

Lessons in Statistical Thinking

Intro Stats for the 21st Century

Daniel T. Kaplan

2024-03-23

Table of contents

1 Data frames	6
1.1 Types of variables	9
1.2 The codebook	10
1.3 Accessing data frames	11
1.4 Computing with data frames	11
2 Data graphics	17
2.1 Point plot	18
2.2 Response and explanatory variables	23
2.3 Categorical variables and jittering	24
2.4 Color and faceting	26
2.5 Graphical annotations	28
3 Variation and density, graphically	30
3.1 The “shape” of variation	30
3.2 Some simple shapes	33
4 Annotating point plots with a model	36
4.1 Simple models	36
4.2 Independence	40
4.3 Multiple explanatory variables	41
5 Data wrangling	45
5.1 Basic data-wrangling operations	45
5.2 Compound wrangling statements	51
5.3 Actions and adverbs; functions and arguments . .	52

5.4 Pivoting (optional)	54
6 Computing with functions and arguments	64
6.1 Chain of operations	64
6.2 What's in a pipe?	67
6.3 Pipes connect to functions	67
6.4 Arguments (inside the parentheses)	68
6.5 Variable names in arguments	71
6.6 Styling with space	72
6.7 Displaying tables	73
7 Databases	76
7.1 <i>E pluribus unum</i>	77
7.2 Join: putting tables together	78
8 Statistical thinking & variation	82
8.1 Measuring variation	84
9 Accounting for variation	87
9.1 Numerical explanatory variables	90
9.2 Multiple explanatory variables	93
9.3 Comparing models with R ²	95
10 Model patterns	97
10.1 Data and patterns: a painterly metaphor	97
10.2 The model specification	98
10.3 “Shapes” of models	99
10.3.1 One explanatory variable	99
10.3.2 Two explanatory variables	101
11 Model functions	108
11.1 Basics of mathematical functions	108
11.2 Statistical models	110
11.3 Training a model	112
11.4 Model functions with multiple explanatory variables	115
11.5 Case study: Get out the vote!	117
11.6 Tradition and “correlation”	121
12 Adjustment	122
12.1 Groupwise adjustment	122
12.2 Adjustment with <i>per</i>	123

12.3 Adjustment by modeling	126
13 Signal and noise	129
13.1 Partitioning data into signal and noise	131
13.2 Model values as the signal	132
13.3 R ² (R-squared)	135
14 Simulation	138
14.1 Pure noise	138
14.2 Simulations with a signal	142
14.3 Example: Heritability of height	143
15 Models for noise	148
15.1 Waiting time	149
15.2 Blood cell counts	151
15.3 Adding things up	153
15.4 Other named distributions	154
15.5 Relative probability functions	155
16 Estimation and likelihood	158
16.1 How likely?	159
16.2 Comparing different hypotheses using likelihood .	161
16.3 Likelihood functions (optional)	164
16.4 Data narrows the likelihood function (optional) .	166
17 R-squared and covariates	169
17.1 Fraction of variance explained	169
17.2 R ² and categorical explanatory variables	172
17.3 Degrees of freedom	174
18 Predictions	177
18.1 Statistical predictions	177
18.2 Prediction via statistical modeling	180
18.3 The prediction interval	185
18.4 Form of a statistical prediction: Categorical outcome	186
19 Sampling and sampling variation	191
19.1 Why sample?	192
19.2 Sampling bias	193
19.3 Sampling variation	196
19.4 Sampling trials	197

19.5 SE depends on the sample size	200
20 Confidence intervals	203
20.1 Formats for confidence intervals	203
20.2 Precision versus accuracy	206
20.3 The confidence <i>level</i>	208
20.4 Calculating confidence intervals (optional)	211
20.4.1 Subsampling	212
20.4.2 Bootstrapping	215
20.5 Decision-making with confidence intervals	217
21 Measuring and accumulating risk	220
21.1 Risk vocabulary	220
21.2 Modeling risk	223
21.3 Logistic regression	225
21.4 Risk	227
21.5 Probability, odds, and log odds	230
22 Effect size	233
22.1 Effect size: Input to output	234
22.1.1 Effect size for <i>quantitative</i> explanatory variable	235
22.1.2 Effect size for <i>categorical</i> explanatory variable	238
22.2 Model coefficients and effect size	239
22.3 Interactions	239
23 Directed acyclic graphs	243
23.1 Influence diagrams	243
23.2 Nodes	246
24 Causal influence and DAGs	252
24.1 Pathways	254
24.2 Blocking correlating and non-correlating pathways using covariates	257
24.3 DAGs and data	261
25 Confounding	267
25.1 Block that path!	273
25.2 Don't ignore covariates!	278

26 Experiment and random assignment	280
26.1 Replication	281
26.2 Example: Replicated bed net trials	281
26.3 Control	282
26.4 Example: Testing the Salk polio vaccine	284
26.5 Random assignment	285
27 Hypothetical thinking	288
27.1 Where do hypotheses come from?	290
27.2 Hypotheses and deduction	292
27.3 Planets and hypotheses	293
28 Competing hypotheses with Bayesian reasoning	296
28.1 Bayesian thinking	301
28.2 Bayes with multiple hypotheses	304
28.3 Accumulating evidence	305
29 Hypothesis testing	306
29.1 The Null hypothesis	307
29.2 Formats for NHT results	311
29.3 Calculating “significance”	313
29.4 The p-value	316
29.5 Power and the alternative hypothesis	318
29.6 False discovery	326
29.7 Hypothesis testing interpreted by a Bayesian . .	326

1 Data frames

The origin of recorded history is, literally, data. Five-thousand years ago, in Mesopotamia, the climate was changing. Retreating sources of irrigation water called for an organized and coordinated response, beyond the scope of isolated clans of farmers. To provide this response, a new social structure – government – was established and grew. Taxes were owed and paid, each transaction recorded. Food grain had to be measured and stored, livestock counted, trades and shipments memorialized.

Writing emerged as the technological innovation to keep track of all this. We know this today because memoranda were incised by stylus on soft clay tablets and baked into permanence. When the records were no longer needed, they were recycled as building materials for the growing settlements and cities. Archaeologists started uncovering these tablets more than 100 years ago, spending decades to decipher the meaning of the stylus marks in clay.

The writing and record-keeping technology developed over time: knots in string, wax tablets, papyrus, vellum, paper, and computer memory. Making sense of the records has always required *literacy*, deciphering marks according to the system and language used to represent the writer’s intent. Today, in many societies, the vast majority of people have been taught to read and write their native language according to the accepted conventions.

Conventions of record keeping diverge from those of everyday language. For instance, financial transaction records must be guarded against error and fraud. Starting in the thirteenth century, financial accountants adopted a practice—double-entry bookkeeping—that has no counterpart in everyday language.

Modern conventions make working with data more accessible and more reliable. Of primary interest to us in these *Lessons* is the organization provided by a “**data frame**,” a structure for holding data as exemplified in Figure 1.

“[Double-entry bookkeeping](#),” records *twice* in two different places, in the form of a credit to an account and a debit from another account.

Column

Row

Column

4 Variables

5 Specimens

mother	father	sex	height
65.0	65	F	59.0
67.0	73	M	67.0
68.5	69	M	70.0
69.0	70	M	68.7
63.0	70	F	62.0

Figure 1: A data frame organizes observed facts into rows and columns. Each column is a variable. Each row is a specimen. Here, there are four variables and five specimens. — The display in Figure 1 shows a small part of a larger data frame holding observations collected by statistician Francis Galton in the 1880s. I will use this data frame repeatedly across these lessons because of the outsized historical role the data played in the development of statistical methodology. The context for the data collection was Galton's attempt to quantify the heritability of biological traits. The particular trait of interest to Galton (probably because it is easily measured) is human stature. Galton recorded the heights of full-grown children and their parents.

The row-and-column organization of a data frame is reminiscent of a spreadsheet. However, data frames have additional organizational requirements that typical spreadsheet software does not enforce. The term “**tidy data**” emphasizes that these requirements are being met.

1. Each variable must consist of the same kind of individual entries. For example, the `mother` variable consists of numbers: a quantity. In this case, the quantity is the mother’s height in inches. It would not be legitimate for an entry in `mother` to be a word or to be a height in meters or something else entirely, for instance, a blood pressure.
2. Each row represents an individual real-world entity. For the data frame shown in Figure 1, each row corresponds to an individual, fully-grown child. We use the term “**unit of observation**” to refer to the *kind of entity* represented in each row. All rows in a data frame must be the same kind of unit of observation. It would not be legitimate for some rows to individual people while others refer to something different such as a house or family or country. If you wanted to record data on families, you would need to create a new data frame where the unit of observation is a family.

We use the word “**specimen**” to refer to an individual instance of the unit of observation. A data frame is a collection of specimens. Each row represents a unique specimen.

The unit of observation in Figure 1 is a full-grown child. The fifth row in that data frame refers to a unique young woman in London in the 1880s (whose name is lost to history). By using the word “specimen” to refer to this woman, we do not mean to dehumanize her. However, we need a phrase that can be applied to a single row of any data frame, whatever its unit of observation might be: a shipping container, a blood sample, a day of ticket sales, and so on.

The collection of specimens comprised by a data frame is often a “**sample**” from a larger group of the units of observation. Galton did not measure the height of every fully-grown child in London, England, the UK, or the World. He collected a

sample from London families. Sometimes, a data frame includes every possible instance of the unit of observation. For example, a library catalog lists comprehensively the books in a library. Such a comprehensive collection is called a “**census**.”

i Example: New-born babies

The US Centers for Disease Control (CDC) publishes a “public use file” each year, a data frame where the unit of observation is an infant born in the US. (The many variables include the baby’s weight and sex, the mother’s age, and the number of prenatal care visits during the pregnancy.) The published file for 2022 contains 3,699,040 rows; that is the number of (known) births in 2022. As such, the CDC data constitutes a **census** rather than a **sample**.

1.1 Types of variables

Each column of a data frame is a variable. The word “variable” is appropriate because the entries within a variable **vary** one from one row to another. Other words with the same root include “variation,” “variety,” and even “diversity.”

Data-frame variables come in two fundamental types:

1. **Quantitative** variables record an “amount” of something. These might just as well be called “numerical” variables.
2. **Categorical** variables typically consist of letters. For instance, the `sex` variable in Figure 1 contains entries that are either `F` or `M`. In most of the data we work with in these *Lessons*, there is a fixed set of entry values called the **levels** of the categorical variable. The levels of `sex` are `F` and `M`.

The distinction between quantitative and categorical variables is fundamental to statistical work. You should be able to discern whether a variable is categorical or quantitative from a glance at a data frame.

We are not doing full justice to the variety of possible variable types by focusing on just two type: quantitative and categorical. You should be aware that there are other kinds, for example, photographs or dates.

i Example (cont.): The CDC births data frame

Among the many variables in the CDC public use file of births are `place` and `diabetes_gest`, which record the place of birth and whether the mother developed gestational diabetes.

The `place` variable is categorical, with these levels:

- “hospital”
- “home (intended)”
- “home (unintended)”
- “freestanding”
- “other”

The `diabetes_gest` variable has two levels: **N** or **Y**.

1.2 The codebook

How are you to know for any given data frame what constitutes the unit of observation or what each variable is about? This information, sometimes called **metadata**, is stored outside the data frame. Often, the metadata is contained in a separate documentation file called a “**codebook**.”

To start, the codebook should make clear what is the unit of observation for the data frame. For instance, we described the unit of observation for the data frame shown in Figure 1 as a fully grown child. This detail is important. For instance, each such child—each specimen—can appear only once in the data frame. In contrast, the same `mother` and `father` might appear for multiple specimens, namely, the siblings of the child.

In the CDC data frame, the unit of observation is a newborn baby. If a birth resulted in twins, each of the two babies will have its own row. In contrast, imagine a data frame for the birth mothers or another for prenatal care visits. Each mother could appear only once in the birth-mothers frame, but the same mother can appear multiple times in the prenatal care data frame.

For quantitative variables, the relevant metadata includes what the number refers to (e.g., mother's height or baby's weight) and the physical units of that quantity (e.g., inches for height or grams for weight).

For categorical variables, the metadata should describe the meaning of each level in as much detail as necessary.

i Example (cont.): CDC births codebook

The codebook for the CDC data is a PDF document entitled "User Guide to the 2022 Natality Public Use File." You can access it on the [CDC website](#).

1.3 Accessing data frames

Most statistics software, including R, makes it easy to access data frames stored as files in any of a variety of formats. (For examples, see Exercise 1.18.)

Almost all the data frames used as examples or exercises in these *Lessons* are stored in files provided by R software “**packages**” such as `{LSTbook}` or `{mosaicData}`. The data frame itself is easily accessed by a simple name, e.g., `Galton`. The location of the data frame is specified by the package name as a prefix followed by a pair of colons, e.g. `mosaicData::Galton`. A convenient feature of this system is the easy access to documentation by giving a command consisting of a question mark followed by the *package-name::data-frame-name*, e.g.

```
?mosaicData::Galton
```

1.4 Computing with data frames

Lessons 2, 3 & 4 cover how to make informative graphics that give an overview of the contents in a data frame. Lesson 5 introduces commands for manipulating the contents of a data frame to put them in a more useful form for the data graphics or data summary task at hand.

This Lesson shows you how to access data frames and their documentation and how to perform simple tasks such as listing the variable names or glimpsing a few rows of a data frame.

There are many software systems for working with data frames. Commonly available spreadsheet software, while suited to some data-entry and data-summarizing tasks, is surprisingly limited when it comes to statistical thinking. The system we will use, RStudio, is one of a handful used by data science professionals. It's available free both as an online, browser-based platform and for installation on a laptop computer or computer server.

Much of the statistical work you do in RStudio consists of writing commands in the R language. The word "language" is off-putting to many people, associating it as they do with natural languages such as Chinese or Spanish, mastery of which takes time and much work. Fortunately, you do not have to learn the R language; you need only a couple dozen R expressions to work through all these *Lessons*.

We continue here under the assumption that you have already been shown how to install and access RStudio by an instructor or other mentor. That person will have arranged to install some additional software written for these *Lessons*, particularly the `{LSTbook}` package.

Each time you open RStudio, load the `{LSTbook}` package using this **command** at the R prompt in the "console" tab.

```
library(LSTbook)
```

i Starting out with R via posit.cloud

Note: Otherwise ...

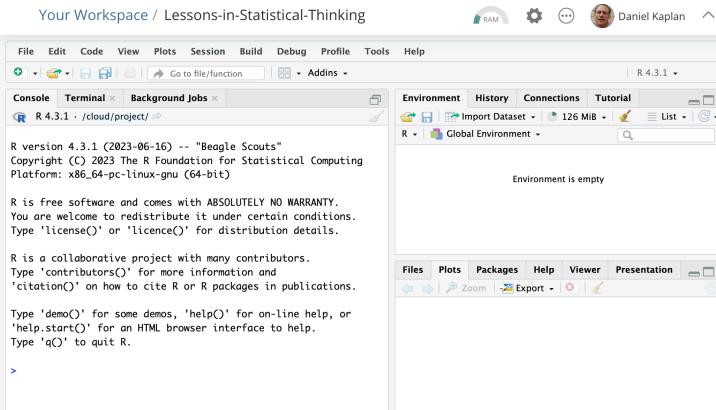
`posit.cloud` is a "freemium" web service. The word "freemium" signals that you can use it for free, up to a point. Fortunately, that point will suffice for you to follow all of these *Lessons*.

1. In your browser, follow [this link](#). This will take you to `posit.cloud` and, after asking you to login via Google or to set up an account, will bring you to a page that will look much like the following. (It may

If you are on your own, the instructions below provide a quick way to get started with minimal effort.

If you are a student using these Lessons as part of a class, check with your instructor who may already have set up a way for you to access RStudio.

take a few minutes.)



2. On the left half of the window, there are three “tabs” labelled “Console,” “Terminal,” and “Background Jobs.” You will be working in the “Console” tab. Click in that tab and you will see a flashing | cursor after the > sign.
3. Give this command, exactly as written, and press return: ::: {.cell}

```
library(LSTbook)
```

Now you are ready to go. :::

All of your work with R will consist of giving commands at the > prompt and pressing return. Possibly the simplest of all commands is merely the name of a data frame. For instance, the `{LSTbook}` package provides, among many others, a data frame named AAUP. Try this as a command:

```
AAUP
```

The result of such a command will be a print-out of the first several rows and columns of the data frame. Some of the data frames provided by `{LSTbook}` have a couple of dozen rows, others have tens of thousands. Printing out the first few rows of

a data frame is useful since it shows the variable names and you can see whether each variable is quantitative or categorical.

To see the codebook for a data frame, simply precede the name with the ? character, for instance:

```
?Births2022
```

RStudio arranges for the codebook to be displayed in the “Help” tab. This allows you to scroll through the documentation, follow web links (if any), and keep the names of the variables displayed in the Help tab while you write commands in the Console tab.

Commands you will use in these *Lessons* will often start with the name of a data frame followed a “**pipeline** symbol |> which is then followed by a description of the action you want to perform. Let’s consider two simple actions:

1. Count the rows in the data frame:

```
AAUP |> nrow()
```

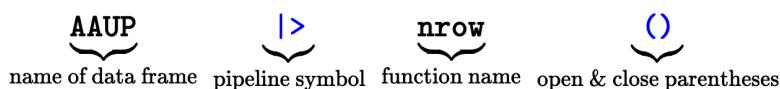
```
[1] 28
```

2. List the names of the variables.

```
AAUP |> names()
```

```
[1] "subject"   "acsal"      "fem"        "unemp"       "nonac"       "nonacsal"    "licensed"
```

These two commands have a similar structure involving four elements.



There are two names in this command: the name of a data frame and a “**function**” name. The function name is how you specify what you want to calculate from the data frame.

There are also two bits of punctuation:

Births2022 [LSTbook] R Documentation
Records on births in the US in 2022
Description
These data come from the Centers for Disease Controls “public use file” recording all 3,699,040 (known) births in the US in 2022. Births2022 is a random sample of size 20,000 from the comprehensive file
Usage
Births2022
Format
A data frame with 20,000 observations on the following 38 variables. The unit of observation is a birth.

- month: 1-12
- dow: Day of week: Sun, Mon, Tues, ...
- place: hospital, home, clinic, etc.
- paternity: is paternity acknowledged. Y, N, and X. X stands for “not applicable” which is shorthand for the mother is married (consequently the

Figure 2: The codebook for the CDC births data frame can be accessed with ?Births2022. When displayed in the RStudio Help tab, you can scroll through the descriptions of all 38 variables.

- the pipeline symbol `|>`, which connects the data frame to the function.
- a pair of open and close parentheses immediately following the function name. Every time you use a function the function name will be followed by parentheses.

i Tables versus data frames

You may notice that the displays of data frames printed in this book are given labels such as Table 1. It is natural to wonder why the word “table” is used sometimes and “data frame” other times.

In these *Lessons* we make the following distinction. A “data frame” stores values in the strict format of rows and columns described previously. Data frames are “machine readable.”

The data scientist working with data frames often seeks to create a **display** intended for human eyes. A “table” is one kind of **display** for humans. Since humans have common sense and have learned many ways to communicate with other humans, a table does not have to follow the restrictions placed on data frames. Tables are not necessarily organized in strict row-column format, can include units for numerical quantities and comments. An example is the table put together by Francis Galton (Figure 3) to organize his measurements of heights.

FAMILY HEIGHTS. from R.E.P (add 6 inches to every entry in the Table)			
Father	Mother	Sons in order of height	Daughters in order of height.
1 18.5	7.0	13.2	9.2, 9.0, 9.0
2 15.5	6.5	13.5, 12.5	5.5, 5.5
3 15.0	about 4.0	11.0	8.0
4 15.0	4.0	10.5, 8.5	7.0, 4.5, 3.0
5 15.0	-1.5	12.0, 9.0, 8.0	6.5, 2.5, 2.5

Figure 3: An excerpt from Francis Galton’s notebook recording the heights of parents and children in London in the 1880s.

We make the distinction between a data frame (for data storage) and a table (for communicating with humans) because many of the operations discussed in later lessons serve the purpose of transforming data frames into human-

facing displays such as graphics (Lesson 2) or tables (Section 6.7.)

Often, a literal display of a data frame may seem inefficient, for instance this view of the `Galton` dataframe which was constructed from Figure 3.

```
Galton
```

Table 1: The records from the table shown in Figure 3 in a data-frame format.

family	father	mother	sex	height	nkids
1	78.5	67	M	73.2	4
1	78.5	67	F	69.2	4
1	78.5	67	F	69	4
1	78.5	67	F	69	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	64	M	71	2
3	75	64	F	68	2

It may seem that the data frame is inefficient, for example repeating the heights of mother and father for all the siblings in a family. But this view of efficiency relates to the use of paper and ink by a table; the computer entity requires a different view of efficiency.

2 Data graphics

The statistical thinker seeks to identify patterns in data, such as possible relationships between variables. Translating a data frame into graphical form—data graphics—is an important tool for revealing or suggesting patterns.

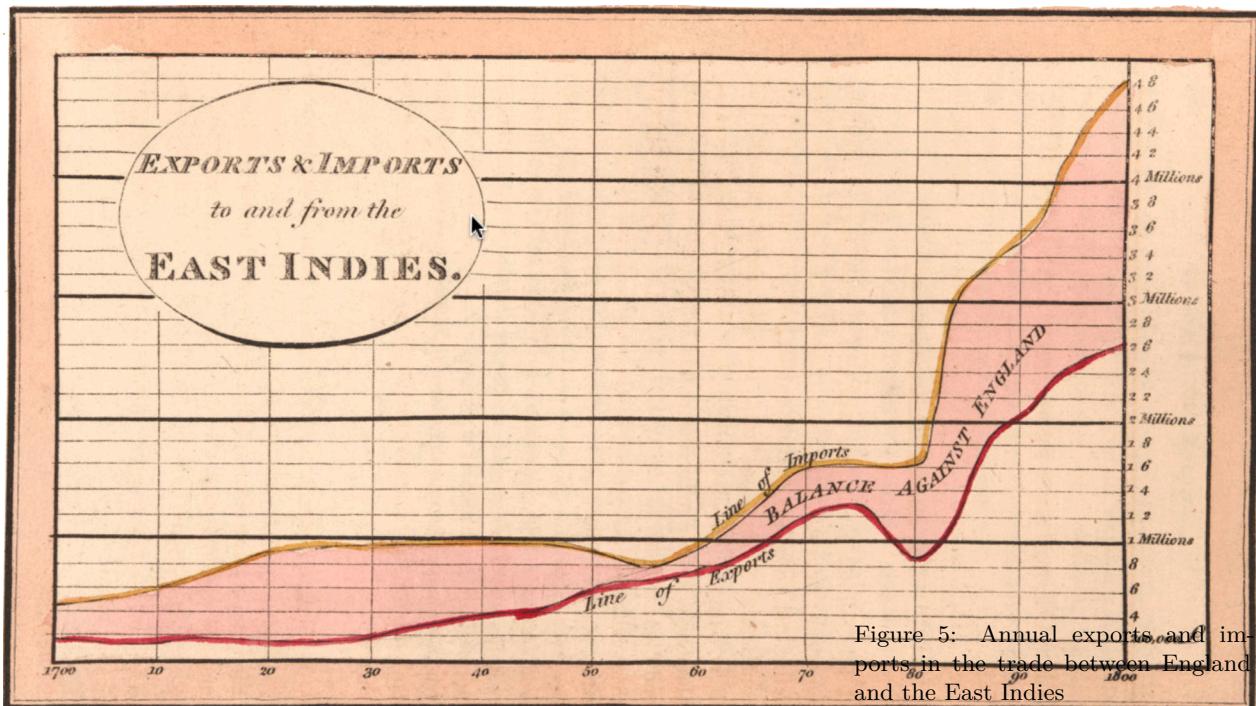


Figure 4: William Playfair's 1801 presentation of year-by-year data on trade between England and the East Indies.
Source: University of Pennsylvania Libraries

Making pictures of data is a relatively modern idea. [William Playfair](#) (1759-1823) is credited as the inventor of novel graphical forms in which data values are presented graphically rather than as numbers or text. To illustrate, consider the data from the 1700s (Table 5) that Playfair turned into a picture.

Playfair's innovation, as in Figure 4, was successful because it was powerful. A pattern that may be obscure in the data frame becomes visually apparent to the human viewer. For example, consider the graphic in Figure 4 displaying data on

Figure 5: Annual exports and imports in the trade between England and the East Indies

Year	Exports	Imports
1700	180	460
1701	170	480
1702	160	490
1703	150	500
1704	145	510
1705	140	525
1706	135	550
1707	125	565
1708	120	580
... and so on to year 1800.		

trade between England and the East Indies in the 1700s. The graphic lets you look up the amount of trade each year, but it also shows patterns, such as the upward *trend* across the decades.

Data graphics can also make it easy to see deviations from trends, for instance, the dip in exports and flattening of imports during 1775-1780.

Students often encounter various types of data graphics as they progress through elementary and high school. Figure 6 shows a few examples commonly found in textbooks. Remarkably, it's rare to encounter such textbook graphic types outside of a statistics course.

Modern data graphic designers are introducing even more variety; their graphics can be captivating, colorful, dynamic, and informative. Some online examples: [how people spend their day](#), [life expectancy](#), [wind patterns \(right now!\)](#), [historical sources of death](#). The graphical types in Figure 6 were all invented long before computers became available to help us work with data.

We won't use such graphical variety in these *Lessons*. Instead, we will use a single basic form of graphic—the “**annotated point plot**”—capable of displaying multiple variables simultaneously and which can combine into one view both the raw data and a summary of the patterns found in the data.

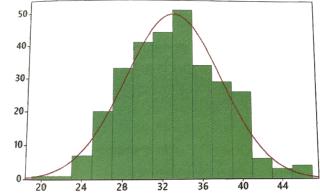
Note to instructors

2.1 Point plot

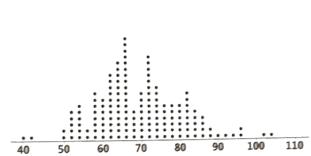
A **point plot** contains a simple mark—a dot—for each row of a data frame. In its most common form, a point plot displays two selected variables from the data frame. One variable is depicted as the vertical coordinate, and the other as the horizontal coordinate.

To illustrate how a point plot relates to the underlying data frame, consider Table 2, where the unit of observation is a city. (The data frame is available in R as `maps::world.cities`.)

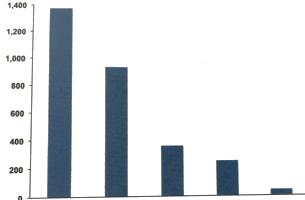
A “point plot” is also known as a “scatter plot.”



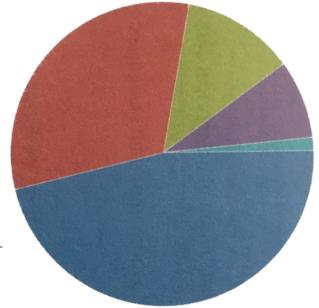
(a) Histogram



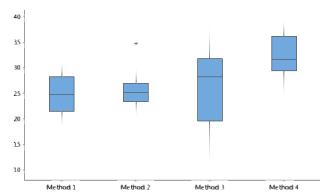
(b) Dot plot



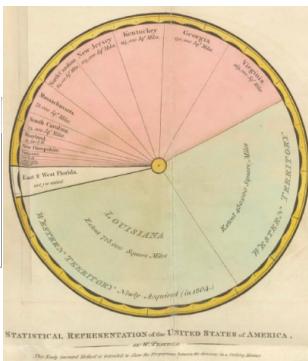
(c) Bar chart



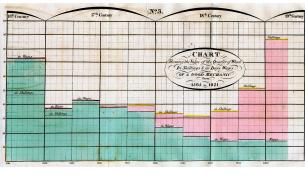
(d) Pie chart



(e) Boxplot



(f) Playfair's pie chart



(g) Playfairs bar chart

1	677
1	8889
2	0000111
2	222222233333
2	444445555555
2	6666667777
2	888888999999
3	000111
3	2223
3	4455
3	666777
3	899999
4	01
4	233
4	
4	6

(h) Stem-and-leaf plot

Figure 6: Some of the graphics styles often featured in statistics textbooks.

Table 2: Basic data on cities in the `maps::world.cities` data frame

name	country.etc	pop	lat	long	capital
Shanghai	China	15017783	31.23	121.47	2
Bombay	India	12883645	18.96	72.82	0
Karachi	Pakistan	11969284	24.86	67.01	0
Buenos Aires	Argentina	11595183	-34.61	-58.37	1
Delhi	India	11215130	28.67	77.21	0
Manila	Philippines	10546511	14.62	120.97	1

... and so on for 10,000 cities

Since `world.cities` contains several variables, many possible *pairs* of variables could be shown in point-plot form. For instance, suppose we choose the `lat` and `long` variables, which specify each city's location in terms of latitude and longitude. Figure 7 shows a point plot of latitude *versus* longitude for world cities. By convention, the word "versus" in the phrase "latitude versus longitude" marks the role of each variable in the point plot: latitude on the vertical axis and longitude on the horizontal axis.

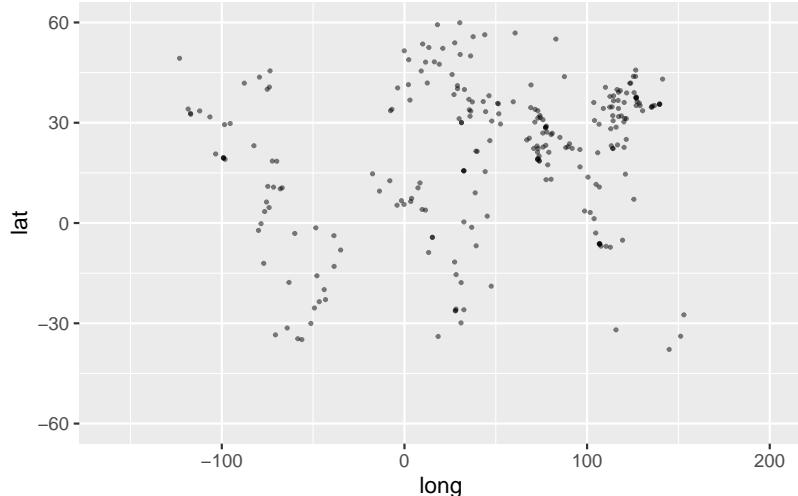


Figure 7: A point plot of the latitude versus longitude of the world's 250 largest population cities.

The dots in Figure 7 hint at some geographical patterns you

Table 3: Some selected variables from the `Anthro_F` data frame.

Wrist	Ankle	Knee	Height	Neck	Biceps	Waist	BFat
18.4	23.5	37.5	1.6637	34.0	32.0	84.3	27.30
13.5	18.0	32.3	1.6002	27.1	23.2	60.2	17.62
18.0	22.5	38.5	1.6510	32.5	28.2	80.1	30.42
19.0	24.5	41.5	1.6637	35.0	34.5	90.0	34.24

learned about in geography class. In general, the purpose of a point plot is to hint at patterns in data.

To show how to construct a point plot, we will work with data on human body shape. The `Anthro_F` data frame records nineteen different measurements of body shape for each of 184 college-aged women. (See Table 3)

In making a point plot of `Anthro_F`, we have to choose *two* variables to display. One variable will determine the vertical position of the dots, the other variable will set the horizontal position. For instance, in Figure 8 we choose `Wrist` for the vertical position and `Ankle` for the horizontal position. In words, the plot is “wrist *versus* ankle,” that is, “vertical *versus* horizontal.” (The codebook for `Anthro_F`—available via the R command `? Anthro_F`—tells us that `Ankle` is measured as the circumference in centimeters, and similarly for `Wrist`.)

```
Anthro_F |> point_plot(Wrist ~ Ankle)
```

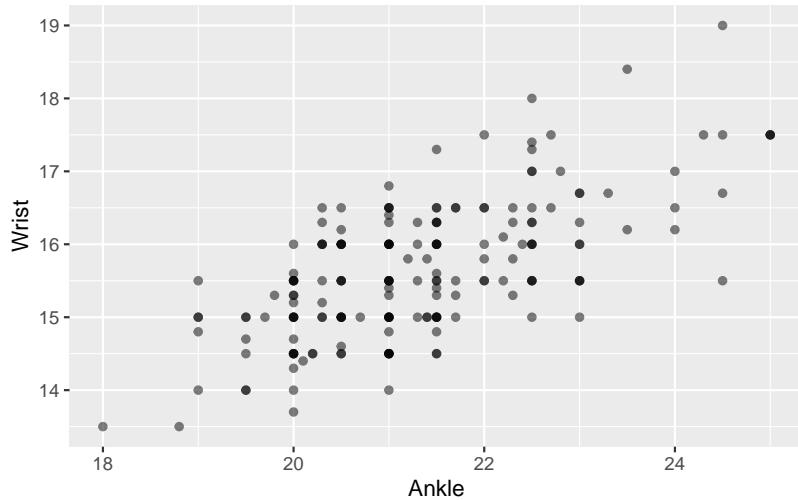


Figure 8: A point plot of wrist versus ankle circumference.

The pattern seen in Figure 8 can be described as an upward-sloping cloud. We will develop more formal descriptions of such clouds in later Lessons. But for now, focus on the R command that generated the point plot.

The point of an R command is to specify what action you want the computer to take. Here, the desired action is to make a point plot based on `Anthro_F` using the two variables `Wrist` and `Ankle`. Look carefully at the command for Figure 8:

```
Anthro_F |> point_plot(Wrist ~ Ankle)
```

The command includes all four of the names involved in the plot:

- The data frame `Anthro_F`
- The action `point_plot`
- The variables involved: `Wrist` and `Ankle`

These names are separated from one another by some punctuation marks:

- `|>`, the “pipe”
- `()`, a pair of parentheses
- `~`, called “tilde”

Don't be daunted by this punctuation, strange though it may seem at first. You will get used to it, since almost all the commands you use in these *Lessons* will have the same punctuation.

Let's annotate the R command that made Figure 8 to identify the different components of the command:



The R command is annotated with colored brackets and labels: `Anthro_F` is labeled "data frame"; the pipe symbol (`|>`) is labeled "pipe"; the function name `pointplot` is labeled "function"; and the argument `Wrist ~ Ankle` is labeled "argument".

Most commands in these *Lessons* start with a data frame named at the start of the command. This is followed by the pipe, which indicates sending the data frame to the next command component. That next component specifies which action to take. By convention, “**Function**” is used rather than “action.” You use differently named functions to carry out different kinds of actions.

The function name is *always* followed by an opening parenthesis. Any details about what action to perform go between the opening and the corresponding closing parentheses. In computer terminology, such details are called “**arguments**.” The detail for the Figure 8 `point_plot` is the choice of the two variables to be used and which one goes on the vertical axis. This detail is written as a “**tilde expression**.” The tilde expression given as the argument to `point_plot()` is `Wrist ~ Ankle`, which can be pronounced as “wrist *versus* ankle” or ‘wrist *tilde* ankle,’ as you prefer.

You will need only a handful of function for these *Lessons*, for instance, `point_plot`, `model_train`, `conf_interval()`, `mutate()`, `summarize()`.

2.2 Response and explanatory variables

Another pronunciation for  is “... as a function of” So, `Wrist ~ Ankle` means “wrist circumference as a function of ankle circumference.” In mathematics, functions are often written using a notation like $f(x)$. In this notation, x is the **input** to

the function `f()`. The word “input” is used in so many different contexts that it’s helpful to use other technical words to highlight the context.

- In **computer notation**, such as `f(x)` or `point_plot(Wrist ~ Knee)`, an expression inside the parentheses is called an **argument**. In `f(x)`, the function is `f()` and the argument is `x`. In `point_plot(Wrist ~ Knee)`, the function is `point_plot()` and the argument is the tilde expression `Wrist ~ Knee`.
- In **statistics**, in the word phrase “wrist circumference as a function of ankle circumference” or, equivalently, the computer expression `Wrist ~ Knee` referring to the `Anthro_F` data frame, we say that `Knee` is an **explanatory variable** and `Wrist` is the **response variable**. In graphics, such as Figure 8, convention dictates that the response variable is shown along the vertical axis and the explanatory is shown along the horizontal axis.

In Figure 8, why did we choose `Ankle` as the explanatory variable and `Wrist` as the response variable for this example? No particular reason. We could equally well have chosen any of the `Anthro_F` variables in either role, depending on our interest. Typically, the statistical thinker will examine several different pairs to gain an understanding of how the various variables are related to one another.

2.3 Categorical variables and jittering

In the previous example, the `point_plot` of `Wrist versus Ankle`, both variables are **quantitative**: the respective joints’ circumference (in cm). `point_plots` are also suited to **categorical** variables. For example, Figure 9 shows a pair of point plots made from the `Penguins` data frame. The unit of observation is an individual penguin. The selected explanatory variable, `species`, is categorical. The response variable, `mass`, is quantitative.

When a categorical variable is used in a plot, the positions on the axis are labelled with the levels of the variable. “Adelie,”

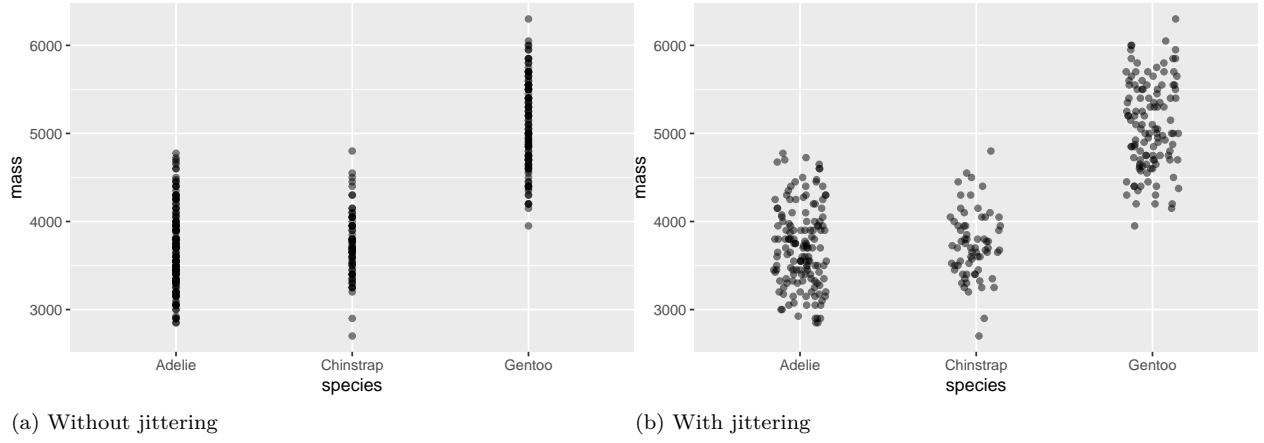


Figure 9: The body mass of individuals of different species of penguins.

