Post 02: More *Grammar of Graphics* Based Visualizations

Monica Wilkinson

11/30/2017

```
# pkgs <- c("ggplot2", "ggalt", "ggfortify")
# install.packages(pkgs)

library(ggplot2)
library(ggalt)
library(ggfortify)
```

Introduction

In class, we learned the basics of the `ggplot2` package, but we were not able to cover the true scope of its capabilities. In this post, I aim to introduce further aspects of the `ggplot2` package.

More `ggplot2`

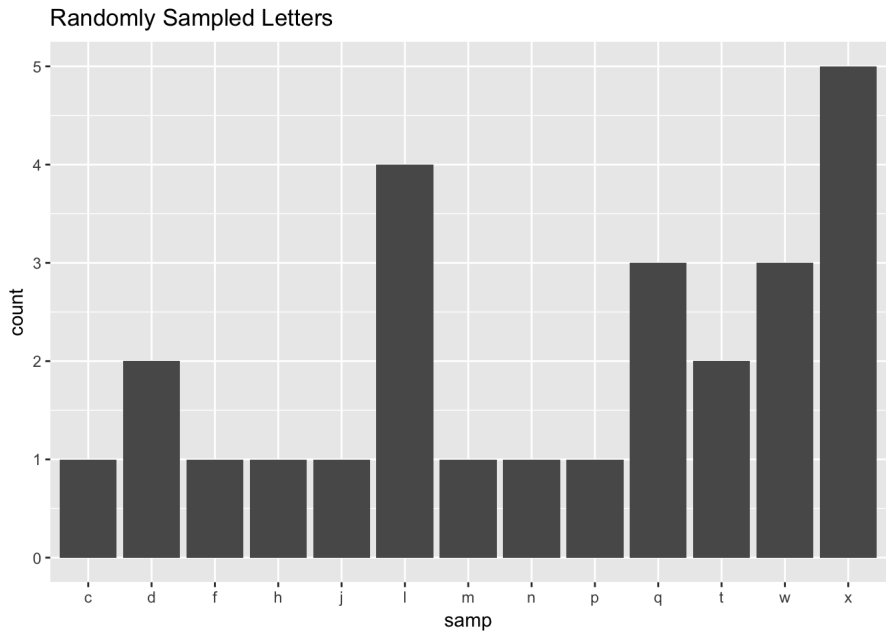
`qplot()`

While in class we refrained from using `qplot` because it lacks the full functionality of `ggplot`, it is still a useful tool that can be used in speedy or preliminary data exploration while still maintaining a similar aesthetic to `ggplot`.

One of the main differences between `qplot` and `ggplot` is that, rather than using the `+` operator to add layers to a plot, `qplot` takes all these added layers as arguments inside the function call. Additionally, `qplot` does not require that input data be in a data frame already.

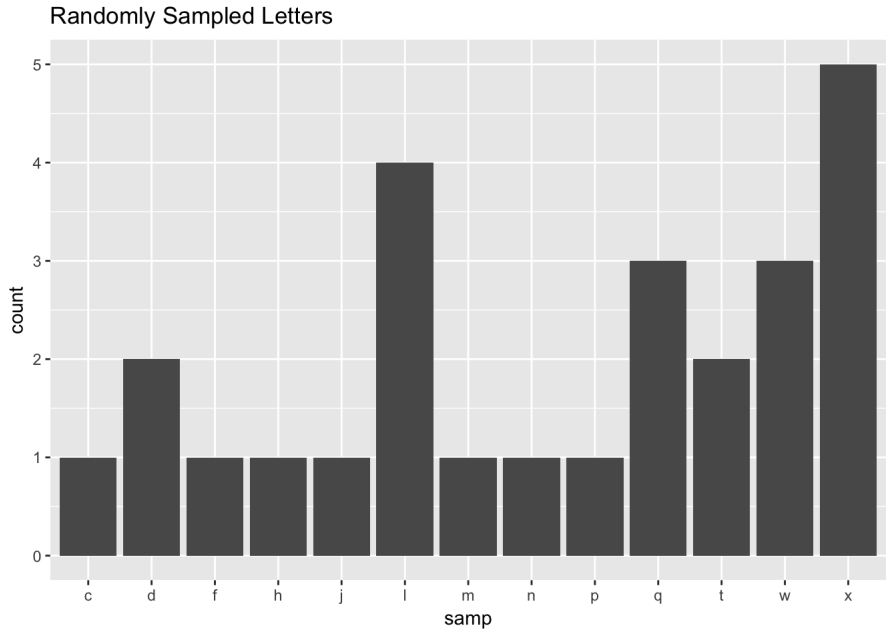
Here is an example

```
# Set seed for reproducibility
set.seed(133)

# Random sample of lowercase letters with replacement
samp <- sample(letters, 26, replace = TRUE)

# Quick Plot of our random sample
qplot(samp, geom = "bar", main = "Randomly Sampled Letters")
```



