Introduction to Business Statistics: A Modeling Approach

Viswa Viswanathan

2026-01-15

Table of contents

Introduction to Business Statistics 3

Preface 4

I Data, Visualization, and Patterns 6

Module 1: Data, Visualization & Patterns 7
Learning Goals . . . 7
Structure of This Module . . . 7

1 Data Frames and Variables 8
Learning outcomes . . . 8
After working though this chapter, you will be able to: . . . 8
1.1 The R Software Package for Statistics . . . 8
1.2 R Data Frames . . . 9
1.3 Variables . . . 10
1.4 Numerical and Categorical Variables . . . 11
1.5 Data Frames in R . . . 11
1.5.1 Viewing the first few rows of a data frame . . . 11
1.6 *Tidy* data . . . 12
1.6.1 Functions . . . 13
1.7 Using R functions . . . 15
1.7.1 *nrow* . . . 15
1.7.2 The *pipe* operator . . . 15
1.7.3 *ncol* . . . 16
1.7.4 *head* . . . 17
1.7.5 *names* . . . 18
1.7.6 *summarize* . . . 18
1.7.7 *count* . . . 19
1.8 Missing values (*NA*s) . . . 20
1.9 Codebook . . . 21

2 Exploring Relationships through Scatterplots 22
Learning outcomes . . . 22
2.1 Plotting the relationship between two numerical variables . . . 22
2.1.1 Finding the point corresponding to a row of the data frame . . . 26

2.1.2 Finding the *price* and *sales_qty* corresponding to a point on the plot . . . 27
2.1.3 Dissecting the code . . . 28
2.1.4 Interpreting the plot . . . 29
2.1.5 Positive and negative relationships . . . 31
2.1.6 Putting a number on the relationship: The correlation coefficient . . . 34
2.1.7 Perfect correlation . . . 35
2.1.8 How the point cloud looks for various correlation coefficients . . . 37
2.2 Plotting a numerical variable against a categorical one . . . 39
2.2.1 A Caveat . . . 39
2.2.2 *point_plot* and "jittering" . . . 41
2.3 Bringing a third variable into the plot . . . 43
2.3.1 Understanding the tilde expression to add a third variable . . . 44
2.4 Bringing a fourth variable into play . . . 45
Mini-Review: Scatterplots and Relationships . . . 46
Question 1 . . . 46
Question 2 . . . 46
Question 3 . . . 46
Question 4 . . . 47
Question 5 . . . 47
Question 6 . . . 47
Question 7 . . . 48
Question 8 . . . 48
Question 9 . . . 48
Question 10 . . . 48
Question 11 . . . 49

II Variation 50

Module 2: Variation 51
Learning Goals . . . 51
Structure of This Module . . . 51

3 Using Violin Plots to Visualize Distribution 52
Learning outcomes . . . 52
After completing this chapter you will be able to: . . . 52
3.1 Distribution . . . 53
3.1.1 Uniform distribution . . . 55
3.1.2 Bell shaped distributions . . . 57
3.1.3 Multi-modal distributions . . . 57
3.1.4 Skewed distributions . . . 59
3.2 Violin plots . . . 60
3.2.1 Elements of the plot . . . 61
3.2.2 Examining the code . . . 61

3.3 Violin plots and distribution shapes . . . 62
3.3.1 More examples . . . 62
3.4 Comparing distributions with violin plots . . . 67

4 Variance and standard deviation **69**
Learning outcomes . . . 69
After completing this chapter you will be able to: . . . 69
4.1 Variable . . . 69
4.2 Why *means* (averages) are not enough . . . 70
4.2.1 Example 1: Two stores, same average sales . . . 70
4.2.2 Example 2: Average delivery time . . . 70
4.2.3 Example 3: Investment Returns . . . 71
4.2.4 The Key Idea . . . 71
4.3 Amount of Variation . . . 73
4.4 Measuring Variation through *variance* . . . 73
4.4.1 Computing Variance: Method 1 – Using deviations from the mean . . . 73
4.4.2 Computing Variance: Preferred Method – Using R . . . 75
4.4.3 Unit of measurement for variance . . . 75
4.5 Another measure of spread: *Standard Deviation* . . . 76
4.5.1 Computing *standard deviation* using R . . . 76
4.6 Optional enrichment topic . . . 77
4.6.1 Computing Variance: Intriguing Method 1 – Using pairwise differences . . . 77

III Introduction to models **79**

Module 3: Introduction to models **80**
Learning Goals . . . 80
Structure of This Module . . . 80

5 The Simplest Model **81**
Learning outcomes . . . 81
After completing this chapter you will be able to: . . . 81
5.1 Companies *commit* when they make decisions . . . 81
5.2 Staffing kiosks with incomplete data: Deciding under uncertainty . . . 82
5.3 Average or *mean* as the decision in the long run . . . 90
5.4 Real world connection: Planning for Customer Demand . . . 92
5.4.1 What the company must do . . . 92
5.4.2 What the company knows . . . 92
5.4.3 Why the decision matters . . . 93
5.4.4 How the company measures mistakes . . . 93
5.4.5 How the decision is evaluated . . . 93
5.4.6 The conclusion . . . 93
5.4.7 Why this matters . . . 94
5.5 Optional enrichment topic: How does skew affect the situation? . . . 94

5.6 Optional enrichment topic: Removing outliers in skewed distributions 95

6 Category Means **97**
6.1 Staffing kiosks at malls and beaches 97

7 What is a model? **102**
7.1 Mathematical models 105
7.1.1 *Response* and *Explanatory* variables 106
7.1.2 Using the model 106
7.2 Purpose of a model 106
7.3 Average is a model too! 107

8 Visualizing Mean Models **108**
8.1 Business scenario – fuel efficiency of vehicle fleet 108
8.1.1 Committing to one number for the entire fleet 109
8.2 We can do better with more information 110

IV More on models – least squares 114

Module 4: Linear Regression Models **115**
Learning Goals 115
Structure of This Module 115

9 Line of Best Fit **116**
9.1 The beach kiosk staffing problem 117
9.1.1 How does this differ from earlier examples we have studied? 118
9.1.2 Toward a rule-based model 119
9.2 Visualizing the model 122
9.3 What do we mean by the *best* model? 125
9.4 Optional enrichment topic: Beyond straight lines 130
9.4.1 When a straight line may not be appropriate 130
9.4.2 A flexible alternative: LOWESS smoothing 131
9.4.3 Models as choices, not defaults 132

10 Visualizing models **133**
10.1 Recap of models 133
10.2 Visualizing simple mean models (no explanatory variable) 134
10.3 Visualizing category mean models (single explanatory variable that is categorical) 137
10.4 Visualizing least-squares line models (single explanatory variable that is numeric) 138

11 Finding the Model Function **142**
11.1 Finding the model function for models with no explanatory variables 142
11.1.1 Building mean models by explicitly computing the mean 142
11.1.2 Building the mean model using *model_train* 144

11.2 Finding the model function for models with one explanatory variable alone . . . 146
11.2.1 Building category mean models by explicitly computing category means 146
11.3 Finding the model function for models with a single numerical explanatory variable 149

12 Amanda Strikes Again **152**

V Explanatory Power of a Model 157

Module 6: Explanatory Power of a Model **158**
Learning Goals 158
Structure of This Module 158

VI Populations, Samples and Inference 159

Module 7: Population, Samples and Inference **160**
Learning Goals 160
Structure of This Module 160

13 Population vs Sample **161**

14 Sampling Variability **162**

VII Probability 163

Module 8: Probability as Variation **164**
Learning Goals 164
Structure of This Module 164

15 Randomness **165**

16 Variability **166**

17 Distributions **167**

VIII Sampling Variation and Estimators 168

Module 9: Sampling Variation and Estimators **169**
Learning Goals 169
Structure of This Module 169

18 SE Estimators **170**

19 Sampling Distributions Slopes **171**

IX Confidence Intervals and Bands **172**

Module 10: Confidence Intervals and Bands **173**
Learning Goals . . . 173
Structure of This Module . . . 173

20 CI Slope **174**

21 CI Bands **175**

X Signal, Noise, and R-Squared **176**

Module 11: Signal and Noise **177**
Learning Goals . . . 177
Structure of This Module . . . 177

22 Variance Decomposition **178**

23 R-Squared **179**

XI Hypothesis Testing via Regression **180**

Module 12: Hypothesis Testing **181**
Learning Goals . . . 181
Structure of This Module . . . 181

24 t-tests for Regression **182**

25 Interpretation **183**

XII Business Applications of Regression **184**

Module 13: Business Applications **185**
Learning Goals . . . 185
Structure of This Module . . . 185

26 Marketing **186**

27 Finance **187**

28 Human Resources **188**

29 Operations **189**

XIII Projects and Datasets **190**

Module 14: Projects **191**
Learning Goals 191
Structure of This Module 191

30 Project Guidelines **192**

31 Datasets **193**

XIV Appendices **194**

32 Appendices **195**
32.1 Learning Goals 195
32.2 Structure of This Module 195

Introduction to Business Statistics

A Modeling and Regression-Based Approach

This text presents a modern, modeling-first approach to business statistics. Rather than a traditional probability-first sequence, we begin with data, relationships, patterns, models, and regression — the tools most relevant to business decision-making in marketing, finance, economics, HR, and analytics.

You will learn:

- how to visualize data
- how to describe relationships
- how to build regression models
- how to interpret uncertainty
- how to make data-driven decisions

Let's get started.

Preface

This book accompanies the course *Introduction to Business Statistics (BQUA2811)* at Stillman School of Business. It takes inspiration from *Lessons in Statistical Thinking* by Daniel Kaplan (*LST*). *LST* takes a modeling-centric approach and changes the ordering of topics significantly from traditional introductory statistics courses. *LST* uses almost no mathematical notation, and only provides intuitive explanation for most concepts – which serves a first statistics course for undergraduate business students very well.

After teaching the course for two semesters broadly using the *LST* approach, I found some need for changes. The following main drivers shaped the changes:

- Students benefit from understanding structure before encountering uncertainty
- A modeling-centric approach helps to provide an overarching course focus
- Courses that depend on BQUA2811 primarily expect students to be competent in linear regression and related statistics concepts.

I wanted to provide more of a business focus and use more business-related data sets than *LST* does. More importantly, I felt that *LST* introduces *statistical models* very early.Even the very first plots – lines and point estimates – that students encounter, show confidence bands. Despite emphasizing the difference between *mathematical models* and *statistical models* and providing many examples and illustrations from daily life, I found that students had a very hard time understanding the idea of a *statistical model*.

I surmised that teaching the idea of models and concepts relating to uncertainty together made it very challenging for students. Therefore I decided to make a big change and restrict the initial discussion of models to just the data set used for modeling without bringing up the idea of inference from the model. Once students grasp the idea of a least squares model independent of uncertainty, learn what it means to *explain* variation, and eventually grasp R-squared, we can then layer on uncertainty that arises when we use a model in broader contexts.

I also learned that even in developing the notion of a model, it is pedagogically very useful to ramp it up gradually starting from mean as the simplest model (when we have no explanatory variables), category means as models when we have a categorical explanatory variable, and finally extending to full-blown least-square models when the explanatory variable is also numerical. Once students grasp the idea of model, it then becomes easier to introduce uncertainty, probability and confidence intervals.

Because of this change, I needed the initial plots to only plot lines instead of confidence bands. However the plotting function from the *LSTbook* package that accompanies *LST* only provides

bands. I therefore created a package named *LSTextras* to extend the functionality of *LSTbook* package in a very small way to enable plotting of point estimates and linear models without confidence bands. I also included several business-related data sets in the package. Combined with the data sets already available in *LSTbook* and the other packages like *ggplot2* that it depends on, students using *LSTextras* will ave no dearth of data to play with and experiment with the concepts that they learn.

The book is organized into 14 modules reflecting this story line.

Part I

Data, Visualization, and Patterns

Module 1: Data, Visualization & Patterns

This module introduces the basic building blocks of statistical thinking: data frames, variables, point plots and relationships. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Describe data frames, variables, and instances

- Perform simple manipulations of data frames using R
- Create and interpret scatterplots using the *point_plot* function
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames

- Scatterplots
- Relationships between variables

1 Data Frames and Variables

We store data in an R object called a *Data frame*. This chapter introduces **data frames**, explains their key components and teaches you how to get some basic information from them. We will learn more in later chapters.

Learning outcomes

After working though this chapter, you will be able to:

- Explain the term *data frame*
- Explain what a column and row of a data frame represent
- Name the two types of variables that we will deal with in this course
- Distinguish between *numerical* and *categorical* variables and provide examples of each
- Explain what the term *level* of a categorical variable represents, and provide examples
- Explain why a column is also called a *variable*
- Explain why a row is called an *instance* or *specimen*
- Demonstrate the following skills related to R:
  - load a data frame from a loaded package
  - explain what the *pipe* operator does
  - look at the first few rows in a data frame
  - look at the last few rows in a data frame
  - retrieve only the rows that satisfy certain conditions
  - retrieve a subset of the columns of a data frame
  - find the number of rows and columns in a data frame
  - compute the average of a numeric column in a data frame
  - use the *count* function to compute the number of rows in a data frame for each level of a categorical variable

1.1 The R Software Package for Statistics

In this course, we will be using the R software package for all computations and visualizations. While R offers a great deal of functionality, we will be using a limited subset that suffices for the course topics. In particular, we will be using two specific R packages that contain all

the functions and data we need for the course. You should install R and RStudio using the instructions provided in Chapter 32. That section also shows you how to use RStudio and how to load a package – specifically the *LSTbook* and *LSTextras* packages – into R.

1.2 R Data Frames

In R, we generally represent data in a simple tabular structure named *data frame* – a table with rows and columns. Table 1.1 shows an example of some initial rows from the *Boston_marathon* data frame contained in the *LSTbook* package.

Table 1.1: Boston Marathon data (first 10 rows).

year	name	country	time	sex	minutes
2022	Evans Chebet	Kenya	02:06:51	male	127
2021	Benson Kipruto	Kenya	02:09:51	male	130
2019	Lawrence Cherono	Kenya	02:07:57	male	128
2018	Yuki Kawauchi	Japan	02:15:58	male	136
2017	Geoffrey Kirui	Kenya	02:09:37	male	130
2016	Lemi Berhanu	Ethiopia	02:12:45	male	133
2015	Lelisa Desisa	Ethiopia	02:09:17	male	129
2014	Mebrahtom “Meb” Keflezighi	United States	02:08:37	male	129
2013	Lelisa Desisa	Ethiopia	02:10:22	male	130
2012	Wesley Korir	Kenya	02:12:40	male	133

We first define the term *data frame* and then explain some terms that the definition uses.

> **Data frame:** *A structured data set organized as rows and columns, where rows correspond to individual observations and columns correspond to variables, with each column containing values of the same type.*

A data frame contains data about some topic. Each row represents an occurrence of the topic. For example, Table 1.1 contains data about the results of the Boston Marathon over the years. Each row represents the result for one year, for one gender. We also refer to a row as an *observation*, or as an *instance*, or in a medical or biology context, as a *specimen*.

Each column represents some attribute of interest about instances. In Table 1.1, we have columns to represent the year of the race, the name, county and sex of the winner and the finishing time.

In statistics, we generally refer to columns of a data frame as *variables*. We explain why in Section 1.3.

In the 1880's, Francis Galton was developing ways to quantify the heritability of traits. As part of this work, he collected data on the heights of adult children and their parents' heights. Table 1.2 shows the initial few rows of that data.

Table 1.2: Galton data (first 10 rows).

family	father	mother	sex	height	nkids
1	78.5	67.0	M	73.2	4
1	78.5	67.0	F	69.2	4
1	78.5	67.0	F	69.0	4
1	78.5	67.0	F	69.0	4
2	75.5	66.5	M	73.5	4
2	75.5	66.5	M	72.5	4
2	75.5	66.5	F	65.5	4
2	75.5	66.5	F	65.5	4
3	75.0	64.0	M	71.0	2
3	75.0	64.0	F	68.0	2

In Table 1.2, each row represents an adult child. The variables in this data frame represent the father's height (*father*), mother's height (*mother*), sex of the adult child (*sex*), height of the adult child (*height*), and the number of children (nkids) in the family. The data frame also has a unique number for each family (*family*).

1.3 Variables

If we look at a column of a data frame, say the column *time* in Table 1.1, we see that each row can potentially have a different value in this column. More generally, the values in a column of a data frame can *vary* from row to row, leading to naming columns of a data frame *variables*. Going forward, we will use the term *variable* to refer to a column of a data frame.

In this course, you will be working with many data frames. You should always be sure to understand what a row of a data frame contains information about. In the case of Table 1.1, each row represents the results of one race. The races for different genders are *different* races, even though people of both genders run together in marathons . In the case of Table 1.2, each row represents one adult child.

> **unit of measurement:** *The object or concept about which each row stores information.*

1.4 Numerical and Categorical Variables

In Table 1.1, the variable *minutes* has stores numbers. We can such variables *numerical* variables.

On the other hand, the variable *sex* can take on only the values *male* or *female*, which are not numbers Similarly, the variable *country* can also take on non-numeric values. We call such variables *categorical*.

The individual values possible or allowed for a particular categorical variable are called its *levels*. Thus the *levels* for the variable *sex* are *male* and *female*. are the *levels* of the variable *sex*. The levels for the variable *country* are country names like *Kenya*, *Japan*, and *Ethiopia*.

1.5 Data Frames in R

This section teaches you how to use data frames in R. We will predominantly use data frames built in to the *LSTbook* and *LSTextras* packages. If you have not already done so, you should install R and RStudio and learn how to use RStudio using the instructions from Chapter 32.

1.5.1 Viewing the first few rows of a data frame

In an RStudio command prompt you can enter the name of the data frame to get a preview of its contents.

```
Boston_marathon
```

```
# A tibble: 175 x 6
    year name        country time     sex    minutes
   <dbl> <chr>       <chr>   <time>   <chr>    <dbl>
 1  2022 "Evans Ch~ Kenya   02:06:51 male      127.
 2  2021 "Benson K~ Kenya   02:09:51 male      130.
 3  2019 "Lawrence~ Kenya   02:07:57 male      128.
 4  2018 "Yuki Kaw~ Japan   02:15:58 male      136.
 5  2017 "Geoffrey~ Kenya   02:09:37 male      130.
 6  2016 "Lemi Ber~ Ethiop~ 02:12:45 male      133.
 7  2015 "Lelisa D~ Ethiop~ 02:09:17 male      129.
 8  2014 "Mebrahto~ United~ 02:08:37 male      129.
 9  2013 "Lelisa D~ Ethiop~ 02:10:22 male      130.
10  2012 "Wesley K~ Kenya   02:12:40 male      133.
# i 165 more rows
```

Let us understand the output above. The first row says:

```
# A tibble: 175 Œ 6
```

This says that the data frame (also called *tibble*) has 175 rows and 6 columns.

The second line mentions the column (variable) names. The third line has some specific notation within angle brackets below each variable. This is telling us what type of value is stored in each column.

- <dbl> means that the value in a column is a *double*, that is, number that could potentially have a fractional part.
- <chr> means that the column has character or non-numeric values
- <time> tells us that the column stores a time - and so on.

If the data frame has many columns, and all of them will not fit in the width of your R console, it will only display the columns that will fit and provide information below about the additional columns. In the present case all columns do fit and so there is no additional information provided.

1.6 *Tidy* data

We encounter tabular data everywhere, each one formatted for the specific purpose it serves. Not all of them lend themselves easily to analysis, especially with software tools. To be suitable for analysis by software tools, they need to follow some minimal set of conventions. Data analysis packages can work very well with *tidy* data. In a tidy data frame:

- ** Rows represent single instances:** Each row represents exactly one instance of the unit of observation. It should not happen that one row represents a person and another row represents a university
- ** All values in a column are of the same type:** Each column represents a variable and every row has the same type of value for that variable. That is, if we have a variable named *salary* then every row represents salary in the same way, for example in dollars and as a number. It cannot be the case that one row represents *salary* in dollars and another in euros. Or that one row represents salary in dollars and another in thousands of dollars
- **Cell values are atomic:** A *cell* sits at then intersection of a row and a column. Each cell contains an *atomic* value that cannot be further decomposed into units that have some specific meaning for the analysis context. For example, in a university context, we cannot have a single cell that contains several course numbers. This would not be allowed because the list of several course numbers can be split up into individual course numbers and these still have specific meaning in a context. What about a cell that contains the name of a product (like ““soap”). That can be split up into the individual letters “s”, “o”, “a”, and, “p”. However, this does not count, since, unlike the course number example, the individual letters do not have any specific context-related meaning.

1.6.1 Functions

You are likely familiar with the term *function* as used in mathematics. You provide an input to the function, it performs some computation and provides an output. For example, we might have a function that takes a number as input and produces its square as output. Given the number 5 as input, it will produce the number 25 as its output. Given the number 1.5 as input, it will produce its square 2.25 as its output.

Pictorially we might represent a function as a box that has one or more inputs and produces a single output. (We can also have functions that take no inputs and produce an output, but we will not consider them now.) See Figure 1.1.

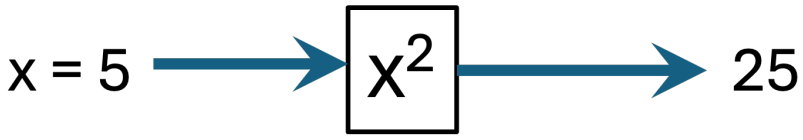

Figure 1.1: A function that squares its input

We might have functions that take more than one input. For example, a function might take two inputs x and y, and produce 23 + 3x + 4y as output. Figure 1.2 shows a function with two inputs.

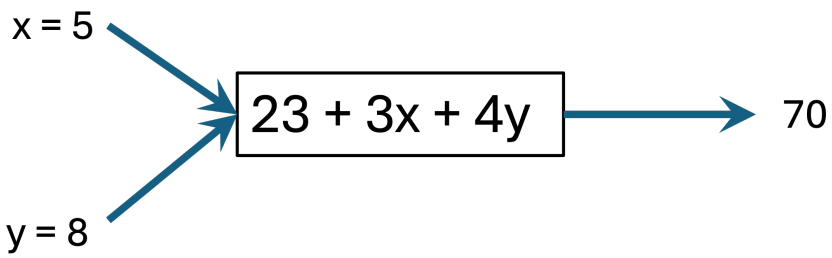

Figure 1.2: A function that computes 23+3x+4y where x and y are its inputs

In computing, we commonly refer to a function's inputs as its *arguments*. Figure 1.3 illustrates this.

The functions we considered in our examples thus far in Figure 1.1, Figure 1.2, and Figure 1.3 have been so simple that we were able to show the actual computation in the box representing the function. Most functions we use in real-life are more complex and we give them evocative names. In such cases, it makes more sense to place the name of the function in the box. Figure 1.4 shows a generic representation of a function named *cost* with three arguments named *arg1*, *arg2*, and *arg3*.

When we use a function to perform some computation, we are said to *invoke* the function. We generally invoke a function by using its name and by supplying the argument values in

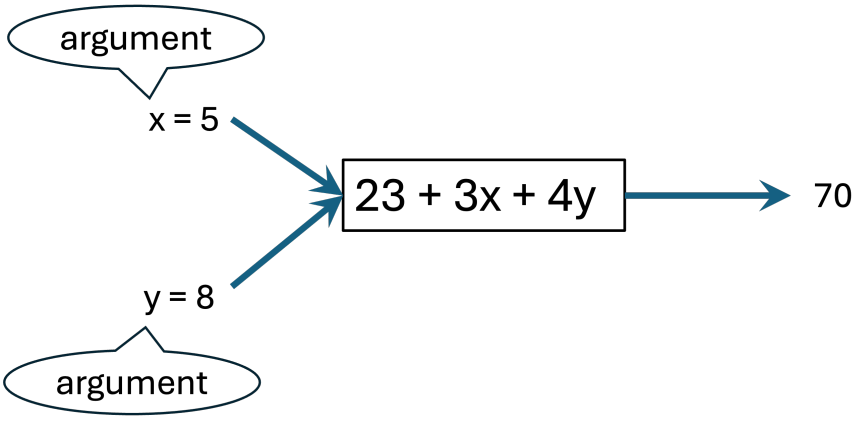

Figure 1.3: We generally refer to a function's inputs as its *arguments* – this function has two arguments, x and y

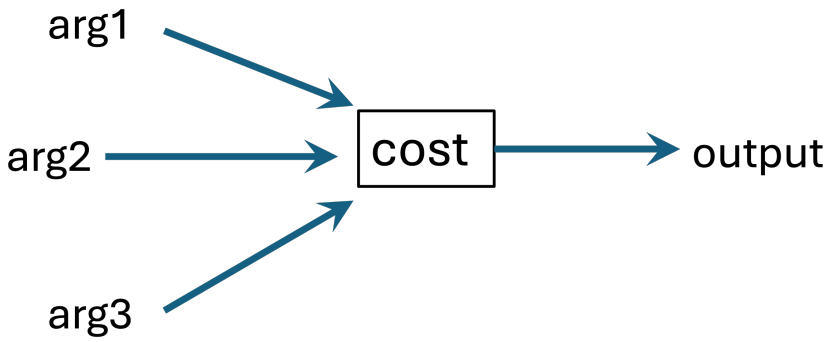

Figure 1.4: Pictorial representation of a function named *cost* that has three arguments

parentheses, as Figure 1.5 shows. We also use the term *passing arguments* to functions to refer to the act of supplying the inputs to a function. We also refer to the result of a function as its *return value*. We will also speak of a function as *returning* something. Figure 1.5 also shows that a function argument can be numeric or non-numeric.

```
total = cost(20, 32, "standard")
```

Figure 1.5: Invoking a named function and storing the function's return value in a variable named *total*

In the next section, we look at using functions to perform computing in R. At that time we will look at a different way of passing arguments to functions.

Almost all forms of computing, including statistical computing, make significant use of functions. In this course, many of the functions we use take data frame as input and produce a data frame as output. Let us now look at how we use functions in R to work with data and to perform statistical computations.

1.7 Using R functions

We will now look at some common R functions that we will use repeatedly in this course.

1.7.1 *nrow*

We will consider the *nrow()* function that allows us to find how many rows a data frame contains. We pass a data frame as argument to the *nrow()* function and it returns a number specifying how many rows the argument data frame has. For example, to find how many rows the *Boston_marathon* data frame has, we can use the following code.

```
nrow(Boston_marathon)
```

```
[1] 175
```

In the code above, we used the style of function invocation that Figure 1.5 introduced.

1.7.2 The *pipe* operator

In this course we also use another method of passing an argument to a function – via *pipes*. W first see the code and then dissect it. We will invoke the *nrow()* function again, but using a different approach. Like the previous code, this also finds the number of rows in the *Boston_marathon* data frame.

```
Boston_marathon |> nrow()

[1] 175
```

In the code above, we are still passing the Boston_marathon data frame as an argument to the *nrow()* function, but we are passing it along a *pipe* which appears in the code as "|>".

Figure 1.6 explains the different components in the code.

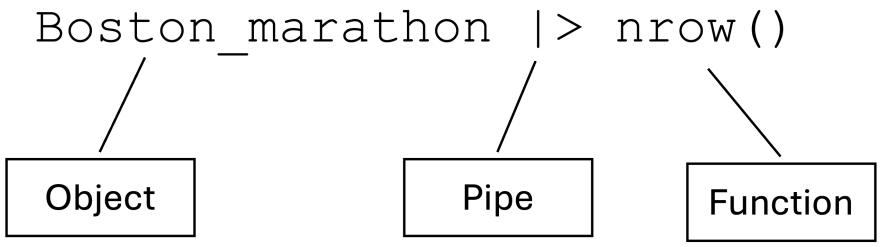

Figure 1.6: Using the pipe operator to supply the Boston_marathon data frame as an argument to the norw function

The pipe operator "|>" has the effect of *injecting* whatever appears on its left hand side as an argument to the function on the right hand side. Figure 1.7 makes this very explicit by literally depicting the pipe operator "|>" as a physical pipe.

Figure 1.7: Seeing the pipe operator literally as a pipe though which the argument flows into a function

Going forward, we will use this *pipe* approach extensively, but also mix in the earlier approach in some places. You will easily adjust to our patterns.

1.7.3 *ncol*

We can use the *ncol()* function to find the number of columns in a data frame.

```
Boston_marathon |> ncol()
```

```
[1] 6
```

The above shows us that the *Boston_marathon* data frame has 6 columns or variables.

1.7.4 *head*

You can use the *head()* function to preview the first several rows of a data frame. This time we will use the *Births2022* data frame from the *LSTbook* package.

```
Births2022 |> head()
```

```
# A tibble: 6 x 38
  month dow   place  paternity meduc feduc married
  <chr> <ord> <chr>  <chr>     <ord> <ord> <chr>
1 05    Sat   hospi~ <NA>      Assoc HS    NA
2 09    Mon   hospi~ <NA>      <NA>  <NA>  NA
3 06    Thurs hospi~ N         HS    <NA>  not_ma~
4 05    Thurs hospi~ Y         Mast~ Bach~ married
5 08    Thurs hospi~ <NA>      HS    HS    NA
6 01    Tues  hospi~ Y         HS    <NA>  not_ma~
# i 31 more variables: fage <dbl>, mage <dbl>,
#   total_kids <int>, interval <ord>,
#   prenatal_start <ord>, prenatal_visits <dbl>,
#   mheight <dbl>, wt_pre <dbl>,
#   wt_delivery <dbl>, diabetes_pre <chr>,
#   diabetes_gest <chr>, hbp_pre <chr>,
#   hbp_gest <chr>, eclampsia <chr>, ...
```

The above previewed the first six rows of *Births2022* by default because we did not specify how many rows we wanted. We can specify that explicitly. The following previews the first 10 rows.

```
Births2022 |> head(10)
```

```
# A tibble: 10 x 38
   month dow   place  paternity meduc feduc married
   <chr> <ord> <chr>  <chr>     <ord> <ord> <chr>
 1 05    Sat   hosp~  <NA>      Assoc HS    NA
 2 09    Mon   hosp~  <NA>      <NA>  <NA>  NA
 3 06    Thurs hosp~  N         HS    <NA>  not_ma~
 4 05    Thurs hosp~  Y         Mast~ Bach~ married
 5 08    Thurs hosp~  <NA>      HS    HS    NA
 6 01    Tues  hosp~  Y         HS    <NA>  not_ma~
 7 07    Mon   hosp~  <NA>      <NA>  <NA>  NA
```

```
 8 11    Thurs hosp~ <NA>       Bach~ Bach~ NA
 9 03    Sat   hosp~ <NA>       HS+   <NA>  NA
10 08    Fri   hosp~ <NA>       Bach~ Bach~ NA
# i 31 more variables: fage <dbl>, mage <dbl>,
#   total_kids <int>, interval <ord>,
#   prenatal_start <ord>, prenatal_visits <dbl>,
#   mheight <dbl>, wt_pre <dbl>,
#   wt_delivery <dbl>, diabetes_pre <chr>,
#   diabetes_gest <chr>, hbp_pre <chr>,
#   hbp_gest <chr>, eclampsia <chr>, ...
```

1.7.5 *names*

We use this to find the names of the columns in a data frame.

```
Births2022 |> names()
```

```
 [1] "month"           "dow"
 [3] "place"           "paternity"
 [5] "meduc"           "feduc"
 [7] "married"         "fage"
 [9] "mage"            "total_kids"
[11] "interval"        "prenatal_start"
[13] "prenatal_visits" "mheight"
[15] "wt_pre"          "wt_delivery"
[17] "diabetes_pre"    "diabetes_gest"
[19] "hbp_pre"         "hbp_gest"
[21] "eclampsia"       "induction"
[23] "augmentation"    "anesthesia"
[25] "presentation"    "method"
[27] "trial_at_labor"  "attendant"
[29] "payment"         "apgar5"
[31] "apgar10"         "plurality"
[33] "sex"             "duration"
[35] "menses"          "weight"
[37] "living"          "breastfed"
```

1.7.6 *summarize*

We often calculate summaries (like the average of a variable) of the data contained in a data frame.

The code below shows how to compute the average of the *time* variable in the *Boston_marathon* data frame.

```
Boston_marathon |>
  summarize(avg_time = mean(time))
```

```
# A tibble: 1 x 1
  avg_time
  <drtn>
1 8635 secs
```

In the above code, we passed the *Boston_marathon* data frame via a pipe to the *summarize* function. Within the *summarize* function, we have used the R function *mean* to compute the average by passing the variable *time* as an argument to the *mean* function. Note that we did not use the *pipe* to pass an argument to the *mean* function.

We chose to name the return value from the *mean* function as *average time*. You do not necessarily need to name the summaries we compute in the *summarize* function, but we strongly recommend it.

At appropriate points in the course, we will discuss other examples of summaries. We will see *variance*, *standard deviation*, and, *correlation coefficient*, among others, at appropriate points in the course.

For most of the summaries we compute we will use the general *summarize* function. Within the *summarize* function, we use specific functions depending on the kind of summary we compute. In the previous example, we used the *mean* function for the average. Later we will see other functions to use within the *summarize* function.

1.7.7 *count*

The *count()* function comes in handy when we want to count how many occurrences of each distinct value are present in a variable. For example, suppose we want to find how many rows are there for each *sex* in the *Boston_marathon* data frame, we can do the following.

```
Boston_marathon |> count(sex)
```

```
# A tibble: 2 x 2
  sex        n
  <chr>  <int>
1 female    50
2 male     125
```

Similarly, if we wanted to find out how many times each *country* occurs, we can do this.

```
Boston_marathon |> count(country)
```

```
# A tibble: 25 x 2
   country                n
   <chr>              <int>
 1 Australia              1
 2 Belgium                2
 3 Canada                17
 4 Colombia               1
 5 Comm. Ind. States      1
 6 England                3
 7 Ethiopia              14
 8 Finland                7
 9 Germany                6
10 Greece                 2
# i 15 more rows
```

As you can see, the *count()* function comes in handy when a variable is categorical. It also works for numerical variables, but will apply much more rarely in these cases.

1.8 Missing values (*NA*s)

Suppose the HR department of a company has a data frame of information about employees with variables (columns) *employee_id*, *first_name*, *last_name*, *base_salary*, *phone_no*, and *base_office*. It is not essential that we have data on every variable for every row. For example, it could be the case that the company does not have a *phone_no* for some employees and does not have a *base_office* for some. These are called *missing values*. A missing value denotes something whose value *we do not know*. A missing value is not the same as a blank character value or a zero numeric value. Blanks and zeros are valid and known values for variables. A *missing value is different – we do not know the value*.

Table 1.3 shows a data frame with **no missing values**. In each row, we see values for every variable.

