Intro to ggplot2

Morgan Sparks

10/9/2018

Intro to ggplot

We're going to learn the basics of ggplot today, especially as it pertains to graphing GLMs that have one continuous and one categorical variable.

Let's start by getting more familiar with our data. We will be using Fisher's Iris dataset which is a dataset built in to R iteslf. The dataset can just be referenced by `iris`. The data set has variables that are sepal length and width, as well as petal length and width for three species of the iris plants.

Also, before we get started, the end all be all for information about ggplot is this site https://ggplot2.tidyverse.org/reference/index.html. If your question isn't answered there, I can almost guarantee it is answered somewhere else, ggplot is quickly becoming the default plotting program of R.

One final note I would like to make about ggplot is that one of the major differences between ggplot and base R plotting is how the graphing program interacts with your dataset. Base R is much more flexible at reading in your data, whereas ggplot prefers your data is basically in a format where every datum is a row itself with all the information for independent variables as columns along that row. For instance if you were looking at something like hatch timing and noted 15 individuals hatched on given day, each of those individuals would have to be a specific line for ggplot to incorporate them. A lot of the reshaping of data can be done with the `dplyr` package.

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

Now, let's look at our data:

```
iris_data <- iris
str(iris_data)
View(iris_data)
```

Getting started with ggplot

The first step to plotting with ggplot is to assign the dataset and variables you will be working with. This creates a completely blank plot, no x or y axes. But, is ready for us to layer upon with different ggplot commands.

```
ggplot(iris_data, aes(x = , y = ))
```

Note that all this does is make a blank plot with our x and y variables specified and scales that ggplot tries to guess based on the range of the data in the dataset.

How to display data in ggplot

ggplot uses geoms (also known more generally as layers) to specify different ways of displaying datasets. Now there are a number which that have all kinds of purposes, but there are a few you are likely to commonly use. These include things like `geom_point` for points, `geom_bar` for barplots, `geom_histogram` for histograms, `geom_boxplot` for boxplots, etc. You get the point. For a much fuller list of available geoms and their use see https://ggplot2.tidyverse.org/reference/index.html#section-layer-geoms.

Let's spend some time looking at our data with different geoms, to get a handle on what our data looks like. Let's think of petal length as our dependent variable in this situation. Let's start with `geom_histogram` to get a sense of the distribution of our data.

```
ggplot(iris_data, aes(x = Petal.Length)) +
  geom_histogram()
```

Perhaps we don't like our bin size, we can easily change it with the following code. Binwidth here is based on your data and is up to you to choose. Here we specifify bins for every 0.1 cm.

```
ggplot(iris_data, aes(x = Petal.Length)) +
  geom_histogram(binwidth = 0.1 )
```

We can also create a smoother density line like we have in class with `geom_density` (adjust here is similar to binwidth above, the 1/4 here means drecrease bindwidth by 1/4. If it were 2, it would mean increase bindwidth by 2. You would need to figure out what is best for your data)

```
ggplot(iris_data, aes(x = Petal.Length)) +
  geom_density(adjust = 1/4)
```

If we look at our histogram and our density plots, it looks like our data might be multimodal. Perhaps it's time to consider our categorical variable. To do this we can create boxplots that give you a sense of the data you are working with. Here it makes sense to look at petal lenght as a function of our only categorical variable, species.

```
ggplot(iris_data, aes(x = Species, y =Petal.Length)) +
  geom_boxplot()
```

Indeed, it looks like we have different distributions for our species.

Okay, now that we have a good sense of the distribution of our data, let's think more about our modeling appraoch. Let's say for the sake of this exercise, we think that plants with longer sepals will also have longer petals.