Note that this would be much more verbose using `ggplot`, and we would need to convert `samp` into a data frame. However, the resulting plot is visually identical.

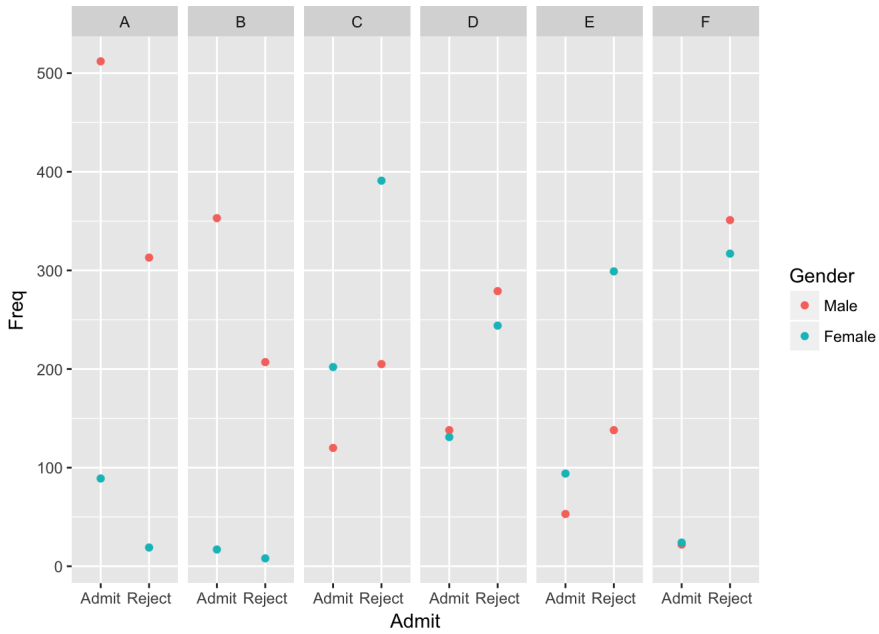
```
ggplot(data.frame(samp), aes(samp)) + geom_bar() + ggtitle("Randomly Sampled Letters")
```