“Chinstrap,” and “Gentoo” in the explanatory variable of Figure 9.

When an axis represent a **quantitative** variable, every possible position on that axis refers to a specific value. For instance, the Adelie penguins range between 2850 and 4775 grams. On the vertical axis itself, marks are made at 3000 and 4000 grams, but we know that every position in between those marks corresponds proportionately to a specific numerical value.

In contrast, when an axis represents a **categorical** variable, positions are marked for each level of that variable. But positions in between marks are not referring to fictitious “levels” that do not appear in the data. For instance, the position on the horizontal axis in Figure 9 that’s halfway between Adelie and Chinstrap is *not* reserved for individual penguins whose species is a mixture of Adelie and Chinstrap; every value of a categorical variables is one of the levels, which are discrete. There are no such penguins! (Or, at least, the concept of “species” doesn’t admit of such.)

Using a coordinate axis to represent discrete categories makes common sense, but we are left with the issue of interpreting the space between those categories. In Figure 9 (left) the point plot has been made ignoring the space between categories. Ev-

ery specimen is lined up directly above the corresponding level. The graphical result is that it's hard to identify a single specimen since the dots are plotted on top of one another..

“**Jittering**” is a simple graphical technique that uses the space between the levels to spread out the dots at random, as in Figure 9 (right). This dramatically reduces overlap and facilitates seeing the individual specimens. Recognize, however, that the precise jittered position of a specimen does not carry information about that specimen. All of the specimens in the column of jittered dots above “Adelie” are the same with respect to species, even though they may have different `mass`.

The `point_plot()` function automatically uses jittering when positioning in graphical space the values of categorical variables.

2.4 Color and faceting

Often, there will be more than one explanatory variable of interest. A penguins mass might not just be a matter of species; there are bigger and smaller individuals within any species. Perhaps, for instance, the body *shape*—not just size—is different for the different species. One way to investigate this possibility is to display body `mass` as explained by *both* species and, say, `bill_length`.

To specify that there are two explanatory variables, place their both their names on the right-hand side of the tilde expression, separating the names with a `+` or a `*`. Figure 10(a) shows a point plot made with two explanatory variables.

Figure 10(b) involves three variables. Consequently each dot has three different graphical attributes:

- position in space along the vertical axis. This is denoted as `y`.
- position in space along the horizontal axis. This is denoted as `x`.
- color, denoted, naturally enough, as `color`.

```
Penguins |> point_plot(mass ~ bill_length + species)
Penguins |> point_plot(mass ~ bill_length + species + sex)
```

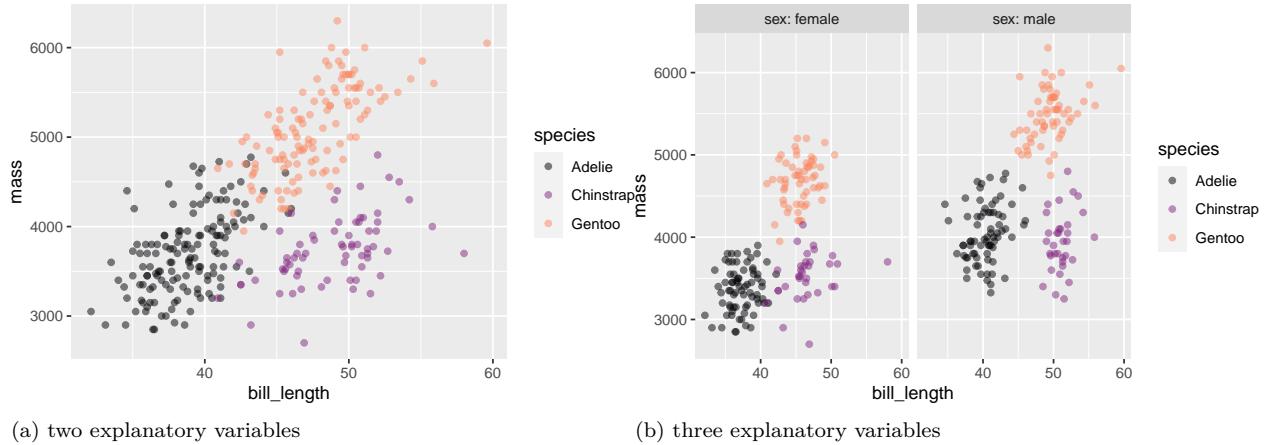


Figure 10: Point plots involving multiple explanatory variables.

In order to avoid long-winded sentences involving phrases like “the horizontal axis represents” we use the word **“mapped”**. For instance, in Figure 10, `mass` is mapped to `y`, `bill_length` is mapped to `x`, and `species` is mapped to color. Each mapping has a `scale` that translates the graphical property to a numerical or, in the case of color, categorical value.

`point_plot()` has been arranged so that the order of variable names in the tilde expression argument, `mass ~ bill_length + species`, exactly determines the mappings of variables to graphical properties. The response variable—that is, the variable named on the left-hand side of the tilde expression—is always mapped to `y`. The first variable on the right-hand side—`bill_length` in Figure 10—is always mapped to `x`. The second variable named on the right-hand side is always mapped to color.

In Figure 10(right), four variables are shown: the response `mass` as well as the three explanatory variables `bill_length`, `species`, and `sex`. Each variable needs to be mapped to a unique graphical property. `point_plot()` maps the third explanatory variable (if any) to a property called **“facet.”** Facets

are drawn as separate sub-panels. The scale for the mapping to facet consists of the labels at the top of each facet.

With `point_plot()`, different but closely related graphs of the same data can be made by swapping the order of variables named in the tilde expression. To illustrate, Figure 11 reverses the mappings `sex` and `species` compared to Figure 10(b). The data are the same in the two plots, but the different orderings of explanatory variables emphasize different aspects of the relationship among the variables. For instance, in `?@fig-mass-bill-species-sex(b)` it's easier to see that the sexes of each species differ in both mass and bill length. Chinstrap males and females have bill lengths that are the most distinct from one another.

```
Penguins |> point_plot(mass ~ bill_length + sex + species)
```

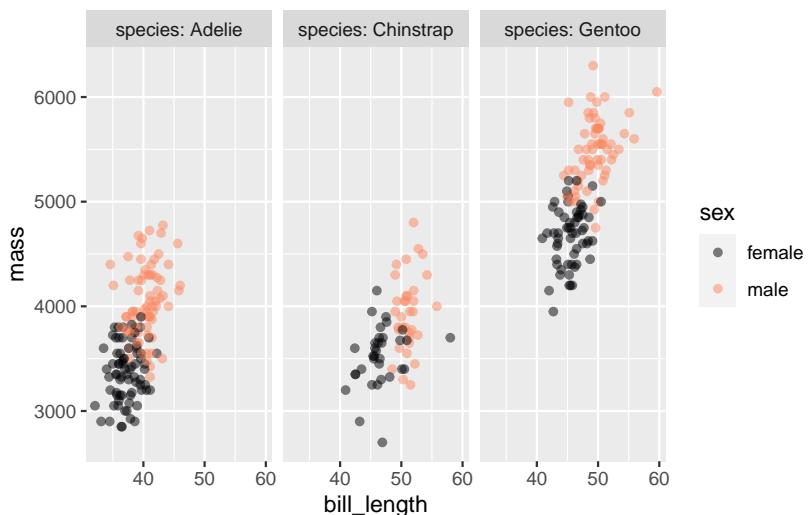


Figure 11: The same data as in Figure 10(b), but with `sex` mapped to color and `species` mapped to facet. This changes the visual impression created.

2.5 Graphical annotations

We can enhance our interpretation of patterns in the dots of a point plot by *adding “notes”* to the graphic, in other words,

“**annotating**” the graphic. Lessons 3 and 4 introduce different formats of statistical annotations that highlight different features of the data.

Here, to illustrate what we mean by a graphical annotation, we will use a familiar non-statistical annotation. Figure 12 replots the locations of world cities with an annotation showing continents and islands.

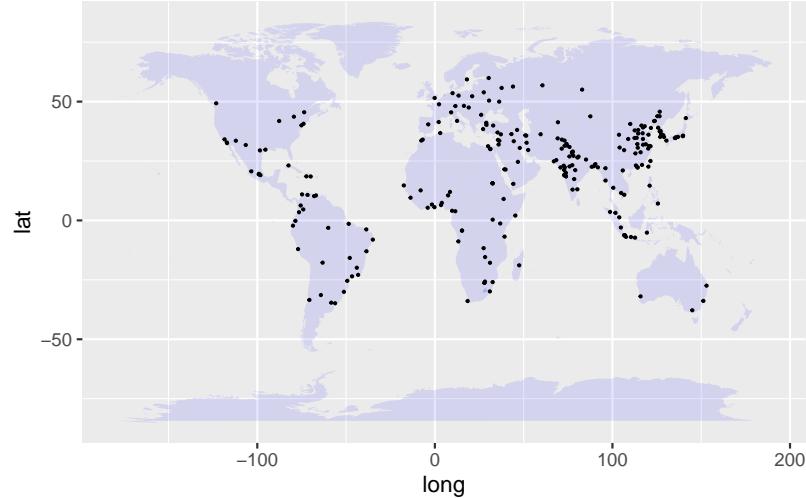


Figure 12: The latitude and longitude of the world’s 250 biggest cities annotated with a map of the continents and major islands.

Data shown without an annotation (Figure 7) may suggest a pattern. Adding an appropriate annotation enables you to judge the existence with the intuited pattern with much more confidence or, conversely, reject the pattern as a cloud-like illusion.

3 Variation and density, graphically

*Variation itself is nature’s only irreducible essence.
Variation is the hard reality, not a set of imperfect
measures for a central tendency. Means and me-
dians are the abstractions.* — Stephen Jay Gould
(1941- 2002), paleontologist and historian of science.

The point plots introduced in Lesson 2 are designed to show relationships between variables. Much of statistical thinking involves discovering, quantifying, and verifying such relationships. We have already introduced a concept structure for talking about relationships between variables: one variable is identified as the **response variable** and others as the **explanatory variables**. In point plots, we use the vertical axis for the response variable and the horizontal axis, color, and faceting for the explanatory variables.

Our focus in this Lesson is on describing the response variable. Remember that the origin of the word “variable” is in the specimen-to-specimen *variation* in values. We will look at variation in two distinct ways: 1) in this Lesson, the *shape* of the variation, and 2) in Lesson 9, the *amount* of the variation.

We mean “variable” in the statistical sense. People also talk about variables in algebra, which is only distantly connected with statistics.

3.1 The “shape” of variation

We turn to a familiar situation to illustrate variation: pregnancy and the duration of gestation—the time from conception to birth. It’s well known that typical human gestation is about nine months. But it varies from one birth to another. We can describe this variation using the `Births2022` data frame, a random sample of 20,000 births from the Centers for Disease Control’s census of 3,699,040 US births in 2022. The `duration` variable records the (estimated) period of gestation in weeks.

Figure 13(a) shows just the `duration` variable. It’s easy to see that durations longer than 45 weeks are rare. “Extremely preterm” births—defined as birth before the 28th week of gestation, are also uncommon. Most common are births at about 39 weeks, that is, about 9 months. The (vertical) spread of the dots shows the extent of variation in duration. The most

The tilde expression used is `duration ~ 1`, where 1 signifies that there are no explanatory variables.

common outcomes are at the value of `duration` where the dots have the most “**density**.”

```
Births2022 |>
  point_plot(duration ~ 1,
              # arguments specifying graphic details
              point_ink = 0.1, size = 0.2, jitter="y")
Births2022 |>
  point_plot(duration ~ 1, annot="violin",
              # graphic details
              point_ink = 0.1, size = 0.2, bw=0.5, jitter="y")
```



(a) Just the (jittered) data. (b) Annotating with a violin plot.

Figure 13: The `duration` (in weeks) of gestation for each of 20,000 randomly selected 2022 births in the US

For many people, the dots drawn in a point plot are reminiscent of seeds or pebbles scattered across an area. Density can be high in some areas, lower in others, negligible or nil in others. The spatial density pattern is called the “**distribution**” of the variable.

Many people can perceive density in a point plot without any need to count or calculate; it is an intuitive mode of perception. To illustrate, Figure 14 is a made-up point plot with five patches of different densities. The densities are 25, 50, 100, 200, and 400 points per unit area. Many people find it easy and immediate to point out the most dense patches and even to put the patches in order by density. However, people are hard put to qualify even the *relative* densities. For instance, the largest patch has a smaller density than the next largest patch, but quantifying this by eye (without being told the densities) is not really possible.

Indeed, a popular synonym for “point plot” is “scatter-plot.”

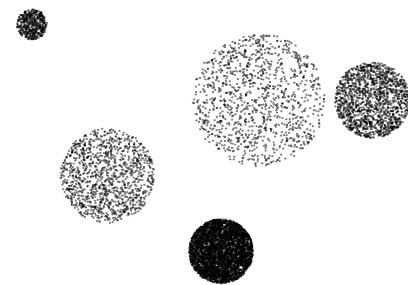


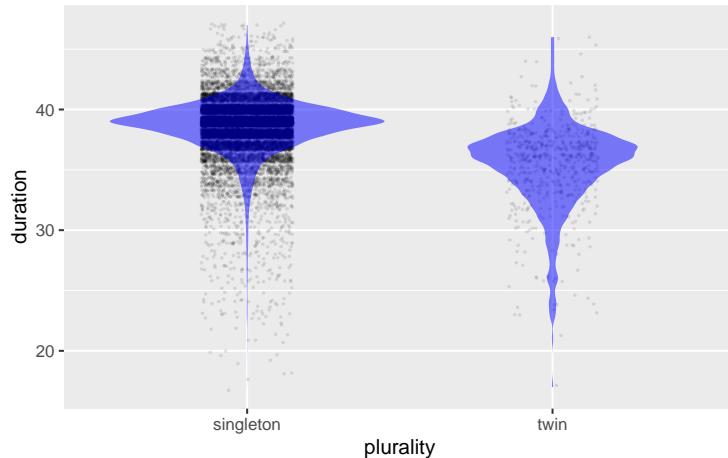
Figure 14: Five point-plot patches of different sizes and densities. The density can be perceived independently of the area.

Our eye gives a qualitative estimate of relative density, not a precise quantitative one. Our graphical perception is more precise when it comes to length or width. Ingeniously, designers of statistical graphics have created an annotation—called a “**violin**

i Example: Do twins take longer?

Violins can be informative when comparing two or more levels of an explanatory variable. To illustrate, consider the duration of gestation for twins versus singletons. Let’s see if the distribution of durations is different for the different kinds of birth.

```
Births2022 |>
  filter(plurality < 3) |>
  mutate(plurality = factor(plurality, labels = c("singleton", "twin"))) |>
  point_plot(duration ~ plurality, annot="violin",
             # the following specify graphics details
             point_ink = 0.1, size = 0.2, bw=0.5,
             jitter="y", model_ink=0.5)
```



In Figure 15, the density of points is vastly different for different levels of plurality. The jitter-column of singletons is much denser than for twins. Singletons are much more common than twins.

Figure 15: The distribution of duration shown separately for singletons, twins, and (a handful of) triplets.

Even though there are many more singletons than twins, the violins are roughly the same width. This is by design. The violins in Figure 15 tell the story of birth-to-birth variation of duration *within* each group. For twins, durations near 36 weeks are much more common than durations near 39 weeks. Similarly, comparing the two violins shows that premature births are much more likely for twins than for singletons. We can see this from the violins despite the fact that the large majority of premature births are of singletons.

It's worth emphasizing the previous point since it will be fundamental to statistical thinking.

3.2 Some simple shapes

There are infinitely many different shapes of distributions. Even so, a few simple shapes are common. These are shown in panels (a)-(d) of Figure 16. Panel (e) is a more complicated shape, infrequently seen in practice. (Unless you are practicing music rather than statistics!)

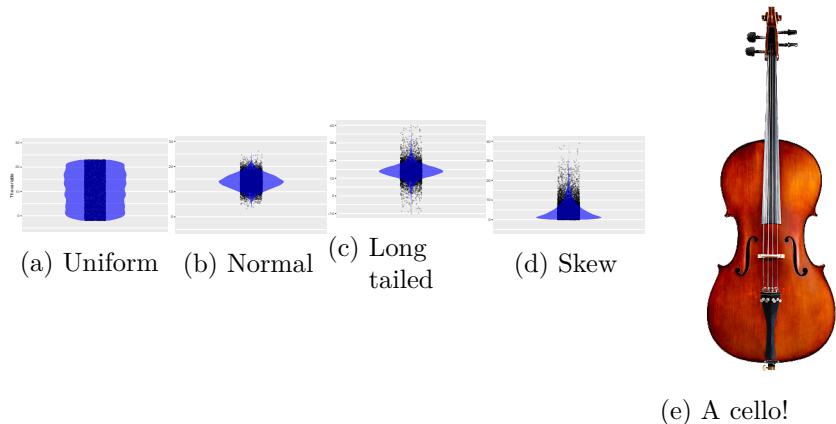


Figure 16: Various distribution shapes

Figure 16(a) is a “uniform” distribution, where each possible value is more or less equally likely. It’s not so common to see this in real-world data. When you do, it’s a good sign

that something artificial or mathematical is behind the data-generating process.

Much more common is the so-called “**normal**” distribution of Figure 16(b). The name given to it, “normal,” is an indication of how commonly it is seen. There is a region of highest density at middle values, with the density falling off symmetrically toward higher and lower values in a “bell-shaped” fashion.

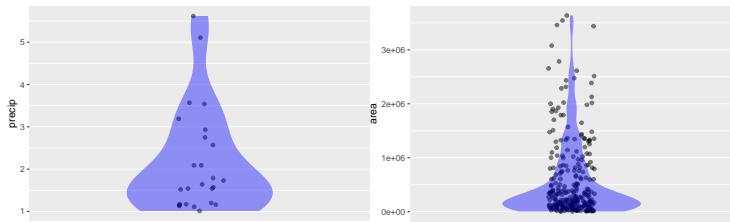
Other common patterns in distribution have a single peak (like the normal distribution) but have “tails” that extend much further than in the normal distribution. These are sometimes called “**long-tailed**” distributions. In Figure 16(c), the long tails are symmetrical around the peak, while in Figure 16(d), there is only one long tail. Such one-sided, long-tailed distributions are called “**skew**” distributions. Skew distributions are particularly common in economic data such as personal or national income.

There have been society-wide consequences to ignoring skewness in favor of “well-behaved,” short-tailed distributions such as the so-called normal distribution. For instance, the 2008 “Great Recession” was partly due to mistakenly high values on mortgage-backed and other financial securities. Financial analysts used valuation techniques that would be appropriate for normal distributions of risky events, but were utterly inadequate in the face of skew distributions.

i Example: Skew storms

A life-threatening setting for skew distributions concerns extreme events like large storms and fires.

```
Monocacy_river |>
  point_plot(precip ~ 1, annot="violin")
US_wildfires |>
  point_plot(area ~ 1, annot="violin")
```



(a) Inches of rain in storms in Maryland (b) Area burned by wildfires in the US each month

Figure 17: Examples of skew distributions

In this Lesson, we have emphasized the “shape” of variation, that is, the pattern of which values are more common and which less common. In Lesson 8 we turn to another aspect of variation that is central to statistical thinking: the **amount of variation**. Variation can be measured numerically, just as distance or position can be measured numerically. Happily, as we will see in Lesson 8, the name of the quantity often used to measure variation has a name—“**variance**

4 Annotating point plots with a model

Lesson Chapter 3 introduced the violin-plot annotation to display graphically the “shape” of variation: which values are more common, which values rare, and which values never seen at all. In this Lesson, we turn to a completely different sort of annotation showing a “**statistical model**.” Models provide a way to summarize quantitatively the **relationships** among variables.

4.1 Simple models

We define a “**simple model**” as a model with a single explanatory variable. We will be using simple models frequently, but also models with more than one explanatory variable. (*All* the models we consider in these Lessons have a single *response* variable.) To illustrate, let’s return to the anthropometric measurements displayed in Figure 8 where the explanatory variable is ankle circumference. Adding a statistical model annotation is accomplished by using the argument `annot = "model"`:

```
Anthro_F |> point_plot(Wrist ~ Ankle, annot = "model")
```

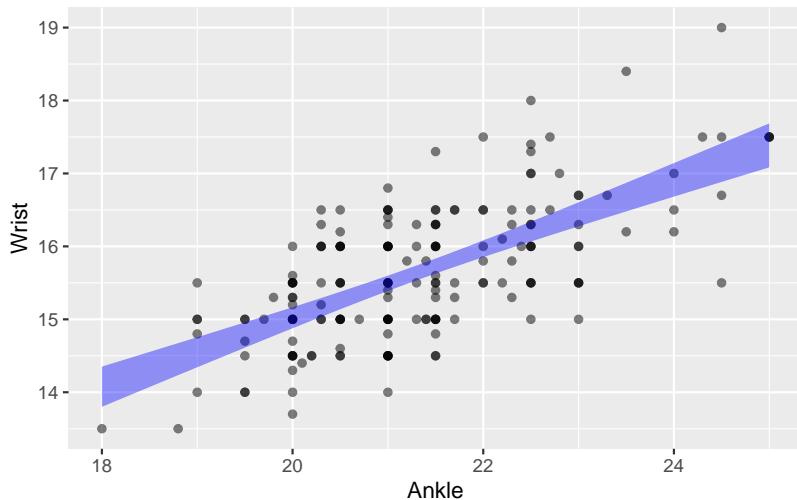


Figure 18: Annotating `Wrist ~ Ankle` point plot with a statistical model.

The model annotation is drawn as a more-or-less straight band, shaded blue, to help distinguish it from individual data points.

By default, `point_plot()` looks for a “**linear**” pattern in the data; the band is straight because we asked for it to be straight. The particular band presented by `point_plot()` is the one that comes as close as possible to the data points. “As close as possible” is defined in a specific way which we’ll investigate later; for now it suffices to note that the band goes nicely through the cloud of data points.

The explanatory variable in Figure 18 is *quantitative*. Model annotations can also be drawn for *categorical* explanatory variables. To illustrate, consider the data in `Birdkeepers`, used in a study of smoking, bird-keeping, and lung cancer. The unit of observation is an individual person. The variable `YR` records the number of years that person smoked, while the categorical variable `LC` indicates whether the person had been diagnosed with lung cancer. The data and a model annotation are shown in Figure 19

```
Birdkeepers |> point_plot(YR ~ LC, annot="model")
```

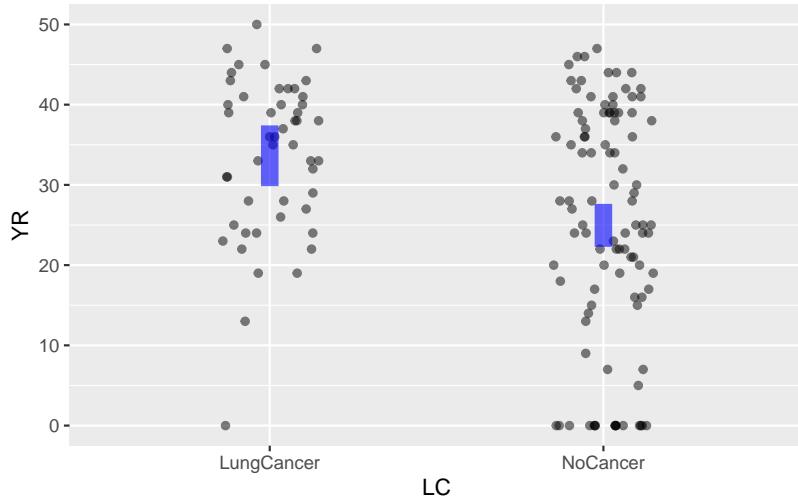


Figure 19: Years of cigarette smoking versus diagnostic status for lung cancer.

For a categorical explanatory variable, the model annotation is a vertical band (or “**interval**”) for each of the categorical levels. As with the band Figure 18, the model annotation in Figure 19 is vertically centered among the data points.

In later Lessons, we will discuss how `point_plot()` chooses the specific model annotation shown in any given case. But

consider these closely related questions:

- Why are the model annotations shown as a band or interval, rather than as a single, simple line or single numerical value for each category?
- What do the model annotations tell us?

The bands or intervals in the model annotations shown by `point_plot()` are there as a reminder that the data are consistent with a range of models. The height of the band/interval shows how large is that range. This is essential to drawing conclusions from the plots. For instance, in Figure 19, the intervals for two levels “lung cancer” and “no cancer” have no vertical overlap. This tells the statistical thinker to be confident in a claim that the typical value of `YR` is genuinely different between the two levels. If the intervals had overlapped vertically, the statistical thinker would know to be skeptical about such a claim.

Similarly, in Figure 18 the model annotation is a sloping band. The slope indicates that `Ankle` and `Wrist` are related to one another: larger `Wrists` tend to go along with larger `Ankles`. If the two variables were unrelated, we would expect the band to run horizontally—zero slope—meaning that the typical wrist circumference is the same for all people, regardless of the ankle circumference. The vertical thickness of the band tells the statistical thinker a range of plausible slopes that match the data. In Figure 8, there is no horizontal line that can be drawn from end to end *within* the band. Thus, the statistical thinker can be confident that there is a non-zero slope describing the relationship between `Wrist` and `Ankle`.

You may have encountered statistical graphics similar to those in `?@fig-point-estimate` but with an essential difference, the model annotations are lines rather than bands or intervals. In `?@fig-point-estimate`, the model annotations are simple lines that in principle are infinitely thin. With a numerical explanatory variable, the model annotation is a line that can have a non-zero slope. In contrast, for a categorical explanatory variable, there is a horizontal line drawn at a single vertical value for each level of the explanatory variable. Such simplified graphics do not recognize that, in reality, there is a range of different

lines that are plausible models. Since we can't tell from graphics like `?@fig-point-estimate` what is the range of plausible models, the annotation provides no guidance about whether to be confident that the data tell us slopes or vertical differences are non-zero.

Statistical thinking makes extensive use of the concept that there is a range of plausible models consistent with the data. Any straight line that falls into the band in Figure 8 is a plausible model of the data. In Figure 19, any pair of values that fall into vertical intervals are a plausible model of the data.

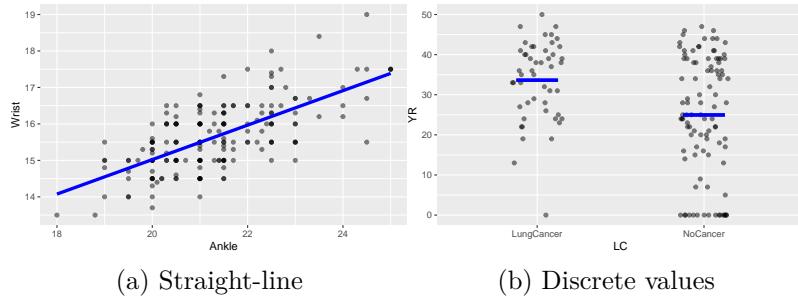


Figure 20: Many statistical texts present models as a straight line or as discrete values. This omits essential information compared to the model annotations generated by `point_plot()`.

i “Trend” or “cause”

Each of the plausible models—as in Figures 18 or 19—describes a specific relationship between the response and explanatory variables. For the wrist/ankle relationship, the plausible models all show a “trend” between ankle size and wrist size. For the smoking-years/lung-cancer relationship, the people with lung cancers “tend” to have smoked for more years than the no-cancer people.

The words “trend” or “tend” are weak. Often, statistical thinkers are interested in stronger statements, like these:

- Larger ankles *cause* larger wrists.
- Smoking for more years *increases the chances* of lung

cancer.

We can call these **opinionated statements** because they make use of some hypothesis about how the world works held by the modeler rather than being forced solely by the data. Many people think it silly to claim that “larger ankles *cause* larger wrists.” It seems much more probable that “larger people have larger wrists and also larger ankles.” On the other hand, many people will be sympathetic to the statement “increases the chances of lung cancer.” They have heard such things from other respected sources.

Some of the techniques covered in these *Lessons* are designed to substantiate or undermine opinionated statements like these. Until we understand and use these techniques, it is dicey to quantitatively support an opinionated statement from data.

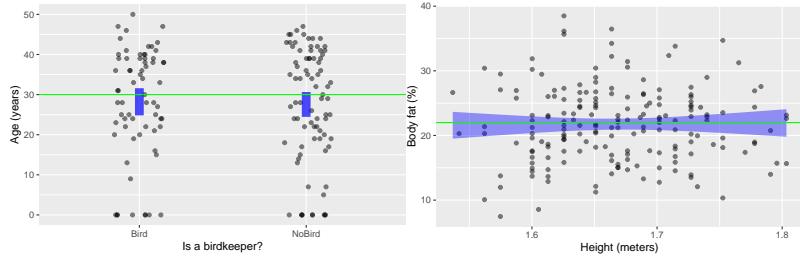
Many statisticians prefer to avoid the whole matter of opinionated statements. Weak, unopinionated language like “trend” or “tend” are used instead. Those preferring more technical-sounding language might use “associated with” or “correlated with.”

But see Lesson 26 for an approach approved by even the most opinion-wary statistician.

4.2 Independence

We use model annotations to display whether variables are related. It’s good to consider as well a particular type of relationship: independence. When the explanatory variable is categorical, the model annotations will be a vertical interval for each level. When the response is independent of the explanatory variable, those intervals will overlap. For instance, in `?@fig-independence(a)` values of YR near 30 are in both vertical intervals.

For a quantitative explanatory variable, as in `?@fig-independence(b)`, independent variables will have a model band that is more-or-less horizontal. That is to say, at least one horizontal line will fall within the band.



(a) Categorical explanatory variable (b) Quantitative explanatory variable

Figure 21: Model annotations consistent with the response and explanatory variables being independent. Panel (a) considers whether the age of the people in `Birdkeepers` is independent of whether the person keeps a bird. Panel (b), based on `Anthro_F` is about the possible relationship between a person's height and body fat as a percent of overall mass

4.3 Multiple explanatory variables

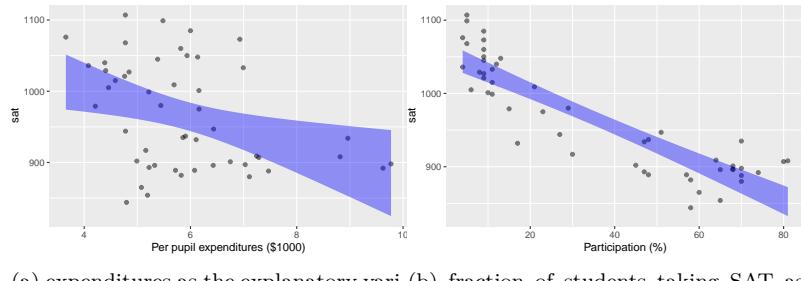
In Lesson 2 we used color and faceting to look at the response variable in terms of up to three explanatory variables. Statistical models can also handle multiple explanatory variables.

We'll illustrate with a commentary from a political pundit about education spending in US schools:

[T]he 10 states with the lowest per pupil spending included four — North Dakota, South Dakota, Tennessee, Utah — among the 10 states with the top SAT scores. Only one of the 10 states with the highest per pupil expenditures — Wisconsin — was among the 10 states with the highest SAT scores. New Jersey has the highest per pupil expenditures, an astonishing \$10,561, which teachers' unions elsewhere try to use as a negotiating benchmark. New Jersey's rank regarding SAT scores? Thirty-ninth... The fact that the quality of schools... [fails to correlate] with education appropriations will have no

effect on the teacher unions' insistence that money is the crucial variable.—George F. Will, (September 12, 1993), “Meaningless Money Factor,” The Washington Post, C7.

The opinionated claim here is that “money is the crucial variable” in educational outcomes. George Will seeks to rebut this claim with data. Fortunately for us, actual data on SAT scores and per pupil expenditures in the mid-1990s is available in the `mosaicData::SAT` data frame. The unit of observation in `SAT` is a US state. Figure 22(a) shows an annotated point plot of state-by-state expenditures and test scores. The trend signaled by the model annotation is that SAT scores are slightly lower in high-expenditure states, consistent will George Will’s observations. But ...



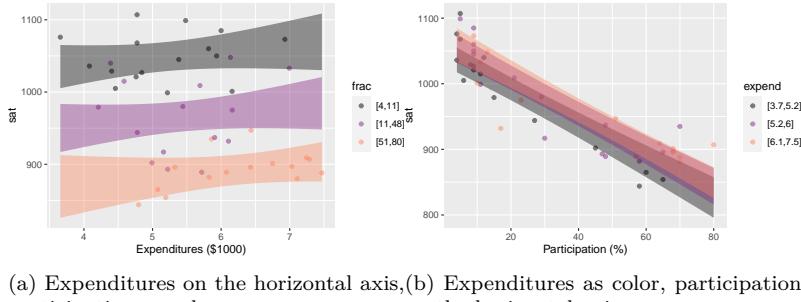
(a) expenditures as the explanatory vari-(b) fraction of students taking SAT as able
able the explanatory variable

Figure 22: SAT scores as a function of per-pupil expenditures and of fraction taking the SAT.

Education is a complicated matter and there are factors other than expenditures that may be playing a role. One of these, shown in Figure 22(b), is that participation in the SAT varies substantially from state to state. In some states, almost all students take the test. In others, fewer than 10% of students take the test. The data show a relationship between participation and scores: scores are consistently higher in low-participation states.

```
SAT |> point_plot(expend ~ frac, annot="model") |>
  add_plot_labels(x = "Participation (%)", y = "Per pupil expenditures ($1000s)")
```

Statistical modeling techniques enable us to use *both* expenditures and participation as explanatory variables. Figure 22 does this with *one variable at a time*. But we can also use both explanatory variables *simultaneously*. Doing so is important especially when there is a relationship between the explanatory variables, as seen in the graph of expenditures versus participation (Figure 23).



(a) Expenditures on the horizontal axis, participation as color.
 (b) Expenditures as color, participation on the horizontal axis.

Figure 24: A model of SAT scores as a function of both expenditures and participation. The model is the same for both (a) and (b), but the horizontal axis and color is reversed from one to the other.

The two panels in Figure 24 tell a consistent story, but with different graphical appearances. For instance, the clear vertical spacing between bands in the left panel indicate that SAT scores are influenced by the participation level, even taking into account expenditures. This appears as the downward slope in the bands in the right panel.

But when we look at expenditures—taking into account participation—we see horizontal bands in the left panel. (More precisely, bands that can encompass a horizontal line.) This indicates that we cannot confidently claim that expenditures are associated with SAT scores. In the right panel, the lack of association between expenditures and SAT scores is signaled by the vertical overlap between the bands.

Many people are discomfitted to hear that looking at the same data in different ways can lead to different conclusions. At this stage in the *Lessons* that may be true for you as well. Even so,

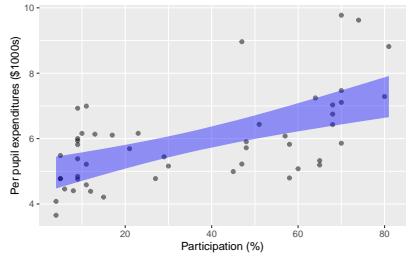


Figure 23: The two explanatory variables shown in Figure 22 are themselves related to one another.

should already have a concrete sense of how we can denote the “different ways of looking at data.” In modeling notation, the perspective `sat ~ expenditures` shows one pattern, while the perspective `sat ~ expenditures + participation` tells another. In Lesson 22 we will see a non-graphical way of looking at models that makes it easier to see the effect of one explanatory variable in the context of others. And in Lessons 23 and 26 we will study the methods used in modern statistics to decide which of two possible models—say `sat ~ expenditures` or `sat ~ expenditures + participation`—is more appropriate to answer questions of causation.

5 Data wrangling

Data wrangling refers to the organization and construction of simple summaries of data, or preparing data in the more nuanced summaries of statistical models. Traditionally, organizing data has been a complicated task involving extensive computer programming. The style of data wrangling is more modern and much less demanding for the human wrangler. Wrangling takes advantage of the realization made only in the last half century or so that a *small set of simple operations* can handle a large variety of re-organization tasks. The data scientist is to learn what are these operations and how to invoke them on the computer.

5.1 Basic data-wrangling operations

The basic structure of every data wrangling operation is that a data frame is the input and another (possibly) modified data frame is the output. This data-frame-in/data-frame-out organization to be divided among a number of small, simple steps, each step involving taking a data frame from the previous steps and supplying the modified frame it to the subsequent steps.

What are these steps? One is to **arrange** the rows of a data frame according to a specific criteria. Another is the elimination or **filtering** of some rows based on a user-specified criteria. **Mutate**, another operation, adds to a data frame new columns that have been calculated from the original columns. The **summarize** operation reduces many rows to one, effectively changing the unit of observation. Still another is **selecting** certain variables from the data frame and discarding the remaining ones.

Each of these five operations is conceptually, simple and relies only on the human data wrangler specifying the criteria for selection or exclusion, how to calculate new variables from old, or the definition of groups for summarization. We will use these five operations—arrange, filter, mutate, select, and summarize—over and over again in the rest of these Lessons.

The Big Five wrangling operations

You will use these throughout the *Lessons*.

1. arrange
2. filter
3. mutate
4. select
5. summarize

Others you will see in examples:

- *pivot*, an example of which is given in this Lesson.
- *join*, covered in Lesson 7.

