

Post01: Data Visualization with ggplot2

Stat 133, Fall 2017

Abigail Stiris

Data Visualization with ggplot2

Introduction

Imagine this: You're presenting your extensive data analysis to your colleagues. You've never been more proud of a project. You load up your slideshow, tell your story, and conclude with a beautiful graph. It's basically artwork at this point. But then you see squints, head scratches, and confused looks. It's all gone downhill.

Sometimes, a clear visualization of even the most insignificant findings can make them stand out and seem more significant. On the other hand, a dull, obscure graph will make an analytical breakthrough seem mundane and unoriginal. That's why visualization methods are just as important as data wrangling, manipulation, and analysis.

Let's take a closer look at how `ggplot2` can enable you to tell impactful stories through your data analysis.

Motivation

When we discussed `ggplot2` in lecture and lab, we learned the basic functions, but somehow Professor Sanchez's graphs just always looked cleaner, easier to read, and more beautiful. I'm setting out to unlock these `ggplot2` hacks so that we Stat 133 students can all achieve new heights of data visualization and be more fluent in the once mysterious package.

Background

The package `ggplot2` was created by Hadley Wickham in 2005. It was designed to be based on the "grammar of graphics (gg)" approach, which takes care of the tedious tasks involved in creating graphs, like legends and labels. Since then, it's grown to be one of the [Top 5](#) most popular R packages, with around 70,000 monthly [direct downloads](#).

Clearly, there's something really significantly useful about this package. Not only does it have an easy-to-understand language, but it's flexible and allows for high levels of abstraction. It does all the hard work for you. Now that we have a solid grasp on the basic features of the package, let's take a closer look beyond the functions and parameters we've already used.

Some Features

`ggplot2` has an extensive [documentation](#). From themes to layers to coordinate systems, the package allows for a lot of flexibility when it comes to visualizing data. Here are some interesting features you probably don't know about.

Customization

In `ggplot2`, graphs can be customized in an endless number of ways to work best for your purposes. The package has a lot to offer in terms of visuals, so it's no wonder why there's a separate feature called `aes`, where you can map data to aesthetics like color, fill, or shape.

Note: Setting vs. Mapping

The difference between setting and mapping aesthetics is whether you're looking to have `ggplot2` categorize your graph in terms of a certain variable, or whether you want to customize the entire output. For example, when we use `geom_point(aes(color = some_variable))`, `ggplot2` will colorize the points on your graph with as many different colors as there are unique values of that variable. But when we instead use `geom_point(color = "red")`, all of the points on the plot will be red.

The following examples will be using some of the NBA data that we've seen in class.

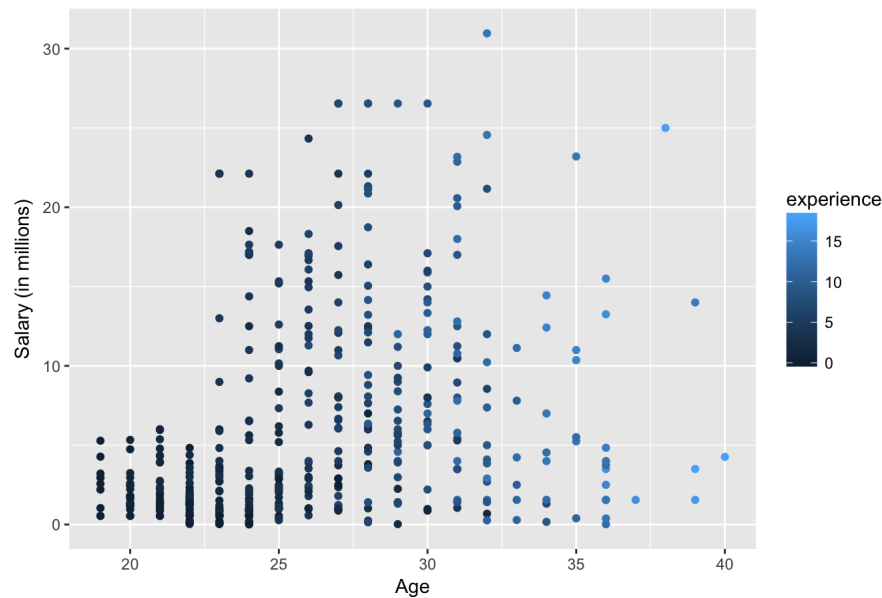
```
# loading packages
library(ggplot2)
library(dplyr)

# reading input file
roster <- read.csv('../data/nba2017-roster.csv', stringsAsFactors = FALSE)
```

After loading the packages and reading the file 'nba2017-roster.csv', we can create plots.

```
ggplot(data = roster, aes(x = age, y = salary/1000000)) +
  geom_point(aes(color = experience)) +
  # mapping experience to color
  ggtitle("Scatterplot of Age and Salary") +
  xlab("Age") +
  ylab("Salary (in millions)")
```