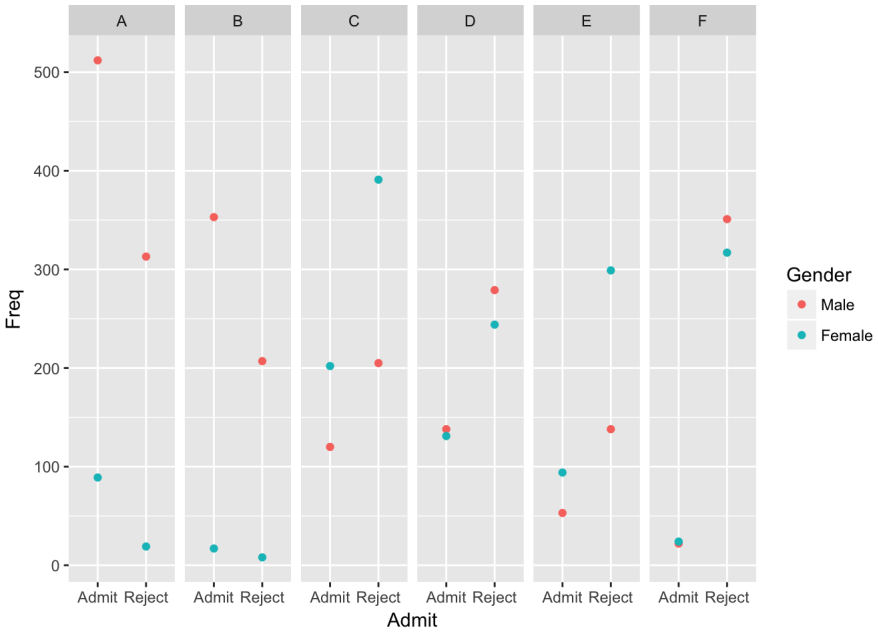
You can get a bit more complex with `qplot` by just passing more arguments into the function.

```
# Edit preloaded UCBAdmissions dataset for better fit in following graphs
UCBAdmissions <- data.frame(UCBAdmissions)
UCBAdmissions$Admit <- as.character(UCBAdmissions$Admit)
UCBAdmissions$Admit[UCBAdmissions$Admit == "Admitted"] <- "Admit"
UCBAdmissions$Admit[UCBAdmissions$Admit == "Rejected"] <- "Reject"
UCBAdmissions$Admit <- as.factor(UCBAdmissions$Admit)
```

```
# Not that qplot chooses between facet_wrap and facet_grid on its own.
# Character vectorassigned to geom is the same as the corresponding suffix of a
# geom_ function that would be used in ggplot.
qplot(Admit, Freq, data = UCBAdmissions, geom = "point", facets = . ~ Dept, color = Gender)
```




```
# This is equivalent to the following implementation of ggplot:
ggplot(UCBAdmissions, aes(x = Admit, y = Freq)) +
  geom_point(aes(color = Gender)) +
  facet_grid(. ~ Dept)
```



You can see how `qplot` can simplify the layering process of `ggplot`.

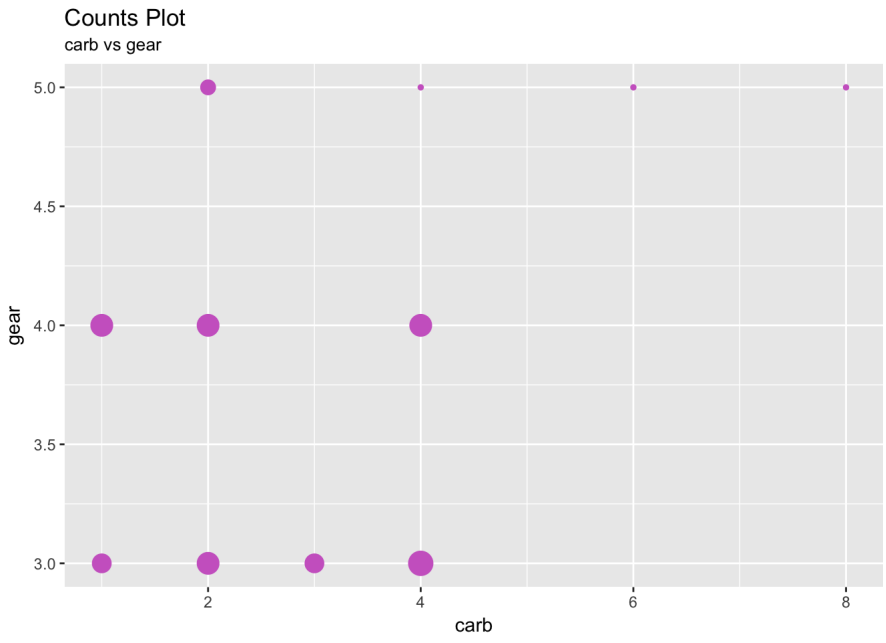
Other `geom` functions

In class, we only covered basic `geom` functions, such as `geom_bar`, `geom_hist`, `geom_point`, `geom_density`, and `geom_abline`. However, there are actually many more than this. Here are some examples of other graphs that are possible using `ggplot2`.