Experts in data wrangling learn additional operations. One that we will use occasionally in examples is **pivoting**, which changes the shape of the data frame without changing its contents. Another expert operation is called **join** and involves combining two data frame inputs into a single frame output. Learning about “joins” is important for two reasons. Join is the essential operation for assembling data from different sources [Sometimes called “data linkage.”](#). For instance, research on educational effectiveness combines data from academic testing with income and criminal records. A second important reason for learning about “join” is to understand why related data is often spread among multiple data frames and how to work with such data. We will consider such “**relational data bases**” in Lesson [7](#).

Learning how to use and understand the basic operations, particularly the big five, can be accomplished with simple examples. To use the operations, you need only know the name of the operation and what kind of auxilliary input is needed to specify exactly what you want to accomplish. We will demonstrate using a compact, made-for-demo data frame, **Nats**, that has both categorical and numerical data. (The example is motivated by the famous [Gapminder](#) organization that combines nation-by-nation economic, demographic, and health data in a way that illuminates the actual (often counter-intuitive) trends.)

1. arrange()

`arrange()` sorts the rows of a data frame in the order dictated by a particular variable. For example:

```
Nats |> arrange(pop)
```

```
| country | year | GDP | pop |
|:-----:|:----:|:---:|:---:|
| Cuba   | 2020 | 80  | 7   |
| Cuba   | 1950 | 60  | 8   |
| Korea  | 2020 | 874 | 32  |
| Korea  | 1950 | 100 | 32  |
```

Figure 25: A made-up, compact data set for simple data wrangling demos. GDP is in \$trillions, pop is in millions.

country	year	GDP	pop
Korea	2020	874	32
Cuba	2020	80	7
France	2020	1203	55
India	2020	1100	1300
Korea	1950	100	32
Cuba	1950	60	8
France	1950	250	40
India	1950	300	700

```

| France | 1950 | 250 | 40 |
| France | 2020 | 1203 | 55 |
| India  | 1950 | 300 | 700 |
| India  | 2020 | 1100 | 1300 |

```

Numerical variables are sorted numerically, categorical variable are sorted alphabetically

For other examples, for instance how to sort in *descending order*, see Exercise Exercise 5.31

2. filter()

The `filter()` function goes row-by-row through its input, determining according to a user-specified criterion which rows will be passed into the output. The criterion is written in R notation, but often this is similar to arithmetic notation. In the following, `pop < 40` states the criterion “population is less than 40,” while `year == 2020` (notice the double equal signs) means “when the year is 2020.”

```
Nats |> filter(pop < 40)
```

```

| country | year | GDP | pop |
|:-----:|:---:|:---:|:---:|
| Korea  | 2020 | 874 | 32 |
| Cuba   | 2020 | 80  | 7  |
| Korea  | 1950 | 100 | 32 |
| Cuba   | 1950 | 60  | 8  |

```

```
Nats |> filter(year == 2020)
```

```

| country | year | GDP | pop |
|:-----:|:---:|:---:|:---:|
| Korea  | 2020 | 874 | 32 |
| Cuba   | 2020 | 80  | 7  |
| France | 2020 | 1203 | 55 |
| India  | 2020 | 1100 | 1300 |

```

For more examples, see Exercise 5.32.

3. mutate()

Sometimes the information needed is already in the data frame, but it is not in a preferred form. For instance, `Nats` has variables about the size of the economy (gross domestic product, `GDP`, in \$billions) and the size of the population (in millions of people). In comparing economic activity between countries, the usual metric is “*per capita* GDP” which is easily calculated by division. The `mutate()` function carries out the operation we specify and gives the result a name that we choose. Here’s how to calculate *per capita* GDP, and store the result under the variable name `GDPpercap`:

```
Nats |> mutate(GDPpercap = GDP / pop)
```

country	year	GDP	pop	GDPpercap
Korea	2020	874	32	27.31
Cuba	2020	80	7	11.43
France	2020	1203	55	21.87
India	2020	1100	1300	0.8462
Korea	1950	100	32	3.125
Cuba	1950	60	8	7.5
France	1950	250	40	6.25
India	1950	300	700	0.4286

Pay particular attention to the argument inside the parentheses, `GDPpercap = GDP / pop`. The `=` symbol means “give the name on the left (`GDP`) to the values calculated on the right (`GDP / pop`). This style of argument, involving the `=` sign, is called a **named argument**. In these **Lessons** `=` will only ever appear as part of a named argument expressions. One consequence is that `=` will only appear inside the parentheses that follow a function name.

4. select()

Data frames often have variables that are not needed for the purpose at hand. In such circumstances, you may discard the unwanted variables with the `select()` command. Select takes as arguments the *names* of the variables you want to **keep**, for instance:

```
Nats |> select(country, GDP)
```

```
| country | GDP |
|:-----:|:----:|
| Korea  | 874 |
| Cuba   |  80 |
| France | 1203 |
| India  | 1100 |
| Korea  | 100 |
| Cuba   |  60 |
| France | 250 |
| India  | 300 |
```

Alternatively, you can specify the variables you want to **drop** by using a minus sign before the variable name, as in this calculation:

```
Nats |> select(-year, -pop)
```

```
| country | GDP |
|:-----:|:----:|
| Korea  | 874 |
| Cuba   |  80 |
| France | 1203 |
| India  | 1100 |
| Korea  | 100 |
| Cuba   |  60 |
| France | 250 |
| India  | 300 |
```

5. summarize()

“To summarize” means to give a brief statement of the main points. For the data-wrangling `summarize()` operation, “brief” means to combine rows. For instance, one summary of the `Nats` data would be the total population of all the countries.

```
Nats |> summarize(totalpop = sum(pop))
```

```
| totalpop |
|:-----:|
|    2174  |
```

The `sum()` function used in the above command merely adds up all the values in its input, here `pop`. `summarize()` is a data-wrangling operation, while `sum()` is a simple arithmetic operation. Functions such as `sum()` are called “reduction functions”: they take a variable as input and produce a single value as output. You will be using over and over again a handful of such reduction functions: `mean()`, `max()`, `min()`, `median()` are probably familiar to you. Also important to our work will be `var()`, to be introduced in Chapter 8, which quantifies the amount of variation in a numerical variable.

The result from the previous command, 2174, is arithmetically correct but is misleading in the context of the data. After all, each country in `Nats` appears twice: once for 1950 and again for 2020. The populations for both years are being added together. Typically, you would want *separate* sums for each of the two years. This is easily accomplished with `summarize()`, using the `.by=` argument: Notice the period at the start of the argument name: `.by =`

```
Nats |> summarize(totalpop = sum(pop), .by = year)
```

```
| year | totalpop |
```

2020	1394
1950	780

Note that the output of the summarize operation and has mostly different variable names and the input, in addition to squeezing down the rows, adding them up, touch, summarize retains only the variables used for grouping and discards the others, but adds in columns For the requested summaries.

5.2 Compound wrangling statements

Each of the examples in Section 5.1 involved just a *single* wrangling operation. Often, data wrangling involves putting together multiple wrangling operations. For instance, we might be interested in finding the countries with above average GDP per capital, doing this separately for 1950 and 2020:

```
Nats |>
  mutate(GDPpercap = GDP / pop) |>
  filter(GDPpercap > mean(GDPpercap), .by=year)
```

country	year	GDP	pop	GDPpercap
Korea	2020	874	32	27.31
France	2020	1203	55	21.87
Cuba	1950	60	8	7.5
France	1950	250	40	6.25

Let's take this R command apart. The high-level structure is

`Nats |> mutate() |> filter()`, or, more abstractly,

object |> *action* |> *action*.

An “object” is something that can be retained in computer storage, such as a data frame. An “action” is an operation that is performed on an object and produces a new object as a result. A great advantage of the pipeline style for commands is that

every statement following the pipe symbol (`|>`) will *always* be an action, no doubt about it.

Another way to spot that something like `mutate()` refers to an action is that the name of the action is directly followed by an opening parenthesis. In R, the pair `(` and `)` means “take an action.” It’s not used for any other purpose.

Constructing a compound wrangling command involves creativity. Like any creative art, mastery comes with experience, failure, and learning from examples such as those in the Exercises.

5.3 Actions and adverbs; functions and arguments

We’ve already mentioned that expressions like `mutate()` or `arrange()` refer to actions. A more technical word than “action” is “function”: `mutate()` and `arrange()` and many others are **functions**. The functions we use have names which, in the ideal situation, remind us of what kind of action the function performs. When we write a function name, the convention in these *Lessons* is to follow the name with a pair of parentheses. This is merely to remind the reader that the name refers to a function as opposed to some other kind of object such as a data frame.

In use, functions generally are written with one or more **arguments**. The arguments are written in R notation and specify the details of the action. They are always placed inside the parentheses that follow the function name. If there is more than one argument, they are separated by commas. An example:

```
select(country, GDP)
```

The action of the `select()` function is to create a new data frame with the columns specified by the arguments. Here, there are two arguments, `country` and `GDP`, which correspond to the two columns that the new data frame will consist of. In English, we might describe `select(country, GDP)` this way: “Whatever is the input data frame, create an output that has only the specified variables.”

On its own, `select(country, GDP)` is not a complete command. It is missing an important component for a complete command: which data frame the action will be applied to. To complete the sentence. In the R pipeline grammar, we specify this using the pipe symbol `|>`, as in `Nats |> select(country, GDP)`.

In terms of English grammar, actions are **verbs** and statements that modify or qualify the action are **adverbs**. For example, the English “run” is a verb, an action word. We can modify the action with adverbs, as in “run swiftly” or “run backward.” In R, such verb phrases would be written as *function(adverb)* as in `run(swiftly)` or `run(backward)`. When there are multiple adverbs, English simply puts them side-by-side, as in “run swiftly backward.” In R this would be `run(swiftly, backward)`.

The wrangling verbs `summarize()` and `mutate()` create columns. It’s nice if those columns have a simple name. You can set the name to be used by preceding the adverb by the name you would want followed by an equal sign. Examples: `summarize(mn = mean(flipper))` or `mutate(ratio = flipper / mass)`.

i Imperatives and objects

In English, a sentence like “Walk the dog!” is an imperative, a command. Similarly, in R, commands are always imperatives. The English imperative sentence, “Jane, walk the dog!” directs the imperative to a particular actor, namely Jane. The R imperative is always directed to “the computer,” as in, “Computer, select the `country` and `GDP` columns for the output.”

“Walk the dog!” has both a verb (“walk”) and a noun (“the dog”). The noun in such an imperative is the **object** of the verb; the entity that the action (walk) is to be applied to.

R structures sentences/commands differently. Every sentence is a command. The actor is always the computer, there’s no reason to state that explicitly. So the imperative in R looks like this:

```
the_dog |> walk
```



[](https://www.youtube.com/watch?v=...)

Click link to play movie clip.

In word order, the object of the action *preceeds* the action.
In data-wrangling commands, the object is always a data frame.

5.4 Pivoting (optional)

In an earlier example, we used `mutate()` to compute a new column called `GDPpercap` by dividing two existing columns, `GDP` and `pop`. With `mutate()`, it's easy to do calculations that involve two or more columns within the same row.

Now consider a similar sounding task, computing `GDPgrowth` by dividing, for each country separately, the 2020 GDP with the 1950 GDP. This cannot be done with a simple `mutate()` step because the information needed for the calculation is spread over two different rows. A clue to the difficulty is that there are not separate columns named, say, `GDP2020` and `GDP1950` that could be combined with a `mutate()` operation.

“**Pivoting**” is a data wrangling operation that reshapes a data frame. Understanding pivoting is essential for the professional data scientist. But, like the construction of compound wrangling statements in general, mastery comes with experience. You won’t need to master pivot to study these *Lessons*, but we do use it behind the scenes in some of the demonstrations. Mainly, it’s worthwhile to learn a little about pivoting in order better to appreciate how data wrangling uses a small number of general-purpose operations to accomplish a huge variety of tasks.

Consider a data frame for which you want to turn information in different rows into a format with that information in different columns. That is, we’re going to take information from a single column in the original, and spread it between two (or more) columns in the output from the operation. Adding columns is effectively making a data frame “**wider**.” We can accomplish the `GDPgrowth` wrangling by pivoting from “**longer**” (that is, more rows) to “**wider**”. Like this:

```
Nats |> pivot_wider(country, values_from = c(GDP, pop), names_from = year)
```

country	GDP_2020	GDP_1950	pop_2020	pop_1950
Korea	874	100	32	32
Cuba	80	60	7	8
France	1203	250	55	40
India	1100	300	1300	700

This is a complicated command, so we will break it down argument by argument.

1. The first argument, `country`, specifies the variable values that will label each row in the result. Even though `country` has eight values, there are only four distinct values so the result will have four rows.
2. The second argument, `values_from = c(GDP, pop)`, tells which columns we are going to make wider. Here, we are creating side-by-side columns for both `GDP` and `pop`.
3. The third argument, `names_from = year`, tells what variable in the original will spread of the columns in the second argument. Since `year` has two distinct values (1950 and 2020), the `values_from` columns will be split into two columns each. If `year` had three distinct values (say, 1980 as well as 1950 and 2020), then the splitting would be into three columns for each of the `values_from` columns.

...:

The pivoted data contains the same information as the original, but organized differently. The new organization makes it easy to do the `GDPgrowth` calculation, since now it is just the ratio of two columns:

```
Nats |>
  pivot_wider(country, values_from = c(GDP, pop), names_from = year) |>
  mutate(GDP_growth = GDP_2020 / GDP_1950) |>
  select(country, GDP_growth)
```

country	GDP_growth
Korea	8.74
Cuba	1.333
France	4.812
India	3.667

As you might suspect, there is also a `pivot_longer()` operation, which merges columns rather than spreading them.

i Exercise 5.27 Q05-113

DRAFT

I want to do some things with babynames, but I have to keep them simple.

NOTE IN DRAFT: Move this to a section for more advanced exercises: perhaps Mini-Projects.

See Z-cat-wear-canoe.qmd

i Exercise 5.28 Q05-114

DRAFT

Compute the mother's weight gain during pregnancy as indicated by `natality2014::Natality_2014_100k` and graph it versus baby's weight `dbwt`. Is there an obvious relationship between the two variables? CHANGE THIS TO `Births2022` where the variables are `weight_pre`, `weight_delivery`, `weight`.

i Exercise 5.29 Q05-115

DRAFT

Average family size from the child's point of view and from the households point of view.

i Exercise 5.30 Q05-116

DRAFT

Parents is just one of the reasonable formats for storing the parent data. Another perfectly good format would have the unit of observation be an individual parent rather than a mother/father pair.

```
{r echo=FALSE, eval=FALSE} #| label:  
tbl-parent #| tbl-cap: "The `Parent` (singular)  
data frame where the unit of observation is an  
individual parent." #| tbl-cap-location: margin  
Parents |> pivot_longer(cols = c("mother",  
"father"), values_to = "parent_height",  
names_to = "role")
```

i Exercise 5.31 Q05-104

SORTING EXERCISES, including sorting in descending order.

i Exercise 5.32 Q05-105

FILTERING EXERCISES

Include %in%

i Exercise 5.33 panda-sleep-cotton

```
“‘{r label='Z-panda-sleep-cottonOE4ul6', include =  
FALSE} set.seed(102)  
Big <- babynames::babynames %>% filter( n > 500 )  
theseNames <- unique(Big %>% sample_n(size =  
10))[[“name”]] %>% head( 3 )  
One <- babynames::babynames |> filter( name %in% the-  
seNames, year %in% c(1912,2012)) |> summarize( nbab-  
bies=sum(n), .by = c(name, sex, year)) |> arrange(year)  
|> tail(10)  
Two <- One |> spread( key=sex, value=nbabies, fill = 0  
)  
Three <- One |> spread( key=year, value=nbabies, fill =
```

```

0 )
Four <- One |> spread( key=name, value=nbabies, fill =
0)
options( xtable.NA.string = "NA" )
Five <- Four |> spread(key=sex, value=year )

```

Consider the following forms for organizing a subset of the `babynames::babynames`.

```

```{r panda-sleep-cotton-1, echo=FALSE, caption="Version One"}
One

{r panda-sleep-cotton-3, echo=FALSE,
caption="Version Two"} Two
{r panda-sleep-cotton-4, echo=FALSE,
caption="Version Three"} Three

```

1. Do all three forms represent the same set of facts? r  
`short_answer(r"--(Yes. For instance, from  
all three forms you can see that in 1912  
there were 8764 baby girls and 22 baby  
boys given the name Mildred.)--")`
2. What is the meaning of a row in each of the tables?
  - Version One r `short_answer(r"--(A name  
given to babies of a specific sex in a  
specific year)--")`
  - Version Two r `short_answer(r"--(A name  
in a year)--")`
  - Version Three r `short_answer(r"--(A name  
given to a sex )--")`
3. Which form of table makes it easiest to calculate  
the ratio of male to female in each name in each  
year? r `short_answer(r"--(For Two. A simple  
application of \tt{mutate()} can calculate  
the ratio.)--")`
4. There are no zeroes in Version One, but there  
are in Versions Two and Three. Why? r  
`short_answer(r"--(In version Two, there`

is a slot for every combination of name and year and sex, even when the count of babies is zero, whereas version One only has slots for non-zero counts of the count of babies. )--")

5. Version Three was “spread” from Version One. What variable sets the spread columns? `r short_answer(r"--(\tt{year})--")`

### **i** Exercise 5.34 kid-bend-dish

Use the `DataComputing::ZipGeography` and `DataComputing::ZipDemography` data tables to create these graphics.

1. Call a ZIP code “crowded” when the population is over 50,000. Make a point plot showing the geographic location of the crowded ZIPs.
2. Pick out the 10,000 zip codes with the highest population. Make a point plot of the latitude versus longitude. (Hints: `arrange()` and `head()`.) Use color to represent the fraction of the population that is over 65.
3. How many zip codes have a `WaterArea` that is more than 50% of the `LandArea`? Make a point plot showing the geographical location of these, with color indicating the population.

### **i** Exercise 5.35 doe-dream-room

Using the `DataComputing::ZipGeography` data table, answer the following questions. In addition to the answer itself, show the statement that you used and the data table created by your statement that contains the answer.

1. How many different counties are there? (Hint: Use `n_distinct()` within `summarize()`, but be careful

- what variable(s) you group by.)
2. Which city names are used in the most states?
  3. Identify cities that include more than 5% of their state population; which of those city names are used in the most states?
  4. Does any state have more than one time zone?
  5. Does any city have more than one time zone?
  6. Does any county have more than one time zone?

### **i** Exercise 5.36 seaweed-tug-kayak

Imagine a data table, `Patients`, with categorical variables `name`, `diagnosis`, `sex`, and quantitative variable `age`. You have a statement in the form

```
Patients %>%
 summarise(count = n(), meanAge = mean(age),
 .by = **SOME_VARIABLES**)
```

Replacing `**SOME_VARIABLES**` with each of the following, tell what variables will appear in the output

- a. `sex`
- b. `diagnosis`
- c. `c(sex, diagnosis)`
- d. `c(age, diagnosis)`
- e. `age`

```
r long_answer()
```

The variables given as arguments to `.by =` will always appear in the output, along with whatever variables are created by `summarise()`.

- a. `sex, count, meanAge`
- b. `diagnosis, count, meanAge`
- c. `sex, diagnosis, count, meanAge`
- d. `age, diagnosis, count, meanAge`
- e. `age, count, meanAge`

```
r end_long_answer()
```

### **i** Exercise 5.37 snail-sing-knife

Each of these tasks can be performed using a single data verb. For each task, say which verb it is:

1. Add a new column that is the ratio between two variables. `r short_answer(r"--(\tt{mutate()})--")`
2. Sort the cases in descending order of a variable. `r short_answer(r"--(\tt{arrange()})--")`
3. Create a new data table that includes only those cases that meet a criterion. `r short_answer(r"--(\tt{filter()})--")`
4. From a data table with three categorical variables A, B, & C, and a quantitative variable X, produce an output that has the same cases but only the variables A and X. `r short_answer(r"--(\tt{select()})--")`
5. From a data table with three categorical variables A, B, & C, and a quantitative variable X, produce an output that has a separate case for each of the combinations of the levels of A and B. `r short_answer(r"--(\tt{summarise()} using \tt{.by = c(A, B)})--")`

### **i** Exercise 5.38 squirrel-by-rug

```
``{r squirrel-buy-rug-default, include = FALSE} # some
row calculations for chunk specs BabyNames_All <- baby-
names::babynames |> nrow()
BabyNames_F <- babynames::babynames |> filter(sex
== "F") |> nrow()
BabyNames_M <- babynames::babynames |> filter(sex
== "M") |> nrow()
```

TITLE GOES HERE: Here is a reminder of what the `BabyNames` data frame looks like. Take this

```
```{r squirrel-buy-rug-1, echo=FALSE, and_so_on= paste0("... and so on for ", BabyNames_All,
set.seed( 102 )
Small <- sample_n( babynames::babynames, size = 10 )
```

```
row.names( Small ) <- NULL  
Small
```

For each of the following outputs, identify the data verb linking the input to the output and write down the details (i.e., arguments) of the operation. (Hints: Look at the number of cases. The order of variables may also be a clue.)

1. Output Table A

```
{r squirrel-buy-rug-A, echo=FALSE,  
and_so_on= paste0("... and so on for ",  
BabyNames_All, " rows altogether.")} Small  
> dplyr::arrange(sex, n )
```

2. Output Table B

```
{r squirrel-buy-rug-B, echo=FALSE, and_so_on=  
paste0("... and so on for ", BabyNames_F, "  
rows altogether.")} Small |> filter(sex == "F"  
)
```

3. Output Table C

```
{r squirrel-buy-rug-C, echo=FALSE, and_so_on=  
paste0("... and so on for ", BabyNames_M, "  
rows altogether.")} Small |> filter(sex == "M")
```

4. Output Table D

```
{r squirrel-buy-rug-D, echo=FALSE}  
babynames::babynames |> summarise(total =  
sum(n) )
```

5. Output Table E

```
{r squirrel-buy-rug-E, echo=FALSE,  
nrow=BabyNames_All} Small |> select(name, n  
)  
r long_answer()
```

- a. `arrange(sex, n)`
- b. `filter(sex == "F")`
- c. `filter(sex == "M", n > 10)`
- d. `summarise(total = sum(n))`
- e. `select(name, n)`

```
r end_long_answer()
```

::::

6 Computing with functions and arguments

You have already seen the R command patterns that we will use throughout these *Lessons*: pipelines composed of actions separated by `|>`, named data frames, functions and their arguments. This Lesson recapitulates and explains those patterns, the better to help you construct your own commands. The Lesson also emphasizes a small technical vocabulary that helps in communicating with other people to share insights and identify errors.

Learning command patterns is a powerful enabler in statistical thinking. The computer is an indispensable tool for statistical thinking. In addition to doing the work of calculations, computer software is a medium for describing and sharing with others statistical techniques and concepts. For instance, three fundamental operations in statistics—randomize, repeat, collect—are easily written in computer languages but have no counterpart in the algebraic notation traditionally used in mathematics education.

This [blog post](#) gives a little history of the long-standing link between statistics and computing.

6.1 Chain of operations

A typical computing task consists of a chain of operations. For instance, each wrangling operation receives a data-frame object from an incoming pipe `|>`, operates on that data frame to perform the action described by the function's name and arguments, and produces another data frame as output. Depending on the overall task, the output from the operation may be piped into another action or displayed on the screen.

There are also operations, like `point_plot()`, that translate a data frame into another kind of object: a *graphic*. Starting with Lesson 11, we will work with `model_train()`, a function that translates a data frame into a *model*. In this Lesson, we will meet two other ways of dealing with the output of a chain of operations: storing the output under a name for later use and formatting a data frame into a table suited to human readers.

Manufacturing processes provide a helpful analogy for understanding the step-by-step structure of a computation. Simple

manufacturing processes might involve one or a handful of work steps arranged in a chain. Complex operations can involve many chains coming together in elaborate configurations. The video in Figure 26 shows the steps of pencil manufacturing. These involves several inputs:

The overall manufacturing process takes several inputs that are shaped and combined to produce the pencil: cedar wood slabs, glue, graphite, enamel paint. Each input is processed in a step-by-step manner. At some steps, two partially processed components are combined. For instance, there is a step that grooves the cedar slabs (which are sourced from another production line). The next step put glue in the grooves. In still another step, the graphite rods (which come from their own production process) are placed into the glue-filled grooves. (Lesson 7 introduces the data-wrangling process, join, that combines two inputs.)

There are several different forms of conveyors in the pencil manufacturing line that carry the materials from one manufacturing step to the next. We need only one type of conveyor—the **pipe**—to connect computing steps.

Manufacturing processes often involve storage or delivery. The video in Figure 26 ends before the final steps in the process: boxing the pencils, warehousing them, and the parts of the chain that deliver them to the consumer end-user. Storage, retrieval, and customer use all have their counterparts in computing processes. By default, the object produced by the computing chain is directly delivered to the customer, here by displaying it in some appropriate place, for instance directly under the computer command, or in a viewing panel or document:

```
Nats |>
  mutate(GDPpercap = GDP / pop) |>
  filter(GDPpercap > mean(GDPpercap), .by = year)
```

country	year	GDP	pop	GDPpercap
Korea	2020	874	32	27.31
France	2020	1203	55	21.87

<https://www.youtube.com/embed/qqs3xfmWr4>

Figure 26: Manufacturing a pencil, step by step.

Cuba	1950	60	8	7.5	
France	1950	250	40	6.25	

In our computer notation, the storage operation can look like this:

```
Nats |>
  mutate(GDPpercap = GDP / pop) |>
  filter(GDPpercap > mean(GDPpercap), .by=year) -> High_income_countries
```

At the very end of the pipeline chain, there is a **storage arrow** (\rightarrow , as opposed to the pipe, $|>$) followed by a **storage name** (`High_income_countries`). The effect is to place the object at the output end of the chain to be stored in computer memory in a location identified by the storage name.

Retrieval from storage is even simpler: just use the storage name as an input. For instance:

[Why do you say ...?](#)

```
High_income_countries |>
  select(-GDP, -pop) |>
  filter(year == 2020) |>
  kable(digits=2)
```

country	year	GDPpercap
Korea	2020	27.31
France	2020	21.87

i Pointing out storage from the start

In a previous example we placed the storage arrow (\rightarrow) at the end of a left-to-right chain of operations. In practice, programmers and authors prefer another arrangement—which we will use from now on in these *Lessons*—where the storage arrow is at the left end of the chain. The storage arrow still points to the storage name. For instance,

```
High_income_countries <- Nats |>
  mutate(GDPpercap = GDP / pop) |>
  filter(GDPpercap > mean(GDPpercap), .by=year)
```

Using this `storage_name <-` idiom it is easier to scan code for storage names and to spot when the output of the chain is to be delivered directly to the customer.

6.2 What's in a pipe?

The pipe—that is, `|>`—carries material from one operation to another. In computer-speak, the word “**object**” describes this material. That is, pipes convey objects.

Objects come in different “**types**.” Computer programmers learn to deal with dozens of object types. Fortunately, we can accomplish what we need in statistical computing with just a handful. You have already met two types:

1. data frames
2. graphics, consisting of one or more **layers**, e.g. the point plot as one layer and the annotations as another layer placed on top.

In later lessons, we will introduce two more types—(3) models and (4) simulations.

6.3 Pipes connect to functions

At the receiving end of a pipe is an operation on the object conveyed by the pipe. A better word for such an operation is “**function**.” It is easy to spot the functions in a pipeline: they **always** consist of a name—such as `summarize` or `point_plot`—followed directly by (and, eventually, a closing). For example, in `::: {.cell}`

```
Nats |>
  mutate(GDPpercap = GDP / pop) |>
  filter(GDPpercap > mean(GDPpercap), .by = year)
```

`:::` the first function is named `mutate`. The function output is being piped to a second function, named `filter`. From now on, whenever we name a function we will write the name followed

by () to remind the reader that the name refers to a function: so `mutate()` and `filter()`. There are other things that names can refer to. For instance, `Nats` at the start of the pipeline is a data frame, and `GDP`, `GDPpercap` and `pop`, and `year` are *variables*. Such names for non-functions are **never** followed directly by (.

i Example: What does `mean` refer to?

Another name appearing in the previous code block is `mean`. What kind of thing does this name refer to?

i Answer

Because the name is directly followed by a parentheses, we know `mean` must refer to a function.

Following our convention for writing function names, we should have written the name as `mean()`, but that would have made the question too easy!

6.4 Arguments (inside the parentheses)

Almost always when using a function the human writer of a computer expression needs to specify some details of how the function is to work. These details are **always** put inside the parentheses following the name of the function. To illustrate, consider the task of plotting the data in the `SAT` data frame. The *skeleton* of the computer command is

```
SAT |> point_plot()
```

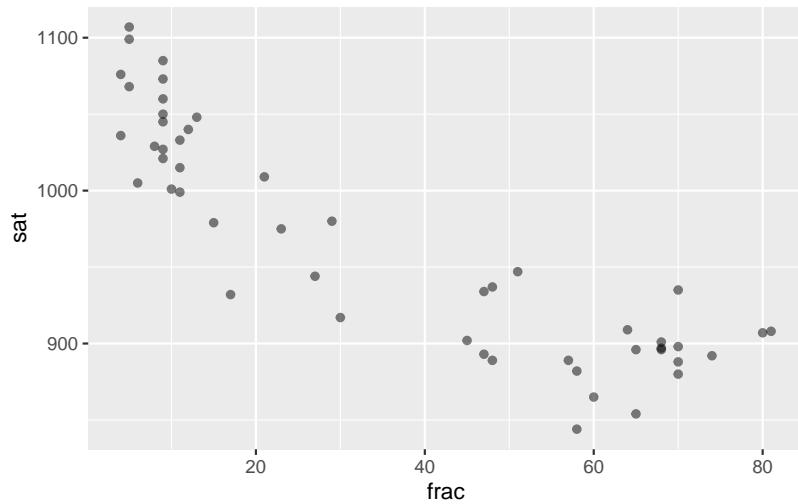
This skeleton is not a complete command, as becomes evident when the (incomplete) command is evaluated:

```
SAT |> point_plot()
```

```
Error in data_from_tilde(data, tilde): argument "tilde" is missing, with no default
```

What's missing from the erroneous command is a detail needed to complete the operation: What variables from `SAT` to map to `y` and `x`. This detail is provided to `point_plot()` as an argument. As you saw in Lesson 2, the argument is written as a **tilde expression**, for instance `sat ~ frac` to map `sat` to `y` and `frac` to `x`. Once we have constructed the appropriate argument for the task at hand, we place it inside the parentheses that follow the function name.

```
SAT |> point_plot(sat ~ frac)
```



Many functions have more than one argument. Some arguments, like the tilde expression argument to `point_plot()`, may be required. When an argument is not required, the argument itself is given a name and it will have a **default value**. In the case of `point_plot()`, there is a second argument named `annot=` to specify what kind of annotation layer to add on top of the point plot. The default value of `annot=` turns off the annotation layer.

Named arguments, like `annot=`, will **always** be followed by a single equal sign, followed by the value to which that argument is to be set. For instance, `point_plot()` allows four different values for `annot=`:

- i. the default (which turns off the annotation)

- ii. `annot = "violin"` specifying a density display annotation
- iii. `annot = "bw"` which creates a traditional “box-and-whiskers” display of distribution that we will not use much in these lessons.
- iv. `annot = "model"` which annotates with a graph of a model

In these Lessons, the single `=` sign always signifies a named argument.

A closely related use for `=` is to give a name to a calculated result from `mutate()` or `summarize()`. For instance, suppose you want to calculate the mean sat score and mean fraction in the `SAT` data frame. This is easy:

```
SAT |> summarize(mean(sat), mean(frac))
```

mean(sat)	mean(frac)
965.9	35.24

We will often use this *unnamed* style when the results are intended for the human reader. But if such a calculation fed down the pipeline to further calculations, it can be helpful to give simple names to the result. Frivolously, we’ll illustrate using the names `eel` and `fish`:

```
SAT |> summarize(eel = mean(sat), fish = mean(frac))
```

eel	fish
965.9	35.24

The reason for the frivolity here is to point out that *you* get to choose the names for the results calculated by `mutate()` and `summarize()`. Needless to say, it’s best to avoid frivolous or misleading names.

[Why do you say ...?](#)

6.5 Variable names in arguments

Many of the functions we use are on the receiving end of a pipe carrying a data frame. Examples, perhaps already familiar to you: `filter()`, `point_plot()`, `mutate()`, and so on.

A good analogy for a data frame is a shipping box. Inside the shipping box: one or more variables. When a function receives the `shipping_box` data frame, it opens it, providing access to each variable contained therein. In constructing arguments to the function, you do not have to think about the box, just the contents. You refer to the contents only by their names. `select()` provides a good example, since each argument can be simply the name of a variable, e.g.

For most uses, the arguments to a function will be an expressions constructed out of variable names. Some examples:

- `SAT |> filter(frac > 50)` where the argument checks whether each value of `frac` is greater than 50.
- `SAT |> mutate(efficiency = sat / expend)` where the argument gives a name (`efficiency`) to an arithmetic combination of `sat` and `expend`.
- `SAT |> point_plot(frac ~ expend)` where the argument to `point_plot()` is an expression involving both `frac` and `expend`.
- `SAT |> filter(expend > median(expend))` where the argument involves calculating the median expenditure across the state using the `median()` reduction function, then comparing the calculated median to the actual expenditure in each state. The overall effect is to remove any state with a below-median expenditure from the output of `filter()`.
- `SAT |> select(-state, -frac)` uses the `-` sign to exclude the variables from the output.



Quotation marks

Sometimes, you will see an argument written as letters and numbers *inside* quotation marks, as in `annot = "model"`. The quotation marks instruct the computer to take the

contents literally instead of pointing to a function or a variable. (In computer terminology, the content of the quotation marks is called a **character string**.)

The style of R commands does not use quotations around the names of objects, functions, and variables are **not** placed in quotations. When you see quotation marks in an example in these Lessons, take note. They are needed, for instance, in saying what kind of annotation should be drawn by `point_plot()`. If you forget to use the quotation marks where they are needed, the computer will signal an error:

```
Penguins |> point_plot(bill_length ~ flipper, annot = model)

Error in match.arg(annot): 'arg' must be NULL or a character vector
```

The error message is terse, but it gives hints; for example, '`arg`' suggests the error is about an argument, `annot` is the name of the problematic argument, and `character` is meant to point you to some issue involving character strings.

6.6 Styling with space

Written English uses space to separate words. It is helpful to the human reader to follow analogous forms in R commands.

- Use spaces around storage arrows and pipes: `x <- 7 |> sqrt()` reads better than `x<-7|>sqrt()`.
- Use spaces between an argument name and its value: `mutate(perc cap = GDP / pop)` rather than `mutate(perc cap=GDP/pop)`.
- When writing long pipelines, put a newline after the pipe symbol. You can see several instances of this in previous examples in this Lesson. DO NOT, however, start a line with a pipe symbol.

6.7 Displaying tables

We are using the word “**table**” to refer specifically to a printed display intended for a human reader, as opposed to data frames which, although often readable, are oriented around computer memory.

The readability of tabular content goes beyond placing the content in neatly aligned columns and rows to include the issue of the number of “significant digits” to present. All of the functions we use for statistical computations make use of internal hardware that deals with numbers to a precision of fifteen digits. Such precision is warranted for internal calculations, which often build on one another. But fifteen digits is much more than can be readily assimilated by the human reader. To see why, let’s display calculate yearly GDP growth (in percent) with all the digits that are carried along in internal calculations:

```
Growth_rate <- Nats |>
  pivot_wider(country,
    values_from = c(GDP, pop),
    names_from = year) |>
  mutate(yearly_growth =
    100.*((GDP_2020 / GDP_1950)^(1/70.)-1)) |>
  select(country, yearly_growth)
Growth_rate
```

country	yearly_growth
Korea	3.14547099309945
Cuba	0.411820047041944
France	2.26982406656688
India	1.87345150307259

...:

GDP, like many quantities, can be measured only approximately. It would be generous to ascribe a precision of about 1 part in 100 to GDP. Informally, this suggests that only the first two or three digits of a calculation based on GDP can have any real meaning.

Table 7: Annual growth in GDP from 1950 to 2020

	Growth rate (%)
Korea	3.1
Cuba	0.4
France	2.3
India	1.9

The problem of significant digits has two parts: 1) how many digits are worth displaying and 2) how to instruct the computer to display only that number of digits. Point (1) often depends on expert knowledge of a field. Point (2) is much more straightforward; use a computer function that controls the number of digits printed. There are many such functions. For simplicity, we focus on one widely used in the R community, `kable()`.

The purpose of `kable()` can be described in plain English: to format tabular output for the human reader. Whenever encountering a new function, you will want to find out what are the *inputs* and what is the *output*. The primary input to `kable()` is a data frame. Additional arguments, if any, specify details of the formatting, such as the number of digits to show. For instance:

```
Growth_rate |>
  kable(digits = 1,
        caption = "Annual growth in GDP from 1950 to 2020",
        col.names = c("", "Growth rate (%))")
```

The output of `kable()`, perhaps surprisingly, is **not** a data frame. Instead, the output is instructions intended for the display's typesetting facility. The typesetting instructions for web-browsers are often written in a special-purpose language called HTML. So far as these *Lessons* are concerned, is not important that you understand the HTML instructions. Even so, we show them to you to emphasize an important point: You can't use the output of `kable()` as the input to data-wrangling or graphics operation.

