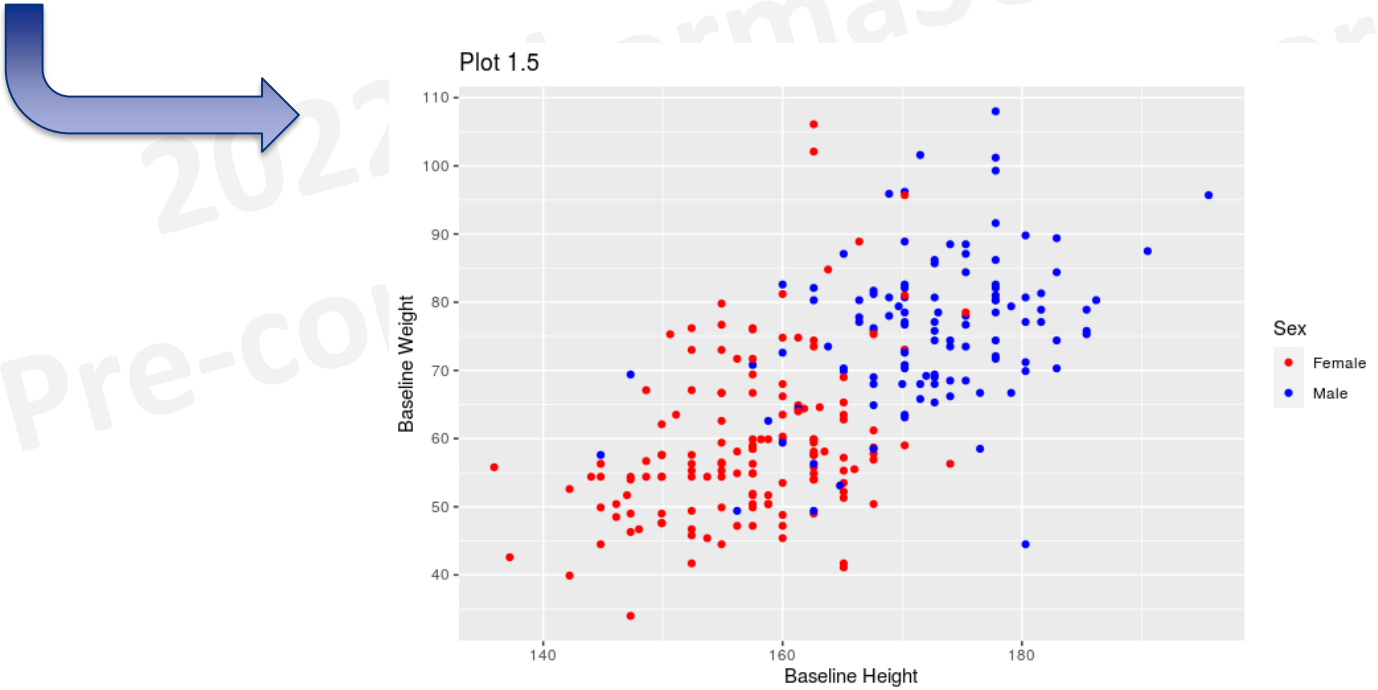
```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point() +
  scale_x_continuous(name = "Baseline Height", breaks = seq(120, 200, by = 20)) +
  scale_y_continuous(name = "Baseline Weight", breaks = seq(30, 120, by = 10))
```



5. Modifying Aesthetics

If we want to look at color, we have control over that as well:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL, color = SEX)) +
  geom_point() +
  scale_x_continuous(name = "Baseline Height", breaks = seq(120, 200, by = 20)) +
  scale_y_continuous(name = "Baseline Weight", breaks = seq(30, 120, by = 10)) +
  scale_color_manual(name = "Sex", values=c("red","blue"), labels = c("Female","Male"))
```

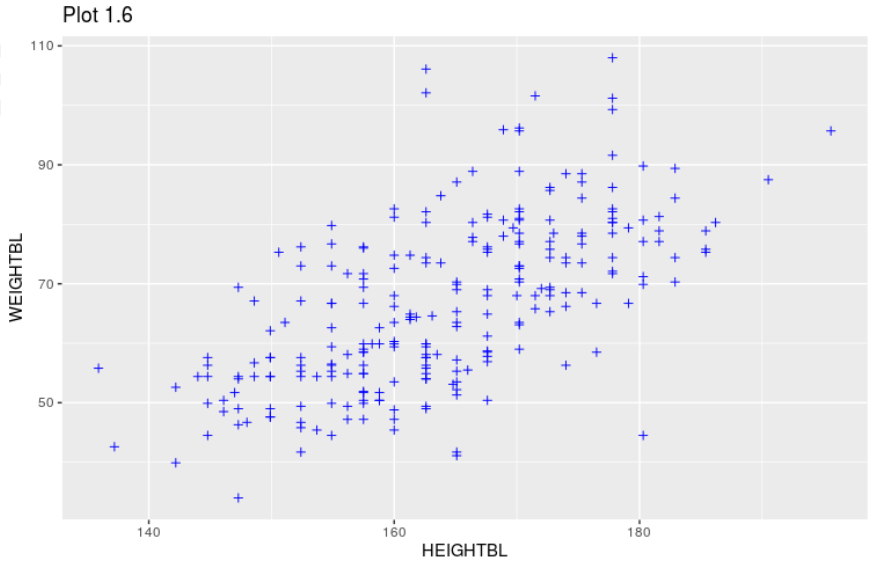


6. Fixed Attributes

Aesthetic mappings alter certain visual characteristics based upon the value of a variable. For example, we previously mapped color onto the values of SEX and shape onto TRT01P.

What if we want to make all of our points blue crosses? Here, we're not mapping the attributes to the value of a given variable; we are simply "fixing" the shape attribute to a 'plus' (for the cross shape) and the color attribute to 'blue'.

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point(shape = 'plus', color = 'blue')
```



Note: The list of available Fixed Attributes is the same as the aesthetic mappings listed in the earlier slide.

6. Fixed Attributes

A couple of notes regarding fixed attributes vs aesthetic mappings:

- Generally, the aesthetic mappings are defined in the ggplot function, and then inherited to the geom layer(s).
- Generally, fixed attributes are defined in the geom layer(s).
- There are situations where you might want to define your aesthetic mappings in the geom layer(s), but that is a topic for another time.

- As always, you can use the R-Studio help for more information on the aesthetics and corresponding scales.

7. Geometries

There are many geometries available to you within ggplot2, each with its own arguments, and aesthetics/attributes. For now, let's take a closer look at geom_point:

- geom_point() includes the following arguments:

```
geom_point(data = NULL,
           mapping = NULL,
           inherit.aes = TRUE,
           show.legend = NA,
           na.rm = FALSE,
           stat = "identity",
           position = "identity",
           ...)
```

Can generally be ignored (i.e., use default values) for most plots.

- Controls any statistical transformation of data.
- Position adjustment.
- Add any valid fixed attributes (geom_point() understands the following aesthetics/attributes: x, y, alpha, colour, fill, group, shape, size, stroke) .

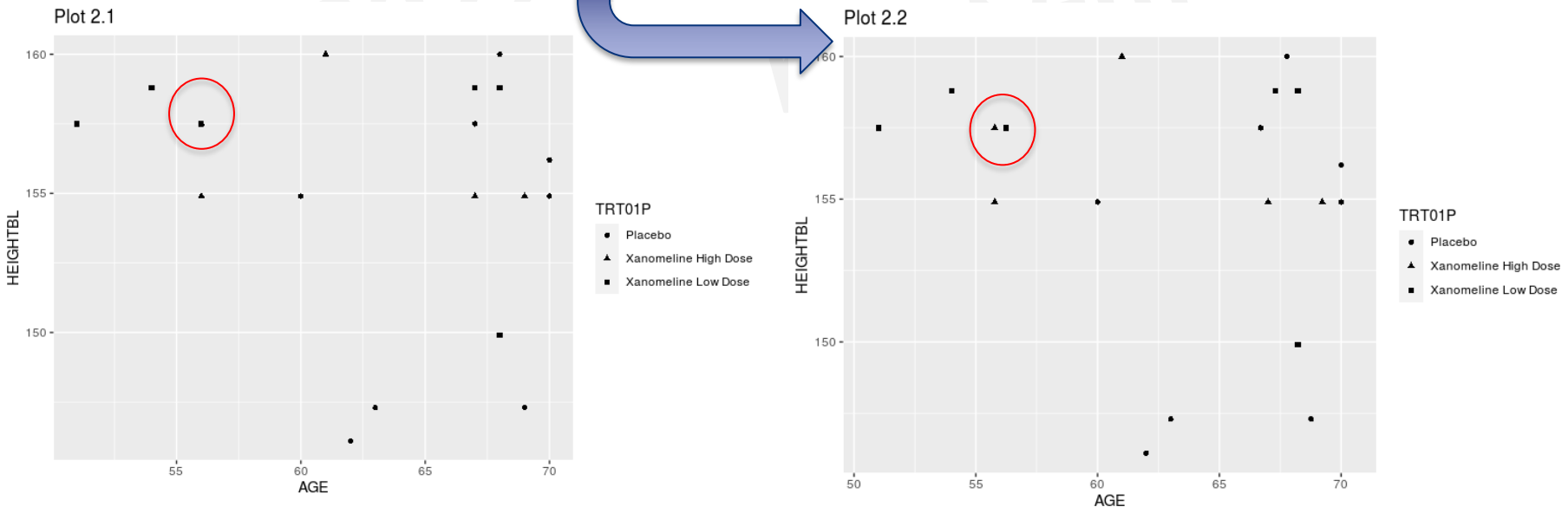
Let's take a quick look at the position parameter on the next slide...