Because these data are continuous, it makes sense to display them as a scatterplots with `geom_point`.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length)) +
  geom_point()
```

Okay, now it looks like generally there is fairly strong relationship between sepal length and petal length. Assuming we've gone through all our standard checks testing the assumptions of our GLM and our data looks good, let's add a linear regression to our plot.

We do this by adding another geom (or layer), in this case `geom_smooth` which is the command for adding models to figures in ggplot

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length)) +
  geom_point() +
  geom_smooth(method = "lm")
  # note that method here refers to the type of model we want to use, there are many types available
  # "lm"" here referes to linear model, but we could fit models like gams or loess with other arguments
```

So now we have a linear model fit to our data. But recall our data set. If you use the `names()` you'll see that not only do we have continuous variables for petals and sepals, but we also have the categorical variable of species which we looked at with our boxplot. There are three species present in our data set and perhaps if we considered species in our model it would look different. Let's do that now by using the group designation inside our aesthetic (`aes()`) argument.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species)) +
  geom_point() +
```

```
  geom_smooth(method = "lm")
```

Now we've fit our model with one continuous and one categorical variable, where we consider both continuous and categorical independent variables in our model. Still, though this plot leaves a lot to be desired. Let's make some simple changes to make it more visually appealing.

Let's start by changing the colours to match the species.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point() +
  geom_smooth(method = "lm")
```

Let's next relable our axes and change axes sizes so our data doesn't extend past our axes labels.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point() +
  geom_smooth(method = "lm") +
  ylim(0,8) + # here you are setting the y axis limits as 0 (the origin) all the way to 8
  xlim(0,8) + # same as above but for the x axis
  labs(x = "Sepal length (cm)", y = "Petal length (cm)")
```

Perhaps we think the points are a little too large and too bold, and want to bring attention just to the line. Let's change point size and opacity, as well as remove the standard error shading on the line.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point(size = 1.3, alpha = 0.33) + #alpha is how we specify opacity, it is on a scale of 0 (compl
  geom_smooth(se = FALSE, method = "lm") +
  ylim(0,8) +
  labs(x = "Sepal length (cm)", y = "Petal length (cm)")
```

How about we change our colors now, especially given the prevalence of colorblindess (inability to tell reds from greens). The easiest way to do this is with `scale_color_brewer` and to specify the palette of your choice (see palettes http://www.sthda.com/english/wiki/colors-in-r). A word of warning, setting colors does and will get complicated quickly. We won't get into the specifics, but just know there a tremendous number of ways and reasons to specify colors in ggplot.

Here we use the "Dark2" palette.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point(size = 1.3, alpha = 0.33) +
  geom_smooth(se = FALSE, method = "lm") +
  ylim(0,8) +
  labs(x = "Sepal length (cm)", y = "Petal length (cm)") +
  scale_color_brewer(palette = "Dark2" )
```

Now, one of the common complaints about base ggplot figures is their background designation. Lot's of people don't like the grey background or the axis lines. Luckily this is an easy change using themes. Again, with ggplot and R in general, there are tons of ways different styles to choose from. My favorite is `theme_classic` as it produces clean figures in a style that most journals like.

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point(size = 1.3, alpha = 0.33) +
  geom_smooth(se = FALSE, method = "lm") +
  ylim(0,8) +
  labs(x = "Sepal length (cm)", y = "Petal length (cm)") +
  scale_color_brewer(palette = "Dark2" ) +
  theme_classic()
```

One of the most important things to do once you've created your finished plot is to save it as a file. Journals have all kinds of demands on how to format plots, but one of the most reliable methods is to save as a PDF. You can do so (and save as other file types) with `ggsave()` command. It will save a copy of your figure in the specified filetype and size to your working directory. See it's application below:

```
ggplot(iris_data, aes(x = Sepal.Length, y =Petal.Length, group = Species, colour = Species)) +
  geom_point(size = 1.3, alpha = 0.33) +
  geom_smooth(se = FALSE, method = "lm") +
  ylim(0,8) +
  labs(x = "Sepal length (cm)", y = "Petal length (cm)") +
  scale_color_brewer(palette = "Dark2" ) +
  theme_classic() +
  ggsave("SepalvsPetal.pdf", width = 4.25, height = 5.5, units = "in")
```

Plotting Least Square Means

Briefly, we want to show you how to produce the least square means in a figure that you would obtain with a GLM that has a continuous and a categorical independent variable. To do this we use the following code.

