Reporting data results #1

Scales

Scales

There are a number of different functions for adjusting scales. These follow the following convention:

```
## Generic code
scale_[aesthetic]_[vector type]
```

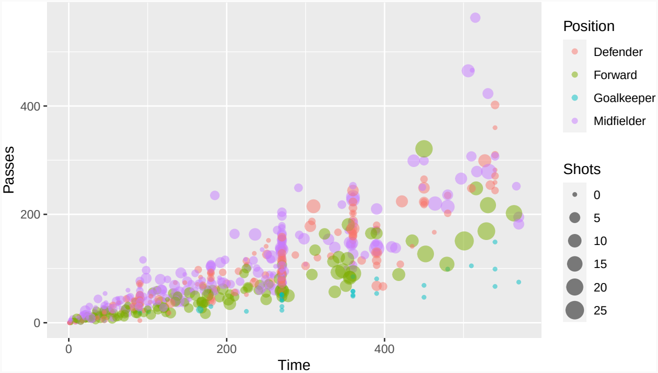
For example, to adjust the x-axis scale for a continuous variable, you'd use `scale_x_continuous`.

You can use a `scale` function for an axis to change things like the axis label (which you could also change with `xlab` or `ylab`) as well as position and labeling of breaks.

Scales

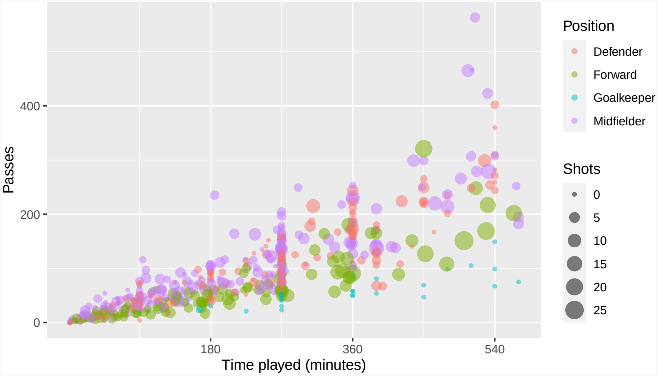
For example, here is the default for plotting time versus passes for the `worldcup` dataset, with the number of shots taken shown by size and position shown by color:

```r
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point(alpha = 0.5)
```



Scales

```
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point(alpha = 0.5) +
  scale_x_continuous(name = "Time played (minutes)",
                     breaks = 90 * c(2, 4, 6),
                     minor_breaks = 90 * c(1, 3, 5))
```



Scales

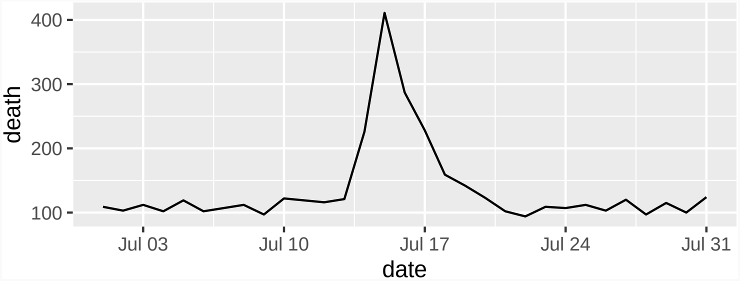
Parameters you might find useful in `scale` functions include:

Parameter	Description
name	Label or legend name
breaks	Vector of break points
minor_breaks	Vector of minor break points
labels	Labels to use for each break
limits	Limits to the range of the axis

Scales

For dates, you can use `scale` functions like `scale_x_date` and `scale_x_datetime`. For example, here's a plot of deaths in Chicago in July 1995 using default values for the x-axis:

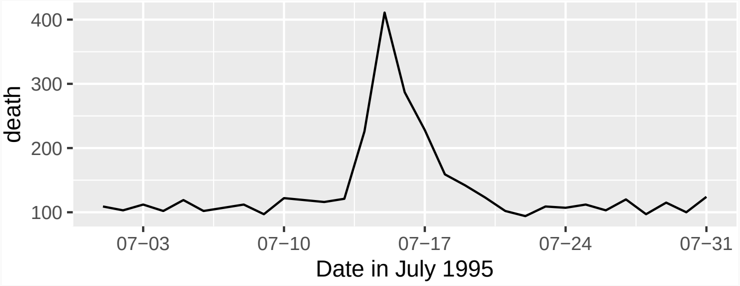
```r
ggplot(chic_july, aes(x = date, y = death)) +
  geom_line()
```



Scales

And here's an example of changing the formatting and name of the x-axis:

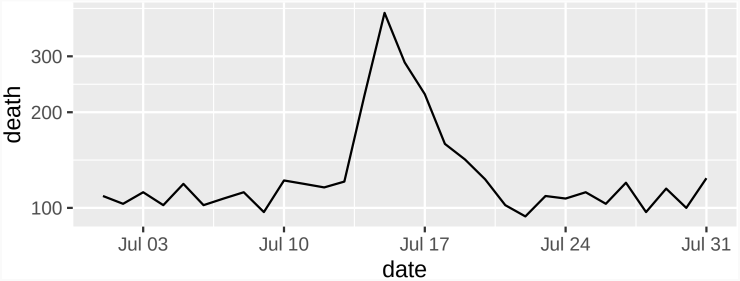
```r
ggplot(chic_july, aes(x = date, y = death)) +
  geom_line() +
  scale_x_date(name = "Date in July 1995",
               date_labels = "%m-%d")
```



Scales

You can also use the scale functions to transform an axis. For example, to show the Chicago plot with “deaths” on a log scale, you can run:

```
ggplot(chic_july, aes(x = date, y = death)) +
  geom_line() +
  scale_y_log10()
```



Scales

For colors and fills, the conventions for the names of the `scale` functions can vary.