7. Geometries

In the example below, we are using the position parameter to deal with overplotting (i.e., overlapping values of age and height):

```
adsl2 <- adsl %>%
         filter(AGE <= 70 & HEIGHTBL <= 160)

ggplot(data = adsl2, aes(x = AGE, y = HEIGHTBL, shape = TRT01P)) +
  geom_point(position = position_dodge(width = .9))
```

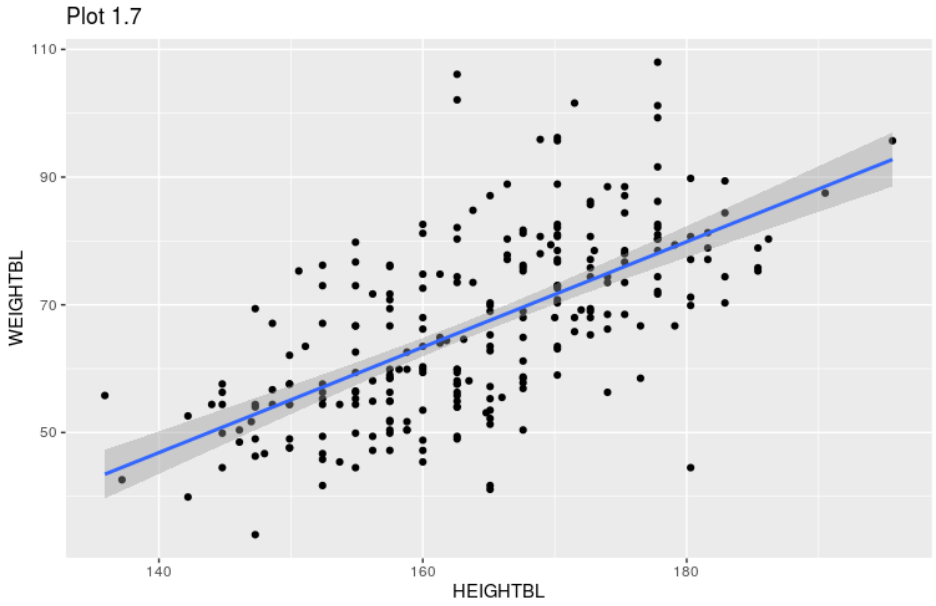


7. Geometries

One final note on geometries… we can add/create more complex graphs by simply layering additional geometries onto our plot.

In the plot below, we've layered a smoothing line to help interpret the trend of the data. Note that the aesthetic mappings to X and Y are inherited to both geoms.

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point() +
  geom_smooth(method=lm)
```



8. Themes

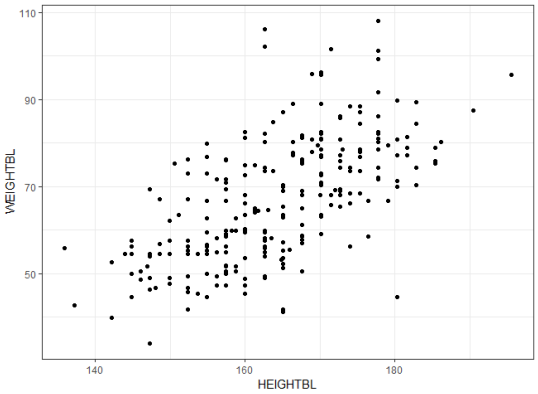
Next we are going to talk about themes, which encompass all non-data ink.

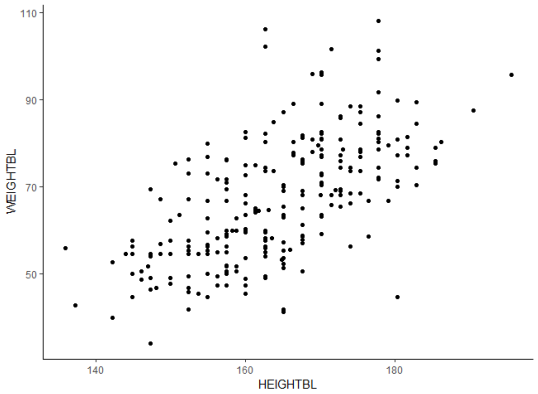
Graphical Elements	
Data	The data frame you want to plot
Aesthetics	The scales onto which we wish to map our data.
Geometries	The visual elements used in our data (i.e., how the plot will look)
Coordinates	The space on which data will be plotted
Statistics	Representations of our data to aid understanding
Facets	Plotting small multiples
Themes	All non-data ink

8. Themes

Throughout this lesson, the plots we have seen have been using the “theme_grey()” theme (default). Let’s look at the same plot using some of the other commonly used themes:

theme_bw(): White background with gridlines
theme_classic(): Similar to theme_bw, but no gridlines
theme_void: Empty theme; only geoms are visible.





To see a list of available themes, use the R-Studio help pane.

8. Themes

Modifying themes…
You have the ability to modify the look of your plot by overriding the default settings for an existing theme.

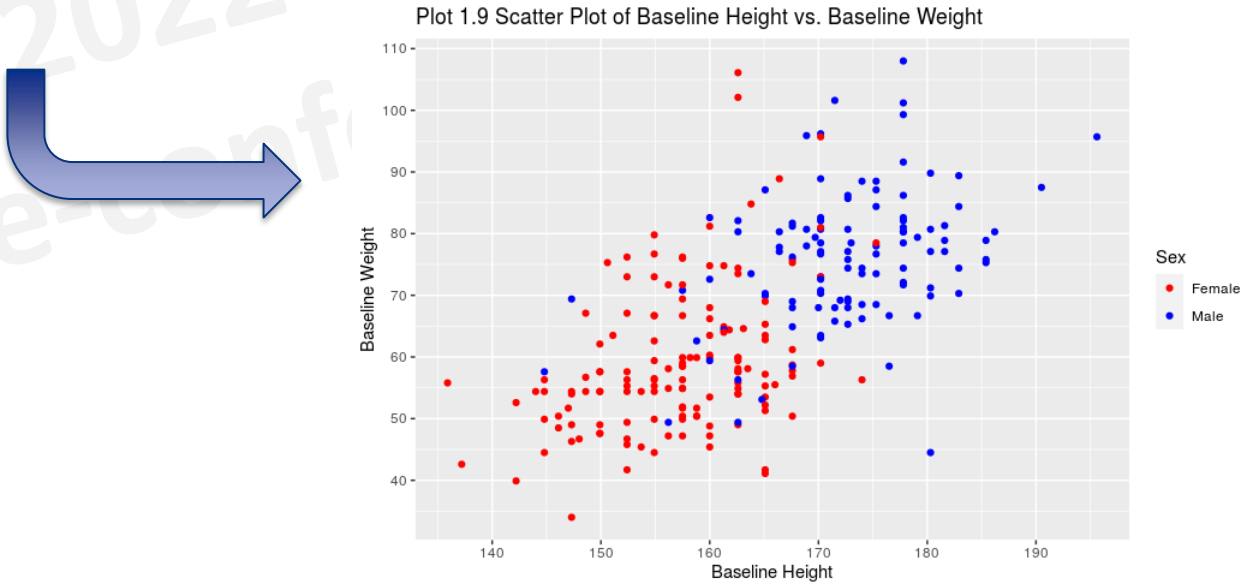
Through the use of the theme() function, you have control over every element of any given theme (i.e., the text, lines, and rectangles). But that is a topic for another time….

8. Themes

Titles:

For now, we will limit our work on themes to simply choosing a theme and adding a title. One way to add titles is using the **labs** function:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL, color = SEX)) +
  geom_point() +
  scale_x_continuous(name = "Baseline Height", breaks = seq(140, 220, by = 10)) +
  scale_y_continuous(name = "Baseline Weight", breaks = seq(40, 200, by = 10)) +
  scale_color_manual(name = "Sex" values=c("red","blue"),labels = c("Female","Male")) +
  labs(title = "Plot 1.9 Scatter Plot of Baseline Height vs. Baseline Weight")
```

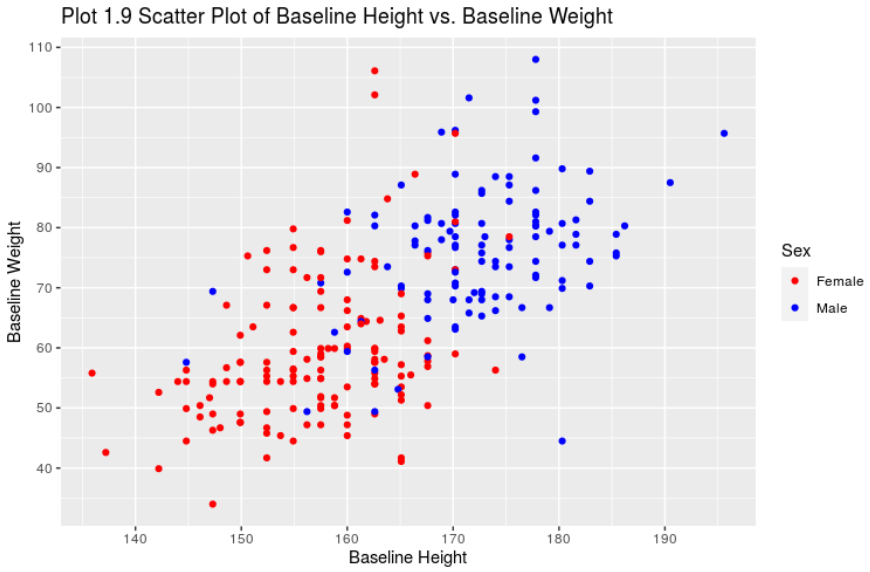


8. Themes

Other text:

The labs() function actually provides us with the ability to modify more than just the title text; this is an alternate way to modify the axis and legend labels:

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL, color = SEX)) +
  geom_point() +
  scale_x_continuous(name = "Baseline Height", breaks = seq(140, 220, by = 10)) +
  scale_y_continuous(name = "Baseline Weight", breaks = seq(40, 200, by = 10)) +
  scale_color_manual(name = "Sex", values=c("red","blue"), labels = c("Female","Male")) +
  labs(title = "Plot 1.9 Scatter Plot of Baseline Height vs. Baseline Weight",
       x     = "Baseline Height",
       y     = "Baseline Weight",
       color = "Sex")
```



9. Coordinates, Statistics and Facets

Finally, we will touch base on the last three layers:

Graphical Elements	
Data	The data frame you want to plot
Aesthetics	The scales onto which we wish to map our data.
Geometries	The visual elements used in our data (i.e., how the plot will look)
Coordinates	The space on which data will be plotted
Statistics	Representations of our data to aid understanding
Facets	Plotting small multiples
Themes	All non-data ink

9. Coordinates, Statistics and Facets

The Coordinate Layer:
When you call the ggplot function, the default coordinate system is the Cartesian coordinate system, where the x and y position act independently to find the location of a point. There are a number of other coordinate systems that are occasionally helpful:

- coord_flip(): Cartesian coordinate system with x and y axes flipped.
- coord_fixed(): Cartesian coordinate system with a fixed aspect ratio.
- coord_map(): Map projections.
- coord_polar(): Polar coordinates.
- coord_trans(): Apply arbitrary transformations to x and y positions, after the data has been processed by the statistic.

The Statistics Layer
Some statistics occur within the geoms (e.g., geom smooth) while others can be called outside of geoms. However, that is a topic for another time.

9. Coordinates, Statistics and Facets

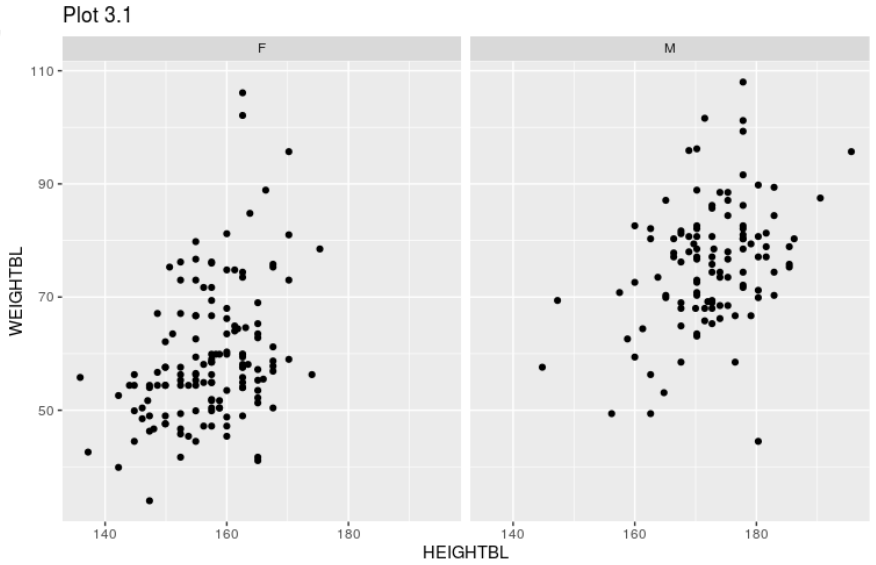
The Facet Layer

Facetting generates small multiples of a plot, each showing a different subset of the data.

There are three types of faceting:

- facet_null(): a single plot, the default.
- facet_wrap(): “wraps” a 1-dimesional ribbon of panels into 2 dimensions.
- facet_grid(): produces a 2-dimesional grid of panels.

```
ggplot(data = adsl, aes(x = HEIGHTBL, y = WEIGHTBL)) +
  geom_point() +
  facet_wrap(~SEX, nrow=1)
```



9. Coordinates, Statistics and Facets

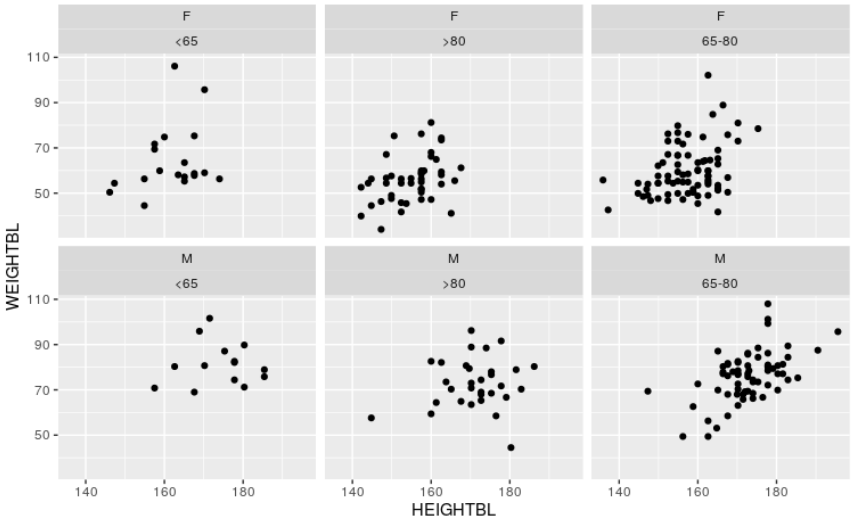
The Facet Layer

facet_wrap() **vs.** facet_grid:

- Wraps a 1d sequence of panels in 2d **vs.** produces a 2-dimesional grid of panels (most useful for two discrete variables).
- Shows only plots with values **vs.** display plots even if some of them are empty.

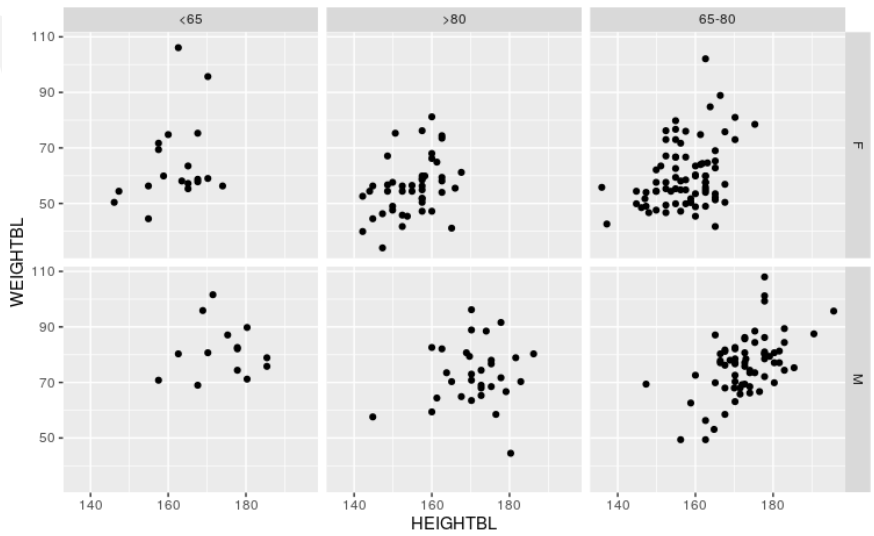
```
ggplot(data = adsl, aes(x = HEIGHTBL,
                        y = WEIGHTBL)) +
  geom_point() +
  facet_wrap(SEX ~ AGEGR1 )
```

Plot 3.2