First, we must create our model:

```
iris_reg <-  lm(Petal.Length ~ Sepal.Length + Species, data = iris_data)
```

We then use the `emmeans` package to produce a table of these results. We run the `emmeans()` by telling it what model (iris_reg) and what categorical variable (Species) we are interested in. It will print a data table with the least square means, as well as other summary information.

```
# install.packages("emmeans") install if you haven't done so

library(emmeans)

emmeans(iris_reg, "Species")
```

Now we want to plot these values as a point with associated standard errors. But recall how we talked about earlier that ggplot is very specific about the data it will accept. Using `str()` we see that `iris_ls` is in and "emmGrid" object (an object type specific to the emmeans package and not transferable to other pakcages), which is not something ggplot can handle, so we must convert the data into a dataframe, the main ggplot data format.

```
# install.packages("emmeans") install if you haven't done so

iris_ls <- emmeans(iris_reg, "Species")

str(iris_ls)

iris_ls_df <- data.frame(iris_ls) #data.frame() changes the object type to a dataframe

str(iris_ls_df) #best to make sure thay actually worked
```

Now we can begin plotting the least squared means using the `iris_ls_df` dataframe we just created. Let's start with a simple version of that figure. Notice the syntax change with `geom_errorbar`, this is due to the fact that you need to create a seperate set of aesthetics (`aes()`) for your ymin and ymax values.

```
ggplot(iris_ls_df, aes(x = Species, y = emmean)) +
  geom_point() +
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_df)
```

Now let's make this figure a little more presentable using some of the tricks we learned earlier.

```
ggplot(iris_ls_df, aes(x = Species, y = emmean, colour = Species)) +
  geom_point(size = 2) +
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_df, width = 0.1) +
  labs(x = "Species", y = "Least square mean of sepal length (cm)") +
  ylim(0,8) +
  scale_color_brewer(palette = "Dark2") +
  guides(colour=FALSE) + #removes the redundant legend, the legend is being set by the colour = Species
  theme_classic(base_size = 14)
```

Something that is becoming increasingly popular with some journals is to show summary statisitics along with the raw data. This is a helpful opportunity to show how we can incorporate multiple datasets into one figure. We do this by makaing the exact same plot but adding another geom with its own specific dataset and aes() values.

```
ggplot(iris_ls_df, aes(x = Species, y = emmean, colour = Species)) +
  geom_point(size = 2) +
  geom_point(aes(x= Species, y = Petal.Length), iris_data, alpha = 0.3, size =0.75, position = "jitter")
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_df, width = 0.1) +
  labs(x = "Species", y = "Least square mean of petal length (cm)") +
  ylim(0,8) +
  scale_color_brewer(palette = "Dark2") +
  guides(colour=FALSE) +
  theme_classic(base_size = 14)
```

Many journals will charge extra to print in color. The best way to get around this is by changing the `scale_color_brewer` to `scale_color_grey`. Also note, that the start and end arguments refer to the range of how shaded they are, where 1=white and 0=black. Because we're showing this on a white background, it's best to not go with 1, but a smaller range. You'll probably have to play with the start and end depending on how many levels you have.

```
ggplot(iris_ls_df, aes(x = Species, y = emmean, colour = Species)) +
  geom_point(size = 2) +
  geom_point(aes(x= Species, y = Petal.Length), iris_data, alpha = 0.3, size =0.75, position = "jitter")
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_df, width = 0.1) +
  labs(x = "Species", y = "Least square mean of petal length (cm)") +
  ylim(0,8) +
  scale_color_grey(start=.5, end=0.2) + # see change here
  guides(colour=FALSE) +
  theme_classic(base_size = 14)
```