We will take a statistical view of the appropriate number of digits to show in Chapter 20.

```
<table>
<caption>Annual growth in GDP from 1950 to 2020</caption>
<thead>
<tr>
<th style="text-align:left;"> </th>
<th style="text-align:right;"> Growth rate (%) </th>
</tr>
</thead>
<tbody>
<tr>
<td style="text-align:left;"> Korea </td>
<td style="text-align:right;"> 3.1 </td>
</tr>
<tr>
<td style="text-align:left;"> Cuba </td>
<td style="text-align:right;"> 0.4 </td>
</tr>
<tr>
<td style="text-align:left;"> France </td>
<td style="text-align:right;"> 2.3 </td>
</tr>
<tr>
<td style="text-align:left;"> India </td>
<td style="text-align:right;"> 1.9 </td>
</tr>
</tbody>
</table>
```

7 Databases

As we move forward through these *Lessons*, an individual data frame will be the launching point for a statistical analysis or a graphical or tabular presentation. Inside every data frame, as you know, each row (that is, specimen) is an instance of the same unit of observation. But data science work often involves combining information about different kinds of unit of observation. For example, a health-care research project will presumably be based on patients: the corresponding data frame has a patient as the unit of observation and will include variables on date of birth, gender, and so on. If the project involves looking at doctor and clinic visits, there will be another data frame in which the unit of observation is a doctor/clinic. If medication is part of the project, there will be a data frame listing each patient's prescriptions and another data frame giving the characteristics of each drug substance. In the prescription data frame, there will be many rows that list the same drug, each such row rendered unique by the patient involved and the date of the prescription. Interested in studying the health consequences of previous illnesses? Then still another data frame will be needed to list each person's medical history, where the unit of observation is a bout of illness in an individual patient.

Suppose the project is to identify illnesses that might be side-effects of drugs. To evaluate a specific hypothesized drug-to-illness path, a basic question is whether those who took the drug are more likely to subsequently suffer the illness than the people who did not take the drug.

The data frame needed to answer this question might be simple: the unit of observation is a patient. The core variables will be (1) whether and when the patient got the illness and (2) whether and when the patient took the drug. As you will see in later Lessons, we can include in the analysis characteristics of each patient so that we can avoid, for instance, comparing elderly drug takers to young adults who never had the drug. This will entail including additional variables to the data frame, but the unit of observation will remain "a patient."

How do we construct the data frame described in the previous paragraph. We will need to combine the illness data frame, the

drug prescription data frame, the drug-substance data frame (to connect together drugs that belong to the same class of substances), and the patient data frame.

This *Lesson* is about how to combine data frames with different units of observation, and how to organize those multiple data frames so that they can easily be combined. The set of well-organized data frames is called a **database**.

Facility in using databases is a core professional skill for data scientists. For the statistical thinker, it is important to know the basics of how databases work so that she can call on data from multiple sources to inform the statistical questions asked.

7.1 *E pluribus unum*

The traditional national motto of the United States is *E pluribus unum*: “out of many, one.” The motto is embossed on coinage and printed on paper currency. It refers to the formation of a single country out of the thirteen original colonies. The historically-minded reader knows that the process of creating one country out of many colonies was difficult. On the political side, representatives from each of the thirteen met together in one body to debate, decide, and reconcile their differences.

With databases, the process—combining multiple data frames into a single one suited for statistical analysis—is much simpler. One reason is that there is no need for all the multiple data frames to meet all together simultaneously. Any combination of data frames can be constructed by a series of steps, each of which involves combining only two data frames at a time.

This Lesson introduces the generic process of combining two data frames with different units of observation. The Lesson also illustrates how to organize systems of data frames so that they can easily be combined into the myriad of forms needed to address the myriad of potential scientific and statistical questions.

A phrase from the *Declaration of Independence* describes this simultaneous as “in General Congress, Assembled.”

7.2 Join: putting tables together

To illustrate wrangling to join tables, we'll work with an authentic database in a familiar setting: student transcripts at a college. At many colleges, the person with authority over the database is called the "registrar." The registrar at one college gave permission to make parts of the database available to the general public so long as the published data is de-identified. This means, for example, that arbitrary codes are used for the names of students, faculty, and departments.

There are three data frames in the (simplified) database: **Grades**, **Sessions** and **Gradepoint**.

Here are a few randomly selected rows from the three data frames:

Consider the familiar student-by-student gradepoint average (GPA). This averages together each student's grades. The **Grades** tables store the grades, but we can't average categorical levels like "B+" or "C". To average, we need to convert each category to a number. This is done *via* the **Gradepoint** table.

The operation is conceptually simple. Add a new column to **Grades** that has the number. Work row-by-row through **Grades**, referring to the policy in **Gradepoint** to fill in the value of the new column for that row. Simple, but tedious!

The `left_join()` wrangling operation involves the two data frames to be combined. For each row in the "left" data frame, the corresponding information from the "right" data frame is added. Like this:

```
Grades |> left_join(Gradepoint)
```

```
Joining with `by = join_by(grade)`
```

Notice that student S31461 took session 2491 as a pass/fail class. He or she (we don't know which, because we don't have permission to publish the table giving such information for individual students) passed the course with a grade of S which doesn't count for student's gradepoint.

Grades

grade	sessionID	sid
A	session2606	S31440
S	session2491	S31461
A	session1904	S31461
A	session2606	S31869
A	session2044	S31905
A	session2491	S32028
A-	session3524	S32328
A	session2044	S32328

sid is the student ID, while **sessionID** identifies which course (in which semester) the student took. Students take multiple courses. For instance, student S32328 took sessions 2044, 2491, and 3524 (among others not listed). Student S31461 is listed twice, once for session 2491 and again for 1904. These two students had one course in common, session 2491. They may have sat next to each other! The same is true in session 2606 for students S31440 and S31869.

: : : : : : : : : : : : : :

Sessions

sessionID	iid	enroll	dept	level	sem
session2044	inst436	16	m	100	FA2001
session2491	inst170	34	n	200	FA2002
session2606	inst143	25	C	300	SP2003
session1904	inst264	26	M	100	SP2001
session3524	inst436	21	g	100	FA2004
session2911	inst268	10	M	300	FA2003
session3822	inst465	25	k	200	SP2005

Each session is taught by an instructor (**iid**), is associated with a department (**dept**). The number of students in that session (**enroll**) is listed, as is the semester in which the session was offered. The **level** indicates whether the course is directed to new students (level 100) or more advanced students (levels 200 and 300).

: : : : : : : : : : : : : :

Gradepoint

grade	gradepoint
AU	NA
S	NA
A	4.00
A-	3.66
B+	3.33
B	3.00
B-	2.66
C+	2.33
C	2.00
C-	1.66
D+	1.33
D	1.00
D-	0.66
NC	0.00

Gradepoint establishes the college's policy in converting letter grades to numbers. An A is translated to 4 gradepoints which NC (no credit) gets zero gradepoints. Pass-fail students who pass (S) don't have the course included in their gradepoint average. Similarly for students who are auditing (AU) the course.

grade	sessionID	sid	gradepoint
A-	session3524	S32328	3.66
A	session2044	S32328	4.00
A	session2491	S32028	4.00
A	session2606	S31869	4.00
S	session2491	S31461	NA
A	session1904	S31461	4.00
A	session2606	S31440	4.00
A	session2044	S31905	4.00
A	session3524	S31548	4.00
A	session2606	S32109	4.00
A	session2044	S31620	4.00
A	session2044	S31458	4.00
A-	session2044	S32205	3.66
A-	session3524	S32322	3.66
A-	session3524	S31506	3.66
A-	session2044	S32352	3.66
A-	session2491	S31827	3.66
A-	session3524	S31914	3.66
A-	session2044	S31914	3.66
A-	session2491	S31953	3.66
A-	session3524	S32373	3.66
A-	session2491	S31419	3.66
A-	session3524	S32406	3.66
B	session3524	S31197	3.00
B	session2911	S32418	3.00
B-	session2911	S32250	2.66
B+	session3524	S31833	3.33
B+	session2044	S32049	3.33
B+	session3524	S32025	3.33
C	session1904	S31194	2.00
S	session2491	S31791	NA
S	session3822	S31647	NA

Once `Gradepoint` has been joined to `Grades`, we can compute the GPA summary for each of the 443 students::

```
Grades |>
  left_join(Gradepoint) |>
  summarize(GPA = mean(gradept, na.rm = TRUE), .by = sid)
```

sid	GPA
S31461	3.94
S31869	3.56
S31440	3.76
S32328	3.52
S32028	3.55
S31905	3.88

In calculating the mean grade point, we've set `na.rm = TRUE` meaning to remove any `NA` values before computing the mean. To judge from the GPA, student S31461 strategically decided to preserve their high GPA by taking a risky course pass/fail.

i Case study: What about the instructor?

Students will be sympathetic to the claim that some instructors are harder grading than others. This makes a student-by-student GPA an unreliable indicator of a student's performance.

Knowing how easy it is to join data frames ... Let's try something different. We can calculate a grade point average for each instructor! This will involve joining the `Grades` and `Sessions` data frames in order to place the instructor's ID next to each of the grades he or she gave out. Join this combined table with `Gradepoint` to get the numerical value of the grade, then average across instructors. We will also keep track of how many students were taught by the instructor.

```
Instructors <- Grades |>
  left_join(Sessions) |>
  left_join(Gradepoint) |>
  summarize(iGPA = mean(grade, na.rm = TRUE,
                        nstudents = sum(enroll, na.rm = TRUE)), .by = iid)
Instructors
```

iid	iGPA
inst143	3.76
inst198	2.99
inst263	2.85
inst501	3.85
inst269	2.72
inst411	3.01
inst459	3.74
inst419	2.95

8 Statistical thinking & variation

The central object of statistical thinking is **variation**. Whenever we consider more than one item of the same kind, [Expressed in terms of data frames, the “same kind” means the unit of observation, the type of specimen that occupies a row of a data frame.]{.aside from}, the items will likely differ from one another. Sometimes, the differences appear slight, as with the variation among fish in a school. Sometimes the differences are large, as with country-to-country variation in population size, land area, or economic production. Even carbon atoms—all made of identical protons, neutrons, and electrons—differ one from another in terms of energy state, bonds to molecular neighbors, and so on.

Sometimes variation is desirable. (How boring it would be to have all students the same!) Sometime the goal is to avoid variation, as with precision-made, interchangeable components intended for assembly-line production. Lack of variation—interchangeability—became an industrial concept around 1800 with the manufacture of guns. Perhaps it’s obvious that achieving interchangeability requires ways to measure with precision, to detect even the smallest differences between a standard part and the parts in production.

In the same era, the increased use of time-keeping in navigation and the need to make precise land surveys across large distances led to detailed astronomical observations. The measurements of the positions and alignment of stars and planets of different observatories were slightly inconsistent even when taken simultaneously. Such inconsistencies were deemed to be the result of error. Consequently, the “true” position and alignment was considered to be that offered by the most esteemed, prestigious, and authoritative observatory.

After 1800, attitudes began to change—slowly. Rather than referring to a single observation as best, astronomers and surveyors used arithmetic to construct an artificial summary by averaging the varying individual observations. The average—typically the arithmetic mean—is called an “**estimate**” reflecting an understanding that the summary itself is imperfect. The

estimate might differ from each of the actual observations, but still was taken as the more authoritative than any of them.

By way of political analogy, in the pre-1800 system, the esteemed observatory's observation was the king or queen and carried absolute authority. The post-1800 system of averages and estimates was more like a democratic process, where the voice of each observation had equal weight in the outcome. This post-1800 democratic conception is still with us; elementary school students learn how to average at the same time they learn about elections and voting.

Up through the early 1900s, words like "error" and "deviation" were the names given to the differences between individual observations and the summary estimate. However, as statistical summaries spread rapidly from one field to another, statisticians had to confront the inconvenient fact that summaries could contain errors.

This convenience might have first become evident when the statistical summaries of different groups—Frenchmen and Englishmen—were compared. If the summaries were exact, a simple numerical comparison would suffice to establish the differences. However, since summaries are not infinitely precise, it is essential to consider their imprecision in making judgments of difference.

The challenge for statistics students is to overcome years of training suggesting wrongly that you can compare groups by comparing averages. The averages themselves are not sufficient. Statistical thinking is based on the idea that **variation** is an essential component of comparison. Comparing averages can be misleading without considering the specimen-to-specimen variation simultaneously.

As you learn to think statistically, it will help to have a concise definition. The following captures much of the essence of statistical thinking:

Statistic thinking is the accounting for variation in the context of what remains unaccounted for.

As we start, the previous sentence may be obscure. It will begin to make more and more sense as you work through these successive *Lessons* where, among other things, you will ...

1. Learn how to measure variation;
2. Learn how to account for variation;
3. Learn how to measure what remains unaccounted for.

8.1 Measuring variation

Yet another style for describing variation—one that will take primary place in these Lessons—uses only a **single-number**. Perhaps the simplest way to imagine how a *single* number can capture variation is to think about the numerical *difference* between the top and bottom of an interval description. We are throwing out some information in taking such a distance as the measure of variation. Taken together, the top and bottom of the interval describe two things: the *location* of the values and how different the values are from one another. These are both important, but it is the difference between values that gives a pure description of variation.

Early pioneers of statistics took some time to agree on a standard way of measuring variation. For instance, should it be the distance between the top and bottom of a 50% interval, or should an 80% interval be used, or something else? Ultimately, the selected standard is not about an interval but something more fundamental: the distances between pairs of individual values.

To illustrate, suppose the `gestation` variable had only two entries, say, 267 and 293 days. The *difference* between these is $267 - 293 = -26$ days. Of course, we don't intend to measure *distance* with a negative number. One solution is to use the absolute value of the difference. However, for subtle mathematical reasons relating to the Pythagorean theorem, we avoid negative numbers by using the *square of the difference*, that is, $(293 - 267)^2 = 676$ days-squared.

To extend this straightforward measure of variation to data with $n > 2$ is simple: look at the square difference between every possible pair of values, then average. For instance, for

Instructors will bring their previous understanding of the measurement of variation to this section. They will likely be bemused by the presentation here. First, this Lesson gives prime billing to the “**variance**” (rather than the “**standard deviation**”). Second, the calculation will be done in an unconventional way.

There are three solid reasons for the departure from the convention. I recognize that the usual formula is the correct, computationally efficient algorithm for measuring variation. That algorithm is usually presented algebraically, even though many students do not parse algebraic notation of such complexity:

$$s \equiv \sqrt{\frac{1}{n-1} \sum_i (x_i - \bar{x})^2} .$$

The first step in the conventional calculation of the standard deviation s is to find the mean value of x , that is

$$\bar{x} = \frac{1}{n} \sum_i x_i$$

For those students who can parse the formulas, the clear implication is that the standard deviation depends on the mean.

The mean and the variance (or its square root, the standard deviation) are independent. Each can take on any value at all without changing the other. The mean and the variance measure two utterly distinct characteristics. The method shown in the text avoids making the misleading link between the mean and the variance.

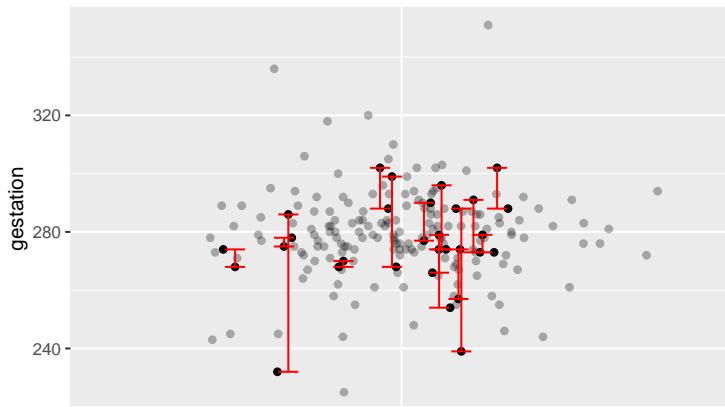
As well, the text's formulation avoids any need to introduce the distracting $n - 1$. The effect of the

$n = 3$ with values 267, 293, 284, look at the differences $(267 - 293)^2$, $(267 - 284)^2$ and $(293 - 284)^2$ and average them! This simple way of measuring variation is called the “modulus” and dates from at least 1885. Since then, statisticians have standardized on a closely related measure, the “**variance**,” which is the modulus divided by 2. Either one would have been fine, but honoring convention offers important advantages; like the rest of the world of statistics, we’ll use the variance to measure variation.

i Variance as pairwise-differences

Figure 27 is a jitter plot of the `gestation` duration variable from the `Gestation` data frame. The graph has no explanatory variable because we are focusing on just one variable: `gestation`. The range in the values of `gestation` runs from just over 220 days to just under 360 days.

Each red line in Figure 27 connects two randomly selected values from the variable. Some lines are short; the values are pretty close (in vertical offset). Some of the lines are long; the values differ substantially.



Only a few pairs of points have been connected with the red lines. To connect every possible pair of points would fill the graph with so many lines that it would be impossible to see that each line connects a pair of values.

The average square of the lines’ lengths (in the vertical di-

Figure 27: Values of `gestation` duration (days) from the `Gestation` data frame. For every pair of dots, there is a vertical distance between them. To illustrate, a handful of pair have been randomly selected and their vertical difference annotated with a red line. The “modulus” is the average squared pairwise vertical difference, where the average is taken over all possible pairs (not just the ones annotated in red). The variance is the modulus divided by 2.

rection) is called the “modulus.” We won’t use this word going forward in these Lessons; we accept that the conventional description of variation is the “variance.” Still, the modulus has a more natural explanation than the variance. Numerically, the variance is half the modulus.

Calculating the variance is straightforward using the `var()` function. Remember, `var()` is similar to the other reduction functions—e.g. `mean()` and `median()`—that distill multiple values into a single number. As always, the reduction functions need to be used *within* the arguments of a data wrangling function. of a set of data-frame rows to a single summary is accomplished with the `summarize()` wrangling command.

```
Gestation |>
  summarize(var(gestation, na.rm = TRUE))
```

```
| var(gestation, na.rm = TRUE) |
|-----|
|      256.9      |
```

A consequence of the use of squaring in defining the variance is the units of the result. `gestation` is measured in days, so `var(gestation)` is measured in days-squared.

i From variance to “standard deviation”

If you have studied statistics before, you have probably encountered the “**standard deviation**.” We avoid this terminology; it is long-winded and wrongly suggests a departure from the normal. Calculating the standard deviation involves two steps: first, find the variance then take the square root.

```
Gestation |>
  summarize(sd = sqrt(var(gestation, na.rm = TRUE)))
```

9 Accounting for variation

Lesson 8 listed three foundations for statistical thinking and provided instruction on the first one:

1. How to measure the amount of variation

In this Lesson, we will take on two and three:

2. Accounting for variation
3. Measuring what remains unaccounted for

In business, accounting keeps track of money and value: assets, receipts, expenditures, etc. Likewise, in statistics, accounting for variation is the process of keeping track of variation. At its most basic, accounting for variation splits a variable into components: those parts which can be attributed to one or more other variables and the remainder which cannot be so attributed.

The variable being split up is called the “**response variable**.” The human modeler chooses a response variable, depending on her purpose for undertaking the work. The modeler also chooses *explanatory variables* that may account for some of the variability in the response variable. There is almost always something left over, variation that cannot be attributed to the explanatory variables. This left-over part is called the “**residual**.”

To illustrate, we’ll return to a historical statistical landmark, the data collected in the 1880s by Francis Galton on the heights of parents and their full-grown children. In that era, there was much excitement, uncertainty, and controversy about Charles Darwin’s *theory of evolution*. Today, we understand the role of DNA in encoding genetic information. But in Galton’s day, the concept of “gene” was unknown.

Galton’s overall project was to quantify the heritability of traits (such as height) from parents to offspring. Darwin had famously measured the beaks of finches on the Galapagos Islands. Galton, in London, worked with the most readily available local

The word “account” has several related meanings. These definitions are drawn from the Oxford Languages dictionaries.

- To “account for something” means “to be the explanation or cause of something.” [Oxford Languages]
- An “account of something” is a story, a description, or an explanation, as in the Biblical account of the creation of the world.
- To “take account of something” means “to consider particular facts, circumstances, etc. when making a decision about something.”

Synonyms for “account” include “description,” “report,” “version,” “story,” “statement,” “explanation,” “interpretation,” “sketch,” and “portrayal.” “Accountants” and their “account books” keep track of where money comes from and goes to.

These various nuances of meaning, from a simple arithmetical tallying up to an interpretative story serve the purposes of statistical thinking well. When we “account for variation,” we are telling a story that tries to explain where the variation might have come from.

Although the arithmetic used in the accounting is correct, the story is not necessarily definitive. For example, if a full-grown child in the Galton data frame, the variables recorded were the height of the child, in inches, the heights of the mother and father and the sex of the child according to the same conventions of Victorian England, where the data were collected.

species: humans. Height is easy and socially acceptable to measure, a good candidate for data collection.

Everyone can see that height varies from person to person. Galton sought to divide the height variation in the children into two parts: that which could be attributed to the parents and that which remained unaccounted for, which we call the “**residual**.” We will start with a mathematically more straightforward project: dividing the variation in the children’s heights into that part associated with the child’s sex and the residual.

As introduced in Lesson 5, we can calculate the average height for females and males using the `summarize()` wrangling verb with a `.by = sex` argument to create a separate summary for each level of `sex`.

```
Galton |>
  summarize(mheight = mean(height), .by = sex)
```

This simple report indicates that the sexes differ, on average, by about 5 inches in height. But this report offers no insight into how much of the person-to-person variation in height is attributable to `sex`. For that, we need to look at the group averages in the context of the individuals. To that end, we will use `mutate()` to assign to each individual a “**model value**” that depends only on `sex`.

```
Model1 <- Galton |>
  mutate(modval = mean(height), .by = sex)
```

Figure 28 shows `?@tbl-galton-model1c` graphically. Some things to notice:

1. The model values are different for males and females.
2. Among the females the model values do *not* vary with `mother` or `father`. The same is true for the males.
3. The model values are not the same as the individual heights.

The numerical difference between each specimen’s height and its model value is called the “**residual**” for that specimen. This is easy to calculate:

Galton *invented* the methods we are going to use in this example.

We often use `summarize()` with reduction functions like `mean()`. Summarize gives one row of output for each of the groups defined by the `.by` argument. Here we are using `mutate()`. The mean will still be calculated on a group-by-group basis, but with `mutate()` the result will contain all the original rows. Each row of a group will get the value averaged over all the rows in that group.

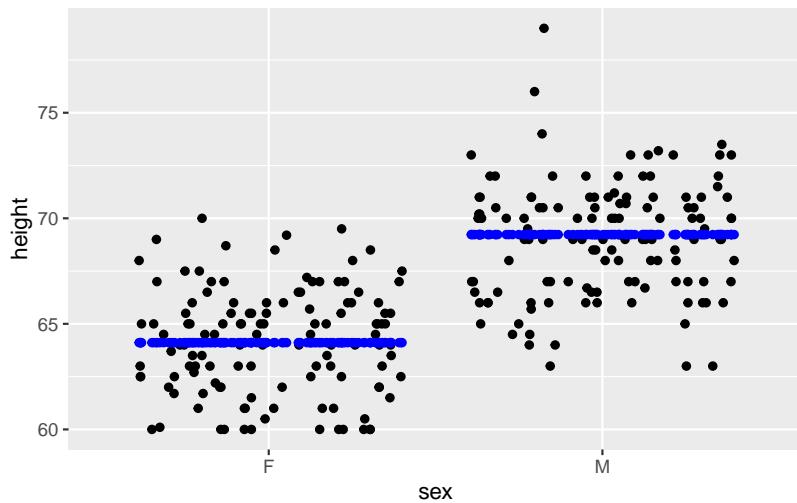


Figure 28: The response variable (height) and the model values from `?@tbl-galton-model1c` plotted versus `sex`. The model values are the same within each sex for all individuals.

```
Model1 <- Model1 |>
  mutate(resid = height - modval)
```

The residuals for the model of `height` versus `sex` are shown in Table ???. Notice that some residuals are positive, meaning that the actual `height` is larger than the model value. Other residuals, when the actual `height` is below the model value, are negative.

As always, the variance is our preferred measure of the amount of variation. There are three variances that are connected together:

- i. The variance of the response variable.
- ii. The variance of the model values.
- iii. The variance (across all specimens) of the specimens' residuals.

Comparing those three variances, we see that the largest one is the variance of the response variable (`height`). This will always be the case.

Note to instructors

```
Model1 |>
  summarize(var(height), var(modval), var(resid))
```

var(height)	var(modval)	var(resid)
12.84	6.549	6.288

Remarkably, the variance of the model values and the variance of the residuals add up precisely to equal the variance of the response variable. In other words, the variation in the response variable is split into two parts: the variation associated with the explanatory variable and the remaining (“residual”) variation.

9.1 Numerical explanatory variables

The previous section used the categorical variable `sex` as the explanatory variable. Galton’s interest, however, was in the relationship between children’s and parents’ height.

`sex` is a *categorical* explanatory variable. Using `mutate(..., .by = sex)` is a good method of handling categorical explanatory variables. However, this does not work for *quantitative* explanatory variables. The reason is that `.by` translates a quantitative variable into a categorical one, with a result for each unique quantitative value: It’s trivial to substitute the height of the `mother` or `father` in place of `sex` in the method introduced in the previous section. However, as we shall see, the results are not satisfactory. Galton’s key discovery was the proper method for relating two *quantitative* variables such as `height` and `mother`.

First, let’s try simply substituting in `mother` as the explanatory variable and using `mean()` to create the model values.

```
Model2 <- Galton |>
  mutate(modval = mean(height),
        resid = height - modval,
        .by = mother)
Model2
```

mother	sex	modval	resid
58	M	67	5.4
58	F	67	-4.1
58	M	67	1.4
58	M	67	2.4
64	M	66	2
66	M	68	5.4
67	M	67	6.3
67	F	67	2.1

The problem is that the model values are not a simple function of `mother`, as seen in Figure 29.

```
Model2 |>
  gf_point(height ~ mother, point_ink = 0.1) |>
  gf_point(modval ~ mother, color="blue", point_ink = 0.3) |>
  gf_line(modval ~ mother, color="blue", linewidth=0.5, alpha = 0.5 ) |>
  gf_rect(62 + 73 ~ 60.25 + 61.75, fill=NA, color = "orange")
```

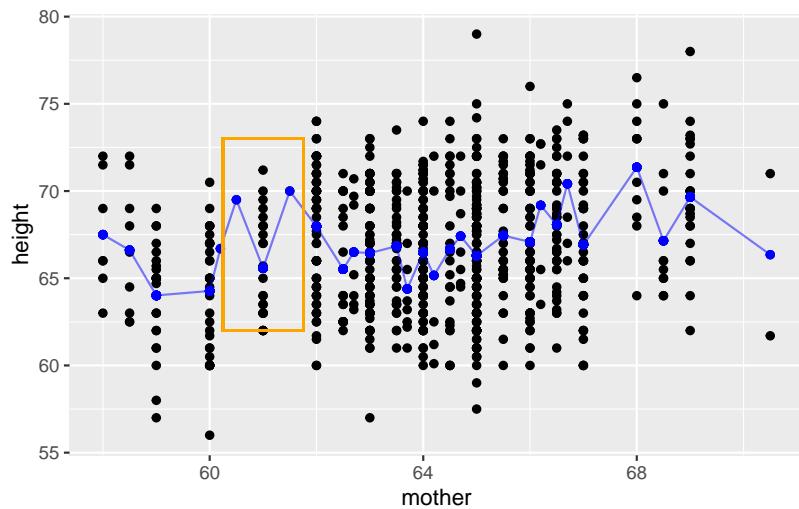


Figure 29: Modeling `height` of the child by the mother's height using `group_by(mother)` and `modval=mean(height)`. It's hard to see any pattern in the model values. A thin line has been added connecting adjacent model values to highlight how unsatisfactory the model is.

It is common sense that the model linking mothers' heights to children's height should be **smooth**, not the jagged pattern seen in Figure 29. The source of the jaggedness is the use of `.by = mother` in `mutate(modval = mean(height), .by = mother)`. The calculation of the mean is done separately and independently for each of the vertical columns of points in Figure 29. To illustrate, compare the model values for 60.5-inch mothers, 61-inch mothers, and 61.5-inch mothers (highlighted by the orange box). In a smooth relationship, for example, the model value for 61.0-inch mothers should be about halfway between the model values for 60.5- and 61.5-inch mothers. Instead, the `.by=mother` model produces a model value for 61.0-inch mothers that is lower than the model values on either side of it. In a smooth model, the model value for mothers of middle height should be somewhere in between the model values for short and for tall mothers.

The solution to the problem of jagged model values is to avoid the absolute splitting into non-overlapping groups by mother's height. Instead, we want to find a smooth relationship. Galton invented the method for accomplishing this. A modern form of his method is provided by the `model_values()` function, which we shall use to construct Model3.

```
Model3 <- Galton |>
  mutate(modval = model_values(height ~ mother),
        resid = height - modval)
```

Notice that the `.by = mother` step has been entirely removed. Notice also that `model_values()` uses the same kind of **tilde expression** as we have employed when plotting. The response

variable is listed on the left of the , the explanatory variable on the right side. In other words, we are modeling the child's `height` as a function of mother's height.

```
Model3 |>
  gf_point(height ~ mother, point_ink = 0.1) |>
  gf_point(modval ~ mother, color="blue", point_ink = 0.3) |>
  gf_line(modval ~ mother, color="blue", linewidth=0.5) |>
  gf_rect(62 + 73 ~ 60.25 + 61.75, fill=NA, color = "orange")
```

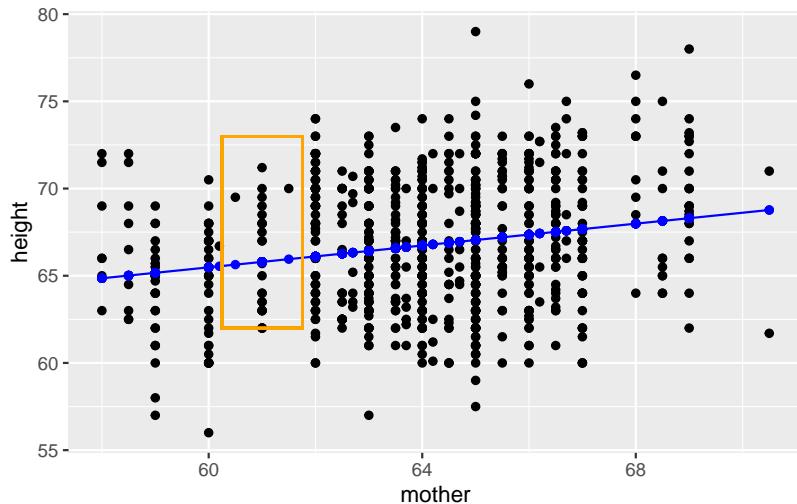


Figure 30: A smooth model of `height` with respect to `mother` created with `model_values(height ~ sex)`.

As always, modeling splits the variance of the response variable into two parts, one associated with the explanatory variable and the other holding what's left over: the residual. Here's the split for Model3 which uses `mother` as an explanatory variable:

```
Model3 |>
  summarize(var(height), var(modval), var(resid))
```

var(height)	var(modval)	var(resid)
12.84	0.52	12.32

9.2 Multiple explanatory variables

The models we work with in these Lessons *always* have exactly one response variable. But models can have any number of *explanatory* variables.

Whatever the number of explanatory variables and however many levels a categorical explanatory variable has the model splits the variance of the response into two complementary pieces: the variance accounted for by the explanatory variables

Note that the idea of “response” and “explanatory” variables refers to a model and are not at all intrinsic to a bare data frame. A data frame can contain many variables, any of which can be used as explanatory variables. The choice of response variable depends on the modeler’s goals.

and the part not accounted for, that is, the *residual* variance.

To illustrate, here is a sequence of models of `height` with different numbers of explanatory variables.

Three explanatory variables

```
### / column: margin
Galton |>
  mutate(modval = model_values(height ~ sex + mother +
    resid = height - modval) |>
  summarize(var(height), var(modval), var(resid)) + father),
```

var(height)	var(modval)	var(resid)
12.84	8.212	4.626

Many statistical terms mean something different in statistical than in everyday use. “Residual” is a pleasant exception: the statistical meaning is closely matched by its everyday dictionary definition.

Two explanatory variables

```
### / column: margin
Galton |>
  mutate(modval = model_values(height ~ sex + mother),
    resid = height - modval) |>
  summarize(var(height), var(modval), var(resid))
```

var(height)	var(modval)	var(resid)
12.84	7.212	5.625

One explanatory variable

```
### / column: margin
Galton |>
  mutate(modval = model_values(height ~ sex),
    resid = height - modval) |>
  summarize(var(height), var(modval), var(resid))
```

var(height)	var(modval)	var(resid)
12.84	6.549	6.288

Zero explanatory variables

```
### / column: margin
# Galton |>
#   mutate(modval = model_values(height ~ 1),
#         resid = height - modval) |>
#   summarize(var(height), var(modval), var(resid))
```

var(height)	var(modval)	var(resid)
12.84	0	12.84

9.3 Comparing models with R²

When selecting explanatory variables, comparing two or more different models sharing the same response variable is often helpful: a simple model and a model that adds one or more explanatory variables to the simple model. The model with *no explanatory variables*, is always the simplest possible model. For example, in Section 9.2, the model `height ~ 1` is the simplest. Compared to the simplest model, the model `height ~ sex` has one additional explanatory variable, `sex`. Similarly, `height ~ sex + mother` has one additional explanatory variable compared to `height ~ sex`, and `height ~ sex + mother + father` adds in still another explanatory variable.

The simpler model is said to be “**nested in**” the more extensive model, analogous to a series of [Matroshka dolls](#). A simple measure of how much of the response variance is accounted for by the explanatory variables is the ratio of the variance of the model values divided by the variance of the response variable itself. This ratio is called “R²”, pronounced “R-squared.”

For instance, R² for the model `height ~ sex + mother` is

$$R^2 = \frac{7.21}{12.84} = 0.56$$



A sequence of five nested Matroshka dolls. Each smaller doll fits inside a larger one.

By comparison, R^2 for the simpler model, `height ~ sex`, is slightly smaller:

$$R^2 = \frac{6.55}{12.84} = 0.51$$

For all models, $0 \leq R^2 \leq 1$. It is tempting to believe that the “best” model in a set of nested models is the one with the highest R^2 , but statistical thinkers understand that “best” ought to depend on the purpose for which the model is being built. This matter will be a major theme in the remaining Lessons.

R^2 is also known as the “coefficient of determination,” a little-used term we shall avoid. Still, it’s worth noting the attitude behind the term; it quantifies the extent to which the response variable is “determined” by Instructor Note: Strictly speaking, the explanatory variables. “all” should be qualified to mean “linear least-squares models with an intercept term.”

10 Model patterns

In these Lessons, we build many models using many different data sources. Notably, we often build multiple models from one data frame. These models use the *same response variable* but different explanatory variables. This enables us to compare different ways of explaining the variation in the response variable.

All of the models will have certain features in common. This Lesson points out those commonalities so that you can “read” a new model with understanding.

10.1 Data and patterns: a painterly metaphor

Figure 31 is a painting of a harbor scene in Istanbul. It’s a rich composition intended for the human eye. There is water, a dozen boats, and a mosque—the Hagia Sophia—in the background.



Figure 31: Paul Signac, *La Corne d’Or*, 1907

It’s tempting to hope that statistical techniques could identify complex patterns like those we see in Figure 31. That task might barely be possible by the most up-to-date forms of artificial intelligence. Statistical modeling, however, is intended to look for much simpler patterns.

To emphasize just how simple such patterns are, ?@fig-paint-strokes zooms in on a tiny area of the painting, containing just a handful of paint strokes.

It goes without saying that those few strokes give no hint about what is going on in the whole painting. We don’t expect them to.

Statistical modeling looks only for very general kinds of patterns. Not a harbor, not a boat, not even a mast or pennant. Much simpler, and much more general.

?@fig-paint-edge, which shows a much larger part of the painting than ?@fig-paint-strokes, illustrates one kind of simple, general pattern: a boundary or “edge.” The edge here is between the bright zone on the lower left and the darker zone on the top right. A statistical model would be able to confirm that there is such an edge and give a little more detail: the orientation of the edge and which side is bright and which side dark.

There is nothing in ?@fig-paint-edge to discern that the edge in question is between the reflection of the sky and the prow of a boat. It’s just an abstract edge.

This Lesson describes how we specify to the computer the kind of simple, general pattern we are looking for in data. It also shows the “shapes” of those patterns not as painting-like image but as a model annotation in a point plot.

10.2 The model specification

Two basic inputs go into constructing a model:

1. A data frame.
2. The **model specification**, which declares which column from the data frame will be the response variable and which other column(s) will be the explanatory variable(s).

For directing the computer, we write the model specification as a **tilde expression**: the name of the response variable goes to

the left of the . The name of the explanatory variable is on the right.

When there is more than one explanatory variable, their names

all go on the right side of  separated by the + symbol which stands for the English word “and” rather than an sum in the arithmetic sense.

From time to time, we refer to models with *no explanatory variables*. In such models, a simple 1 goes to the right of the

Occasionally, we will use the * symbol instead of + for reasons that will be pointed out whenever we come to such a situation. We will also sometimes use mathematical functions such as `log()` or `ns()` in the model specification.

 . The reasons for doing this require some explanation, which will be provided in later Lessons.

10.3 “Shapes” of models

Although the response variable in a regression model is always quantitative, explanatory variables can be either quantitative or categorical. Regression models may sometimes involve tens or thousands of explanatory variables in professional work. Almost all the models used in these Lessons will have one or two explanatory variables (and, occasionally, zero explanatory variables). This suffices for introducing the concepts and methods of statistical thinking.

It is convenient to think of the various combinations of explanatory variables in terms of the “shape” of a graph of the model. There are two basic shapes for models with a single explanatory variable: one shape when the explanatory variable is *categorical* and another shape when the explanatory variable is quantitative.

We illustrate with the `CPS85` data frame. `CPS85` records a small survey of workers’ `wages` (in 1985) and includes both numerical and categorical variables. The unit of observation is an individual worker. The categorical variable `sector` records the type of each worker’s job; levels for `sector` include clerical, manufacturing, sales, service, etc.

In the following subsections, we compare the shapes of several models, all of which use `wage` as the response variable.

10.3.1 One explanatory variable

First, consider models with a single explanatory variable. When that explanatory variable is categorical, the model shape consists of potentially different values for each level of the explanatory variable. Figure 32 shows two examples:

When the explanatory variable is quantitative, the model values are arrayed on a smooth curve, as in Figure 33.

