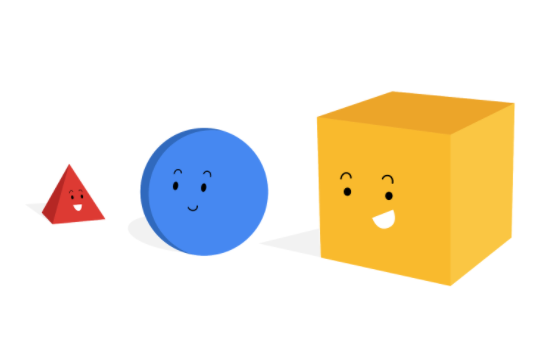
Aesthetic attributes

In this reading, you will learn about the three basic aesthetic attributes to consider when creating ggplot2 visualizations in R: **color, size,** and **shape.** These attributes are essential tools for creating data visualizations with ggplot2 and are built directly into its code.



**Aesthetics in ggplot2**

**Ggplot2** is an R package that allows you to create different types of data visualizations right in your R workspace. In ggplot2, an **aesthetic**is defined as a visual property of an object in your plot.

There are three aesthetic attributes in ggplot2:

* **Color**: this allows you to change the color of all of the points on your plot, or the color of each data group
* **Size**: this allows you to change the size of the points on your plot by data group
* **Shape**: this allows you to change the shape of the points on your plot by data group

Here’s an example of how aesthetic attributes are displayed in R:

**ggplot(data, aes(x=distance, y= dep\_delay, color=carrier, size=air\_time, shape = carrier)) +**

**geom\_point()**

By applying these aesthetic attributes to your work with ggplot2, you can create data visualizations in R that clearly communicate trends in your data.

**Additional resources**

For more information about aesthetic attributes, check out these resources:

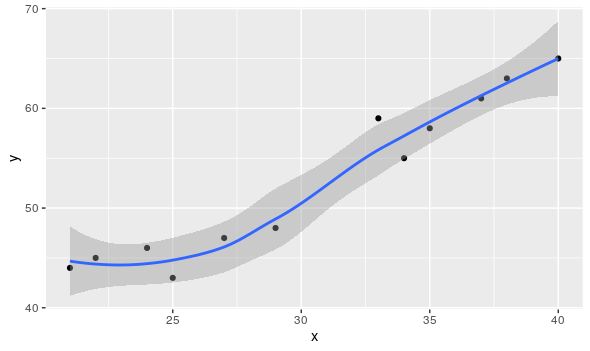
* [**Data visualization with ggplot2 cheat sheet**](https://ggplot2.tidyverse.org/): RStudio’s cheat sheet is a great reference to use while working with ggplot2. It has tons of helpful information, including explanations of how to use geoms and examples of the different visualizations that you can create.
* [**Stats Education’s Introduction to R**](http://statseducation.com/Introduction-to-R/modules/graphics/aesthetics/): This resource is a great way to learn the basics of ggplot2 and how to apply aesthetic attributes to your plots. You can return to this tutorial as you work more with ggplot2 and your own data.
* [**RDocumentation aes function**](https://www.rdocumentation.org/packages/ggplot2/versions/3.3.3/topics/aes): This guide describes the syntax of the aes function and explains what each argument does.

Smoothing

In this reading, you will learn about smoothing in ggplot2 and how it can be used to make your data visualizations in R clearer and easier to follow. Sometimes it can be hard to understand trends in your data from scatter plots alone. **Smoothing** enables the detection of a data trend even when you can't easily notice a trend from the plotted data points. Ggplot2’s smoothing functionality is helpful because it adds a **smoothing line** as another layer to a plot; the smoothing line helps the data to make sense to a casual observer.

| **Example code** |
| --- |
| **ggplot(data, aes(x=distance, y= dep\_delay)) + geom\_point() + geom\_smooth()** |

The example code creates a plot with a trend line similar to the blue line below.