For example, to adjust the color scale when you're mapping a discrete variable (i.e., categorical, like gender or animal breed) to color, you'd use `scale_color_hue`. To adjust the color scale for a continuous variable, like age, you'll use `scale_color_gradient`.

Scales

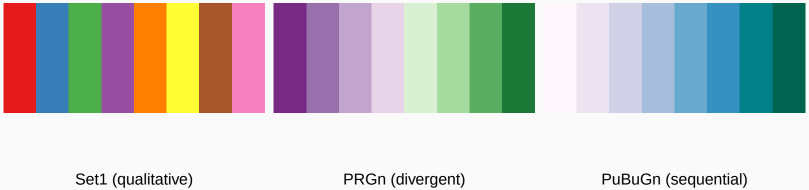
For any color scales, consider starting with `brewer` first (e.g., `scale_color_brewer`).

Scale functions from `brewer` allow you to set colors using different palettes. You can explore these palettes at http://colorbrewer2.org/.

Scales

The Brewer palettes fall into three categories: sequential, divergent, and qualitative. You should use sequential or divergent for continuous data and qualitative for categorical data. Use `display.brewer.pal` to show the palette for a given number of colors.

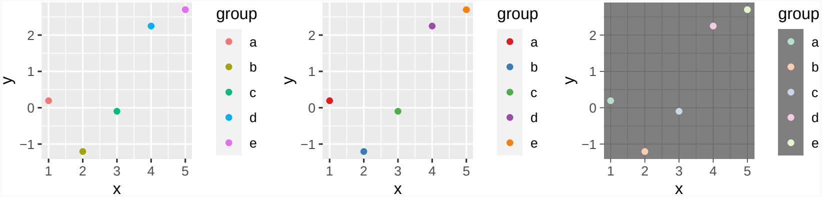
```r
library("RColorBrewer")
display.brewer.pal(name = "Set1", n = 8)
display.brewer.pal(name = "PRGn", n = 8)
display.brewer.pal(name = "PuBuGn", n = 8)
```



Scales

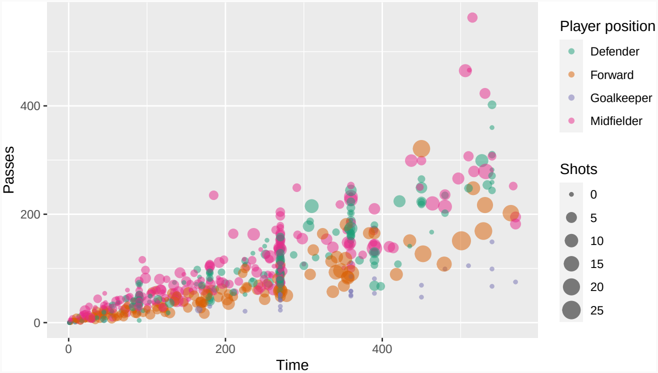
Use the palette argument within a scales function to customize the palette:

```
a <- ggplot(data.frame(x = 1:5, y = rnorm(5),
                       group = letters[1:5]),
            aes(x = x, y = y, color = group)) +
  geom_point()
b <- a + scale_color_brewer(palette = "Set1")
c <- a + scale_color_brewer(palette = "Pastel2") +
  theme_dark()
grid.arrange(a, b, c, ncol = 3)
```



Scales

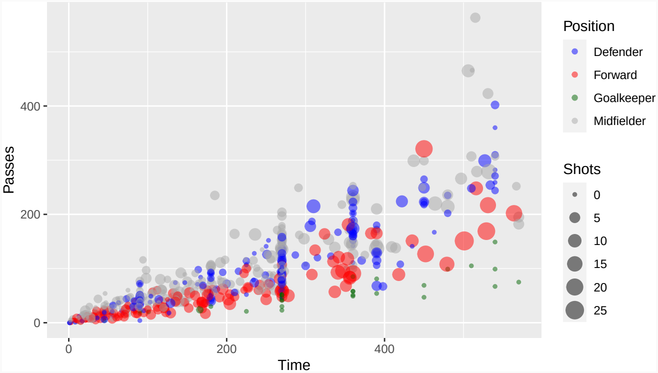
```r
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point(alpha = 0.5) +
  scale_color_brewer(palette = "Dark2",
                     name = "Player position")
```



Scales

You can also set colors manually:

```
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point(alpha = 0.5) +
  scale_color_manual(values = c("blue", "red",
                                "darkgreen", "darkgray"))
```



Excellent references

Some excellent further references for plotting are:

- Chapter 3 of *R for Data Science* by Garrett Grolemund and Hadley Wickham (http://r4ds.had.co.nz/)
- *Data Visualization* by Kieran Healy (https://socviz.co/)
- *R Graphics Cookbook* by Winston Chang (https://r-graphics.org/)
- Google images

For more technical details about plotting in R:

- *ggplot2: Elegant Graphics for Data Analysis* by Hadley Wickham
- *R Graphics* by Paul Murrell