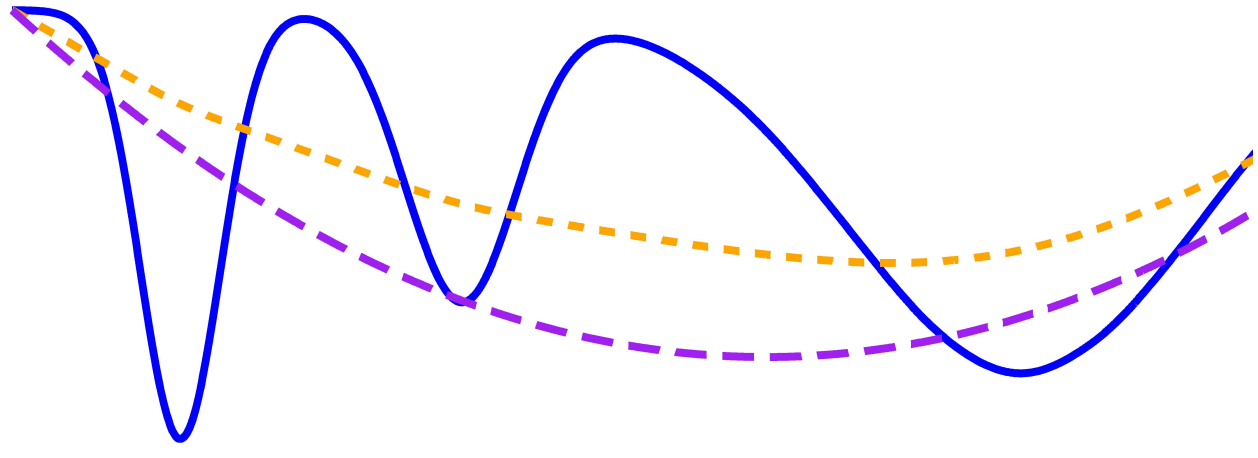


ADAM J. FLEISCHHACKER

THE BUSINESS ANALYST'S GUIDE TO BUSINESS ANALYTICS

INTRO TO MODEL-DRIVEN BUSINESS ANALYTICS WITH THE TIDYVERSE AND BRMS.



Contents

<i>1</i>	<i>Welcome to The Business Analyst's Guide to Business Analytics</i>	<i>7</i>
<i>2</i>	<i>Becoming a Data-Driven Business Analyst</i>	<i>9</i>
	<i>2.1 The Business Analyst's Perspective of Business Analytics</i>	<i>9</i>
	<i>2.2 The Real World and the Math World</i>	<i>11</i>
	<i>2.3 This is Your Journey</i>	<i>12</i>
	<i>2.4 This book is to be consumed in front of a computer.</i>	<i>12</i>
<i>3</i>	<i>Introduction</i>	<i>13</i>
	<i>3.1 Tackling Uncertainty with Random Variables</i>	<i>14</i>
	<i>3.2 Using Graphical Models to Represent Decision Problems</i>	<i>15</i>
	<i>3.3 Objectives</i>	<i>16</i>
	<i>3.4 Preview of What's Next</i>	<i>19</i>
<i>4</i>	<i>The Computing Environment</i>	<i>21</i>
	<i>4.1 Installing R</i>	<i>21</i>
	<i>4.2 Installing RStudio Desktop</i>	<i>22</i>
	<i>4.3 Getting Help</i>	<i>22</i>
	<i>4.4 Verify the installation</i>	<i>22</i>
	<i>4.5 Install and verify R-packages</i>	<i>23</i>

5	<i>R: Basic Usage</i>	27
5.1	<i>Using the Console</i>	27
5.2	<i>Writing Scripts Using the Source Code Editor</i>	28
5.3	<i>Saving Scripts & Working Directories</i>	29
5.4	<i>R-objects: Scalars, Vectors, Lists, and Dataframes</i>	31
5.5	<i>Functions</i>	36
6	<i>dplyr: Manipulating Data Frames</i>	39
6.1	<i>Tibbles</i>	40
6.2	<i>Reducing Cognitive Load</i>	41
6.3	<i>Filter Rows</i>	42
6.4	<i>Arrange Rows</i>	43
6.5	<i>Select() columns</i>	45
6.6	<i>Add new columns with mutate()</i>	47
6.7	<i>summarize() values</i>	47
6.8	<i>Commonalities</i>	47
6.9	<i>Grouped Operations</i>	48
6.10	<i>Chaining with %>%</i>	49
6.11	<i>Cheatsheets And Some Variants of The Five Verbs</i>	51
6.12	<i>Getting Help</i>	52
6.13	<i>Hadley Wickham</i>	52
7	<i>dplyr: Data Manipulation For Insight</i>	53
7.1	<i>Data Loading and Cleaning</i>	54
7.2	<i>Lateness Calculation</i>	57
7.3	<i>Bringing in Product Category Information</i>	61
7.4	<i>Answering the CEO's Questions</i>	64
7.5	<i>Notes about Data Wrangling from Twitter</i>	65
7.6	<i>Getting Help</i>	66

8	<i>Why visualize data?</i>	69
9	<i>ggplot2: Using the Grammar of Graphics</i>	73
9.1	<i>Specifying a Plot</i>	73
9.2	<i>More Information</i>	77
10	<i>Four Steps of Visualization</i>	79
10.1	<i>Purpose</i>	79
10.2	<i>Content</i>	80
10.3	<i>Structure</i>	80
10.4	<i>Formatting</i>	83
10.5	<i>Taking It Further</i>	84
11	<i>ggPlot2: A Fully Worked Example</i>	85
11.1	<i>Purpose</i>	85
12	<i>Representing Uncertainty</i>	95
12.1	<i>Random Variables With Assigned Probability Distributions</i>	95
12.2	<i>Representative Samples</i>	99
12.3	<i>Mathematics As A Simulation Shortcut</i>	104
12.4	<i>Big Picture Takeaways</i>	106
12.5	<i>Getting Help</i>	106
13	<i>Bayesian Inference</i>	107
13.1	<i>An Illustrative Example</i>	107
13.2	<i>Generative Modelling</i>	108
13.3	<i>Joint Distributions</i>	110
13.4	<i>Bayesian Updating</i>	111
13.5	<i>Inference Summary So Far</i>	114

14	<i>greta:Bayesian Updating</i>	115
14.1	<i>A Uniform Prior for Bernoulli Parameter</i>	115
14.2	<i>Bayesian Updating With greta</i>	116
14.3	<i>Investigating the Posterior Distribution</i>	119
15	<i>Joint Distributions Tell You Everything</i>	123
15.1	<i>Joint Distributions</i>	123
15.2	<i>Marginal Distributions</i>	124
15.3	<i>Conditional Distributions</i>	125
15.4	<i>MAP Estimates</i>	126
15.5	<i>Limitations of Joint Distributions</i>	127
16	<i>The Beta Distribution</i>	129
16.1	<i>A Beta Prior for Bernoulli Parameter</i>	130
17	<i>Parameter Estimation</i>	139
17.1	<i>Normal Distribution</i>	139
17.2	<i>Gamma Distribution</i>	142
17.3	<i>Student t Distribution</i>	144
17.4	<i>Poisson Distribution</i>	148
17.5	<i>Posterior Predictive Checks</i>	151
18	<i>Multi-Level Modelling</i>	157
18.1	<i>The Gym Data</i>	158
18.2	<i>Complete Pooling</i>	160
18.3	<i>No Pooling</i>	164
18.4	<i>Partial Pooling</i>	168
19	<i>Bibliography</i>	173

Welcome to *The Business Analyst's Guide to Business Analytics*

This is a Bayesian business analytics textbook made feasible by recent advances in Bayesian computing, most notably the use of better sampling techniques. Using Bayesian inference means we can now combine data with domain knowledge to extract interpretable and insightful results that lead us towards better outcomes.

This book is still very much a draft, but it is the first book of its kind, so you might find it useful in its current form. Some highlights:

- Content is accessible to anyone, even most analytics beginners. If you have taken a stats course, you are good to go.
- Provides a solid foundation and an implementable workflow for anyone wading into the Bayesian inference waters.
- Assumes no knowledge of R. Provides introduction to R, RStudio, and the Tidyverse.
- First textbook using **greta** for Bayesian inference.
- First textbook addressing a complete business analytics workflow including data manipulation, data visualization, modelling business problems with graphical models, translating graphical models into code, and presenting insights back to stakeholders.
- Provides a complete workflow within the R-ecosystem; there is no need to learn several programming languages or work through clunky interfaces between software tools.