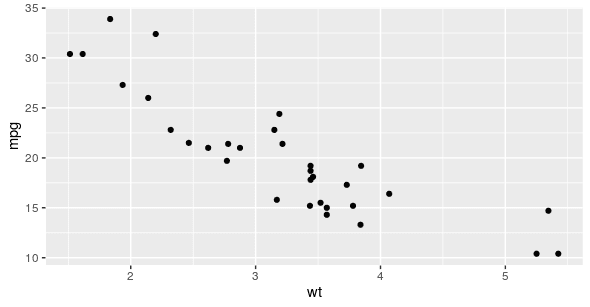


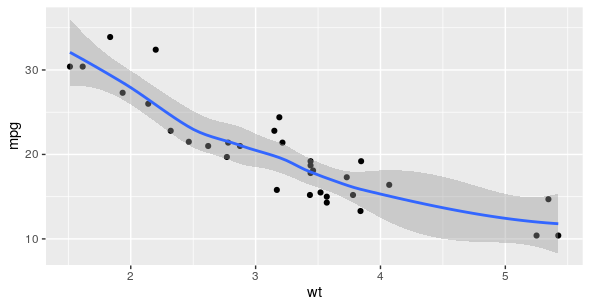
**Two types of smoothing**



| **Type of smoothing** | **Description** | **Example code** |
| --- | --- | --- |
| **Loess smoothing** | The loess smoothing process is best for smoothing plots with less than 1000 points. | **ggplot(data, aes(x=, y=))+  geom\_point() +       geom\_smooth(method="loess")** |
| **Gam smoothing** | Gam smoothing, or generalized additive model smoothing, is useful for smoothing plots with a large number of points. | **ggplot(data, aes(x=, y=)) + geom\_point() +         geom\_smooth(method="gam", formula = y ~s(x))** |

The smoothing functionality in ggplot2 helps make data plots more readable, so you are better able to recognize data trends and make key insights. The first plot below is the data before smoothing, and the second plot below is the same data after smoothing.





**Additional resource**

For more information about smoothing, refer to the Smoothing section in the [**Stats Education’s Introduction to R**](http://statseducation.com/Introduction-to-R/modules/graphics/smoothing/) course. It includes detailed descriptions and examples of how to use the different types of smoothing in ggplot2. It also includes links to other lessons about ggplot2. You can explore these to get more familiar with plotting data in R.

Filtering and plots

By this point you have likely downloaded at least a few packages into your R library. The tools in some of these packages can actually be combined and used together to become even more useful. This reading will share a few resources that will teach you how to use the filter function from **dplyr** to make the plots you create with **ggplot2** easier to read.



**Example of filtering data for plotting**

Filtering your data before you plot it allows you to focus on specific subsets of your data and gain more targeted insights. To do this, just include the dplyr filter() function in your ggplot syntax.

| **Example code** |
| --- |
| **data %>%    filter(variable1 == "DS") %>%   ggplot(aes(x = weight, y = variable2, colour = variable1)) +   geom\_point(alpha = 0.3,  position = position\_jitter()) + stat\_smooth(method = "lm")** |

**Additional resources**

To learn more details about ggplot2 and filtering with dplyr, check out these resources:

* [**Putting it all together: (dplyr+ggplot)**](https://rladiessydney.org/courses/ryouwithme/03-vizwhiz-1/#1-4-putting-it-all-together-dplyr-ggplot)**:** The RLadies of Sydney’s course on R uses real data to demonstrate R functions. This lesson focuses specifically on combining dplyr and ggplot to filter data before plotting it. The instructional video will guide you through every step in the process while you follow along with the data they have provided.
* [**Data transformation:**](https://r4ds.had.co.nz/transform.html)This resource focuses on how to use the filter() function in R, and demonstrates how to combine filter() with ggplot(). This is a useful resource if you are interested in learning more about how filter() can be used before plotting.
* [**Visualizing data with ggplot2:**](https://datacarpentry.org/dc_zurich/R-ecology/05-visualisation-ggplot2.html) This comprehensive guide includes everything from the most basic uses for ggplot2 to creating complicated visualizations. It includes the filter() function in most of the examples so you can learn how to implement it in R to create data visualizations.

Adding annotations in R



Annotations are a useful way to add notes to your plot. They help you explain the plot’s purpose, highlight important data points, or comment on any data trends or findings the plot illustrates. You have already learned how to add notes as labels, titles, subtitles, and captions. You can also draw arrows or add shapes to your plot to create more emphasis. Usually you add these kinds of annotations in your presentation application after you have saved the visualizations. But, you can now add lines, arrows, and shapes to your plots using **ggplot2**.