Table 1.3: A data frame with no missing values – in every row, every variable has a value

channel	advertising_spend_k	weekly_sales_k
Search Ads	28.1	152
Search Ads	30.6	152
Search Ads	93.4	369
Search Ads	51.3	289
Retail Promo	54.6	175
Search Ads	86.6	342
Retail Promo	46.5	169
Retail Promo	41.4	142

Table 1.3: A data frame with no missing values – in every row, every variable has a value

channel	advertising_spend_k	weekly_sales_k
Search Ads	78.1	322
Retail Promo	41.1	182

On the other hand Table 1.4 shows a small data frame **with missing values**.

Table 1.4: A data frame with missing values – the *NA*s in columns *phone_no* and *base_office* represent missing values

employee_id	first_name	last_name	base_salary	phone_no	base_office
10210	Betty	Chu	67000	+1 (777) 555-1212	New Jersey
24532	Jason	Fingleton	85000	+1 (732) 555-1212	NA
56437	Shim	Sung	76000	NA	New York
20320	Ravi	Shankar	100000	+91 81487 03210	New Delhi
67582	Emily	Weitz	87000	NA	San Francisco

1.9 Codebook

Before using a data frame, we need to understand it well. That means, at the very least, that we know the *unit of observation*, the meaning of each *variable* and the *units of measurement* for a variable where applicable. The *codebook* for a data frame provides this information. You can use the ? operator for this purpose as the example code below shows.

```
?Boston_marathon
```

This command will show the codebook in the bottom right pane of RStudio.

2 Exploring Relationships through Scatterplots

This chapter covers the basics of generating and interpreting scatterplots.

The term *statistics* often conjures up thoughts of numerical computations. However, before we perform various computations, we often get a feel for the data and the underlying patterns by first visualizing the data. In this course we will use only a few standard plots to communicate essential statistical ideas.

In fact, use a single function – *point_plot* – to generate all of our plots. Although R offers numerous, sophisticated plotting features to plot almost anything we can imagine, we will keep things simple and focus only on what we need to support the concepts that our course covers.

Learning outcomes

Interpret point plots - Explain what a point means (x value and y value). - Map between plot row in a data frame. - Describe direction/strength/absence of relationship.

Use R to generate point plots

- Use point_plot(y ~ x) for numeric–numeric and point_plot(y ~ x_cat) for numeric–categorical.
- Use + color_var and + facet_var in the formula.

Explain and interpret correlation - Explain what correlation measures (direction + strength of linear association) and interpret sign/magnitude.

2.1 Plotting the relationship between two numerical variables

We will rely extensively on a single plotting function named *point_plot()*. It lives in the *LSTbook* package. We will be loading both the *LSTbook* package and the *LSTextra*s package. Once we load these packages the *point_plot()* function becomes available for us to use.

Let us first look at the *price_demand* data frame that is built into the *LSTextras* package and is therefore available as soon as you load the package.

A hypothetical company has gathered data for a single product. Over time, and across very similar sales areas, the company has charged different prices for the same product, It has recorded the monthly sales for each price. The data frame *price_demand* contains the price and the corresponding sales quantities. We would like to study if the two variables *price* and *sales_qty* are related in any way.

First let us see some rows from the data frame.

Table 2.1: Price Demand data (first 15 rows)

price	sales_qty
11.8	1260
12.1	1215
13.6	1214
12.3	1298
12.4	1160
13.7	1138
12.6	1174
11.2	1220
11.7	1313
11.9	1283
13.3	1170
12.6	1247
12.6	1283
12.3	1263
11.8	1259

The full data frame has 500 rows. With so many rows, we cannot easily get a handle on the overall pattern by looking at the data directly. Visualizing the data might help us. We will do several things in this section:

- generate the plot using R code
- understand how each element of the plot is related to the original data frame
- understand the various elements of the code so that you can use similar code to generate plots that you many need
- interpret the plot

Quick Check

In the *price_demand* data frame, what does one row represent?

Suggested answer

The price charged during a particular month and the corresponding sales quantity..

Quick Check

In the *price_demand* data frame, which variable does a company have control over or that it can set and which one is the outcome variable?

Suggested answer

price is the input/explanatory variable; *sales quantity* is the outcome/response

Quick Check

If the data frame has 500 rows, how many (price, sales_qty) pairs exist?

Suggested answer

500

The code and the plot appear below. For now, ignore the code. We will discuss it in detail after analyzing the plot itself.

```
price_demand |>
  point_plot(sales_qty ~ price)
```

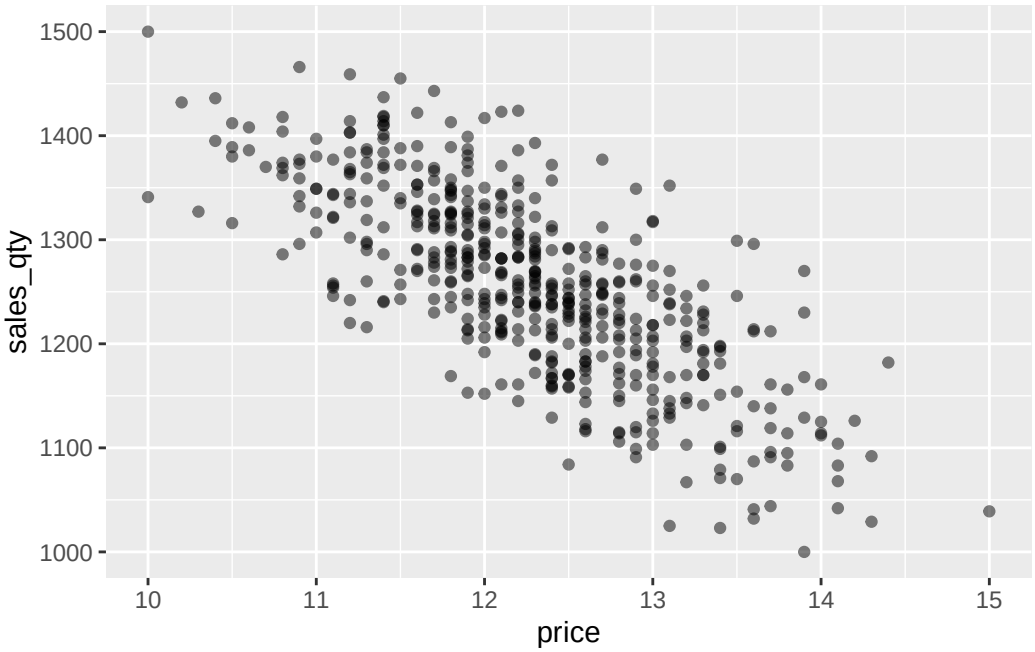

Figure 2.1: Price demand relationship

Figure 2.1 shows the relationship between *price* and *sales_qty*.

Quick Check

Figure 2.1 which variable appears on the x-axis and which one on the y-axis?

Suggested answer

x-axis: *price* y-axis: *sales_qty*

Quick Check

What two numbers determine the location of a point in Figure 2.1?

Suggested answer

Its x-value (*price*) and its y-value (*sales_qty*)

Quick Check

If a point is at x = 14 and y = 1100, what does it mean in context?

Suggested answer

There was an observation (row in the *price_demand* data frame) where the price was about 14 and the monthly sales quantity was about 1100.

Note the following about the plot:

- The plot has two axes – the x-axis along the width and the y-axis along the height.
- The plot shows the values of price on the x-axis and the values of sales_qty on the y-axis.
- If you take any single point on the plot, you can drop a vertical line from the point to the x-axis and the point where it intersects the x-axis is the *price* corresponding to the point.
- Similarly, if you draw a horizontal line from the point to intersect the y-axis, you get the *sales_qty* corresponding to that point.
- Each point of the plot corresponds to one row of the data frame. So how many points are on the plot? 500, since the data frame has 500 rows.
- Thus given a point on the plot, we should be able to identify a corresponding row of the data frame
- Given a row of the data frame, we should be able to identify a corresponding point on the plot.

Quick Check

True or False: Two different rows can produce the exact same point.

Suggested answer

True. If two rows have identical price and sales_qty, they would overlap at the same location.

Quick Check

If you see an extreme point far above the rest, what does that imply about its row in the data frame?

Suggested answer

The row on which the point is based has an unusually high sales_qty value relative to the others (for its price).

Quick Check

Why might it be hard to identify the exact row for a point just from the plot?

Suggested answer

Many points can be close together or overlapping, and the plot shows approximate positions rather than exact values.

2.1.1 Finding the point corresponding to a row of the data frame

Let us now try to find the point on the plot corresponding to the first row in Figure 2.1.

The first row of the data frame has *price = 11.8* and *sales_qty = 1260*. Therefore, we can find the point by drawing a vertical line from 11.8 on the x-axis to intersect a horizontal line from 1260 on the y-axis. Figure 2.2 illustrates this. We have highlighted the point by enlarging it.

Quick Check

Suppose a row has *price = 12.5* and *sales_qty* = 1200. Describe how to locate its point on the plot.

Suggested answer

Find 12.5 on the x-axis, move up to where y is 1200, and mark the intersection.

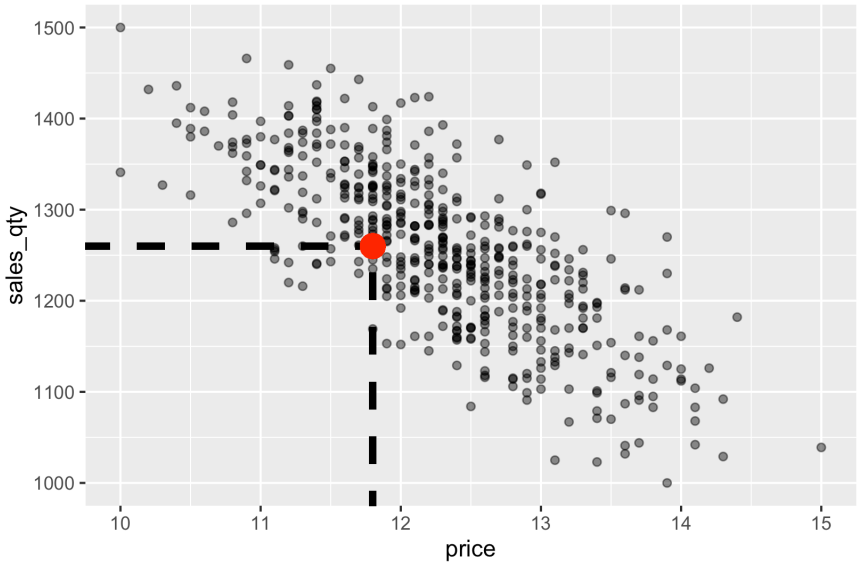

Figure 2.2: Finding the first row of the *price_demand* data frame on the plot: *price = 11.8*, *sales_qty = 1260*

> **Quick Check**
>
> If you draw a vertical line at *price* = 13, you will cut through several points. What is common to all of these?

> **Suggested answer**
>
> They all correspond to rows that have *price* = 13.

Similarly, let us locate the point corresponding to row 11 (*price = 13.3*, *sales_qty = 1170*).

following the same approach we arrive at Figure 2.3.

2.1.2 Finding the *price* and *sales_qty* corresponding to a point on the plot

We can reverse the above process to find the variable values (and hence the actual row) corresponding to a point on the plot. We will simply drop a vertical line from the point down to the x-axis to get the *price* and draw a horizontal line from the point to the y-axis to determine the *sales_qty*. The figure will look similar to the prior two figures.

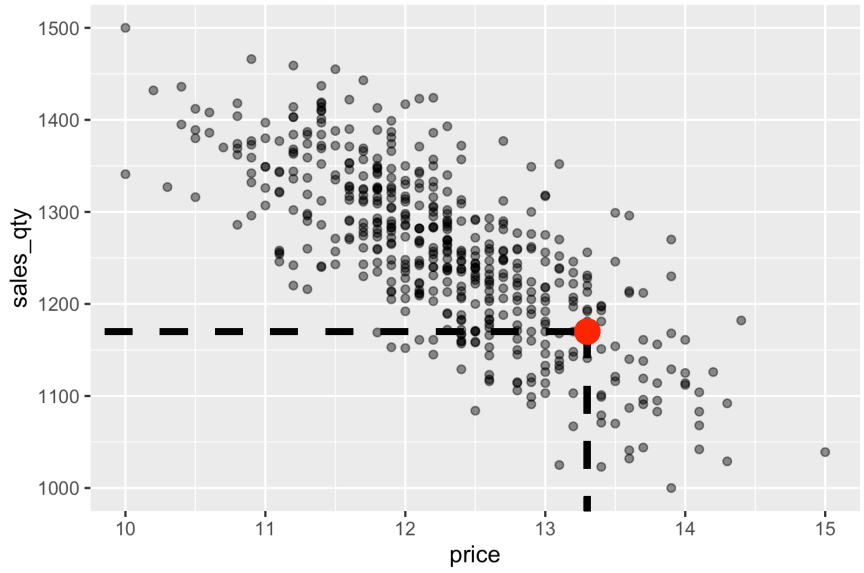

Figure 2.3: Finding the eleventh row of the *price_demand* data frame on the plot: *price = 13.3*, *sales_qty = 1170*

2.1.3 Dissecting the code

Figure 2.4 decodes the various elements of the code that we used to generate out first point plot in Figure 2.1. We see that we pass the data frame *price_demand* to the *point_plot* function through the *pipe* operator. We add the tilde expression for the plot as an additional argument.

Figure 2.5 shows us that the *point_plot* function maps the variable on the left of the tilde operator to the y-axis of the plot and maps the variable to the right of the tilde expression to the x-axis of the plot.

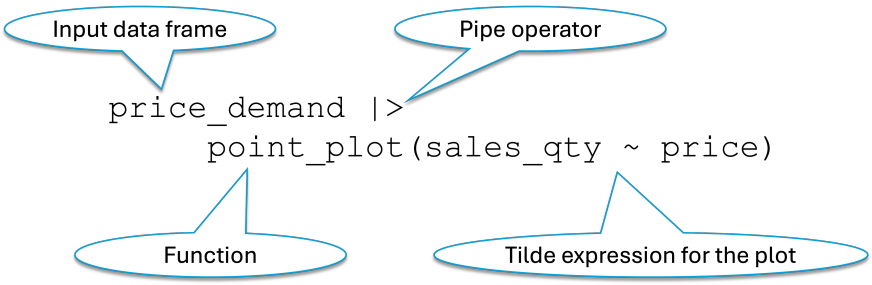

Figure 2.4: Passing the *price_demand* data frame as argument to the *point_plot* function through the pipe operator

> Quick Check
>
> In a y ~ x formula for point_plot, what does the tilde mean operationally?

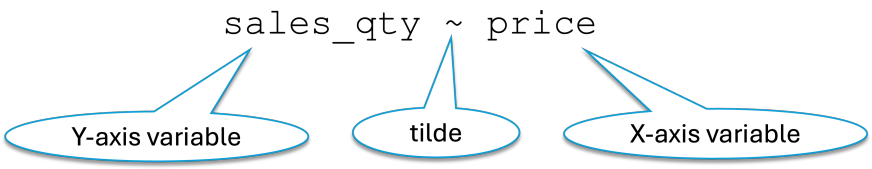

Figure 2.5: Using a tilde expression to map variables to the axes

Suggested answer

It separates the y-variable (left) from the x-variable (right), mapping them to the plot axes.

Quick Check

Write the formula you would use to plot weekly_sales_k on the y-axis against advertising_spend_k on the x-axis. Write only the tilde expression.

Suggested answer

weekly_sales_k ~ advertising_spend_k

Quick Check

If you accidentally write price ~ sales_qty, what changes in the appearance of the plot?

Suggested answer

The axes swap: price becomes the y-axis and sales quantity becomes the x-axis.

2.1.4 Interpreting the plot

Figure 2.6 repeats the plot of *price* versus *sales_qty*.

```
price_demand |>
  point_plot(sales_qty ~ price)
```

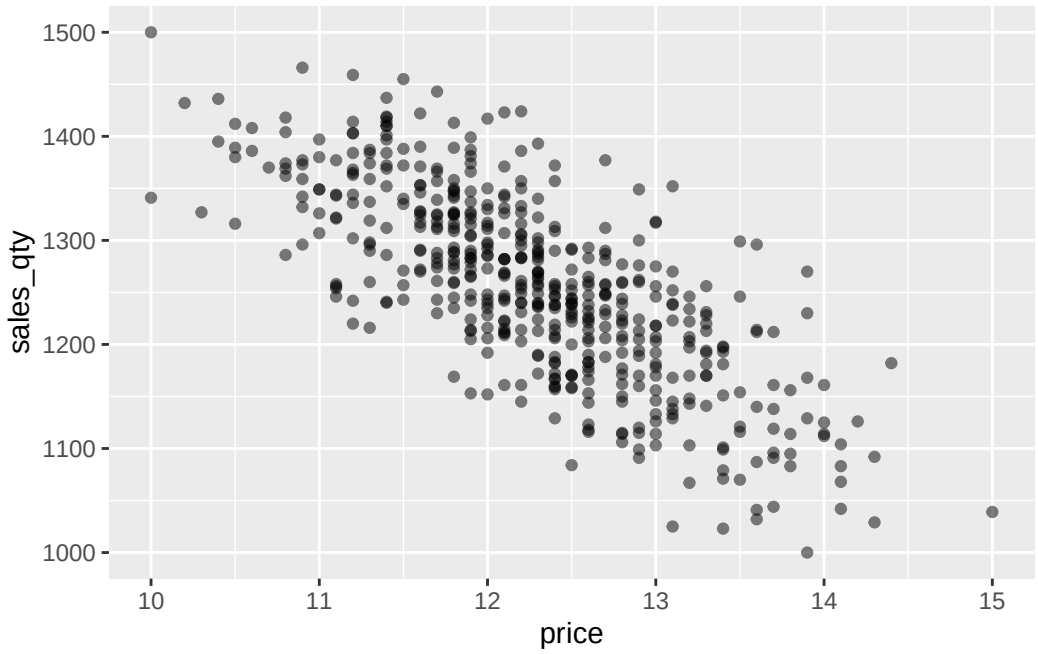

Figure 2.6: Price demand relationship (again)

We can see a clear trend showing that, in general, higher prices correspond to lower values of sales quantity.

To be sure, this is only a trend and not a strict rule. That is, given any two points, we do not have a guarantee that the point with the higher price will always have a lower sales quantity. However, this will be true for most of the pairs of points. Figure 2.7 illustrates this with some randomly selected pairs of points. We see that most of the lines point down and to the right – meaning that for most pairs of points, the one with higher *price* has a lower *sales_qty*. However, we also see a few examples where the opposite is true.

This is why we only calls this a *trend* and not a strict rule about the relationship between price and sales_qty.

Quick Check

Describe a quick visual procedure to decide if the relationship is positive, negative, or neither.

Suggested answer

Look at the overall slope of the cloud from left to right: up = positive, down = negative, no slope = neither.

Quick Check

In the price vs sales plot, why do we call it a trend and not a rule?

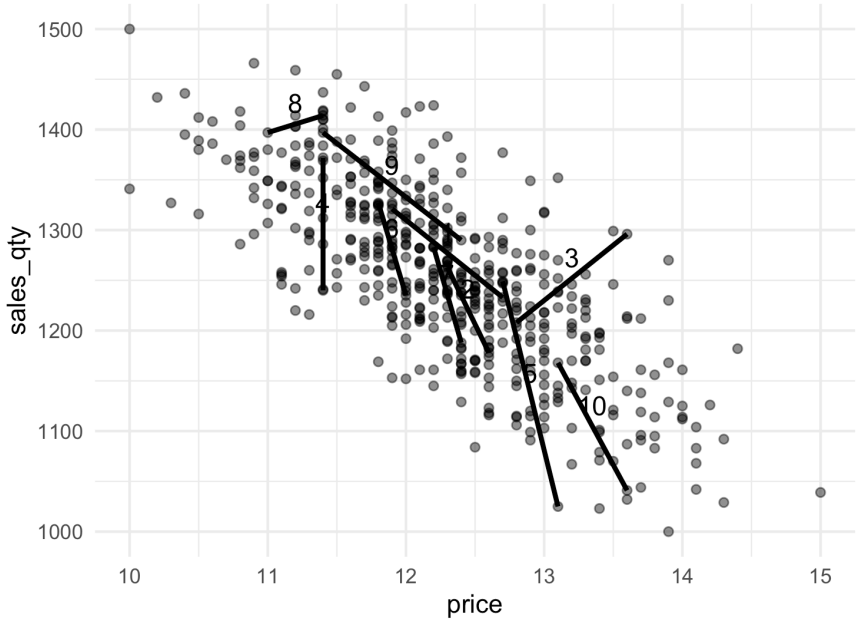

Figure 2.7: Line segments connecting some pairs of points showing that for most pairs of points the point with higher price has lower sales_qty

Suggested answer

Because points vary; higher price usually means lower sales, but not for every single pair of observations.

Quick Check

What would a “no linear relationship” scatterplot look like?

Suggested answer

A roughly pattern less cloud with no clear upward or downward slope.

2.1.5 Positive and negative relationships

In general, when we discuss the relationship between two numerical variables, if the general trend is that higher values of one correspond to higher values of the other, then we have a positive relationship. On the other hand, is higher values of one variable correspond to lower values of the other then we have a negative relationship.

What kind of relationship does Figure 2.6 show – positive or negative?

You got it right if you said *negative* – higher values of *price* correspond to lower values of *sales_qty*.

Figure 2.8 shows two positively correlated variables. Here, higher values of one variable are generally related to higher values of the other. We used the *advertising_sales_channel* data frame for this plot.

```
advertising_sales_channel |>
  point_plot(weekly_sales_k ~ advertising_spend_k)
```

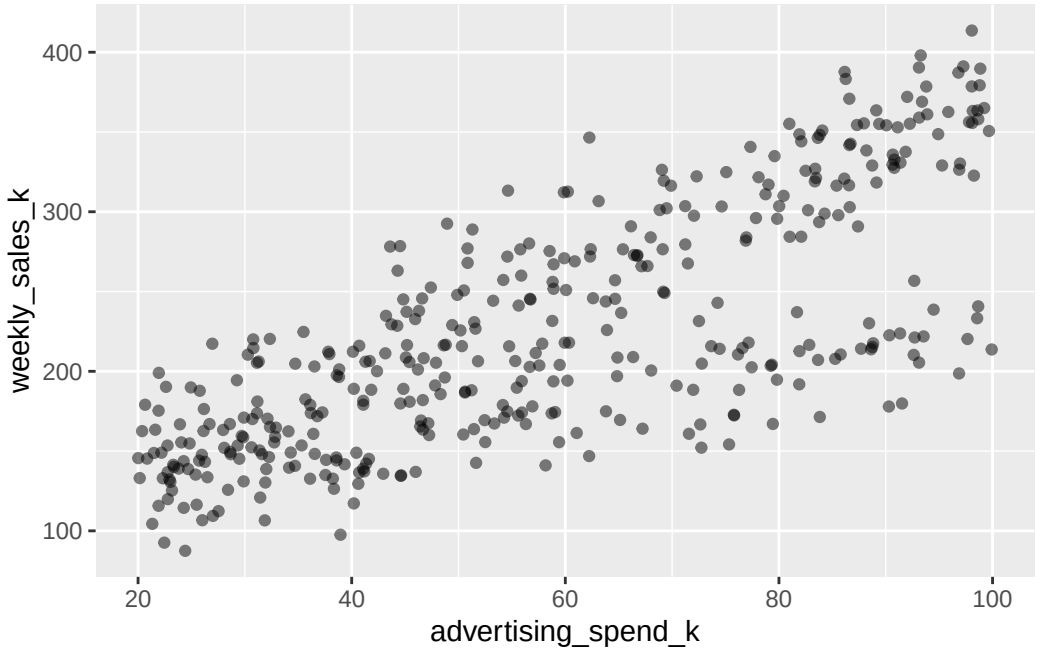

Figure 2.8: Weekly sales and advertising spend have a positive relationship – higher advertising generally has higher weekly sales

The data frame *returns_dpo* contains hypothetical data on various firms' daily stock returns and the number of days they take to pay their suppliers. Figure 2.9 shows the relationship between these variables. We cannot spot any trend. Higher or lower values of one variable do not indicate higher or lower values of the other. Hence there is neither a positive relationship nor a negative one.

```
returns_dpo |>
  point_plot(daily_return ~ dpo)
```

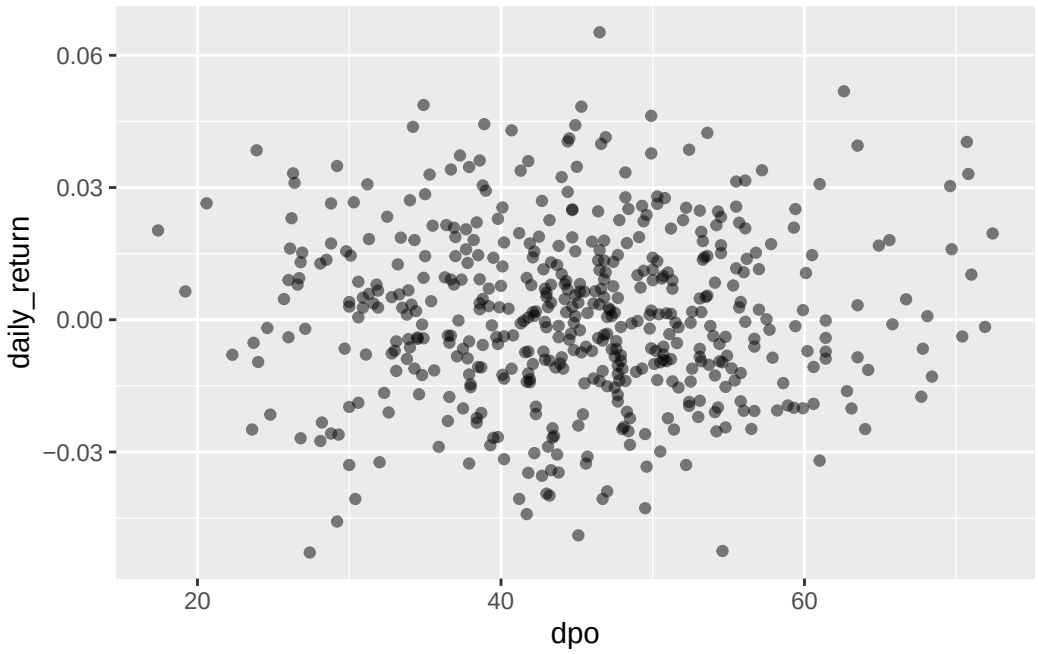

Figure 2.9: Days payment outstanding (dpo) does not have a relationship to the daily stock return

How to decide visually if there's a relationship

Imagine the point cloud as a blob

Ask: does it slope up, slope down, or neither? *Ask:* tight blob or wide blob?

Quick Check

Label each plot as positive, negative, or no clear relationship: (a) price vs sales_qty, (b) advertising vs weekly sales, (c) dpo vs daily return.

Suggested answer

(a) negative, (b) positive, (c) no clear relationship.

Quick Check

If a scatterplot has a positive relationship, what do you expect most left-to-right line segments between random pairs of points to do?

Suggested answer

Point up and to the right.

2.1.6 Putting a number on the relationship: The correlation coefficient

Thus far, we have been talking about relationships between variables. Exact sciences like mathematics and statistics like to be precise and assign numbers where possible. Statistics commonly uses the *correlation coefficient* to quantify the relationship between two numerical variables. This section just introduces the *correlation coefficient* and shows you how to compute it using R. We will get into the mechanics of the actual computation later in the book.

The correlation coefficient can take a value between -1 and +1. A value of 0 signifies *no linear relationship*. -1 signifies a perfect negative correlation and +1 signifies a perfect positive correlation. We discuss these ideas below.

Quick Check

What does the sign of the correlation coefficient tell you?

Suggested answer

Direction: positive means upward trend (as we go to the right); negative means downward trend (as we go to the right).

Quick Check

What does the magnitude (absolute value, that is, value without the sign) tell you?

Suggested answer

Strength of the linear pattern: closer to 1 means points lie closer to a straight line.

Quick Check

True or False: r = 0 proves there is no relationship of any kind.

Suggested answer

False. It indicates no *linear relationship*; there could still be a curved or other non-linear relationship. We show an example later.

We look at some examples now. Revisiting the *price_demand* data frame, let us use R to compute the correlation coefficient between the variables *price* and *sales_qty*.

```
price_demand |>
  summarize(price_sales_qty_correlation = cor(price, sales_qty))

# A tibble: 1 x 1
  price_sales_qty_correlation
                        <dbl>
```

```
1                    −0.752
```

We are computing a summary and hence use the *summarize* function (see Section 1.7.6). Since the summary we are computing now is the correlation coefficient, we use the *cor* function within the *summarize* function. We have chosen to call the computed result *price_sales_qty_correlation*. We could have named it anything we wanted, but chose the sensible approach of giving it a meaningful name.

Figure 2.1 had already shown us the negative relationship between *price* and *sales_qty*. The correlation confirms it, ... and makes it more precise.

> Quick Check
>
> If the scatterplot slopes down and the correlation is -0.75, do these agree?

> Suggested answer
>
> Yes. Negative sign matches the downward trend; 0.75 magnitude suggests a fairly strong linear relationship.

> Quick Check
>
> If two variables have correlation coefficient close to +1, what will the scatterplot look like?

> Suggested answer
>
> Points clustered tightly around a line sloping upward.

Let us compute the correlation coefficient between advertising spend and the weekly sales from the *advertising_sales_channel* data frame. From Figure 2.8 we saw that these two variables are positively related. Let us see what the correlation coefficient says.

```
advertising_sales_channel |>
  summarize(ad_sales_corr = cor(advertising_spend_k, weekly_sales_k))
```

```
# A tibble: 1 x 1
  ad_sales_corr
          <dbl>
1         0.760
```

Sure enough, we see a positive correlation coefficient.

2.1.7 Perfect correlation

Consider Figure 2.10 showing strong, but imperfect, correlation.

```
price_demand |>
  point_plot(sales_qty ~ price)
```

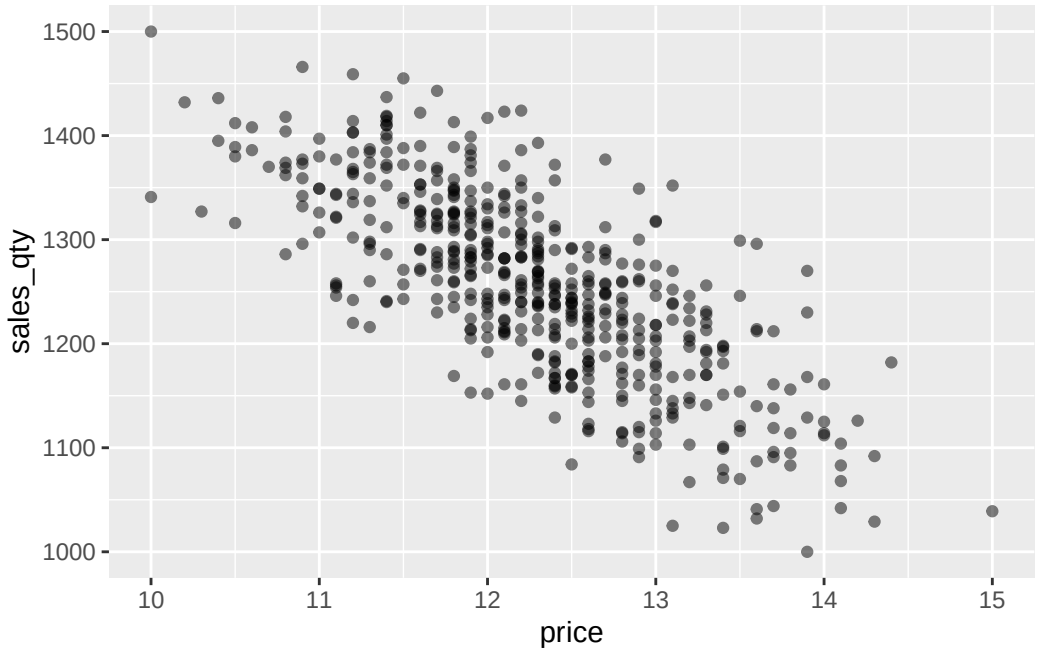

Figure 2.10: Price and sales_qty are not perfectly correlated – correlation coefficient is -0.75

We have already established that *price* and *sales_qty* are negatively related. That is, there is a general pattern that higher values of *price* have lower values of *sales_qty*.

In Figure 2.10, if we look only at the points corresponding to a particular value of *price*, say 13, the points along this vertical line range from approximately 1100 to 1320. That is, given a value of *price*, we cannot be sure about the exact value of the corresponding value of *sales_qty*. However, knowing *price* does give us some idea about *sale_qty*. This shows relationship, but not perfect correlation.

When the correlation coefficient is +1 or -1, we have perfect correlation. In this case, knowing the value of one variable enables us to precisely know the value of the other variable as well. This happens when all the points lie on a straight line instead of being dispersed as a cloud as in Figure 2.10.

Figure 2.11 shows a situation when all points lie on a perfect straight line. In this case, given a value for one variable, we can determine exactly the value of the other. We can determine it visually or by plugging the value of the known variable into the following equation and calculating the value of the other variable.

```
total_cost = 10000 + 45*units_produced
```

In Figure 2.11 the line slopes upward as it goes from left to right. When the line on which all points fall slopes down as it goes right, the correlation is still perfect, but the line slopes down as it goes to the right and so we have a correlation coefficient of -1.

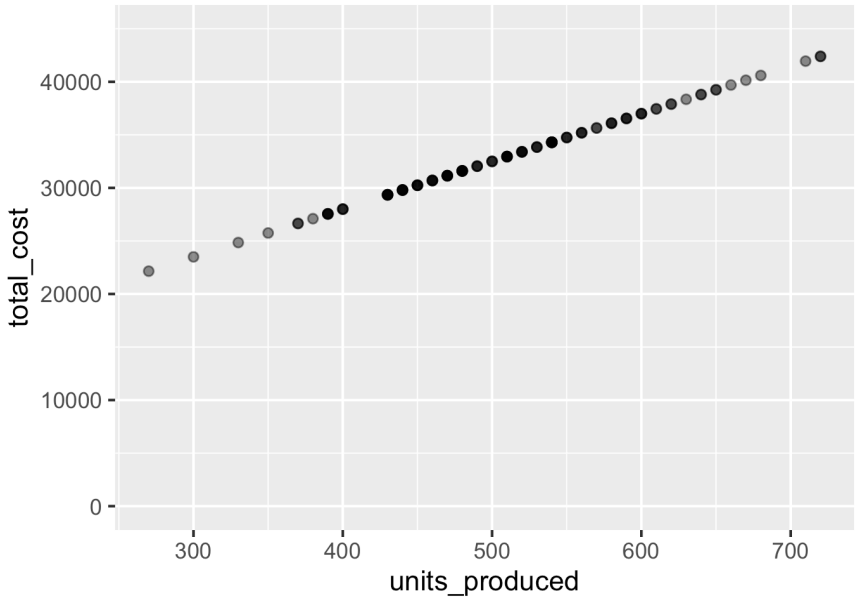

Figure 2.11: Perfect correlation arises when all the points lie perfectly on a straight line – the variables in this plot have a correlation coefficient of +1

2.1.8 How the point cloud looks for various correlation coefficients

Figure 2.12 shows the point plots for various values of correlation coefficients between -1 and +1.

