A Short Guide Through the Hadleyverse

Brian A. Fannin

# Introduction

Although R is maintained by a core development team, there is one individual- not a member of that team- who has had a tremendous influence on the use of the language. Hadley Wickham is a highly prolific developer of R packages that have received an enthusiastic following in the R community. This is likely due to the fact that his packages are general, easy to use and span a wide range of common data manipulation problems, from working with dates and character, to easily summarizing large data sets and visualization of information. The suite of tools that he has developed has lead to the term "Hadleyverse" to describe

# ggplot2

We'll start with a look at ggplot2, a visualization package. ggplot2 is based on the idea of a "grammar of graphics" (hence the 'gg' in ggplot2) developed by Leland Wilkinson. This attempts to formalize the rules inherent in translating data to visual form.

Underneath the hood, a plot exists as a list.

* Data (of course)
* Mapping (huh?)
* Layers
* Scales

## Data

set.seed(1234)  
df <- data.frame(Cohort = rep(c("A","B"), 10)  
 , weight = rnorm(20, 170, 20)  
 , height = rnorm(20, 160, 10))  
  
library(ggplot2)  
plt <- ggplot(df)  
str(plt)

## List of 9  
## $ data :'data.frame': 20 obs. of 3 variables:  
## ..$ Cohort: Factor w/ 2 levels "A","B": 1 2 1 2 1 2 1 2 1 2 ...  
## ..$ weight: num [1:20] 146 176 192 123 179 ...  
## ..$ height: num [1:20] 161 155 156 165 153 ...  
## $ layers : list()  
## $ scales :Classes 'ScalesList', 'ggproto' <ggproto object: Class ScalesList>  
## add: function  
## clone: function  
## find: function  
## get\_scales: function  
## has\_scale: function  
## input: function  
## n: function  
## non\_position\_scales: function  
## scales: NULL  
## super: <ggproto object: Class ScalesList>   
## $ mapping : list()  
## $ theme : list()  
## $ coordinates:Classes 'CoordCartesian', 'Coord', 'ggproto' <ggproto object: Class CoordCartesian, Coord>  
## aspect: function  
## distance: function  
## expand: TRUE  
## is\_linear: function  
## labels: function  
## limits: list  
## range: function  
## render\_axis\_h: function  
## render\_axis\_v: function  
## render\_bg: function  
## render\_fg: function  
## train: function  
## transform: function  
## super: <ggproto object: Class CoordCartesian, Coord>   
## $ facet :Classes 'FacetNull', 'Facet', 'ggproto' <ggproto object: Class FacetNull, Facet>  
## compute\_layout: function  
## draw\_back: function  
## draw\_front: function  
## draw\_labels: function  
## draw\_panels: function  
## finish\_data: function  
## init\_scales: function  
## map: function  
## map\_data: function  
## params: list  
## render\_back: function  
## render\_front: function  
## render\_panels: function  
## setup\_data: function  
## setup\_params: function  
## shrink: TRUE  
## train: function  
## train\_positions: function  
## train\_scales: function  
## vars: function  
## super: <ggproto object: Class FacetNull, Facet>   
## $ plot\_env :<environment: R\_GlobalEnv>   
## $ labels : list()  
## - attr(\*, "class")= chr [1:2] "gg" "ggplot"

names(plt)

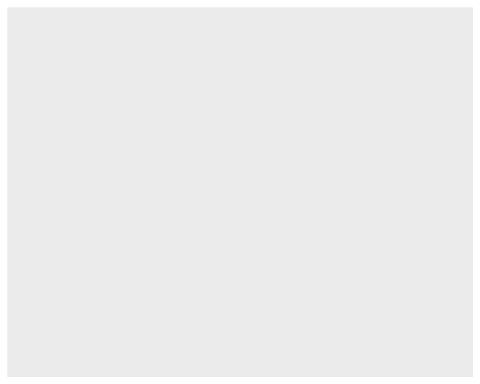
## [1] "data" "layers" "scales" "mapping" "theme"   
## [6] "coordinates" "facet" "plot\_env" "labels"

class(plt)

## [1] "gg" "ggplot"

Let's try to plot this.

plt



We are told that there are no layers to plot and indeed, if we query that element of the list, we'll see that it's empty.

plt$layers

## list()

length(plt$layers)