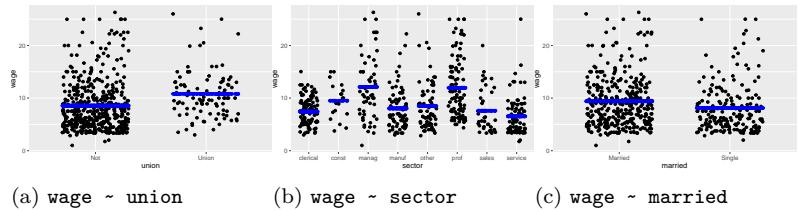


Figure 32: Examples of regression models with a single categorical explanatory variable.

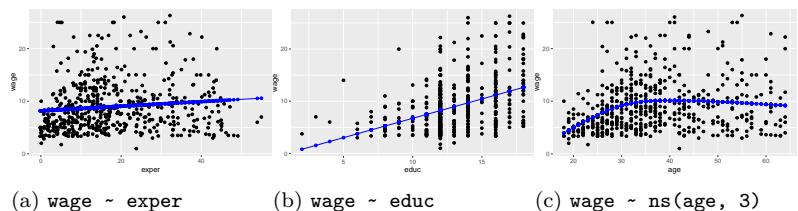


Figure 33: Examples of regression models with a single quantitative explanatory variable.

10.3.2 Two explanatory variables

Explanatory variables can be either quantitative or categorical. With two explanatory variables, one is mapped to x and the other to color. Given that the response variable is always mapped to y, there are four combinations possible, each of which has a distinctive graphical format:

Example	Horizontal axis (x)	Color
Figure 34	categorical	categorical
Figure 37	categorical	quantitative
Figure 38	quantitative	categorical
Figure 39	quantitative	quantitative

Two categorical explanatory variables

```
Whickham |>
  point_plot(age ~ smoker + outcome, annot="model",
             point_ink = 0.05, model_ink=0.7)
```

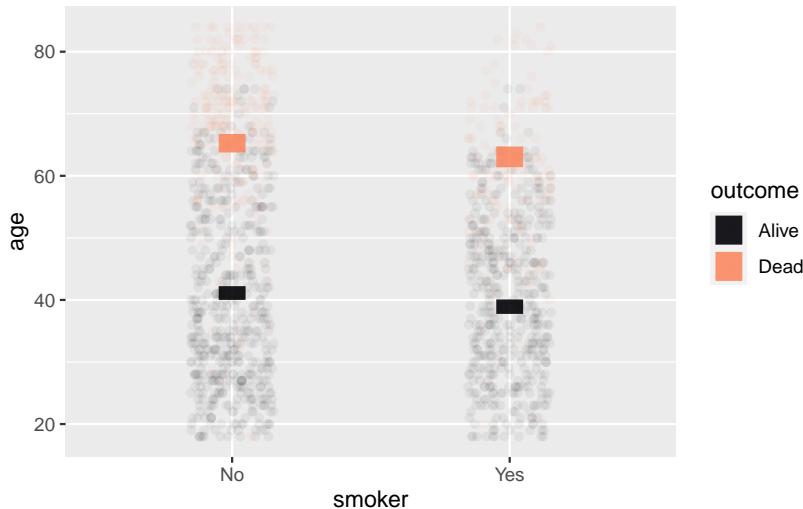


Figure 34: `age ~ smoker + outcome`

This example shows data from a survey of female voters in the UK. Each voter's age and smoking status were recorded at an initial interview. The interview was followed up 20 years

later, at which point some of the original interviewees were dead and others still living, recorded in the variable `outcome`. Unsurprisingly, the older interviewees were much more likely to have died during the 20-year follow-up. The model values show the difference in mean ages between the smokers and non-smokers separately for the survivors and non-survivors. With two categorical variables, each with two levels, there are four distinct model values.

Categorical & quantitative

This example shows (full-grown) child's height as a function of the child's sex and his or her mother's height.

```
# The code version to appear in the text
Galton |>
  point_plot(height ~ sex + mother, point_ink = 0.2)
Galton |>
  mutate(modval = model_values(height ~ sex + mother)) |>
  point_plot(modval ~ sex + mother) |>
  gf_lims(y = c(55, 80))
```

Reading such a graph takes patience. We've tried to help by separating the data and model-value layers. In the data layer, you can easily see that some males are taller than almost all females, and some females are shorter than nearly all males. The model layer strips away the residuals, producing a discernible pattern: the shorter children of either sex tend to have shorter mothers (black) and that taller children of each sex tend to have taller mothers (orange).

The model-value layer shows the extent of the relationship between mother's and child's height more clearly. (This is exactly what models are supposed to do!) You can see that the model values differ for children of the shortest mothers and of the tallest mothers. The difference is about 3 inches of child's height.

The model values are faithful to the data, but leave out the residuals. The raw data include the residuals. The non-zero size of residuals means that children of the shortest mothers differ in height from the model values. Similarly for the children of

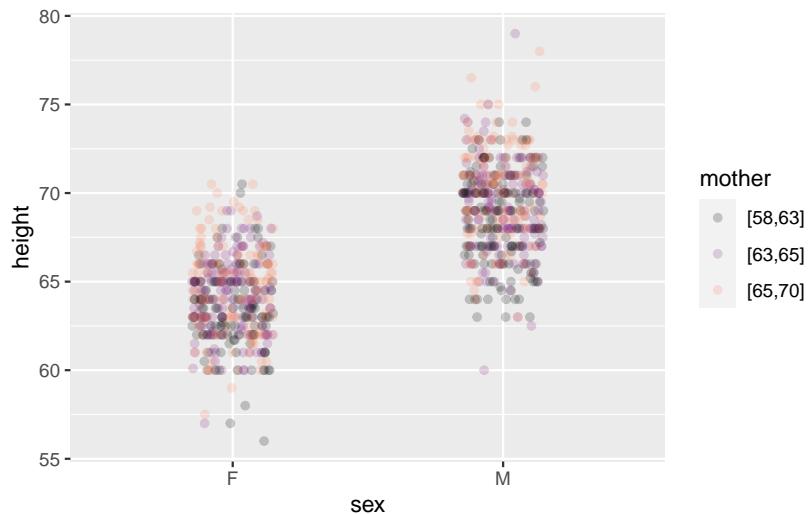


Figure 35: Data layer

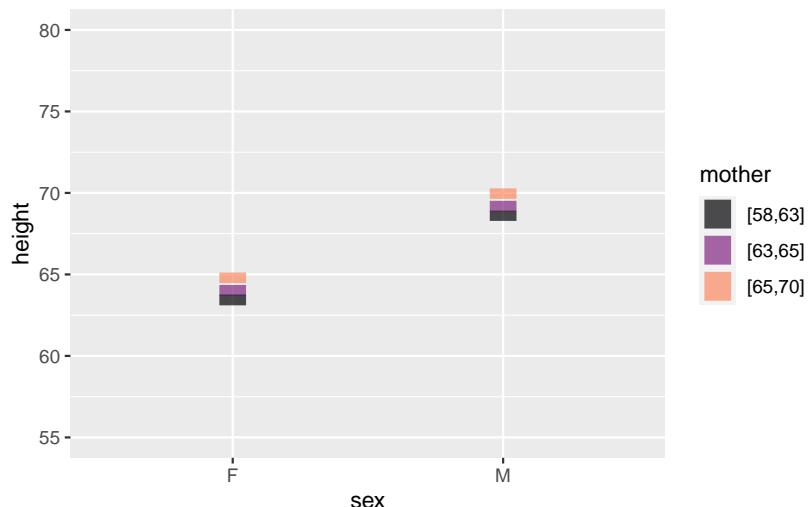


Figure 36: Model-value layer

Figure 37: `height ~ sex + mother`

the tallest mothers. The result is, in the raw data, that some children of the shorter mothers are in fact taller than some children of the taller mothers. The model values, by stripping away the residual child-to-child differences, make the trends easier to see.

Quantitative & categorical

This example shows the same data and model as the previous example. The only difference is that the quantitative explanatory variable is mapped to `x` while the categorical explanatory variable is mapped to `color`.

Point for point, the model values in Figure 38 are the same as in Figure 37. But the new arrangement spreads them out differently in space. In Figure 38 the model values are organized along two straight lines, one for each sex. The **slope** of the lines indicates the relationship between mother's and child's heights. The vertical offset between the lines is the difference in model values for the two sexes.

Figure 38 is easier to read than Figure 37. This illustrates a simple principle for effective graphics: When a model has one quantitative and one categorical explanatory variable, map the quantitative variable to the horizontal axis.

```
Galton |>
  point_plot(height ~ mother + sex, annot="model",
             point_ink = 0.1, model_ink=0.7)
```

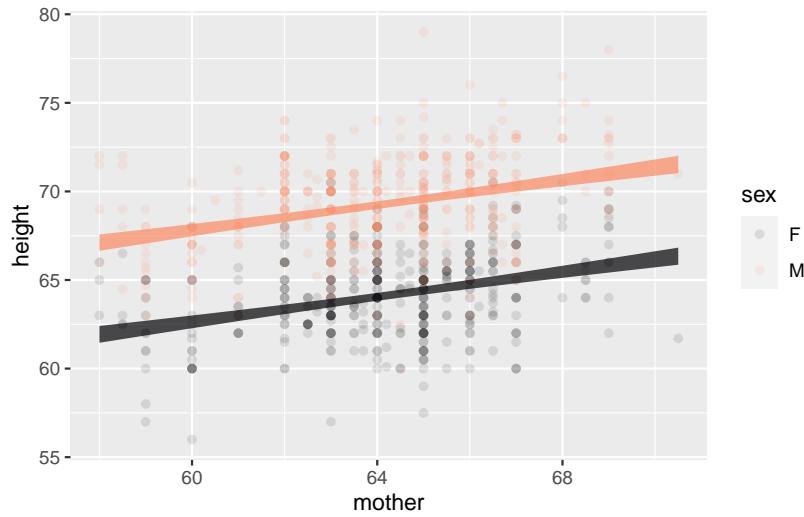
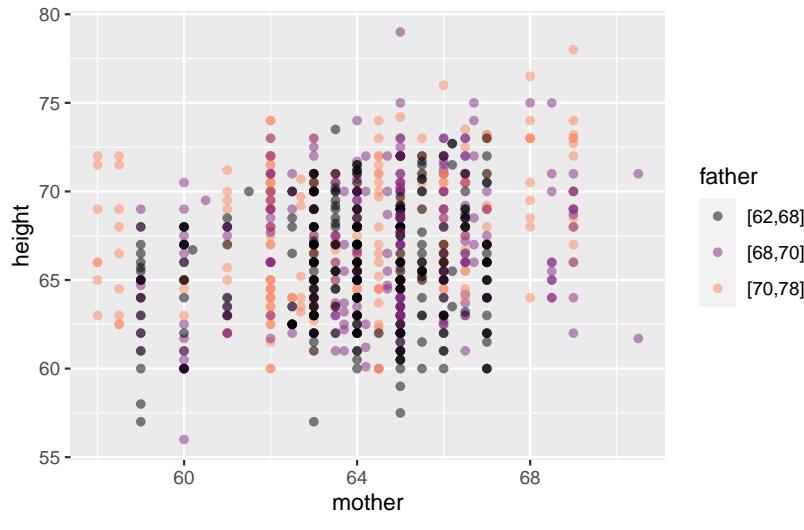


Figure 38: Mapping the quantitative explanatory variable to the horizontal axis.

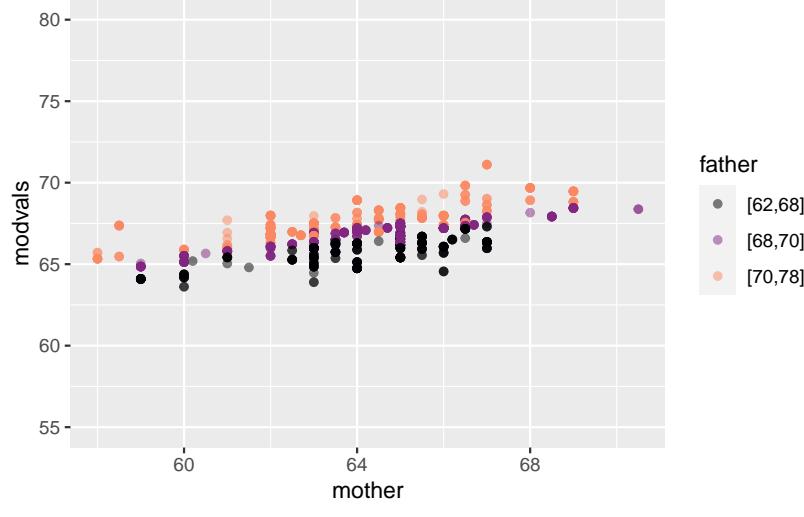
Two quantitative explanatory variables

This example draws on the same data frame as the previous two examples, but we use the mother's and father's heights for the explanatory variables. Both these explanatory variables are quantitative.

```
Galton |> point_plot(height ~ mother + father)
Galton |>
  mutate(modvals =
    model_values(height ~ mother + father)) |>
  point_plot(modvals ~ mother + father) |>
  gf_lims(y=c(55,80))
```



(a) Data layer



(b) Model-value layer

Figure 39: `height ~ mother + father`

As in Figure 37, mapping a *quantitative* variable to color makes the graph hard to read. To simplify, we've separated the data layer from the model layer.

It's almost impossible to see the relationship between fathers and children's heights in the data layer. By stripping away the child-to-child residuals, the model-value layer clarifies the

pattern. `father` is mapped to color, so the color strata represents the father/child relationship: shorter fathers (black) are tend to be lower on the y scale that represents the child's height. Taller fathers (orange) are associated with higher y values, that is, taller children. `mother` is mapped to x, so the mother/child relationship appears in the upward slope of the cloud of model-values, similar to the slope in Figure 38.

11 Model functions

From the start of these Lessons, we have talked about revealing patterns in data, particularly those that describe relationships among variables. Graphics show patterns in a way that is particularly attuned to human cognition, and we have leaned on them heavily. In this Lesson, we turn to another form of description of relationships between variables: simple mathematical functions.

11.1 Basics of mathematical functions

We will need only the most basic ideas of mathematics to enable our work with functions. There won't be any algebra required.

1. In mathematics, a function is a relationship between one or more inputs and an output. In our use of functions for statistical thinking, the output corresponds to the response variable, the inputs to the explanatory variables.
2. In mathematical notation, functions are conventionally written idiomatically using single-letter names. For instance, letters from the end of the alphabet— x , y , t , and z —are names for function inputs. The convention uses letters from the start of the alphabet as stand-ins for numerical values; these are called **parameters** or, equivalently, **coefficients**. These conventions are almost 400 years old and are associated with Isaac Newton (1643-1727).

Not quite 300 years ago, a new mathematical idea, the function, was introduced by Leonhard Euler (1707-1783). Since the start and end of the alphabet had been reserved for names of variables and parameters, a convention emerged to use the letters f and g for function names.

3. To say, “Use function f to transform the inputs x and t to an output value,” the notation is $f(x, t)$. To emphasize: Remember that $f(x, t)$ stands for the **output** of the function. Statistics often uses the one-letter name

style, but when the letters stand for things in the real world, it can be preferable to use names that remind us what they stand for: `age`, `time_of_day`, `mother`, `wrist`, `prices`, and such.

- Mathematical functions are idealizations. Importantly, they differ from much of everyday experience. Every mathematical function may have only one *output value* for any given *input value*. We say that mathematical functions are “**single valued**. For instance, the mathematical value $f(x = 3, t = 10)$ will be the same every time it is calculated. In everyday life, a quantity like `cooking_time(temperature=300)` might vary depending on other factors (like `altitude`) or even randomly.

When functions are graphed, the single-valued property is shown using a thin line for the function value, as it depends on the inputs. (See Figure 40.)

- In contrast to the large variety encountered in mathematics courses, we will need only the three function types shown in Figure 40:
 - Straight-line
 - Sigmoid curve, resembling a highly-slanted letter *S*.
 - Discrete-input, where the input is a categorical level. The function values, one for each level of the categorical input, are drawn as short horizontal strokes.
- A **formula** is an arithmetic expression written in terms of input names and coefficient names, for example, $ax + b$. We write $f(x) \equiv ax + b$ to say that function f is defined by the formula $ax + b$. All three function types in (5) use two coefficients, a and b . The sigmoid function uses an S-shaped translation between $ax + b$ and the function output value.

Function type	Algebraic	Math names	Statistics names
Straight-line	$f(x) \equiv ax + b$	a is “slope”	a is coefficient on x

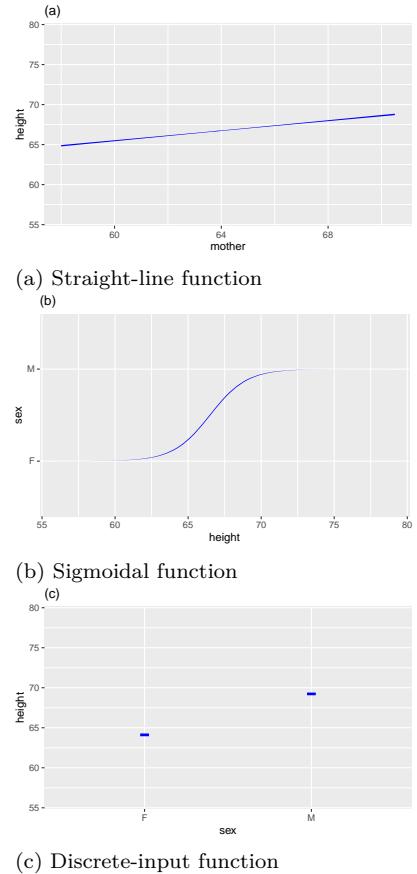


Figure 40: Three examples of single-valued functions.

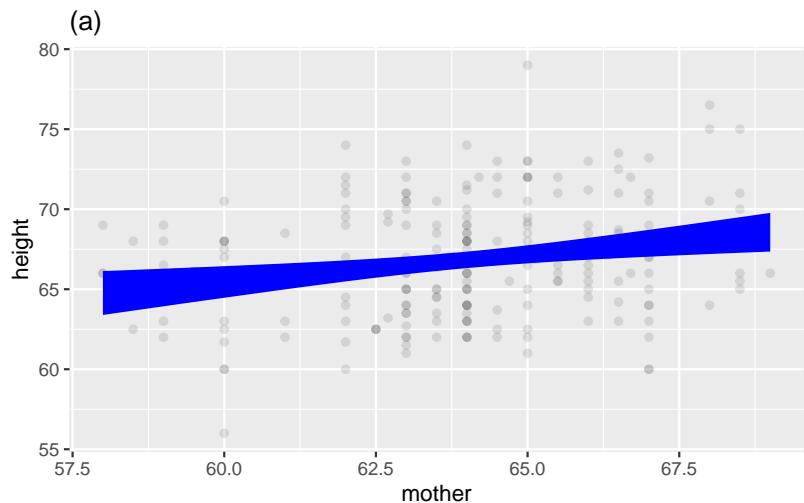
Function type	Algebraic	Math names	Statistics names
.	.	b is “intercept”	b is “intercept”
Sigmoid	$f(x) \equiv S(ax + b)$	a is “steepness” b is “center”	a is coefficient on x b is “intercept”
Discrete- input	$f(x) \equiv b + \begin{cases} 0 & \text{when } x = F \\ a & \text{when } x = M \end{cases}$	b is intercept .	b is “intercept” a is “sexM coefficient”

In all three cases, the a coefficient quantifies how the function output changes in value as the input x changes. For the straight-line function, a is the slope. Similarly, a is the steepness halfway up the curve for the sigmoid function. And for the discrete-input function, a is the amount to add to the output when the input x equals the particular categorical level (M in the above example).

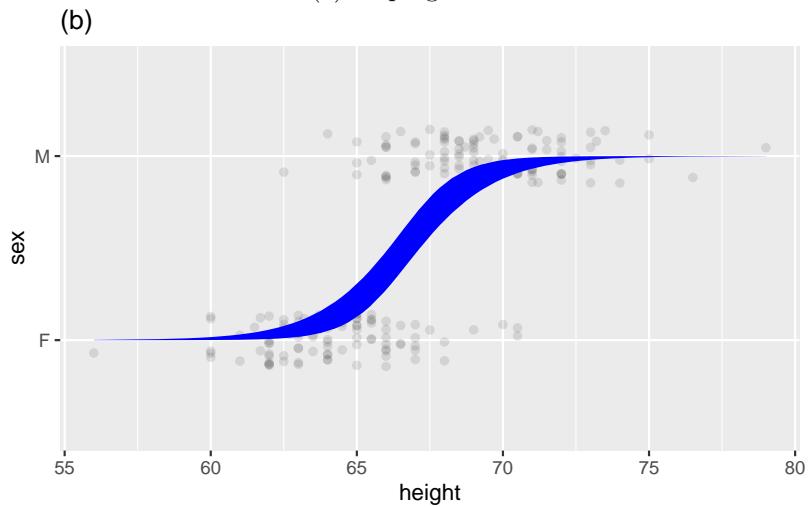
11.2 Statistical models

Many mathematical functions are used in statistics, but to quantify a relationship among variables rooted in data, statistical thinkers use models that resemble a mathematical function but are **bands** or **intervals** rather than the thin marks of single-valued function graphs. Figure 41 shows three such statistical models, each of which corresponds to one of the mathematical functions in Figure 40.

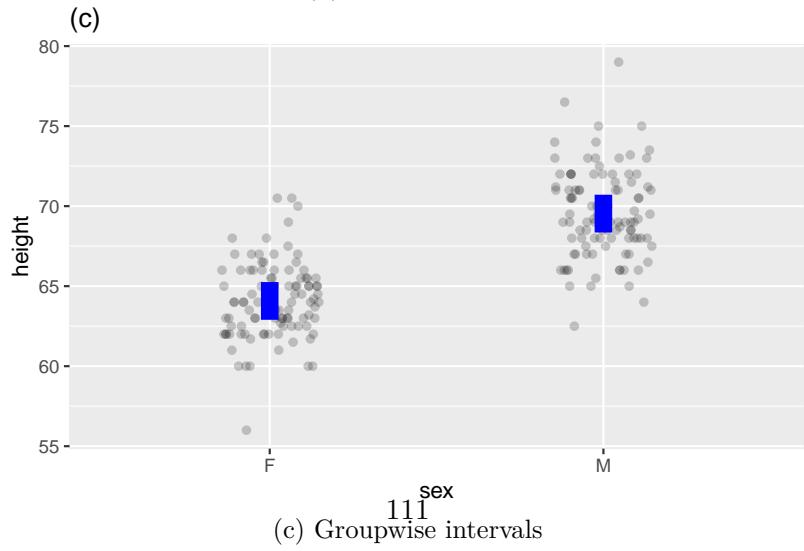
Quantifying uncertainty is a significant focus of statistics. The bands or intervals—the vertical extent of the model annotation—are an essential part of a statistical model. In



(a) Sloping band



(b) Sigmoid band



(c) Groupwise intervals
111

Figure 41: Statistical models constructed from the Galton data frame.

contrast, single-valued mathematical functions come from an era that didn't treat uncertainty as a mathematical topic.

To draw a model annotation, the computer first finds the single-valued mathematical function that passes through the band or interval at the mid-way vertical point. We will identify such single-valued functions as “**model functions**.” Model functions can be written as **model formulas**, as described in Section 11.1.

Another critical piece is needed to draw a model annotation: the vertical spread of the statistical annotation that captures the uncertainty. This is an *essential* component of a statistical model. Before dealing with uncertainty, we will need to develop concepts and tools about randomness and noise as presented in Lessons 13 through 19.

For now, however, we will focus on the model function, particularly on the interpretation of the coefficients. We won’t need formulas for this. Instead, focus your attention on two kinds of coefficients:

- the intercept, which we wrote as b when discussing mathematical functions. In statistical reports, it is usually written (**Intercept**).
- the other coefficient, which we named **a** to represent the slope/steepleness/change, always measures how the model function output changes for different values of the explanatory variable. If x is the name of a *quantitative* explanatory variable, the coefficient is called the “ x -coefficient.” But for a categorical explanatory variable, the coefficient refers to *both* the name of the explanatory variable and the particular level to which it applies. For example, in Figure 41(c), the explanatory variable is **sex** and the level is M, so the coefficient is named **sexM**.

11.3 Training a model

The model annotation in an annotated point plot is arranged to show the model function and uncertainty simultaneously. To

construct the model in the annotation, `point_plot()` uses another function: `model_train()`.

Now that we have introduced model functions and coefficients, we can explain what `model_train()` does:

`model_train()` finds numerical values for the coefficients that cause the model function to align as closely as possible to the data. As part of this process, `model_train()` also calculates information about the uncertainty, but we put that off until later.)

“Train” is meant in the sense of “training a pet” or “vocational training.” `model_train()` has nothing to do with miniature-scale transportation layouts found in hobbyists’ basements.

Use `model_train()` in the same way as `point_plot()`. A data frame is the input. The only required argument is a tilde expression specifying the names of the response variable and the explanatory variables, just as in `point_plot()`.

As you know, the output from `point_plot()` is a *graphic*. Similarly, the output from wrangling functions is a *data frame*. The output of `model_train()` is not a graphic (like `point_plot()`) or a data frame (like the wrangling functions). Instead, it is a new kind of thing that we call a “**model object**.”

```
Galton |> model_train(height ~ mother)
```

```
Call:  
stats::lm(formula = tilde, data = data)  
  
Coefficients:  
(Intercept)      mother  
        46.6908       0.3132
```

Recall that *printing* is default operation to do with the object produced at the end of a pipeline. Printing a data frame or a graphic displays more-or-less the entire object. But for model objects, printing gives only a glimpse of the object. This is because there are multiple perspectives to take on model objects, for instance, the model function or the uncertainty.

Choose the perspective you want by piping the model output into another function, two of which we describe here:

`model_eval()` looks at the model object from the perspective of a model function. The arguments to `model_eval()` are values for the explanatory variables. For instance, consider the height of the child of a mother who is five feet five inches (65 inches):

```
Galton |> model_train(height ~ mother) |> model_eval(mother = 65)
```

mother	.lwr	.output	.upr
60	60	70	70

The output of `model_eval()` is a data frame. The `mother` column repeats the input value given to `model_eval()`. `.output` gives the model output: a child's height of 67 inches. There are two other columns: `.lwr` and `.upr`. These relate to the uncertainty in the model output. We will discuss these in due time. For the present, we simply note that, according to the model, the child of a 65-inch tall mother is likely to be between 60 and 74 inches

`conf_interval()` provides a different perspective on the model object: the *coefficients* of the model function.

```
Galton |> model_train(height ~ mother) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	40	50	50
mother	0.2	0.3	0.4

The form of the output is, as you might guess, a data frame. The `term` value identifies which coefficient the row refers to; the `.coef` column gives the numerical value of the coefficient. Once again, there are two additional columns, `.lwr` and `.upr`. These describe the uncertainty in the coefficient. Again, we will get to this in due time.

i Regression models versus classifiers

There are two major kinds of statistical models: **regression models** and **classifiers**. In a regression model, the response variable is always a *quantitative* variable. For a classifier, on the other hand, the response variable is *categorical*.

These Lessons involve only **regression** models. The reason: This is an introduction, and regression models are easier to express and interpret. Classifiers involve multiple model functions; the bookkeeping involved can be tedious. (We'll return to classifiers in 21.)

However, one kind of classifier is within our scope because it is also a regression model. How can that happen? When a categorical variable has only two levels (say, dead and alive), we can translate it into zero-one format. A two-level categorical variable is also a numerical variable but with the numerical levels zero and one.

When the response variable is zero-one, we can use regression techniques. Often, it is advisable to use a custom-built technique called **logistic regression**. `model_train()` knows when to use logistic regression. The sigmoidal shape is a good indication that logistic regression is in use. (See, e.g. Figure 41(b))

“Regression” is a strange name for a statistical/mathematical technique. It comes from a misunderstanding in the early days of statistics, which remains remarkably prevalent today. See Additional Topic Enrichment topic 11.1 Regression to the mean.

11.4 Model functions with multiple explanatory variables

The ideas of model functions and coefficients apply to models with multiple explanatory variables. To illustrate, let's return to the `Galton` data and use the heights of the `mother` and `father` and the child's `sex` to account for the child's height.

The printed version of the model doesn't give any detail ...

```
Galton |>
  model_train(height ~ mother + father + sex)
```

Call:
stats::lm(formula = tilde, data = data)

Coefficients:
(Intercept) mother father sexM
15.3448 0.3215 0.4060 5.2260

... but the coefficients tell us about the relationships:

```
Galton |>
  model_train(height ~ mother + father + sex) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	9.954	15.34	20.74
mother	0.2601	0.3215	0.3829
father	0.3487	0.406	0.4633
sexM	4.943	5.226	5.509

There are four coefficients in this model. As always, there is the intercept, which we wrote b in Section 11.1. But instead of one a coefficient, each explanatory variable has a separate coefficient.

The intercept, 15.3 inches, gives a kind of baseline: what the child's height would be before taking into account `mother`, `father` and `sex`. Of course, this is utterly unrealistic because there must always be a mother and father.

Like the a coefficient in Section 11.1, the coefficients for the explanatory variables express the change in model output per change in value of the explanatory variable. The mother coefficient, 0.32, expresses how much the model output will change for each inch of the mother's height. So, for a mother who is

65 inches tall, add $0.32 \times 65 = 20.8$ inches to the model output. Similarly, the `father` coefficient expresses the change in model output for each inch of the father's height. For a 68-inch father, that adds another $0.41 \times 68 = 27.9$ inches to the model output.

The `sexM` coefficient gives the increase in model output when the child has level M for `sex`. So add another 5.23 inches for male children.

There is no `sexF` coefficient, but this is only a matter of accounting. R chooses one level of a categorical variable to use as a baseline. Usually, the choice is alphabetical: "F" comes before "M," so females are the baseline.

11.5 Case study: Get out the vote!

There is perennial concern with voter participation in many countries: only a fraction of potential voters do so. Many civic organizations seek to increase voter turnout. Political campaigns spend large amounts of money on advertising and knock-on-the-door efforts in competitive districts. (Of course, they focus on neighborhoods where the campaign expects voters to be sympathetic to them.) However, civic organizations don't have the fund-raising capability of campaigns. Is there an inexpensive way for these organizations to get out the vote?

Consider an experiment in which get-out-the-vote post-cards with messages of possibly different persuasive force were sent randomly to registered voters before the 2006 mid-term election.

The message on each post-card was one of the following:

- The "Neighbors" message listed the voter's neighbors and whether they had voted in the previous primary elections. The card promised to send out the same information after the 2006 primary so that "you and your neighbors will all know who voted and who did not."
- The "Civic Duty" message was, "Remember to vote. DO YOUR CIVIC DUTY—VOTE!"

See Alan S. Gerber, Donald P. Green, and Christopher W. Larimer (2008) "Social pressure and voter turnout: Evidence from a large-scale field experiment." *American Political Science Review*, vol. 102, no. 1, pp. 33–48

Table 17: The `Go_vote` data frame.

sex	yearofbirth	primary2004	messages	primary2006	hhszie
male	1941	abstained	Civic Duty	abstained	2
female	1947	abstained	Civic Duty	abstained	2
male	1951	abstained	Hawthorne	voted	3
female	1950	abstained	Hawthorne	voted	3
female	1982	abstained	Hawthorne	voted	3
male	1981	abstained	Control	abstained	3

- The “Hawthorne” message simply told the voter that “YOU ARE BEING STUDIED!” as part of research on why people do or do not vote. [The [name comes from studies](https://en.wikipedia.org/wiki/Hawthorne_effect) conducted at the “Hawthorne Works” in Illinois in 1924 and 1927. Small changes in working conditions inevitably *increased* productivity for a while, even when the change undid a previous one.] {.aside}
- A “control group” of potential voters, picked at random, received no post-card.

The voters’ response—whether they voted in the election—was gleaned from public records. The data involving 305,866 voters is in the `Go_vote` data frame. Three of the variables are of clear relevance: the type of get-out-the-vote message (in `messages`), whether the voter voted in the upcoming election (`primary2006`), and whether the voter had voted in the previous election (`primary2004`). Other explanatory variables—year of the voter’s birth, sex, and household size—were included to investigate possible effects.

It’s easy to imagine that whether a person voted in `primary2004` has a role in determining whether the person voted in `primary2006`, but do the experimental `messages` sent out before the 2006 primary also play a role? To see this, we can model `primary2006` by `primary2004` and `messages`.

However, as you can see in Table 17, both `primary2006` and

`primary2004` are categorical. Using a categorical variable in an explanatory role is perfectly fine. But in regression modeling, the response variable must be *quantitative*. To conform with this requirement, we will create a version of `primary2006` that consists of zeros and ones, with a one indicating the person voted in 2006. Data wrangling with `mutate()` and the `zero_one()` function can do this:

```
Go_vote <- Go_vote |>
  mutate(voted2006 = zero_one(primary2006, one = "voted"))
```

After this bit of wrangling, `Go_vote` has an additional column:

sex	yearofbirth	primary2004	messages	primary2006	hhszie	voted2006
female	1965	abstained	Control	abstained	2	0
male	1944	voted	Neighbors	voted	2	1
female	1952	voted	Civic Duty	voted	4	1
female	1947	voted	Neighbors	abstained	2	0
female	1943	abstained	Control	voted	2	1
female	1964	voted	Civic Duty	abstained	2	0
male	1970	abstained	Hawthorne	voted	2	1
female	1969	abstained	Hawthorne	abstained	2	0

No information is lost in this conversion; `voted2006` is always 1 when the person voted in 2006 and always 0 otherwise. Since `voted2006` is numerical, it can play the role of the response variable in regression modeling.

For reference, here are the means of the zero-one variable `voted2006` for each of eight combinations of explanatory variable levels: four postcard messages times the two values of `primary2004`. Note that `voted2006` is a zero-one variable; the means will be the proportion of 1s. That is, the mean of `voted2006` is the *proportion* of voters who voted in 2006.

```
Go_vote |>
  summarize(vote_proportion = mean(voted2006),
            .by = c(messages, primary2004)) |>
  arrange(messages, primary2004)
```

messages	primary2004	vote_proportion
Control	abstained	0.24
Control	voted	0.39
Civic Duty	abstained	0.26
Civic Duty	voted	0.4
Hawthorne	abstained	0.26
Hawthorne	voted	0.41
Neighbors	abstained	0.31
Neighbors	voted	0.48

For each kind of message, people who voted in 2004 were likelier to vote in 2006. For instance, the non-2004 voter in the control group had a turnout of 23.7%, whereas the people in the control group who did vote in 2004 had a 38.6% turnout.