You can see the following: - in the plot with correlation coefficient = -1 (r = -1), the points align perfectly on a straight line *sloping downwards* as the x values increase - at +1 they align perfectly on a line *sloping upwards* as the x values increase - as the coefficient goes from -1 to 0, the points get more scattered away from perfect alignment on a straight line until at 0 there is no pattern whatsoever - as the correlation coefficient increases from 0 to +1, we see that the tendency to align on a straight line increases until perfect alignment at r = 1.

! Magnitude and sign of the correlation coefficient

Sign: Direction (positive/negative)
Magnitude: Tightness around a line (near 0 = no linear pattern; near ś1 = points nearly on a line)

Quick Check

What does “perfect correlation” mean visually?

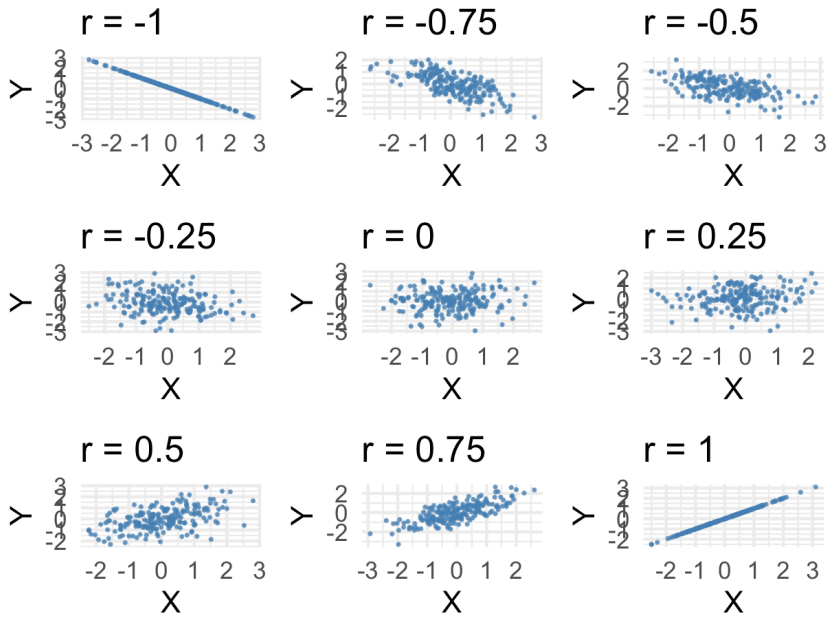

Figure 2.12: Point plots for various correlation-coefficients (r)

Suggested answer

All points lie exactly on a straight line.

Quick Check

If correlation is -1, does that mean the variables are weakly related?

Suggested answer

No. It means perfectly related but in a negative (downward) direction.

Quick Check

As r moves from 0 toward 1, what changes visually?

Suggested answer

The point cloud becomes more tightly aligned around an upward-sloping line.

! Correlation vs. *Causation*

In Figure 2.10, we can see that as *price* increases, *sales_qty* decreases. We have said that these two variables are *correlated*. From this alone, can we infer that higher *price* **caused** lower *sales_qty*?
We need to be very careful here. Figure 2.10 only shows the *correlation*. From that correlation alone we should infer anything more. Specifically, we cannot infer that the *price* changes **cause** changes in *sales*.
Of course, in the real world, we know that price increases generally lead to drops in sales. But that knowledge lies outside of this data. It is quite possible that, in this situation something else might be playing a role. We just don't know from the data we are seeing. Establishing **causation** is part of *causal inference* which uses other rigorous techniques to establish that something actually caused something else. We do not go into causal inference in this course.

2.2 Plotting a numerical variable against a categorical one

In this course, we will only assign a numerical variable to the y-axis. Thus, if a plot has a categorical variable, it will only occupy the x-axis.

Numerical and categorical variables differ in how they behave in plots. In Figure 2.8 we have the variable *advertising_spend_k* on the x-axis. Being a numerical variable, it can potentially take on any value on the x-axis. On the other hand, look at Figure 2.13.

Here we have the variable *bank_account_type* on the x-axis and it can take on only one of two values *"Checking"* and *"Savings"*. This is why each row falls at only one of two x positions: one for *Checking*, and one for *Savings*. So points stack vertically. If the row corresponds to a "Checking" account, it falls on the left stack of points and on the right stack otherwise.

2.2.1 A Caveat

Correlation measures the strength of a *linear relationship*; a curved pattern can have low correlation coefficient even if there is a strong relationship.

Figure 2.14 shows a strong non-linear (loosely means "not a **straight** line") relationship. The points align very close to a curved line. This has a low correlation coefficient because if you try to run a straight line through these points, no matter how hard you try, you cannot draw a line that is even reasonably close to all points. The correlation coefficient is -0.125 – close to zero!

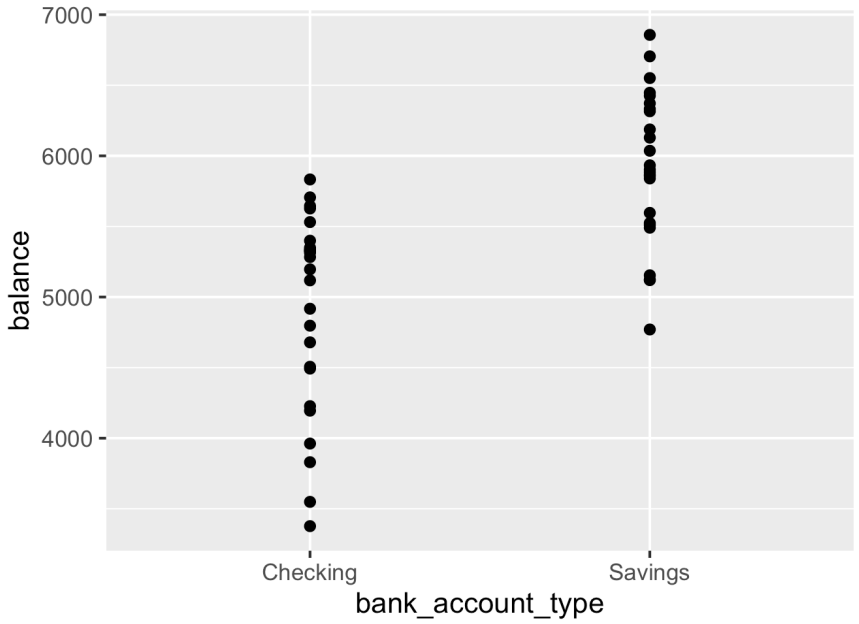

Figure 2.13: Plotting a categorical variable against a numerical one – using the data frame *acct_type_balance* (not plotted using the *point_plot* function)

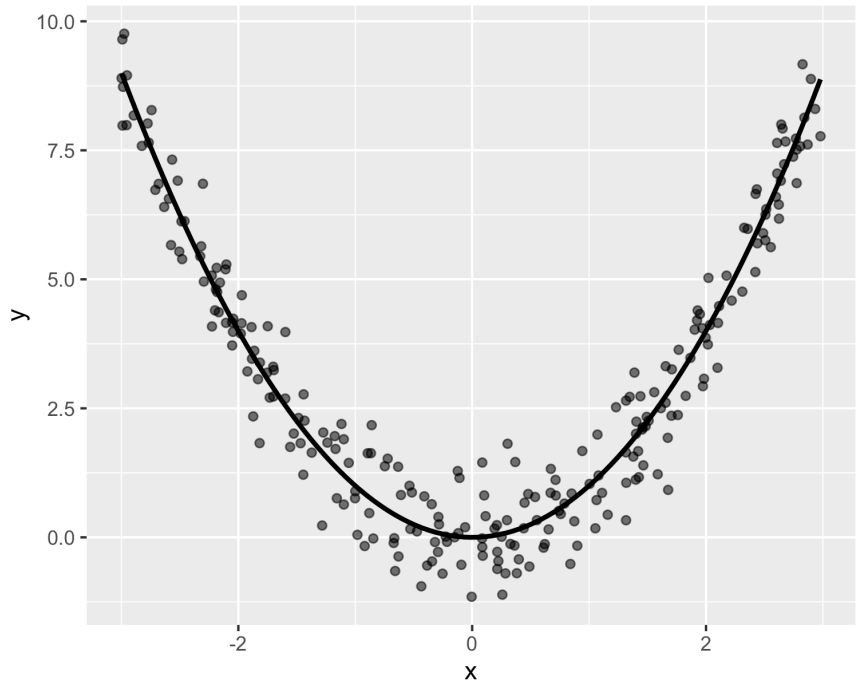

Figure 2.14: Strong non-linear relationship, but weak correlation coefficient

Quick Check

If *bank_account_type* has two categories, how many distinct x positions exist conceptually?

Suggested answer

Two: one for *Checking* and one for *Savings*.

Quick Check

Why do points form vertical "stacks" for each category?

Suggested answer

Because many rows share the same category (same x position) but differ in balance (y).

2.2.2 *point_plot* and "jittering"

If we have two *Checking* accounts with the same or very similar balance, their points will overlap. Because of this over plotting, the plot in Figure 2.13 does not enable us to look at all the points.

To overcome this problem, the *point_plot* function plots this chart a bit differently. See Figure 2.15.

Important: Jittering changes only the display position of points to reduce overlap; it does not change the underlying data values

```
acct_type_balance |>
  point_plot(balance ~ bank_account_type)
```

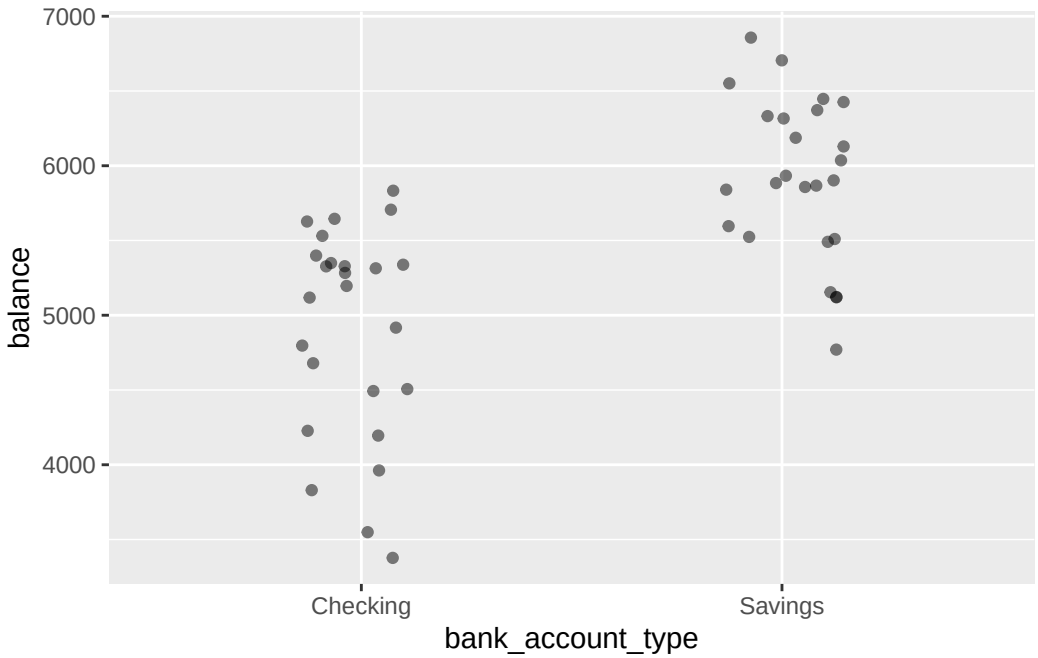

Figure 2.15: Plot of account type against account bakance using the *point_plot function*: it jitters the points along the x-axis to reduce overplotting

To enable us to see the points more clearly, the *point_plot* function randomly "jitters" the points along the x-axis to more clearly separate points that might be over plotted or close together.

Quick Check

What problem does jittering solve?

Suggested answer

Over plotting. Points hiding under other points.

Quick Check

Does jittering change the data?

Suggested answer

No. It only changes where points are drawn to make overlap visible.

Quick Check

In a jittered plot, if we see two points with the same y-value and the same x-category, what would have happened to them on an un-jittered plot??

Suggested answer

They would have overlapped and we would only have a single point visible.

2.3 Bringing a third variable into the plot

Two-dimensional plots can normally accommodate only two variables – one on each axis. However, we can often get more insights if we manage to get more variables into our plots. The *point_plot* functions allows us to bring one more dimension to our plots through color.

In Figure 2.8, we saw the positive relationship between *weekly_sales_k* and *advertising_spend_k* in the *advertising_sales_channel* data frame. That data frame has another variable *channel*. Each row corresponds to one of two sales channels – *Search Ads* and *Retail Promo*. We might want to study separately for each channel the relationship between advertising spend and the weekly sales.

```
advertising_sales_channel |>
  point_plot(weekly_sales_k ~ advertising_spend_k + channel)
```

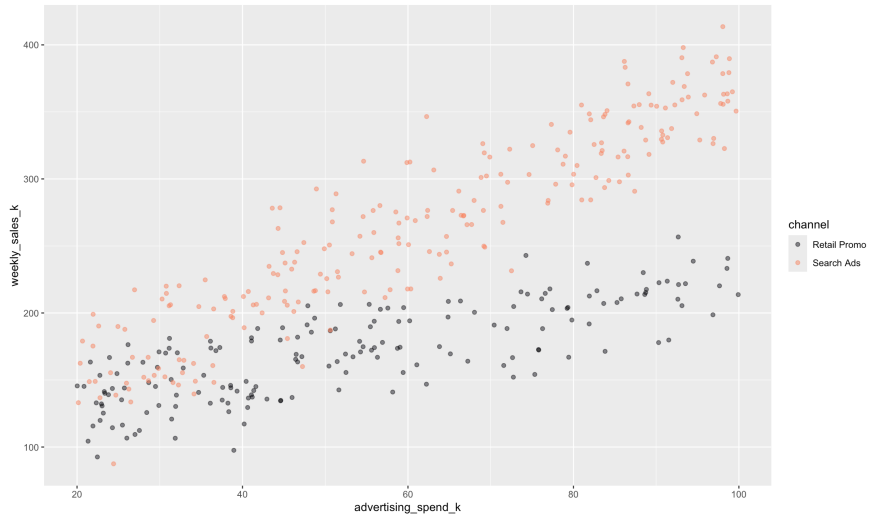

Figure 2.16: Determining the color of points based on a third variable – the last variable in the tilde expression for the plot achieves this

Figure 2.16 inserts, through color, the variable *channel* into the plot of *weekly_sales_k* against *advertising_spend_k* from the *advertising_sales_channel* data frame. From it we can clearly see that the points corresponding to each sales channel clearly shows the positive relationship that we saw earlier. But this plot also shows that for a given value of *advertising_spend_k* the *Retail Promo* channel generally has a comparatively higher *weekly_sales_k* value. Adding the variable *channel* through color enabled us to see more into the data than before.

2.3.1 Understanding the tilde expression to add a third variable

Figure 2.17 shows the tilde expression we used in a point plot of two variables to color the points based on a third variable.

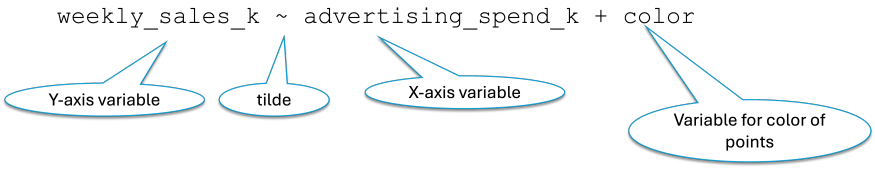

Figure 2.17: To color the points based on the value of a third variable, we just *add* it to the end of the tilde-expression after a plus sign

Figure 2.18 shows another example. In this, we use the *mpg* data frame which contains data on cars. We plot the city mileage of cars (variable *cty*) against their engine displacement (variable *displ*). We color each point based on the class of the vehicle (variable *class*).

```
mpg |>
  point_plot(cty ~ displ + class)
```

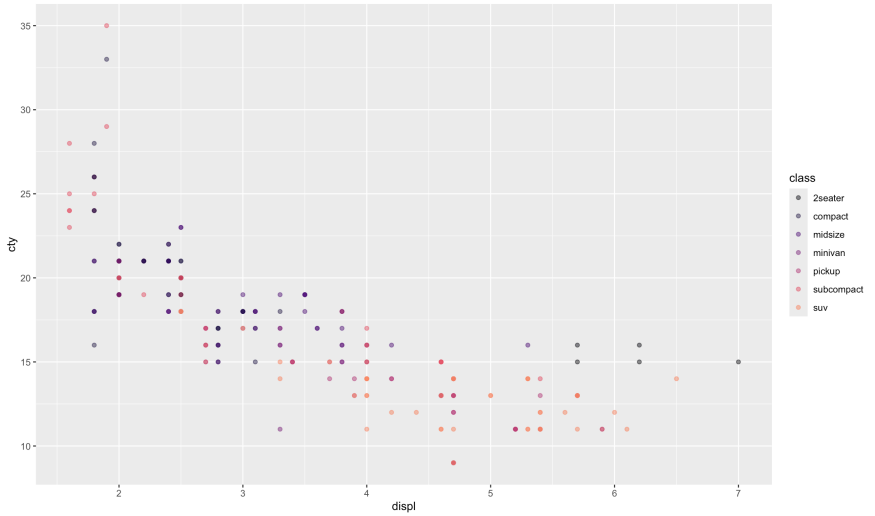

Figure 2.18: The scatterplot shows the relationship between *displ* and *cty* from the *mpg* data frame – the variable *class* in the tilde expression determines the color of the points

Quick Check

In weekly_sales_k ~ advertising_spend_k + channel, what does channel control?

Suggested answer

The color of points.

Quick Check

What new insight can color reveal that the two-variable plot cannot?

Suggested answer

Differences in level or pattern between groups (e.g., Retail Promo tends to have higher sales at the same ad spend).

2.4 Bringing a fourth variable into play

We can bring a fourth variable into play as well through *facets*. Replacing the tilde expression above with:

```
hwy ~ displ + class + drv
```

generates Figure 2.19.

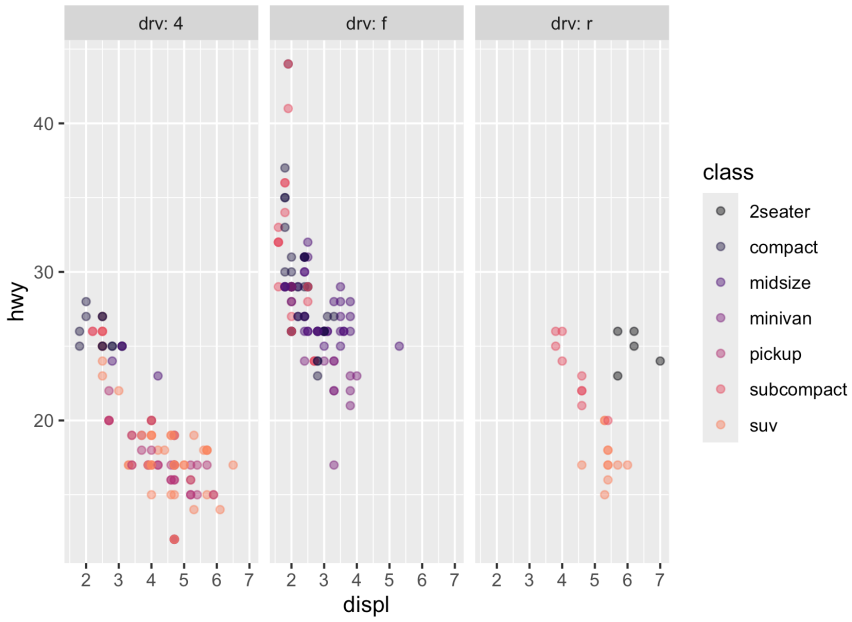

Figure 2.19: The scatterplot shows the relationship between *displ* and *cty* from the *mpg* data frame – the variable *class* in the tilde expression determines the color of the points and the variable *drv* divides the plot into *facets*

In this figure, we use the *mpg* data frame and generate a basic scatterplot of the relationship between the highway mileage (variable *hwy*) and the engine displacement (variable*displ*). The third variable *class* determines the color of the points. The fourth variable *drv* segments the plots into *facets* – one for each possible value of *drv*.

> **Important**
>
> **Tilde expression summary**
> y ~ x y on vertical axis, x on horizontal axis
> y ~ x + color color encodes the third variable
> y ~ x + color + facet facet splits into panels by the fourth variable

Mini-Review: Scatterplots and Relationships

> **Note**
>
> Answer the questions below to check your understanding of how scatterplots represent data and relationships between variables.

Question 1

In a scatterplot of sales_qty ~ price, what does the x-coordinate of a point represent?

> Suggested answer
>
> The x-coordinate represents the value of price for that observation.

Question 2

In the same scatterplot, what does the y-coordinate of a point represent?

> Suggested answer
>
> The y-coordinate represents the value of sales_qty for that observation.

Question 3

If a scatterplot is based on a data frame with 500 rows, how many points appear in the plot?

Suggested answer

500 points — one for each row of the data frame.

Question 4

Describe how you would locate on the plot the point corresponding to a row with price = 12.5 and sales_qty = 1200.

Suggested answer

Locate 12.5 on the x-axis, move vertically until reaching 1200 on the y-axis, and mark the intersection.

Question 5

What visual pattern indicates a **negative relationship** between two numerical variables?

Suggested answer

A downward-sloping cloud of points: as x increases, y generally decreases.

Question 6

What does it mean if a scatterplot shows **no clear trend**?

Suggested answer

There is no apparent linear relationship — higher or lower values of one variable do not predict higher or lower values of the other.

Question 7

What does the **sign** of the correlation coefficient tell you?

> Suggested answer
>
> It tells you the direction of the linear relationship: positive for upward trends and negative for downward trends.

Question 8

A correlation coefficient close to 0 means what?

> Suggested answer
>
> There is little or no **linear** relationship between the variables, even though a nonlinear relationship may still exist.

Question 9

Why does `point_plot()` jitter points when the x-axis variable is categorical?

> Suggested answer
>
> To reduce over plotting so that points with the same category and similar values do not overlap visually.

Question 10

In the formula
`weekly_sales_k ~ advertising_spend_k + channel`,
what role does channel play in the plot?

Suggested answer

It determines the color of the points, allowing different groups to be distinguished visually.

Question 11

What additional role does a fourth variable play when added as
y ~ x + color_var + z?

Suggested answer

It divides the plot into separate panels (facets), one for each value of the fourth variable.

Part II

Variation

Module 2: Variation

Variation plays a very important role in statistics. Some even go to the extent of viewing the discipline of statistics itself as the study of variation.

This module discusses variation generally by visualizing it through *violin* plots. It then introduces the key statistical concepts of *variance* and *standard deviation* and how to compute these using R.

Learning Goals

- Explain what variability in a variable represents
- Plot and interpret violin plots
- Identify broad characteristics of distribution shapes
- Compute variance and standard deviation of a small set of numbers by hand
- Use R to compute the variance and standard deviation of variables in a data frame
-

Structure of This Module

- Violin plots
- Variance and standard deviation

3 Using Violin Plots to Visualize Distribution

This chapter talks about visualizing variability.

Learning outcomes

After completing this chapter you will be able to:

Demonstrate these outcomes related to shapes of distributions:

- Given a distribution as histogram, explain what the height of a bar represents
- Given a distribution as a violin plot, explain what the width of a violin at any point represents
- Given a violin plot, identify the point(s) having the highest and lowest densities
- Identify whether a given distribution shown either as a histogram or as a violin plot or as a density plot is
  - uniform
  - symmetric bell shaped
  - multi-modal
  - skewed and in which way

Demonstrate these outcomes related to generating violin plots using R:

- Given a data frame and a numerical variable, write R code to generate a violin plot to study the distribution of the variable
- Explain the two important parts of the plot area in a violin plot
- In the plot, identify which part represents the data and which part represents the *annotation*
- Explain the tilde expression used to generate violin plots of single variables
- Generate a violin plot to compare the distributions of a single numerical variable corresponding to a categorical variable

We have already looks at the concept of *variable*. We have used the term to refer to a column of a data frame. We explained the rationale for the name from the fact that the values in a single column of a data frame *vary*. That is, not all the rows of a column have the same value. For example, we see in Table 3.1 that the values in the column *minutes* vary – that is, not all the values in the column are the same. The same is true of every column.

Table 3.1: Boston Marathon data (20 randonmly selected rows).

year	name	country	time	sex	minutes
2000	Elijah Lagat	Kenya	02:09:47	male	130
2009	Deriba Merga	Ethiopia	02:08:42	male	129
2015	Lelisa Desisa	Ethiopia	02:09:17	male	129
1970	Ron Hill	England	02:10:30	male	130
1969	Yoshiaki Unetani	Japan	02:13:49	male	134
1966	Kenji Kimihara	Japan	02:17:11	male	137
1977	Michiko (Miki) Gorman	United States	02:48:33	female	169
1995	Uta Pippig	Germany	02:25:11	female	145
1982	Charlotte Teske	Germany	02:29:33	female	150
2015	Caroline Rotich	Kenya	02:24:55	female	145
2006	Robert Kipkoech Cheruiyot	Kenya	02:07:14	male	127
1997	Fatuma Roba	Ethiopia	02:26:23	female	146
1995	Cosmas Ndeti	Kenya	02:09:22	male	129
2010	Teyba Erkesso	Ethiopia	02:26:11	female	146
2000	Catherine Ndereba	Kenya	02:26:11	female	146
1991	Wanda Panfil	Poland	02:24:18	female	144
1957	John J. Kelley	United States	02:20:05	male	140
1922	Clarence H. DeMar	United States	02:18:10	male	138
1964	Aurele Vandendriessche	Belgium	02:19:59	male	140
1903	John C. Lorden	United States	02:41:29	male	161

Take care to note that we are not saying that every value in a column has to be distinct. Values can repeat. However we have variety in a column as long as all the values are not the same. Very rarely will you come across data frames in which all the values of a column are the same. In this case the column does not serve any useful purpose for data analysis.

In this course, we deal only with variability in numerical variables. Statistics deals with variability in categorical variables as well, but this book does not consider this in any depth.

3.1 Distribution

Consider a variable with maximum value of 200 and minimum value 10. Let us suppose that we have 500 values for this variable (500 rows in the data frame). This means that the 500 values are all in the range 10 to 200.

We are interested in knowing how these 500 numbers are *distributed* across the range of 10 to 200.

- It could be the case that the 500 numbers are more or less uniformly distributed over their range with approximately the same number of occurrences throughout the range.

- It could also be the case that the numbers are mostly concentrated in the middle – near 105 or so and as we go further away in either direction, the concentration of numbers drops off.
- Or the numbers could be crowded near, the lower extreme, sat at around 50 or so and be much more sparse away from this point.
- Or it could be that there is heavy concentration around 50 and 150 and less elsewhere.

The pattern of concentration of values of a variable across its entire range is called the **distribution** of the variable.

In this chapter we focus on visualization alone. The next chapter goes into measuring variation precisely.

Quick Check

In your own words, what does it mean to say a variable has a distribution?

Suggested answer

A distribution describes how often different values of a variable occur and how those values are spread out across their range.

Quick Check

Why is it misleading to describe a data set using only a single number like the mean?

Suggested answer

Because very different data sets can share the same mean while having different spreads, shapes, or extreme values.

Quick Check

How does a distribution answer the question: “What values are typical?”

Suggested answer

It shows where values cluster most densely and which values occur rarely.

3.1.1 Uniform distribution

Suppose we have 1000 numbers with each one being an integer between 1 and 100. Some of the numbers in this data might be 45, 23, 46, 89, 1, 23, 45, 56, and so on. If each number occurs exactly 10 times – that is we have 10 ones, 10 twos and so on up to 10 hundreds, we have what is called as the *uniform distribution*. In a uniform distribution, each number in the range of the list occurs the same number of times.

Figure 3.1 shows a *histogram* of a set of numbers. A *histogram* breaks up the whole range of the data points into *bins*. In our example, the numbers range from 1 to 100. A histogram might break this up into some number of *bins*. If the number of bins is 10, then the numbers between 1 and 10 will fall unto 1 bin. The numbers between 11 and 20 into the next bin and so on. In this example, the bin-width is 10 because each bin spans a range of size 10. The x-axis of the histogram shows the number values and a bar for each bin. The y axis shows how many of the numbers fall into each bin and so the pattern formed by the heights of the bars helps us to see the *distribution* of the numbers.

Figure 3.1 shows numbers that are perfectly uniformly distributed. We have used a bin-width of 1 to have each individual integer fall into its own bin. In a *uniform distribution* every number in the overall range of the numbers occurs exactly the same number of times.

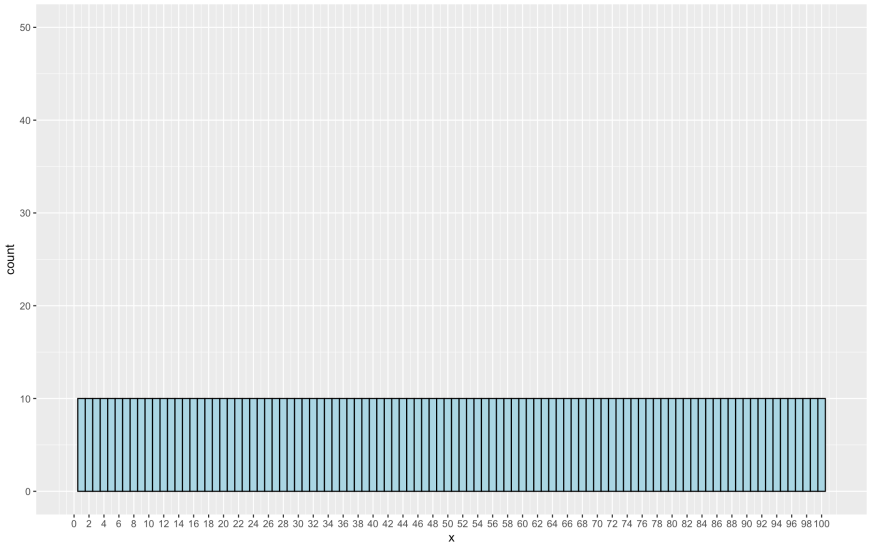

Figure 3.1: Perfect uniform distribution – each number between 1 and 100 occurs the same number of times (10 times), as you can see from the height of the bars

Our current example deals with integers, but in general when we have numerical variables, we will deal with numbers that also have a fractional part. Histograms work in the same way for those too, except that we cannot have one bin for each distinct value as there is potentially an infinite number of values in any range.

Figure 3.2 shows an example of a set of numbers distributed approximately uniformly. For example, Each number does not occur exactly the same number of times, but they are loosely

similar. For example, the number 2 seems to appear around 13 times and the number 25 seems to occur around 18 times. In practice, uniformly distributed numbers will look more like Figure 3.2 than Figure 3.1.

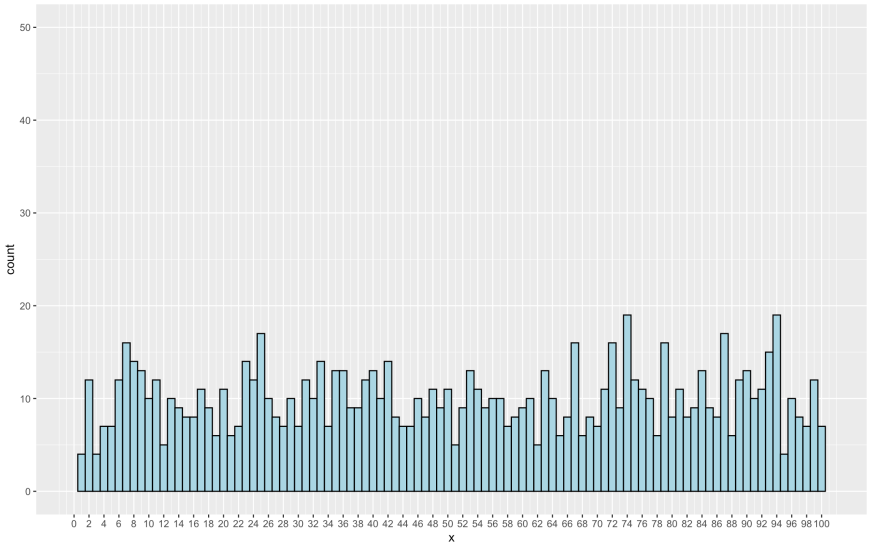

Figure 3.2: Approximate uniform distribution – each number between 1 and 100 occurs more or less the same number of times

A numerical variable does not necessarily have to be distributed uniformly. That is each number in the range does not have to occur an equal number of times exactly or approximately. We will now look at some other common possibilities.

Quick Check

Would you expect to find truly uniform distributions in real business data??

Suggested answer

No. Most business processes involve constraints, preferences, or natural variation that cause some values to occur more often than others.

Quick Check

Give one business example where a roughly uniform distribution might appear.

Suggested answer

Randomly assigned identification numbers (like school student ids) or lottery numbers.

3.1.2 Bell shaped distributions

Figure 3.3 and Figure 3.4 show two *bell shaped* distributions. Both have their peaks around the middle of the range between 1 and 100. Which means that numbers closer to the middle – or closer to 50 – occur more frequently than those far away from 50.