To use some of the datasets and functions that accompany this book, make sure you install the **causact** R package (if you do not know how to do this, no worries ... you will learn this as you go through the book):

```
devtools::install_github("flyaflya/causact")
```

Please note that this work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License (<https://creativecommons.org/licenses/by-nd/4.0/>):

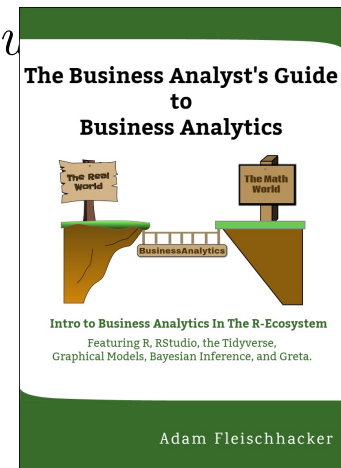


Figure 1.1: Book to be available soon.

[//creativecommons.org/licenses/by-nd/4.0/](https://creativecommons.org/licenses/by-nd/4.0/)). To support this work send feedback/follow me via Twitter:

Follow ?

and please stay tuned to this page for the launch of the upcoming printed version.

8

Why visualize data?

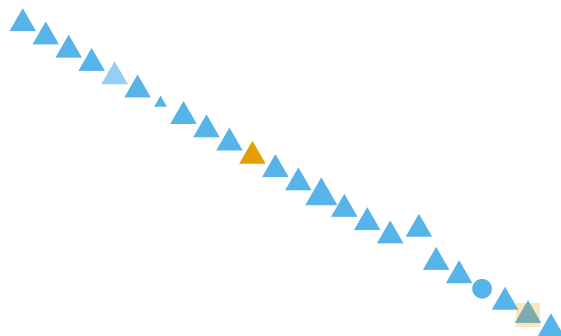


Figure 8.1: How many pattern violations do you see?

A large portion of thoughts and concepts in this chapter are inspired by Noah Ilinsky. See his talk here: (<https://youtu.be/R-oiKt7bUU8>)

VISUALIZATION MAKES DATA ACCESSIBLE TO THE HUMAN BRAIN. Evolution has wired our eyes and brain to be very sophisticated in pattern recognition. This includes detecting patterns and violations of patterns in regards to position, color, size, shape, gaps, trends, etc. For example, in Figure 8.1, your brain will easily detect seven pattern violations.

Data presented in tables or even in statistical summaries are rarely as forthcoming with insight as is a good visualization. Anscombe (1973) constructed four fictitious datasets to illustrate this point - each dataset consisting of x - y value pairs: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$. The four datasets, known as Anscombe's quartet, have virtually indiscernible statistical properties. However, the distinguishing characteristics of each dataset is very evident when graphed. Due to the cogency of the arguments made by Anscombe, these datasets have been built-in to R. We can see the data in tabulated form using the following lines:

```
# install.packages("dplyr", dependencies = TRUE)
library("dplyr")
```

```
## retrieve the anscombe dataset
ansDF = anscombe %>% as_tibble()

## notice the x-values for the first three datasets are
## the same and scanning the y-values yields little insight
ansDF

## # A tibble: 11 x 8
##       x1     x2     x3     x4     y1     y2     y3
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  10.    10.    10.     8.  8.04  9.14  7.46
## 2   8.     8.     8.     8.  6.95  8.14  6.77
## 3  13.    13.    13.     8.  7.58  8.74 12.7
## 4   9.     9.     9.     8.  8.81  8.77  7.11
## 5  11.    11.    11.     8.  8.33  9.26  7.81
## 6  14.    14.    14.     8.  9.96  8.10  8.84
## 7   6.     6.     6.     8.  7.24  6.13  6.08
## 8   4.     4.     4.    19.  4.26  3.10  5.39
## 9  12.    12.    12.     8. 10.8   9.13  8.15
## 10   7.     7.     7.     8.  4.82  7.26  6.42
## 11   5.     5.     5.     8.  5.68  4.74  5.73
## # ... with 1 more variable: y4 <dbl>
```

Notice how the x-values for each of the first three datasets, (i.e. x1, x2, and x3) are the same, yet patterns in the corresponding y-values (i.e. y1, y2, and y3, respectively) are not easily discernible. This tabulated form of data yields little insight.

One might think that statistical transformations can yield more insight. Assuming a linear relationship, the `lm` function in R can be used to get linear regression output. Notice (below) that the output for both the slope coefficient (≈ 0.5) and the y-intercept (≈ 3) is nearly identical for all four datasets and one might (wrongly) assume the datasets to be quite similar as a result:

```
model1 = lm(y1 ~ x1, data = ansDF) ##predict y1 using x1
model2 = lm(y2 ~ x2, data = ansDF) ##predict y2 using x2
model3 = lm(y3 ~ x3, data = ansDF) ##predict y3 using x3
model4 = lm(y4 ~ x4, data = ansDF) ##predict y4 using x4
##show results of regression (i.e. intercept and slope)
coef(model1)

## (Intercept)          x1
##  3.0000909    0.5000909
```

For this book, basic linear regression knowledge is assumed - for an introduction or refresher on linear regression, please consult OpenIntro's introductory statistics textbook: <https://www.openintro.org/stat/textbook.php>

```
coef(model2)
```

```
## (Intercept)          x2
##    3.000909    0.500000
```

```
coef(model3)
```

```
## (Intercept)          x3
##    3.0024545    0.4997273
```

```
coef(model4)
```

```
## (Intercept)          x4
##    3.0017273    0.4999091
```

Despite the statistical transformation (i.e. regression output) yielding nearly identical insights, visualizing the data, as shown in Figure 8.2, tells a much richer story; the type of story we want to tell as we use data visualization for both exploratory analysis and managerial persuasion.

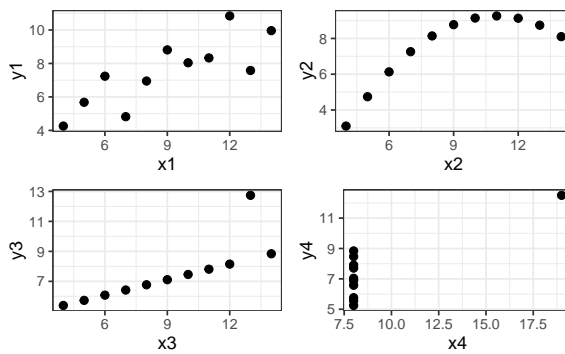


Figure 8.2: This visual depiction of Anscombe's quartet yields much more insight than simply viewing the data in tabular form or relying on output of a linear regression.

In summary, tabulated data is a struggle to read and statistical transformations might not tell the true story of the underlying data; both of these methods of looking at data can fall short of one's expectations. Only by graphing data can we readily see patterns that tabulation or statistical tests fail to help us with. Only by graphing data can we convince and persuade. In subsequent chapters, we will gain access to a rich grammar of graphics and associated R tools to make beautiful visualizations of data.

ggplot2: Using the Grammar of Graphics

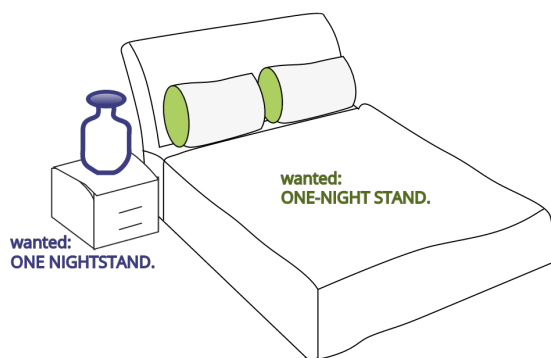


Figure 9.1: Grammar helps to convey meaning efficiently. Humorously, this cartoon compares the grammatical implications of spacing and hyphens. A one-night stand is suggestive of a short romantic encounter whereas a nightstand is simply a bedside table.

In language, rules of grammar are used to convey meaning when words are combined. Through these rules, readers can correctly comprehend the meaning an author wishes to convey. For example, Figure 9.1 is similar to a meme circulating on Facebook that shows how English grammar, in this case spacing and the use of a hyphen, changes the meaning of words. Just like with words, graphics also have an underlying grammar which can be leveraged to accurately describe a graphic or visual. This grammar, formalized in the lengthy and terse work of Wilkinson (2006) has thankfully been made much more accessible to R-users via Hadley Wickham’s excellent `ggplot2` package (Wickham, 2009). Once we learn to use this grammar properly, good graphics become easier to both describe and create.

9.1 Specifying a Plot

English grammatical rules specify that a *complete sentence* satisfies three conditions:

1. It begins with a capital letter.
2. It includes an ending punctuation mark like a period(.) or question mark(?).

3. It contains a main clause with a subject and verb.

Analogously, there are conditions required by the `ggplot2` package's implementation of the grammar of graphics to specify a *complete plot*:

1. It begins with a dataset.
2. It includes a geometric object, called a *geom*, along with that *geom*'s minimal required set of *aesthetic mappings* which specify how data is to be transformed into a visual display.