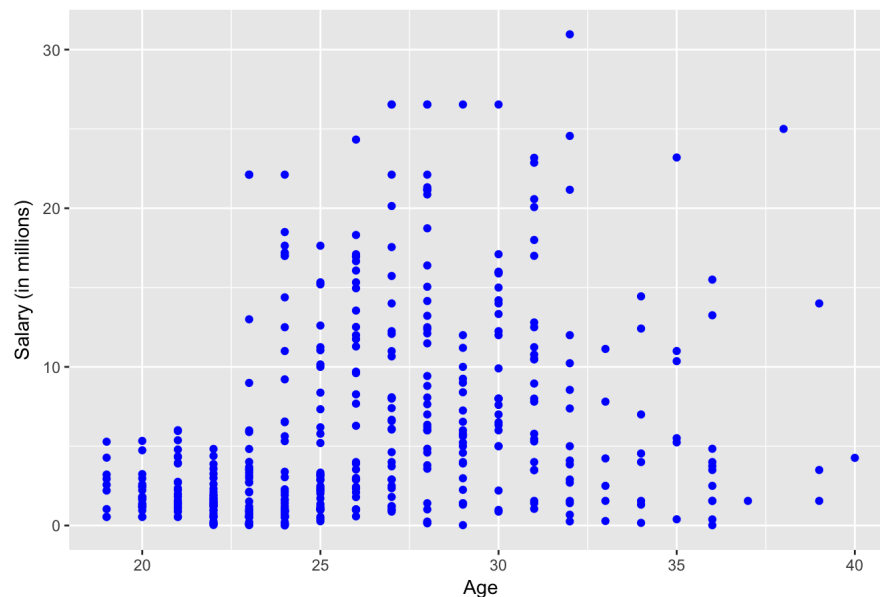
Scatterplot of Age and Salary



Here we see that `ggplot2` creates a legend with the `experience` values in order to color the points on the scatterplot, and we see that the highest values are usually associated with age (which is self-evident) and salary. If instead, we want to color all of the points blue, here's how the code would change.

```
ggplot(data = roster, aes(x = age, y = salary/1000000)) +
  geom_point(color = "blue") +
  # setting color to blue
  ggtitle("Scatterplot of Age and Salary") +
  xlab("Age") +
  ylab("Salary (in millions)")
```

Scatterplot of Age and Salary



Let's move on to new aesthetic features.

Aesthetics: Color

Colors in `ggplot2` can be specified with strings such as "red", RGB hexadecimal digits, or NA for a transparent color. Even without using RGB digits, the options when it comes to strings are endless. There's "turquoise2" and "mediumorchid" and even "maroon4" ! (Sadly, no "maroon5" yet though.)