```
ggplot(data = adsl, aes(x = HEIGHTBL, y =
                        WEIGHTBL)) +
  geom_point() +
  facet_grid(SEX ~ AGEGR1)
```

Plot 3.3



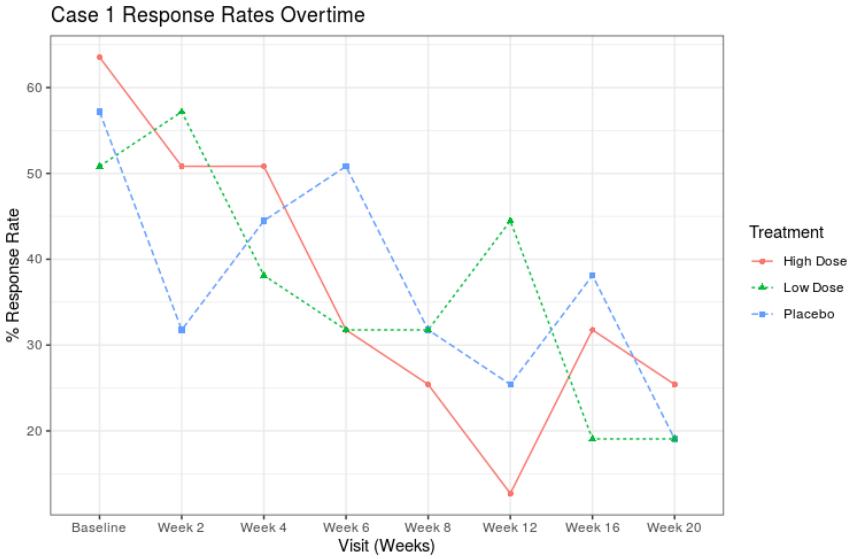
10. Summary

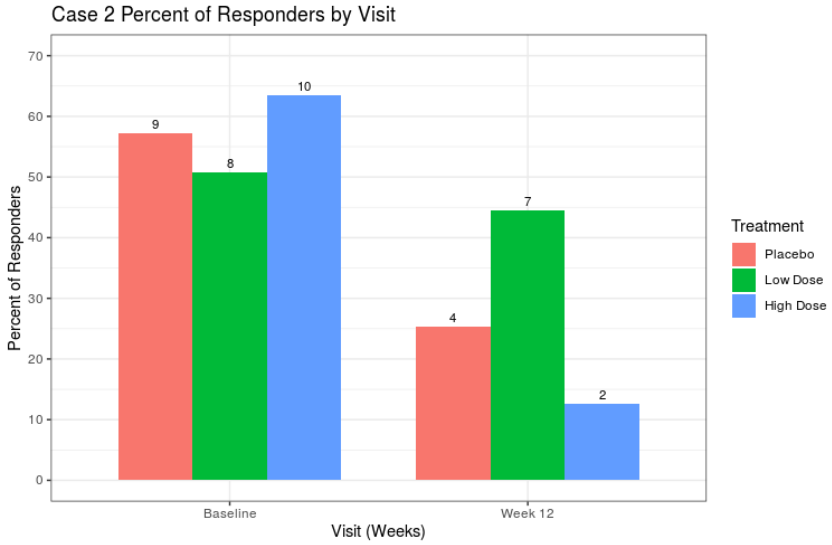
- With ggplot2, you begin a plot with the function ggplot(), which creates a coordinate system. Then you can build a graph upon that coordinate system, layer by layer.
- The data layers, aesthetic layer and geometries layer are required.
- You can use the other layers to improve your graph.

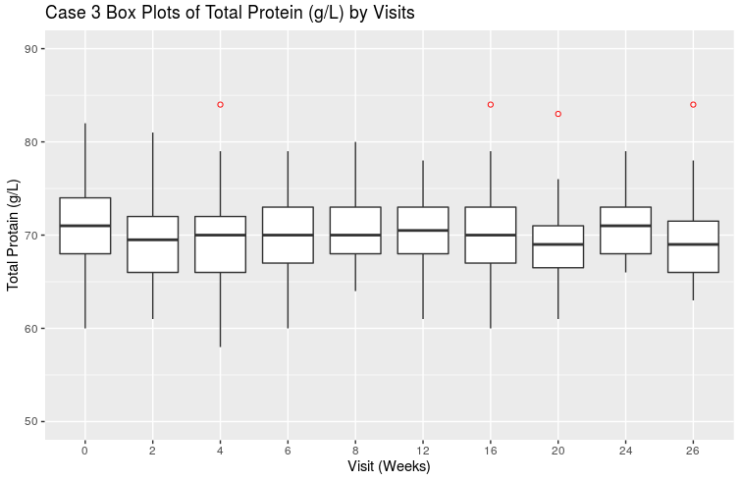
GGPLOT2 Intermediate – Common Figures

Aug 18, 2022

In this session, we will use our knowledge from the previous lesson to create some other common figures…





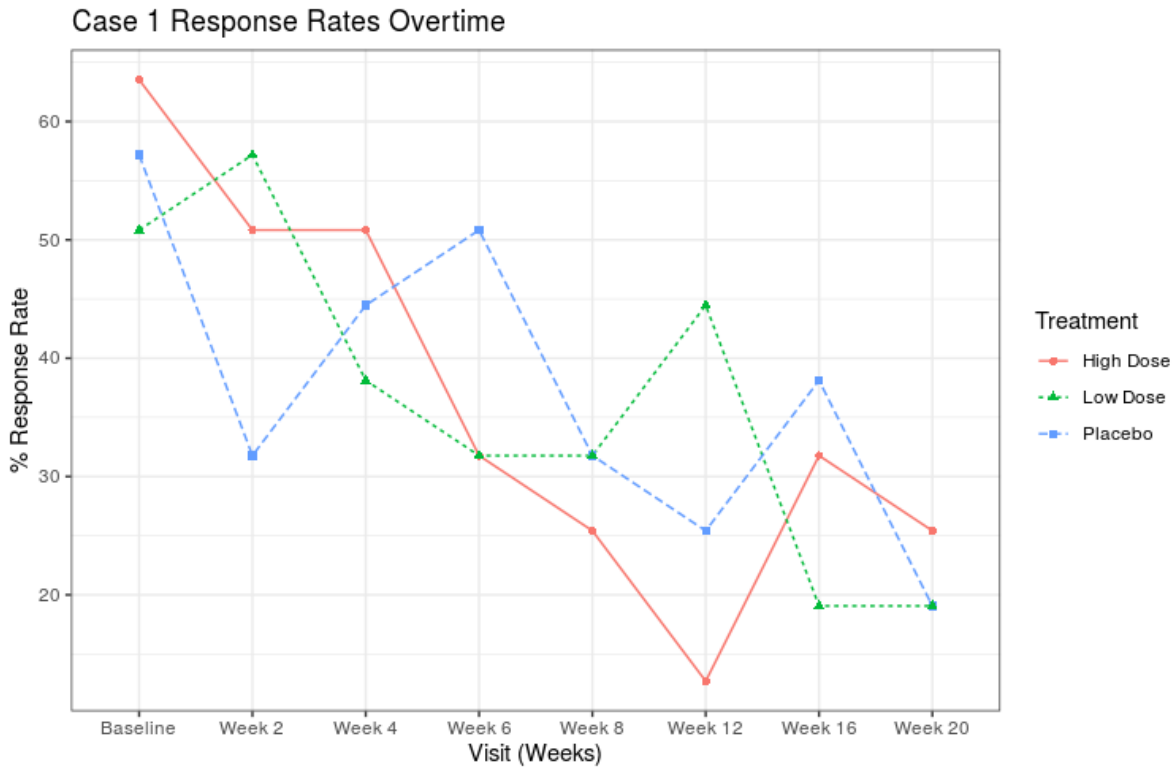


Go ahead load the ggplot2, dplyr and haven.

```
library(ggplot2)
```

Data: rspsum

rspsum				
AVISITN	AVISIT	TRT	RESPONSE	COUNT
0	Baseline	Placebo	57.17916137	9
0	Baseline	High Dose	63.53240152	10
0	Baseline	Low Dose	50.82592122	8
2	Week 2	Placebo	31.76620076	5
2	Week 2	High Dose	50.82592122	8
2	Week 2	Low Dose	57.17916137	9
4	Week 4	Placebo	44.47268107	7
4	Week 4	High Dose	50.82592122	8
4	Week 4	Low Dose	38.11944091	6
6	Week 6	Placebo	50.82592122	8
6	Week 6	High Dose	31.76620076	5
...				



Case 1: Line Graphs

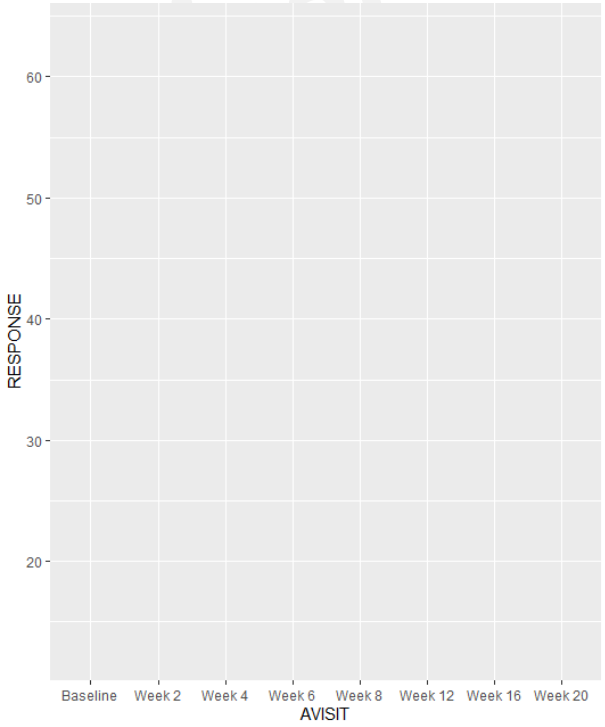
Let's begin creating a line graph… Remember, that any basic plot requires us to define the data, the aesthetic mappings (aes), and the geometries (geom_xxx).

For this example, we are going to work with the following response data:

rspsum

AVISITN	AVISIT	TRT	RESPONSE	COUNT
0	Baseline	Placebo	57.17916137	9
0	Baseline	High Dose	63.53240152	10
0	Baseline	Low Dose	50.82592122	8
2	Week 2	Placebo	31.76620076	5
2	Week 2	High Dose	50.82592122	8
2	Week 2	Low Dose	57.17916137	9
4	Week 4	Placebo	44.47268107	7
4	Week 4	High Dose	50.82592122	8
4	Week 4	Low Dose	38.11944091	6
6	Week 6	Placebo	50.82592122	8
6	Week 6	High Dose	31.76620076	5
…				

```
ggplot(data = rspsum, aes(y=RESPONSE, x=AVISIT))
```



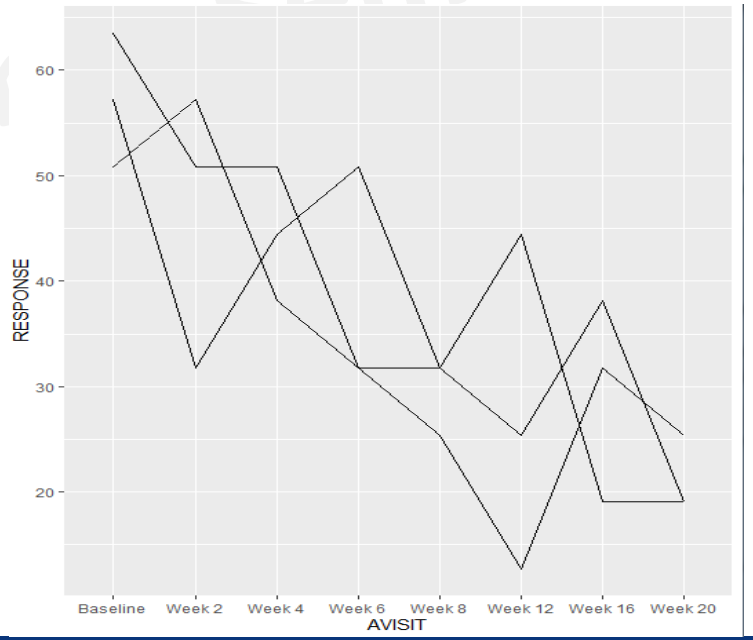
Case 1: Line Graphs

Now that we have the data, we can create a line plot of response over time by mapping y to RESPONSE and x to AVISIT, and then adding the geometry layer:

```
ggplot(data=rspsum, aes(y=RESPONSE, x=AVISIT))+
 geom_line()
```

We can further break this out by treatment group, by mapping group to treatment.

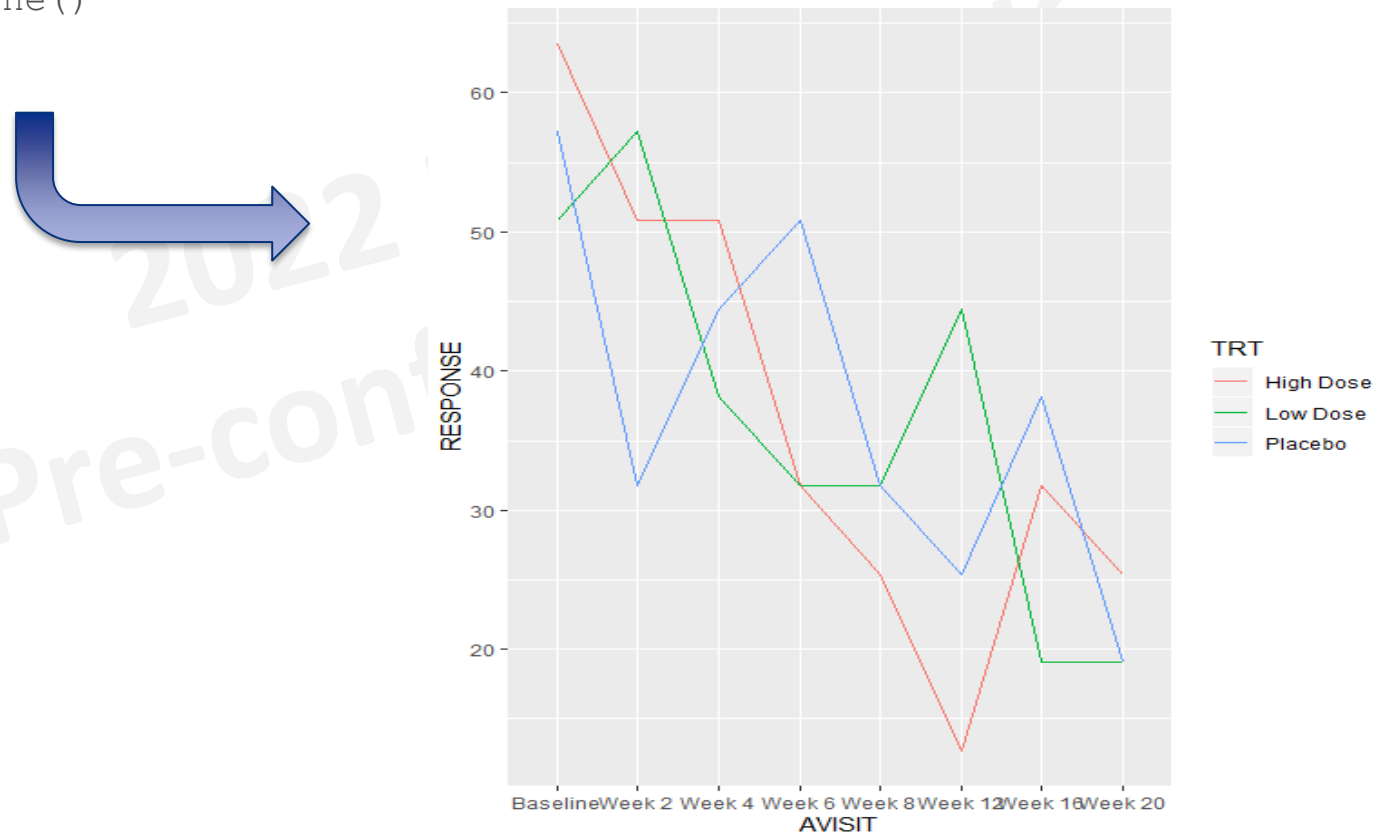
```
ggplot(data=rspsum, aes(y=RESPONSE, x=AVISIT, group = TRT)) +
geom_line()
```



Case 1: Line Graphs

We have the response over time grouped by treatment, but it is difficult to tell which group is which. Let's correct this by mapping color to TRT as well:

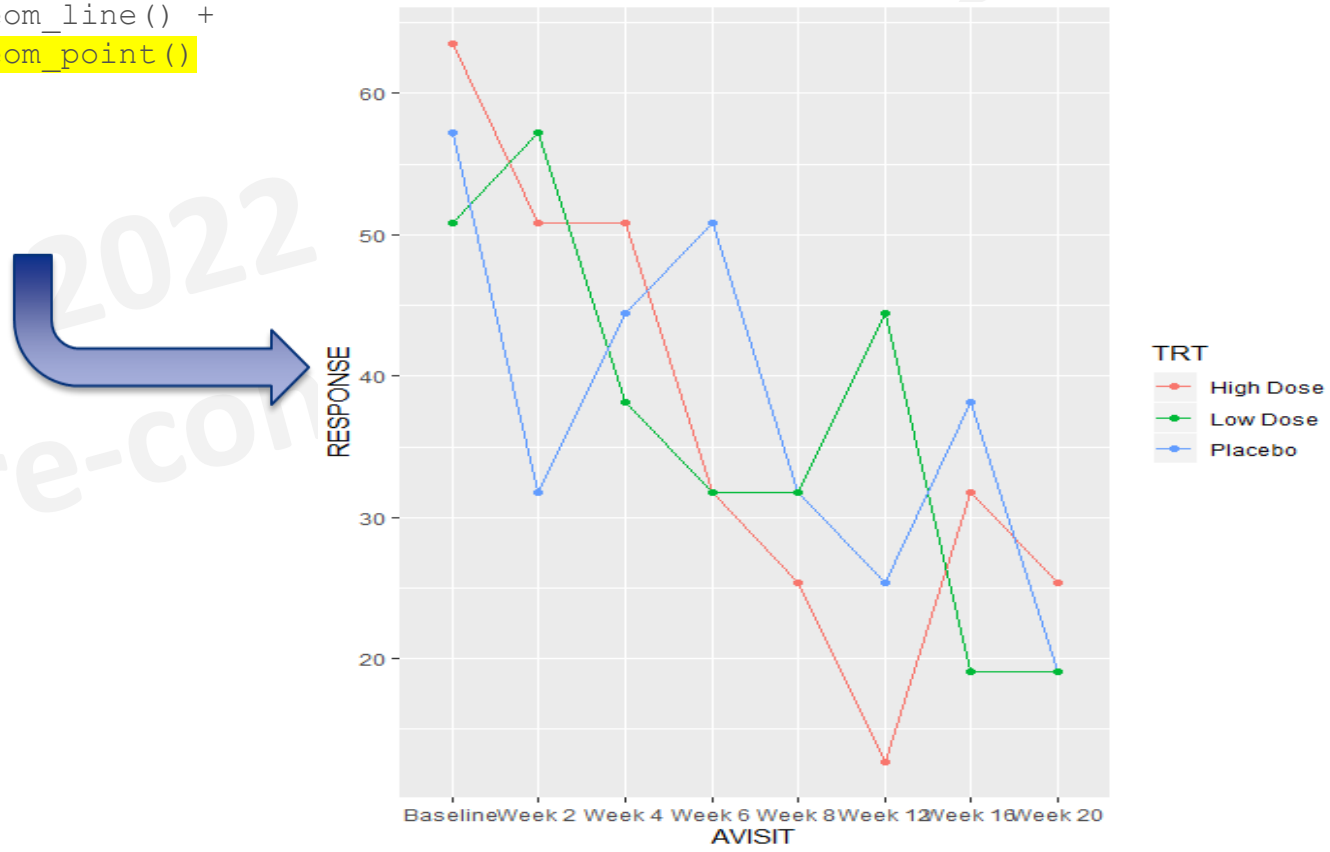
```
ggplot(data=rspsum, aes(y=RESPONSE, x=AVISIT, group=TRT, color = TRT))+
 geom_line()
```



Case 1: Line Graphs

Now we get to see some of the power of the layered approach to graphics. If we wish to add points to our lines, we simply add another geometry layer:

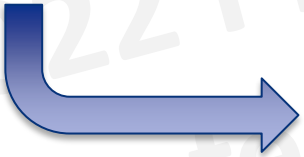
```
ggplot(data = rspsum, aes(y = RESPONSE, x = AVISIT, group = TRT, color = TRT)) +
       geom_line() +
       geom_point()
```

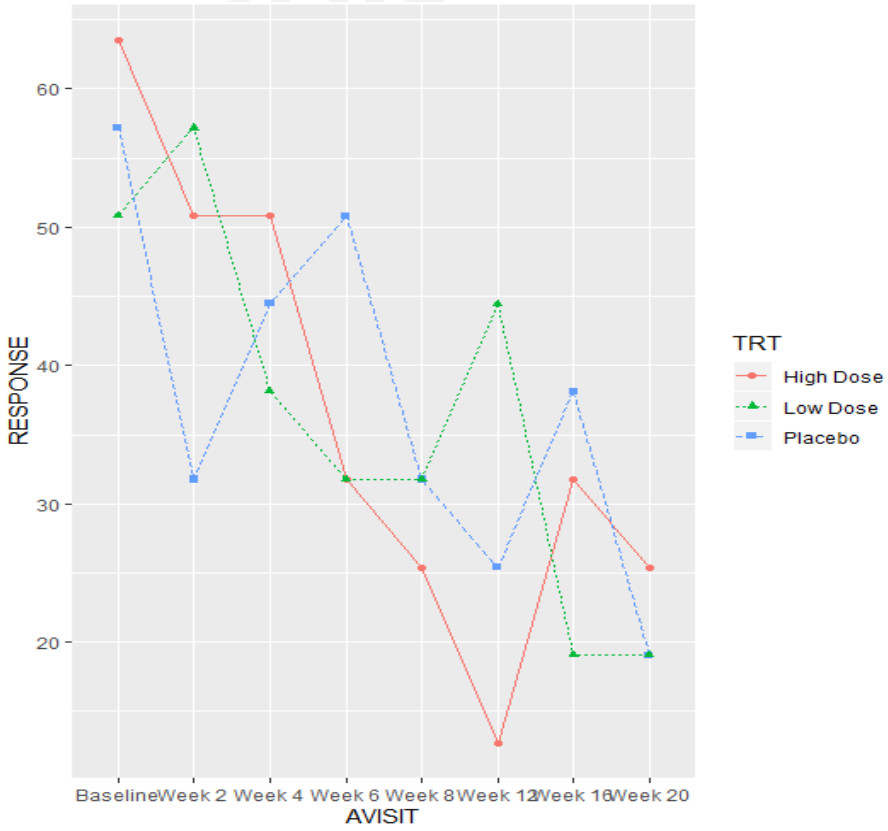


Case 1: Line Graphs

Now let's add some additional aesthetic mappings to help the viewability of our plot; geom_line allows us to control the linetype while geom_point allows us to control the shape of the points:

```
ggplot(data=rspsum, aes(y=RESPONSE, x=AVISIT, group=TRT, color = TRT)) +
  geom_line(aes(linetype=TRT) ) +
  geom_point(aes(shape=TRT) )
```

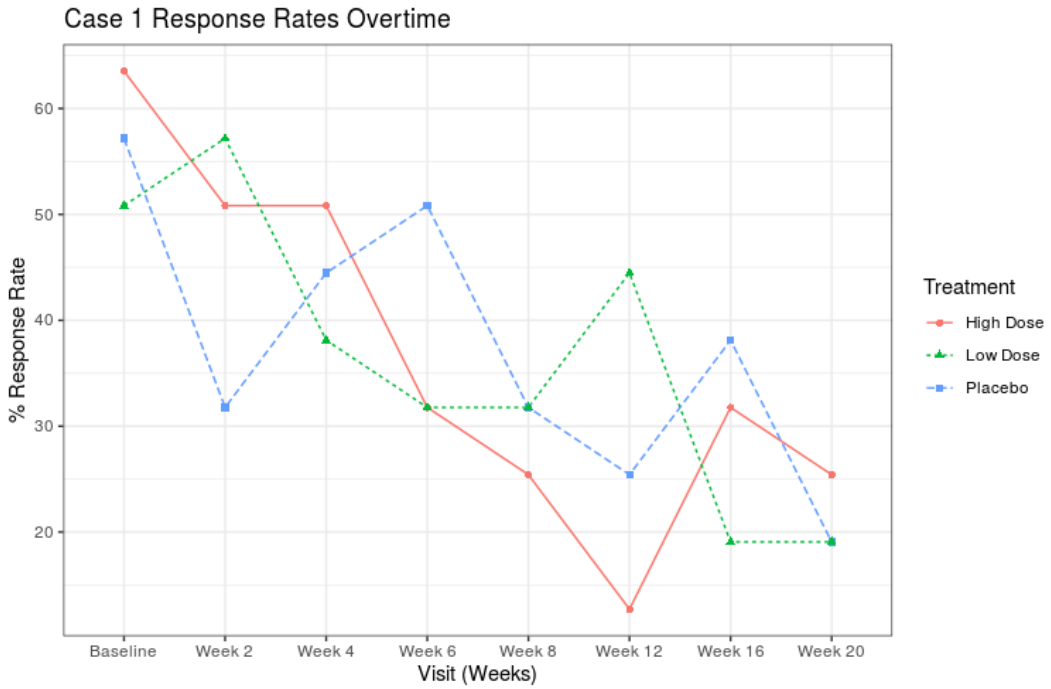




Case 1: Line Graphs

Finally, let's polish our plot up by adding more descriptive text and changing the theme:

```
ggplot(data=rspsum, aes(y=RESPONSE, x=AVISIT, group=TRT, color=TRT)) +
  geom_line(aes(linetype=TRT) ) +
  geom_point(aes(shape=TRT) ) +
  labs(title = 'Case 1 Response Rates Overtime',
            x = 'Visit (Weeks)',
            y = '% Response Rate',
        shape ='Treatment',
    linetype ='Treatment',
        color ='Treatment') +
   theme_bw()
```



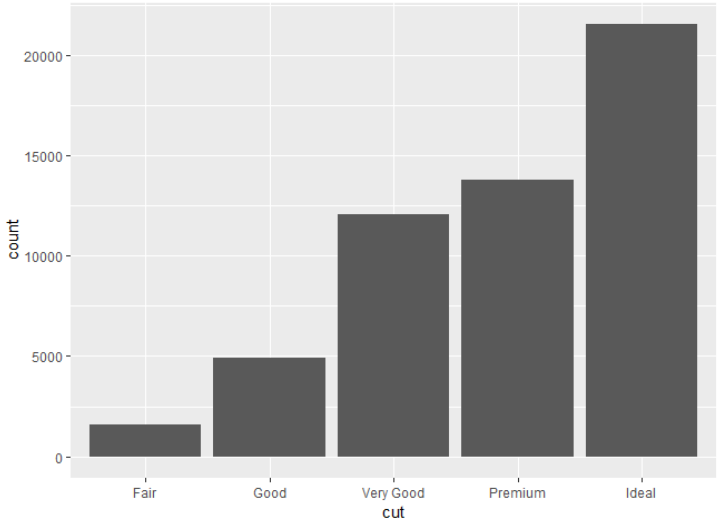
Case 2: Bar Graphs

Next, let's look at Bar Graphs. We'll start off by looking at the diamonds dataset (comes with the ggplot2 package):

diamonds									
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.290	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
…									

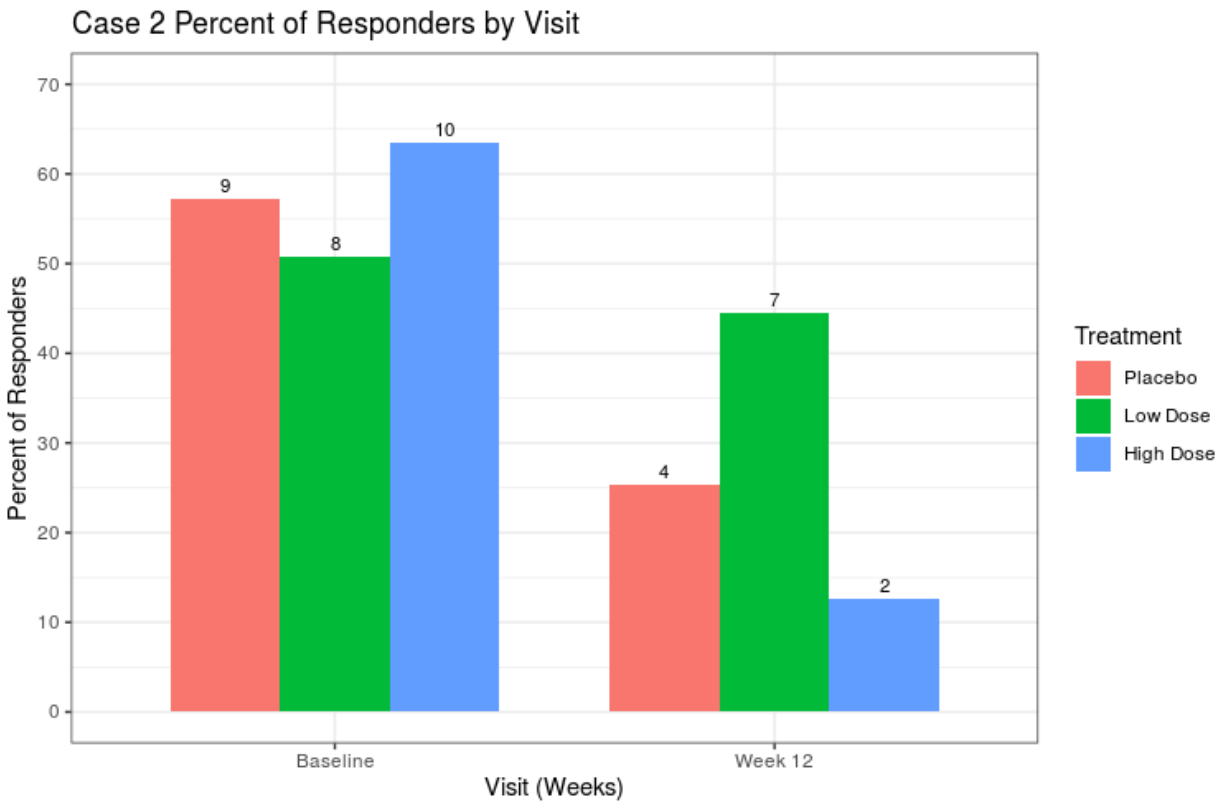
The following chart displays the total number of diamonds in the diamonds datasets, grouped by cut. Notice that we've mapped the aesthetic X to the variable cut. Y, count, represents the number of observations in each group. It is calculated by geom_bar.