`geom_count`

This function is similar to `geom_point`, but overlapping points create larger "bubbles" rather than simply laying on top of each other. It is similar to using `geom_point` with the `alpha` or opacity set below 1, but it is much better at illustrating overlapping points if different points on your graph have widely ranging numbers of overlapping points.

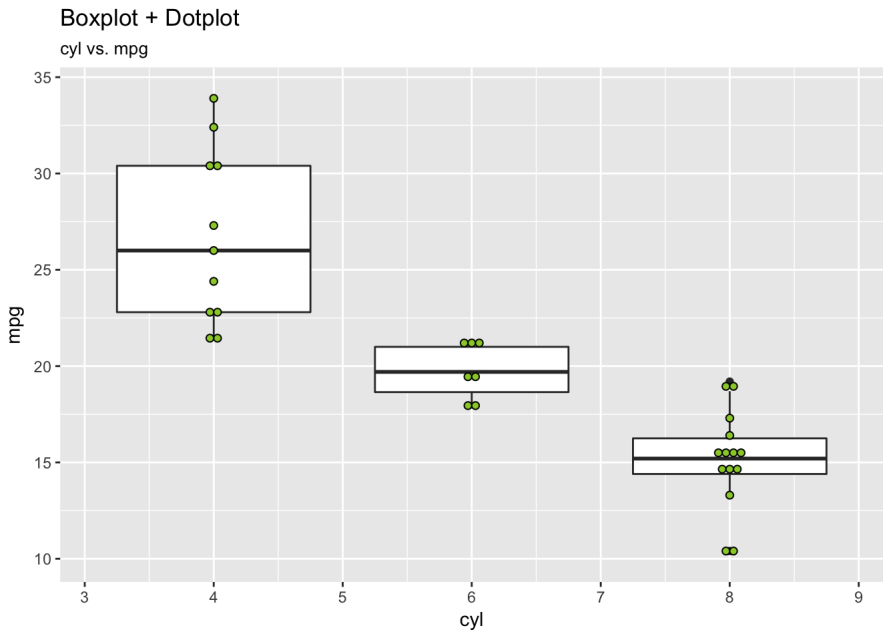
```
ggplot(mtcars, aes(carb, gear)) +
  geom_count(col="orchid3", show.legend=F) +
  labs(title="Counts Plot",
       subtitle = "carb vs gear")
```



`geom_boxplot`

While boxplots are very easy to understand and use, we can also add add a dotplot to show the distribution of points.