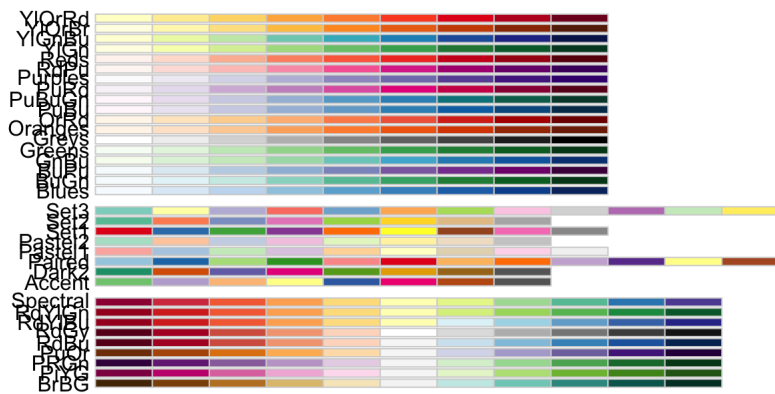
cornsilk3	dodgerblue4	gray45	gray3	gray69	lemonchiffon2	mediumorchid	palevioletred4	slateblue	
cornsilk2	dodgerblue3	gray44	gray2	gray68	lemonchiffon1	mediumblue	palevioletred3	skyblue4	
cornsilk1	dodgerblue2	gray43	gray1	gray67	lemonchiffon	mediumaquamarine	palevioletred2	skyblue3	
cornsilk	dodgerblue1	gray42	gray0	gray66	lawngreen	maroon4	palevioletred1	skyblue2	yellowgreen
cornflowerblue	dodgerblue	gray41	gray	gray65	lavenderblush4	maroon3	palevioletred	skyblue1	yellow4
coral4	dimgray	gray40	greenyellow	gray64	lavenderblush3	maroon2	paleturquoise4	skyblue	yellow3
coral3	dimgray	gray39	green4	gray63	lavenderblush2	maroon1	paleturquoise3	sienna4	yellow2
coral2	deepskyblue4	gray38	green3	gray62	lavenderblush1	maroon	paleturquoise2	sienna3	yellow1
coral1	deepskyblue3	gray37	green2	gray61	lavenderblush	magenta4	paleturquoise1	sienna2	yellow
coral	deepskyblue2	gray36	green1	gray60	lavender	magenta3	paleturquoise	sienna1	whitesmoke
chocolate4	deepskyblue1	gray35	green	gray59	khaki4	magenta2	palegreen4	sienna	wheat4
chocolate3	deepskyblue	gray34	gray100	gray58	khaki3	magenta1	palegreen3	seashell4	wheat3
chocolate2	deeppink4	gray33	gray99	gray57	khaki2	magenta	palegreen2	seashell3	wheat2
chocolate1	deeppink3	gray32	gray98	gray56	khaki1	linen	palegreen1	seashell2	wheat1
chocolate	deeppink2	gray31	gray97	gray55	khaki	limegreen	palegreen	seashell1	wheat
chartreuse4	deeppink1	gray30	gray96	gray54	ivory4	lightyellow4	palegoldenrod	seashell	violetred4
chartreuse3	deeppink	gray29	gray95	gray53	ivory3	lightyellow3	orchid4	seagreen4	violetred3
chartreuse2	darkviolet	gray28	gray94	gray52	ivory2	lightyellow2	orchid3	seagreen3	violetred2
chartreuse1	darkturquoise	gray27	gray93	gray51	ivory1	lightyellow1	orchid2	seagreen2	violetred1
chartreuse	darkslategrey	gray26	gray92	gray50	ivory	lightyellow	orchid1	seagreen1	violetred
cadetblue4	darkslategray4	gray25	gray91	gray49	indianred4	lightsteelblue4	orchid	seagreen	violet
cadetblue3	darkslategray3	gray24	gray90	gray48	indianred3	lightsteelblue3	orangered4	sandybrown	turquoise4
cadetblue2	darkslategray2	gray23	gray89	gray47	indianred2	lightsteelblue2	orangered3	salmon4	turquoise3
cadetblue1	darkslategray1	gray22	gray88	gray46	indianred1	lightsteelblue1	orangered2	salmon3	turquoise2
cadetblue	darkslategray	gray21	gray87	gray45	indianred	lightsteelblue	orangered1	salmon2	turquoise1
burlywood4	darkslateblue	gray20	gray86	gray44	holupink4	lightslategray	orangered	salmon1	turquoise
burlywood3	darkseagreen4	gray19	gray85	gray43	holupink3	lightslategray	orange4	salmon	tomato4
burlywood2	darkseagreen3	gray18	gray84	gray42	holupink2	lightslateblue	orange3	saddlebrown	tomato3
burlywood1	darkseagreen2	gray17	gray83	gray41	holupink1	lightskyblue4	orange2	royalblue4	tomato2
burlywood	darkseagreen1	gray16	gray82	gray40	holupink	lightskyblue3	orange1	royalblue3	tomato1
brown4	darkseagreen	gray15	gray81	gray39	honeydew4	lightskyblue2	orange	royalblue2	tomato
brown3	darksalmon	gray14	gray80	gray38	honeydew3	lightskyblue1	olivedrab4	royalblue1	thistle4
brown2	darkred	gray13	gray79	gray37	honeydew2	lightskyblue	olivedrab3	royalblue	thistle3
brown1	darkorchid4	gray12	gray78	gray36	honeydew1	lightseagreen	olivedrab2	rosybrown4	thistle2
brown	darkorchid3	gray11	gray77	gray35	honeydew	lightsalmon4	olivedrab1	rosybrown3	thistle1
blueviolet	darkorchid2	gray10	gray76	gray34	gray100	lightsalmon3	olivedrab	rosybrown2	thistle
blue4	darkorchid1	gray9	gray75	gray33	gray99	lightsalmon2	oldlace	rosybrown1	tan4
blue3	darkorchid	gray8	gray74	gray32	gray98	lightsalmon1	navyblue	rosybrown	tan3
blue2	darkorange4	gray7	gray73	gray31	gray97	lightsalmon	navy	red4	tan2
blue1	darkorange3	gray6	gray72	gray30	gray96	lightpink4	navajowhite4	red3	tan1
blue	darkorange2	gray5	gray71	gray29	gray95	lightpink3	navajowhite3	red2	tan
tanchedalmond	darkorange1	gray4	gray70	gray28	gray94	lightpink2	navajowhite2	red1	steelblue4
black	darkorange	gray3	gray69	gray27	gray93	lightpink1	navajowhite1	red	steelblue3
bisque4	darkolivegreen4	gray2	gray68	gray26	gray92	lightpink	navajowhite	purple4	steelblue2
bisque3	darkolivegreen3	gray1	gray67	gray25	gray91	lightgrey	moccasin	purple3	steelblue1
bisque2	darkolivegreen2	gray0	gray66	gray24	gray90	lightgreen	mistyrose4	purple2	steelblue
bisque1	darkolivegreen1	gray	gray65	gray23	gray89	lightgray	mistyrose3	purple1	springgreen4
bisque	darkolivegreen	goldenrod4	gray64	gray22	gray88	lightgoldenrodyellow	mistyrose2	purple	springgreen3
beige	darkmagenta	goldenrod3	gray63	gray21	gray87	lightgoldenrod4	mistyrose1	powderblue	springgreen2
azure4	darkkhaki	goldenrod2	gray62	gray20	gray86	lightgoldenrod3	mistyrose	plum4	springgreen1
azure3	darkgray	goldenrod1	gray61	gray19	gray85	lightgoldenrod2	mintcream	plum3	springgreen
azure2	darkgreen	goldenrod	gray60	gray18	gray84	lightgoldenrod1	midnightblue	plum2	snow4
azure1	darkgray	gold4	gray59	gray17	gray83	lightgoldenrod	mediumvioletred	plum1	snow3
azure	darkgoldenrod4	gold3	gray58	gray16	gray82	lightcyan4	mediumturquoise	plum	snow2
aquamarine4	darkgoldenrod3	gold2	gray57	gray15	gray81	lightcyan3	mediumspringgreen	pink4	snow1
aquamarine3	darkgoldenrod2	gold1	gray56	gray14	gray80	lightcyan2	mediumslateblue	pink3	snow
aquamarine2	darkgoldenrod1	gold	gray55	gray13	gray79	lightcyan1	mediumseagreen	pink2	slategrey
aquamarine1	darkgoldenrod	ghostwhite	gray54	gray12	gray78	lightcyan	mediumpurple4	pink1	slategray4
aquamarine	darkcyan	gainsboro	gray53	gray11	gray77	lightcoral	mediumpurple3	pink	slategray3
antiquewhite4	darkblue	forestgreen	gray52	gray10	gray76	lightblue4	mediumpurple2	peru	slategray2
antiquewhite3	cyan4	floralwhite	gray51	gray9	gray75	lightblue3	mediumpurple1	peachpuff4	slategray1
antiquewhite2	cyan3	firebrick4	gray50	gray8	gray74	lightblue2	mediumpurple	peachpuff3	slategray
antiquewhite1	cyan2	firebrick3	gray49	gray7	gray73	lightblue1	mediumorchid4	peachpuff2	slateblue4
antiquewhite	cyan1	firebrick2	gray48	gray6	gray72	lightblue	mediumorchid3	peachpuff1	slateblue3
aliceblue	cyan	firebrick1	gray47	gray5	gray71	lemonchiffon4	mediumorchid2	peachpuff	slateblue2
white	cornsilk4	firebrick	gray46	gray4	gray70	lemonchiffon3	mediumorchid1	papayawhip	slateblue1

List of ggplot2 color names

There's also the option of using **Color Brewer**.