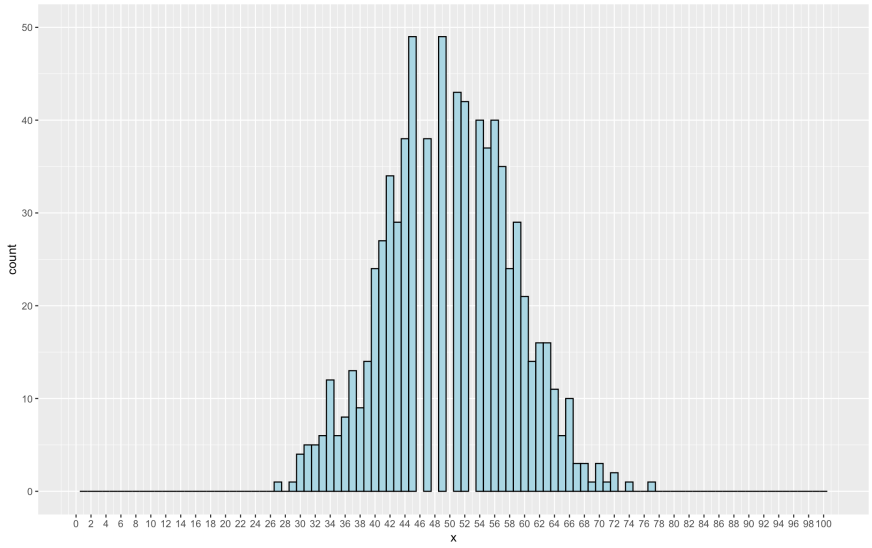

Figure 3.3: Symmetric bell shaped distribution with a peak at around 50

Both Figure 3.3 and Figure 3.4 show bell shaped distributions that are also symmetric. That is, the shape to the left of the peak is approximately similar to the shape on the right.

So, both of these are bell shaped and symmetric, and yet they are obviously different? Can you describe how they are different?

We see that the spread* of the numbers is quite different. In Figure 3.3 the numbers are more concentrated near the middle than is the case in Figure 3.4. Consequently, the peak in the first figure is higher. For example, the numbers 47 and 51 occur nearly 50 times. In Figure 3.4, the highest frequency is around 35. We call the ends of the bell shape as *tails*. We see that Figure 3.4 has *fatter* tails.

We have seen three different possibilities for how a set of 1000 numbers in the range 1 to 100 can be distributed. Even though the averages of a set of numbers and even their range might be the same they can still be *distributed* very differently.

3.1.3 Multi-modal distributions

In Figure 3.3 and Figure 3.4 we saw distributions that each had a single peak. This does not have to be the case. shows a case where we have two peaks.

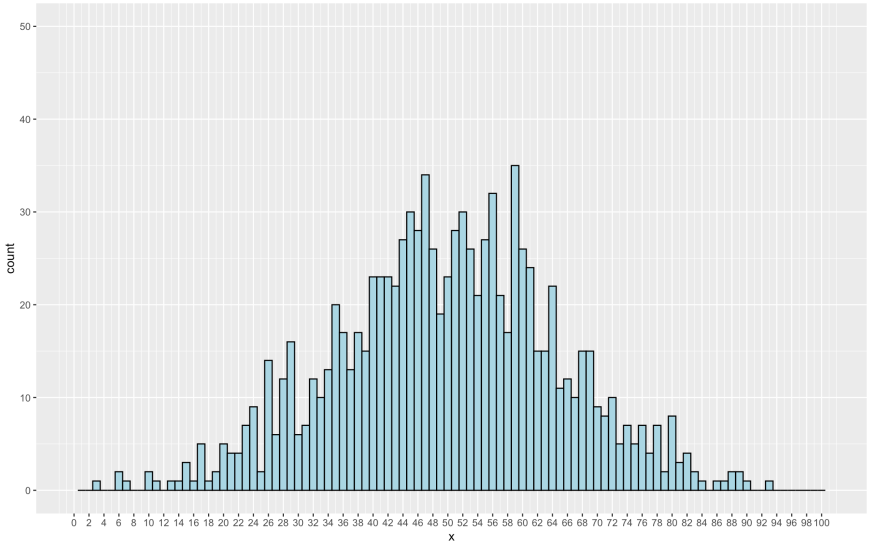

Figure 3.4: Symmetric bell shaped distribution with a peak at around 50 and higher spread than in Figure 3.3

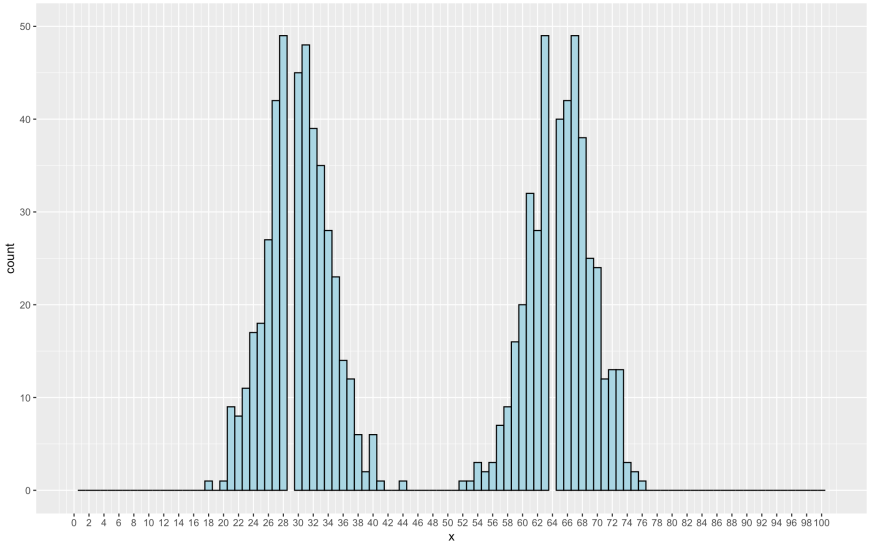

Figure 3.5: Distribution with more than one peak – multi-modal distribution

3.1.4 Skewed distributions

Thus far, we have seen distributions in which the shape has always been symmetric – explicitly so in the bell-shaped examples, but also true for the *uniform* examples. Figure 3.6 and Figure 3.7 show examples of asymmetric distributions. The first has a long *tail* on the right and is said to be *right-skewed* and the second has its *tail* on the left and is said to be *left-skewed*.

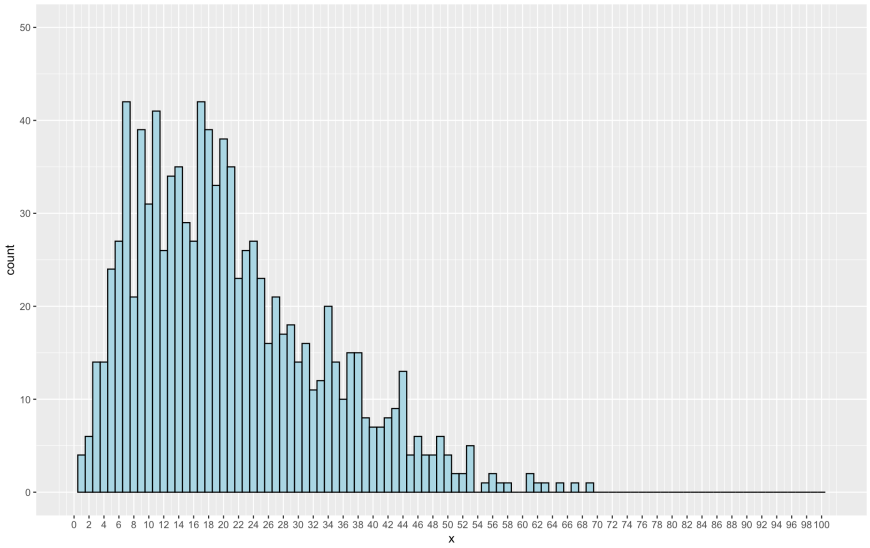

Figure 3.6: Asymmetric distribution with a right skew

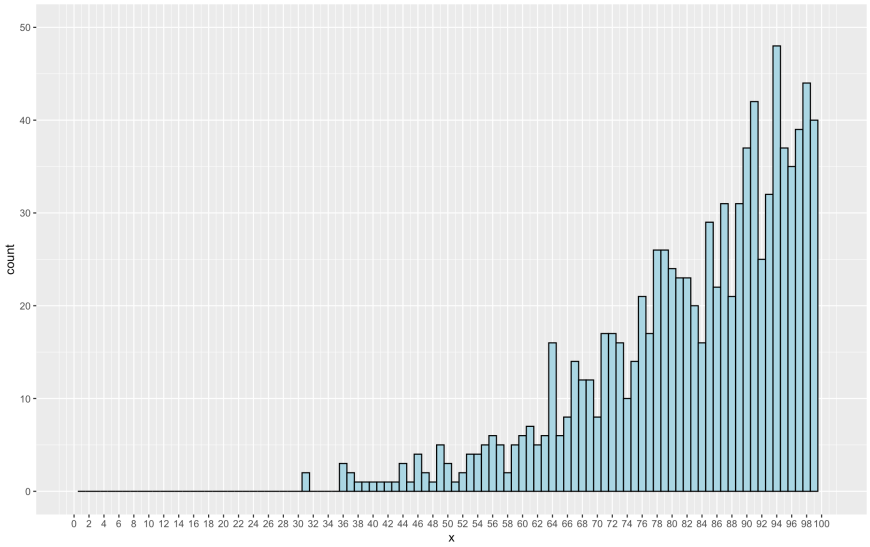

Figure 3.7: Asymmetric distribution with a left skew

3.2 Violin plots

Now that we have clarified the idea of a *distribution* let us look at distributions of numerical variables through *violin* plots. These plots serve the same function as the histograms we have looked at in the previous section, but we can do more.

> A violin plot is not a new idea. It is a different lens on the same distribution concepts you already saw in the previous section.

Let us start off by looking at a violin plot of the variable *minutes* from the *Boston_marathon* data frame

```
Boston_marathon |>
  point_plot(minutes ~ 1, annot = "violin")
```

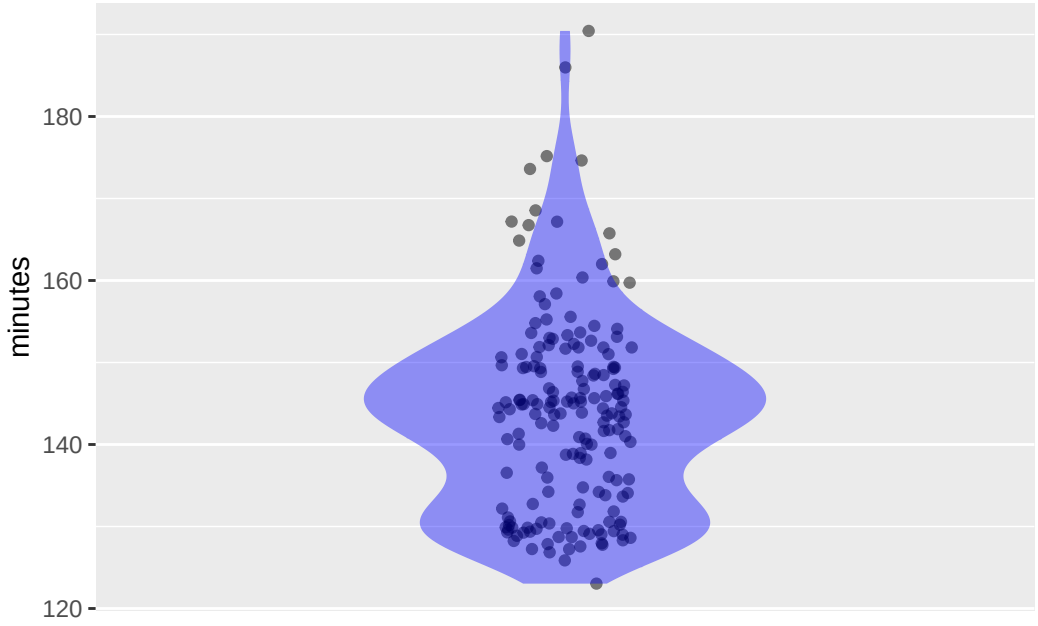

Figure 3.8: Violin plot of the *minutes* variable from the *Boston_marathon* data frame

The width of the violin at any point on the y-axis shows the **relative frequency** of data values close to that point. The widest part of the violin occurs at around 145 minutes. This means that the maximum concentration of winning times in the data set is close to 145 minutes. The plot does not tell us exactly what the frequency is. The width only shows the relative frequency. Looking at where the violin is very narrow, we can say that very few runners took more than 175 minutes to complete the race. Also, relatively low numbers of people took less than about 125 minutes. We can see a higher density of points around the broad regions of the violin and a low density in the narrow areas of the violin.

> ⚠ Important:
>
> The width of a violin **does not represent the number of observations at an exact value**. It represents a **smoothed estimate** of how densely values occur near that value as compared to the density of values near other values.

3.2.1 Elements of the plot

Until now, the plots we have generated using *point plot* have only shown the actual points. However, in Figure 3.8 we see two distinct elements:

- a jittered plot of the individual points
- a violin-shaped solid area – an *annotation* encompassing the points themselves

The points represent the raw data and the violin adds an *annotation* that interprets the data in some way. Here, the interpretation is a violin plot that helps us visualize the distribution of the values.

3.2.2 Examining the code

We examine the code now. In the code for Figure 3.8, we can see two arguments for the *point_plot* function. The first argument specifies the tilde expression mapping the axes. The second specifies the type of *annotation* we want. Figure 3.9 shows the code with the two arguments labeled.

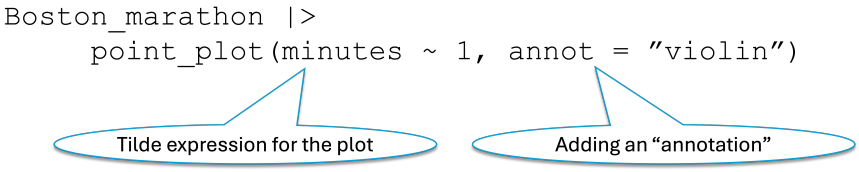

Figure 3.9: The code to generate Figure 3.8 passes two arguments to the *point_plot* function – a tilde expression and an *annotation* specification

Figure 3.10 explains the tilde expression used for the plot. Mapping of *minutes* to the y-axis conforms to what we did before. This plot deals with just a single variable – we have no variable on the x-axis. When there is no variable on the x-axis, we conventionally that by just using “1” for the x-axis variable.

Figure 3.11 explains the second argument in generating a *violin plot*.

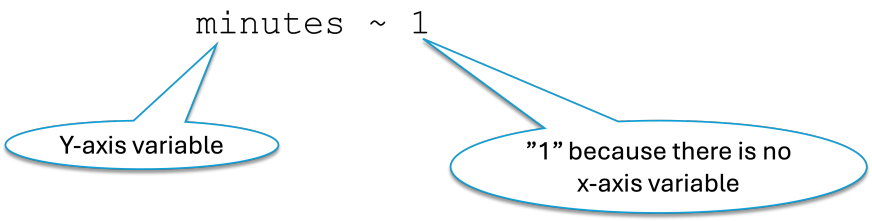

Figure 3.10: The tilde expression of Figure 3.9 we place *minutes* on the y-axis and since the x-axis has no variable, we just state "1"

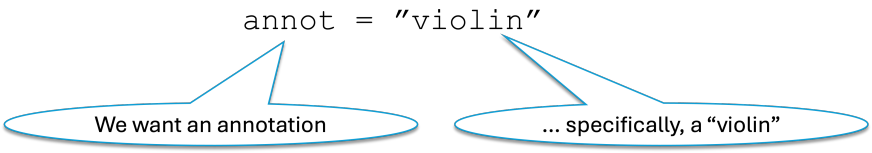

Figure 3.11: We specify that we want a *violin annotation*

3.3 Violin plots and distribution shapes

The violin plot displays vertical reflectional symmetry. We can get all the information that the plot conveys just by slicing the plot vertically down the middle and looking at either of the two chunks.

When we first talked about the various distribution shapes like *uniform*, *symmetric bell-shaped*, and *skewed*, we used histograms to convey the ideas. We can deduce the same information from violin plots as well. In the next section, we will see a few more examples. We will discuss distribution shapes in the context of those examples.

What to look for in a violin plot

- Where are most values concentrated?
- Are there one or multiple peaks?
- Is the distribution symmetric or skewed?
- Are there long tails or unusual shapes?

3.3.1 More examples

We see a few more examples to reinforce the idea. Figure 3.12 shows a violin plot of the variable *price* from the *price_demand* data frame.

```
price_demand |>
  point_plot(price ~ 1, annot = "violin")
```

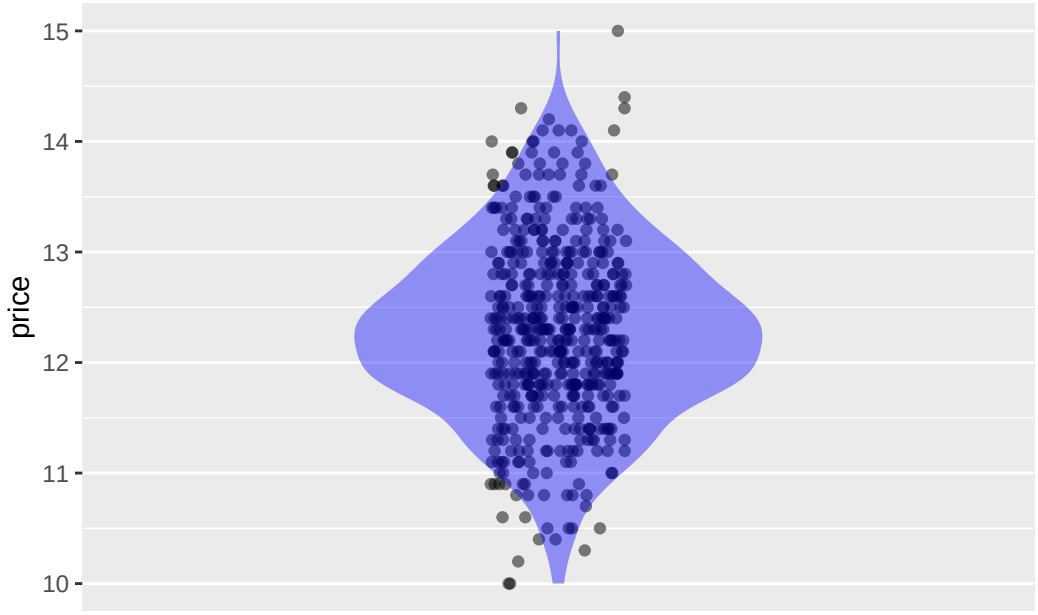

Figure 3.12: Violin plot of the *price* variable from the *price_demand* data frame

If we look at only the left or right half of Figure 3.12, turn it right by 90 degrees and get rid of the display of the individual points, we get Figure 3.13 – a symmetric bell-shaped distribution.

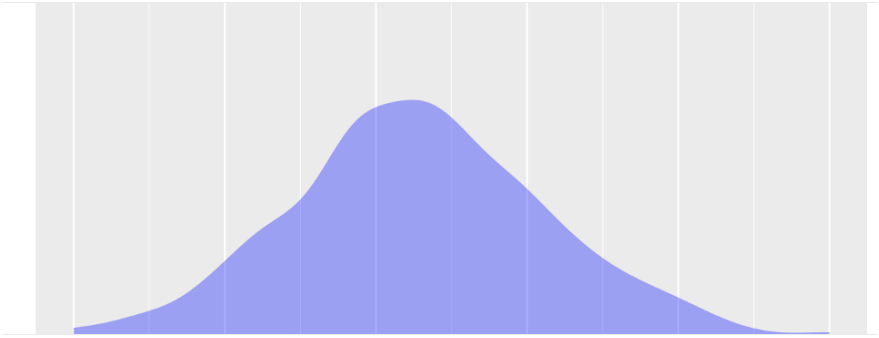

Figure 3.13: Violin from Figure 3.12 split and rotated 90 degrees right to reveal the distribution in the form we had seen before

Figure 3.14 shows a violin plot of the *balance* variable from the *acct_type_balance* data frame.

```
acct_type_balance |>
  point_plot(balance ~ 1, annot = "violin")
```

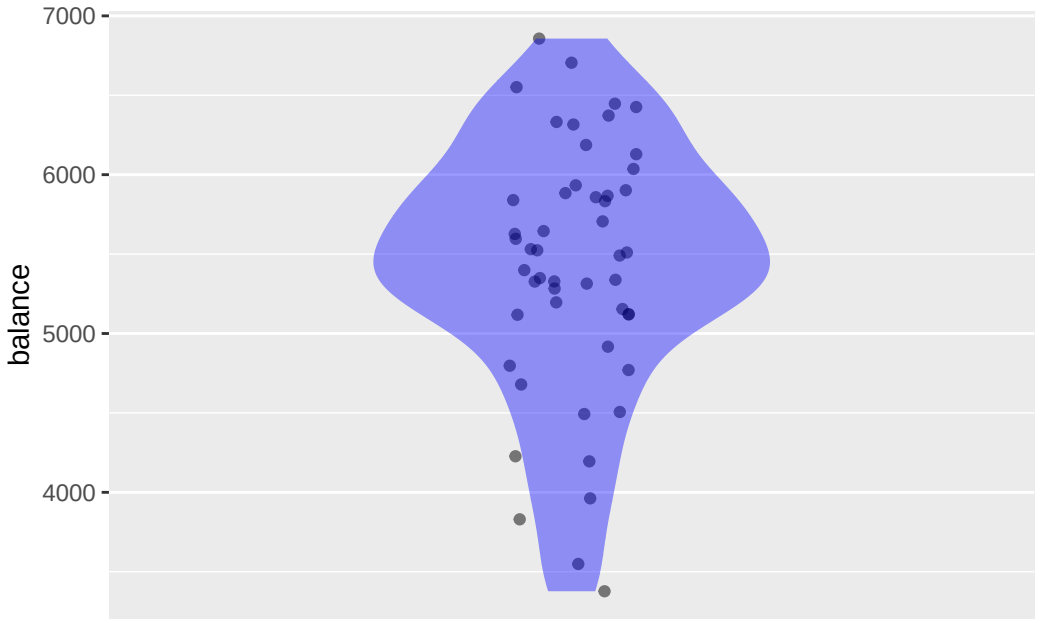

Figure 3.14: Violin plot of the *balance* variable from the *acct_type_balance* data frame

We can see from Figure 3.12 that *price* is bell-shaped and unimodal,but not symmetric. It is skewed towards the lower values because the tail is at the bottom. If we slice and rotate it as before then we would say that it is skewed left.

Next we look at the distribution of *advertising_spend_k* from the *advertising_sales_channel* data frame. Figure 3.15 shows the violin plot. From it we see that *advertising_spend_k* follows a nearly uniform distribution, but not perfectly so. It is mildly bi modal.

```
advertising_sales_channel |>
  point_plot(advertising_spend_k ~ 1, annot = "violin")
```

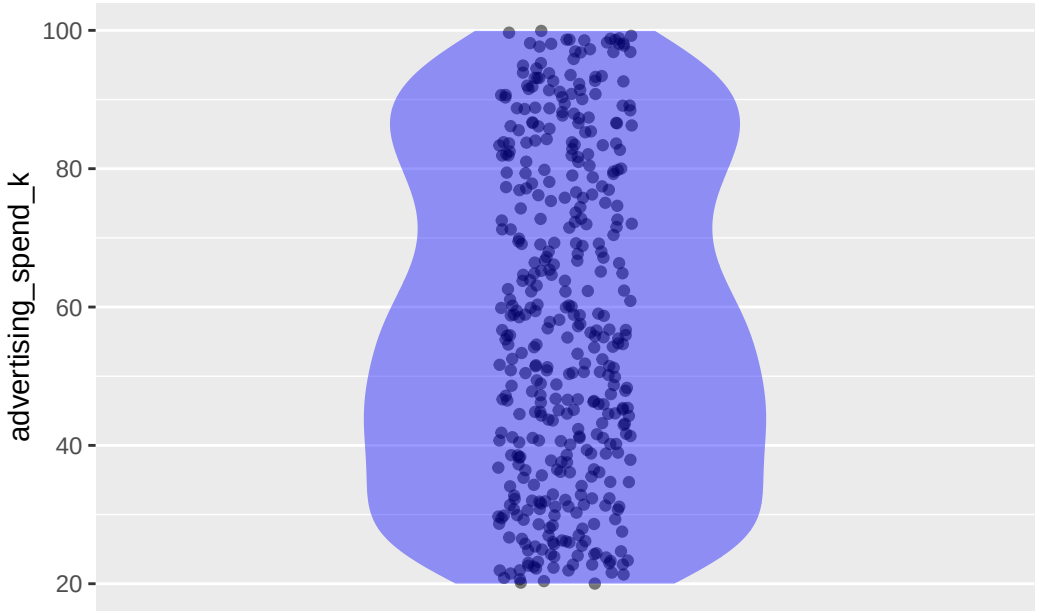

Figure 3.15: Violin plot of the *advertising_spend_k* variable from the *advertising_sales_channel* data frame

Quick Check

Where would the violin be widest for a symmetric bell-shaped distribution?

Suggested answer

Near the center, where values occur most frequently.

Quick Check

If you have a symmetric bell-shaped violin plot what does it say about values on the high and low ends of the range?

Suggested answer

Extreme high and low values occur with similar rarity.

Quick Check

Would you generally expect bell-shaped distributions in measurement based data?

Suggested answer

Yes. Take people's heights for instance. They will tend to be bell shaped with the maximum density around the average height. Because many small, independent influences combine to produce values near an average. This is actually a very important theorem in Statistics and is called the *Central Limit Theorem*. We will not be studying it in this course. This is not to say that most real-life data tend to be symmetric bell-shaped. For instance, incomes tend to be skewed towards higher values (or to the right). A few common distributions capture a surprisingly large number of real-world phenomena.

Quick Check

Describe how mulch-modal distributions look on a violin plot.

Suggested answer

As multiple bulges or wide regions separated by narrower sections.

Quick Check

What might cause a business data set to have more than one mode?

Suggested answer

The data may combine different subgroups, such as entry-level and senior employees. Or data for different products and so on. We will encounter these later in the course.

Quick Check

What would the violin plot look like for a distribution wish is skewed towards the higher values?

Suggested answer

A violin with its broad portion down at the bottom and a long stem towards the top.

Quick Check

What do you think skew does to the average or mean?

Suggested answer

The average gets pulled towards the direction of the skew. A small number of extreme values can pull the mean away from where most observations lie.

3.4 Comparing distributions with violin plots

Thus far, we have examined distributions of individual variables in isolation. In statistics we often want to compare distributions. For example, in the *acct_type_balance* data frame, how does the distribution of *balance* for *Checking* accounts compare with that for *Savings* accounts?

```
acct_type_balance |>
  point_plot(balance ~ bank_account_type, annot = "violin")
```

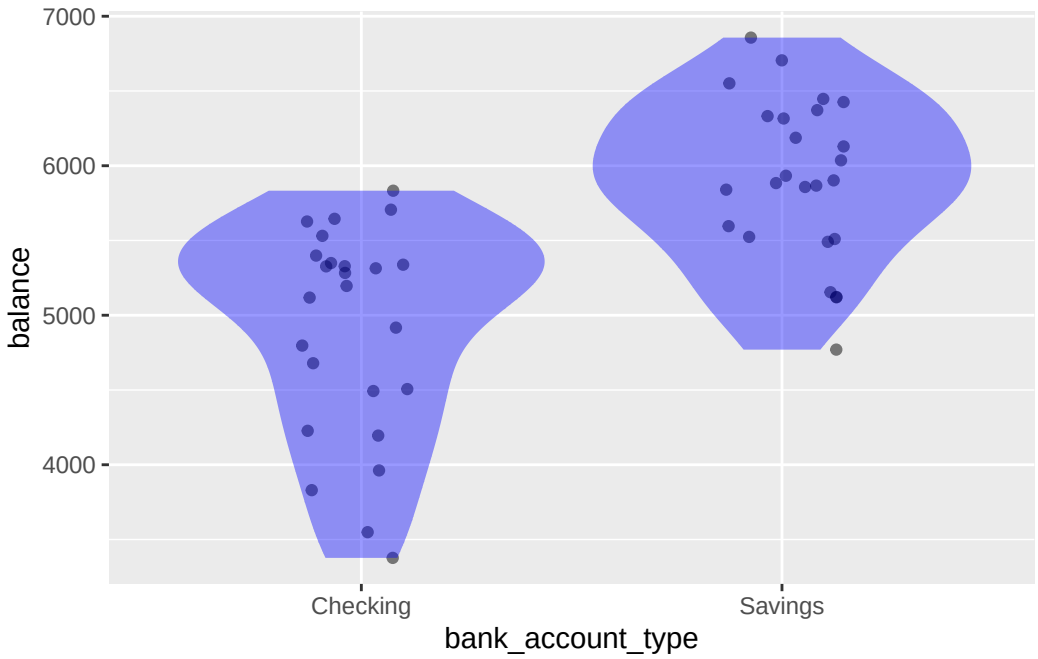

Figure 3.16: Violin plots of the *balance* variable for different accoiunt types – from the *price_demand* data frame

Since we want to compare the two violins corresponding to the two different account types, we now have *bank_account_type* on the x-axis and our tilde expression reflects this.

Figure 3.16 shows several things: - account balances in *Savings* accounts are generally larger - account balances of *Savings* accounts are distributed in an almost symmetrical bell shape with the maximum density around $6,000. - account balances of *Checking* accounts are generally smaller than those of *Savings* accounts - account balances of *Checking* accounts are bell shaped, but skewed towards lower values with the maximum density around $5,400 or so.

- As another example of comparing distributions, let us use the *advertising_sales_channel* data frame to compare the distribution of *weekly_sales_k* for different *channel*s .

```
advertising_sales_channel |>
  point_plot(weekly_sales_k ~ channel, annot = "violin")
```

Figure 3.17: Violin plots of the *weekly_sales_k* variable for different sales channels – from the *advertising_sales_channel* data frame

Figure 3.17 tells us the following: - *weekly_sales_k* values are generally lower for the *Retail Promo* channel - for the *Retail Promo* channel *weekly_sales_k* is distributed systematically with three short peaks - *weekly_sales_k* values are almost uniformly distributed for the *Search Ads* channel and have a much larger range than the *Retail Sales* channel

4 Variance and standard deviation

This chapter discusses the key statistical concepts of *variance* and *standard deviation*.

Learning outcomes

After completing this chapter you will be able to:

- Given two sets of numbers with 10 or less numbers, identify which has higher variance and explain why
- Generate a set of numbers with zero variance
- Given a set of no more than five numbers (at most two digits) compute their variance by hand
- Explain the units of measurement for variance
- Explain the relationship between variance and standard deviation
- Explain the units of measurement for standard deviation
- Use R to compute the variance and standard deviation of a numerical variable of a data frame

4.1 Variable

We use the term “variable” to describe a column of a data frame. Why? Let us make things more concrete. A data frame with data on employees of a company might have a column named “salary” to store the salaries of employees. Salaries of employees vary. That is, we know that not everyone in the company has the same salary (generally speaking). So, it makes eminent sense to call a column a “variable.”

Similarly, the same data frame might have another column named “Position” that stores the position of each employee. Some example values might be “Manager”, “Sales associate”, and “IT Specialist.” Here too we see “variability” because not all the values in the column are the same. So the term “variable” applies to columns whether they contain numeric or categorical values. However, for the rest of this document, we concern ourselves only with numerical variables.

4.2 Why *means* (averages) are not enough

In business, averages are often the first numbers we look at. Average revenue, average delivery time, average customer wait time, average return on investment—these figures are easy to compute and easy to communicate.

But averages alone can be dangerously misleading.

To see why, consider the following examples.

4.2.1 Example 1: Two stores, same average sales

Suppose two retail stores each reports **average daily sales of $10,000**.

- **Store A** has daily sales that are usually between $9,500 and $10,500.
- **Store B** has daily sales that swing wildly, ranging from $3,000 on slow days to $25,000 on busy days.

From the average alone, the two stores appear identical. From a management perspective, they are not.

Store A is predictable. Staffing, inventory, and cash flow planning are relatively straightforward. Store B is risky. Some days it is overstaffed and overstocked; on other days it cannot meet demand.

The average hides this difference. What distinguishes the two stores is **variability**, not the mean.

4.2.2 Example 2: Average delivery time

A logistics manager is choosing between two suppliers. Both advertise an **average delivery time of 5 days**.

- Supplier X delivers in 4–6 days almost every time.
- Supplier Y sometimes delivers in 2 days and sometimes in 12 days.

If you only look at averages, the suppliers appear almost equivalent.

If your business depends on reliable delivery schedules, they are not.

In this context, variability directly affects:

- Production planning
- Inventory holding costs
- Customer satisfaction

Once again, the average tells only part of the story.

4.2.3 Example 3: Investment Returns

Two investment portfolios have the same **average annual return of 8%**.

- Portfolio A produces returns close to 8% year after year.
- Portfolio B alternates between very high gains and significant losses.

Many investors would consider Portfolio B riskier, even though the averages are identical. That perception of risk comes from **how spread out the returns are**, not from the mean itself.

4.2.4 The Key Idea

Averages describe the *center* of the data, but they say nothing about how the data are **spread out**.

In business settings, variability affects:

- Risk
- Reliability
- Predictability
- Planning and decision-making

To fully understand a data set, we need measures that describe **how much values differ from the average**. That is the role of **variance** and **standard deviation**.

Quick Check

Two companies have the same average monthly revenue. Company A's revenue is very stable from month to month, while Company B's revenue fluctuates widely. Why might a manager prefer Company A, even though the averages are the same?

Suggested answer

A manager might prefer Company A because its stable revenue makes planning easier and reduces risk. Predictable revenue helps with budgeting, staffing, inventory management, and cash flow. Company B's wide fluctuations introduce uncertainty, even though its average revenue is the same.

Quick Check

Two suppliers both report an average delivery time of five days. One supplier delivers in a narrow range of times, while the other sometimes delivers very early and sometimes very late. What important information does the average delivery time fail to capture?

Suggested answer

The average delivery time does not capture how consistent or reliable deliveries are. It ignores variability. A supplier with unpredictable delivery times can disrupt production schedules and increase costs, even if the average delivery time is acceptable.

Quick Check

Explain why averages alone are insufficient for assessing risk in investment decisions.

Suggested answer

Averages do not show how much returns vary from year to year. Two investments can have the same average return but very different levels of risk. Higher variability in returns generally means greater uncertainty and risk for the investor.

Quick Check

In your own words, explain what variability means in a business context.

Suggested answer

Variability refers to how much outcomes differ from their average value. In business, it reflects unpredictability in areas such as sales, delivery times, costs, or returns, which can affect planning and decision-making.

Quick Check

Give one example of a business decision where variability might matter more than the average.

Suggested answer

Examples include delivery times in a supply chain, customer wait times in a service operation, investment returns, manufacturing defect rates, or daily demand for a product. In these cases, high variability can cause operational problems even if the average appears acceptable.

In the sections that follow, we will develop these measures and learn how to interpret them in practical business contexts.

! Grasp this

Variance and standard deviation quantify how much data values typically deviate from the mean.