One final note. Let's say you didn't la=ike the names of the x-axis labels. You could change them using `scale_x_discrete()`. The reason for this is because we are using discrete (categorical) variables and breaking our data by species using the `colour = Species` command in out aesthetic (`aes()`). But because we're changing the x-scale (read x-axis) we need to use the `scale_x_discrete()` as opposed to the `scale_colour_brewer()`, which only would change the legend labels.

```
ggplot(iris_ls_df, aes(x = Species, y = emmean, colour = Species)) +
  geom_point(size = 2) +
  geom_point(aes(x= Species, y = Petal.Length), iris_data, alpha = 0.3, size =0.75, position = "jitter")
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_df, width = 0.1) +
  labs(x = "Species", y = "Least square mean of petal length (cm)") +
  ylim(0,8) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_discrete(breaks = c("setosa", "versicolor", "virginica" ), # takes your original labels and s
                   labels = c("I. setosa", "I. versicolor", "I. virginica" )) + # tells the function wh
```

```
  guides(colour=FALSE) +
  theme_classic(base_size = 14)
```

Graphing interactions with multiple categorical variables.

Let's say we're interested in plotting the interaction of multiple categorical variables. In this dataset we have only one categorical variable. But we can simply add another created from made up data. We will add another column we generate with made up site names for a categorical variable with three levels.

```
vec <- c("Gaspe", "Bonaventure", "Perce") # create a vector with three site names from where these were


Site <- sample(vec, 150, TRUE)  # randomly sample from the "vec" vector 150 times so we have 150 datapo

iris_data_sites <- cbind(iris_data, Site) #cbind just means column bind, it's a way to stick two datase

View(iris_data_sites)
```

Now let's calculate the least squares for our new data and our new categorical variables

```
iris_int_reg <- lm(Petal.Length ~ Species + Site, data = iris_data_sites) #agian, set up your model

emmeans(iris_int_reg, ~ Site + Species) #calculate least squares for Site and Species

iris_ls_int <- data.frame(emmeans(iris_int_reg, ~ Site + Species)) # coerce into a dataframe
```

Let's plot the interaction of these two factors. Because we're comparing two factors, we'll have to make two different plots. Lets first set it up by breaking down by ***site differences for each species***. Also, note that in place of colour, we are now using fill because barplots are filled as opposed to given color. A weird ggplot idiosyncracy. Specifying colour in this instance would give the lines around the boxes colours.

```
ggplot(iris_ls_int, aes(x= Species, y = emmean, fill = Site)) +
  geom_bar(stat = 'identity', position = "dodge") + # identity here just means plot the value given in
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_int, position = position_dodge(widt
  labs(x = "Species", y = "Least square mean of petal length (cm)") +
  ylim(0,8) +
  scale_fill_brewer(palette = "Dark2") + # see change here
  theme_classic(base_size = 14)
```

Now because we are interested in what best way to display the interaction, let's switch the `x =` and `fill =` variable to change this figure around so it's ***species differences at each site***.

```
ggplot(iris_ls_int, aes(x= Site, y = emmean, fill = Species)) +
  geom_bar(stat = 'identity', position = "dodge") +
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_int, position = position_dodge(widt
  labs(x = "Site", y = "Least square mean of petal length (cm)") +
  ylim(0,8) +
  scale_fill_brewer(palette = "Dark2") +
  theme_classic(base_size = 14)
```

And if you wanted to plot in grey change the code to be:

```
ggplot(iris_ls_int, aes(x= Site, y = emmean, fill = Species)) +
  geom_bar(stat = 'identity', position = "dodge") +
  geom_errorbar(aes(ymin =emmean - SE, ymax = emmean + SE), iris_ls_int, position = position_dodge(widt
  labs(x = "Site", y = "Least square mean of petal length (cm)") +
```

```
  ylim(0,8) +
  scale_fill_grey(start = 0.7, end = 0.1) + # notice the changes we made here from the last grey plot
  theme_classic(base_size = 14)
```

Final thoughts

This code is a rough framework that you should be able to play around with to creat other similar graphs. You could start by changing the x and y variable to see how those relationships are similar and different. Also, I would recommend trying other geoms. Perhaps try and take that boxplot we created early on and make a more visually appealing version. A lot of that basic information for changing plots is on the ggplot website I listed at the start of the exercise.