Color Brewer was briefly mentioned in lecture, but let's find out exactly how it can be used to take your graphs to the next level. The package **RColorBrewer** provides an extensive selection of "palettes" that can be used with `ggplot2` graphics. These can be accessed using the `display.brewer.all()` function.

```
library(RColorBrewer) # loading RColorBrewer package
display.brewer.all() # all RColorBrewer palettes
```

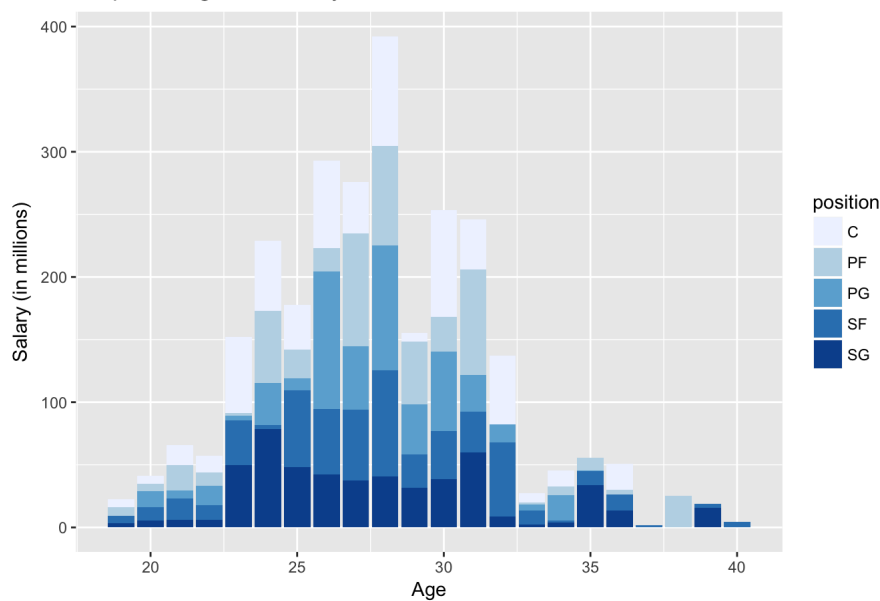


These palettes can be applied to various plots when it comes to fill or color. Not only is there the option to use numbers, but the `palette` parameter can also be a string. The default palette is a blue gradient.

Example 1: `scale_fill_brewer` – default palette

```
ggplot(data = roster, aes(x = age, y = salary/1000000,
                          fill = position)) +
  geom_bar(stat = "identity") +
  ggtitle("Barplot of Age and Salary: Default Palette") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  scale_fill_brewer() # default palette
```

Barplot of Age and Salary: Default Palette

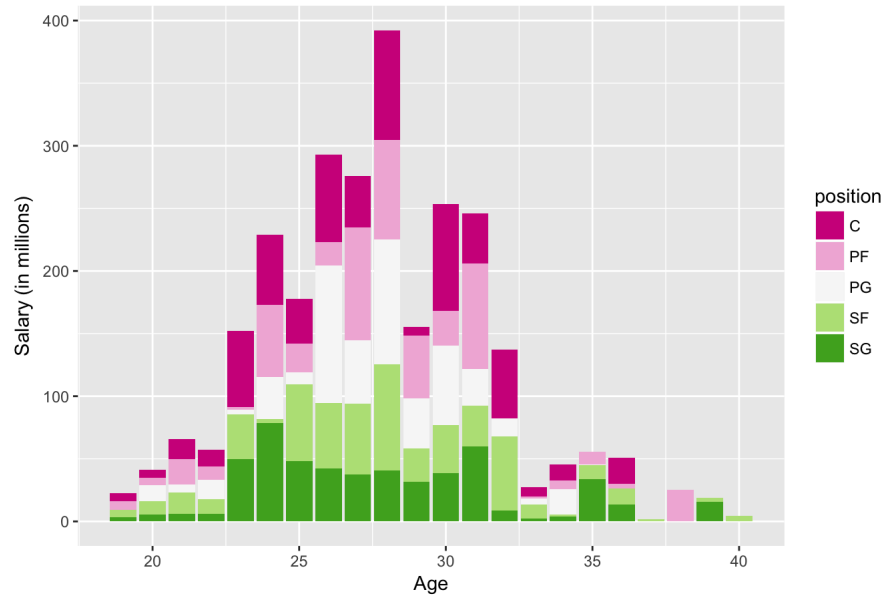


You can also choose one of the custom palettes from `RColorBrewer`, like "PiYG".