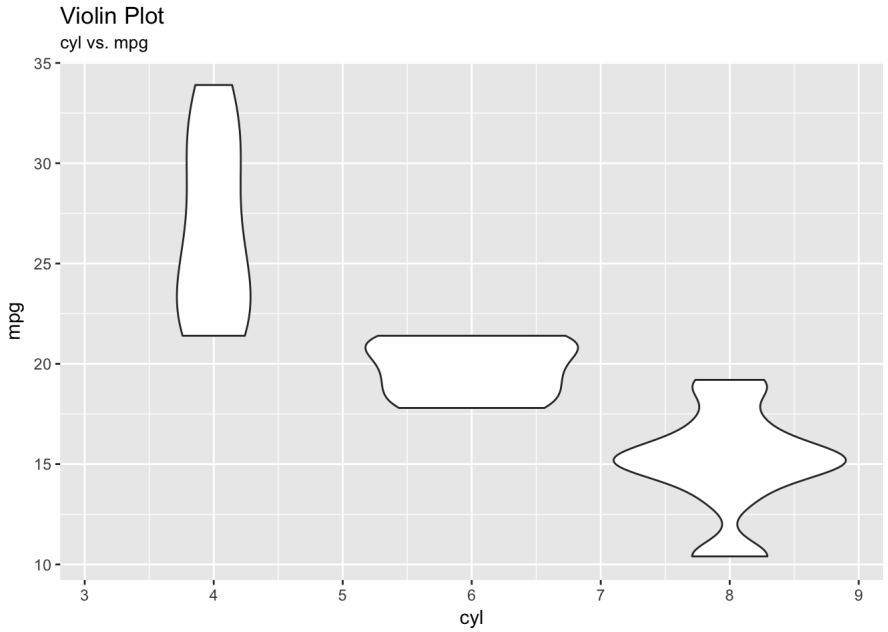
```
ggplot(mtcars, aes(cyl, mpg, group = cyl)) +
  geom_boxplot() +
  geom_dotplot(binaxis='y',
               stackdir='center',
               dotsize = .5,
               fill="olivedrab3") +
  labs(title = "Boxplot + Dotplot",
       subtitle = "cyl vs. mpg")
```



`geom_violin`

This function is very similar in usage and in information to `geom_boxplot`, but it more elegantly illustrates the density of points

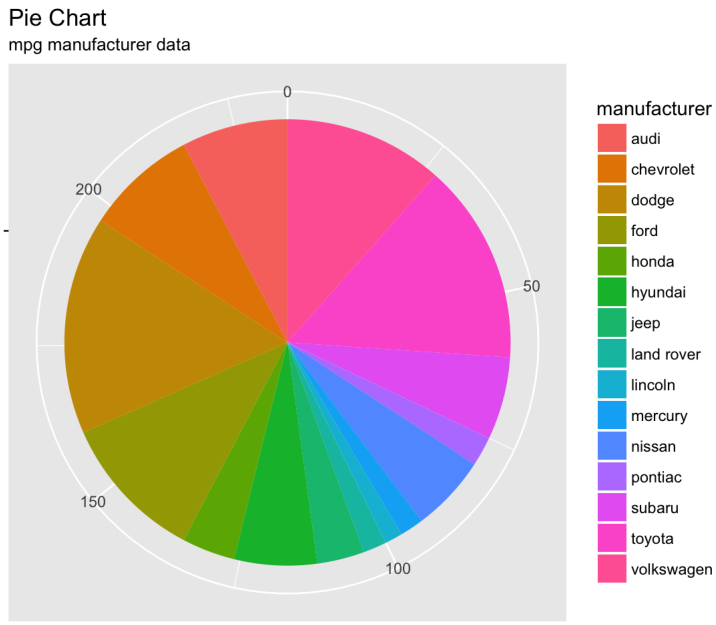
```
ggplot(mtcars, aes(cyl, mpg, group = cyl)) +
  geom_violin() +
  labs(title = "Violin Plot",
       subtitle = "cyl vs. mpg")
```



Pie Chart

While `ggplot2` does not have its own function to create pie charts, you can easily create a pie chart by creating a bar chart and adding a `coord_polar()` layer.