We can use the `starwars` dataset from the `dplyr` package to illustrate these two conditions:

```
# install.packages("ggplot2")
library("dplyr") ##load for starwars dataset
library("ggplot2") ##load for plotting

ggplot(data = starwars) + ## specify dataset. Use "+" to continue with more plot specifications
  geom_point(mapping = aes(x = height, y = mass)) ## pick type of geom to display and map it graphical
```

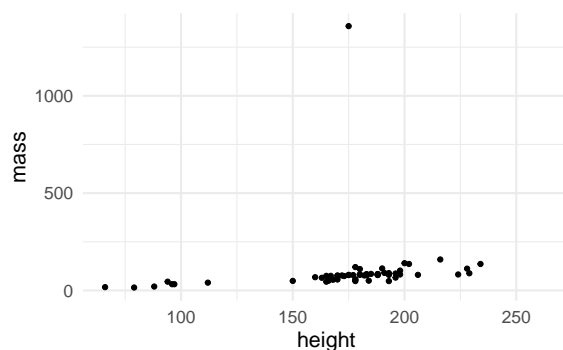


Figure 9.2: Output from `ggplot()` function - note, your output might look slightly different due to some graphical defaults that I use behind the scenes.

The initial `ggplot()` function call initiates the creation of a plot using a given dataset. The `geom_point()` function adds a graphical *layer* to the plot where:

- points will be used to represent each row of the `starwars` dataframe,
- the x-position of each point is determined by the value of the `height` variable,
- the y-position of each point is determined by the value of the `mass` variable, and
- intelligent defaults are set for every other decision required to make the corresponding plot.

See <https://ggplot2.tidyverse.org/reference/#section-layer-geoms> for complete list of available geometric objects.

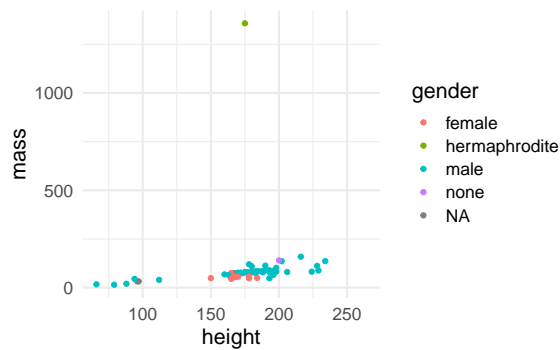
9.1.1 Simple Plot Variations

ADD ANOTHER AESTHETIC MAPPING for a useful variation. To discover more aesthetics that can be controlled when using `geom_point()`, use `?geom_point` to open the **Help** pane in the lower-right of RStudio. Scrolling down, you will discover the aesthetics shown in Figure 9.3 can be controlled when using this layer.

```
knitr::include_graphics("graphics/aes.PNG")
```

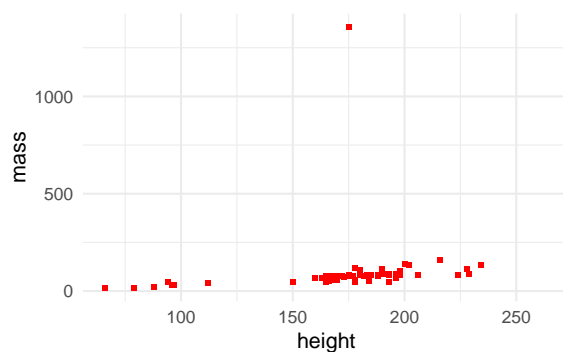
One way to control an aesthetic is to map it to the data by specifying the mapping within the `aes()` function call.

```
ggplot(data = starwars) +  
  geom_point(mapping = aes(x = height, y = mass, color = gender))
```



The other way is to map the aesthetic to a constant outside of the `aes()` function, but within the `geom` function, like this:

```
ggplot(data = starwars) +  
  geom_point(mapping = aes(x = height, y = mass), shape = 15, color = "red")
```



where color and shape are specified.

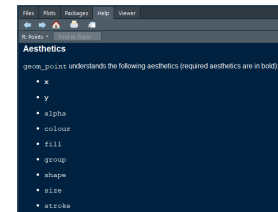
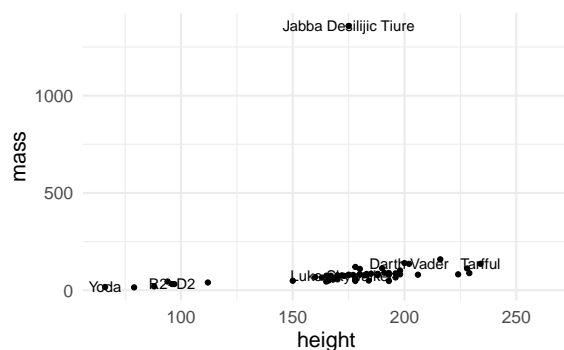


Figure 9.3: Aesthetics that can be controlled when using `geom_point()`.

For more information on what shapes and colors are available, execute the following R function `vignette("ggplot2-specs")` to open up details in the **Help** pane of RStudio.⁴

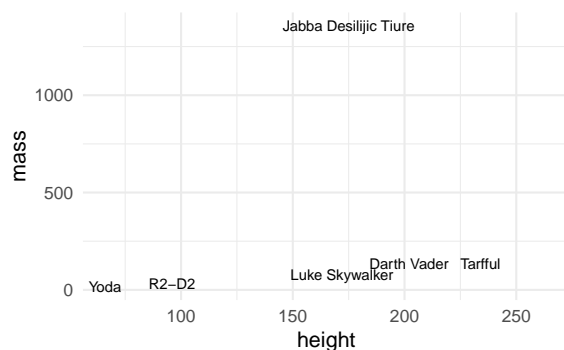
MULTIPLE LAYERS OF `geoms` can be put on one plot. For example, we might want to name the points:

```
ggplot(data = starwars) +  
  geom_point(mapping = aes(x = height, y = mass)) +  
  geom_text(mapping = aes(x = height, y = mass, label = name), check_overlap = TRUE)
```



To make the function call more concise, avoid the redundancy of mapping x-y positions for every geom by specifying mapping defaults in the initial `ggplot()` function and we can also omit the `data` and `mapping` argument names when specifying those in the order that the function expects:

```
## WARNING: check_overlap = TRUE will omit data  
## that overwrites a previous data point.  
## it is good for labelling outliers like Jabba.  
ggplot(starwars, aes(x = height, y = mass)) +  
  geom_text(aes(label = name), check_overlap = TRUE)
```



9.1.2 Other Elements of the Grammar

Above, we learned that specifying a *complete plot* required a dataset, a geom, and a minimal set of mappings. Behind the scenes, other grammatical elements were chosen by default; in reality, you can make

all of these other decisions explicit. Some of these other elements included a coordinate system, a statistical transformation, and scales:

- **Coordinate Systems** A coordinate system (coord for short) determines how data coordinates are mapped to the plane of a graphic. The default coordinate system is a two-axis system, think x- and y-coordinates, called the cartesian coordinate system.
- **Statistical Transformations:** A statistical transformation is a way of manipulating or transforming data prior to its display. It usually is used to summarize data in a meaningful way such as when creating a histogram of one variable or summarizing the relationship of two variables using a linear regression line. These transformations are optional, but can prove useful as shortcuts to get from data to useful visuals.
- **Scales:** Whereas aesthetic mappings relate data to attributes that you can visually perceive (e.g. color, symbol shapes, fill, etc.), scales dictate how the mapping from data to attribute is performed. For example, a scale might determine which colors are mapped to which values in the data.

We will learn about these other elements on an as needed basis.

9.2 More Information

The aim of the grammar of graphics is to provide an efficient language for describing visualizations. Using this grammar, however, is no guarantee that a visual will prove to be compelling. The subsequent chapter aims to help with that endeavor. For now, we recognize the grammar for what it is, namely a strong foundation for understanding and describing a wide range of graphics.

(<https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf>) is a link to the data visualization cheatsheet produced by RStudio. Gain additional proficiency with ggplot, by experimenting with the various layers, functions, scales, etc. that are described.

10

Four Steps of Visualization

There are four main steps to visualization:

1. Purpose - why are we using a visual?
2. Content - what information should be shown?
3. Structure - how should we visualize the content?
4. Formatting - how can the visual be made more informative, persuasive, and/or aesthetic?

10.1 Purpose

You are taking a year of your life to go kayaking along the coast of Greenland and you need a chart to navigate. You can't use paper because it will get wet and will be hard to fold/unfold; there is every chance you could drop it in the water. For six months of the year it is dark and you don't have your PC with you in the kayak. There is no cellular service to use a mobile device. Also, it is too cold to risk using anything that involves taking your gloves off. All three items in Figure @ref{purpose} are designed to show the Greenland coastline, which one is the best for this situation?



Figure 10.1: Which of these maps of the Greenland coastline is the best?