Similar information is presented more compactly by the coefficients for a basic model:

```
Go_vote |>
  model_train(voted2006 ~ messages + primary2004, family = "lm") |>
    conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	0.2331	0.2355	0.238
messagesCivic Duty	0.01302	0.01804	0.02305
messagesHawthorne	0.02028	0.02529	0.03031
messagesNeighbors	0.07533	0.08034	0.08536
primary2004voted	0.1494	0.1527	0.156

It takes a little practice to learn to interpret coefficients. Let's start with the `messages` coefficients. Notice that there is a coefficient for each of the levels of `messages`, with "Control" as the reference level. According to the model, 23.6% of the control group who did not vote in 2004 turned out for the 2006 election. The `primary2004voted` coefficient tells us that people

who voted in 2004 were 15.3 percentage points more likely to vote in 2006 than the 2004 abstainers.

Each non-control postcard had a higher voting percentage than the control group. The manipulative “Neighbors” post-card shows an eight percentage point increase in voting, while the “Civic Duty” and “Hawthorne” post-cards show smaller changes of about two percentage points each.

11.6 Tradition and “correlation”

The reader who has already encountered statistics may be familiar with the word “**correlation**,” now an everyday term used as a synonym for “relationship.” “**Correlation coefficient**” refers to a numerical summary of data invented almost 150 years ago. Since the correlation coefficient emerged very early in the history of statistics, it is understandably treated with respect by traditional textbooks.

We don’t use correlation coefficients in these *Lessons*. As might be expected for such an early invention, they describe only the simplest relationships. Instead, the regression models introduced in this Lesson enable us to avoid over-simplifications when extracting information from data.

We will discuss the difference between “percent” and “percentage point” in Lesson 21. In brief: “percent” refers to a *fraction* while “percentage point” is a *change in a fraction*.

12 Adjustment

The phrase “all other things being equal” is a critical qualifier in describing relationships. To illustrate: A simple claim in economics is that a high price for a commodity reduces the demand. For example, increasing the price of gasoline will reduce demand as people avoid unnecessary driving or purchase electric cars. Nevertheless, the claim can be considered obvious only with the qualifier *all other things being equal*. For instance, the fuel price might have increased because a holiday weekend and the attendant vacation travel has increased the demand for gasoline. Thus, higher gasoline prices may be associated with higher demand unless holding constant other variables such as vacationing.

The Latin equivalent of “all other things being equal” is sometimes used in economics: “**ceteris paribus**”. The economics claim would be, “higher prices are associated with lower demand, *ceteris paribus*.”

Although the phrase “all other things being equal” has a logical simplicity, it is impractical to implement “all.” So instead of the blanket “all other things,” it is helpful to consider just “some other things” to be held constant, being explicit about what those things are. Other phrases along the same lines are “adjusting for ...,” “taking into account ...,” and “controlling for”

12.1 Groupwise adjustment

“**Life expectancy**” is a statistical summary familiar to many readers. Life expectancy is often the evidence provided in debates about healthcare policies or environmental conditions. For instance, consider this pull-quote from the [Our World in Data website](#):

“Americans have a lower life expectancy than people in other rich countries despite paying much more for healthcare.”

The numbers in Table ?? faithfully reflect the overall situation in the different countries. Yet, without adjustment, they are not well suited to inform about specific situations. For example, life expectancies are usually calculated *separately* for males and females, acknowledging a significant association of life expectancy with sex, not just the availability of medical care. We will call such a strategy “**groupwise adjustment**” because it’s based on acknowledging difference between groups. You’ll see similar groupwise adjustment of life expectancy on the basis of race/ethnicity.

Over many years teaching epidemiology at Macalester College, I asked students to consider life-expectancy tables and make policy suggestions for improving things. Almost always, their primary recommendations involved improving access to health care, especially for the elderly.

But life expectancy is not mainly, or even mostly, about old age. Two critical determinants are infant mortality and lethal activities by males in their late teenage and early adult years. If we want to look at conditions in the elderly, we need to consider elderly people separately, not mixed in with infants, children, and adolescents. For reasons we won’t explain here, with life expectancy calculations it’s routine to calculate a separate “life expectancy at age X” for each age year. Table ?? shows, according to the World Health Organization, how many years longer a 70-year old can expect to live. The 30-year difference between Japan and Somalia seen in Table ?? is reduced, for 70-year olds, to about a decade. The differences between males and females are similarly reduced

12.2 Adjustment with *per*

The US government’s Centers for Medicare Studies gives [some numbers about the age distribution](#) of “personal health-care” spending:

“In 2020, children (0-18) accounted for 23 percent of the population and 10 percent of personal health care (PHC) spending, working age adults (19-64)

Table 18: Life expectancy at birth for several countries and territories.
[Source](#)

Country	Female	Male
Japan	87.6	84.5
Spain	86.2	80.3
Canada	84.7	80.6
United States	80.9	76.0
Bolivia	74.0	71.0
Russia	78.3	66.9
North Korea	75.9	67.8
Haiti	68.7	63.3
Nigeria	63.3	59.5
Somalia	58.1	53.4

Table 19: Life expectancy at age 70. (Main source: [World Health Organization](#)) average of 65-74 year olds)

Country	Female	Male
Japan	21.3	17.9
Canada	18.0	15.6
Spain	17.0	14.0
United States	18.3	16.3
Russia	16.2	12.2
Bolivia	13.6	13.0
Haiti	12.9	12.1
Somalia	11.6	9.7

accounted for 60 percent of the population and 53 percent of PHC, and older adults (65 and older) account for 17 percent of the population and 37 percent of PHC.”

The textual presentation of data presents relevant information but obscures the patterns. The author would have done better by placing the numbers that are to be compared to one another next to one another. A tabular organization makes it much easier to compare the relative population sizes of the age groups. For instance, the population column of Table ?? shows at a glance that the population of children and older adults are about the same.

Similarly, the table’s “PHC spending” column makes it obvious that PHC spending is much higher for working age adults than for either children or older adults.

In comparing the spending between groups, it can be helpful to *take into account* the differing population sizes. A **per capita** adjustment—spending divided by population size—accomplishes it. For instance, the per capita adjustment for children is 10% / 23%, that is, 0.43. Table 21 shows the *per capita* spending for all three age groups.

Table 21: Adjusting spending for the size of the population gives a clearer indication of how spending compares between the different age groups.

group	age	population	spending	spending per capita
children	0-18	23%	10%	0.43
working age adults	19-64	60%	53%	0.88
older adults	65+	17%	37%	2.18

Including the *per capita* adjusted spending in the table makes it easy to see an important pattern: older adults have much higher health spending (per person) than the other groups.

The literal meaning of “*per capita*” is “for each head.” But the method of adjusting by dividing one quantity by another has much broader applications. To illustrate, let’s return to the

group	age span	population	PHC spending
children	0-18	23%	10%
working age	19-64	60%	53%
older adults	65+	17%	37%

Table 3: A tabular arrangement of the data from the Centers for Medicare Studies

example of college grades from Section 7.2. There, we calculated using simple wrangling each student’s grade-point average and an instructor grade-giving average. The instructor’s grade-giving average varies so much that it seems short-sighted to neglect it as a factor in determining a student’s grade in that instructor’s courses.

An adjustment for the instructor can be made by constructing a *per-type* index. An instructor gave each grade, but instead of considering the grade literally, let’s divide the grade by the grade-giving average of the instructor involved.

We can consider the instructors’ iGPA to calculate an instructor-adjusted GPA for students. We create a data frame with the instructor ID and numerical grade point for every grade in the **Grades** and **Sessions** tables. First, we use “joins” to bring together the tables from the database.

```
Extended_grades <- Grades |>
  left_join(Sessions) |>
  left_join(Gradepoint) |>
  select(sid, iid, sessionID, gradepoint)
```

Next, calculate the instructor-by-instructor “grade-giving average” (gga):

```
Instructors <- Extended_grades |>
  summarize(gga = mean(gradepoint, na.rm = TRUE), .by = iid)
```

Joining the **Instructors** data frame with **Extended_grades** puts the grade earned and the average grade given next to one another. The unit of observation is still a student receiving a grade in a class session.

```
With_instructors <-
  Extended_grades |>
  left_join(Instructors)
```

sid	iid	sessionID	gradepoint
S32418	inst268	session2911	3
S32328	inst436	session3524	3.66
S32250	inst268	session2911	2.66
S32049	inst436	session2044	3.33
S31914	inst436	session2044	3.66
S31905	inst436	session2044	4
S31833	inst436	session3524	3.33
S31461	inst264	session1904	4
S31197	inst436	session3524	3
S31194	inst264	session1904	2

A few of the 6,124 rows in the **Extended_grades** table.

iid	gga
inst436	3.584
inst264	2.974
inst268	3.062

Three rows from the **Instructors** data frame.

sid	iid	sessionID	gradepoint	gga
S32310	inst436	session2193	3.66	3.584
S31794	inst436	session2541	3.66	3.584
S32289	inst264	session2235	4	2.974
S31461	inst264	session1904	4	2.974
S32211	inst268	session2650	2.33	3.062
S32250	inst268	session2911	2.66	3.062

A few rows of the result of joining `Extended_grades` with `Instructors`. The data frame has 6,124 rows in total.

Make the *per* adjustment by dividing `gradepoint` by `gga` to create a grade index. We will then average this index for each student to create each student's instructor-adjusted GPA (`adj_gpa`), shown in Table ??.

```
Adjusted_gpa <-
  With_instructors |>
  mutate(index = gradepoint / gga) |>
  summarize(adj_gpa = mean(index, na.rm = TRUE), .by = sid)
```

12.3 Adjustment by modeling

We will use the word “**adjustment**” to name the statistical techniques by which “other things” are considered. Those other things, as they appear in data, are called “**covariates**.”

There are two phases for modelling-based adjustment, one requiring careful thought and understanding of the specific system under study, the other—the topic of this section—involving only routine, straightforward modeling calculations.

Phase 1: Choose relevant covariates for adjustment. This almost always involves familiarity with the real-world context.

Phase 2: Build a model with the covariates from Phase 1 as explanatory variables.

sid	adj_gpa
S31197	0.9578
S31914	1.04
S31461	1.148
S32250	0.9284
S31194	0.9978
S32418	0.9328

... for 443 rows altogether

To illustrate, we return to the college grades example in Section 12.2. There, we did a *per* adjustment of each grade by the average of all the grades assigned by the instructor (the “grade-giving average”: `gga`).

Now we want to examine how to incorporate other factors into the adjustment, for instance class size (`enroll`) and class `level`. We will also change from the politically unpalatable instructor-based grade-given average to using department (`dept`) as a covariate.

To start, we point out that the conventional GPA can also be found by modeling `gradepoint ~ sid`.

```
Joined_data <- Grades |>
  left_join(Sessions) |>
  left_join(Gradepoint)
Raw_model <-
  Joined_data |>
  model_train(gradepoint ~ sid)
```

The model values from `Raw_model` will be the unadjusted (raw) GPA. We can compute those model values by making a data frame with all the input values for which we want an output:

```
Students <- Grades |> select(sid) |> unique()
```

Now evaluate `Raw_model` for each of the inputs in `Students` to find the model value (called `.output` by `model_eval()`).

```
Raw_gpa <- Raw_model |>
  model_eval(Students) |>
  select(sid, raw_gpa = .output)
```

The advantage of such a modeling approach is that we can add covariates to the model specification in order to adjust for them. To illustrate, we will adjust using `enroll`, `level`, and `dept`:

```
Adjustment_model <-
  Joined_data |>
  model_train(gradepoint ~ sid + enroll + level + dept)
```

As we did before with `Raw_model`, we will evaluate `Adjustment_model` at all values of `sid`. But we will also *hold constant* the enrollment, level, and department by setting their values. For instance, in the following, we look at every student as if their classes were all in department D, at the 200 level, and with an enrollment of 20.

```
Inputs <- Students |>
  mutate(dept = "D", level = 200, enroll = 20)
Model_adjusted_gpa <-
  Adjustment_model |>
  model_eval(Inputs) |>
  rename(modeled_gpa = .output)
```

This is the core of adjustment: comparing individual specimens *after* putting them on the same footing, that is, *ceteris paribus*.

In Section 12.3, we calculated three different versions of the GPA:

1. The *raw* GPA, which we calculated in two equivalent ways, with `summarize(mean(gradepoint), .by = sid)` and with the model `gradepoint ~ sid`.
2. The grade-given average used to create an index that involves `gradepoint / gga`.
3. The model using covariates `level`, `enroll`, and `dept`.

The statistical thinker knows that GPA is a social construction, not a hard-and-fast reality. Let's see to what extent the different versions agree.

13 Signal and noise

Imagine being transported back to June 1940. The family is in the living room, sitting around the radio console, waiting for it to warm up. The news today is from Europe, the surrender of the French in the face of German invasion. Press the play button and listen ...

Your browser does not support the audio tag.

The spoken words from the recording are discernible despite the hiss and clicks of the background noise. The situation is similar to a conversation in a sports stadium. The crowd is loud, so the speaker has to shout. The listener filters out the noise (unless it is too loud) and recovers the shouted words.

Engineers and others make a distinction between **signal** and **noise**. The engineer aims to separate the signal from the noise. That aim applies to statistics as well.

There are many sources of noise in data. Every variable has its own story, part of which is noise from measurement errors and recording blunders. For instance, economists use national statistics, like GDP, even though the definition is arbitrary (a Hurricane can raise GDP!), and early reports are invariably corrected a few months later. Historians go back to original documents, but inevitably many of the documents have been lost or destroyed: a source of noise. Even in elections where, in principle, counting is straightforward, the voters' intentions are measured imperfectly due to "hanging chads," "butterfly ballots," broken voting machines, spoiled ballots, and so on.

In this Lesson, we will take the perspective that every measurement or observation, whether quantitative or categorical, is a mixture of *signal* and *noise*. An important objective of data analysis is to identify the signal by filtering out the noise.

Consider the college-grades setting introduced in Chapter 12. The core individual measurements or observations are the student ID and the grade. Every student knows that each grade contains a noisy component caused by random factors: feeling unwell during the final exam, missed an important class meeting when you were on a varsity trip, found an unexpected extra

hour for study, and so on. As well, the student ID is potentially noisy: grades get transposed between students, a record is lost in transmission to the registrar,

We have natural expectations that the student ID will be noiseless or, if not perfect, that errors will be vanishingly rare. To this end, colleges employ information technology (IT) specialists: engineers who design and manage computer database systems, transmission protocols, course-support software, and grade-entry user interfaces. When there is an error, the IT professionals debug the system, and make the necessary changes.

In contrast, most colleges have no quality assurance program or staff to help measure or reduce the amount of noise in a grade. But, from earlier Lessons, we now have the tools to measure noise and look for factors that introduce noise.

i Noise in hiring

On several occasions, the author has testified in legal hearings as a statistical expert. In one case, the US Department of Labor audited a contractor's records with several hundred employees and high employee turnover. The spreadsheet files led the Department to bring suit against the contractor for discriminating against Hispanics. The hiring records showed that many Hispanics applied for jobs; the company hired none. An open-and-shut case.

The lawyers for the defense asked me, the statistical expert, to review the findings from the Department of Labor. The lawyers thought they were asking me to check the arithmetic in the hiring spreadsheets. As a statistical thinker, I know that arithmetic is only part of the story; the origin of the data is critically important. So, I asked for the complete files on all applicants and hires the previous year.

The spreadsheet files and the paper job applications were in accord; there were many Hispanic applicants. But the ethnicity data on the paper job application form was not always consistent with the data on hiring spreadsheets. It turned out that whenever an applicant was hired, the con-

tractor (per regulation) got a report on that person from the state police. The report returned by the state police had only two available race/ethnicities: white and Black. The contractor's personnel office filled in the hired-worker spreadsheet based on the state police report. So all the Hispanic applicants who were hired had been transformed into white or Black by the state police. Noise. The Department of Labor dropped its suit. The audit had identified noise, not the signal of discrimination.

13.1 Partitioning data into signal and noise

Recall that we contemplate every observation and measurement as a combination of signal and noise.

$$\text{individual observation} \equiv \text{signal} + \text{noise}$$

From an isolated, individual specimen, say student `sid4523` getting a grade of B+, there is no way to say what part of the B+ is signal and what part is noise. But from an extensive collection of specimens, we can potentially identify patterns across them, treating them collectively rather than as individuals.

$$\text{response variable} \equiv \text{pattern} + \text{noise}$$

To make a sensible partitioning of the *amount* of signal and the *amount* of noise, we need those two amounts to add up to the *amount* of the response variable.

$$\text{amount}(\text{response variable}) \equiv \text{amount}(\text{pattern}) + \text{amount}(\text{noise})$$

We must carefully choose a method for measuring *amount* to ensure the above relationship holds. An example comes from chemistry: When two fluids are mixed, the *volume* of the mixture does not necessarily equal the sum of the volumes of the individual fluids. The same is true if we measure the amount by the number of molecules; chemical reactions can increase

or decrease the number of molecules in the mixture from the sum of the number of molecules in the individual fluids. There is, however, a way to measure *amount* that honors the above relationship: amount measured by the *mass* of the fluid.

13.2 Model values as the signal

Our main tool for discovering patterns in data is modeling. For example, the pattern linking the body mass of a penguin to the sex and flipper length is:

```
Penguins |> model_train(mass ~ sex + flipper) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-5970	-5410	-4850
sexmale	268	348	427
flipper	44.1	47	49.8

Our choice of explanatory variables sets the type of signal we are looking for. In the 1940 news report from France, the signal of interest is human speech; our ears and brains automatically separate the signal from the noise. But suppose we were interested in another kind of signal, say a generator humming in the background or the dots and dashes of a spy's Morse Code signal. We would need a different sort of filtering to pull out the generator signal, and the speech and dots and dashes (and anything else) would be noise. Identifying the dots and dashes calls for still another kind of filtering.

The same is true for the penguins. If we look for a different type of signal, say body mass as a function of the bill shape, we get utterly different coefficients:

```
Penguins |>
  model_train(mass ~ bill_length + bill_depth) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	2550	3410	4270
bill_length	62.9	74.8	86.8
bill_depth	-179	-146	-112

Given the type of signal we seek to find, and the model coefficients for that type of signal, we are in a position to make a claim about what is the signal and what is the measurement in an individual penguin's body mass. Simply evaluate the model for that penguin's values of the explanatory variables to get the signal. What's left over—the **residuals**—is the noise.

To illustrate, lets look for the **sex** & **flipper** signal in the penguins:

```
With_signal <-
  Penguins |>
  mutate(signal = model_values(mass ~ sex + flipper),
        residuals = mass - signal)
```

It's time to point out something special about the residuals; there is no pattern component in the residuals. We can see that by modeling the residuals with the explanatory variables used to define the pattern:

```
With_signal |>
  model_train(residuals ~ sex + flipper) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-562	3.7e-11	562
sexmale	-79.4	1.82e-12	79.4
flipper	-2.84	-1.66e-13	2.84

The coefficients are zero! This means that the residuals do not show any sign of the pattern—everything about the pattern is contained in the signal!

A right triangle provides an excellent way to look at the relationship among the signal, residuals, and the response variable. We just saw that the residuals have nothing in common with the signal. This is much like the two legs of a right triangle; they point in utterly different directions!

For any triangle, any two sides add up to meet the third side. This is much like the response variable being the sum of the signal and the residuals. A right triangle has an additional property: the sum of the square lengths of the two legs gives the square length of the hypotenuse. For the penguin example, we can confirm this Pythagorean property when we use the **variance** to measure the “amount of” each component.

```
With_signal |>
  summarize(var(mass),
            var(signal) + var(residuals))
```

$$\frac{\text{var}(\text{mass})}{648370} \quad \frac{\text{var}(\text{signal}) + \text{var}(\text{residuals})}{648370}$$

i Signal to noise ratio

Engineers often speak of the “signal-to-noise” (SNR) ratio. In sound, this refers to the loudness of the signal compared to the loudness of the noise. For sound, the signal-to-noise ratio is often measured in decibels (dB). An SNR of 5 dB means that the signal is three times louder than the noise. You can listen to examples of noisy music and speech at [this web site](#), part of which looks like this:

Pop music

Input SNR	Noisy	BT	Plain Cosparse	Social Cosparse	Clean
5dB	Play/Pause				
10dB	Play/Pause				

Press the links in the “Noisy” column. The noisiest examples have an SNR of 5 dB. Press the play/pause button to hear the noisy recording, then compare it to the de-noised

transmission—the signal—by pressing play/pause in the “Clean” column.

It’s easy to calculate the signal-to-noise ratio in a model pattern; divide the amount of signal by the amount of noise:

```
With_signal |>  
  summarize(var(signal) / var(residuals))
```

$$\frac{\text{var}(\text{signal})/\text{var}(\text{residuals})}{4.2}$$

The signal is about four times larger than the noise. Converted to the engineering units of decibels, this is 6.2 dB. You can get a sense for what this means by listening to the 5 dB recordings and judging how clearly you can hear the signal.

13.3 R² (R-squared)

Statisticians measure the signal-to-noise ratio using a measure called R². It is equivalent to SNR, but compares the signal to the response variable instead of to the residuals. In our penguin example, `mass` is the response variable we chose.

```
With_signal |>  
  summarize(R2 = var(signal) / var(mass))
```

$$\frac{\overline{\text{R2}}}{0.8058}$$

R² has an attractive property: it is always between zero and one. You can see why by considering a right triangle: a leg can never be longer than the hypotenuse, and a leg can never be shorter than zero.

We've already met two perspectives that statisticians take on a model: `model_eval()` and `conf_interval()`. R^2 provides another perspective often (too often!) used in scientific reports. The `R2()` model-summarizing function does the calculations, adding in auxilliary information that we will learn how to interpret in due course.

```
Penguins |>
  model_train(mass ~ sex + flipper) |>
  R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
333	2	0.806	685	0.805	0	2	330

i Example: College grades from a signal-to-noise perspective

Returning to the college-grade example from Lesson 12 The usual GPA calculation is effectively finding a pattern in students' grades: :::{.cell}

```
Pattern <- Grades |>
  left_join(Sessions) |>
  left_join(Gradepoint) |>
  model_train(gradept ~ sid)
```

The R^2 of the pattern is:

```
Pattern |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
5700	440	0.32	5.6	0.27	0	440	5200

Is 0.32 a large or a small R^2 ? Researchers argue about such things. We will examine how such arguments are framed in later Lessons (especially Lesson 29).

An unconventional but, I think, helpful perspective is provided by the engineers' way of measuring the signal-to-noise ratio: decibels. For the `gradepoint ~ sid` pattern, the SNR is 3.2 dB. GPA appears to be a low-fidelity, noisy signal. :::

i A preview of things to come

We've pointed to the model values as the signal and the residuals as the noise. We will add another perspective on signal and noise in upcoming Lessons. The model coefficients will be treated as the signal for how the system works, the `.lwr` and `.upr` columns listed alongside the coefficients will measure the noise.

14 Simulation

Carl Wieman is a Nobel-prize-winning physicist and professor of education at Stanford University. Weiman writes, “For many years, I had two parallel research programs: one blasting atoms with lasers to see how they’d behave, and another studying how people learn.” Some of Wieman’s work on learning deals with the nature of “expertise.” He points out that experts have ways to monitor their own thinking and learning; they have a body of knowledge relevant to checking their own understanding.

This lesson presents you with tools to help you monitor your understanding of statistical methods. The monitoring strategy is based on **simulation**. In a simulation, you set up a hypothetical world where you get to specify the relationships between variables. The simulation machinery lets you automatically generate data collected in this hypothetical world. You can then apply statistical methods, such as regression modeling, to the simulated data. Then check the extent to which the results of the statistical methods, for instance model coefficients, agree with the hypothetical world that you created. If so, the statistical method is doing its job. If not, you have discovered limits to the statistical method, and you can explore how you might modify the method to give better agreement.

Like our data, the simulation involves a mixture of **signal** and **noise**. You implement the signal by setting precise mathematical rules for the relationships between your simulation variables. The noise comes in when we modify the mathematical rules to include measured amounts of noise.

14.1 Pure noise

We regard the residuals from a model as “noise” because they are entirely disconnected from the pattern defined by the tilde expression that directs the training of the model. There might be other patterns in the data—other explanatory variables, for instance—that could account for the residuals.

For simulation purposes, having an inexhaustible noise source guaranteed to be unexplainable by any pattern defined over any set of potential variables is helpful. We call this **pure noise**.

There is a mathematical mechanism that can produce pure noise, noise that is immune to explanation. Such mechanisms are called “**random number generators**.” R offers many random number generators with different properties, which we will discuss in Lesson 15. In this Lesson, we will use the `rnorm()` random number generator just because `rnorm()` generates noise that looks generically like the residual that come from models. But in principle we could use others. We use `datasim_make()` to construct our data-generating simulations. Here is a simulation that makes two variables consisting of pure random noise.

```
noise_sim <- datasim_make(  
  x <- rnorm(n),  
  y <- rnorm(n)  
)
```

Once a data-generating simulation has been constructed, we can draw a sample from it of whatever size `n` we like:

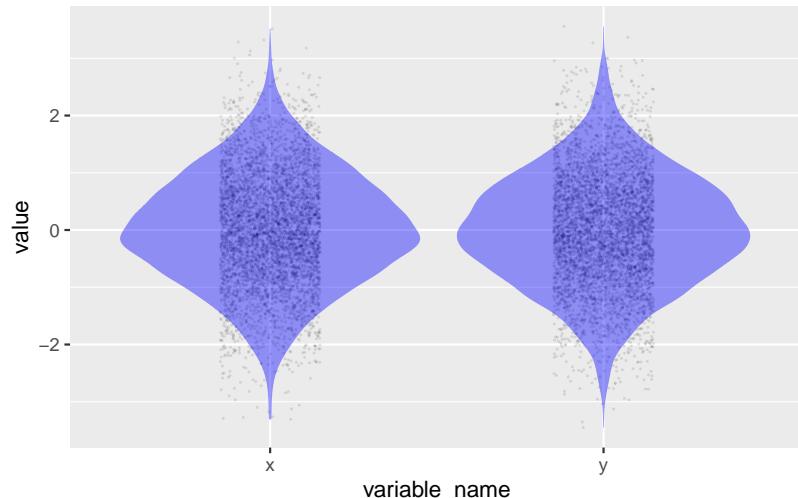
```
set.seed(153)  
noise_sim |> sample(n = 5)
```

x	y
2.819	-0.3191
-0.5242	-1.32
1.195	-2.286
-1.741	-0.7891
-0.4499	-0.8129

Although the numbers produced by the simulation are random, they are not entirely haphazard. Each variable is unconnected to the others, and each row is independent. Collectively, however, the random values have specific properties. The output above shows that the numbers tend to be in the range -2 to 2. In

?@fig-xy-random, you can see that the distribution of each variable is densest near zero and becomes less dense rapidly as the values go past 1 or -1. This is the so-called “normal” distribution, hence the name `rnorm()` for the random-number generator that creates such numbers.

```
set.seed(106)
noise_sim |> datasim_run(n=5000) |>
  pivot_longer(c(x, y),
              names_to = "variable_name", values_to = "value") |>
  point_plot(value ~ variable_name, annot = "violin",
              point_ink = 0.1, size = 0)
```



```
caption = "Distribution of the `x` and `y` variables from the simulation."
```

Another property of the numbers generated by `rnorm(n)` is that their mean is zero and their variance is one.

```
noise_sim |> sample(n=10000) |>
  summarize(mean(x), mean(y), var(x), var(y))
```

mean(x)	mean(y)	var(x)	var(y)
0.000382	0.00152	0.998	1

i But they aren't exactly what they ought to be!

Most people would likely agree that the means and variances in the above report are *approximately* zero and one, respectively, but are not precisely so.

This has to do with a subtle feature of random numbers. We used a sample size $n = 10,000$, but we might equally well have used a sample size 1. Would the mean of such a small sample be zero? If this were required, the number would hardly be random!

The mean of random numbers from `rnorm(n)` won't be exactly zero (except, very rarely and at random!). But the mean will tend to get closer to zero the larger that n gets. To illustrate, here are the means and variances from a sample that's 100 times larger: $n = 1,000,000$:

```
noise_sim |> sample(n=1000000) |>  
  summarize(mean(x), mean(y), var(x), var(y))
```

mean(x)	mean(y)	var(x)	var(y)
0.00118	0.000586	1	0.999

The means and variances can drift far from their theoretical values for small samples. For instance:

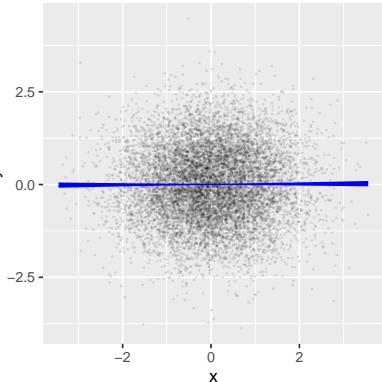
```
noise_sim |> sample(n=10) |>  
  summarize(mean(x), mean(y), var(x), var(y))
```

mean(x)	mean(y)	var(x)	var(y)
0.342	-0.155	0.424	0.776

Recall the claim made earlier in this Lesson that `rnorm()` generates a new batch of random numbers every time, unrelated to previous or future batches. In such a case, the model $y \sim x$ will, in principle, have an x -coefficient of zero. R^2 will also be zero, as will the model values. That is, x tells us nothing about y . `?@fig-noise_sim` verifies the claim with an annotated point

plot:

```
noise_sim |> sample(n=10000) |>
  point_plot(y ~ x, annot = "model",
             point_ink = 0.1, model_ink = 1, size=0.1)
  gf_theme(aspect.ratio = 1)
```



14.2 Simulations with a signal

The model values in `?@fig-noise_sim` are effectively zero: there is no signal in the `noise_sim`. If we want a signal between variables in a simulation, we need to state the data-generating rule so that there is a relationship between `x` and `y`. For example:

```
signal_sim <- datasim_make(
  x <- rnorm(n),
  y <- 3.2 * x + rnorm(n)
)
```

```
signal_sim |> sample(n=10000) |>
  point_plot(y ~ x, annot = "model",
             model_ink = 1, point_ink = 0.1, size=0.1)
```

Figure 42: A sample of ten-thousand points from `noise_sim`. The round cloud is symptomatic of a lack of relationship between the `x` and `y` values. The model values are effectively zero; `x` has nothing to say about `y`.

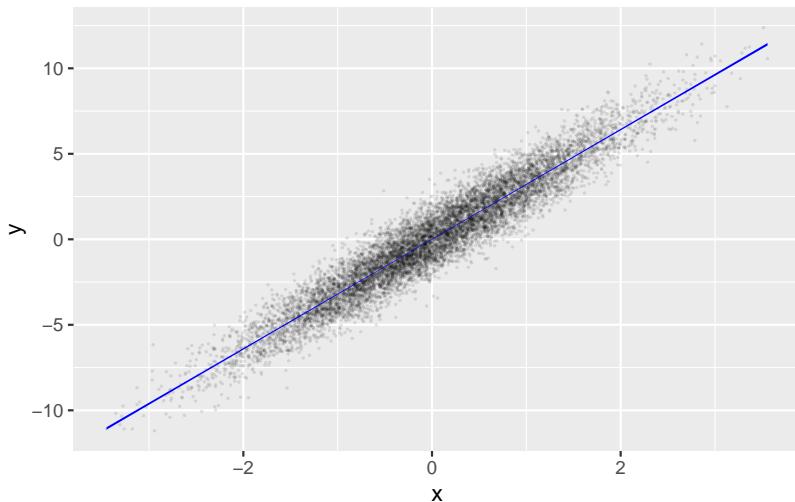


Figure 43: A sample of ten-thousand points from `signal_sim` where the y values are defined to be $y \leftarrow 3.2 * x + \text{rnorm}(n)$. The cloud is elliptical and has a slant.

14.3 Example: Heritability of height

Simulations can be set up to implement a **hypothesis** about how the world works. The hypothesis might or might not be on target. It's not even necessary that the hypothesis be completely realistic. Still, data from the simulation can be compared to field or experimental data.

Consider the following simulation in which each row of data gives the heights of several generations of a family. The simulation will be a gross simplification, as is often the case when starting to theorize. There will be a single hypothetical “mid-parent,” who reflects the average height of a real-world mother and father. The children—“mid-children”—will have a height mid-way between real-world daughters and sons.

```
height_sim <- datasim_make(
  mid_grandparent <- 66.7 + 2 * rnorm(n),
  mid_parent <- 17.81 + 0.733 * mid_grandparent + 0.99 * rnorm(n),
  mid_child <- 17.81 + 0.733 * mid_parent + 0.99 * rnorm(n),
  mid_grandchild <- 17.81 + 0.733 * mid_child + 0.99 * rnorm(n)
)
```

Note that the formulas for the heights of the mid-parents, mid-children, and mid-grandchildren are similar. The simulation imagines that the heritability of height from parents is the same in every generation. However, the simulation has to start from some “first” generation. We use the grandparents for this.

We can sample five generations of simulated heights easily:

```
sim_data <- height_sim |> sample(n=100000)
```

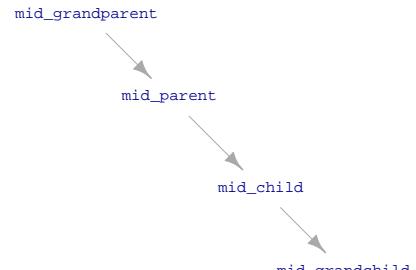


Figure 44

The simulation results compare well with the authentic Galton data:

```
Galton2 <- Galton |> mutate(mid_parent = (mother + father)/2)
Galton2 |> summarize(mean(mid_parent), var(mid_parent))
```

mean(mid_parent)	var(mid_parent)
66.66	3.066

```
sim_data |> summarize(mean(mid_parent), var(mid_parent))
```

mean(mid_parent)	var(mid_parent)
66.71	3.098

The mean and variance of the mid-parent from the simulation are close matches to those from Galton. Similarly, the model coefficients agree, with the intercept from the simulation data including one-half of the Galton coefficient on sexM to reflect the mid-child being halfway between F and M.

```
Mod_galton <- Galton2 |>
  model_train(height ~ mid_parent + sex)
Mod_sim   <- sim_data |>
  model_train(mid_child ~ mid_parent)
Mod_galton |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	9.804	15.2	20.6
mid_parent	0.6521	0.7329	0.8137
sexM	4.945	5.228	5.511

```
Mod_sim |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	17.84	18.07	18.3
mid_parent	0.7257	0.7292	0.7327

```
Mod_galton |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
898	2	0.6382	789.4	0.6374	0	2	895

```
Mod_sim |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
1e+05	1	0.6275	168464	0.6275	0	1	99998

Each successive generation relates to its parents similarly; for instance, the mid-child has children (the mid-grandchild) showing the same relationship.

```
sim_data |> model_train(mid_grandchild ~ mid_child) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	17.43	17.68	17.94
mid_child	0.7311	0.7349	0.7387

... and all generations have about the same mean height:

```
sim_data |>  
  summarize(mean(mid_parent), mean(mid_child), mean(mid_grandchild))
```

mean(mid_parent)	mean(mid_child)	mean(mid_grandchild)
66.71	66.72	66.71

However, the simulated variability *decreases* from generation to generation. That's unexpected, given that each generation relates to its parents similarly.

```
sim_data |>  
  summarize(var(mid_parent), var(mid_child), var(mid_grandchild))
```

var(mid_parent)	var(mid_child)	var(mid_grandchild)
3.098	2.626	2.396

To use Galton's language, this is "regression to mediocrity," with each generation being closer to the mean height than the parent generation.

i Note

FYI ... Phenotype vs genotype

Modern genetics distinguishes between the "phenotype" of a trait and the "genotype" that is the mechanism of heritability. The phenotype is not directly inherited; it reflects outside influences combined with the genotype. The above simulation reflects an early theory of inheritance based on "phenotype." (See Figure 45.) However, in part due to data like Galton's, the phenotype model has been rejected in favor of genotypic inheritance.

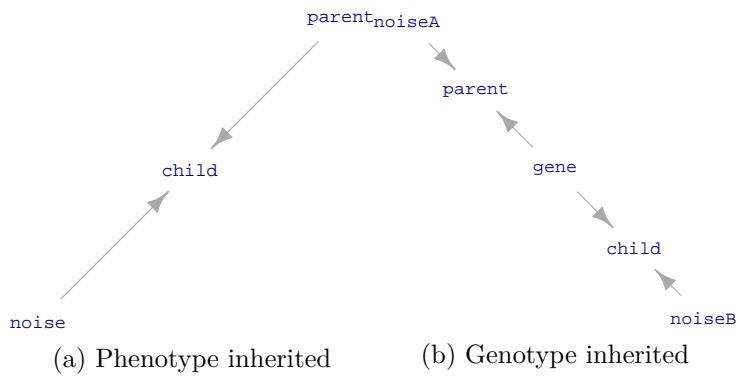


Figure 45: Two different models of genetic inheritance. The *phenotypic* model reflects very early ideas about genetics. The *genotypic* model is more realistic.

15 Models for noise

 Note in draft

Check `340-Degraded-Probability-models.qmd` to make sure everything has been folded into this Lesson.

We have been using a core framework for interpreting data: Each variable is a combination of signal and noise. The **signal** is that part of the variable that can be accounted for by other variables; the **noise** is another component that arises independently and is therefore unrelated to any other variables.

This is not intended to be a profound statement about the inner workings of the universe. The complementary roles of signal and noise is merely an accounting convention. Often, a source of variation that we originally take to be noise is found to be attributable to another variable; when this happens, the source of variation is transferred from the noise category to the signal category. As you will see in later Lessons, things also sometimes go the other way: we think we have attributed variation to a signal source but later discover, perhaps with more data, perhaps with better modeling, that the attribution is unfounded and the variation should be accounted as noise.

In this Lesson, we take a closer look at noise. As before, the hallmark of pure noise is that it is *independent* of other variables. However, now we focus on the *structure to noise* that corresponds to the setting in which the noise is generated.

It's likely that you have encountered such settings and their structure in your previous mathematics studies or in your experience playing games of chance. Indeed, there is a tradition in mathematics texts for using simple games to define a setting, then deriving the structure of noise—usually called **probability**—from the rules of the game. Example: Flipping a coin successively and counting the number of flips until the first head is encountered. Example: Rolling two dice then adding the two numerical outcomes to produce the result. The final product of the mathematical analysis of such settings is often a *formula* from which can be calculated the likelihood of any

specific outcome, say rolling a 3. Such a formula is an example of a “**probability model**.”

Our approach will be different. We are going to identify a handful of simple contexts that experience has shown to be particularly relevant. For each of these contexts, we will name the corresponding probability model, presenting it not as a formula but as a random number generator. To model complex contexts, we will create **simulations** of how different components work together.

15.1 Waiting time

Depending on the region where you live, a large earthquake is more or less likely. The timing of the next earthquake is uncertain; you expect it eventually but have little definite to say about when. Since earthquakes rarely have precursors, our knowledge is statistical, say, how many earthquakes occurred in the last 1000 years.

For simplicity, consider a region that has had 10 large earthquakes spread out over the last millenium: an average of 100 years between quakes. It’s been 90 years since the last quake. What is the probability that an earthquake will occur in the next 20 years? The answer that comes from professionals is unsatisfying to laypersons: “It doesn’t matter whether it’s been 90 years, 49 years, or 9 years since the last one: the probability is the same over any 20-year future period.” The professionals know that an appropriate probability model is the “**exponential distribution**.”

The exponential distribution is the logical consequence of the assumption that the probability of an event is independent of the time since the last event. The probability of an event in the next time unit is called the “**rate**.” For the region where the average interval is 100 years, the rate is $\frac{1}{100} = 0.01$ per year.

The `rexp()` function generates random noise according to the exponential distribution. Here’s a simulation of times between earthquakes at a rate of 0.01 per year. Since it is a simulation, we can run it as long as we like.

```

Quake_sim <- datasim_make(interval <- rexp(n, rate = 0.01))
Sim_data <- Quake_sim |> sample(n=10000)
Sim_data |>
  point_plot(interval ~ 1, annot = "violin",
             point_ink = 0.1, size = 0.1) |>
  add_plot_labels(y = "Years between successive earthquakes")

```

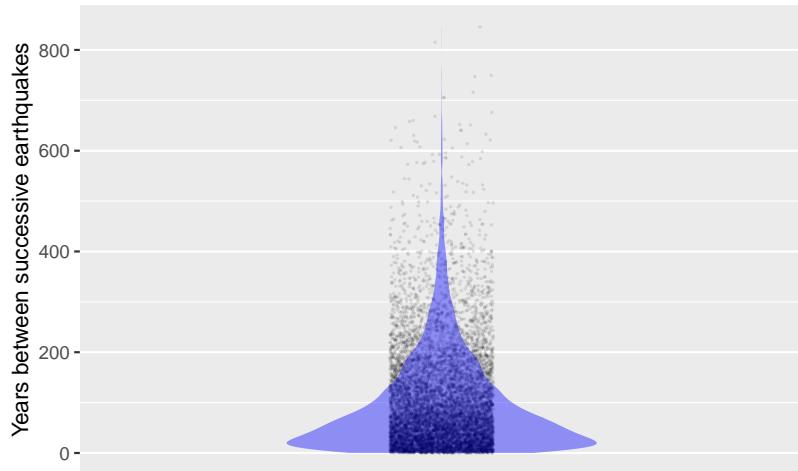


Figure 46: Interval between successive simulated earthquakes that come at a rate of 0.01 per year.

It seems implausible that the interval between 100-year quakes can be 600 years or even 200 years, or that it can be only a couple of years. But that's the nature of the exponential distribution.

The mean interval in the simulated data is 100 years, just as it's supposed to be.