Example 2: `scale_fill_brewer` – custom palette

```
ggplot(data = roster, aes(x = age, y = salary/1000000,
                          fill = position)) +
  geom_bar(stat = "identity") +
  ggtitle("Barplot of Age and Salary: 'PiYG' Palette") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  scale_fill_brewer(palette = "PiYG") # "PiYG" palette
```

Barplot of Age and Salary: 'PiYG' Palette

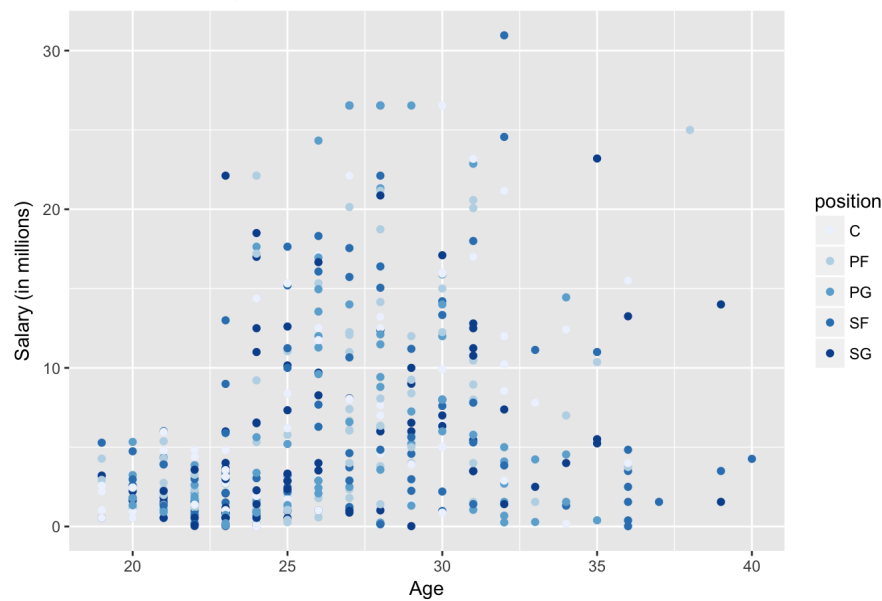


Another option is to use `scale_color_brewer`, which is especially useful for scatterplots. Here is the default blue gradient palette.

Example 3: `scale_color_brewer` – default palette

```
ggplot(data = roster, aes(x = age, y = salary/1000000,
                          color = position)) +
  geom_point() +
  ggtitle("Scatterplot of Age and Salary: Default Palette") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  scale_color_brewer() # default palette
```

Scatterplot of Age and Salary: Default Palette

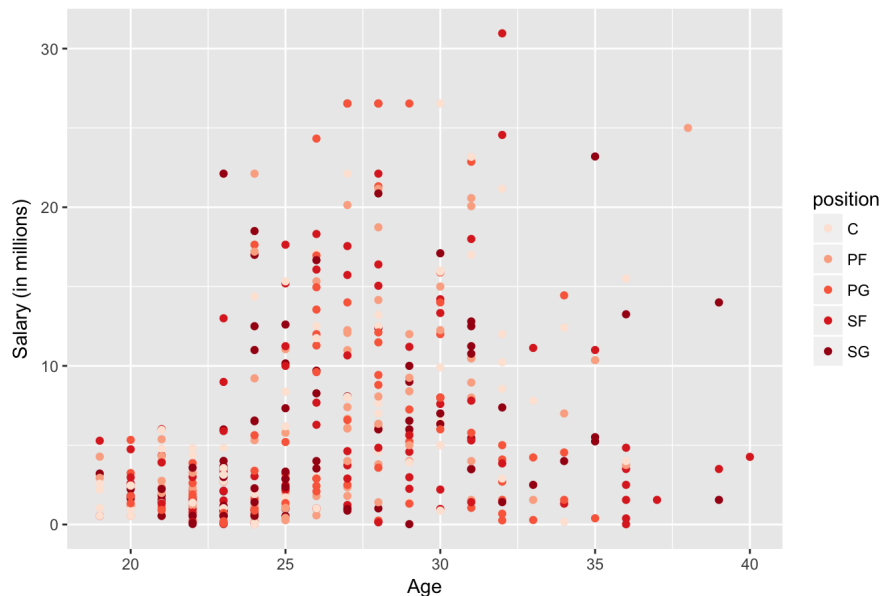


And here is a plot using the "Reds" palette.

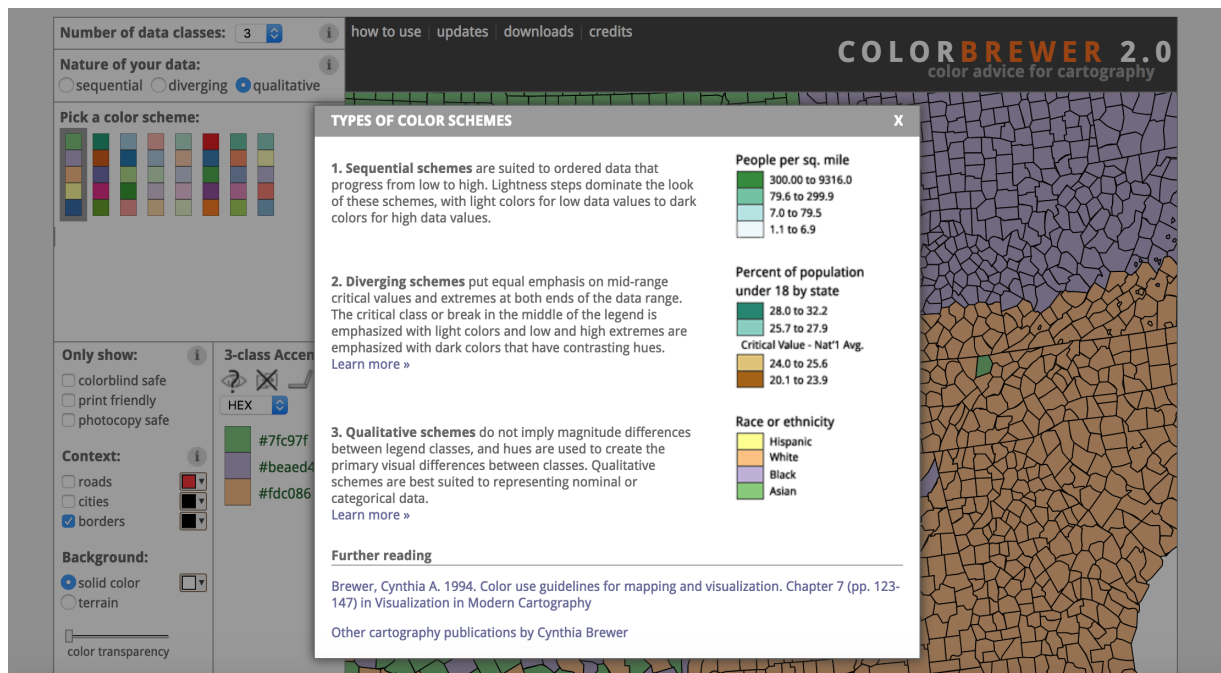
Example 4: `scale_color_brewer` – custom palette

```
ggplot(data = roster, aes(x = age, y = salary/1000000,
                          color = position)) +
  geom_point() +
  ggtitle("Scatterplot of Age and Salary: 'Reds' Palette") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  scale_color_brewer(palette = "Reds") # "Reds" palette
```