With some deliberation about the above story, you might discover that the wood map of the coastline has a lot of advantages: it floats, it can be felt in the dark, it does not require electricity, and it is waterproof. It is a clear winner of the three choices. However, a different user with a different *purpose* may react differently. A scientist looking

for changes in the coastline might value the historic map. A tourist looking for a nice view might prefer the map on the phone. Just like in this tangible example, when making a visualization a clear purpose is required. Without a clear sense of purpose, there will be no clear winner among the infinite visualizations that can be used.

10.2 *Content*

No matter how good a visualization might look aesthetically, if its content does not match the purpose of your analytic investigation, then it will be useless. For all visualizations, think of what matters.

Below are four questions and guidelines that can help ensure you think about content:

1. What data matters?
2. What relationships matter?
3. How might your visualization's purpose be reflected in the data and the relationships among the data?
4. Is all of the data that has been collected truly necessary? Choosing data to be excluded is often as important as choosing data to be included.

10.3 *Structure*

Structure is our choice in how to display the content we have collected. In choosing structure, we seek to accomplish four goals:

1. Choose a structure that reveals the most important relationship you are trying to convey. For example, look at Figure 10.2. The author's use of 3D graphics detracts from the message they are trying to send.
2. Choose meaningful layout and axes! The human brain excels at perceiving changes in position on a 2D surface (like a screen or piece of paper). Hence, the data mapped to the two axes for your plots become your two most important choices in mapping content to its visual representation.
3. Informed by purpose and content. Look at Figure ??, what is the author trying to convey? Show the graph most relevant to the story or persuasive argument you are trying to tell.

Structural choices can effect a reader's ability to comprehend the message of a data visualization. Figure ?? proves very helpful in making design choices:

For each encoding in Figure ?? there are six qualities of the data described in the other columns:

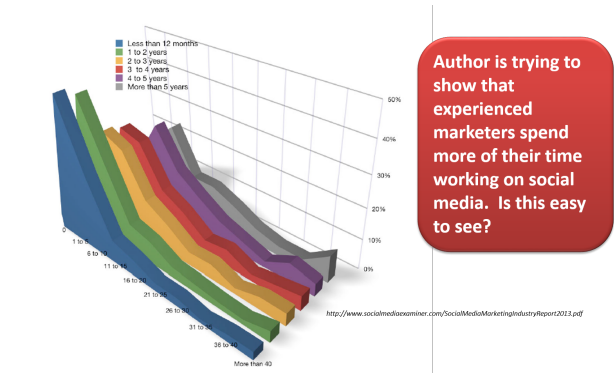


Figure 10.2: Does the author succeed in their purpose?

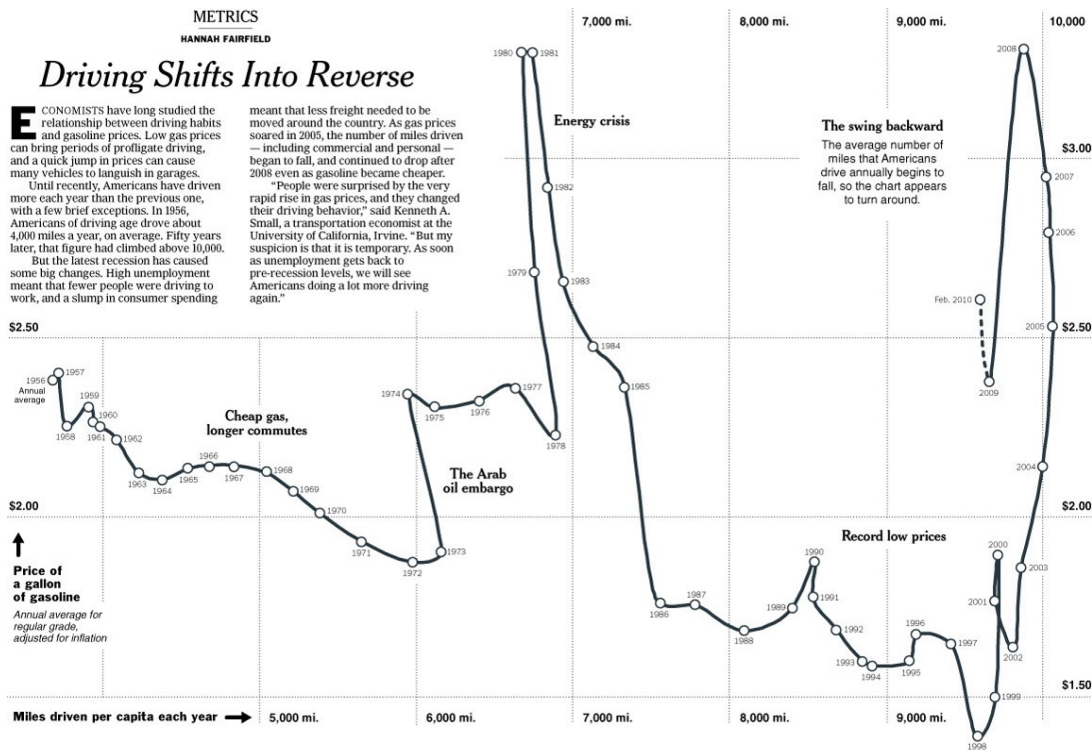





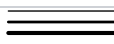





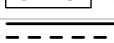
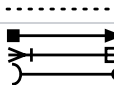
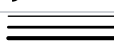


Figure 10.3: The axes are used to display the most important relationship in the data. Other formatting choices can then help tell the story.

Properties and Best Uses of Visual Encodings

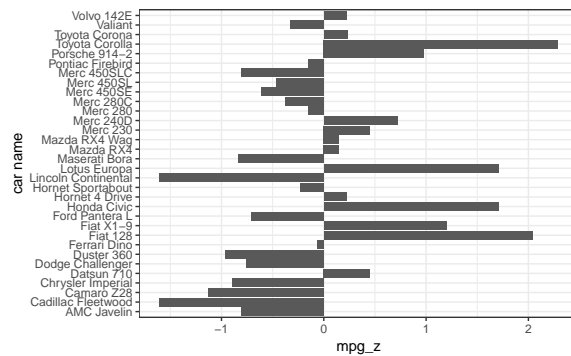
Example	Encoding	Ordered	Useful values	Quantitative	Ordinal	Categorical	Relational
	position, placement	yes	infinite	Good	Good	Good	Good
1, 2, 3; A, B, C	text labels	optional (alphabetical or numbered)	infinite	Good	Good	Good	Good
	length	yes	many	Good	Good		
	size, area	yes	many	Good	Good		
	angle	yes	medium/few	Good	Good		
	pattern density	yes	few	Good	Good		
	weight, boldness	yes	few		Good		
	saturation, brightness	yes	few		Good		
	color	no	few (< 20)			Good	
	shape, icon	no	medium			Good	
	pattern texture	no	medium			Good	
	enclosure, connection	no	infinite			Good	Good
	line pattern	no	few				Good
	line endings	no	few				Good
	line weight	yes	few		Good		



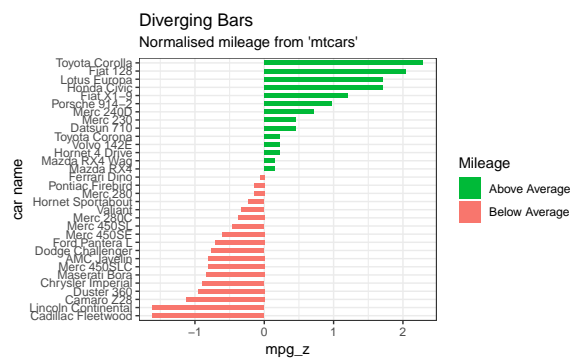
1. **Ordered:** Ordered data is data that can be sorted along a dimension. As examples, numerical data can be sorted along a number line, text labels can be sorted alphabetically, and some survey scales (e.g. “Mostly Agree”, “Slightly Agree”, “Neutral”, “Disagree”) can be sorted along a more qualitative dimension like agreeability or favorability. Visual encodings that reflect order include positioning, length, and size, but do not include color or shape where it is not obvious that green is better than yellow, or squares are larger than circles.
2. **Useful Values:** This column indicates how many distinct values can be represented by the encoding. For example, humans can easily distinguish 10 carefully chosen colors from one another, but it is impossible to choose twenty colors where each is easily distinguished from all others. Hence, color is only a good encoding for data with only a handful of unique values (e.g. gender, responses to yes/no questions, car manufacturer, etc.). Position on the other hand could be a good encoding for real-valued numbers as very small changes in position are detectable by the human eye. In contrast, something like the angular change of a line could also represent a real-valued number, but cannot be used with a lot of data as small changes in angle are hard to detect.
3. **Quantitative:** This column indicates whether the encoding works well for numeric data.
4. **Ordinal:** This column indicates whether the encoding works well for ordered data - both numeric or categorical.
5. **Categorical:** This column indicates whether the encoding works well for non-numeric data - data that takes on one of a finite number of values.
6. **Relational:** This column indicates encodings that can emphasize a relationship between data points (e.g. whether two people are connected on a social network).