## [1] 0

Ok, so let's add some layers. It will quickly be evident

plt <- plt + geom\_point()  
plt

## Error: geom\_point requires the following missing aesthetics: x, y



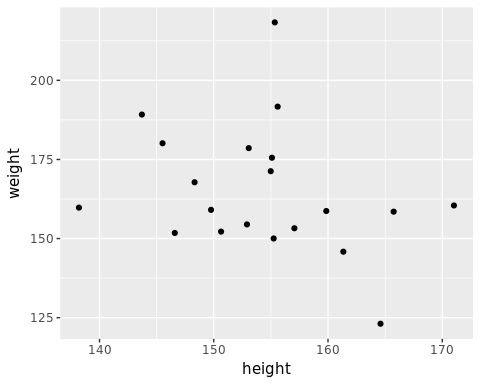
This error is a bit tougher to diagnose, so I'll cheat and tell you what's going on. We haven't said anything about what our data is *doing*. We know that we want to display points and we've fed in some data, but beyond that we don't know anything. So, given that we've got some numbers and we've got a visual element (points), but we haven't *mapped* those two concepts to one another. In this trivial case, it's fairly easy to imagine how we might do that. ggplot2 defaults to the Cartesian view of x and y axes. If we identify those, we should be in business.

length(plt$mapping)

## [1] 0

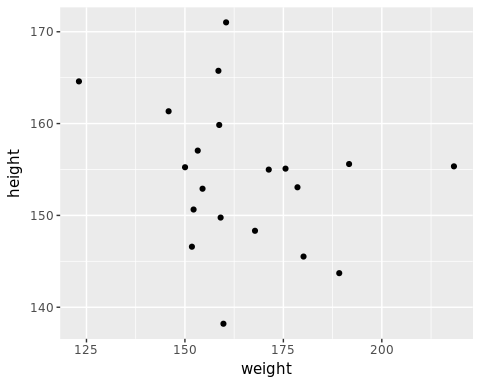
In ggplot2, the mapping is controlled by *aesthetics*. Aesthetics arer anything visual in your plot that relates to data.

plt <- plt + ggplot2::aes(x = height, y = weight)  
plt



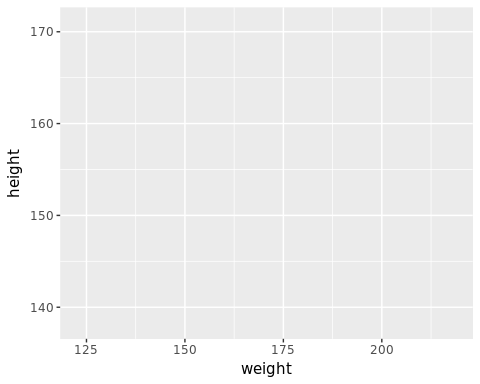
We can change things by giving our plot a different aesthtic.

plt <- plt + aes(x = weight, y = height)  
plt



What if we had a mapping, but no layer? We'll get the same "no layers in plot" error that we got earlier. Don't believe me? Have a look:

plt$layers <- list()  
plt



We'll get things back to normal by adding back our geom.

plt <- plt + geom\_point()

Pause for a minute and think about the differences between this and base graphics. For one, rather than calling plot and then making subsequent function calls, we have an *object*. We can add to this, change it, remove things and it will just happily exist. It isn't visual at all until we tell it to print itself. Two, we need three basic pieces of information: data, mapping and layers. Data is easy to understand. Layers are some form of geometric shapes and the mapping is what glues those two things together.

You're probably chomping at the bit to figure out how to add bells and whistles like axis titles, data labels and the other bits and pieces that play a prominent role in Excel. Set that aside for now and let's focus on the most important element of visualization: learning more about your data by *looking* at it. We'll start with just a single variable.

## Single variable

## Multi variable

## Grouped data

## Date and time information

## Color

## Geographic data

Where can I find ggplot2 on CRAN? Right here[[1]](#footnote-35)

# dplyr

Use dplyr with ggplot

# tidyr

By now, I hope you're beginning to understand how the interplay of these distinct tools adds up to something amazing.

# Others

## lubridate

## stringr

1. <https://cran.r-project.org/web/packages/ggplot2/index.html> [↑](#footnote-ref-35)