Scatterplot of Age and Salary: 'Reds' Palette



RColorBrewer is a great tool for customizing your graphs. There are a variety of options you can use, including ones that are colorblind safe, print friendly, and photocopy safe. The color schemes can be used with **sequential** schemes (with ordered data from low to high), **diverging** schemes (with mid-range critical values emphasized in light colors and extremes in dark colors), or **qualitative** schemes (with categorical data). More info can be found on the [website](#).



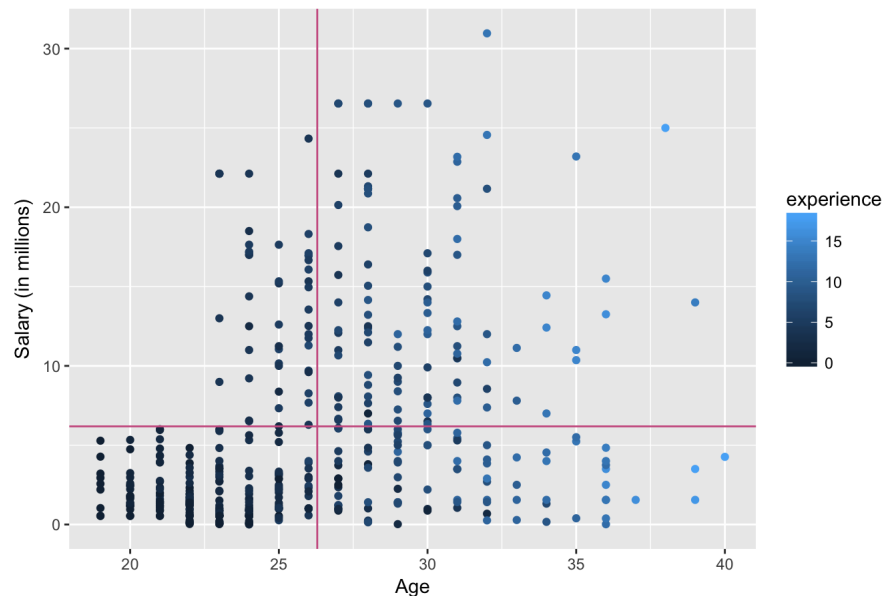
More Aesthetics: *Lines*

Lines in `ggplot2` are very useful for summarizing data and putting your findings into perspective. For example, just by adding the averages (in snazzy "hotpink3") to the same scatterplot above, you can get an idea of how the points compare. The function `geom_vline` can be used to add vertical lines by specifying the `xintercept`, and `geom_hline` is used for adding horizontal lines by specifying the `yintercept`.

You can also choose to map the intercepts to variables or to set them to a specific value. For these plots, I'll be using `aes` to map the intercepts to either the `age` or `salary` value.

```
ggplot(data = roster, aes(x = age, y = salary/1000000)) +
  geom_point(aes(color = experience)) +
  ggtitle("Scatterplot of Age and Salary") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  geom_vline(aes(xintercept = mean(age)),
    color = "hotpink3") +
  # vertical line
  geom_hline(aes(yintercept = mean(salary / 1000000)),
    color = "hotpink3") # horizontal line
```

Scatterplot of Age and Salary

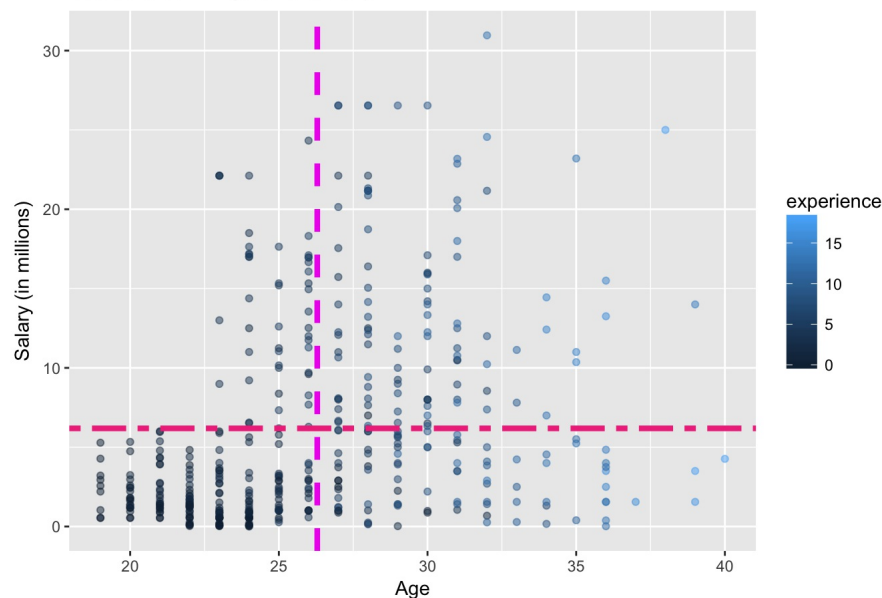


Sometimes, highlighting the x and y axes is more useful. In that case, you'll want to set your `xintercept` and `yintercept` values to 0 rather than mapping to a variable. Or, you can also control the slope and intercept of your graph using `geom_abline`.