4.3 Amount of Variation

Measuring the amount of variation plays an important role in statistics. If a column has no variation, then we cannot make much use of the information in the column. When a column does have variation, we often want to know the extent of variation in the values of the column. We want to ascribe a specific number – that is, *measure* the variation in a column. We call this measure the *variance* of the values in a column. When the numbers in a column are all the same, then we say that the column has no variance – that is, the variance is zero.

Consider the following three sets of 8 numbers each: - Set 1: (1, 2, 2, 1, 2, 1, 2, 1) - Set 2: (2, 6, 8, 2, 9, 2, 10, 9) - Set 3: (1, 3, 2, 3, 2, 2, 3, 4)

Which of the three sets has the highest amount of variation? That is, which set has numbers that differ by the most (among the three sets)?

Can you order the sets in increasing order of their overall variability?

Overall, we can see that the numbers in Set 1 are much closer to each other than those in the other two sets. The numbers in Set 2 are the furthest apart from each other and those in Set 3 are in-between. Therefore the order is:

Variance of Set 1 < Variance of Set 3 < Variance of Set 2

4.4 Measuring Variation through *variance*

From the foregoing, we see that the variance of a set of numbers is a measure of how much the numbers differ from each other. A set of numbers in which the elements are generally very close together has a low variance and a set that contains numbers that differ from each other by a lot has high variance. Over and above just describing what low and high variance look like, we would like to assign a specific number to the variance of a set of numbers.

In reality we will compute the variance using the R function *var*. We describe two procedures below just to give you a good feel for what the variance represents.

4.4.1 Computing Variance: Method 1 – Using deviations from the mean

We had earlier mentioned that *variance* is the extent to which the numbers in a collection deviate from the average. We use the following steps.

1. Find the mean
2. Find each number's difference from the mean
3. Square the differences

4. Average the squared differences (but use (n-1) in the denominator. We show a worked example below.

Again, we use the same set of numbers (2, 4, 5, 9). Let us suppose that these numbers represent the prices of some items in a shop (in the US) and therefore they represent USD amounts.

4.4.1.1 Steps to compute variance using deviations from the mean

1. **Find the average of the numbers.**

   The sum of the numbers is:

   2 + 4 + 5 + 9 = 20

   The average is: 20 / 4 = 5

2. **Find the squared deviations from the mean for each number.**

Number	Difference from mean	Difference squared
2	3	9
4	1	1
5	0	0
9	4	16

3. **Divide the total of the squared deviations from the prior step by (n 1), where n is the number of elements.**

   The sum of the squared deviations is: 9 + 1 + 0 + 16 = 26

   Dividing by (n 1): 26 / 3 = 8.66667

> **!** Zero variance?
>
> Callout content goes here.

What are the characteristics of a set of numbers with zero variance? Think a little before reading on.

If the variance has to be zero then. the average of the differences also has to be zero, which means each of the differences has to be zero. That in turn means that all the numbers have to be the same. This makes a lot of sense, because if there is zero or no *variance*, then the numbers do not vary!

4.4.2 Computing Variance: Preferred Method – Using R

We have a data frame named *variance_example* that has a variable *num* with the values (2, 4, 5, 9). We can use the following R code to compute its variance.

```
variance_example |>
  summarize(num_var = var(num))
```

```
# A tibble: 1 x 1
  num_var
    <dbl>
1    8.67
```

Here is another example using the *mpg* data frame to compute the variance of one of its variables.

```
mpg |>
  summarize(hwy_var = var(hwy))
```

```
# A tibble: 1 x 1
  hwy_var
    <dbl>
1    35.5
```

In the previous section we computed the variance of a set of numbers that represented USD values. What are the units for variance?

4.4.3 Unit of measurement for variance

Whenever we measure anything, we measure it in terms of some *unit of measurement*. For example, we might use meter as the *unit of measurement* for people's heights, and USD as the *unit of measurement* for US company profits. In fact, in any data frame, whenever we see a numerical variable, we need to be aware of its *unit of measurement*.

In the two methods where we computed the variance, we found some differences in each step. These differences were differences between USD amounts and hence the differences have units of USD. We then squared the differences. The unit of measurement for the squares would be USD-squared. We then averaged these in one approach and divided by a number in the other approach to get the variance. Therefore, variance also has USD-squared as its *unit of measurement*.

In general, when we compute the variance of a variable measured in some units (say, u, the variance has u-squared as its unit of measurement. That is, if we have a variable expressed in meters, then the variance of the variable will be in meters-squared.

We generally have a clear mental notion of units like USD and meters, but not units like USD-squared or meters-squared.

4.5 Another measure of spread: *Standard Deviation*

We have seen that *variance* has squared units and we cannot easily relate to these. Statisticians have therefore given us another measure of variability that has the same units as the original variable. This is the *standard deviation* and we compute it as the square-root of the variance.

If a variable is measured in inches, its variance has inches-squared as its unit. However, when we compute its square root, we get the *standard deviation* with inches as units.

We can now mentally relate the original variable values to the *standard deviation* and therefore statisticians use this measure very widely. This is not to say that variance is not widely used as well. It is, and we will use it extensively in this course as well.

> **!** What information does the standard deviation give us
>
> The standard deviation tells us the typical distance between a data value and the mean.

4.5.1 Computing *standard deviation* using R

We can use the *sd* function inside the *summarize* function to compute the *standard deviation*

Let us compute the *standard deviation* of the city mileage (variable *cty*) in the *mpg* data frame.

```
mpg |>
  summarize(cty_sd = sd(cty))
```

```
# A tibble: 1 x 1
  cty_sd
   <dbl>
1   4.26
```

Just to confirm that the computed *standard deviation* is in fact the square root of the *variance* let us compute both.

```
mpg |>
  summarize(cty_var = var(cty), cty_sd = sd(cty))
```

```
# A tibble: 1 x 2
  cty_var cty_sd
    <dbl>  <dbl>
1    18.1   4.26
```

4.6 Optional enrichment topic

We saw two ways to compute variance. There is another method, which, of course, gives the same result, but is quite intriguing, but intuitive.

4.6.1 Computing Variance: Intriguing Method 1 – Using pairwise differences

We first look at an intuitive (but *impractical*) method for computing the variance of a set of numbers. Variance represents how different each member of a set of numbers is from the rest. So the difference between each number and the rest of them plays a central role. Our first approach to computing the variance involves looking at every possible pair of numbers in our set and seeing how different the two numbers of the pair are from each other. We then combine all of these pairwise differences into a single number to represent the overall variability in the set. #### Steps to compute variance using pairwise differences

Let us assume that we want to compute the variance of the set of 4 numbers (2, 4, 5, 9)

1. **Identify each possible pair of numbers from the set.**

   (2, 4), (2, 5), (2, 9)
   (4, 5), (4, 9)
   (5, 9)

   There are 6 such pairs.

2. **Compute the difference for each pair and square it.**

Pair	Difference	Difference squared
(2, 4)	2	4
(2, 5)	3	9
(2, 9)	7	49
(4, 5)	1	1
(4, 9)	5	25
(5, 9)	4	16

3. **Find the average of the squared differences.**

   The sum of the squared differences is:

   4 + 9 + 49 + 1 + 25 + 16 = 104

   The average is:

   104 / 6 = 17.33333

4. **Divide the average from the prior step by 2.**

   17.33333 / 2 = 8.66667

   So, the variance of the set (2, 4, 5, 9) is **8.66667**.

Part III

Introduction to models

Module 3: Introduction to models

This module gently introduces the idea of a model, and then discusses the simplest possible model – the mean. It also introduces the concepts of outcome and explanatory variables. It extends the idea of the mean as the model with no explanatory variable to cover the case of a model with one categorical explanatory variable and shows that in this case the model is the category mean.

Learning Goals

**After completing this module, you will be able to:

- Explain the term model, generally and in our context
- Intuitively explain why the mean is the best model in the absence of any other information
-

Structure of This Module

- Mean as a model
- Category mean as model

5 The Simplest Model

This chapter helps you to dip your toes into the vast and important topic of models.

Learning outcomes

After completing this chapter you will be able to:

- Explain the term model from a general perspective, and from the perspective of statistics
- Explain why, given the values of only one variable and in the absence of any other information, the mean is the best model
- Explain in what sense the mean is the *best* model
- Given a model equation, identify the *outcome* variable and the *explanatory* variable(s)
- Describe the terms *outcome* variable and **explanatory* variable
- Match the terminology between the two pairs (*outcome* variable, *explanatory* variable) and (independent variable, *dependent* variable)
- Given a model with a single numerical *explanatory* variable and a value for the *explanatory* variable, compute the model value.
- Explain why we place a *hat* on the *outcome* variable in a model expression
- Explain why we call a model's output an *estimate*
- List and describe the two uses of models covered in this chapter

5.1 Companies *commit* when they make decisions

Good decision-making lies at the heart of effective business management.

In practice, companies must commit to decisions before they know how it will play out in the real-world. Many important business decisions cannot be revised freely once they are made. Examples include setting prices, determining how much inventory to stock, choosing staffing levels, or deciding where to locate a factory.

Once a decision is made, it influences many individual events that unfold over time.

Consider pricing. A company sets a single price for a product, and then thousands of customers independently decide whether or not to buy at that price. One decision by the firm affects many separate outcomes.

This has an important implication: *we cannot judge the quality of a decision based on a single outcome*.

Suppose a company sets a price of $10 and the first customer who walks in buys the product. Was $10 the right price? We cannot really say. That single outcome may simply be good luck. If we observe the decisions of 1,000 customers, however, a clearer picture begins to emerge. Over many independent events, good decisions tend to perform well *on average*.

In business, then, decisions must be made in advance, under uncertainty, and their quality must be judged by how well they balance out across many realizations and not by whether they happen to succeed in one particular instance.

This perspective will guide how we think about models in this chapter. A good model is not one that gets lucky once. It is one that performs reliably across many possible outcomes.

> **!** Definition of model for our present purposes
>
> A model is a simple rule or equation that uses data to produce predictions or explanations, while deliberately ignoring some details of reality.

5.2 Staffing kiosks with incomplete data: Deciding under uncertainty

Amanda, the owner of *Random Treats* walks into a conference room with a small group of business school interns.

“I run two small sales kiosks,” she says. “One at the *Beach*. One at the *Mall*. I need to set staffing levels for the next month.”

Amanda holds up a messy spreadsheet printout.

“Customer counts from the last 20 days,” she says. “But the person who collected the data forgot to record *which kiosk* each number came from. My problem is that I have to set the staffing level for the next month at my kiosks and I have to base that on the expected number of customers who will show up at the kiosks.”

Someone laughs nervously. Amanda does not.

“I don’t have time to redo data collection,” she says. “So we work with what we have. But first, I want to see how you think.

Based on the data we have, I want to settle on a single number for how many customers will show up at the kiosks each day. I will use that number to set the staffing level for the kiosks – same for both kiosks.”

She grabs a marker.

“Here’s a tiny practice version of the problem, with small numbers that you can wrap your head around. Instead of the messy numbers on the spreadhseet, assume that these small numbers were the number of customers who showed up over the past several days at a kiosk (beach or mall). Now tell me what single number I should use as the representative demand at a kiosk?”

She writes the following set of 10 numbers on the board:

4, 2, 6, 1, 8, 1, 3, 9, 2, 4

You should think of this set as representing possible outcomes—number of customers who show up at a kiosk. I am keeping the numbers small so that we can first arrive at a method. We can then apply the chosen method to the messy real-world numbers. You do not know which value will actually occur on any given day, but you have reason to believe that the overall pattern will be similar to these numbers.”

Angela asks “But why should you keep the same number each day? Can we not adjust staffing day by day?”

Amanda says “Good thinking, Angela. I would love to be efficient, but I cannot do that. Firstly, I want to give some stability to the staff so that they can rest asusred that they have guaranteed employment for the whole month. Secondly, I run many businesses and I can only commit a few hours to this business. So today is the day I make the staffing decision for the next month for this business. I can revisit this decision only next month.”

“Got it” says Angela.

Amanda . . . “OK, guys, put on your thinking hats.”.

“Here are the rules of the game:

1. You as a team commit to a single number for expected sales on any given day at either kiosk. This is your guess, and you must choose it before any outcomes – actual sales – are revealed over the month. You can assume that the pattern will be similar to the data I have given you.
2. I then pretend to be the real-world, and determine the first day’s demand by randomly selecting one number (customer demand) from our set. This number I pick might be the same as, lower than, or higher than your committed number. The closer it is to the number I pick randomly, the better. I will compute the difference between what you committed to and the real-world – what I picked randomly.
3. Now comes the twist. I square the difference to obtain the *penalty* for that round.
4. We repeat steps 2 and 3, twenty times just to get a reasonable number of trials, and add up all the *penalties*.

The result of step 4 is your *total penalty*.

Obviously, your goal is to keep this total penalty as small as possible to keep the number you committed to close to the actual number of customers (the numbers I picked randomly from our representative data) .

So, remember, you commit once and the real world plays out day after day. At the end of 20 days, we want to see the overall penalty. It is no good doing really well on a few days and being way off on others. Consistency is key."

Steve seemed perplexed. "But Amanda, why not just add the differences? Why square them and then add?"

"I like that Steve. You are thinking well about this already" said Amanda. In our business we can afford small deviations from reality, but the farther our decisions are from reality, we pay a far heftier price. So I want to really avoid large deviations from reality and squaring the differences and choosing the solution that produces the lowest penalty take care of this aspect. For instance, if we do not square the differences, then an error of 10 is only twice as bad as an error of 5. Whereas qith squaring, an error of 10 is 4 times as bad as an error of 5. I really want to minimize big errors."

"Oh, I get it." Steve said. " We did something similar when we calculated variances in our statistics class."

Amanda: "Nice connection Steve!"

"OK team. Why don't you discuss for 10 minutes and give me a number while I grab a coffee?"

10 minutes later Dave announces "Amanda, we have discussed. Our number is 6".

"Thanks, Dave. I will now randomly generate the number of customers for 20 days based on our data and we can jointly compute the total penalty."

Table 5.1 shows the computation and the results. The total penalty was 219.

Table 5.1: Decision, random choices, and resulting penalties (Decision = 6)

Decision =		6
random_choice	diff	penalty
9	-3	9
3	3	9
9	-3	9
4	2	4
2	4	16
6	0	0
2	4	16
8	-2	4
4	2	4

Decision =		6
random_choice	diff	penalty
1	5	25
2	4	16
2	4	16
2	4	16
2	4	16
3	3	9
9	-3	9
2	4	16
6	0	0
9	-3	9
2	4	16
Total penalty =		219

Quick Check

Explain how the penalty of 9 was computed for the first row of Table 5.1.

Suggested answer

The decision is 6. The random choice selected 9 customers. The difference is -3 (or 3 depending on what you placed first in computing the difference. Since we are squaring, this does not matter.) The squares of -3 and 3 are both 9.

Amanda: “The total penalty was 219. Can you reduce the penalty?”

Paul: “Let’s try 7.”

Table 5.2 shows the results. The total penalty this time is slightly higher at 230.

Table 5.2: Decision, random choices, and resulting penalties (Decision = 7)

Decision =		7
random_choice	diff	penalty
4	3	9
4	3	9
3	4	16
4	3	9
4	3	9
8	-1	1
8	-1	1
9	-2	4
4	3	9

Decision =		7
random_choice	diff	penalty
3	4	16
2	5	25
3	4	16
2	5	25
3	4	16
8	-1	1
4	3	9
8	-1	1
1	6	36
4	3	9
4	3	9
Total penalty =		230

Amanda: “Well that raised the penalty slightly. Any ideas how to proceed further? If we keep trying different numbers, we could be here all night and still not know if we have the best solution!

What number will produce the lowest penalty? Honestly, I do not know and I need your bright minds to help me with this.

I am going to leave you all to work through this as a team for an hour while I attend another meeting. When I come back, I hope you will be able to give me a good solution and convince me that it is indeed good!.”

Igor had been silent all along and finally spoke up tentatively.

“I think we should try many numbers from 1 to 9 – the range of this data – in steps of 0.25 or something small. For each number, we should compute the total penalty by doing something like what Amanda had us do earlier. But trying just 20 times for each number leaves us at the mercy the random selection process giving freak results. Instead we should aim for a large number of trials – like 1000. Then the chance factor will be considerably reduced.

We can then plot a chart showing the total penalty for each decision and directly see the decision for which total penalty is the lowest.”

Dave wanted to clarify. “Are you saying, Igor, that you will first pick 1 as the choice and then generate 1000 random choices from the data set. For each random choice you will compute the difference from the random choice and 1 and square it and add them all up to get the total penalty. That would be the total penalty for the choice 1.

You wll then repeat the procedure for 1.25 and get its total penalty. And then 1.5, and so on until 9 (which is the largest number in our data). You will then have a table with two columns: *choice* and *total_penalty*. You will then plot this.”

“Exactly” said Igor and sketched a rough plot (Figure 5.1) on the back of an envelope. “This is what it will look like. Of course, I have not done the computations and so my sketch is just an idea now. I suspect that the total penalty will be lowest at some decision and increase on either side of it. Once we have the plot, we can then find the decision at which the total penalty is the smallest.”

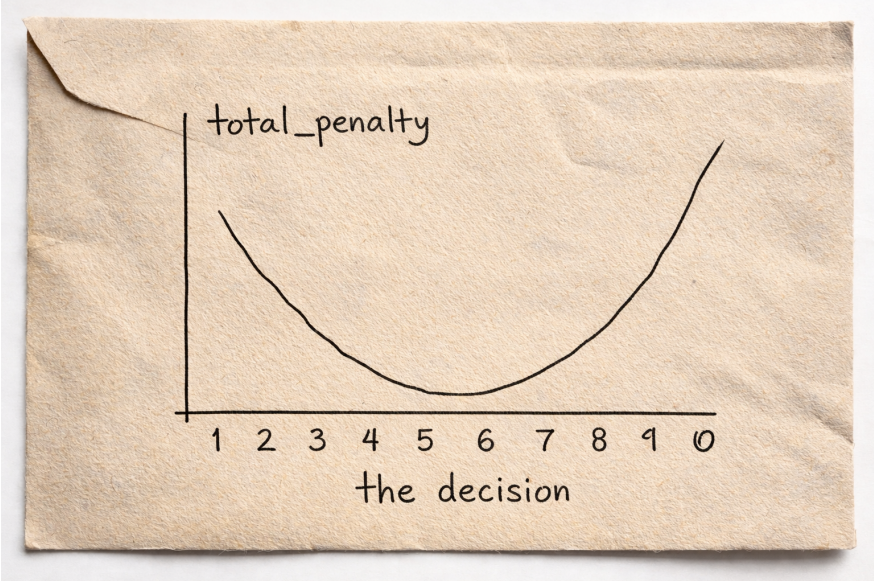

Figure 5.1: Igor’s back of the envelope sketch of the plot that he thought would help them identify the best decision

“If we can generate this plot, we might be able to sell the idea to Amanda” Steve said.

Quick Check

How do you think Igor’s sketch in Figure 5.1 might be useful in finding the best decision?

Suggested answer

If the plot does in fact come out to have this shape, then the decision where the total penalty is lowest would be good. The plot once actually generated can help us to find the best decision.

“But Igor, this is easier said than done. How can we complete all these computations in an hour? In fact we might not finish in a week!” bemoaned Angela. Igor agreed. Perhaps he was being too much in the clouds.

Suzie joined the fray. “Angela, If what I am thinking of is correct, this might not be as hard as it seems. I learned R in a class. I think I can do this in R quickly and get a nice plot that Igor is talking about. We can have something ready for Amanda by the time she comes back from her meeting.”

Everyone exulted “You can do that? Wow!” and Suzie and Igor got to work.

They defined the problem clearly to an AI chatbot and asked it to generate a program in the R language that could generate the plot. They described precisely what they wanted the program to do. This part took some time as they wanted to be very sure that the chatbot had the right information.

Once they were done, the chatbot then gave them the code in less than a minute. She looked over it carefully, fixed a few things and then ran it.

Soon Suzie had a nice plot of the total errors for every possible value of the decision from 1 through 9 in intervals of 0.2. In fact she had made 10,000 trials for each value instead of just 1,000 as Igor had suggested. It looks like a guess of 4 would produce the smallest total penalty. The team had its answer for Amanda! Figure 5.2 shows the plot. The plot shows that a decision of 4 generates the lowest total penalty.

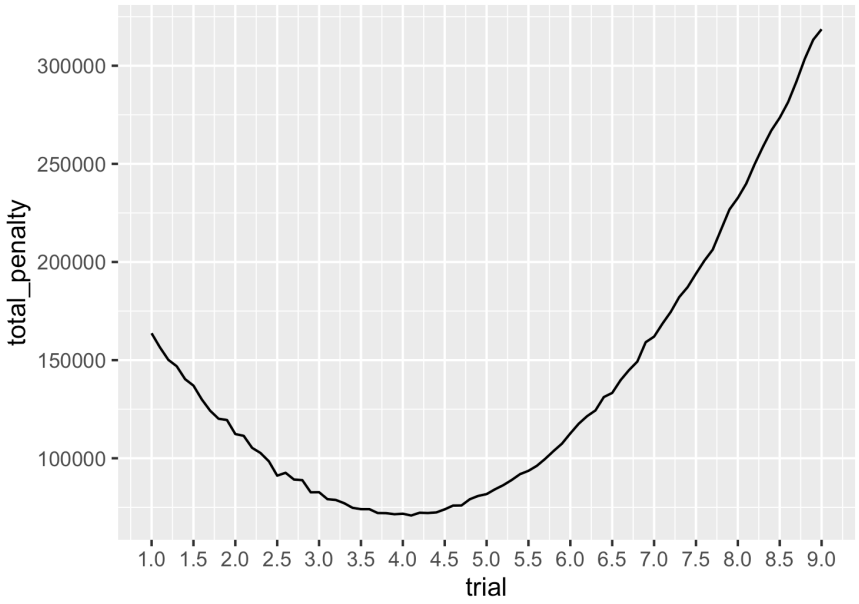

Figure 5.2: Results of Suzie's program showing the various possible decisions and their total penalties: the lowest total penalty occurs at a decision of 4

High-fives all round.

Quick Check

How many values of decision did Suzie's program try out?

Suggested answer

She tried the values 1, 1.2, 1.4, 1.6, 1.8, or 5 values between 1 and 2. Similarly 5 between 2 and 3 and so on That would be 8 times 5 or 40 up to 8.75. and then there is 9. So she tried a total of 41 different decision values.

Quick Check

How many individual differences did her program compute?

Suggested answer

For each decision her program tried 10,000 trials and hence computed 10,000 differences for a total of 10,000*41, or 410,000 differences.

“Wait a minute” said Angela. With Amanda’s trials, the total penalty was close to 200. Now it looks like our lowest is close to 70,000. What is going on? Have me done something wrong here? I have heard that AI chatbots can hallucinate. Have wee been hit with a major hallucination?”

Doom spread in the room. Amanda would be in the room shortly. Have they let her down badly?

Silent Igor came to their rescue again. “The total penalty in Amanda’s case was computed over 20 trials. Suzie’s total penalties are based on 10,000 trials. Naturally they will be higher. This is an apples to oranges comparison.

To make these comparable, we should be comparing the average penalty **per trial** to remove the direct effect of the number of trials on the total penalty. So we should divide Amanda’s total penalty by 20 and Suzie’s by 10,000 to get the average penalty per trial for each of them. Then we can compare them.”

David immediately did some mental math and came back with “Well, for the numbers we tried with Amanda, the average penalty per trial was approximately 220/20, or 11. For Suzie’s numbers, the plot says that the total penalty for the best decision of 4 is around 70,000. We divide that by 10,000 trials and get a penalty per trial of 7. So we did improve significantly on what we initially came up with.”

The group heaved collective sign of relief.

Igor said “Since Suzie did 10,000 trials for each number, we can be almost certain that 4 is in fact *the best solution*.

The team had finished with 10 minutes to spare. They spent the remaining time preparing a neat presentation for Amanda describing exactly what they had done. They also explained how they had used AI to help in this process.

When she returned, Amanda took one look at the chart and asked a few questions about how they had generated it. She was convinced that these young interns had found the best solution to her kisok staffing problem.

Amanda said “Thank you all very much. You have done a great job! You accomplished a lot in a short amount of time. Do you know why that happened?

You worked as a team. Your ideas built upon each other and you were able to achieve a lot.

How do I know even though I was not in the room? Well, I have seen this time and again, and you will too in your working lives. That's the strength of teams – some say Wisdom of Teams.

We will meet next month to review how things went and to refine our decisions."

5.3 Average or *mean* as the decision in the long run

David had a thought. "Let us try the decision value of 4 using Amanda's approach and see what total penalty we get for 20 trials."

They did, and Table 5.3 shows that indeed the total penalty for this is lower than the first two and the average penalty per trial is 166/20. This is higher than the average of 7 with 10,000 trials, but we can chalk it up to using only 20 trials.

Table 5.3: Decision, random choices, and resulting penalties (Decision = 4)

Decision =		4
random_choice	diff	penalty
8	-4	16
2	2	4
1	3	9
2	2	4
4	0	0
4	0	0
8	-4	16
2	2	4
8	-4	16
8	-4	16
4	0	0
4	0	0
9	-5	25
1	3	9
3	1	1
9	-5	25
2	2	4
2	2	4
1	3	9
2	2	4
Total penalty =		166

So what is special about 4? Given another set of numbers, do we have to repeat this charting process that Suzie did for each set of numbers?

Well, 4 happens to be the average of the numbers. It turns out that when we have a large number of trials, we will get the best results, that is the lowest total penalty, when we always guess the average of the numbers. In this case the average is 4.

So, we see that in the absence of any other information to help us predict, the best prediction is the *mean*. Figure 5.3 shows this pictorially.

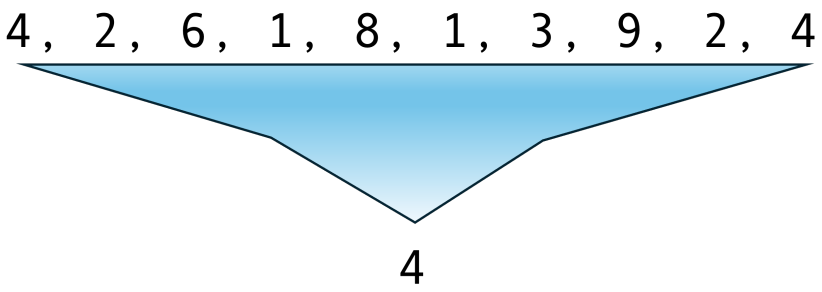

Figure 5.3: Many numbers, but just a single prediction!

What we have said above **does not mean that we are guaranteed to get the smallest total error if we commit to the average**. Clearly, if someone were lucky enough that the random picks all ended up exactly equal to or very close to their decision just by chance, then their total error will be zero! But this is *extremely unlikely*.

However, those occurrences are extraordinarily unlikely with a large number of trials, as happens in business where a decision is tested by thousands of events. *Statistically* the best course would be to pick the average.

Quick Check

In what sense is the mean the “best” model?

Suggested answer

“Best” means minimizing overall prediction error, not being perfect.

Quick Check

Can a model be “best” and still be inaccurate?

Suggested answer

Yes. A model can be the best available option given limited information and still perform poorly in absolute terms.

Quick Check

Does it make intuitive sense that the average turns out to be the best model? Explain why or why not.

Suggested answer

It makes sense to guess a number sort of in the middle so that we are generally close to all numbers that could turn up on random choices. See s@ec-simple-model-enrichment-1 if you want to thin k about this more.

! Interpret the word “best model” with caution

When we say the mean is the best model, we do not mean it is always accurate, fair, or appropriate. We mean it is best according to a specific criterion, under specific information constraints.

Now you know why I titled the chapter *The simplest Model*!

5.4 Real world connection: Planning for Customer Demand

How does this small “practice game” relate to the real world? Let us return to the manager’s kiosk problem.

A company operates two small sales kiosks in the same city—one at the *Beach* and one at the *Mall*. The company must decide how many workers to schedule each day.

5.4.1 What the company must do

- The staffing decision must be made **in advance**
- Once the schedule is set, it cannot be easily changed without disrupting employees
- In this first round, the manager insists on **one number** to use everywhere (the same staffing plan for both kiosks)

5.4.2 What the company knows

- The company has historical data on daily customer counts—but in an unhelpful format
- The counts are **not labeled by kiosk** (Beach vs Mall)
- At this point, the company does not have any data to explain *why* customer traffic changes. It simply observes that demand varies

Here are the 20 historical customer counts (mixed together, with no kiosk identification):

89, 95, 131, 101, 103, 134, 109, 75, 86, 91, 177, 151, 152, 143, 123, 194, 155,

5.4.3 Why the decision matters

- If too few workers are scheduled:
  - customers wait longer
  - service quality suffers
  - some customers leave without buying
- If too many workers are scheduled:
  - workers are idle
  - labor cost rises
  - managers waste attention adjusting schedules

5.4.4 How the company measures mistakes

- Being slightly wrong is not very costly
- Being very wrong is much more costly
- The manager therefore measures the cost of a decision as:
  - the **square of the difference** between planned customers and actual customers

5.4.5 How the decision is evaluated

- The decision is **not judged by a single day**
- A decision that works well once may simply be lucky
- Instead, the manager looks at performance **across many days**
- Daily customer counts vary, but the staffing decision stays the same

5.4.6 The conclusion

- The manager must choose a single number to plan for
- Under a squared-penalty rule, the number that minimizes total long-run cost is the **average** of past customer counts (here, about **123.8**)
- Choosing a smaller number leads to frequent large shortages
- Choosing a larger number leads to frequent over-staffing
- The average balances these errors over time

5.4.7 Why this matters

- This is the first "model" you learn in this book: a model that predicts the same value every time
- It is simple, but it is not arbitrary—it is the **best** choice given the information available
- Later chapters improve on this model by adding explanatory information (first categories, then numerical predictors)

5.5 Optional enrichment topic: How does skew affect the situation?

In our practice game, we used the numbers:

4, 2, 6, 1, 8, 1, 3, 9, 2, 4

in our game. That set has numbers across the entire range. Is it the case that the average performed well because of this? Could it be that the average will not perform well if most of the numbers fall within in a small range and a few extreme cases tend to push the average up?

For, example, let us take the following set of numbers:

2, 1, 2, 1, 3, 3, 2, 1, 10, 9

The mean is 3.4. However, this set has eight of its numbers less than or equal to 3. Perhaps a decision below the average will work better?

Let us play the game 10,000 times and see which decision performs best.

Figure 5.4 shows the results. We see that the mean is still the best *decision*.

Why does this happen? Choosing the average definitely leads to small penalty increases most of the time. However, if we went below the average, the relatively rare cases in which the random choice is high, it incurs a very large penalty and that offsets any benefits of going below the average, because we square the difference and that amplifies the error.

What if we did not square the error and instead treated the absolute difference as the penalty. In this case, the *median* is the best choice. Pretty cool result if you as me! We will not go further into that topic, but it has its applications. In most practical applications, we use the squared-difference as the penalty.

Pause & Think

1. If two data sets have the same mean, could the mean be a better model for one than the other?
2. What additional information might help you judge if it will work better for one than for the other?

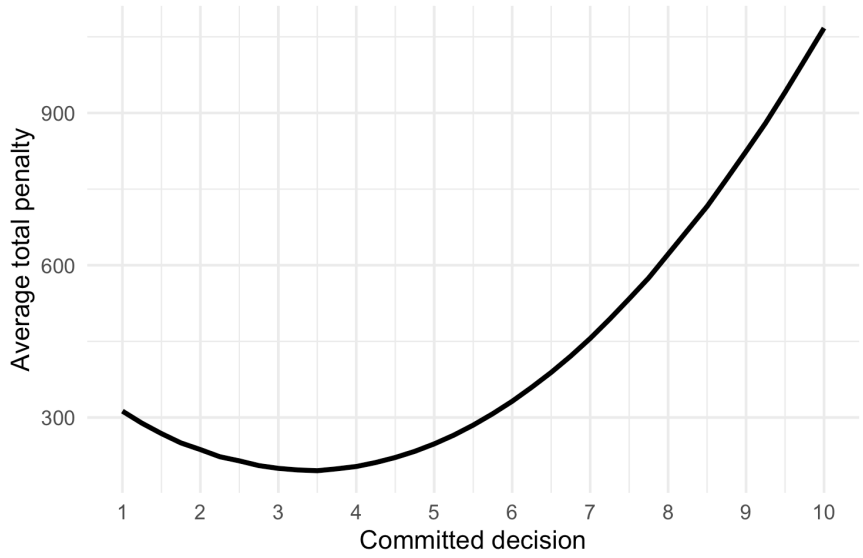

Figure 5.4: Average of *total_penalty* vs various values for *decision* based on 10,000 plays of the game – shows that the lowest total penalty overall occurs when *decision* equals the mean of the numbers

Suggested answers

1. Yes. The mean works better when observations are tightly clustered.
2. Variability, spread, or deviation from the mean would help evaluate model quality.

5.6 Optional enrichment topic: Removing outliers in skewed distributions

Continuing from the prior discussion, when faced with a skewed distribution we have a peculiar situation where our intuition tells us to make a decision where most points lie, but cold computation tells us otherwise.

One may feel that it makes no sense to choose the mean and accumulate a slightly higher penalty in every event just to balance out the huge penalties that come rarely. But that is what makes sense when we use the quadratic penalty function.

If the penalty is quadratic, but we still want to not decide on the average, then we can eliminate the extreme points and reduce the skew. Then we can use the average as before. From a practical viewpoint what that would mean is that we take steps to avoid extreme events by some means.

Interestingly, if we do not square the differences and simply add the absolute values of the differences to compute the total penalty, then the median (or the middle value among the

numbers when they are ordered by magnitude) is the best decision. I find that result very cool!

Can you think of an experiment to compare removing outliers and not removing them in the context of skewed distributions to see if using the mean in both cases reduces the total penalty significantly?

You can delve deeper into this topic if you are interested.

6 Category Means

Angela, Steve, Dave, Igor, and Suzie – our interns from the previous chapter, sat around the table in the conference room waiting for Amanda and discussed the previous night's exciting professional basketball game.

Last chapter, the interns learned an important lesson: when you must commit to **one number** without any further information, and you measure mistakes more harshly the bigger they are (by using a squared penalty), the best long-run choice is the average.

Amanda walked in exactly at 9 am. WIth a bright smile, she said "Team, I have some good news. First off, we did very well last month. We had far fewer days of bad mismatches from our staffing levels with the customer counts. Your model improved significantly on what we were doing earlier.

Secondly, I have better data this time and I am hoping we can improve our decision further."