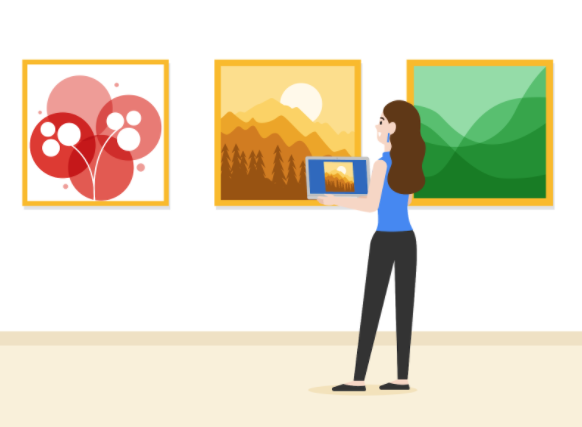
**Resources**

Check out these resources to learn more:

* [**Create an annotation layer**](https://ggplot2.tidyverse.org/reference/annotate.html): This guide explains how to add an annotation layer with ggplot2. It includes sample code and data visualizations with annotations created in ggplot2.
* [**How to annotate a plot in ggplot2**](https://www.r-graph-gallery.com/233-add-annotations-on-ggplot2-chart.html)**:** This resource includes explanations about how to add different kinds of annotations to your ggplot2 plots, and is a great reference if you need to quickly look up a specific kind of annotation.
* [**Annotations**](https://ggplot2-book.org/annotations.html)**:** Chapter eight of the online ggplot2 textbook is focused entirely on annotations. It provides in-depth explanations of the different types of annotations, how they are used, and detailed examples.
* [**How to annotate a plot**](https://www.r-bloggers.com/2017/02/how-to-annotate-a-plot-in-ggplot2/)**:** This R-Bloggers article includes explanations about how to annotate plots in ggplot2. It starts with basic concepts and covers more complicated information the further on you read.
* [**Text Annotations**](https://viz-ggplot2.rsquaredacademy.com/textann.html)**:** This resource focuses specifically on adding text annotations and labels to ggplot2 visualizations.

Saving images without ggsave()

In most cases, **ggsave()** is the simplest way to save your plot. But there are situations when it might be best to save your plot by writing it directly to a graphics device. This reading will cover some of the different ways you can save images and plots without ggsave(), and includes additional resources to check out if you want to learn more.



A graphics device allows a plot to appear on your computer. Examples include:

* A window on your computer (screen device)
* A PDF, PNG, or JPEG file (file device)
* An SVG, or scalable vector graphics file (file device)

When you make a plot in R, it has to be “sent” to a specific graphics device. To save images without using ggsave(), you can open an R graphics device like **png()** or **pdf()**; these will allow you to save your plot as a .png or .pdf file. You can also choose to print the plot and then close the device using **dev.off()**.

| **Example of using png()** | **Example of using pdf()** |
| --- | --- |
| **png(file = "exampleplot.png", bg = "transparent") plot(1:10) rect(1, 5, 3, 7, col = "white") dev.off()** | **pdf(file = "/Users/username/Desktop/example.pdf",     width = 4,      height = 4)  plot(x = 1:10,       y = 1:10) abline(v = 0) text(x = 0, y = 1, labels = "Random text") dev.off()** |

To learn more about the different processes for saving images, check out these resources:

* [**Saving images without ggsave()**](https://ggplot2.tidyverse.org/reference/ggsave.html#saving-images-without-ggsave-): This resource is pulled directly from the ggplot2 documentation at [**tidyverse.org**](https://www.tidyverse.org/). It explores the tools you can use to save images in R, and includes several examples to follow along with and learn how to save images in your own R workspace.
* [**How to save a ggplot**](https://www.datanovia.com/en/blog/how-to-save-a-ggplot/): This resource covers multiple different methods for saving ggplots. It also includes copyable code with explanations about how each function is being used so that you can better understand each step in the process.
* [**Saving a plot in R:**](https://www.datamentor.io/r-programming/saving-plot/) This guide covers multiple file formats that you can use to save your plots in R. Each section includes an example with an actual plot that you can copy and use for practice in your own R workspace.