To get different effects on your lines, you can change `linetype`, `size`, or `alpha` (transparency). Here are some of those features showcased on the previous plot. I'll set `linetype` to "dashed" on the vertical line and "twodash" on the horizontal line so that the two are more distinct.

```
ggplot(data = roster, aes(x = age, y = salary/1000000)) +
  geom_point(aes(color = experience), alpha = 0.5) +
  ggtitle("Scatterplot of Age and Salary") +
  xlab("Age") +
  ylab("Salary (in millions)") +
  geom_vline(aes(xintercept = mean(age)), color = "magenta2",
    linetype = "dashed", size = 1.5) +
  # vertical line
  geom_hline(aes(yintercept = mean(salary / 1000000)),
    color = "violetred2", linetype = "twodash",
    size = 1.5) # horizontal line
```

Scatterplot of Age and Salary



I've been using scatterplots and barplots to show off some of the different aesthetic elements, but now let's go beyond them to a few new types of visualization.

Plots

With `ggplot2`, a variety of different plots can be used to fit different types of data. The options go beyond the basics that we're familiar with and can even be used to make art (for advanced data scientists).

Violin Plot

We've used box plots as tools for visualizing the basic distributions of data, but they aren't as helpful for understanding the variation in the data. That limitation is taken care of with `geom_violin`. Let's look at an example using our NBA data.

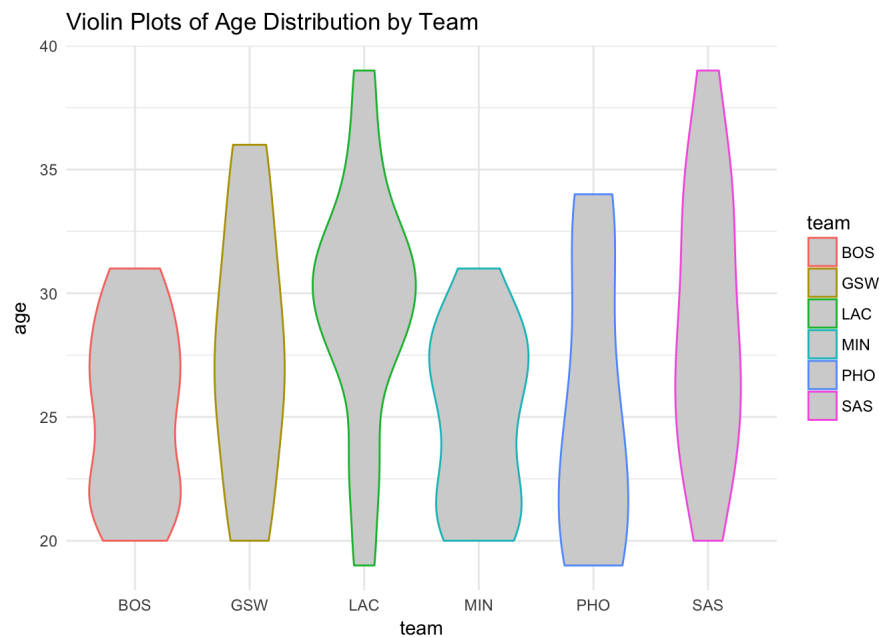
Example: geom_violin

Suppose we're interested in analyzing the distribution and variation of player age by team. I've created a `mini_roster` of only 6 teams, just so that the graph is easier to read.

```
mini_roster <- roster %>%  
  filter(team %in% c("GSW", "MIN", "SAS", "PHO", "LAC", "BOS"))
```

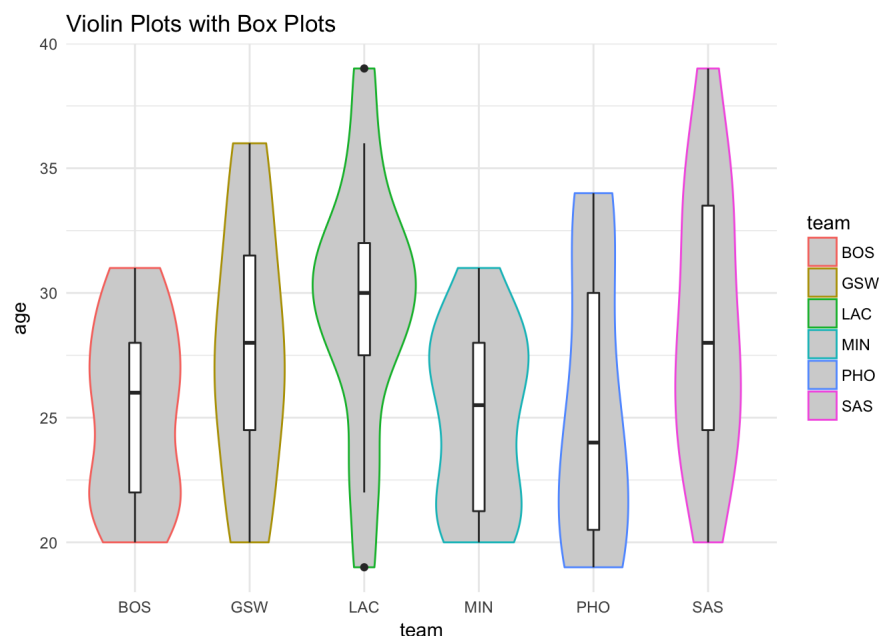
Violin plots are similar to boxplots, but also display the distribution of the data. Wider portions represent a higher probability, while narrower portions represent a lower probability. So, for example, an age of around 30 is most common on the LA Clippers (LAC) team.

```
# manipulating the roster to get age by team  
age_by_team <- mini_roster %>%  
  select(team, age) %>%  
  group_by(team)  
  
# creating the violin plot  
ggplot(data = age_by_team, aes(x = team, y = age)) +  
  geom_violin(aes(color = team), fill = "lightgray") +  
  theme_minimal() +  
  ggtitle("Violin Plots of Age Distribution by Team")
```