6.1 Staffing kiosks at malls and beaches

"I have better data this time" Amanda says, pulling up a spreadsheet on her laptop. "The same two kiosks. Same city. Same staffing problem. But . . . "

She points to the new column.

"This time, the person collecting data included the kiosk."

The interns lean in. The data now tells them *where* each customer count came from: *Mall* or *Beach*.

The key difference from the previous chapter is that the decision no longer has to be the same everywhere. Instead, the company can make one decision for the Mall kiosk and a different decision for the Beach kiosk, using the information it has about location. Amanda hoped that this can lead to an even better match between the staffing and the actual customer numbers.

Here is the data:

customers	kiosk
89	Mall
95	Mall
131	Mall

customers	kiosk
101	Mall
103	Mall
134	Mall
109	Mall
75	Mall
86	Mall
91	Mall
177	Beach
151	Beach
152	Beach
143	Beach
123	Beach
194	Beach
155	Beach
81	Beach
161	Beach
126	Beach

Amanda taps the screen.

"Last month I asked for, and you gave me, *one* figure for the number of customers for both kiosks" she says. "Now we have more information and want to do better. Can you use this extra information?"

In this scenario, we need to decide a customer level to use for determining staff level – but separately for each kiosk.

- If we are given only the data in column 1 (the *customers* counts), then based on the previous chapter, our best choice is the overall average.
- If we are given the entire table, then we have additional information and can possibly determine a different number for each kiosk.

Initially it seems to many of the interns that the new information really does not add any value. David says, "Why not just adopt the overall average again like last month? I mean how is this kiosk information helping us?"

Angela says "Let us consider an extreme scenario. What if the sales at the Beach kiosk is generally twice that of the sales in the Mall kiosk. In that case, predicting the same number for both does not seem right. Surely we will not have the same staffing level for both. So differentiating the two should make sense."

"I agree" Suzie says. "We should first look at the data. Even if the difference in sales is not as dramatic as Angela mentioned, it might still make sense to differentiate. I have computed the averages for the two kiosks. Here they are." Suzie then shows Table 6.2.

Table 6.2: Average sales at the two kiosks

kiosk	avg_customers
Mall	101
Beach	146

David: “I see your point. These sales do seem quite different with the beach kiosk getting 40% more customers per day on average. Perhaps a different model value does make sense after all. What should the value be though for each kiosk?”

Again, Igor speaks last, a little less timid than he was the last time around. “If we treat the data separately for each kiosk then we really have two problems that are identical to our problem from last month.”

Peter: “Then should we just say the model value for each kiosk’s is its own average sales?”

The rest nods in agreement. They cannot see any flaw in this.

Peter: “For the *Mall* kiosk, we can use the average of the *customers* column for only the *Mall* rows, and similarly for the *Beach* kiosk.”

Angela: “Can we verify this with a chart like we did the last time? Except that we will have one chart per kiosk.”

Suzie: “Since the computer is going to do all the work, we can also compare the average penalty per trial for the two kiosks and compare that with using the overall average.” ”

Suzie’s plots for the average penalty for different guesses for each of the kiosks appear in Figure 6.1.

Dave says “As we expected, the best decisions for the two kiosks is still the average, but computed separately for each kiosk. In the plot for the beach kiosk, the lowest average penalty occurs at its average of 146 and at 101 for the mall kiosk.”

Suzie also reported the average penalties if we used the overall average of number of customers and if we used the specific averages for each kiosk.

- Average penalty per choice if we use the overall average number of customers: 1114
- Average penalty per choice if we use kiosk-specific average for Mall: 325
- Average penalty per choice if we use kiosk-specific average for Beach: 880

We can see that the penalty is far lower if we use the kiosk-specific averages rather than one overall average. Why is the average penalty so much higher for the beach kiosk than the mall kiosk? Let us see the variability in the data for these two kiosks. See Figure 6.2.

```
kiosk_mall_beach |>
  point_plot(customers ~ kiosk, annot = "violin")
```

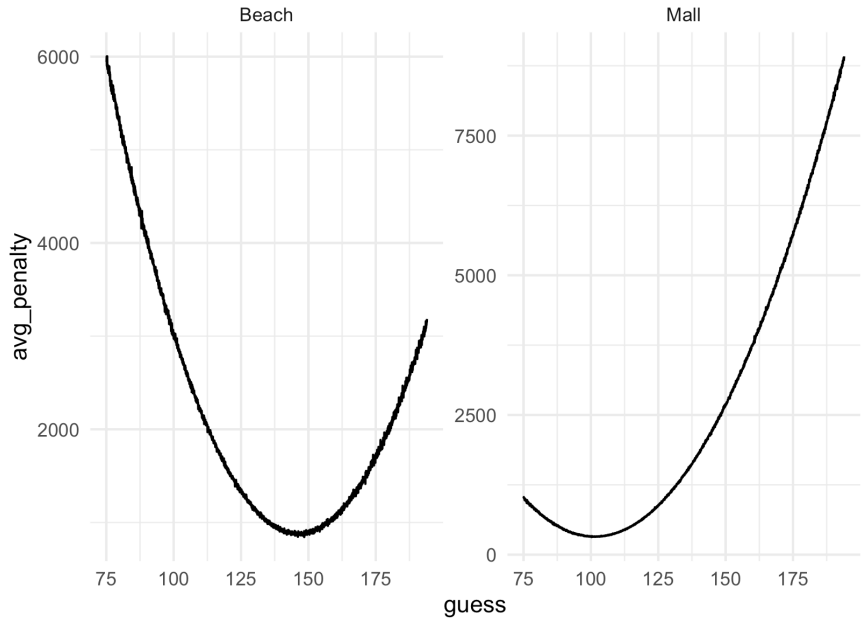

Figure 6.1: Plots of guess against average penalty for the two kiosks: As expected, choosing the average (101 for mall, and 146 for beach) generates the lowest average penalty

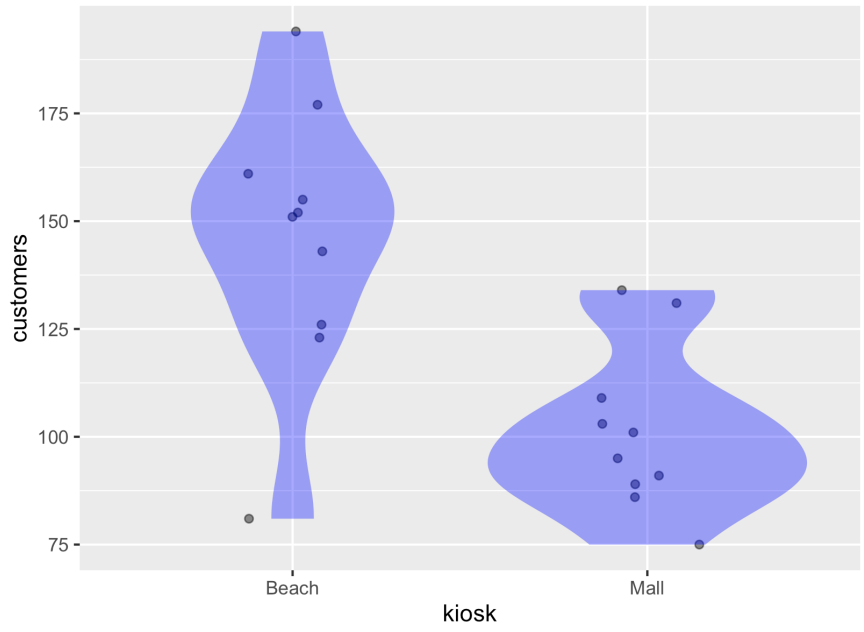

Figure 6.2: Comparison of the distributions of the customer traffic for the two kiosks: Beach has a lot more variability

Since the beach kiosk has a lot more variability than the mall kiosk, the penalty is much more as choosing the average as the decision will be off from the actual value by quite a bit on many days.

The interns present their findings to Amanda.

Amanda studies the plots and the reduction in average penalties and says “This is good. It looks like having more information surely helps us to make better decisions.”

“She continues, “the lesson is not ‘always use the mean. but’ use the best mean you’re allowed to use, given the information you have.”

7 What is a model?

Now that we have looked at the overall mean and category means as good rules for decisions (in the absence of more information), we are ready to learn about models.

People use the word *model* in many senses. In one of the more common usages, a model is an approximate representation of something for some purpose.

Figure 7.1 and Figure 7.2 show the image of a house and a miniature model of it. The miniature is a good representation of the house in some respects. For example, it depicts the overall shape and some exterior details quite accurately. If someone were about to build a new house and the architect showed them this model, they would get a good idea about the overall external appearance of the house and the color scheme as well. However, this model differs from the actual house in many ways as well – one cannot actually step inside it! We cannot get a feel for how roomy the interior will feel. We do not know how the house will look in its actual surroundings. It is accurate in some respects and inaccurate in others.

Should we consider the model in Figure 7.2 good? In the words of statistician George E. P. Box, “All models are wrong. But some are useful for certain purposes.”

All models are wrong in the sense that they are only *representations*. However, as the miniature model of the house in Figure 7.2, they can still be useful for some purpose.

We will be building many models in this course and should remember the following:

- Models are simplifications: They inherently omit details and idealize reality to make complex systems manageable.
- Utility over accuracy: The goal isn’t perfect truth, but practical application and insight for a specific goal, whether it’s scientific, business, or everyday decision-making.
- Context matters: A model’s usefulness depends on the purpose; a simple model might be great for a quick estimate, while a complex one is needed for detailed forecasting.
- Pragmatic approach: We should focus on how wrong a model can be and still be helpful, rather than getting stuck on its imperfections.

How to apply this philosophy:

- Question your models: Continuously ask how and where a model breaks down.
- Understand its limitations: Know the assumptions and simplifications behind the model.
- Adapt and evolve: Be willing to refine or discard models as new information emerges or situations change.

Figure 7.1: Picture of a house

Figure 7.2: Model of the house in Figure 7.1

7.1 Mathematical models

In this course, we will be building a specific kind of models – *statistical models*. However, before getting to *statistical* models, we will consider some simpler *mathematical models*. The term *mathematical model* can connote many different things. Let us clarify the sense in which we use the term. Suppose I have a data frame *homes* with data about many homes. For each home, we have its *age*, *floor-area_sqft*, *number_of_bedrooms*, *number_of_bathrooms*, and *price*. Here are the initial rows in the data frame.

```
homes |>
  head()
```

```
# A tibble: 6 x 5
    age floor_area_sqft number_of_bedrooms
  <dbl>           <dbl>               <dbl>
1    20            1901                   4
2    55            1614                   3
3    29            1738                   3
4    62            2137                   4
5    66            1480                   3
6     3            2492                   5
# i 2 more variables: no_of_bathrooms <dbl>,
#   price <dbl>
```

We might build a model to determine the value of *price* given the values of one or more of the other variables. In this case, let us use the variable *floor_area_sqft*. Remember, this is a model and so we do not expect it be be exact. We do not aim for the model to determine the actual *price*, but only to compute a good *estimate* of the *price*. (We have obviously simplified the situation for pedagogical purposes. House prices depend on many other factors. But that need not hold us back. You will still learn the underlying concepts.).

What might the model look like? Equation 7.1 shows the model. *For now, do not worry about how we arrived the model.*

$$\text{price} = 61{,}399 + 394\,\text{floor\_area\_sqft} \tag{7.1}$$

In the model, we have placed a *hat* over *price* because the model computes not the actual price of a home with the specified floor area and instead only computes an estimate. In statistics, we place a *hat* over quantities that estimate other quantities.

Equation 7.1 shows a *linear* model. We call it *linear* because all variables are only raised to the fist power. We have not squared or raised a variable to any other power. In this course we build only *linear models*.

We call the equation that Equation 7.1 shows a **model function**.

In our context, we use the term *model* to denote a mathematical equation of the kind that Equation 7.1 shows. Our models help us to determine the value of a variable in a data frame for any given instance. Specifically, in the house price example that we just considered, the model in Equation 7.1 enables us to estimate the *price* of a home, given its floor area.

7.1.1 *Response* and *Explanatory* variables

We call the variable whose value the model estimates as the *response* variable. The variables based on which the model estimates the *response* variable are called *explanatory* variables. We can also refer to the *response* variable as the *dependent* variable, and *explanatory* variables as *independent* variables.

7.1.2 Using the model

The house in the fifth row of the data frame has floor area of 1480. Its actual price is $553,000. What price does the model estimate? Well, we can plug the floor area into the model – Equation 7.1 – and find out. If we plug it in, we get $645,999. The model overestimates this price.

The house in the fourteenth row of the data frame has floor area of 1293 Its actual price is $561,000. What price does the model estimate? Well, we can plug the floor area into the model – Equation 7.1 – and find out. If we plug it in, we get $572,134. Quite close.

You should try out some more cases and wee how the model does. We will learn a lot more about such models, including precisely measuring their *quality* as we go forward.

I won't blame you if you are wondering "We already have the prices in the data frame. Why do we need a model to calculate these?"

Consider again the home in the fifth row. Why did the model overestimate the price? Well, you should consider that this particular home was not the only one whose floor area was in the vicinity of 1480. The data set has other homes with similar floor areas and their prices vary. If we look at homes that have floor areas between 1430 and 1530, we see that their prices range from $553,000 to $647,000. In fact, many homes below 1480 square feet in floor area have higher prices. How could that be? Well, variables other than floor area play a role in price too. So our model makes an estimate based on the whole data frame.

The next section addresses two main uses of models.

7.2 Purpose of a model

We use models in two main ways:

1. *Prediction*: Suppose we have built a model using the data in a data frame to estimate the *price* of a home, given its *floor_area_sqft*. We can potentially use this model to estimate the price of any home whose floor area we know and whose price we do not know – even if this home is not in the data set.
2. *Explanation*: Our model in Equation 7.1 tells us that the price of a home is related to its floor area. This makes sense and we could have said this even without a model. However, when people gather data about phenomena that they do not fully understand, and want to understand, they often build models to understand and explain which variables seem to be related and in what way to a variable of interest.

So, for our purposes in this course, we can define the term *model* as:

> A model is a simple story we tell about data to help us summarize, explain, or predict.

You might have noted that both of the above purposes actually require us to use a model in contexts that go outside the data that we based the model on. When we start using models outside of the data we used to build the model, we come face to face with the essence of statistics. We will get to that later in the course.

7.3 Average is a model too!

In Equation 7.1, we had a *response* variable and an *explanatory* variable. What if we do not have any explanatory variable? In the context of our home prices example, not having an *explanatory* variable is equivalent to saying:

- "I have a data set of 100 homes"
- "Here are their prices"
- "I am not telling you anything else about these homes"
- "I have selected a random home from my data set"
- "Guess the price of the home I have in mind"

Based on the game we played in Chapter 5, you first compute the average of the prices as $733,440. Your best *model* to make your guess is:

$$\text{price} = 733{,}440 \tag{7.2}$$

That is, like in the game we played, no matter which home I choose, you always state the average as your guess. Equation 7.2 shows the *model function* when we have no

The big difference between the models in Equation 7.1 and Equation 7.2 is that the second one *has no explanatory variable*. When we are not given any additional helpful information to estimate a value, our best estimate – *best model* – is the average!

8 Visualizing Mean Models

In the prior chapters, we have established the overall *mean* as the *best model* when we have no other *explanatory* variable, and the category mean as the *simplest*best model* when we have a categorical explanatory variable.

In this chapter we use the *point_plot* function to visualize these models.

8.1 Business scenario – fuel efficiency of vehicle fleet

In Chapter 5, we used a set of numbers to play a game. Now we will extend the idea to actual data frames.

You manage a company that operates a fleet of rental cars used primarily for city driving. The fleet includes a wide range of vehicles, from compact sedans to larger SUVs, each with different city fuel efficiency.

For planning purposes, the company must commit to one single number to represent the expected city fuel efficiency of a car drawn at random from the fleet. This number is used repeatedly—for estimating fuel costs, setting reimbursement rates, and budgeting operating expenses.

The company does not know in advance which specific car will be rented on a given day. Some days the car will be more fuel-efficient than expected, and on other days less fuel-efficient.

If the company assumes fuel efficiency that is too high, it underestimates fuel costs. If it assumes fuel efficiency that is too low, it overestimates costs and ties up capital unnecessarily. Larger errors in either direction are more costly than smaller ones.

The decision is not evaluated based on a single rental. Instead, it is judged by how well it performs across many rentals over time.

In this setting, the question becomes: “What single number should the company commit to so that total long-run cost is as small as possible?”

The data about your fleet is in the *mpg* data. It has many variables, but we focus here on the variable *cty* showing us the city miles per gallon of each car.

Assuming (artificially) that the company is only allowed to use the variable *cty* what single number should they commit to?

8.1.1 Committing to one number for the entire fleet

Based on our discussion in Chapter 5, we know that the the company should commit to the average of the variable *cty*. We can compute the average using the *summarize* and *mean* functions as the code below shows.

```
mpg |>
  summarize(avg_cty = mean(cty))

# A tibble: 1 x 1
  avg_cty
    <dbl>
1    16.9
```

We can also visualize it thus:

```
mpg |>
  point_plot(cty ~ 1,
             annot = "model",
             interval = "none")
```

This generates Figure 8.1.

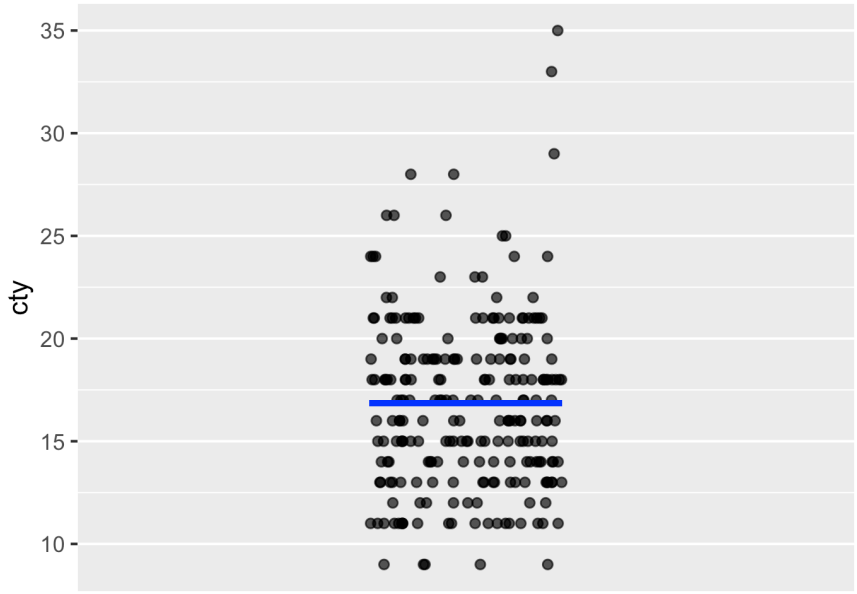

Figure 8.1: Visualizing mean *cty* as the model

The model shows the scatter plot of the points as well as a line. This line represents the model. You can see that it falls exactly at the average of the *cty* values.

Take a look at the code:

The tilde expression

```
cty ~ 1
```

in the code sets *cty* ans=d the response variable and, by placing 1 on the RHS, shows that we have no explanatory variables.

The part

```
annot = "model
```

Causes a *model* to be visualized as well. We know that the mean is our model and this plot visualizes it by drawing a line at the average *cty*.

For now, you can ignore:

```
interval = "none"
```

So, if we have to commit to one number as our estimate for the city mileage irrespective of other details about a car like its *class*, *drv* or anything else, our best commitment is to the overall average. As we have seen in Chapter 7, this is a model with just a response variable and no explanatory variables. We can write it as Equation 8.1.

$$\text{cty} = 16.9 \tag{8.1}$$

We placed a *hat* on *cty* because this is just an estimate or a model value and not an actual data point.

8.2 We can do better with more information

Instead of having to commit to one single number for the fuel efficiency for the entire fleet, what if we could use a different number for each class of vehicle? In out *mpg* data set, we have vehicles of different classes – variable *class*. This has values like *Compact*, *SUV* and *Pickup truck*.

Based on Chapter 6, we know that for each class of vehicle, we should commit to the average of *cty* for that class.

We can compute these averages easily:

```
mpg |>
  summarize(avg_cty = mean(cty), .by = class)
```

```
# A tibble: 7 x 2
  class      avg_cty
  <chr>        <dbl>
1 compact       20.1
2 midsize       18.8
3 suv           13.5
```

```
4 2seater      15.4
5 minivan      15.8
6 pickup       13
7 subcompact   20.4
```

Note how the above code is very similar to what we would use for computing the overall average, but we have added:

```
.by = class
```

To show that we do not want just one average for the whole data frame, but we want separate averages for each *class* of vehicle. As simple as that!

Let us visualize this model:

```
mpg |>
  point_plot(cty ~ class, annot = "model",
             interval = "none")
```

Figure 8.2 shows the model visualization.

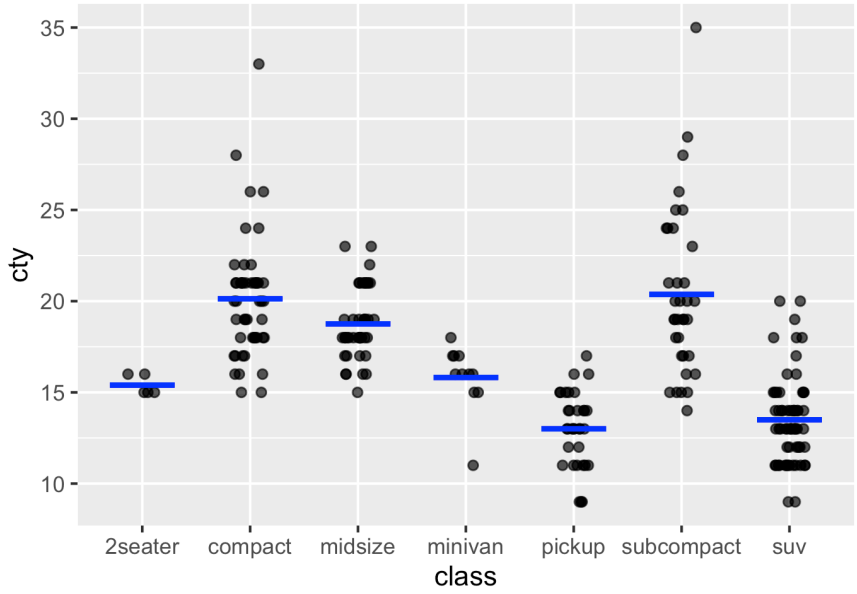

Figure 8.2: Visualizing the more sophisticated model of mean *cty* by *class* as the model when we have *class* as an explanatory variable

How do we express this model as an equation like we did with Equation 8.1?

Equation 8.2 shows the model as an equation.

$$\text{cty} = \begin{cases} 20.1 & \text{if class} = \text{"compact"} \\ 18.8 & \text{if class} = \text{"midsize"} \\ 13.5 & \text{if class} = \text{"SUV"} \\ 15.4 & \text{if class} = \text{"2seater"} \\ 15.8 & \text{if class} = \text{"minivan"} \\ 13.0 & \text{if class} = \text{"pickup"} \\ 20.4 & \text{if class} = \text{"subcompact"} \end{cases} \tag{8.2}$$

Instead of class, we can have *drv* as the explanatory variable as well. Here are code snippets that will compute the model and plot it.

```
mpg |>
  summarize(avg_cty = mean(cty), .by = drv)
```

```
# A tibble: 3 x 2
  drv   avg_cty
  <chr>   <dbl>
1 f        20.0
2 4        14.3
3 r        14.1
```

Note that we only had to change the .by to reflect that we are now computing averages by *drv*.

```
mpg |>
  point_plot(cty ~ drv, annot = "model",
             interval = "none")
```

We just changed the RHS of the tilde expression to *drv* instead of *class*.

Figure 8.3 shows the visualization.

Equation 8.3 shows the model in equation form.

$$\text{cty} = \begin{cases} 20.0 & \text{if drv} = \text{"f"} \\ 14.3 & \text{if drv} = \text{"4"} \\ 14.1 & \text{if drv} = \text{"r"} \end{cases} \tag{8.3}$$

The *mpg* data frame has another column *fl* to represent the fuel type of the vehicle. We now want to use *fl* as the explanatory variable. Try out the following using the above as examples.

- Write R code to compute the average of *cty* for each fuel type
- Write R code to visualize the model with *cty* as the response variable and *fl* as the explanatory variable.
- Write out the model equation for this scenario.

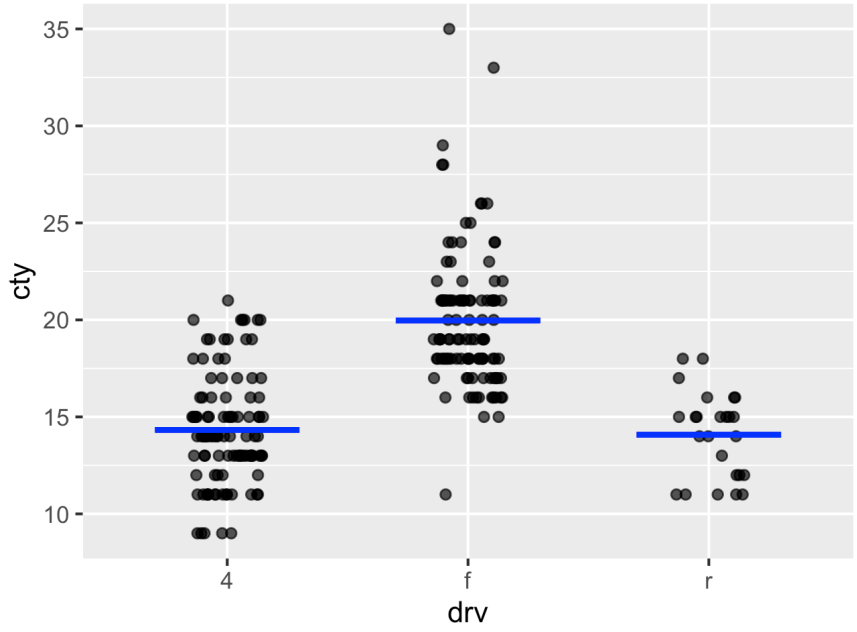

Figure 8.3: Visualizing the model of mean *cty* by *drv* as the model when we have *drv* as an explanatory variable

Part IV

More on models – least squares

Module 4: Linear Regression Models

This module fills in the details on top of the intuitive foundation for a linear model that the previous module laid.It introduces the idea of the least squares best fit line and its corresponding model. The module covers R code for building linear models for the case of a single numerical explanatory variable, a single categorical variable and a combination of one numerical and one categorical explanatory variable.It shows how to distill the model function based on the regression output.

Learning Goals

- a
- b
- c

Structure of This Module

- mention that the model gives an estimated value, but defer detailed discussion until later
- R code for building model with no explanatory variable and reconstructing the model function
- R code for building model with one numeric explanatory variable and reconstructing the model function
- R code for building model with one numeric explanatory variable and one categorical explanatory variable and reconstructing the model function

9 Line of Best Fit

In prior chapters we have looked at the situation when we have a numerical response variable and either no explanatory variable or a single categorical explanatory variable. In this chapter we look at the important case when the response variable and the explanatory variable are numerical.

In our running story, this is the third meeting of our interns with Amanda who owns a business that operates kiosks in a city at two locations. Last month, the interns used kiosk labels to help set different staffing levels for the Beach and the Mall.

Now Amanda brings a new kind of hint.

"It's not just location that seems to affect the number of customers we get." she says. "Today, I need you all to help me with a decision for the beach kiosk. At the Beach kiosk, demand seems to swing with temperature. I want a model that uses what we know *today* about tomorrow's temperature (the forecast) to predict customer traffic for tomorrow. This time, I will use the prediction not for staffing, but to decide on stock levels of items. We do not want to under or overstock too much and so good predictions can really help me."

She adds one constraint immediately:

"Keep it simple. I need something I can explain in one sentence."

In this chapter, we continue our model-building journey. A model is simply a rule that takes information we have and produces a predicted value for the response.

- With no explanatory variable, we learned in Chapter 5, that our model is "always predict the mean."
- With a single categorical explanatory variable, Chapter 6 showed us that our model should be "predict the mean of the category."

Quick Check

When we use the mean as a model, what value do we predict for every observation? Why is this considered a model rather than just a summary?

Answer

The model predicts the same value—the mean—for every observation.
It is a model because it provides a rule for generating predicted values, not just a numeri-

cal description of the data.

Quick Check

How does a category-mean model improve on the overall mean model?
What additional information does it use?

Answer

A category-mean model allows different predictions for different groups.
It improves on the overall mean model by using information about group membership instead of treating all observations as identical.

In this chapter we consider a single numerical explanatory variable. In this case we need something quite different.

9.1 The beach kiosk staffing problem

We again consider a situation where the company gets a hint before making a decision—but now the hint is numerical, not categorical.

Amanda is focused on the Beach kiosk. (The Mall kiosk is steadier, but the Beach kiosk shas more variance in daily customer numbers and so just using the average can be costly.

Management has noticed that customer traffic at the Beach kiosk varies strongly with the day's high temperature.

Over many past days, the company has recorded:

- the day's high temperature (in řF), and
- the number of customers who visited the beach kiosk that day.

Let us assume that data are available for 25 different days. (As before, for understanding the ideas, the exact number of days is not important.)

Table 9.1 shows the historical data.

Table 9.1: Daily temperature and customer counts

day	temperature	customers
D01	58	98
D02	60	104
D03	62	122
D04	64	113

day	temperature	customers
D05	66	117
D06	68	133
D07	70	126
D08	72	115
D09	74	123
D10	76	128
D11	78	145
D12	80	141
D13	58	106
D14	60	107
D15	62	105
D16	64	127
D17	66	120
D18	68	103
D19	70	128
D20	72	121
D21	74	120
D22	76	130
D23	78	127
D24	80	132
D25	58	98

Based this, the company must decide how many customers will show up tomorrow. They will u8se that to determine how much of each item to stock for sale. It can use an extremely reliable temperature forecast for its city.

The problem now is this:

> Given the forecast high temperature for a day, how many customers should the company plan for?

9.1.1 How does this differ from earlier examples we have studied?

In Chapter 5, we used no hints at all, and the mean turned out to be the best model.

In Chapter 6, our hint was categorical, and the mean within each category turned out to be the best model.

This time, the situation is fundamentally different. Our hint, temperature, is numerical, and so none of the earlier "mean-based" approaches applies directly.

One possibility would be to convert temperature into categories such as *Cold*, *Warm*, and *Hot*, and then apply the category-means approach. While this would work, it is somewhat artificial.

Two days that differ by only one degree could end up in different categories. For example, calling 50řF Cold and 51řF Warm introduces an arbitrary cutoff that has no real business justification.

Temperature does not naturally fall into clear bins. Can we do better?

9.1.2 Toward a rule-based model

Rather than assigning a few separate numbers, Amanda now wants a rule that:

> takes the day's temperature as input and produces an estimated number of customers as output.

Many such rules are possible. The company wants a simple rule that takes a temperature as input and produces a predicted number of customers as output. Let us assume that the same general pattern will continue for temperatures. It is not going to be exactly the same, but the general range and the distribution will remain the same for the next month. Therefore, for any given temperature the model's or rule's output should reasonably match the actual number of customers in the data set.

Figure 9.1 shows what Amanda is looking for at a high level.

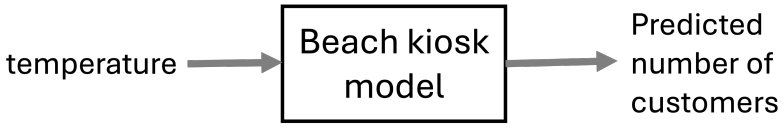

Figure 9.1: Model that takes a temperature as input and produces a predicted number of customers for the beackh kiosk – the model should be based on the data in Table 9.1

We know that with higher temperatures more people will hit the beach.

Peter has an idea. "Since Amanda is looking for a simple rule, should we consider consider a model which looks something like this?" He draws a picture. Figure 9.2). "Of course, I just made up the numbers 15 and 2. With this model, if we input a temperature of 70, the model will predict 155 customers."

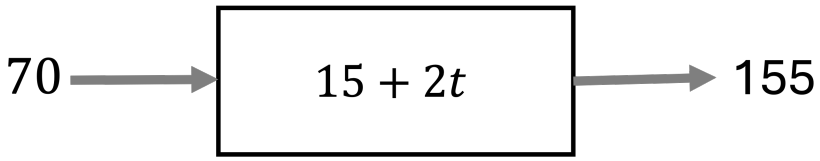

Figure 9.2: Caption

Peter further clarifies "I did not mean the specific numbers 15 and 2 literally in Figure 9.2, I am really saying that we could look for a model that has the general shape in this figure" and he quickly sketches Figure 9.3

"We just need to find *good* values for *a* and *b*.

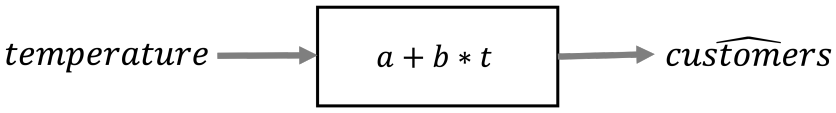

Figure 9.3: Peter's second explanatory sketch showing that he wants a model of this general form, but wants to find good values for *a* and *b* so that the model generates good predictions

Peter then writes Equation 9.1 to explain the same idea and adds "I placed a *hat* on top of *customers* because the equation only calculates an *estimate* and this could be different from an actual value."

$$\text{customers} = a + b\,\text{temperature} \tag{9.1}$$

Angela says "That's a great idea. Amanda wanted something simple. Of course we gave her very simple models earlier too, but now we have to incorporate *temperature* and I cannot imagine something that is adjusts its predictions to the temperature and is simpler than what Peter has shown."

"If we fix the values of *a* and *b*, eq-kiosk-cust-temp-model produces a predicted number of customers for any temperature. For example, if we fix a value of 15 for *a* and 2 for *b* as Peter did before, then Equation 9.2 shows our model to determine the number of customers for any temperature. If we plug in a temperature of 70, the model will give us 155 as the estimated number of customers."

$$\text{customers} = 15 + 2\,\text{temperature} \tag{9.2}$$

Igor says "I like this idea too. Its prediction changes smoothly as temperature changes, and it avoids arbitrary cutoffs."

Dave sounds a cautionary note "This is all great guys, but how do we find the *best* values for *a* and *b*? And what do we mean by *good* choices?"

Suzie: "I have an idea about how we might define *good*. Suppose we have a model – let us say for the time being that we use Equation 9.2. That is, given any temperature, we plug it into Equation 9.2 and it spits out the estimated number of customers.

Suppose we plug in 66 for temperature. Equation 9.2 says the number of customers is 15 + 2*66, or 147. In our data, the number of customers for 66 degrees is 117 for one of the rows where the temperature is 66."