10.4 *Formatting*

The last stage of data visualization is formatting. In exploratory data analysis, the work on this step should be minimal. However, once you go beyond exploration and your visualization is to be shared, then you should spend a lot of time formatting your work. In the next chapter, we show how to go from an exploratory plot, something like this:



to something you can be more proud of when sharing (and also more persuasive!):



10.5 Taking It Further

There are too many decisions made in visualization to comprehensively cover every scenario. I strongly encourage you to take a look at Claus Wilke's work "Fundamentals of Data Visualization" (<http://serialmentor.com/dataviz/>) for more tips and tricks on aligning purpose, content, structure, and formatting decisions.

ggPlot2: A Fully Worked Example

In this chapter, we will show how to use ggplot by way of example. The purpose is defined for you, but the content, structure, and formatting must be decided as we go through the example:

1. Purpose - Coors field is rumored to be the easiest field to score runs at. Is it true?
2. Content - what information should be shown?
3. Structure - how do we visualize the content?
4. Formatting - making it more informative, persuasive, and/or aesthetic?

Reputation as a home run-friendly park [\[edit \]](#)

At 5,200 feet (1,580 m) above sea level, Coors Field is by far the highest park in the majors. The next-highest, [Chase Field in Phoenix](#) stands at 1,100 feet (340 m). Designers knew that the stadium would give up a lot of home runs, as the lower air density at such a high [elevation](#) would result in balls traveling farther than in other parks. To compensate, the outfield fences were placed at an unusually far distance from home plate, thus creating the largest outfield in Major League Baseball today, according to Business Insider.^[12] In spite of the pushed-back fences, for many years Coors Field not only gave up the most home runs in baseball, but due to the resultant large field area, the most [doubles](#) and [triples](#) as well.^[13]

In its first decade, the above-average number of home runs earned Coors Field a reputation as the most

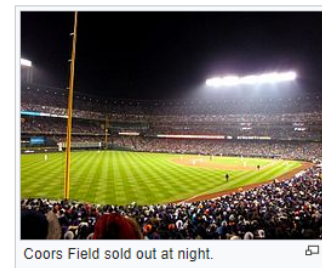


Figure 11.1: Coors Field baseball stadium has a reputation as a park that is friendly to batters. http://en.wikipedia.org/wiki/Coors_Field

11.1 Purpose

As shown in Figure 11.1, Coors field has a reputation for being friendly to batters - particularly for home runs. Our purpose is to form a data-driven opinion about the validity of this statement and create a persuasive visualization to support our opinion.

```
###the following line loads data from the 2010 - 2014 baseball seasons. The following data gets loaded
# Date      : The date the baseball game was played
# Home      : A three letter code indicating the "home" team
# Visitor   : A three letter code indicating the "visiting" team
```

```

# HomeScore :      # of runs scored by the home team
# VisitorScore :  # of runs scored by the visiting team
###note: If HomeScore > VisitorScore, then the Home team wins the game.
###      If HomeScore < VisitorScore, then the Visitor team wins the game.
library(causact)
library(dplyr)
library(ggplot2)
baseball1DF = baseballData %>% as_tibble()

```

Now, we use dplyr to manipulate the data to determine if Coors Field is a baseball park where alot of runs are scored. (note: there are lots of ways to do this, we will keep our analysis simple)

```

baseball1Data2 = baseball1DF %>%
  mutate(totalRuns = HomeScore + VisitorScore) %>%
  group_by(Home) %>%
  summarise(avgRuns = mean(totalRuns)) %>%
  arrange(desc(avgRuns))

```

11.1.1 Walkthrough of ggplot's capabilities

Let's choose our axes to be the two most important content features:

```

ggplot(data = baseball1Data2, aes(x = Home, y = avgRuns)) +
  geom_point()

```

Change structure to highlight the differences:

```

ggplot(data = baseball1Data2, aes(x = Home, y = avgRuns)) +
  geom_bar(stat = "identity")

```

Explore reversing the axes:

```

ggplot(data = baseball1Data2, aes(x = Home, y = avgRuns)) +
  geom_bar(stat = "identity") +
  coord_flip()

```

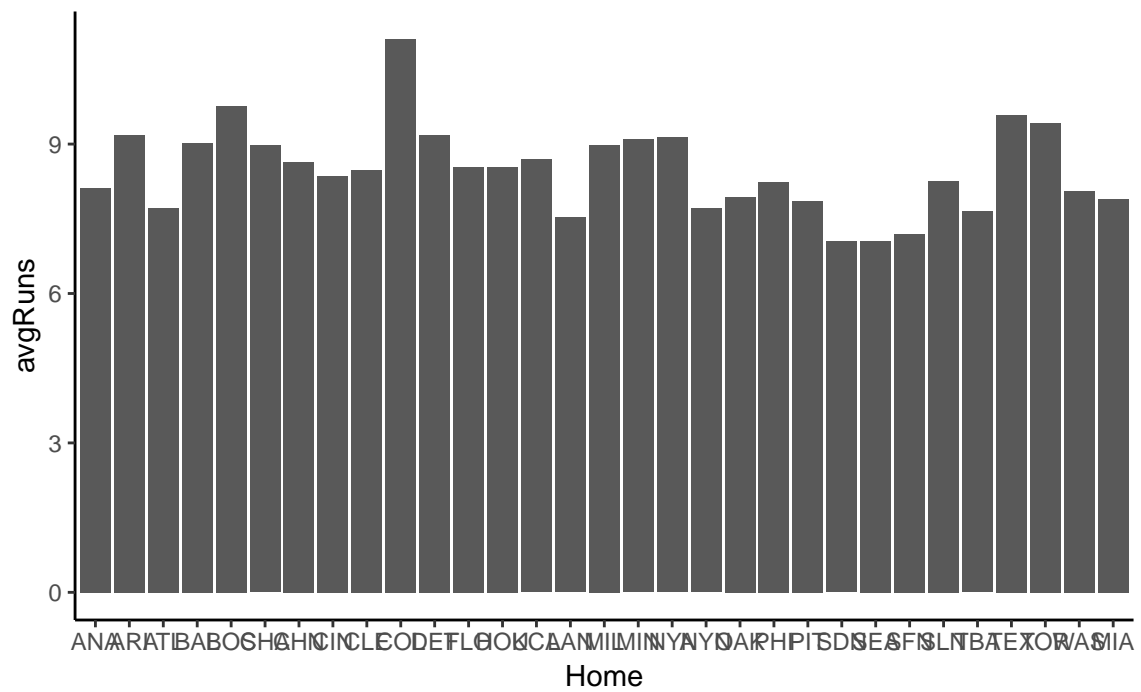
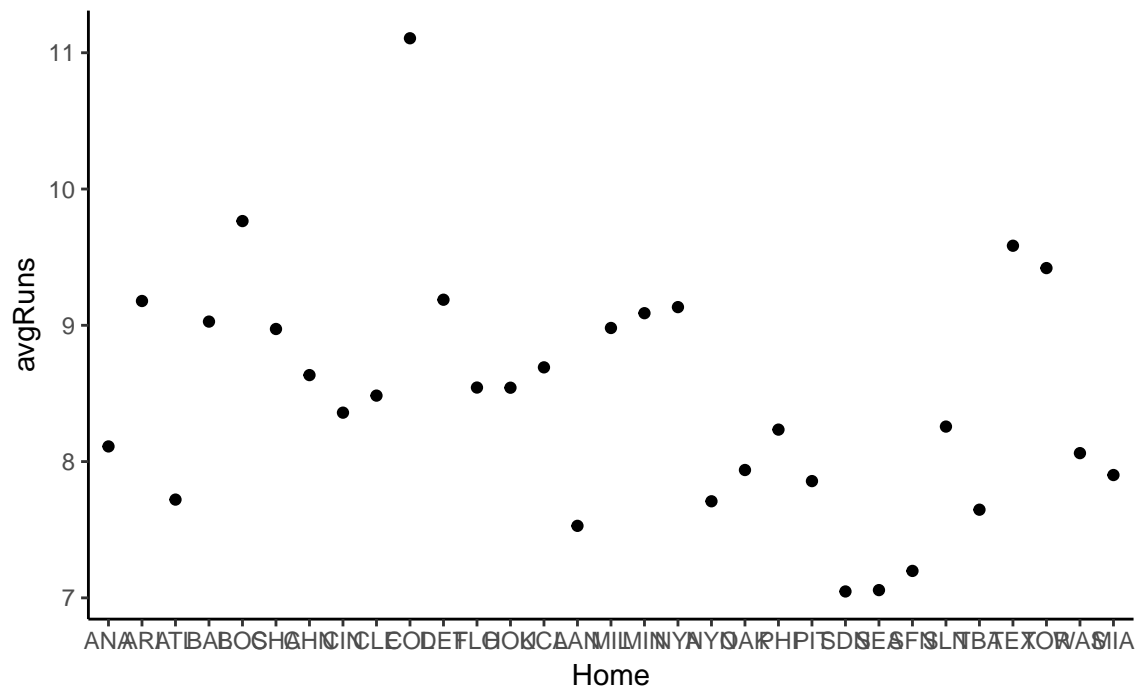
And highlight Coors Field:

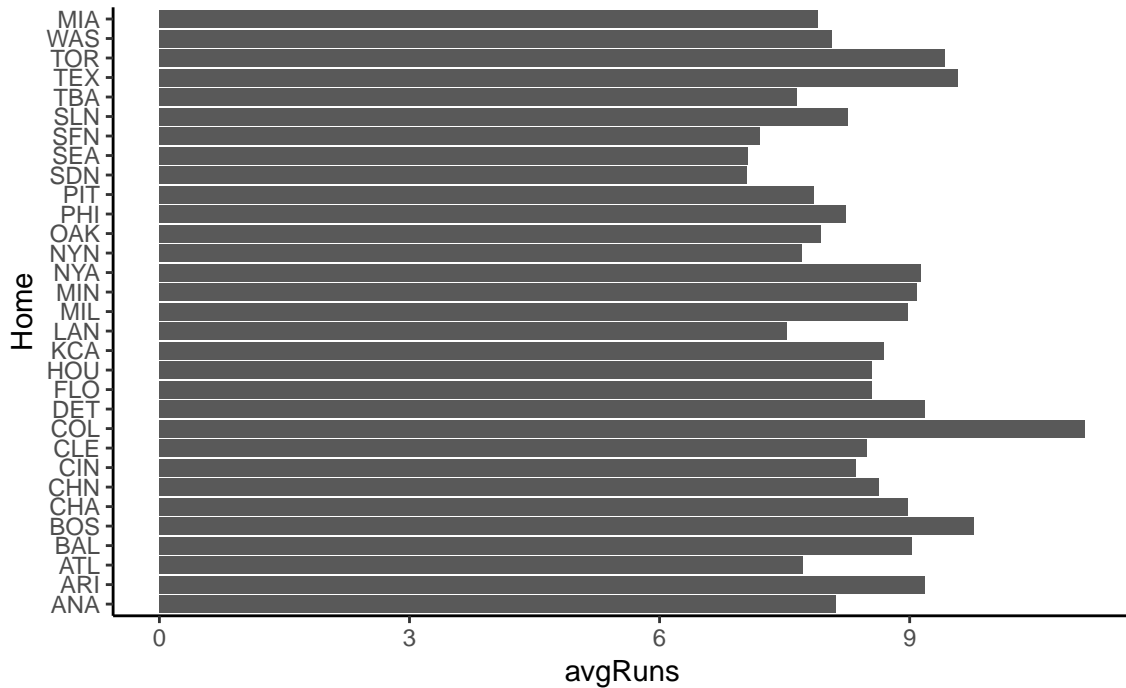
```

# Create column for data that we wish to map to the color aesthetic
baseball1Data3 = baseball1Data2 %>%
  mutate(CoorsField = ifelse(Home == "COL", "Coors Field", "Other Stadium"))

# Plot the new dataframe
ggplot(data = baseball1Data3, aes(x = Home, y = avgRuns, fill = CoorsField)) +

```





```
geom_bar(stat = "identity") +
coord_flip()
```

Reorder vertical axis labels (i.e. x-axis due to coord_flip()):

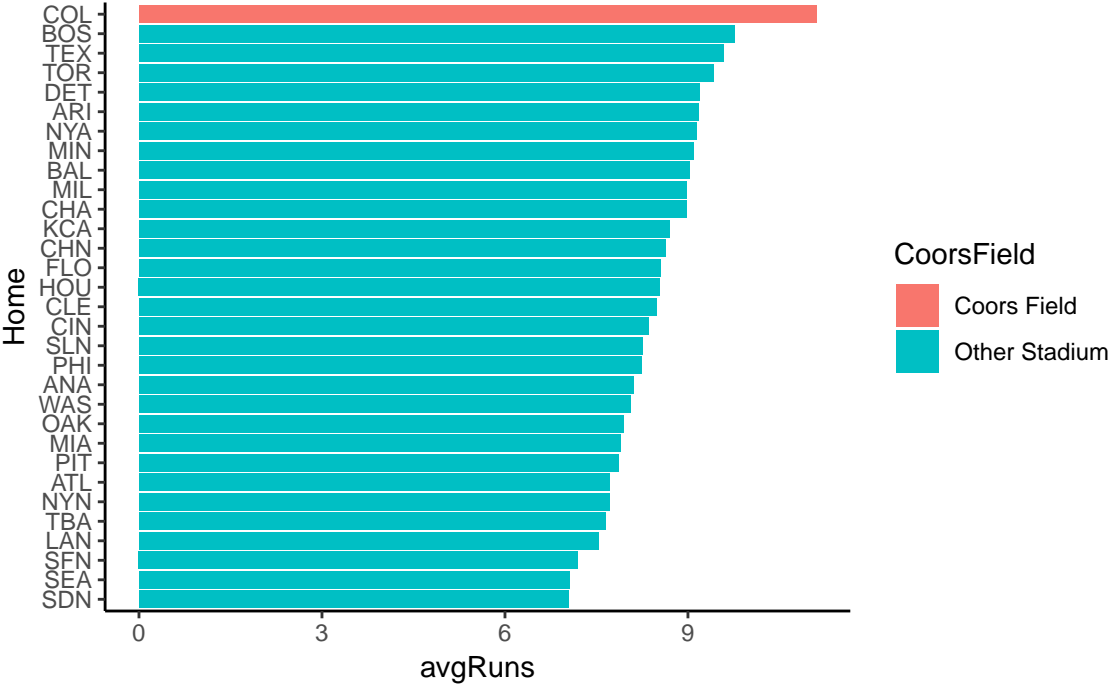
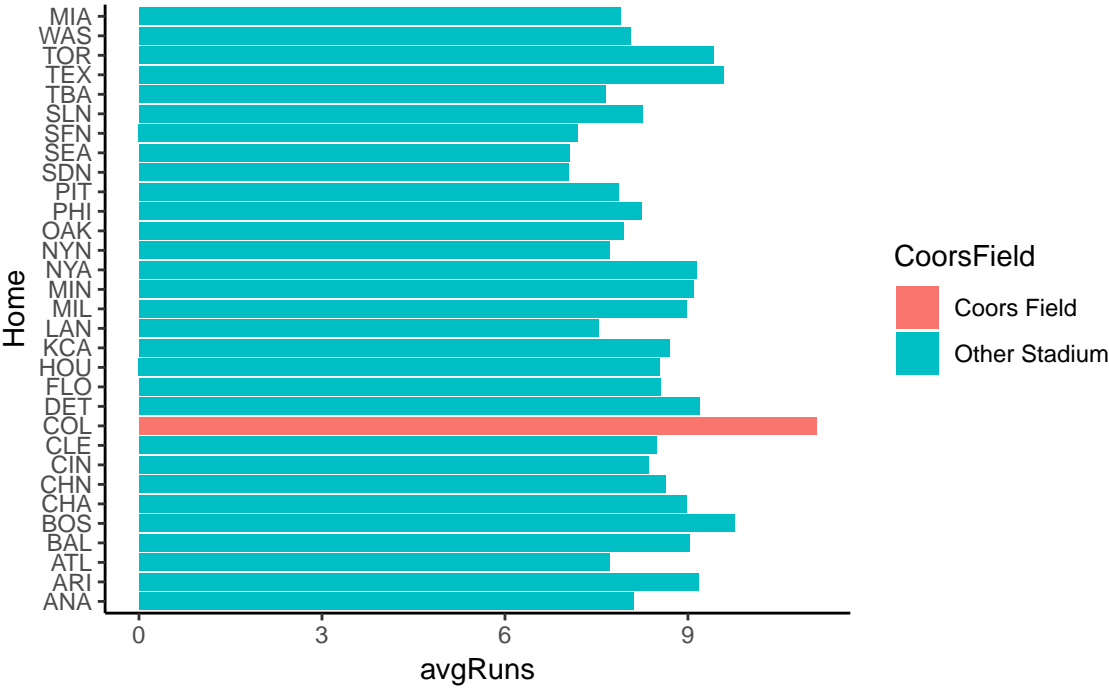
```
ggplot(data = baseballData3, aes(x = Home, y = avgRuns, fill = CoorsField)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home))
```