If we add a `boxplot` to the above graph, we can also see the median, range, and quartiles. Here we see how the violin plot goes beyond the boxplot by displaying not just the average distribution, but the probability that a value will fall within the range as well.

```
ggplot(data = age_by_team, aes(x = team, y = age)) +  
  geom_violin(aes(color = team), fill = "lightgray") +  
  theme_minimal() +  
  geom_boxplot(width = 0.1) +  
  ggtitle("Violin Plots with Box Plots")
```

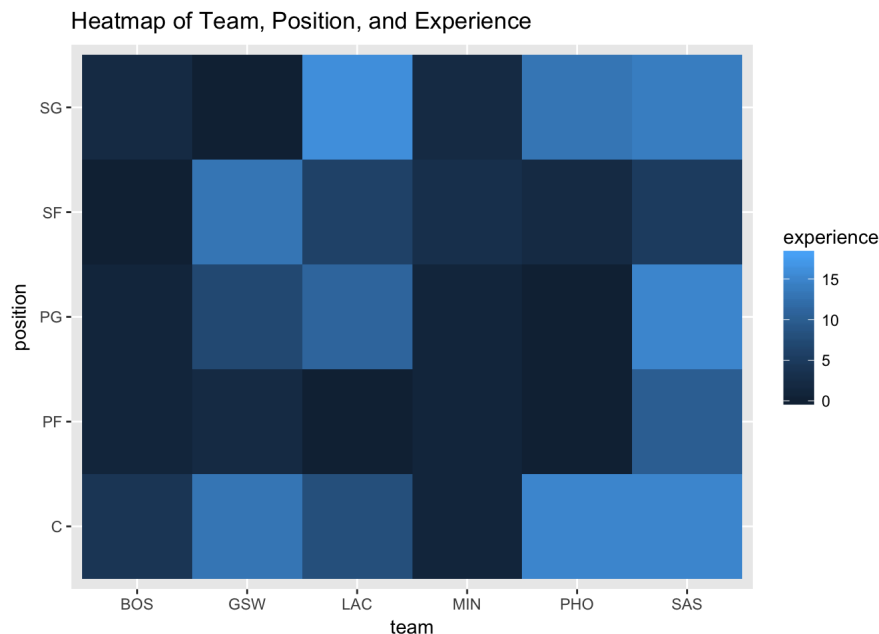


Heatmap

Using a heatmap, we can add another variable to our analysis of player age by team. In this case, we can create a heatmap using `geom_tile`.

```
# manipulating the roster to get position and experience
experience_by_position <- mini_roster %>%
  select(team, position, experience) %>%
  group_by(team)

# creating the heatmap
ggplot(data = experience_by_position, aes(x = team, y = position)) +
  geom_tile(aes(fill = experience)) +
  ggtitle("Heatmap of Team, Position, and Experience")
```



This is another way to visualize the data, where we can plot each team and each position, and see the squares with the lightest blue as an indicator of greatest years of experience in relation to each team's position.

Discussion

The package `ggplot2` is rich with all kinds of different features and parameters. While there is an infinite number of ways to visualize and interpret data, I hope this post will be a useful resource for understanding some of the basic and not-so-basic outputs you can achieve. Of course, this is only the beginning and there are numerous other ways to use `ggplot2` to its full potential.

Conclusions

Visualizing data is incredibly important. With clear and aesthetically pleasing graphics, the stories you tell will be far more compelling. It takes a lot of thought to determine the best way to represent your findings because the final product can make or break your presentation, no matter how much work you put in to the data manipulation. You'll learn more about your own data while coming up with the best way to display it. It's definitely worthwhile to explore `ggplot2` and other graphics packages so that you can take advantage of the tools they provide.

References

1. <http://ggplot2.org/>
2. <http://ggplot2.tidyverse.org/reference/>
3. <http://ggplot2.tidyverse.org/articles/ggplot2-specs.html>
4. <https://www.datacamp.com/community/blog/the-5-most-downloaded-r-packages>
5. <https://www.rdocumentation.org/packages/ggplot2/versions/2.1.0?>
6. <http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html>
7. <https://www.r-bloggers.com/the-grammar-of-graphics-ggplot2-package/>
8. [http://www.cookbook-r.com/Graphs/Colors_\(ggplot2\)/](http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/)
9. <http://sape.inf.usi.ch/quick-reference/ggplot2/colour>
10. <https://cran.r-project.org/web/packages/RColorBrewer/RColorBrewer.pdf>
11. <http://colorbrewer2.org/>
12. <https://blog.modeanalytics.com/violin-plot-examples/>
13. <https://www.superdatascience.com/ggplot2-setting-vs-mapping-aesthetics/>