David's eyes light up "Oh, we can treat this just like we did earlier and get the penalty by squaring!. This penalty will become 30*30, or 900 for that one row."

Igor then observes "For any given value of *a* and *b*, we can compute the penalty corresponding to each row of the data. With *a* = 15 and *b* = 2, we can compute the penalties for each row."

He then worked for a few minutes to come up with penalties for all the rows of Table 9.1 and came up with Table 9.2.

Table 9.2: Igor's computations: For each row he plugged in the temperature into Equation 9.2 and computed est_customers (estimated customers by Equation 9.2) and then computed the squared difference between the actual value of *customers* and *est_customers*

day	temperature	customers	est_customers	diff	penalty
D01	58	98	131	-33	1089
D02	60	104	135	-31	961
D03	62	122	139	-17	289
D04	64	113	143	-30	900
D05	66	117	147	-30	900
D06	68	133	151	-18	324
D07	70	126	155	-29	841
D08	72	115	159	-44	1936
D09	74	123	163	-40	1600
D10	76	128	167	-39	1521
D11	78	145	171	-26	676
D12	80	141	175	-34	1156
D13	58	106	131	-25	625
D14	60	107	135	-28	784
D15	62	105	139	-34	1156
D16	64	127	143	-16	256
D17	66	120	147	-27	729
D18	68	103	151	-48	2304
D19	70	128	155	-27	729
D20	72	121	159	-38	1444
D21	74	120	163	-43	1849
D22	76	130	167	-37	1369
D23	78	127	171	-44	1936
D24	80	132	175	-43	1849
D25	58	98	131	-33	1089

Angela said: "Well the total penalty is 28,312. How do we know if different values for *a* and *b* can give us a lower penalty?. In earlier weeks we only had to try different values for one thing. Here we have two things *a* and *b* to think about. What do we do?"

Suzie says: "But we have made progress. We have at least clearly defined the problem. We need to find the values for *a* and *b* that will give us the lowest possible total penalty along the lines of Igor's Table 9.2. That is, we try many different values for &*a* and *b* and compute Igor's table (Table 9.2) for that combination. We then select the combination taht produces the lowest total penalty. We want the lowest total penalty because that will mean that we have got the model predictions as close as we can to the real values in the data."

Peter says “Hey, this reminds me of something I saw my dad doing for his work. He always says that visualizing makes things easier. WHy don’t we try that first?” ”

9.2 Visualizing the model

Peter continued: “Graphically, Equation 9.2 represents a straight line. Let us generate a scatter plot of the data first.” He wrote come code and generated Figure 9.4 which showed the relationship between *temperature* and *customers*.

```
kiosk |>
  point_plot(customers ~ temperature)
```

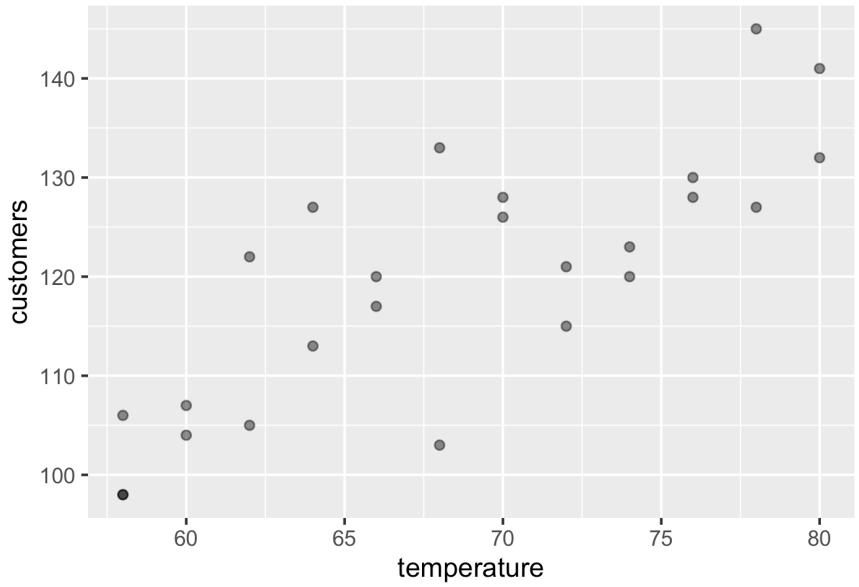

Figure 9.4: Scatterplot of *temperature* against number of *customers* from the *kiosk* data frame

Quick Check

Looking at the scatter plot of temperature versus customers:

- Does the relationship appear perfectly linear?
- Does that prevent us from using a linear model?

Explain briefly.

Answer

- The relationship is not perfectly linear; there is noticeable scatter around what seems to be a somewhat linear pattern.
- This does not prevent us from using a linear model. A linear model is a simplification that captures the overall trend, not a claim that all points lie exactly on a line.

Peter then said: "We have been using Equation 9.2 as our tentative model to explore. In my pre-calculus class, i learned that this is actually the equation of a straight line. I can add it to the earlier plot."

He then generated Figure 9.5 which has both the earlier scatter plot and the line from Equation 9.2. (You need not worry about how he added the line yet. We will learn that shortly.)

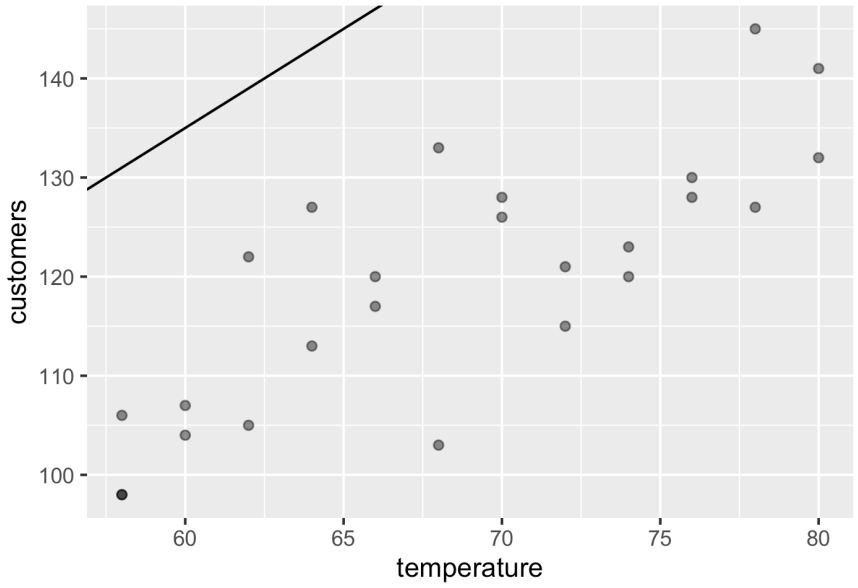

Figure 9.5: Adding a line showing the kiosk model with $a = 15$ and $b = 2$

Igor says: "Wow this model is way off the mark! We can reduce the total penalty by a lot."

Peter smiles: "Please explain to us lesser mortals Igor!"

Try to answer by yourself before reading what Igor has to say.

Igor says: "Well, the line represents the model predictions. If the predictions are good, they will be near the actual points. But our line,representing the prediction for any temperature is way above the general region where the points actually lie! For example, we see from Figure 9.5 that two days in our historical data have had a temperature of 64 degrees. The number of customers for those two days have been 113 and 127. But the tentative model from Equation 9.2 predicts 143. In fact, we can take any actual row of our data and see that the model predictions are way higher than the general area of the points. So, we clearly did not

make great initial choices for *a* and *b*! No offense Peter, I know you were just illustrating the form, and not trying to be exact."

Angela says: "OK, if we want the model predictions to reflect actual data, we would want the line to be within the point cloud and perhaps generally go through the middle of the cloud. That would make the predictions at least fall in the general region of the actual values. Let us try again with different choices for *a* and *b*. How should we change the values? Should we increase or decrease each one?"

Igor says: "In Equation 9.2 *a* represents where the line intercepts the y-axis at x = 0. I learned this in my pre-calc class. In that class they called *a* the *intercept*."

Suzie jumped up "I know why. they call it the intercept because that is where the line *intersects* the y-axis. But in this plot, the line does not intersect the y-axis at all."

Igor: "It does intersect the y-axis, if only we zoom out and show the origin of the plot where x=0 and y=0). Currently the x-axis only starts from around 55."

Peter quickly modified the plot to show the origin. Figure 9.6 shows his new plot. This is effectively the same plot as Figure 9.5, only zoomed out so that we can see more.

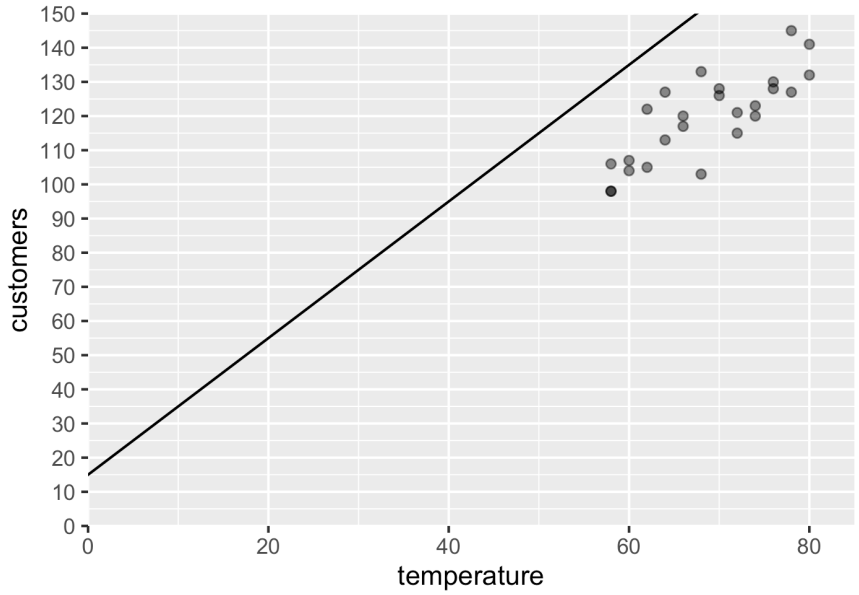

Figure 9.6: Model with *a* = 15 and *b* = 2 plotted to reveal the origin of the plot so as to see the intercept

Even though a temperature of 0řF is well outside our data range, visualizing the intercept helps us understand what changing *a* actually does to the line.

Suzie: "From the plot, we can see that the line correctly intersects the y-axis at 15 since *a = 15*. What can we do to improve the line so that it goes through the points instead of way above them?"

David opines “To make the line go through the point cloud, it looks like we need to make the line less steep, almost like grabbing the top right of the line and pulling it down. How do we achieve that?”

Peter’s pre-calculus comes to the rescue again. “The value for *b* determines the steepness of the line. So, we can decrease it to get the line to go through the cloud of points. Perhaps we should decrease it from 2 to 1.5 and see what happens.”

Peter modifies his plot again. Figure 9.7 shows his new plot.

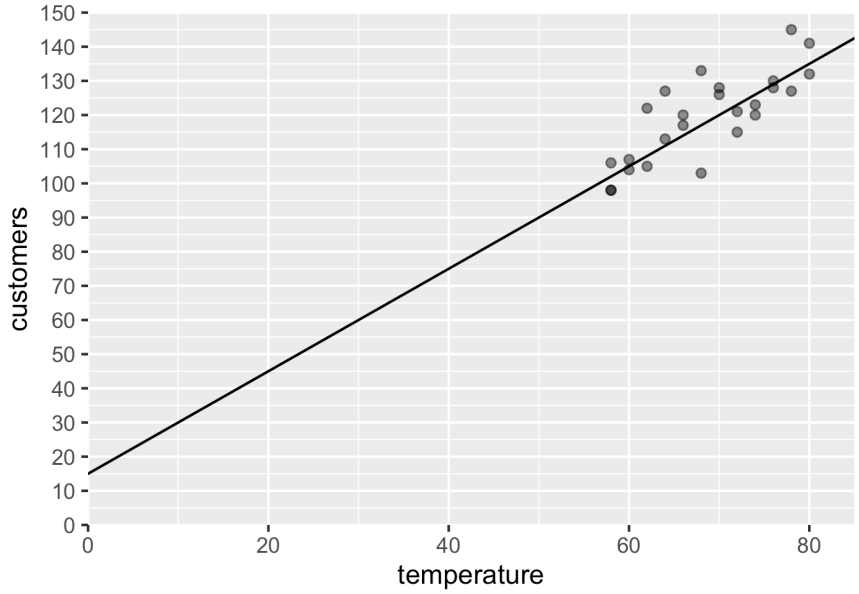

Figure 9.7: Second attempt: model with $a = 15$ and $b = 1.5$

Angela says: “Wow! that looks so much better! But can we do even better?”

Igor produced the table for this model and announced: “We have smashed the total penalty from 28312 to 1469! But I still don’t know if that is the lowest possible. Maybe there are other values of *a* and *b* that can reduce it even more.”

Peter, tired of generating more and more plots, says “We can keep on trying – we have an infinite number of choices for *a* and *b*. How do we find the *best* line? What does *best* even mean?

9.3 What do we mean by the *best* model?

In the previous section, the team tried a few values for *a* and *b* and our second attempt proved to be much better than the first. However, we want the *best* line. But what does that mean?

David says “Before visualizing the models, we had already identified what we mean by *best*. We had already established that the *best* model or *line*, is the ones that generates the lowest

total penalty as we computed in Table 9.2. That is, we choose the values for *a* and *b* and mimic the process we used for @btl-igor-computations-for-15-2 and choose the values that produce the smallest total penalty. But how do we do that when we have an infinite number of possibilities."

Angela says: "Can we not repeat what Suzie did and generate a plot for various values and visually see what the best values are?"

Peter says: "Guys, before we think about that, I can help to visualize the sort of computations we did in Table 9.2 but for the line in Figure 9.7." and comes up with Figure 9.8.

He goes on to explain: "We are after a model that produces *good* predictions. Therefore the difference between the actual value and the model prediction (the column *diff* in Table 9.2) matters. The lower that difference is the better." He then generates Figure 9.8 to visualize the previous model ($a = 15$ and $b = 1.5$) with the differences between the actual values and model predictions explicitly marked for three chosen points. These differences are usually called *errors* or even better, *residuals*.

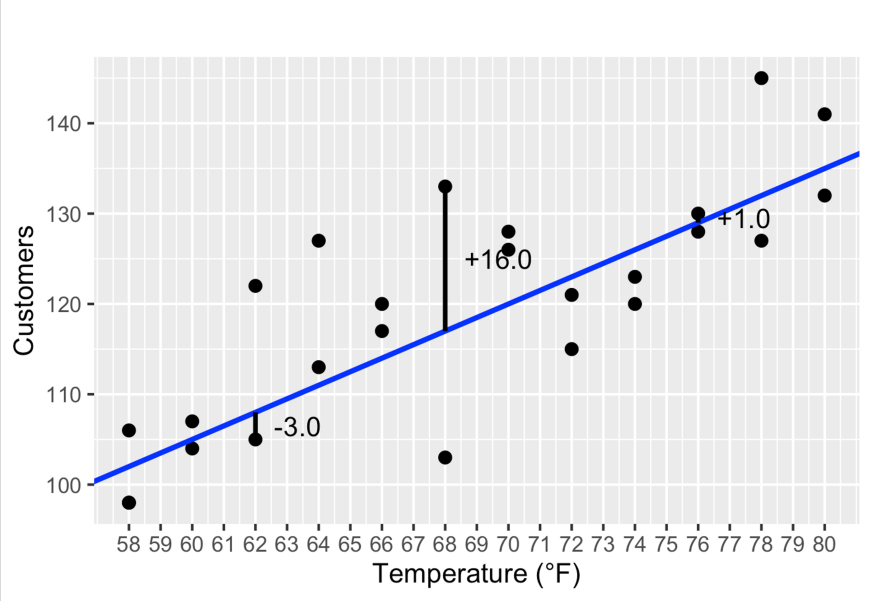

Figure 9.8: Model with $a = 15$ and $b = 1.5$ visualized along with the *residual* or error for three chosen points

> **Common Misconception**
>
> The error or *residual* for a prediction is **not** the shortest distance to the line. It is the **vertical difference** between the observed value and the predicted value. This is because our model predicts the response (customers) and we compare that to the actual temperature for any point.

David points to Figure 9.8 and says: "For the point at *temperature* = 62, the actual data had *customers* = 105, whereas our model predicts 108. So the model is off by -3 (if we compute

the residual using this equation." and he writes out Equation 9.3.

$$\text{residual} = \text{actual value} \ \ \text{model prediction} \tag{9.3}$$

Peter continues: "“For the data point with *temperature* = 68, the model predicts 116, whereas the actual value was 132 for a residual of +16. Finally, for the point where the *temperature* is 76, the model prediction is 129, whereas the actual value was 130. The model did well on this point. In this way, we can compute the *residual* for every point and square it. We are not particularly concerned about the actual sign of the residual since we are going to square it anyway. This is exactly what Igor showed us in Table 9.2."

Quick Check

If a model predicts 120 customers for a given temperature, and the actual number is 132:

- What is the prediction error?
- Is it positive or negative?

Answer

The prediction error is 132 120 = 12.
The error is positive because the actual value is higher than the predicted value.

Angela says: "If we draw any line through these points, each line will have low *absolute residuals* for some points and higher ones for others. But we have many points and so we want an *overall measure of how close the line is to all the points* by considering all the *absolute residuals*. Even if a line is spot-on for a few points, it is not very useful if it is wildly off for many others. That overall measure is in fact the total penalty from Table 9.2. It is very nice that we have seen the computations and also seen its visualization to make sure we understand all this. I sure am glad we are doing this as a team!"

Igor says, "Yes, now I too see the connection between my table (Table 9.2) and Peter's visualization of the concept. So, we will square the residuals from all the points and add them all up. That is how much total penalty the line/model has when we consider *all* data points. As Amanda wanted, squaring punishes bigger residuals much more than smaller ones. I have heard that this is a standard theme in statistics."

What's the deal with squaring?

We square errors for the same reason we used squared deviations when defining variance:

- positive and negative errors should not cancel out
- larger errors should count more than smaller ones

Least squares regression that we study in this course is built on the same logic as vari-

ance. In a later chapter, we will see this idea pop when we look at the explanatory power of a model. Hang tight till then!

Quick Check

Why don't we choose the line that minimizes the **sum of errors** instead of the **sum of squared errors**?

Answer

Positive and negative errors would cancel out if we summed errors directly.
Squaring errors ensures that all errors contribute positively and that larger errors are penalized more heavily.

Clearly, we want a line that is as close as possible to all the points overall. Honestly, the line in Figure 9.8 seems to be pretty good, at least visually.

David says: "For this model (a = 15, b = 1.5), the total of all the squared residuals is: 1,469. Is there a line that can give an even lower value? What are the values for *a* and *b* for which we get the *lowest* possible total of all the squared residuals?"

The team now understands exactly what they mean by the *best* model. They are looking for values for *a* and *b* that will result in the lowest total penalty. But they are stuck on to how to actually get it.

Catching the thread about trying to generate a plot to identify the best values just like Suzie did in the first week, Angela returns to her idea. "Suzie, can you do you pull off your plotting magic again, please?"

Suzie: "Well angela, that would be a three-dimensional plot because we will have *a*, *b*, and the *total penalty* all on the same plot. I can do it, but I am afraid it will not be much help."

Igor: "Wht not just print out a large table of all combinations of *a* and *b* and the total penalty for each combination? We can then look through the table and select the best value – or even ask the program to look through the table and find the values for us?"

Suzie says "We can Igor, but I have two issues. Firstly, when I try the various combinations of *a* and *b*, I can only do it in some intervals like 0.1 or something like that. What if the best solution was between two of my values? We will then not get an exact solution, but we will get something very close. A bigger problem is what if Amanda comes up with two more variables to use in our predictions? Then trying all combinations can generate a very large number of rows, perhaps millions. There has to be a better way."

Steve, who has been silent for a long time has actually been having a nagging feeling that he has done something similar in one of his classes. He now says "I think we learned something like this in my econ class. Rather than making Suzie write a whole new program, I think there is a function in R, based in calculus that can readily do this for us! Let me try."

And he runs the code below: which generates the *best* values for *a* and *b* that guarantee that the *sum of squared residuals* is the smallest possible.

```
kiosk |>
  model_train(customers ~ temperature) |>
  coef()

(Intercept) temperature
      22.06        1.42
```

Steve then explains: “Ah, now I remember what the output means. We see two values in the output: *intercept* and *temperature*. The first is *a* and the second is *b*! These are exactly the values of *a* and *b* that minimize the sum of squared residuals.”

From the output, Steve then writes the model equation Equation 9.4, and says:

“This equation shows the *best model* for our problem, given that we want a straight line and want to minimize the sum of squared residuals.”

“Our econ teacher called this the least-squares regression line.”

$$\text{customers} = 22.06 + 1.42\,\text{temperature} \tag{9.4}$$

Key Idea

Since there is an infinite number of possible choices for *a* and *b*, we need a systematic way to find the values that minimize the total of the squared residuals. Mathematically, this problem can be solved using calculus. Practically, we sit back and let software do this for us.

Pause & Think

Complete the sentence:
“A least squares regression line is the line that _____.”

Answer

A least squares regression line is the line that minimizes the sum of squared vertical prediction errors, or the sum of squared residuals (same thing).

Igor is at it again. He announces “The total penalty for this is 1388. This beats the 1469 we got earlier. And, according to Steve, this is the *best*. I am quite impressed that we had eyeballed a solution very close to the best!”

The team found a line that *minimizes the sum of squared residuals*. Technically this is called the *least-squares criterion* and plays a very important role in statistics and data analysis.

They explained the model to Amanda. After a lot of questioning to understand their decision criteria, Amanda agreed to use this model for the next month.

Amanda was thrilled with their work. “Great work team. Talk to me when you graduate. I want to assemble a business analytics team for my businesses and I sure can do with a team like yours.”

Pause & Think

Suppose tomorrow’s high temperature is 73řF. Your model predicts 124 customers.

- Does this mean exactly 124 customers will arrive?
- What other factors (besides temperature) might cause the actual number to be higher or lower?

What does this tell you about what a regression model can—and cannot—do?

Suggested answer

- No. 124 is only a model prediction. In reality the number of customers may be 124 or higher or lower.
- The model acts as if only the temperature determines the number of customers. In reality, many other factors play a role too. Perhaps there is a football game in town and that draws people away. Perhaps the humidity is unusually high and people stay away. Many factors play a role in this outcode. Our model uses a simple rule that can potentially perform decently without being accurate.

In later chapters we will see that we can include more variables in a model and improve its performance.

9.4 Optional enrichment topic: Beyond straight lines

In this chapter, we have assumed that a straight line is a reasonable model for the relationship between temperature and customer traffic. In many business settings, this assumption works well: simple models are easy to interpret, easy to communicate, and often good enough for decision-making.

However, not all relationships are well captured by a straight line.

9.4.1 When a straight line may not be appropriate

In some situations, a scatter plot may reveal patterns such as:

- curvature (the relationship bends),
- diminishing returns (the effect of increases slows down), or
- threshold effects (behavior changes after a certain point).

For example, at very high temperatures, customer traffic at a beach kiosk might level off or even decline as conditions become uncomfortable. A straight line cannot capture this kind of behavior well, no matter how its slope and intercept are chosen.

In such cases, forcing a linear model may lead to systematic residual patterns, indicating that the model is missing something important.

9.4.2 A flexible alternative: LOWESS smoothing

One way to explore nonlinear relationships is through a technique called LOWESS (short for locally weighted scatter plot smoothing).

Figure 9.9 shows an example of a LOWESS model using the *mpg* data frame. In this data frame the *displacement* of an engine is related to the highway mileage *hwy*, but not in a linear way. The LOWESS model better describes it than a straight line would. Like the earlier line model, this too can take in a value for *displ* and provide an estimated value for *hwy*.

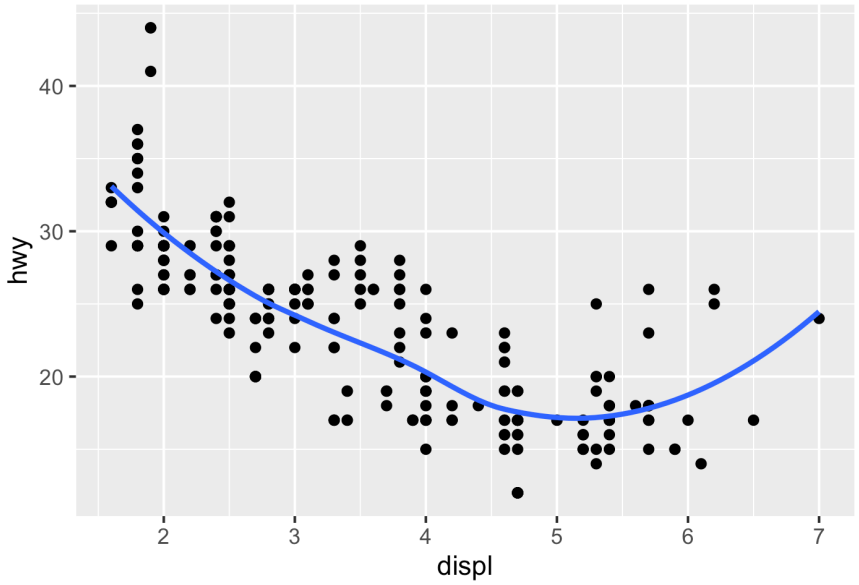

Figure 9.9: LOWESS model for *displacement* vs *cty* in the *mpg* data frame showing how it captures the curvature in the relationship better than a lien can

Rather than fitting a single global rule like a straight line, LOWESS works by:

- focusing on a small neighborhood of points around each value of the explanatory variable, and

- fitting simple local models that adapt to the data in that neighborhood.

The result is a smooth curve that follows the overall pattern of the data without requiring us to specify a particular functional form in advance.

LOWESS is especially useful for:

- exploratory analysis,
- visualizing trends, and
- diagnosing whether a straight-line model is reasonable.

It is not intended to replace regression models in all cases, but rather to help us understand the structure of the data.

9.4.3 Models as choices, not defaults

The key takeaway is not that linear models are “wrong,” but that models are choices.

- A straight line is a good choice when the relationship is approximately linear and interpretability matters.
- A nonlinear model may be a better choice when the data clearly suggest curvature or changing behavior.
- Flexible tools like LOWESS help us see what the data are trying to tell us before we commit to a specific model.

In all cases, the central modeling question remains the same:

> Does this model capture the important structure in the data well enough to support the decisions we want to make?

This mindset—treating models as purposeful approximations rather than automatic formulas—will guide us throughout the rest of this book.

10 Visualizing models

In this chapter, we will first revisit how to build visualize the two *mean* models we saw earlier in Chapter 5 and Chapter 6. To be sure, we already covered these in Chapter 8, but repeat it with different examples to provide reinforcement.

We will then learn how to visualize line models that we learned about in Chapter 9. This will set the stage for us to learn how to *build* these models and construct the models as equations.

10.1 Recap of models

From prior chapters, we have learned that a model takes as input the values of the explanatory variable(s) (if any) and outputs an estimated value of the outcome variable.

We started off in Chapter 5 by showing that, in the absence of an explanatory variable, the *mean* is the best model. For example, our *mpg* data frame contains data about several cars. Let us suppose that our outcome variable is *hwy*, the highway mileage of a car.

If we are given only the data from that single column *hwy*, and nothing else, the best *model* we can come up with for *estimating* the highway mileage of a car is to always provide the average highway mileage from the data set.

We then considered, in Chapter 6, the case where we had a single explanatory variable, a categorical one and saw that here too the same idea applies, except that the best model for a category is the average for that category.

We then studied, in Chapter 9 the case of a single explanatory variable again, but this time the explanatory variable was numerical. We learned there about the *least squares criterion* to find the best model.

Chapter 8 showed us how to visualize the models we built around the idea of the *mean* as a model in Chapter 5 and @#sec-category-means. In this chapter, we use new examples to briefly review how to visualize those models and also compute and write out the model functions. We then move on to visualizing our line models from Chapter 9 and how to build and reconstruct the model function as an equation.

We will first learn how to visualize the models. We then look at how to generate the model function in equation form.

10.2 Visualizing simple mean models (no explanatory variable)

We take a different dataset this time. The data frame *txhousing* contains real estate data for the state of Texas for several years. We first follow the good practice of understanding the dataset before jumping in and building models. Table 10.1 shows the first 10 rows .

```
txhousing |>
    head(10)
```

Table 10.1: First 10 rows from the *txhousing* data frame

city	year	month	sales	volume	median	listings	inventory	date
Abilene	2000	1	72	5380000	71400	701	6.3	2000
Abilene	2000	2	98	6505000	58700	746	6.6	2000
Abilene	2000	3	130	9285000	58100	784	6.8	2000
Abilene	2000	4	98	9730000	68600	785	6.9	2000
Abilene	2000	5	141	10590000	67300	794	6.8	2000
Abilene	2000	6	156	13910000	66900	780	6.6	2000
Abilene	2000	7	152	12635000	73500	742	6.2	2000
Abilene	2000	8	131	10710000	75000	765	6.4	2001
Abilene	2000	9	104	7615000	64500	771	6.5	2001
Abilene	2000	10	101	7040000	59300	764	6.6	2001

We see that for the first 10 rows, *city* and *year* are the same, but there is variation in the other variables. The dataset has 8602 observations or rows. Use the command ?txhousing in RStudio to answer the following questions.

Quick Check

What is the *unit of observation* int his dataset? Unfortunately, the codebook does not give sufficient details. Se if you can use ChatGPT or other chatbot to find the answer.

Suggested answer

For every city in Texas, the dataset has its real estate activity for one month. The data spans 12 months each from 2000 to 2014 and for 2015 7 months from Jan to July. There are 46 cities. For each city we have data for 180 + 7 = 187 months. Therefore the unit of observation is the real estate activity for a single city for a single month.

Quick Check

Explain precisely what the first row tells us.

Suggested answer

The first row tells us that in Jan 2000, the city of Abilene had 72 sales, 701 listings, and so on.

Quick Check

Why does the data frame have 8602 rows?

Suggested answer

187 months times 46 cities = 8602.

If you had access to only the column *listings* and we had to build a model to predict *listings* based on no other information, what would you predict?

```
txhousing|>
    point_plot(listings ~ 1, annot = "model", interval = "none")
```

In this example, our outcome variable is *listings* and we have no explanatory variable. Therefore we use "1" on the RHS of the tilde expression.

This code shows you how to add a model as an annotation. We had used "annot=" earlier for generating violin plots. Here we use:

annot = "model", interval = "none"

The first part asks the function to annotate the scatter plot with a model. For now, we will just use the second part "interval = . . . " without explanation. Later in the course, you will see why we use this.

Let us see one more example of visualizing a simple mean model.

The data frame *Boston_marathon* contains data about the results of the Boston Marathon over the years.

Like we did with the *txhousing* data frame, we should first understand the data frame before processing it. Take a look at the codebook for the *Boston_marathon* data frame and answer the following questions. Try to recall how we got the codebook for *txhousing* and use the same approach here. The codebook is not too informative, but just viewing it using View(Boston_marathon) might tell you what you need to know.

Quick Check

How many variables does the data frame have?

Suggested answer

Six

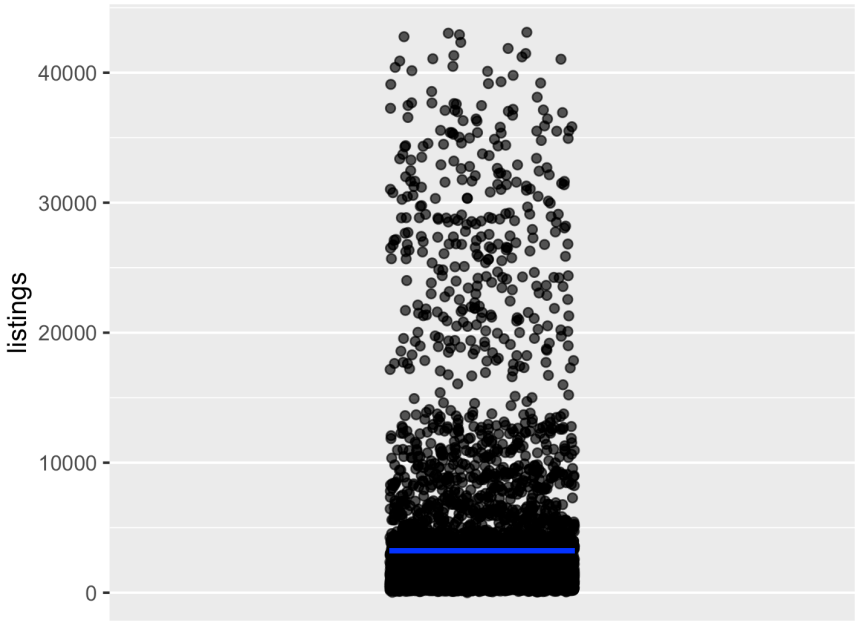

Figure 10.1: Scatter plot of *listings* from the *txhousing* data frame annotated with a model: the model is the blue horizontal line at the average value of listings (3217)

Quick Check

What is the unit of observation? That is, what does a row tell us about?

Suggested answer

Each row tells us about the result of a year's race for a gender. The unit of observation therefore is the result of a race in a year – with the races for males and females being considered as different races.
We have two rows for years starting from 1972 when women started participating. Prior to that, from 1897 to 1971, we have only one row per year.

Quick Check

Which column shows the finishing time?

Suggested answer

The variable *time* shows the finishing time in seconds and the colvariableumn *minutes* shows it in minutes.

Quick Check

How can you verify the suggested answer from the previous question?

Suggested answer

Multiply *minutes* by 60 and that should equal *time*. However, we cannot use the variable *time* as it is not numerical. (Take a close look.)