```
Sim_data |> summarize(mean(interval), var(interval))
```

mean(interval)	var(interval)
100.3	9898

To illustrate the claim that the time until the next earthquake does not depend on how long it has been since the previous earthquake, let's calculate the time until the next earthquake for those intervals where we have already waited 100 years since the past one. We do this by filtering the intervals that last more than 100 years, then subtracting 100 years from the interval get the time until the end of the interval.

```
Sim_data |> filter(interval > 100) |>
  mutate(remaining_time = interval - 100) |>
  point_plot(remaining_time ~ 1, annot = "violin",
              point_ink = 0.1, size = 0.1) |>
  add_plot_labels(y = "Remaining time until earthquake")
```

Even after already waiting for 100 years, the time until the earthquake has the same distribution as the intervals between earthquakes.

15.2 Blood cell counts

A red blood cell count is a standard medical procedure. Various conditions and illnesses can cause red blood cells to be depleted from normal levels, or vastly increased. A hemocytometer is a microscope-based device for assisting counting cells. It holds a standard volume of blood and is marked off into unit squares of equal size. (Figure 48) The technician counts the number of cells in each of several unit squares and calculates the number of cells per unit of blood: the cell count.

The device serves a practical purpose: making counting easier. There are only a dozen or so cells in each unit square, the square can be easily scanned without double-counting.

Individual cells are scattered randomly across the field of view. The number of cells varies randomly from unit square to unit square. This sort of context for noise—how many cells in a randomly selected square—corresponds to the “**poisson distribution**” model of noise.

Any given poisson distribution is characterized by a *rate*. For the blood cells, the rate is the average number of cells per unit

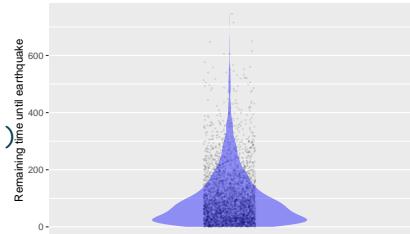


Figure 47: For those intervals greater than 100 years, the remaining time until the earthquake occurs.

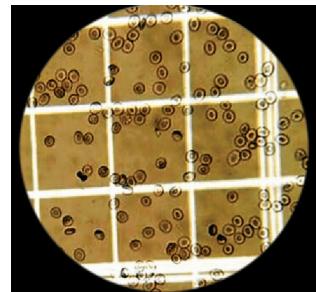


Figure 48: A microscopic view of red blood cells in a hemocytometer.
[Source](#)

square. In other settings, for instance the number of clients who enter a bank, the rate has units of customers per unit time.

The `rpois()` function generates random numbers according to the poisson distribution. The rate parameter is set with the `lambda =` argument.

Example: Medical clinic logistics

Consider a chain of rural medical clinics. As patients come in, they randomly need different elements of care, for instance a specialized antibiotic. Suppose that a particular type of antibiotic is called for at random, say, an average of two doses per week. This is a rate of $2/7$ per day. But in any given day, there's likely to be zero doses given, or perhaps one dose or even two. But it's unlikely that 100 doses will be needed. Figure 49 shows the outcomes from a simulation:

```
dose_sim <- datasim_make(doses <- rpois(n, lambda = 2/7))
Sim_data <- dose_sim |> sample(n = 1000)
Sim_data |> point_plot(doses ~ 1, point_ink = 0.1) |>
  add_plot_labels(y = "Doses given daily")
```

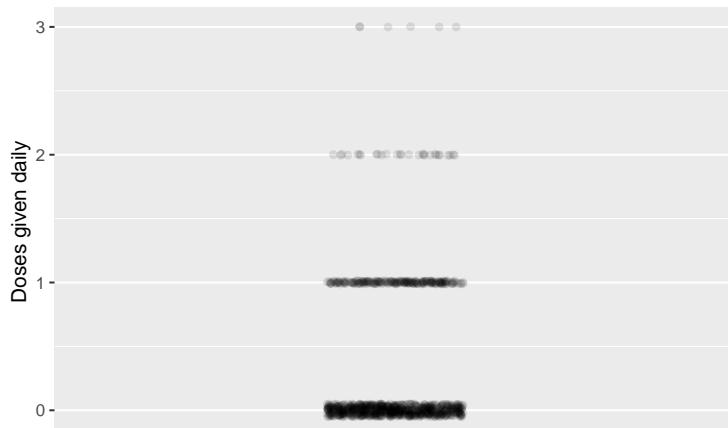


Figure 49: Simulation using `rpois()`.

```
Sim_data |>
  summarize(n(), .by = doses) |> arrange(doses)
```

doses	n()
0	754
1	212
2	28
3	6

```
Sim_data |>
  summarize(mean(doses), var(doses))
```

mean(doses)	var(doses)
0.286	0.2965

Even though, on average, less than one-third of a dose is used each day, on about 3% of days—one day per month—two doses are needed. Even for a drug whose shelf life is only one day, keeping at least two doses in stock seems advisable. To form a more complete answer, information about the time it takes to restock the drug is needed.

15.3 Adding things up

Another generic source of randomness comes from combining many independent sources of randomness. For example, in Section 14.3, the simulated height of a grandchild was the accumulation over generations of the random influences from each generation of her ancestors. A bowling score is a combination of the somewhat random results from each round. The eventual value of an investment in, say, stocks is the sum of the random up-and-down fluctuations from one day to the next.

The standard noise model for a sum of many independent things is the “**normal distribution**,” which you already met in Les-

son 14 as `rnorm()`. There are two parameters for the normal distribution, called the **mean** and the **standard deviation**. To illustrate, let's generate several variables, `x1`, `x2`, and so on, with different means and standard deviations, so that we can compare them.

```
Sim <- datasim_make(
  m1s0.2 <- rnorm(n, mean = 1, sd = 0.2),
  m2s0.4 <- rnorm(n, mean = 2, sd = 0.4),
  m0s2.0 <- rnorm(n, mean = 0, sd = 2.0),
  m1.5s1.3 <- rnorm(n, mean = -1.5, sd = 1.3)
)
Sim |> sample(n=10000) |>
  pivot_longer(everything(), values_to = "value", names_to = "var_name") |>
  point_plot(value ~ var_name, annot = "violin",
             point_ink = .05, model_ink = 0.7, size = 0.1)
```

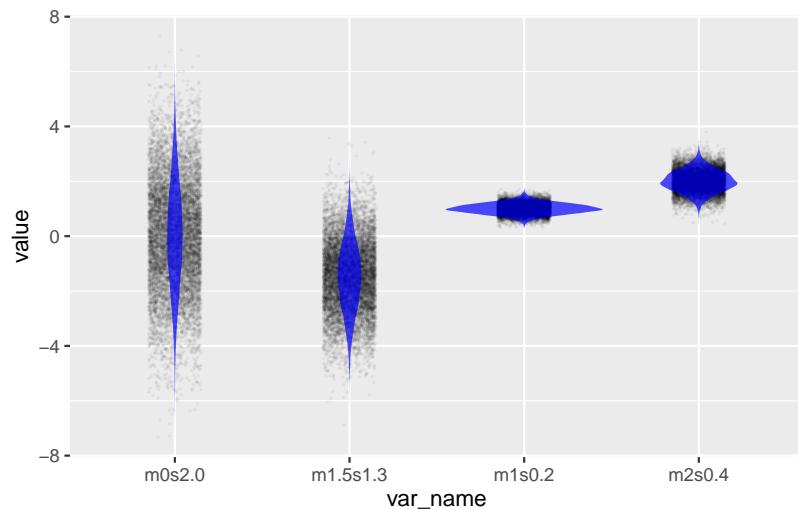


Figure 50: Four different normal distributions with a variety of means and standard deviations.

15.4 Other named distributions

There are many other named noise models, each developed mathematically to correspond to a real or imagined situation. Examples: chi-squared, t, F, hypergeometric, gamma, weibull,

beta. The professional statistical thinker knows when each is appropriate.

15.5 Relative probability functions

The thickness of a violin annotation indicates which data values are common, and which uncommon. A noise model is much the same when it comes to **generating** outcomes: the noise model tells which outcomes are likely and which unlikely.

The main goal of a statistical thinker or data scientist is usually to extract information from *measured* data—not simulated data. Measured data do not come with an official certificate asserting that the noise was created this way or that. Not knowing the origins of the noise in the data, but wanting to separate the signal from the noise, the statistical thinker seeks to figure out which forms of noise are most likely. Noise models provide one way to approach this task.

In simulations we use the **r** form of noise models—e.g., `rnorm()`, `rexp()`, `rpois()`—to create simulated noise. This use is about *generating* simulation data; we specify the rules for the simulation and the computer automatically generates data that complies with the rules.

To figure out from data what forms of noise are most likely, another form for noise models is important, the **d** form. The **d** form is not about generating noise. Instead, it tells how likely a given outcome is to arise from the noise model. To illustrate, let's look at the **d** form for the normal noise model, provided in R by `dnorm()`.

Suppose we want to know if -0.75 is a likely outcome from a particular normal model, say, one with mean -0.6 and standard deviation 0.2.. Part of the answer comes from a simple application of `dnorm()`, giving the -0.75 as the first argument and specifying the parameter values in the named arguments:

```
dnorm(-0.75, mean = -0.6, sd = 0.2)
```

```
[1] 1.505687
```

The answer is a number, but this number has meaning only in comparison to the values given for other inputs. For example, here's the computer value for an input of -0.25

```
dnorm(-0.25, mean = -0.6, sd = 0.2)
```

```
[1] 0.4313866
```

Evidently, given the noise model used, the outcome -0.25 is less likely outcome than the outcome -0.75.

A convenient graphical depiction of a noise model is to plot the output of the `d` function for a range of possible inputs, as in Figure 51:

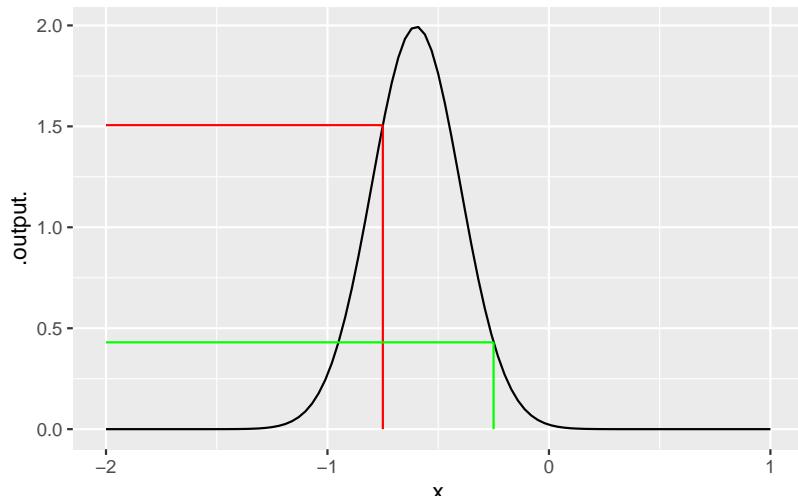


Figure 51: The function `dnorm(x, mean = -0.6, sd = 0.2)` graphed for a range of values for the first argument. The colored lines show the evaluation of the model for inputs -0.75 and -0.25.

This is **NOT** a data graphic, it is the graph of a mathematical function. Data graphics always have a variable mapped to `y`, whereas mathematical function are graphed with the function output mapped to `y` and the function input to `x`.

The output value of `dnorm(x)` is a **relative probability**, not a literal probability. Probabilities must always be in the range

0 to 1, whereas a relative probability can be any non-negative number. The function graphed in Figure 51 has, for some input values, output greater than 1. Even so, one can see that -0.75 produces an output about three times greater than -0.25.

The function-graphing convention makes it easy to compare different functions. Figure 52 shows the noise models from Figure 50 graphed as a function:

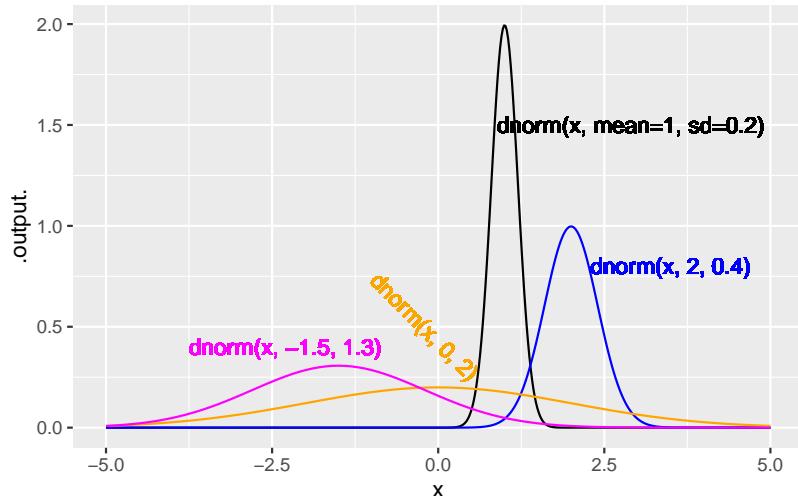


Figure 52: The four noise models from Figure 50 shown as functions.

16 Estimation and likelihood



Figure 53: A workplace safety sign.

A sign showing the number of days since the last accident is common at construction or industrial workplaces. Perhaps such signs are better than a generic “Accidents happen! Be careful!” The days-since sign always points to an actual accident, not just a theoretical possibility, and gives a small visual encouragement after each new accident-free day.

From Lesson 15, we have a model for the time between accidents: the exponential distribution. This is only a model. No law of physics says that accidents happen randomly at a given rate, nor is there a reason to think that every day or task is equally dangerous. Still, knowing about the exponential model helps to put the data—48 days in -Figure 53 —in a context. For instance, suppose the past average rate of accidents at the worksite was one per 10 days. Would the current tally of 48 days be good evidence that something has changed to make the worksite safer?

In principle, the days-since-the-last-accident indicator can be informative. For instance, if there had been 480 consecutive accident-free days many people would understandably conclude that the worksite is now safer than it was historically. But the situation is not always so clear: If an accident occurs only three days after the last, would it be fair to conclude that the worksite is more dangerous than in the days when accidents happened about once every ten days?

This Lesson introduces a technical concept, “**likelihood**,” that can provide a ready answer to such questions. We’ll define

“likelihood” here, but it will likely take some time to make sense of the definition.

Likelihood is a number in the same format as a probability. Likelihood comes into play *after* we have observed some data. With the data in hand, we consider one at a time each of a set of hypotheses. A hypothesis relevant to the workplace safety context would be an accident rate of 0.1 per day. Another hypothesis is an accident rate of 0.02 per day. Still another hypothesis is a rate of 0.13 accidents per day. There are many reasonable hypotheses, but for a likelihood calculation we take just one at a time. *In a world where the given hypothesis is true, the likelihood from that hypothesis is the probability of seeing the observed data.*

Note that the likelihood is based on an assumption: the given hypotheses. By comparing the likelihoods from different hypotheses we can get a handle on which hypotheses are more believable than others. It cannot be over-emphasized that a likelihood calculation is always rooted in a hypothesis. Various equivalent phrases can be used to describe the situation: the calculation is done “under a hypothesis,” or “given a hypothesis,” or “under the assumption that ...,” or, as we said above, “In a world where the given hypothesis is true.”

16.1 How likely?

Using the technology of noise models, it is comparatively easy to calculate a likelihood. The idea is to create a world where the given hypothesis is true and collect data from it. The tools for creating that world are mathematical or computational; we do not have to form a large sphere orbiting the sun.

An easy way to form such a hypothetical world is via simulation. For example, we know from Lesson 15 that it is conventional to use an exponential noise model to represent the duration of intervals between random events. If the world we want to create is that where the accident rate is 0.1 per day, we simply set the `rate` parameter of `rexp()` to that value when we generate data.

```
Accident_sim <- datasim_make(
  days <- rexp(n, rate = 0.1)
)
```

In the real world, it can take a long time to collect data, but with simulations it is practically instantaneous. Let's collect 10,000 simulated accidents from this made-up world where the accident rate is 0.1 per day:

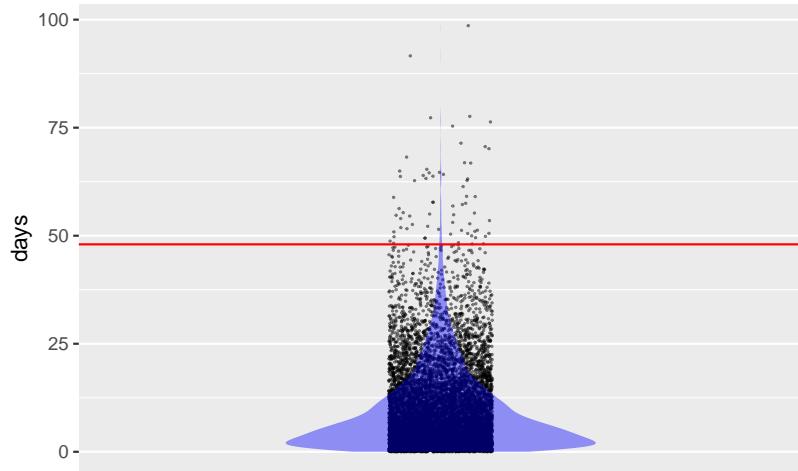


Figure 54: Times between accidents for 10,000 accidents, assuming a mean time between accidents of 10 days. The red line marks 48 days, the datum given in Figure 53.

Figure 54 shows that—if the accident rate were, as hypothesized, 0.1 per day—it's very unlikely for an accident-free interval to reach 48 days. The calculation is a simple matter of wrangling the simulated data:

```
Sim_data |> summarize(mean(days >= 48))
```

$$\begin{array}{c} \hline \text{mean}(days >= 48) \\ \hline \hline 0.0067 \end{array}$$

In a world where the accident rate were 0.1 per day, any given interval will be 48 days or longer with a probability near 1%.

To make use of a calculated likelihood, we need to compare it to something else, usually one or more other likelihoods calculated under different hypotheses.

16.2 Comparing different hypotheses using likelihood

To illustrate the use of likelihood, consider the seemingly simple context of deciding between two alternatives. I say “seemingly” because the situation is more nuanced than a newcomer might expect and will be dealt with in detail in the “Hypothetical Thinking” section of these Lessons.

Imagine the situation of an insurance company and a new driver. It’s reasonable to expect that some new drivers are better than others. Rough fairness suggests that prudent, careful, responsible drivers should pay less for insurance than risky drivers.

The insurance company has lots of data on new drivers insured over the last 20 years. The company can use this data—hundreds of thousands of drivers—to measure risk. The company’s actuaries discover that, using all manner of data, it can divide all the drivers into two groups. For one group—the high-risk group—the rate is one accident every 24 months. The low-risk group averages one accident per 72 months. (Remember, these are new drivers.)

The insurance company decides to use a framework of a *probationary period*. Initially, the driver is on probation. Insurance fees will be high, reflecting the overall riskiness of new drivers. After several months of accident-free driving without any moving violations, the insurance fee will be lowered. For the others, the insurance fee will go up.

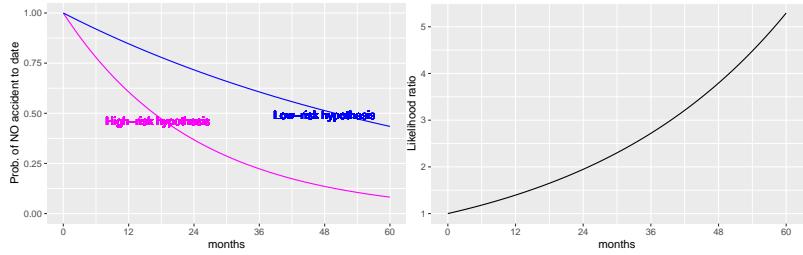
How long should the probationary period last? Likelihood provides an approach to answering the question.

The company approaches each new driver with two competing hypotheses: the high-risk hypothesis and the low-risk hypotheses. Initially, it doesn’t know which hypothesis to assign to an individual driver. It will base its eventual decision on the driver’s accumulated driving record. The duration of the probation period—how much time is accumulated without an

accident—will be set so that the likelihood of the low-risk hypothesis is twice that of the high-risk hypothesis.

We won't go into the detailed algebra of calculating the likelihood; the results are in Figure 55. There are two curves, each showing the probability of *not being in an accident* as a function of months driving. Why two curves? Because there are two *hypotheses* being entertained: the low-risk and the high-risk hypothesis.

Why twice? We will come back to this point after working through some details.



(a) Likelihood given each hypothesis (b) Likelihood ratio of the two hypotheses

Figure 55: Likelihood as a function of accident-free months driving for the low-risk and the high-risk hypotheses. The likelihood ratio compares the low-risk drivers to the low-risk drivers.

Initially, at zero months of driving, the probability of no accident to date is 1, regardless of which hypothesis is assumed. (If you haven't driven yet, you haven't been in an accident!) After 12 months of driving, about 60% of presumed high-risk drivers haven't yet been in an accident. For the presumed low-risk drivers, 85% are still accident-free.

At 24 months, only 37% of presumed high-risk drivers are accident-free compared to 72% of presumed low-risk drivers. Thus, at 24 months, the likelihood of the low-risk hypothesis is twice that of the high-risk hypothesis. A probationary period of 24 months matches the “twice the likelihood criterion” set earlier.

Why do we say, “presumed?” Individual drivers don't have a label certifying them to be low- or high-risk. The likelihoods refer to an imagined group of low-risk drivers and a different imagined group of high-risk drivers. The calculations behind

Figure 55 reason from the assumed hypothesis to whether it's likely to observe no accidents to date. But we *use* the calculations to support reasoning in the other direction: from the observed accident-free interval to a preference for one or the other of the hypotheses.

Let's be careful not to get ahead of ourselves. Likelihood calculations are an important *part* of statistical methods for making decisions, but they are hardly the only part. We are using a likelihood ratio of two in this example for convenience in introducing the idea of likelihood. A systematic decision-making process should look at the benefits of a correct classification and the costs of an incorrect one, as well as other evidence in favor of the competing hypotheses. We will see how to incorporate such factors in Lesson 28. In Lesson 29 we will see what happens to decision making when no such evidence is available or admissible.

i Distinguishing between “probability” and “likelihood”

A challenge for the statistics student when studying uncertainty is the many synonyms used in everyday speech to express quantitative uncertainty. For instance, all these everyday expressions mean the same thing:

- The *chance* of the picnic being cancelled is 70%.
- The *probability* of the picnic being cancelled is 70%.
- The *likelihood* of the picnic being cancelled is 70%.
- The *odds* of the picnic being cancelled are 70%.

The technical language of statistics makes important distinctions between *probability*, *likelihood*, and *odds*. We will leave “odds” for Lesson 21, when we discuss the accumulation of risk. For now, let's compare “probability” and “likelihood.”

“Probability” and “likelihood” have much in common. For instance, both are expressed numerically, e.g. 70% or 0.023. The difference between “probability” and “likelihood” involves

1. The kind of event they are used to describe

This is where we come back to “twice.”

This use of “odds” is mathematically incorrect, but common in practice.

If the chance is 70%, then the corresponding odds are 7-to-3. See Lesson 21.

2. The reasoning that lies behind them
3. The uses to which they are put

	Probability	Likelihood
Numerically	Between 0 and 1.	greater than or equal to zero
Event	An (as yet) uncertain outcome	An observed outcome
Logic	Based on mathematical axioms	Based on competing hypotheses
Use	e.g. prediction of an outcome	Evaluating observed data in terms of competing hypotheses

16.3 Likelihood functions (optional)

Likelihood calculations are widely used in order to estimate parameters of noise models from observed data. In Section 16.2 we looked at using the likelihood of observed data for each of two hypotheses. Parameter estimation—e.g. the rate parameter in the exponential or the poisson noise models—provides a situation where each numerical value is a potential candidate for the best parameter.

To help understand the reasoning involved, Figure 56 shows a typical graph for probability and another graph typical of likelihood.

Both graphs in Figure 56 show **functions**. A typical use for a **probability** function is to indicate what values of the **outcome** are more or less probable. The function can only be graphed when we **assume the parameter** for the noise model. Here, the assumed parameter is a rate of 0.1, that is, an average of one event every ten years. As anticipated for an exponential distribution, an interval of, say, 5 years is more probable than an interval of 10 years, which is more probable than an interval of 20 years.

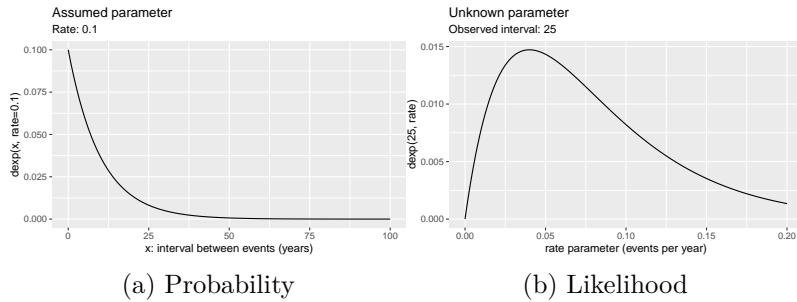


Figure 56: Probability versus likelihood, shown graphically.

In contrast, a typical use for a **likelihood** function is to figure out what **parameter values** accord more strongly than others with the observation. The likelihood function can only be graphed when we **know the observed value**. Here, the observed interval between events was 25 years. This single observation of an interval leads to a rate parameter of 0.04 being the most likely, but other rates are almost equally likely. Which rates are unlikely: below something like 0.005 or above something like 0.2 per year.

For a probability function, the interval duration is mapped to x . In contrast, for a likelihood function, the rate parameter is mapped to x .

Although the two graphs in Figure 56 have different shapes, they are both closely related to the same noise model. Recall that the R functions implementing the exponential noise model are `rexp(nsamps, rate=)` and `dexp(interval, rate=)`. The probability graph in Figure 56 shows the function `dexp(x, rate=0.1)` plotted against x . The likelihood graph, in contrast, shows the function `dexp(x=25, rate)` plotted against rate. The same `dexp(x, rate)` function is shown in both. What's different is which argument to `dexp(x, rate)` is set to a constant and which argument varies along the x -axis. In likelihood calculations, x is fixed at the *observed* value (after the event) and rate is left free to vary. In probability calculation, rate is fixed at the *assumed* value and x is left free to vary.

The fixed x value in a likelihood function comes from an observation from the real world: data. The observation is a measure-

ment of an event that's *already happened*. We use the observation to inform our knowledge of the parameter. On the other hand, the fixed parameter value in a probability calculation might come from anywhere: an assumption, a guess, a value our research supervisor prefers, a value made up by a textbook writer who wants to talk about probability.

16.4 Data narrows the likelihood function (optional)

The likelihood function in Figure 56 comes from a single observation of 25 year between events. A single observation can only tell you so much; more data tells you more. To see how this works with likelihood, we will play a game.

In this game I have selected a rate parameter. I'm not going to tell you what it is until later, but I will give you some data. Here are ten observed intervals (which I generated with `rexp(10, rate=____)`), where `_____` was filled in with my selected rate.

<hr/> <hr/> interval
100
270
36
27
140
42
27
54
120
34

The likelihood function for the first observation, 100.5, is `dexp(100.5, rate) ~ rate`. The likelihood for the second observation, 269.9, is `dexp(269.9, rate) ~ rate`. And so on for each of the ten observations.

When there are multiple observations, such as the 10 shown above, the likelihood of the whole set of observations is the *product* of the individual likelihoods. To illustrate, the first panel of Figure 57 shows the likelihood for the first ten observations.

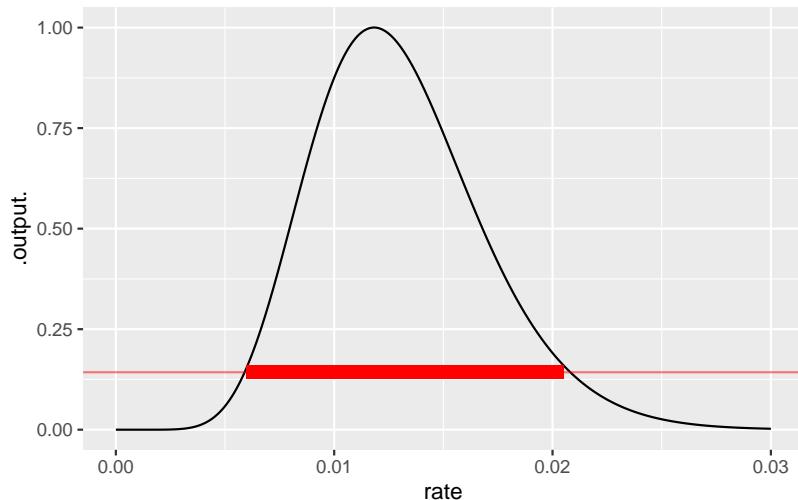
The peak is wide and falls off slowly from its maximum. After 30 observations, the peak is narrower. After 100 observations, narrower still.

The rate at which the maximum likelihood occurs gives the single most likely parameter value *given the observed data*. Notice in Figure 57 that the peak location shifts from panel to panel. This is natural since the data is different for each panel.

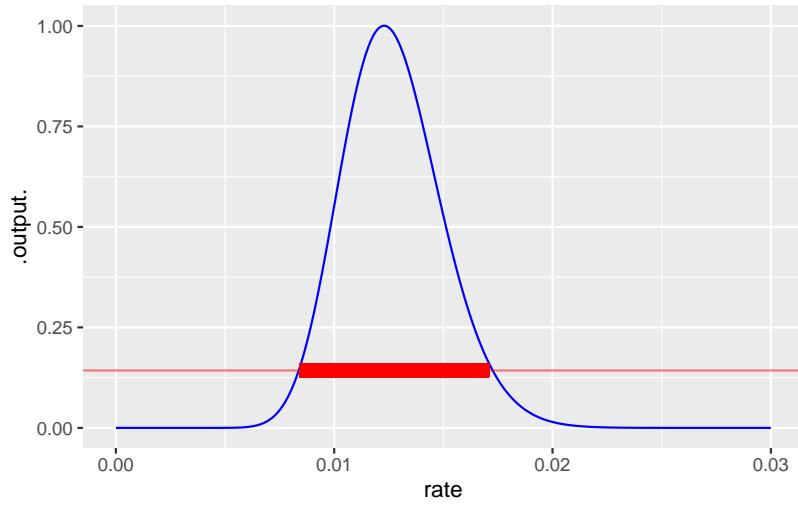
Finding the location of the maximum is straightforward, but the likelihood function can also be used to construct a band of rates which are all plausible given the data. This band (shown in red in Figure 57) corresponds to the *width* of the peak and is a standard way of indicating how precise a statement can be made about the parameter. More data rules out more possibilities for the parameter. A rough and ready rule for identifying the band of plausible parameters looks at the two parameter values where the likelihood falls to about 1/7 of its maximum value.

Now it's time to reveal the rate parameter used to generate the observations: 0.012. That is well inside each of the likelihood peaks.

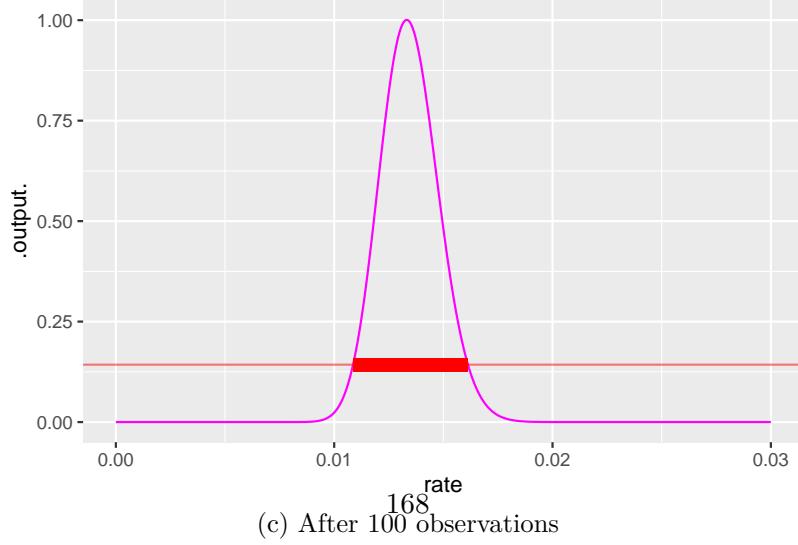
In the driver insurance example, we used a ratio of 1/2. Different ratios are appropriate for different purposes.



(a) First 10 observations



(b) After 30 observations



(c) After 100 observations
168

Figure 57: The likelihood calculated from multiple observations. The thin red line is drawn at $1/7$ the height of the peak.

17 R-squared and covariates

In Lessons 9 and 13, we introduced a commonly used measure, R^2 , which measures how much variation in a response variable a model can be accounted for using the explanatory variables.

i Do we need R^2 ?

If you read scientific papers that involve analysis of data, you are likely to encounter R^2 (or its little cousin, r). R^2 and r have a long history. r was [introduced in 1888](#) as a way of quantifying the relationship between two quantitative variables. [R² was introduced in 1921](#) as linear regression techniques started to get traction in statistical methodology.

Such a long pedigree may explain why R^2 is used so much in research literature. If only for this reason, you must learn about R^2 . R^2 also offers a window into a statistical artifact that surprises most newcomers to statistics, which you will read about in this Lesson.

For models with only quantitative explanatory variables or zero-one explanatory variables, model coefficients, and confidence intervals provide more information, but even so, R^2 can highlight when explanatory variables overlap. And when there are categorical explanatory variables, particularly ones with many levels, R^2 can genuinely contribute to understanding how much each explanatory variable contributes to a model. But some care is needed to make sure that R^2 is put in the context of the sample size and the number of coefficients in a model. We'll get to that.

17.1 Fraction of variance explained

R^2 has a simple-sounding interpretation in terms of *how much* of the variation in a response variable is accounted for by the explanatory variables.

Recall that we measure variation using the **variance**. R^2 compares the variance “captured” by the explanatory variable to

the amount of variation in the response variable on its own. To illustrate, consider the `Hill_races` data frame. Taking the winning running `times` as the response variable, we might wonder how much of the variation in time is accounted for by the race characteristics, for instance by the length of the race course (in km).

Here's the variance of the `time` variable:

```
Hill_racing |> summarize(var(time, na.rm = TRUE), sd(time, na.rm = TRUE))
```

var(time, na.rm = TRUE)	sd(time, na.rm = TRUE)
9754276	3123

As always, the units of the variance are the square of the units of the variable. Since `time` is in seconds, `var(time)` has units of “seconds-squared.” The standard deviation, which is the square root of the variance, is often easier to understand as an “amount.” That the standard deviation is about 3000 s, about an hour, means that the running times of the various races collected in `Hill_racing` range over hours: very different races are included in the data frame.

Naturally, the races differ from one another. Among other things, they differ in `distance` (in km). We can model `time` versus `distance` and look at the coefficients:

```
Hill_racing |> model_train(time ~ distance) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-296.1	-210.9	-125.7
distance	374.5	381	387.6

The units of the `distance` coefficient are seconds-per-kilometer (s/km). Three hundred eighty seconds per kilometer is a pace slightly slower than six minutes per km, or about ten miles per hour: a ten-minute mile. These are the winning times in the

races. You might be tempted to think that these races are for casual runners.

R^2 provides another way to summarize the model.

```
Hill_racing |> model_train(time ~ distance) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2226	1	0.8548	13096	0.8548	0	1	2224

The R^2 for the model is 0.85. A simple explanation is that the race distance explains 85% of the variation from race to race in running time: the large majority. This is no surprise to those familiar with racing: a 440 m race takes much less time than a 10,000-meter race. What might account for the other 15% of the variation in time? There are many possibilities.

An important feature of Scottish hill racing is the ... hills. Many races feature substantial climbs. How much of the variation in race `time` is explained by the height (in m) of the `climb`? R^2 provides a ready answer:

```
Hill_racing |> model_train(time ~ climb) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2224	1	0.765	7234	0.7649	0	1	2222

The height of the `climb` *also* explains a lot of the variation in `time`: about three-quarters of it.

To know how much of the `time` variance `climb` and `distance` together explain, don't simply add together the individual R^2 . By trying it, you can see why in this case: the amount of variation explained is $85\% + 76\% = 161\%$. That should strike you as strange! No matter how good the explanatory variables, they can never explain more than 100% of the variation in the response variable.

The source of the impossibly large R^2 is that, to some extent, both `time` and `climb` share in the explanation; the two explanatory variables each explain much the same thing. We avoid such double-counting by including both explanatory variables at the same time:

```
Hill_racing |> model_train(time ~ distance + climb) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2224	2	0.9223	13187	0.9223	0	2	2221

Taken together, `distance` and `climb` account for 92% of the variation in race `time`. This leaves at most 8% of the variation yet to be explained: the **residual variance**.

17.2 R^2 and categorical explanatory variables

Consider this question: Does the `name` of the race (along with 565 other race names recorded as the `race` variable) influence the running time for the race?

Here's a simple model:

There are 154 levels in the `race` variable, which records the name of the race. (`name` is the name of the runner.) Examples: Glen Rosa Horseshoe, Ben Nevis Race, ...