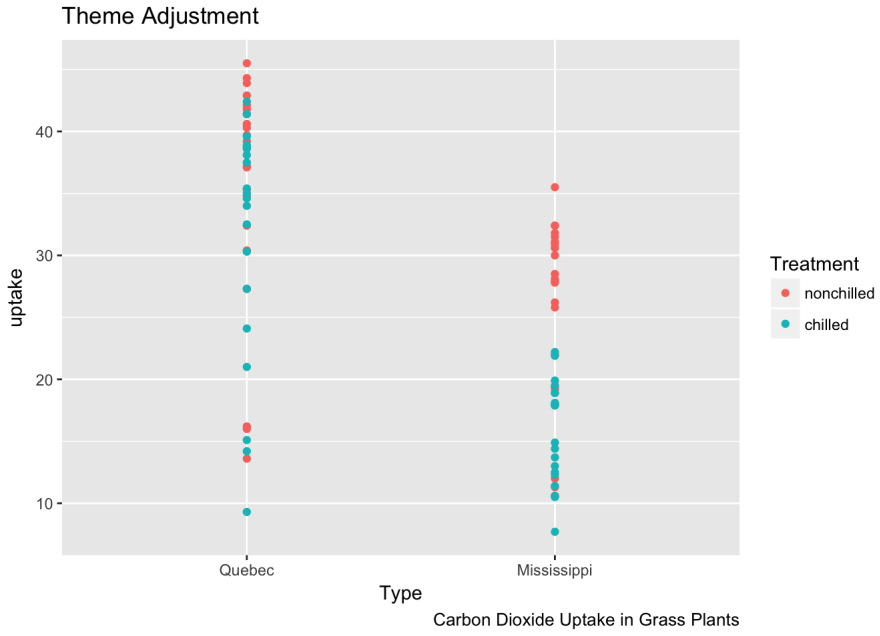
```
ggplot(mpg, aes(x = "", fill = factor(manufacturer))) +
  geom_bar(width = 1) +
  labs(title = "Pie Chart",
       subtitle = "mpg manufacturer data",
       fill = "manufacturer",
       x = NULL, y = NULL) +
  coord_polar(theta = "y")
```



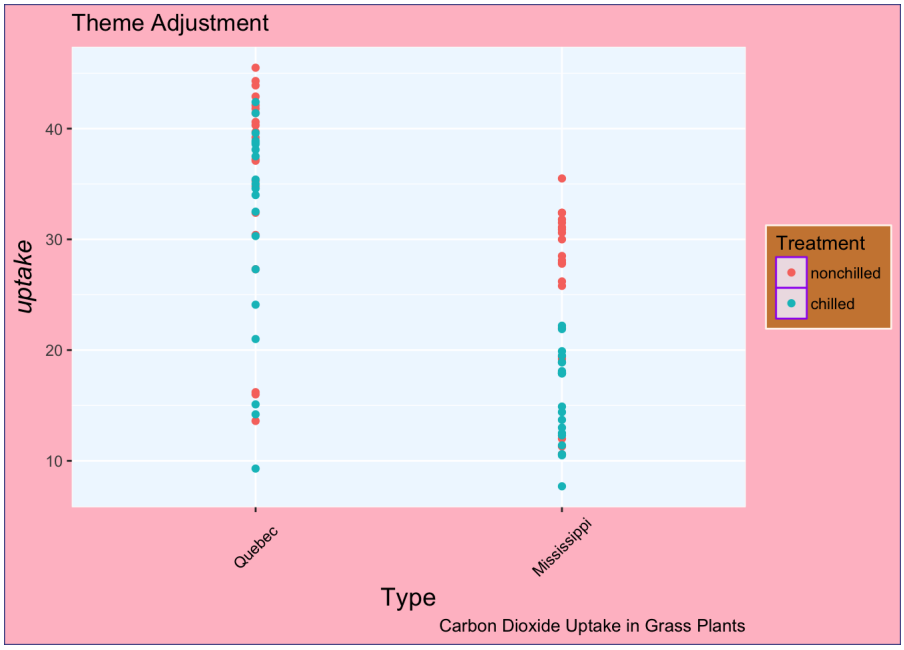
`theme`

The `theme` function is an additional layer that can be added to a ggplot to modify non-data elements of the plot's appearance. These elements and functions that can be passed into `theme()` can modify plot elements such as text font size, plot background color, axis ticks, legend position, plot margins, etc.

```
# base plot, no theme change
plot <- ggplot(CO2, aes(Type, uptake)) +
  geom_point(aes(color = Treatment)) +
  labs(title = "Theme Adjustment",
       caption = "Carbon Dioxide Uptake in Grass Plants")
plot
```