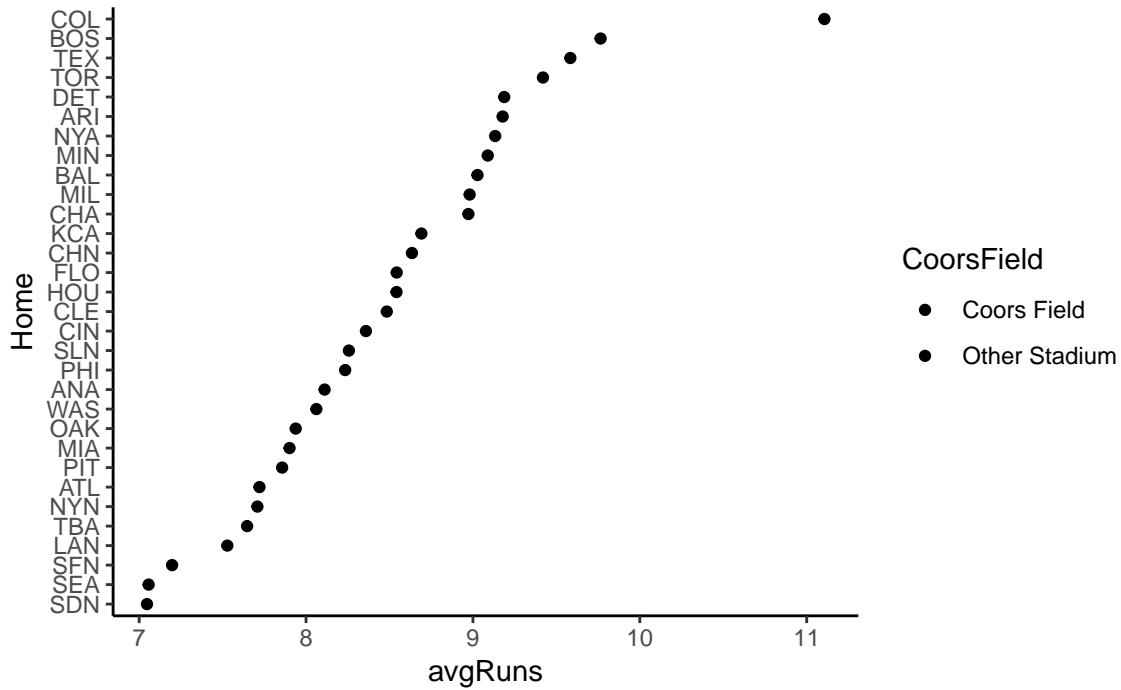
Work on formatting as there is too much blue space... try points instead of bars:

```
ggplot(data = baseballData3, aes(x = Home, y = avgRuns, fill = CoorsField)) +
  geom_point() +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home))
```

Color the points using the color attribute as fill doesn't work for points:

```
ggplot(data = baseballData3, aes(x = Home, y = avgRuns, color = CoorsField)) +
  geom_point() +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home))
```





Make the points bigger:

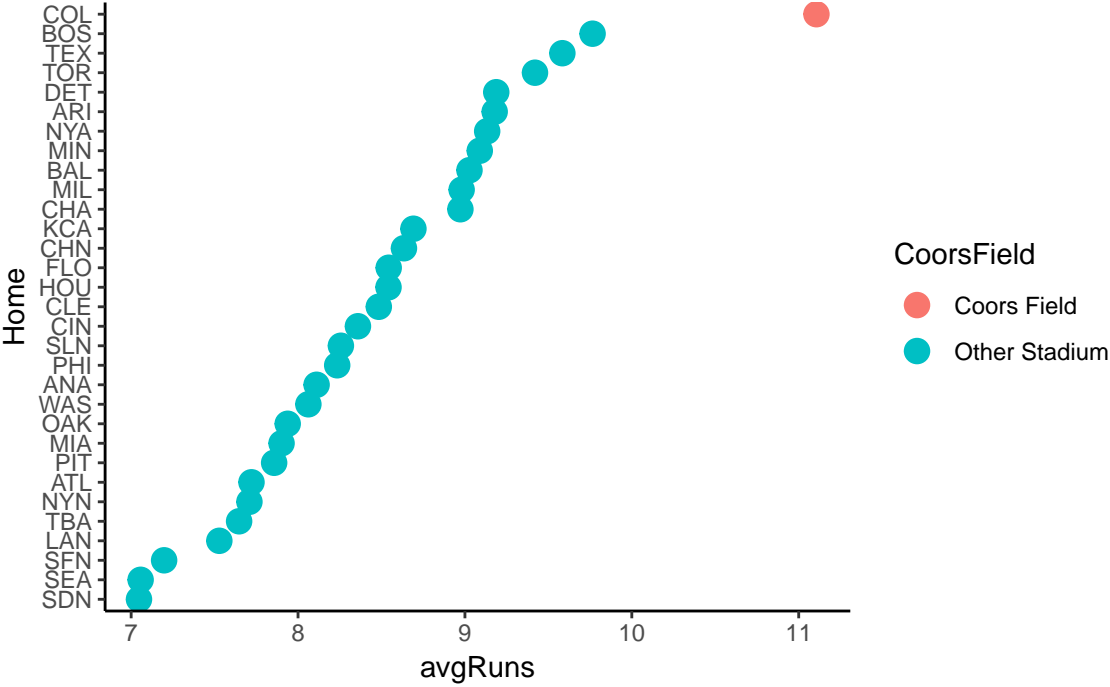
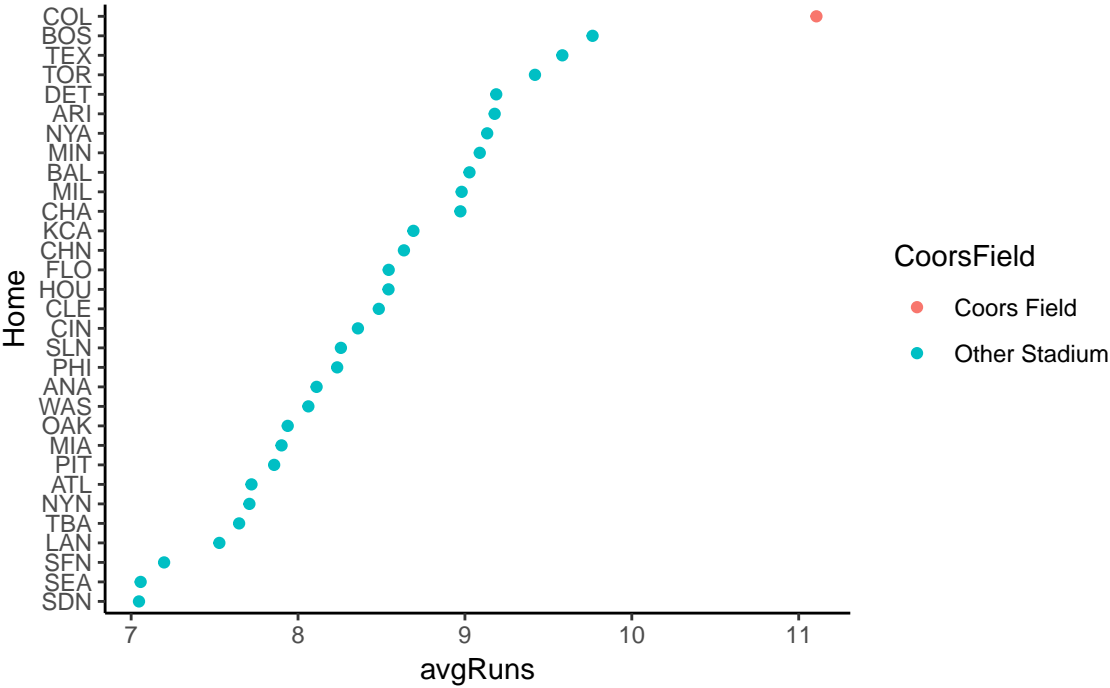
```
ggplot(data = baseballData3, aes(x = Home, y = avgRuns, color = CoorsField)) +
  geom_point(size = 4) +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home))
```