Let us visualize a model for the variable *minutes*. As before, we generate the scatter plot and then annotate with a model. We changed the data frame and the tilde expression. Figure 10.2 shows the plot. The model line is at *minutes = 144*.

```
Boston_marathon |>
  point_plot(minutes ~ 1, annot = "model",
             interval = "none")
```

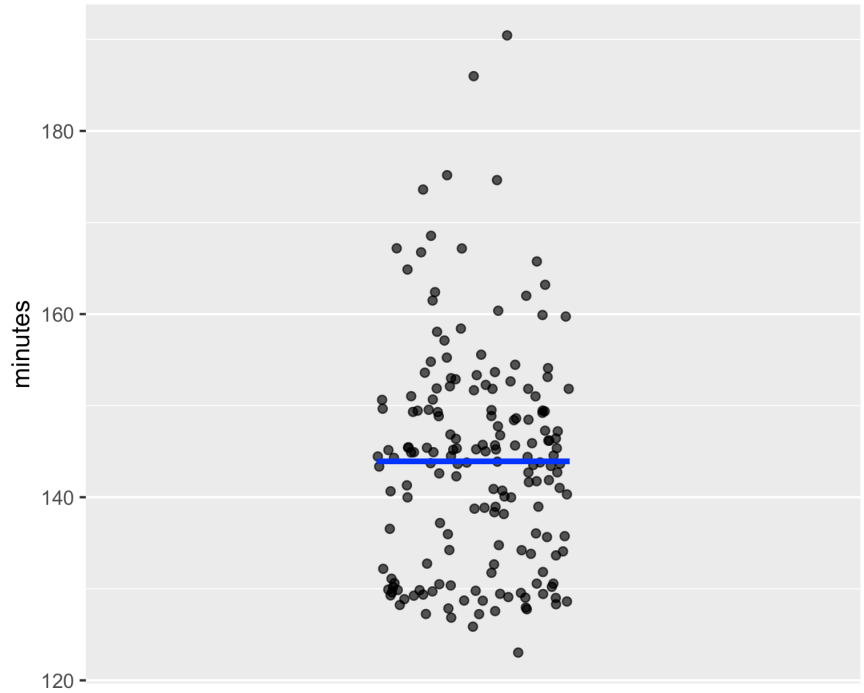

Figure 10.2: Scatter plot of *minutes* from the *Boston_marathon* data frame annotated with a model

10.3 Visualizing category mean models (single explanatory variable that is categorical)

We built a model for *minutes* from the *Boston_marathon* data frame assuming no explanatory variable. What if we used *sex* as the explanatory variable? Figure 10.3 shows the model. This time we have a separate mean model for each sex.

```
Boston_marathon |>
```

```
point_plot(minutes ~ sex, annot = "model",
           interval = "none")
```

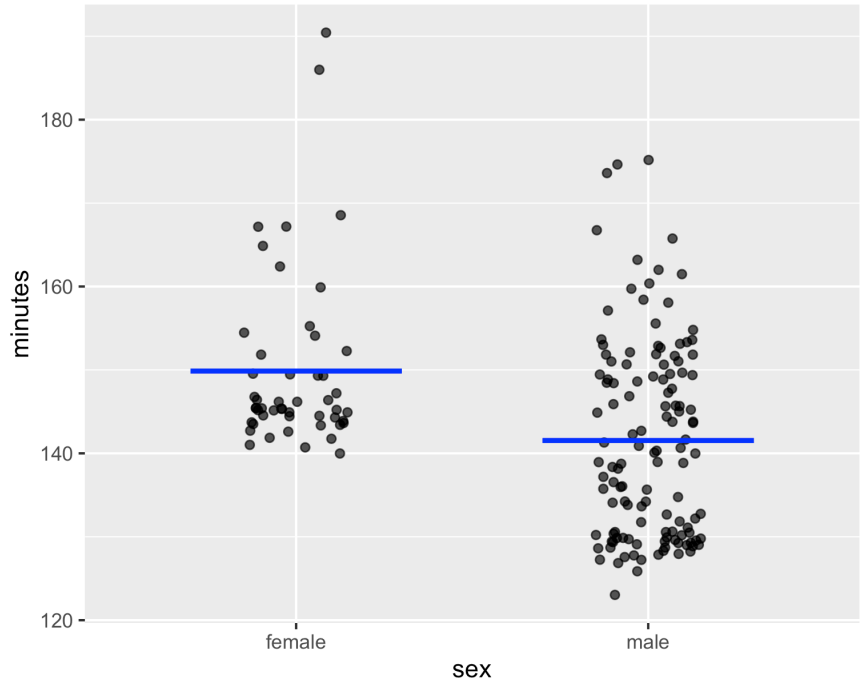

Figure 10.3: Boston marathon model for *minutes* with *sex8 as the explanatory variabl

For another example, we now use the *mpg* data frame and plot a model for the city. mileage (variable *cty*) with the kind of drive (variable *drv*) as the explanatory variable. Figure 10.4 shows the plot. This time we see that the model is the mean for each kind of drive.

```
mpg |>
  point_plot(cty ~ drv, annot = "model",
             interval = "none")
```

10.4 Visualizing least-squares line models (single explanatory variable that is numeric)

We use the *Anthro_F* data frame. Take a look at its codebook and answer the following question.

> Quick Check
>
> What is the unit observation?

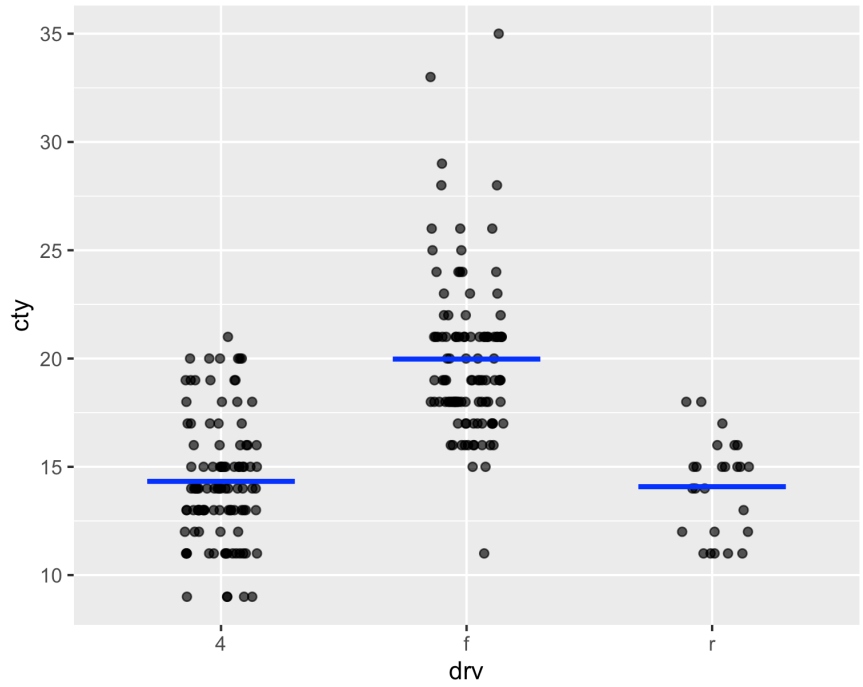

Figure 10.4: Model of category means for *cty ~ drv* for the *mpg* data frame

Suggested answer

Each row gives us information about a single college woman in the age range 18-25. SO the unit of observation is a *single college woman in the age range 18-25*.

We would like to visualize a model that predicts the body fat (variable *BFat*) with *Wrist* as the explanatory variable. Figure 10.5 shows the scatter plot and the least squares line.

```
Anthro_F |>
  point_plot(BFat ~ Wrist, annot = "model",
             interval = "none")
```

As we saw from Chapter 5, the *best* model would be to predict the average. Do you recall how to compute the average?

```
txhousing |>
    summarize(avg_listings = mean(listings))

# A tibble: 1 x 1
  avg_listings
         <dbl>
1           NA
```

Why is the result not a number? Why did we get *NA*?

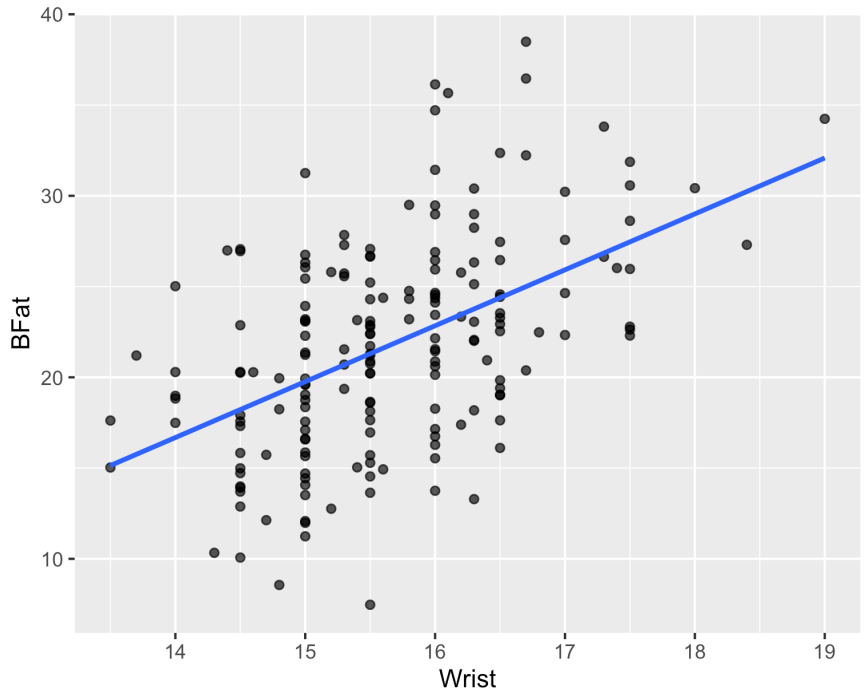

Figure 10.5: Least squares line model for the *Anthro_f* data frame: *BFat* as the response variable with *Wrist* as the explanatory variable

You might recall from Section 1.8 that *NA* means missing value. Why is the mean a missing value? The *txhousing* data frame has some missing values in the column *listings*. As a result, it is unable to compute the average. Suppose I asked you what is 2 + 3, you can say "5." since you have both the numbers that you want to add up.

However, if I asked you what is *NA* + 5? You cannot say what it is because you do not know the values of the two numbers you are trying to sum. In general, whenever *NA* is involved in an operation, the result is always *NA*.

This is very inconvenient. Wht does the system simply not ignore the *NA*s and compute the average of the remaining numbers? By default it does not do that, but we can make the system do that in the following way.

```
txhousing |>
    summarize(avg_listings = mean(listings, na.rm = TRUE))

# A tibble: 1 x 1
  avg_listings
         <dbl>
1        3217.
```

When you add "na.rm = TRUE", R leaves out the missing values and computes the average of the rest. Remember, R is case sensitive and so you have to type na.rm in lowercase and TRUE in uppercase. Otherwise you will get an error.

So we saw that the mean of *listings* is 3217 and that is our model value.

We can write the model function as Equation 11.1

$$\text{listings} = 3217 \tag{10.1}$$

We had built the mean model by explicitly writing code to compute the average.

In the earlier chapters, we have built This can get cumbersome as we will need to build the model differently depending on the type of explanatory variable. Fortunately, the *LSTbook* package provides a single function – the *model_train* function through which we can build all the models for this course.

Let us see how to use it.

11 Finding the Model Function

Our journey so far has introduced the idea of models and shown us the intuition, and rationale behind models. We have also seen how to visualize the models. However, we have not lingered too much on systematically building models and reconstructing the model function from what R shows us about the models.

This chapter closes the loop.

11.1 Finding the model function for models with no explanatory variables

When we have a numerical outcome variable and no explanatory variable, Chapter 5 showed us that the *best* model would be to predict the mean or average. We look at two ways of building such models:

- building the model by explicitly computing the mean
- using the model_train function (our go to method from now on)

11.1.1 Building mean models by explicitly computing the mean

Do you remember how to compute the mean oif a variable?

> Quick Check
>
> Given a hypothetical data frame named *dat* do you recall how to compute the mean of one of its variables named *col* using R?

> Suggested answer
>
> dat |> summarize(avg_col = mean(col))

Using the *txhousing* data frame, we can build a model for the number of listings as follows:

```
txhousing |>
    summarize(avg_listings = mean(listings))
```

```
# A tibble: 1 x 1
  avg_listings
         <dbl>
1           NA
```

Why is the result not a number? Why did we get *NA*?

You might recall from Section 1.8 that *NA* means missing value. Why is the mean of *listings* a missing value? The *txhousing* data frame has some missing values in the column *listings*. As a result, our code is unable to compute the average. Suppose I asked you what is 2 + 3, you can say "5." since you have both the numbers that you want to add up.

However, if I asked you what is *NA* + 5? You cannot say what it is because you do not know the values of the two numbers you are trying to sum. In general, whenever *NA* is involved in an operation, the result is always *NA*.

This is very inconvenient. Wht does the system simply not ignore the *NA*s and compute the average of the remaining numbers? By default it does not do that, but we can make the system do that in the following way.

```
txhousing |>
    summarize(avg_listings = mean(listings, na.rm = TRUE))
```

```
# A tibble: 1 x 1
  avg_listings
         <dbl>
1        3217.
```

When you add "na.rm = TRUE", R leaves out the missing values and computes the average of the rest. Remember, R is case sensitive and so you have to type "na.rm" in lowercase and "TRUE" in uppercase. Otherwise you will get an error.

So we saw that the mean of *listings* is 3217 and that is our model value.

We can write the model function as Equation 11.1

$$\text{listings} = 3217 \tag{11.1}$$

We have built the mean model by explicitly writing code to compute the average.

> Quick Check
>
> In Chapter 5, we saw a situation where the interns had to find a single fixed number to represent the number of customers who will show up at a kiosk on any day. What could be a business analogue of that in the *listings* example from the *txhousing* data frame?

Suggested answer

The analogue would be: We need to build a model for the number of listings in a typical month, and we were allowed to give just a single fixed number for it. Also, like before, bigger errors are penalized more strongly through squaring prediction errors.

11.1.2 Building the mean model using *model_train*

The *model_train* function greatly simplifies the process of building models irrespective of the form of the model. We do not need to worry about what types of explanatory variables and how many of them are involved.

Let us see how to use it to build the model for *listings* with no explanatory variables.

```
txhousing |>
    model_train(listings ~ 1)
```

```
A trained model relating the response variable "listings"
to explanatory variable "".

To see relevant details, use model_eval(), conf_interval(),
R2(), regression_summary(), anova_summary(), or model_plot(),
or the native R model-reporting functions.
```

We have built the model, but the output does not give us any details about the model to enable us to reconstruct the model function.

We can fix that easily by calling another function – *coef* – to get the model *coefficients*.

```
txhousing |>
    model_train(listings ~ 1) |>
    coef()
```

```
(Intercept)
       3217
```

The output says that the *intercept* is 3217 (rounded). The model output does not show any other coefficients because we do not have any explanatory variables. From this, we can reconstruct the model function as Equation 11.2 (same as Figure 10.1). We explain below exactly how we went from the coefficients in the output to Equation 11.2.

$$\text{listings} = 3217 \tag{11.2}$$

You night recall from Chapter 9 that in general when we have a single explanatory variable, we are trying to find the best values for *a* and *b* in an equation of the from that Equation 11.3 provides to compute an estimated value for the outcome variable (hence the *hat*)

$$\text{outcome} = a + b \text{ explanatory\_var} \tag{11.3}$$

The *intercept* in the model coefficients output by the *coef* function corresponds to *a*. If there is another coefficient in the output that corresponds to *b*. In the current example we have only the *intercept* and no other coefficient and so we use only *a* in reconstructing the model function as Equation 11.2.

Let us consider another example using the *mpg* data frame. We will now build a model for *cty* with no explanatory variables.

```
mpg |>
  model_train(cty ~ 1) |>
  coef()
```

```
(Intercept)
       16.9
```

We see that the *intercept* is 16.86 (rounded). This is, of course, the mean of *cty*. Equation 11.4 shows the model function.

$$\text{cty} = 16.9 \tag{11.4}$$

> **Quick Check**
>
> Can you think of a business scenario where this sort of model might be used?

> **Suggested answer**
>
> If a company had a fleet of cars and needed to estimate something based on the city mileage of the fleet as a whole and we had to settle on juse a single representative number for the whole fleet, then the average would be the best – assuming again that bigger errors are more heavily penalized.

Let us close out with another example from the *Anthro_F* data frame to predict body fat (variable *BFat) with no explanatory variables.

```
Anthro_F |>
  model_train(BFat ~ 1) |>
  coef()
```

```
(Intercept)
       21.8
```

This shows an *intercept* of 21.8 (rounded). Equation 11.5 shows the model function.

$$\text{BFat} = 21.8 \tag{11.5}$$

Quick Check

As before, what might be a usage scenario for this model?

Suggested answer

If, for some reason we needed a representative number – a single number – for the body fat of a group of college age females in the age range 18-25 (as the data set on which we based the model), then this model would work.

Steps to build the model function for models with no explanatory variable

1. Determine the tilde expression for your model (like listings ~ 1, or some such)
2. Pipe the data frame to the *model_train* function and pass the tilde expression as well
3. Pipe the output of the *model_train* function to the *coef()* function
4. Look at the results and map the *intercept* to *a*
5. Construct the model function by substituting the name of the outcome variable and the value of *a* in the equation with the intercept:

$$\text{outcome\_var} = a$$

11.2 Finding the model function for models with one explanatory variable alone

Mirroring the process from the previous section, we have two ways of building these models:

- building the model by explicitly computing the category means
- using the model_train function (our go to method from now on)

11.2.1 Building category mean models by explicitly computing category means

Let us use the *Boston_marathon* data frame to build a model for *minutes* with *sex* as the explanatory variable.

```
Boston_marathon |>
  summarize(avg_minutes = mean(minutes), .by = sex)

# A tibble: 2 x 2
  sex    avg_minutes
  <chr>        <dbl>
```

```
1 male          142.
2 female        150.
```

In the above code, if we had not used ".by = sex", we would have got the overall mean. But using ".by = sex" computes a separate mean for each value of *sex*.

We now have the mean of *minutes* for each sex and can build the model function as:

$$\text{minutes} = \begin{cases} 142 & \text{if sex} = \text{"male"} \\ 150 & \text{if sex} = \text{"female"} \end{cases} \tag{11.6}$$

As before, this is needlessly cumbersome. We can do this using the *model_train* function like before.

```
Boston_marathon |>
  model_train(minutes ~ sex) |>
  coef()
```

```
(Intercept)      sexmale
     149.87        -8.33
```

The output looks a bit different now. We have the *intercept* as before. But we have a single coefficient named *sexmale*.

When we have categorical explanatory variables, the *model_train* function reports coefficients a bit differently than what we did earlier.

Our explanatory variable has two possible values, *female* and *male*. The intercept represents the model value for one of these – typically, the alphabetically lowest one – in this case *female* as *f* comes alphabetically before *m*.

So the coefficient 149.86, which we wil round to 10 is the model value for *female* as our earlier computations also showed. This is the base value. The coefficient for *sexmale* shows the coefficient value for *male* relative to that for the base. This means that the coefficient for *male* is *less* (because it is negative) than that for female by 8.33. The coefficient for *male* is approximately 142.

So we can construct the model function as:

$$\text{minutes} = \begin{cases} 142 & \text{if sex} = \text{"female"} \\ 142 & 8 \text{ if sex} = \text{"male"} \end{cases} \tag{11.7}$$

Equation 11.7 is effectively the same as Equation 11.6, but written slightly differently.

You should read it as, if *sex* is "female" then the model value is 150. If *sex* is "male" then the model value is 8 minutes lower (negative sign).

Does this make sense? It does, because we know that males run slightly faster than females and so their finishing time will be lower.

> Quick Check
>
> What would be the use case scenario for this model?

> Suggested answer
>
> If the race organizers are planning to do something based on the typical race completion time and can do something different for male and female participants, then the average finishing times for each gender would be a good model.

Let us consider one more example using the *acct_type_balance* data frame with *balance* as the outcome variable and *bank_account_type* as the explanatory variable.

```
acct_type_balance |>
  model_train(balance ~ bank_account_type) |>
  coef()
```

```
      (Intercept) bank_account_typeSavings
             4901                      1016
```

The output shows the intercept as 4901. Our explanatory variable *bank_account_type* has two possible values *Savings* and *Checking*. From the previous example, we know that the *coef()* function will treat one of these as the base.

> Quick Check
>
> WHich account tyoe will it treat as the base? *Checking* or *Savings*?

> Suggested answer
>
> *Checking** because “C” is alphabetically before “S”.

Checking is the base and its average is reported as the *intercept* and the model value for *Savings* is reported relative to the model value for *Checking*.

The model function therefore becomes:

$$\text{balance} = \begin{cases} 4901 & \text{if bank\_account\_type} = \text{"Checking"} \\ 4901 & +1016 \text{ if bank\_account\_type} = \text{"Savings"} \end{cases} \tag{11.8}$$

The model says that savings accounts generally have a higher account balance than checking accounts. Makes sense.

Steps to build the model function for models with one categorical explanatory variable

1. Determine the tilde expression for your model (like balance ~ bank_account_type, or some such)
2. Pipe the data frame to the *model_train* function and pass the tilde expression as well
3. Pipe the output of the *model_train* function to the *coef()* function
4. Find which category has been used as the base and note the intercept
5. Note the coefficients for each of the other categories
6. Construct the model function by substituting the name of the outcome variable, and the value of the intercept in the following equation.
7. Then add conditions for each additional category suitably substituting the appropriate category names.

$$\text{outcome} = \begin{cases} intercept & \text{if explanatory variable} = \text{base category} \\ intercept & + \text{coef1 if explanatory variable} = \text{"category 1"} \\ intercept & + \text{coef2 if explanatory variable} = \text{"category 2"} \\ intercept & + \text{coef3 if explanatory variable} = \text{"category 3"} \end{cases}$$

11.3 Finding the model function for models with a single numerical explanatory variable

Unlike the two mean models, we have no direct computation approach. We will just use the *model_train* function.

We will build a model using the *mpg* data frame with the highway mileage (variable *hwy*) as the outcome variable and the engine displacement (*displ*) as the explanatory variable.

```
mpg |>
  model_train(hwy ~ displ) |>
  coef()
```

```
(Intercept)       displ
      35.70       -3.53
```

As before, we substitute the "intercept* for *a* and in the case of a numerical explanatory variable, we substitute the other coefficient for *b*.

Equation 11.9 shows the model function.

$$\text{hwy} = 36\ \ 3.5\,\text{displ} \tag{11.9}$$

Let us see another example. We use the *Hill_racing* data frame to build a model with *time* (finishing time measured in seconds) as the outcome variable, and)*distance* (measured in km) as the explanatory variable.

```
Hill_racing |>
  model_train(time ~ distance) |>
  coef()
```

```
(Intercept)    distance
       -211         381
```

We see that the *intercept* is -211. The coefficient for *distance* is 381.

Let us build the model function before talking about the crazy-seeming negative *intercept*.

$$\text{time} = 211 + 381\,\text{displ} \tag{11.10}$$

Does it make sense that the coefficient for *distance* is positive? Per Equation 11.10, as *distance* increases, the estimated *time* will increase. This makes sense.

What about the *intercept* of -211?

From Equation 11.10, we see that if there is a hypothetical race with distance of 0 km, then the model says that people will finish the race 211 seconds before the race starts!

Obviously that makes no sense as time works in the normal world. SO what is going on?

Models are approximate and we can only expect reasonable results within the range of data on which they were built. We need to be wary of interpreting models very outside of their scope. In our dataset, the shortest race is 1.1 km and the average is 10.7 km.

For the 1.,1 km race, the model predicts a time of 208 seconds or around 3.5 minutes, which seems reasonable – given that these are "hill" races.

Quick Check

What would be a scenario where this model can help?

Suggested answer

The organizers are getting ready to conduct the races again. Now, they want to give an estimated completion time (one single number) for each race. They can use this model to find this time based on the distance of each race.

Steps to build the model function for models with one numerical explanatory variable

1. Determine the tilde expression for your model (like hwy ~ displ, or some such)
2. Pipe the data frame to the *model_train* function and pass the tilde expression as well

3. Pipe the output of the *model_train* function to the *coef()* function
4. Note the *intercept* and the *coefficient* for the explanatory variable (expl_coeff)
5. Construct the model function by substituting the name of the outcome variable, and the value of the intercept in the following equation.
6. Then add conditions for each additional category suitably substituting the appropriate category names.

$$\text{outcome} = \text{intercept} + \text{expl\_coeff explanatory variable}$$

12 Amanda Strikes Again

In prior chapters we have looked at three different situations in the specific context of Amanda's kiosks. Table 12.1 shows the scenarios and the models we found for each.

Table 12.1: Model types we have looked at so far for the kiosk scenarios

Response variable	Explanatory var	Best model
Num customers, one for both kiosks	None	Mean customers
Num customers, one per kiosk	Kiosk location	Mean for each kiosk
Number of customers	Temperature	Least squares line

Table 12.2 summarizes the situations in more generic terms than the specific kiosk context and the best model for each. (Recall that in this book we consider only numerical response variables.)

Table 12.2: Model types we have looked at so far

Response variable type	Explanatory variable type	Best model
Numerical	None	Mean
Numerical	Categorical	Category means
Numerical	Numerical	Least squares line

In our running story, this is the fourth meeting of our interns with Amanda. Amanda owns a business that operates kiosks in a city at two locations.

"Team, our beach kiosk performed really well last month. Using the temperature to estimate the number of customers really helped us stay efficient. We hit our lowest food waste thus far, and run out of stock by a lot either, at least compared to past months. Your models are helping."

The interns feel very happy to hear this.

Amanda continues. "I now have more data and have reason to believe that the number of customers who turn up at the mall kiosk also seems to be related to the temperature, but not quite in the same way as the beach kiosk. I am not able to put my finger on it, but something tells me that we can do better with a model that considers both temperature and the kiosk location."

She then shared the data frame *kiosk_beach_mall_temp* with the interns.

She reiterates her earlier constraint:

“Keep it simple. I need something I can explain in one sentence.”

She emails the team the data and heads out, wondering if the team will pull off something good once more. While exuding confidence externally, internally she has a nagging doubt that she might be stretching this young team of undergrads too much.

Her confidence energizes the team and they all think through this in silebnce for a few minutes.

Suzie is the first to speak up. “When we gave one number per kiosk, our model looked like this.” and she wrote out Equation 12.1

$$\text{num customers} = \begin{cases} 101 & \text{if kiosk} = \text{"Mall"} \\ 146 & \text{if kiosk} = \text{"Beach"} \end{cases} \tag{12.1}$$

When we predicted the number of customers based on the temperature for the beach kiosk alone our model became this” and she wrote out Equation 12.2. “How do we continue from here?”

$$\text{customers} = 22.06 + 1.42\,\text{temperature} \tag{12.2}$$

Everyone sat back, looking intently at the two simple equations.

Then David spoke up. “Can we think of some combination of the two? In the case of the temperature model, we had a general model like this in which we found the best values for *a* and *b*.” and he writes out Equation 12.3.

$$\text{customers} = a + b\,\text{temperature} \tag{12.3}$$

David then continues: “Why not just add another variable *c* to this?” And he wrote out Equation 12.4

$$\text{customers} = a + b\,\text{temperature} + c\,\text{kiosk\_location} \tag{12.4}$$

“Now we just have to find the *best* values for *a*, *b*, and *c*. That’s all!”

“Terrific!” Angela says. “Looks like we are done!” People nod in agreement.

“Not so fast Angela.” Igor says, and everyone seems mildly irritated as if he is just being picky.

But, Igor continues “Kiosk location is either beach or mall. How do we multiply that by the value of *c*?”

They fall into a deep silence again. They felt that Amanda sure gave them a tough nut to crack this time.

Suzie once again tries to connect the old model to the new one. She says “Igor is right in that we cannot multiply *c* by the kiosk location. But can we not something like we did earlier with if the kiosk is beach then something, otherwise something sort of approach?” and she points to Equation 12.1.

Long silence once again. Then Igor speaks out. “Perhaps our model should look something like this.”

$$\text{customers} = a + b\,\text{temperature} + \begin{cases} c_{\text{Mall}} & \text{if kiosk} = \text{"Mall"} \\ c_{\text{Beach}} & \text{if kiosk} = \text{"Beach"} \end{cases}$$

Time was running out. Amanda would be back any time now.

David spoke up finally “Peter, last time you wrote some code to get the model. Why not just try that again? Perhaps it can already do what we want.”

Just then Amanda comes back. The team was quite embarrassed that they did not have a solution for her. However, she senses the mood in the room and says “It looks like you have worked hard, but have not hit upon any solution yet.”

Peter replies “We have made some progress, but are not quite there yet. Can you give us a little more time?” He then laid out exactly where they stood.

Amanda senses their disappointment and immediately says: “You seem to be on a good path here. I have a sense that you can pull this off. Let us meet in a few days. Good luck!”

Over the next day, some read up on statistics. Others were all over chatbots and Igor consulted his professor.

They met briefly in the university cafeteria over lunch and agreed to meet as a team later that day to see where they stood.

At the meeting, Peter started “We were on the right track.”

Igor agreed. I spoke to Dr. Berlisky from the statistics department and he said that this problem actually involves a somewhat more advanced concept called *interactions effects*, However, he suggested a simple idea. Just separate the data for thw two kiosks and build separate models for each! I wonder why we did not think of that. It might be more complex than Amanda would want. But from what the prof said, I think we have no choice.”

The team agreed.

Peter wrote up the code once again to first visualize both models.

```
kiosk_beach_mall_temp |>
  point_plot(customers ~ temperature + kiosk,
             annot = "model",
             interval = "none")
```

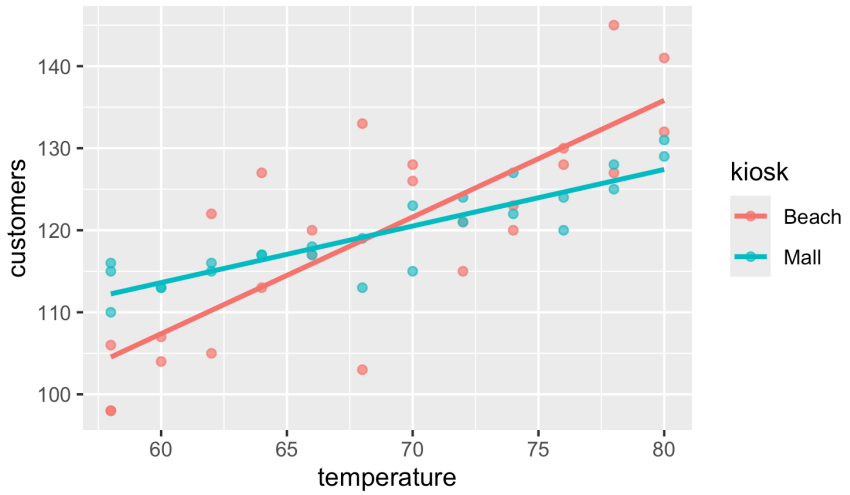

Figure 12.1: Separate least-square lines for the two kiosks showing that the beach kiosk is more sensitive to temperature

David says “Amanda’s instinct that the two kiosks relate duqite differently seems correct. The beach kiosk seems much more sensitive to temperature.”

Steve says “Yeah. At lower temperatures it has higher sales, but when it gets warmer, more people seem to want to hist the beach. Makes perfect sense to me!”

The team feels happy that they can explain the common sense behind the model.

Peter then writes the code to get separate models for the two kiosks. First for the Mall kiosk.

```
kiosk_beach_mall_temp |>
  filter(kiosk == "Mall") |>
  model_train(customers ~ temperature) |>
  coef()
```

From the output, they reconstructed the model for the Mall kiosk as:

$$\text{customers} = 72 + 0.7\,\text{temperature} \tag{12.5}$$

```
kiosk_beach_mall_temp |>
  filter(kiosk == "Beach") |>
  model_train(customers ~ temperature) |>
  coef()
```

From the output, they reconstructed the model for the Mall kiosk as:

$$\text{customers} = 22 + 1.4\,\text{temperature} \tag{12.6}$$

From Figure 12.1, it seemed clear that the two kiosks were quite different in how they related tgo temperature and therefore two separate models seemed to make sense.

Later that week, they met with Amanda again, a little fearful that they did not stick to her "simple" requirement and were giving two different models. They told her honestly that there was a more advanced approach that would fit both into a single model, but were not sufficiently knowledgeable to present that approach, although the results would be the same as with two separate models.

Fortunately, Amanda was happy with their model. She says "Although I had asked for something simple, I guess when we want the model to be more and more specific, we have to tolerate some complexity after all. I am fine with this."

The team resolved to learn more about the *interaction effects* that Dr. Berlinsky had spoken about and present that to Amanda when possible.

Part V

Explanatory Power of a Model

Module 6: Explanatory Power of a Model

We turn our attention now to measuring the quality of a model based on the notion of a model's ability to *explain* the variability in the response variable.

Learning Goals

- a
- b
- c

Structure of This Module

- What does it mean for a model to explain variability in the response variable?
- Computing the model values and residuals
- the variance equation
- R-squared

Part VI

Populations, Samples and Inference

Module 7: Population, Samples and Inference

At this point, we have learned about models in general, and linear regression models in particular. In addition, we can also now use R to build linear models and to construct the model function based on the output of the R code. We have also looked at the quality of a model based on its R-squared.

Until now we have looked at a mdoel as a mathematical function into which we can plug in the value of explanatory variable(s) and get an estimated value for the response variable.

We now transition from mathematical functions to *statistical models*. We first define the terms between *population* and *sample* and reveal that much of statistics deals with *inference*, that is, learning something about the population based on just a sample.

Learning Goals

- a
- b
- c

Structure of This Module

- Population and sample
- Statistc

13 Population vs Sample

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

14 Sampling Variability

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part VII

Probability

Module 8: Probability as Variation

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

15 Randomness

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

16 Variability

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

17 Distributions

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part VIII

Sampling Variation and Estimators

Module 9: Sampling Variation and Estimators

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

18 SE Estimators

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

19 Sampling Distributions Slopes

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part IX

Confidence Intervals and Bands

Module 10: Confidence Intervals and Bands

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

20 CI Slope

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

21 CI Bands

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part X

Signal, Noise, and R-Squared

Module 11: Signal and Noise

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

22 Variance Decomposition

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

23 R-Squared

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part XI

Hypothesis Testing via Regression

Module 12: Hypothesis Testing

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

24 t-tests for Regression

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

25 Interpretation

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part XII

Business Applications of Regression

Module 13: Business Applications

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

26 Marketing

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

27 Finance

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

28 Human Resources

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

29 Operations

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part XIII

Projects and Datasets

Module 14: Projects

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

Structure of This Module

- Data frames
- Point plots
- Relationships between variables

30 Project Guidelines

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

31 Datasets

The mean of a numerical variable is the “best guess” of its value when no other information is available — in the specific sense that it minimizes the sum of squared errors.

library(ggplot2) mean(mtcars$mpg)

Part XIV

Appendices

32 Appendices

This module introduces the basic building blocks of statistical thinking: data frames, variables, and point plots. We begin with visualization because patterns reveal what models will later formalize.

32.1 Learning Goals

- Understand data frames, variables, and instances
- Create and interpret point plots
- Recognize patterns: direction, form, strength

32.2 Structure of This Module

- Data frames
- Point plots
- Relationships between variables