```
ggplot(data = diamonds, aes(x = cut)) +
  geom_bar()
```



Data: rspsum

rspsum				
AVISITN	AVISIT	TRT	RESPONSE	COUNT
0	Baseline	Placebo	57.17916	9
0	Baseline	High Dose	63.53240	10
0	Baseline	Low Dose	50.82592	8
12	Week 12	Placebo	25.41296	4
12	Week 12	High Dose	12.70648	2
12	Week 12	Low Dose	44.47268	7



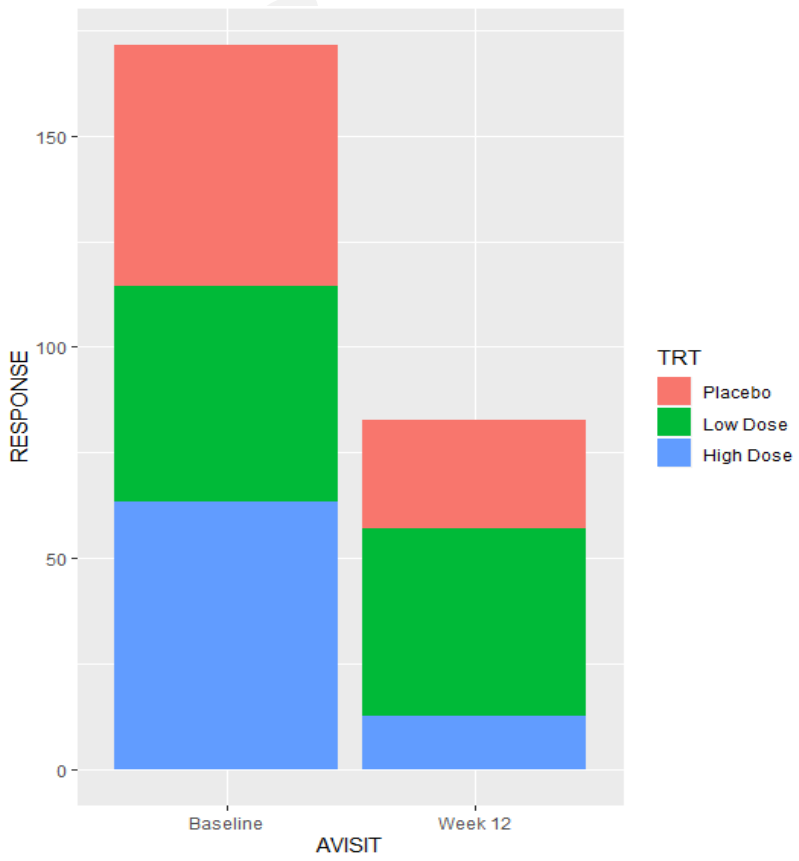
Case 2: Bar Graphs

Alternatively, we can use the stat parameter to work with a summary data set (like we did with the line plot). Let's look at the treatment response at Baseline and Week 12:

rspsum				
AVISITN	AVISIT	TRT	RESPONSE	COUNT
0	Baseline	Placebo	57.17916137	9
0	Baseline	High Dose	63.53240152	10
0	Baseline	Low Dose	50.82592122	8
2	Week 2	Placebo	31.76620076	5
2	Week 2	High Dose	50.82592122	8
2	Week 2	Low Dose	57.17916137	9
4	Week 4	Placebo	44.47268107	7
4	Week 4	High Dose	50.82592122	8

By setting the parameter stat = "identity", we can now display the value of the RESPONSE variable directly (rather than having ggplot derive it).

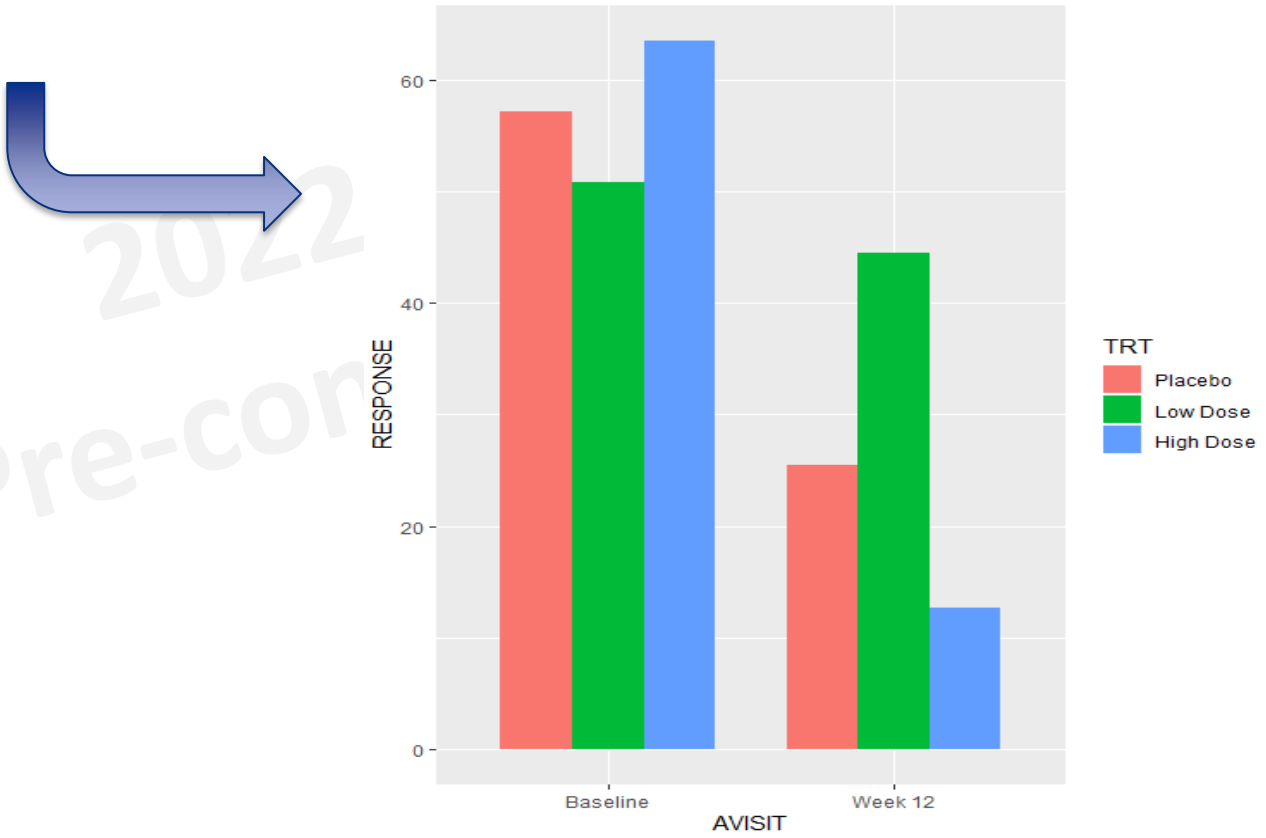
```
ggplot(rspsum, aes(y=RESPONSE, x=AVISIT, fill=TRT)) +
  geom_bar(stat = "identity")
```



Case 2: Bar Graphs

Now let's use the position parameter to "un-stack" the bars, and the width parameter to adjust the width of the bars:

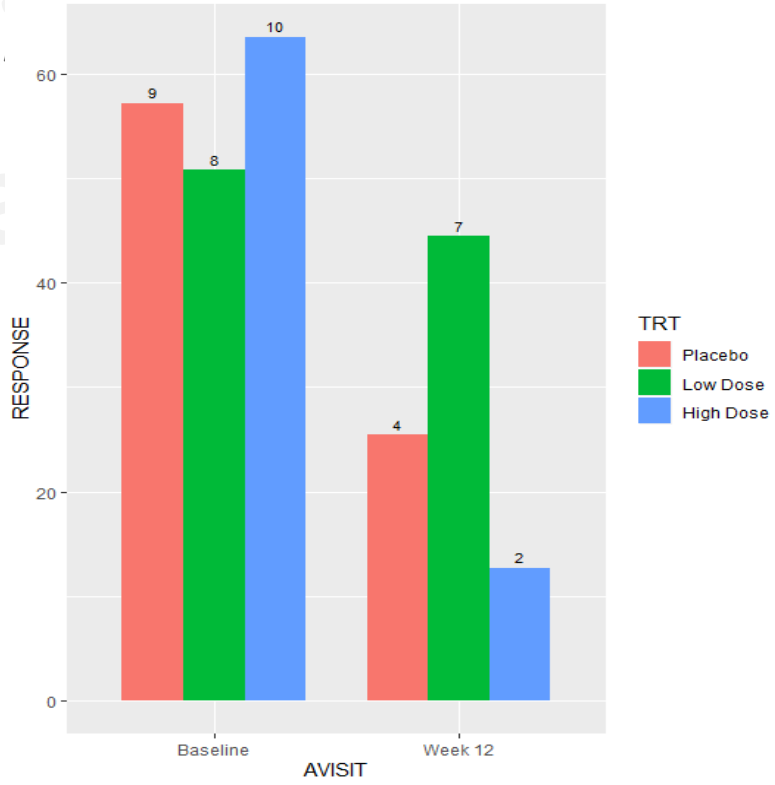
```
ggplot(rspsum, aes(y=RESPONSE,  x=AVISIT, fill = TRT)) +
  geom_bar(stat = "identity", position = 'dodge', width = 0.75)
```



Case 2: Bar Graphs

And once again, showing the power the layered approach to graphics, we can layer labels on top of our bars using geom_text, and the 'number' function in the 'scales' package:

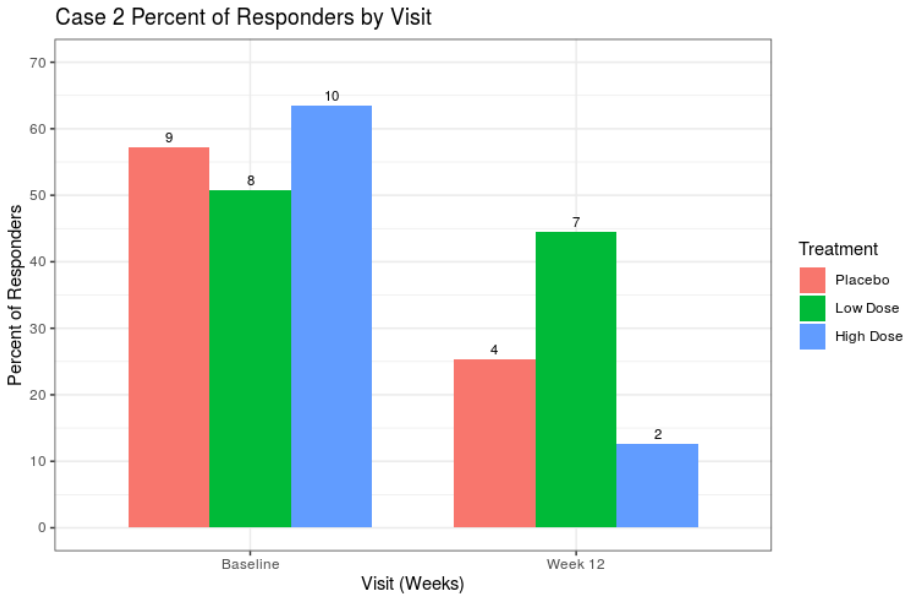
```
ggplot(rspsum, aes(y=RESPONSE,  x=AVISIT, fill = TRT)) +
  geom_bar(stat = "identity", position = 'dodge', width = 0.75) +
  geom_text(aes(label = COUNT),
            position  = position_dodge(width = .75),
            vjust     = -0.5,
            size      = 3)
```



Case 2: Bar Graphs

And to finish off our plot, we can adjust the scales, labels and theme exactly the same way as before:

```
ggplot(rspsum, aes(y = RESPONSE, x = AVISIT, fill = TRT)) +
  geom_bar(stat = "identity", position = 'dodge', width = 0.75) +
   geom_text(aes(label = COUNT),
             position = position_dodge(width = .75),
             vjust    = -0.5,
             size     = 3) +
  scale_y_continuous(limits=c(0, 70), breaks=seq(0, 70, 10)) +
  labs(title = 'Case 2 Percent of Responders by Visit',
          x = 'Visit (Weeks)',
          y = 'Percent of Responders',
       fill ='Treatment') +
   theme_bw()
```



Case 3: Box Plots

The final type of plot that we are going to look at is boxplots. Not surprisingly, this type of plot is generated using the boxplot geom.

For this plot, we are going to use the ADLBC dataset (subsetting for TRTA 'Xanomeline High Dose' and PARAMCD = 'PROT'), and we'll map x to the AVISITN variable and y to the AVAL variable.

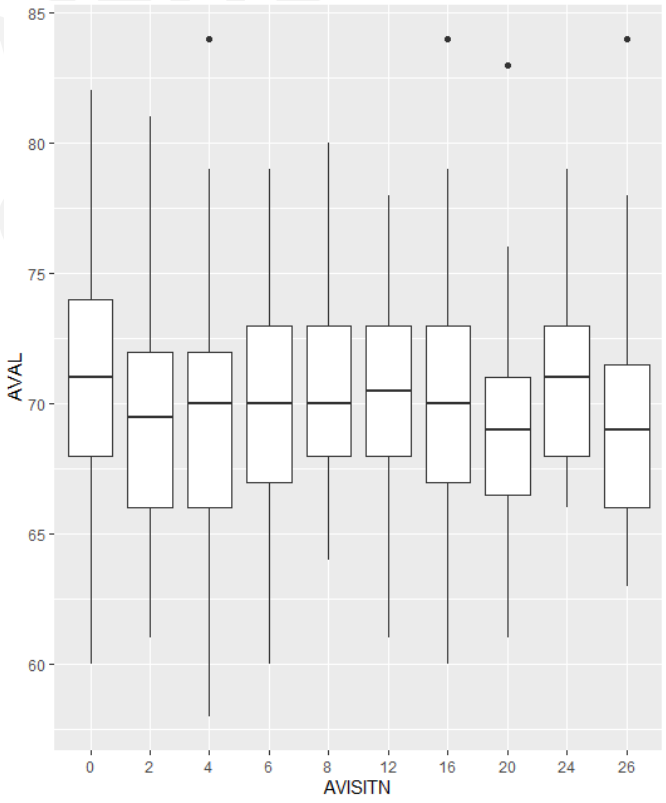
SUBJID	AVISITN	AVISIT	PARAM	PARAMCD	AVAL
1028	0	Baseline	Protein (g/L)	PROT	72
1028	2	Week 2	Protein (g/L)	PROT	72
1028	4	Week 4	Protein (g/L)	PROT	69
1028	6	Week 6	Protein (g/L)	PROT	69
1028	8	Week 8	Protein (g/L)	PROT	67
1028	12	Week 12	Protein (g/L)	PROT	78
1028	16	Week 16	Protein (g/L)	PROT	62
1028	20	Week 20	Protein (g/L)	PROT	66
1028	24	Week 24	Protein (g/L)	PROT	67
1028	26	Week 26	Protein (g/L)	PROT	70
1034	0	Baseline	Protein (g/L)	PROT	75
1034	2	Week 2	Protein (g/L)	PROT	74
1034	4	Week 4	Protein (g/L)	PROT	71
1034	6	Week 6	Protein (g/L)	PROT	69
1034	8	Week 8	Protein (g/L)	PROT	74
…					

Case 3: Box Plots

Again, we can create a basic plot by calling the ggplot function, specifying the data, the aesthetic mappings, and calling the appropriate geom; in this case, geom_boxplot.

The upper whiskers extend to the largest value within 1.5 IQR over the 75th percentile, lower whiskers extend to the smallest value within 1.5 IQR below the 25th percentile.

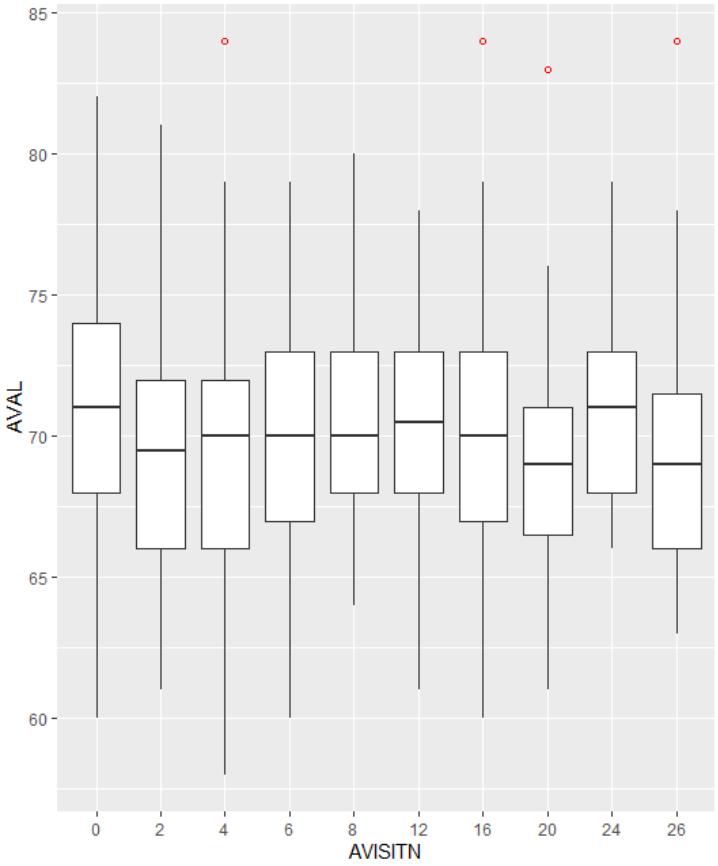
```
ggplot(data = chem, aes(y=AVAL,  x=AVISITN)) +
   geom_boxplot()
```



Case 3: Box Plots

The boxplot geom provides a number of parameters to control the appearance of our plots; here, we are changing the shape and color of our outliers.

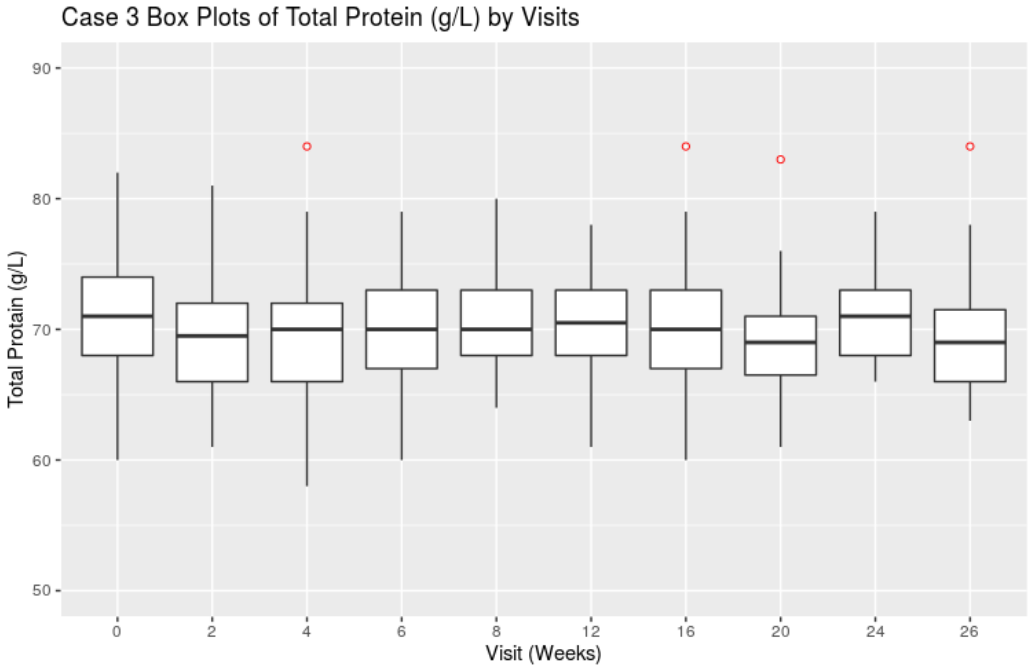
```
ggplot(chem, aes(y=AVAL,  x=AVISITN)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 1)
```



Case 3: Box Plots

Finally, let's put on finishing touches on the plot by adjusting the scales, labels and theme.

```
ggplot(chem, aes(y=AVAL,  x=AVISITN)) +
  geom_boxplot() +
  geom_boxplot(outlier.colour = "red", outlier.shape = 1) +
  scale_y_continuous(limits=c(50, 90), breaks=seq(50, 90,10)) +
  labs(title = 'Case 3 Box Plots of Total Protein (g/L) by Visits',
       x     = 'Visit (Weeks)',
       y     = 'Total Protain (g/L)') +
  theme()
```



10. Summary

- In this session, we've continued to observe the power of the layered approach to graphics:
  - We can build basic plots with very little effort
  - Then, we can make the plots increasingly more sophisticated by simply adding additional layers to out plots.

- You now have the ability to create Scatterplots, Line plots, Bar charts, and Boxplots.

- If you want to learn additional plots, remember… a new plot is only a very basic geom call away; start with the basic call, and build your layers from there.

Thanks!