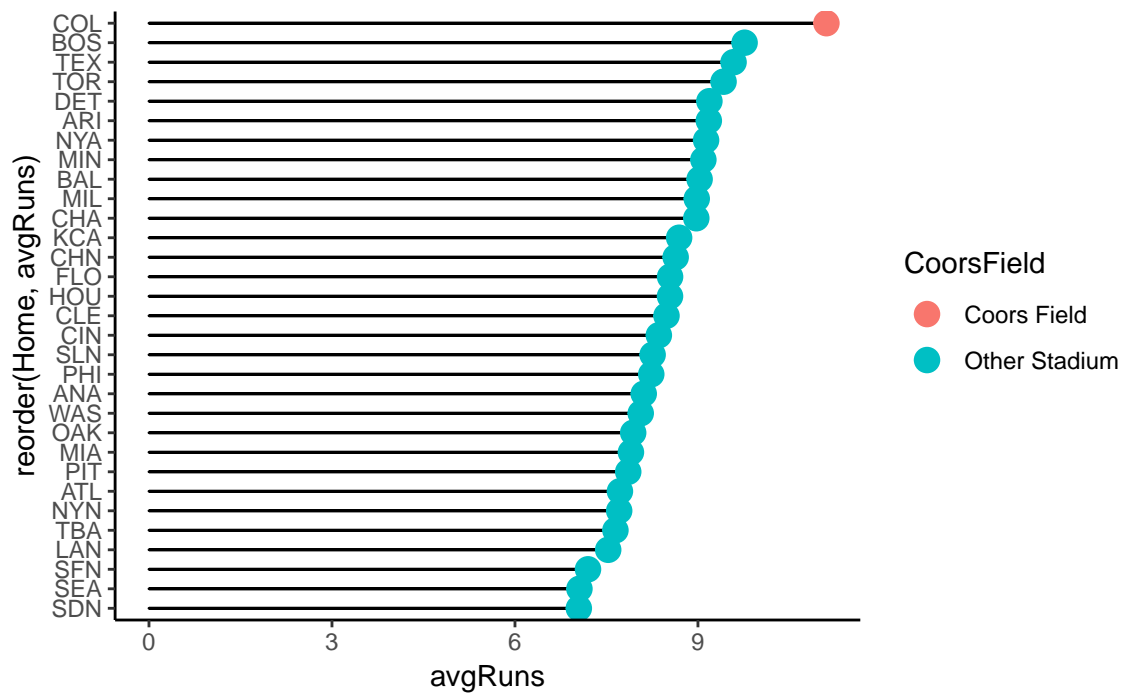
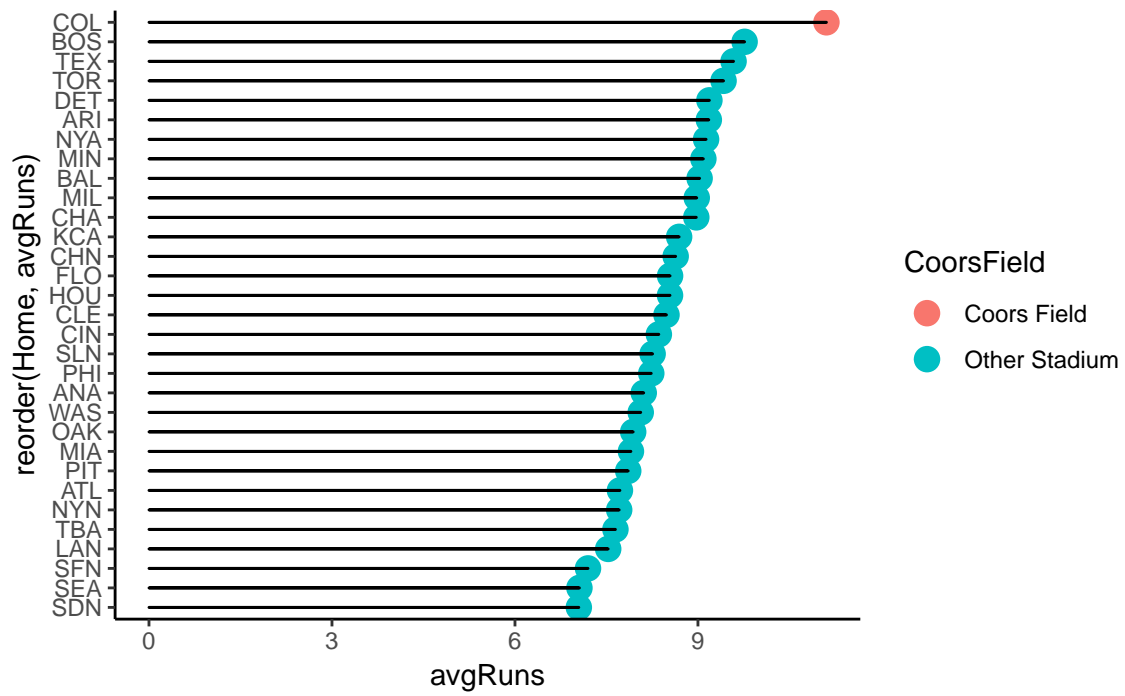
Add thin bars to points:

```
ggplot(data = baseballData3, aes(x = reorder(Home, avgRuns), y = avgRuns, color = CoorsField)) +
  geom_point(size = 4) +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home)) +
  geom_bar(stat="identity", fill = "black", color = "black", width = 0.01)
```

Plot points on top of lines by reversing the order of how the layers are drawn:

```
ggplot(data = baseballData3, aes(x = reorder(Home, avgRuns), y = avgRuns, color = CoorsField)) +
  geom_bar(stat="identity", fill = "black", color = "black", width = 0.001) +
  geom_point(size = 4) +
  coord_flip() +
  scale_x_discrete(limits = rev(baseballData3$Home))
```



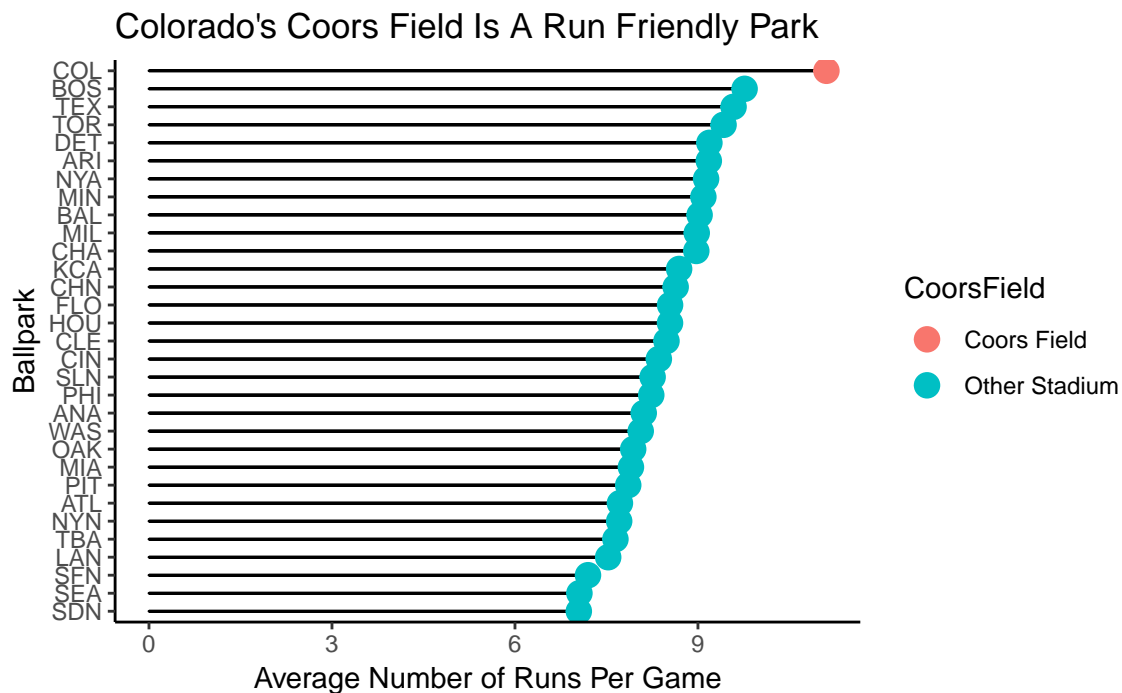


Save plot to an object and then add some title information:

```
myPlot = ggplot(data = baseballData3, aes(x = reorder(Home, avgRuns), y = avgRuns, color = CoorsField))
```

Add label information with persuasive title indicating your opinion:

```
myPlot +  
  ggtitle("Colorado's Coors Field Is A Run Friendly Park") +  
  xlab("Ballpark") +  
  ylab("Average Number of Runs Per Game")
```



Exercise 11.1. Class Exercise: A businessman that flies out of Newark airport (code: EWR) frequently believes that different airlines have varying degrees of delay depending on what day of the week it is (Monday, Tuesday, etc.). For example, he believes that it is much less probable to be delayed on a Tuesday flight than on a Monday flight. Explore the `nycflights13` dataset and create successively more complex visualizations using `ggplot` to either confirm or deny the businessman's suspicions. Try to make a recommendation for the businessman's travel patterns.:

```
#install.packages("nycflights13")
library("nycflights13")
flights = flights %>% as_tibble()
flights
```

```
## # A tibble: 336,776 x 19
##   year month   day dep_time sched_dep_time
##   <int> <int> <int>   <int>         <int>
##  1  2013     1     1     517             515
##  2  2013     1     1     533             529
##  3  2013     1     1     542             540
##  4  2013     1     1     544             545
##  5  2013     1     1     554             600
##  6  2013     1     1     554             558
##  7  2013     1     1     555             600
##  8  2013     1     1     557             600
##  9  2013     1     1     557             600
## 10  2013     1     1     558             600
## # ... with 336,766 more rows, and 14 more
## #   variables: dep_delay <dbl>,
## #   arr_time <int>, sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>,
## #   time_hour <dtm>
```