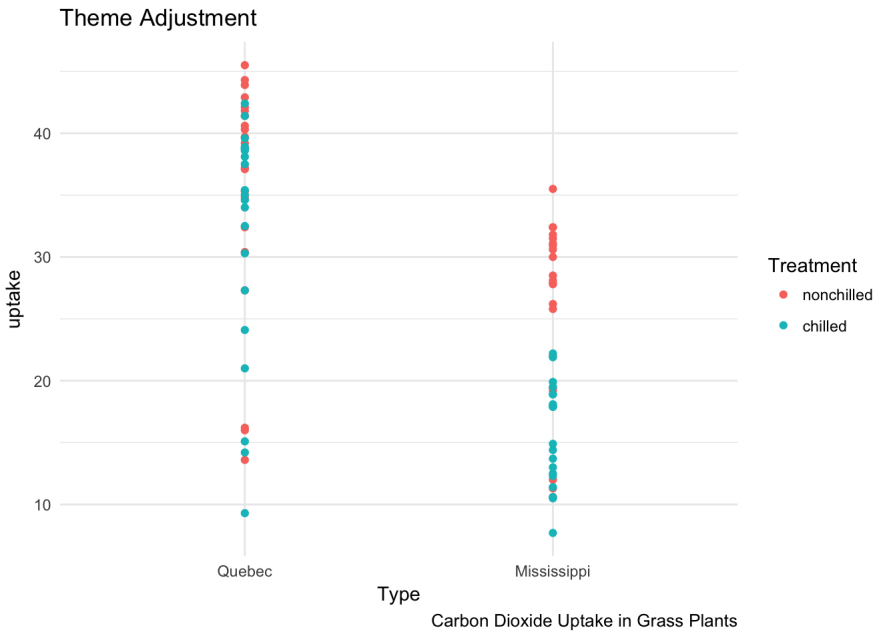



```
plot +
  theme(axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14, face = 'italic'),
        axis.text.x = element_text(color = "grey0",
                                   angle = 45,
                                   vjust = 0.5),
        panel.background = element_rect(color = "grey99", fill = "aliceblue"),
        plot.background = element_rect(color = "royalblue4", fill = "pink"),
        legend.background = element_rect(color = "seashell", fill = "peru"),
        legend.key = element_rect(color = "purple", fill = "lavenderblush2"))
```



As you can see, you can really go crazy with these theme elements! Alternatively, ggplot also provides a few preset themes that can simply be added as another layer to a plot. If you find this approach to be easier, there are also many more preset themes available in the `ggthemes` package.

```
plot +
  theme_minimal()
```



Conclusion

While I have introduced a few other aspects of `ggplot2`, these examples have still been relatively simple. There are a wide variety of other packages designed to work in conjunction with `ggplot2` to provide even more functionality to the package. This allows users to create more and more different types of graphs using different types, and to think of and present data in new, thought-provoking ways.

Resources

1. http://r-statistics.co/ggplot2-Tutorial-With-R.html
2. http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html
3. http://ggplot2.org/book/
4. https://stackoverflow.com/questions/5322836/choosing-between-qplot-and-ggplot-in-ggplot2
5. https://github.com/tidyverse/ggplot2/wiki/Graph-Panel-Attributes
6. https://github.com/tidyverse/ggplot2/wiki/Axis-Attributes
7. http://sape.inf.usi.ch/quick-reference/ggplot2/colour

This file was created using:

R version 3.4.1 (2017-06-30) – “Single Candle” Copyright (C) 2017 The R Foundation for Statistical Computing Platform: x86_64-apple-darwin15.6.0 (64-bit)

RStudio Version 1.0.153 – © 2009-2017 RStudio, Inc. Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit/604.3.5 (KHTML, like Gecko)