```
Race_name_model1 <- Hill_racing |> model_train(time ~ race)
```

R^2 offers a much easier-to-interpret summary than the model coefficients in this situation. Here are the model coefficients:

```
Race_name_model1 |> conf_interval() -> Goo
```

term	.lwr	.coef	.upr
(Intercept)	1300	2010	2710
raceAlex Brett Cioch Mhor	1950	2710	3480
raceAlva Games Hill Race	-1340	-580	182
raceAonach Mor UKA event (men)	-1600	-23.7	1550
raceAonach Mor UKA event (women)	-893	329	1550

term	.lwr	.coef	.upr
raceAonach Mor Uphill	-1060	-224	612

The reference race is the Aberfeldy Games Race. (Why? Because Aberfeldy is first alphabetically.) Each coefficient on another race gives a result by comparison to Aberfeldy.

The question that started this section was not about the Alva Games Hill Race, the Aonach Mor Uphill, or any of the others, but about the whole *collection* of differently named races. The R^2 summary brings all the races together to measure the amount of `time` variance “explained” by the race names:

```
Race_name_model1 |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2226	153	0.9505	260.3	0.9469	0	153	2072

This is a strikingly large R^2 . Based on this, some people might be tempted to think that a race’s name plays a big part in the race outcome. Statistical thinkers, however, will wonder whether the race name encodes other information that explains the race outcome. For example, we can look at how well the race name “explains” the race distance and climb:

```
Hill_racing |> model_train(distance ~ race) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2236	153	1	Inf	1	0	153	2082

```
Hill_racing |> model_train(climb ~ race) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2234	152	1	Inf	1	0	152	2081

The race name “explains” 100% of the variation for both ‘distance’ and ‘climb’. That’s because each distinct race has its own `distance` and `climb`. So, the race name carries all the information in the `distance` and `climb` variables.

By *adjusting* for `distance` and `climb`, we can ask more focused questions about the possible role of the race name in determining. First, recall that just `distance` and `climb` account for 92% of the variance in `time`.

```
Hill_racing |> model_train(time ~ distance + climb) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2224	2	0.9223	13187	0.9223	0	2	2221

Adding in `race` increases the R^2 by only three percentage points, to 95%. (See exercise .)

17.3 Degrees of freedom

Using categorical variables with a large number of levels are used as explanatory variables, a new phenomenon becomes apparent, a sort of mirage of explanation. To illustrate, consider the model `time ~ name`. There are five-hundred sixty-seven unique runners in the `Hill_racing` data.

```
Hill_racing |> model_train(time ~ name) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2226	565	0.4294	2.211	0.2351	0	565	1660

The runner’s identity accounts for about 43% of the variance in running `time`. Understandably, the R^2 is not much higher: runners participate in multiple races with different `distances` and `climbs` so it’s natural for an individual runner to have a spread of running `time`.

Let's experiment to illustrate the difficulty of interpreting R^2 when there are many levels in a categorical explanatory variable. We will create a new variable consisting only of random noise.

```
Hill_racing <- Hill_racing |> mutate(noise = rnorm(n()))
```

Naturally, there is no genuine explanation of noise. For instance, `distance` and `climb` account for 92% of the actual running times, but a trivial percentage of the `noise`:

```
Hill_racing |> model_train(noise ~ distance + climb) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2234	2	0.00177	1.978	0.0008755	0.1385	2	2231

In contrast, `name`, with its 567 different levels, seems to “explain” a lot of `noise`:

```
Hill_racing |> model_train(noise ~ name) |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
2236	566	0.2468	0.9661	-	0.6927	566	1669

The 567 names explain about one-quarter of the variance in `noise`, which ought not to be explainable at all!

How can `name` explain something that it has no connection with? First, note that the `Hill_racing` sample size is $n=2236$. (You can see the sample size in all R^2 reports under the name `n`.) When we fit the model `noise ~ name`, there will be 567 different coefficients, one of which is the intercept and 566 of which relate to `name`. This number—labelled `k` in the R^2 report—is called the “**degrees of freedom**” of the model.

In general, models with more degrees of freedom can explain more of the response variable, even when there is nothing to

explain. On average, the R^2 in a nothing-to-explain situation will be roughly k/n . For the `noise ~ name` model, the k -over- n ratio is $566/2236 = 0.25$.

i Small data

In some situations, a sample may include just a handful of specimens, say $n = 5$. A simple model, such as $y \sim x$, will have a small number of degrees of freedom. With $y \sim x$, there are two coefficients: the intercept and the coefficient on x . With only a single non-intercept coefficient, the model degrees of freedom is $k = 1$.

Nonetheless, the typical R^2 from such a model, even when y and x are completely unrelated, will be at least $k/n = 0.20$. It's tempting to interpret an R^2 of 0.20 as the sign of a relationship between y and x . To avoid such misinterpretations, statistical formulas and software carefully track k and n and arrange things to compensate.

One simple compensation for model degrees of freedom is "**adjusted R^2** ." The adjustment is roughly this: take R^2 and subtract k/n . Insofar as there is no relationship between the response and explanatory variables, this will bring R^2 down to about zero. An adjusted R^2 greater than zero indicates a relationship between the response and explanatory variables. Adjusted R^2 is useful when the goal is to ascertain whether there is a *substantial* relationship. This goal is common in fields such as econometrics.

Statistics textbooks favor other styles of adjustment that are, perhaps surprisingly, not oriented to pointing to a substantial relationship. A famous style of adjustment is encapsulated in the **t statistic**, which applies to models with only a single degree of freedom. A generalization of t to models with more degrees of freedom is the **F statistic**.

18 Predictions

Everyday life is awash in predictions. Weather **forecasts** state the chances of rain, usually as a percentage: 0%, 10%, 20%, ..., 90%, 100%. We bring in a car for repairs and are given an **estimate** for the eventual bill. Doctors often give patients a **prognosis** which can come in a form like “you should be feeling better in a few days” or, for severe illnesses such as cancer, a 5-year survival rate. Economists offer their **informed guesses** about the direction that the economy is heading: unemployment will go down, interest rates up. In horse racing, the betting odds signal a **hunch** about the eventual outcome of a race.

In every case, a prediction refers to an “**event**” from which multiple outcomes are possible. Your team may win or lose in *next Sunday’s game*. It might rain or not *next Tuesday*. The events in these simple examples are your team’s performance in the specific game to be played on the upcoming Sunday or the precipitation next Tuesday. Events can also refer to extended periods of time, for instance the forecast for the number of hurricanes next year.

18.1 Statistical predictions

A “**statistical prediction**” has a special form not usually present in everyday, casual predictions. A statistical prediction assigns a number to every possible outcome of an event. The number is a *relative probability*. For example, a casual prediction of the outcome of next Sunday’s game might be “We will win.” A statistical prediction assigns a number to each possible outcome, for instance: win 5, lose 4 which signals that winning is only slightly more probable than losing.

When there are just two possible outcomes, people often prefer to state the probability of one outcome, leaving the probability of the other outcome implicit. A prediction of win 5, lose 4 translates to a 5/9 probability of winning, that is, 55.6%. The implicit probability of the other outcome, losing, is 1 - 55.6% or 44.4%.

Admittedly, saying, “The probability of winning is 55.6%,” is pretty much equivalent to saying, “The game could go either way.” Indeed, what could justify the implied precision of the number 55.6% is not apparent when, in fact, the outcome is utterly unknown.

The numerical component of a statistical prediction serves three distinct tasks. One task is to convey uncertainty. For a single event’s outcome (e.g., the game next Sunday), the seeming precision of 55.6% is unnecessary. The uncertainty in the outcome could be conveyed just as well by a prediction of, say, 40% or 60%.

A second task is to signal when we are saying something of substance. Suppose your team hardly ever wins. A prediction of 50% for win is a strong indication that the predictor believes that something unusual is going on. Perhaps all the usual players on the other team have been disabled by flu and they will field a team of novices. Signaling “something of substance” relies on comparing a prior belief (“your team hardly ever wins”) with the prediction itself. This comparison is easiest when both the prediction and the prior belief are represented as numbers.

Yet a third task has to do with situations where the event is repeated over and over again. For instance, the probability of the house (casino) winning in a single spin of roulette (with 0 and 00) is 55%. For a single play, this probability provides entertainment value. Anything might happen; the outcome is entirely uncertain. But for an evening’s worth of repeated spins, the 55% probability is a guarantee that that the house will come out ahead at the end of the night.

For a *categorical* outcome, it’s easy to see how one can assign a relative probability to each possible outcome. On the other hand, for a *numerical* outcome, there is a theoretical infinity of possibilities. But we can’t write down an infinite set of numbers!

The way we dealt with numerical outcomes in Lesson 15 was to specify a noise model along with specific numerical parameters. And that is the common practice when making predictions of numerical outcomes. An example: Rather than predicting the

win/lose outcomes of a game, we might prefer to predict the “point spread,” the numerical difference in the teams scores. The form of a prediction could be: “My statistical prediction of the point spread is a normal probability model with a mean of 3 points and a standard deviation of 5 points.”

As a shorthand for stating a probability model and values for parameters, it’s common to state statistical predictions of numerical outcomes as a “**prediction interval**,” two numbers that define a range of outcomes. The two numbers come from a routine calculation using probability models. Routinely, the two numbers constitute a 95% prediction interval, meaning that a random number from the noise model will fall in the interval 95% of the time.

Intervals from a noise model

Consider a prediction of a numerical outcome taking the form of a normal noise model with these parameters: mean 10 and standard deviation 4. Such a prediction is saying that any outcome such as generated by `rnorm(n, mean=10, sd=4)` is equally likely. Figure 58 shows a set of possible outcomes. The prediction is that any of the dots in panel (a) is equally likely.

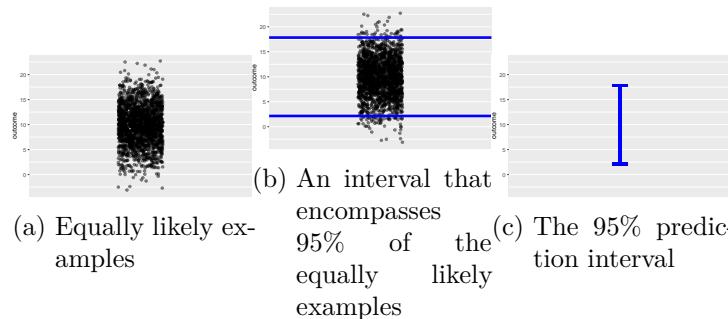


Figure 58: Presentations for a prediction of `rnorm(n, mean=10, sd=4)`

The upper and lower ends of the prediction interval are not hard boundaries; outcomes outside the interval are possible. But such outcomes are uncommon, happening

in only about one in twenty events.

18.2 Prediction via statistical modeling

The basis for a statistical prediction is *training data*: a data frame whose unit of observation is an event and whose variables include the event's outcome and whatever explanatory variables are to be used to form the prediction. It is up to the modeler to decide what training events are relevant to include in the training data, but all of them must have available values for the outcome.

There are, of course, other forms of prediction. A *mechanistic prediction* is based on "laws" or models of how a system works. Often, mechanistic predictions use a small set of data called "initial conditions" and then propagate these initial conditions through the laws or models. An example is a prediction of the location of a satellite, which draws on the principles of physics.

Much of the process of forming a statistical prediction is familiar from earlier Lessons. There is a *training phase* to prediction in which the training data are collected and a model specification is proposed.

The response variable in the model specification will be the variable recording the outcome of the training events. As for the explanatory variables, the modeler is free to choose any that she thinks will be informative about the outcome. The direction of causality is not essential when creating a prediction model. Indeed, some of the best prediction models can be made when the explanatory variables are a *consequence* of the response variable to be predicted. The training phase is completed by training the model on the training events—we will call it the "**prediction model**"—and storing the model for later use. As usual, the prediction model includes the formula by which the model output is calculated, but more is needed. In particular, the model includes *information about the residuals identified in the fitting process*. For instance, the prediction model might store the *variance* of the residuals.

The *application phase* for a prediction involves collecting “*event data*” about the particular event whose outcome will be predicted. Naturally, these event data give values for the explanatory variables in the prediction model. However, the value of the response variable is unknown. (If it were known, there would be no need for prediction!) The prediction model is evaluated to give a model output. The full prediction is formed by combining the model output with the information about residuals stored in the prediction model.

To illustrate, we will use the `Anthro_F` data frame that records, for 184 individuals, various body measurements such as wrist circumference, height, weight, and so on. Almost all the measurements were made with readily available instruments: a weight scale, a ruler, and a flexible sort of ruler called a measuring tape. But one of the measurements is more complex: `BFat` is the amount of body fat in proportion to the overall weight. It is calculated from the *density* of the body. Density is body volume divided by weight; measuring volume involves a water immersion process, depicted in Figure 59.

It is unclear what genuine medical or athletic-training value the body-fat measurement might have, but some people fix on it to describe overall “fitness.” The difficulty of the direct measurement (Figure 59) motivates a search for more convenient methods. We will look at calculating body fat percentage using a formula based on easy-to-make measurements such as weight and waist circumference.

This is a *prediction* problem because the body fat percentage is unknown and we want to say something about what it would likely be if we undertook the difficult direct measurement. It might be more natural to call this a *translation* problem; we translate the easy-to-make measurements into a difficult-to-make measurement. Indeed, prediction models are a common component of artificial intelligence systems to recognize human speech, translate from one language to another, or even the ever-popular identification of cat photos on the internet.

To build the prediction model, we need to provide a model specification. There are many possibilities: any specification with `BFat` as the response variable. Data scientists who build prediction models often put considerable effort into identifying

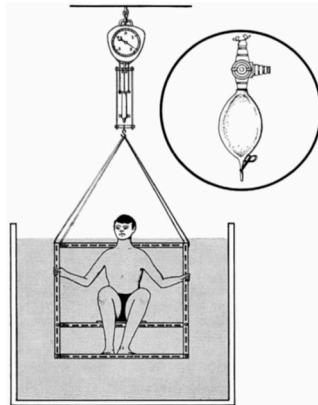


Figure 59: Diagram of the apparatus for measuring body volume. The inset shows a secondary apparatus for measuring the air remaining in the lungs after the subject has breathed out as far as practicable. Source: Durnin and Rahama (1967) *British Journal of Nutrition* 21: 681

suitable model specifications, a process called “**feature engineering**.” For simplicity, we will work with `BFat ~ Weight + Height + Waist`. Then, we fit the model and store it for later use:

```
BFat_mod <- Anthro_F |> model_train(BFat ~ Weight + Height + Waist)
```

Now, the application phase. A person enters the fitness center eager to know his body fat percentage. Lacking the apparatus for direct measurement, we measure the explanatory variables for the prediction model:

- Subject: John Q.
- Waist: 67 cm
- Weight: 60 kg
- Height: 1.70 m

To make the prediction, evaluate the prediction model on these values:

```
BFat_mod |> model_eval(Waist=67, Weight=60, Height=1.70)
```

Waist	Weight	Height	.lwr	.output	.upr
67	60	1.7	12.93	20.15	27.37

A statistically naive conclusion is that John Q’s body fat percentage is 20. Since the `BFat` variable in `Anthro_F` is recorded in percent, the prediction will have those same units. So John Q. is told that his body fat is 20%.

The statistical thinker understands that a prediction of a numerical outcome such as body fat percentage ought to take the form of a noise model, e.g. a normal noise model with mean 20% and standard deviation 3.5%. The `model_eval()` function is arranged to present the noise model as a prediction interval so that the prediction would be stated as 13% to 27%. These are the numbers reported in the `.lwr` and `.upr` columns of the report generated by `model_eval()`.

i How good is the prediction?

Figure 60 shows the training data values for BFat. These are authentic values, but it is correct as well to think of them as equally-likely *predictions* from a no-input prediction model, $\text{BFat} \sim 1$. The story behind such a no-input prediction might be told like this: “Somebody just came into the fitness center, but I know nothing about them. What is their body mass?” A common sense answer would be, “I have no idea.” But the statistical thinker can fall back on the patterns in the training data.

The red I shows the no-input prediction translated into a 95% prediction interval.

```
Anthro_F |> point_plot(BFat ~ 1) |>
  gf_errorbar(13 + 27 ~ 0.8, color="blue", width=0.1) |>
  gf_errorbar(33 + 10.5 ~ 1.2, color = "red", width = 0.1)
```

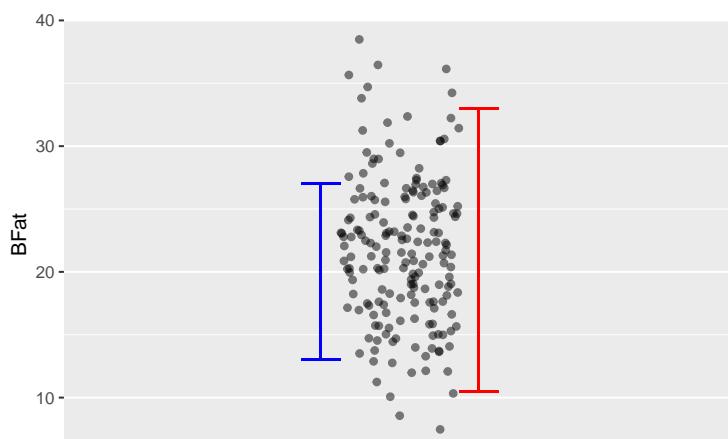


Figure 60: The prediction interval (blue I) overlaid on the training data values for BFat. The red I marks the prediction interval for the model $\text{BFat} \sim 1$, which does not make use of any measurements as input.

The blue I shows the 95% prediction interval for the model $\text{BFat} \sim \text{Weight} + \text{Height} + \text{Waist}$. The blue I is clearly

A no-input prediction model is sometimes called a “Null model,” the “null” indicating the lack of input information. We will return to “null” in Lesson 29.

shorter than the red I; the input variables provide some information about BFat.

Whether the prediction is helpful for Joe Q depends on context. For instance, whether Joe Q or his trainer would take different action based on the blue I than he would for the red I interval. For example, would Joe Q., as a fitness freak, say that the prediction indicates that he has his body fat at such a good value that he should start to focus on other matters of importance, such as strength or endurance.

Such decision-related factors are the ultimate test of the utility of a prediction model. Despite this, some modelers like to have a way to measure a prediction's quality without drawing on context. A sensible choice is the ratio of the length of the prediction interval compared to the length of the no-input prediction interval. For example, the blue interval in Figure 60 is about 60% of the length of the red, no-input interval. Actually, this ratio is closely related to the prediction model's R^2 , the ratio being $\sqrt{1 - R^2}$.

Another critical factor in evaluating a prediction is whether the training data are relevant to the case (that is, Joe Q.) for which the prediction is being made. That the training data were collected from females suggests that there is some sampling bias in the prediction interval for Joe Q. Better to use directly relevant data. For Joe Q.'s interests, perhaps much better data would be from males and include measurement of their fitness-freakiness.

In everyday life, such “predictions” are often presented as “measurements.” Ideally, all measurements should come with an interval. This is common in scientific reports, which often include “error bars,” but not in everyday life. For instance, few people would give a second thought about Joe Q.’s height measurement: 1.70 meters. But height measurements depend on the time of day and the skill/methodology of the person doing the measurement. More likely, the Joe Q measurement should be 1.70 ± 0.02 meters. Unfortunately, even in technical areas such as medicine or economics, measurements typically are not re-

ported as intervals. Keep this in mind next time you read about a measurement of inflation, unemployment, GDP, blood pressure, or anything else.

Consider the consequences of a measurement reported without a prediction interval. Joe Q might be told that his body fat has been measured at 20% (without any prediction interval). [Looking at the internet](#), Joe might find his 20% being characterized as “acceptable.” Since Joe wants to be more than “acceptable,” he would ask the fitness center for advice, which could come in the form of a recommendation to hire a personal trainer. Had the prediction interval been reported, Joe might desist to take any specific action based on the measurement and might (helpfully) call into question whether body fat has any useful information to convey beyond what’s provided by the easy-to-measure quantities such as weight and height.

I am not endorsing such internet statements. Experience suggests they should be treated with extreme or total skepticism.

18.3 The prediction interval

Calculation of the prediction interval involves three components:

1. The model output as calculated by applying the model function to the prediction inputs. This is reported, for example, in the `.output` column from `model_eval()`. The model output tells where to center the prediction interval.
2. The size of the residuals from the model fitting process. This is usually the major component of the length of the prediction interval. For instance, if the variance of the residuals is 25, the length of the prediction interval will be roughly $4 \times \sqrt{25}$.
3. The length of the confidence interval for example as reported in the model annotation to `point_plot()`. This usually plays only a minor part in the prediction interval.

The `.lwr` and `.upr` bounds reported by `model_eval()` take all three factors into account.

Be careful not to confuse a confidence interval with a prediction interval. The prediction interval is always longer, usu-

ally much longer. To illustrate graphically, Figure 61 shows the confidence and prediction intervals for the model $\text{BFat} \sim \text{Waist} + \text{Height}$. (We are using this simpler model to avoid overcrowding the graph. In practice, it's usually easy to read the prediction interval for a given case from the `model_eval()` report.)

```
Pred_model <- Anthro_F |> model_train(BFat ~ Waist + Height)
Pred_model |> model_plot(interval = "confidence", model_ink = 0.3)
Pred_model |> model_plot(interval = "prediction", model_ink = 0.3)
```

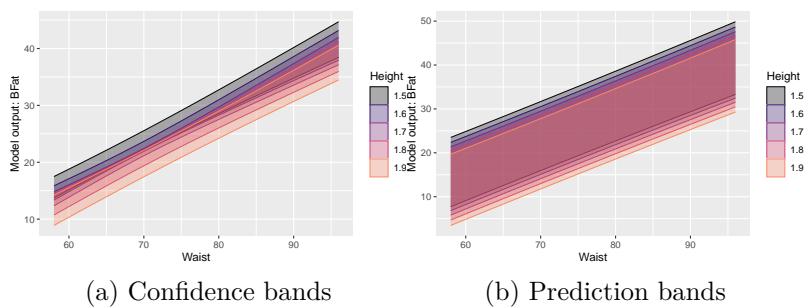


Figure 61: Confidence and prediction bands from the model
 $\text{BFat} \sim \text{Waist} + \text{Height}$

⋮

Unfortunately, many statistics texts use the phrase “predicted value” to refer to what is properly called the “model value.” Any predicted value in a statistics text ought to include a prediction interval. Since texts often report only confidence intervals, it’s understandable that students confuse the confidence interval with the prediction interval. This is entirely misleading. The confidence interval gives a grossly rosey view of prediction; the much larger prediction interval gives a realistic view.

18.4 Form of a statistical prediction: Categorical outcome

As stated previously, the proper form for a statistical prediction is assigning a relative probability to each possible outcome. For **quantitative** response variables, such assignment is described

by a noise model, but usually, a shorthand in the form of a “prediction interval” is used to summarize the noise model.

When the response variable is **categorical**, a statistical prediction takes the form of a list of relative probabilities, one for each level of the response variable. What’s potentially confusing here is that there is no “prediction interval” when presenting a prediction of a categorical variable, just the single number assigned to each level of the response variable.

In these Lessons, we treat only one kind of categorical response variable: one with two levels, which can therefore be converted to a zero-one variable. This enables us to use regression models in much the same way as for quantitative response variables. We typically use different regression methods for a quantitative response than a zero-one response. Quantitative response variables usually call for a *linear* regression method, while *logistic* regression is used for a zero-one response variable.

Although these Lessons emphasize zero-one response variables, building models of multi-level categorical response variables is also possible. We won’t go into detail here, but such models are called “**classifiers**” rather than regression models. A classifier output is already in the proper format for prediction: assignment of a relative probability to each possible level of the response.

Returning to zero-one response variables, we will illustrate the prediction process using a classic setting for zero-one variables: mortality.

The **Whickham** data frame comes from a one-in-six survey, conducted in 1972-1974, of female registered voters in a mixed urban and rural district near Newcastle upon Tyne, UK. Two observables, age and smoking status, were recorded. The outcome of interest was whether each participant would die within the next 20 years. Needless to say, all the participants were alive at the time of the survey.

With the `age` and `smoker` observables alone, building a meaningful prediction model of 20-year mortality is impossible. There is a vast *sampling bias* since all the survey participants were alive during data collection. To assemble training data, it was necessary to wait 20 years to see which participants

Table 63: A few cases from the `Whickham` training data.

age	smoker	outcome
37	No	Alive
23	No	Alive
56	Yes	Alive
75	Yes	Dead
67	No	Dead

remained alive. This `outcome` was recorded in a follow-up survey in the 1990s. `Whickham` is the resultant training data.

With the training data in hand, we can build a prediction model. Naturally, the `outcome` is the response variable. Based on her insight or intuition, the modeler can choose which explanatory variables to use and how to combine them. For the sake of the example, we'll use both predictor variables and their *interaction*.

```
Whickham |>
  point_plot(outcome ~ age * smoker, annot = "model",
             point_ink=0.3, model_ink=0.7)
```

Figure 62 shows the `Whickham` data and the `mortality ~ age * smoker` prediction model constructed from it. The model is shown, as usual, with confidence bands. But that is not the appropriate form for the prediction.

To get the prediction, we simply train the model ...

```
pred_model <- Whickham |>
  mutate(mortality = zero_one(outcome, one="Dead")) |>
  model_train(mortality ~ age * smoker)
```

... and apply the model to the predictor values relevant to the case at hand. Here, for illustration, we'll predict the 20-year survival for a 50-year-old smoker. (Since all the `Whickham` data is about females, the prediction is effectively for a 50-year-old female.)

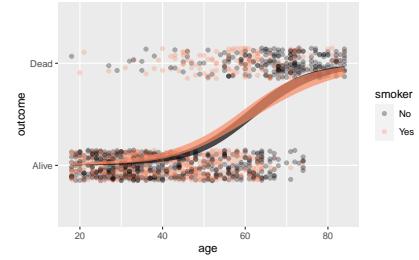


Figure 62: The `Whickham` training data and the prediction model constructed from it.

```
pred_model |> model_eval(age = 50, smoker = "Yes", interval = "none")
```

age	smoker	.output
50	Yes	0.24

The output of `model_eval()` is a data frame that repeats the values we gave for the predictor variables `age` and `smoker` and gives a model output (`.output`) as well. Since Whickham's `mortality` variable is a two-level categorical variable, logistic regression was used to fit the model and the model output will always be between 0 and 1. We interpret the model output as the probability that the person described by the predictor values will die in the next 20 years: 24%.

The ideal form of a prediction for a categorical outcome lists every level of that variable and assigns a probability to each. In this case, since there are only two levels of the outcome, the probability of the second is simply one minus the probability of the first: $0.76 = 1 - 0.24$.

Table 65: Prediction for a 50-year old smoker.

outcome	probability
Dead	24%
Alive	76%

In practice, most writers would give the probability of survival (76%) and leave it for the reader to infer the corresponding probability of mortality (24%).

⚠ The model value from a logistic model *is* the prediction.

When the response variable is a two-level categorical variable, which can be converted without loss to a zero-one variable, our preferred regression technique is called “**logistic regression**.” This will be discussed in Lesson 21 but you have already seen logistic regression graphically:

the S-shaped curve running from zero to one.

The model value from logistic regression for any given set of inputs is a number in the range zero to one. Since the model value is a number, you might anticipate the need for a prediction interval around this number, just as for non-zero-one numerical variables. However, such an interval is not needed. The model value from logistic regression is itself in the proper form for a prediction. The model output is the probability assigned to the level of the response variable represented by the number 1. Since there are only two levels for a zero-one variable, the probability assigned to level 0 will be the complement of the probability assigned to level 1.

i Example: Differential diagnosis

A patient comes to an urgent-care clinic with symptoms. The healthcare professional tries to diagnose what disease or illness the patient has. A diagnosis is a prediction. The inputs to the prediction are the symptoms—neck stiffness, a tremor, and so on—as well as facts about the person, such as age, sex, occupation, and family history. The prediction output is a set of probabilities, one for each medical condition that could cause the symptoms.

Doctors are trained to perform a *differential diagnosis*, where the current set of probabilities informs the choices of additional tests and treatments. The probabilities are updated based on the information gained from the tests and treatments. This update may suggest new tests or treatments, the results of which may drive a new update. The popular television drama *House* provides an example of the evolving predictions of differential diagnosis in every episode.

19 Sampling and sampling variation

A food market will give you a sample of an item on sale: a tiny cup of a drink or a taste of a piece of fruit or other food item. Laundry-detergent companies sometimes send out a sample of their product in the form of a small foil packet suitable for only a single wash cycle. Paint stores keep small samples on hand to help customers choose from among the possibilities. A fabric sample is a little swatch of cloth cut from a bigger bolt that a customer is considering buying. A dictionary definition of sample is, “a small part or quantity intended to show what the whole is like.”

In contrast, in statistics, a **sample** is always a *collection* of multiple items. The individual items are **specimens**, each one recorded in its own row of a data frame. The entries in that row record the measured attributes of that specimen.

The collection of specimens is the sample. In museums, the curators put related specimens—fossils or stone tools—into a drawer or shelf. Statisticians use data frames to hold their samples. A “sample” is akin to words like “a herd,” “a flock,” “a pack,” or “a school,” each of which refers to a collective. A single fish is not a school; a single wolf is not a pack. Similarly, a single row is not a sample but a specimen.

The dictionary definition of “sample” uses the word “whole” to describe where the sample comes from. Similarly, a statistical sample is a collection of specimens selected from a larger “whole.” Traditionally, statisticians have used the word **“population”** as the name for the “whole.” “Population” is a good metaphor; it’s easy to imagine the population of a state being the source of a sample in which each individual is a specific person. But the “whole” from which a sample is collected does not need to be a finite, definite set of individuals like the citizens of a state. For example, you have already seen how to collect a sample of any size you want from a DAG.

An example of a sample is the data frame `Galton`, which records the heights of a few hundred people sampled from the population of London in the 1880s.

From *Oxford Dictionaries*

Our *modus operandi* in these Lessons takes a sample, stores the sample one specimen per row a data frame and summarizes the whole sample in the form of one or more numbers: a “**sample summary**.” Typically, the sample summary is the coefficients of a regression model, but it might be something else such as the mean or variance of a variable.

As a concrete example, consider this sample summary of Galton.

```
Galton |> summarize(var(height), mean(height))
```

var(height)	mean(height)
12.84	66.76

The summary includes two numbers, the variance and the mean of the `height` variable. Each of these numbers is called a “**sample statistic**.” In other words, the summary consists of two sample statistics.

There can be many ways to summarize a sample. Here is another summary of Galton:

```
Galton |>
  model_train(height ~ mother + father + sex) |>
  coef()
```

```
(Intercept) mother father sexM 15.3447600 0.3214951 0.4059780
5.2259513
```

This summary has four numbers, each of which is a regression coefficient. Here too, each of the regression coefficients is a “sample statistic.”

19.1 Why sample?

To understand samples and sampling, it helps to start with a collection that is not a sample. A non-sample data frame

contains a row for every member of the literal, finite “population.” Such a complete enumeration—the inventory records of a merchant, the records kept of student grades by the school registrar—has a technical name: a “**census**.” Famously, many countries conduct a census of the population in which they try to record every resident of the country. For example, the US, UK, and China carry out a census every ten years.

In a typical setting, recording every possible observation unit is unfeasible. Such incomplete records constitute a “**sample**.” One of the great successes of statistics is the means to draw useful information from a sample, at least when the sample is collected with a correct methodology.

Sampling is called for when we want to find out about a large group but lack time, energy, money, or the other resources needed to contact every group member. For instance, unlike the 10-year *census*, France collects *samples* from its population at short intervals to collect up-to-date data while staying within a budget. The name used for the process—the *recensement en continu* (“rolling census”)—signals the intent. Over several years, the *recensement en continu* contacts about 70% of the population. As such, it is not a “census” in the narrow statistical sense.

Another example of the need to sample comes from quality control in manufacturing. The quality-control measurement process is often destructive: the measurement process consumes the item. In a destructive measurement situation, it would be pointless to measure every single item. Instead, a sample will have to do.

Even a population “census” inevitably leaves out some individuals.

19.2 Sampling bias

Collecting a reliable sample is usually considerable work. An ideal is the “simple random sample” (SRS), where all of the items are available, but only some are selected—completely at random—for recording as data. Undertaking an SRS requires assembling a “sampling frame,” essentially a census. Then, with the sampling frame in hand, a computer or throws of the dice can accomplish the random selection for the sample.

Understandably, if a census is unfeasible, constructing a perfect sampling frame is hardly less so. In practice, the sample is assembled by randomly dialing phone numbers or taking every 10th visitor to a clinic or similar means. Unlike genuinely random samples, the samples created by these practical methods do not necessarily represent the larger group accurately. For instance, many people will not answer a phone call from a stranger; such people are underrepresented in the sample. Similarly, the people who can get to the clinic may be healthier than those who cannot. Such unrepresentativeness is called “**sampling bias**.”

Professional work, such as collecting unemployment data, often requires government-level resources. Assembling representative samples uses specialized statistical techniques such as stratification and weighting of the results. We will not cover the specialized methods in this introductory course, even though they are essential in creating representative samples. The table of contents of a classic text, William Cochran’s *Sampling techniques* shows what is involved.

All statistical thinkers, whether experts in sampling techniques or not, should be aware of factors that can bias a sample away from being representative. In political polls, many (most?) people will not respond to the questions. If this non-response stems from, for example, an expectation that the response will be unpopular, then the poll sample will not adequately reflect unpopular opinions. This kind of **non-response bias** can be significant, even overwhelming, in surveys.

Survival bias plays a role in many settings. The `mosaicData::TenMileRace` data frame provides an example, recording the running times of 8636 participants in a 10-mile road race and including information about each runner’s age. Can such data carry information about changes in running performance as people age? The data frame includes runners aged 10 to 87. Nevertheless, a model of running time as a function of age from this data frame is seriously biased. The reason? As people age, casual runners tend to drop out of such races. So the older runners are skewed toward higher performance.

i Examples: Returned to base

An inspiring story about dealing with survival bias comes from a World War II study of the damage sustained by bombers due to enemy guns. The sample, by necessity, included only those bombers that survived the mission and returned to base. The holes in those surviving bombers tell a story of survival bias. Shell holes on the surviving planes were clustered in certain areas, as depicted in Figure 63. The clustering stems from survivor bias. The unfortunate planes hit in the middle of the wings, cockpit, engines, and the back of the fuselage did not return to base. Shell hits in those areas never made it into the record.

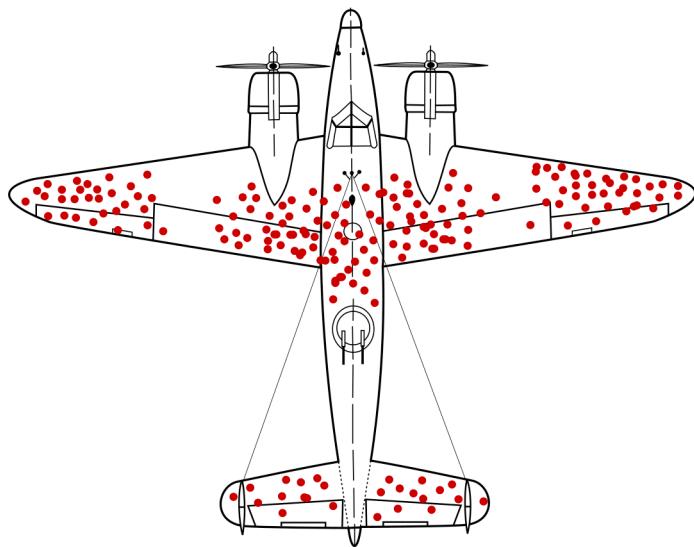


Figure 63: An illustration of shell-hole locations in planes that returned to base. [Source: Wikipedia](#)

i Sampling bias and the “30-million word gap”

For the last 20 years, conventional wisdom has held that lower socio-economic status families talk to their children less than higher status families. The quoted number is a gap of 30 million words per year between the low-status and high-status families.

The 30-million word gap is due to ... mainly, sampling bias.

This [story from National Public Radio](#) explains some of the sources of bias in counting words spoken. More comes from the original data being collected by spending an hour with families in the early evening. That's the time, later research has found, that families converse the most. More systematic sampling, using what are effectively “word pedometers,” puts the gap at 4 million words per year.

19.3 Sampling variation

Usually we work with a single sample, the data frame at hand. As always, the data consists of signal combined with noise. To see the consequences of sampling on summary statistics such as model coefficients, consider a “thought experiment.” Imagine having multiple samples, each collected independently and at random from the same source and stored in its own data frame. Continuing the thought experiment, calculate sample statistics in the same way for each data frame, say, a particular regression coefficient. In the end, we will have a collection of equivalent sample statistics. We say “equivalent” because each individual sample statistic was computed in the same way. But the sample statistics, although equivalent, will differ one from another to some extent because they come from different samples. Sample by sample, the sample statistics *vary* one to the other. We call such variation among the summaries “**sampling variation**.”

The proposed thought experiment can be carried out. We just need a way to collect many samples from the same data source. To that end, we use a data simulation as the source. The simulation provides an inexhaustible supply of potential samples. Then, we will calculate a sample statistic for each sample. This will enable us to see sampling variation directly.

Our standard way of measuring the amount of variation is with the *variance*. Here, we will measure the variance of a sample statistic from a large set of samples. To remind us that the variance we calculate is to measure sampling variation, we will give it a distinct name: the “**sampling variance**.”

The **ing** in sampling

Pay careful attention to the “ing” ending in “sampling variation” and “sampling variance.” The phrase “sample statistic” does not have an “ing” ending. When we use the “ing” in “sampling,” it is to emphasize that we are looking at the variation in a sample statistic from one sample to another.

The simulation technique will enable us to witness essential properties of the sampling variance, particularly how it depends on sample size n .

19.4 Sampling trials

We will use `sim_02` as the data source, but the same results would be found with any other simulation.

```
print(sim_02)
```

```
Simulation object
-----
[1] x <- rnorm(n)
[2] a <- rnorm(n)
[3] y <- 3 * x - 1.5 * a + 5 + rnorm(n)
```

You can see from the mechanisms of `sim_02` that the model $y \sim x + a$ will, ideally, produce an intercept of 5, an x -coefficient of 3, and an a -coefficient of -1.5. We can verify this using a very large sample size:

```
sim_02 |> sample(n=100000) |>
  model_train(y ~ x + a) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	5
x	3
a	-1.5

The coefficients in the large sample are very close to what's expected. But if the sample size is small, the coefficients appear further off target.

```
sim_02 |> sample(n=25) |>
  model_train(y ~ x + a) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	5.11
x	2.92
a	-1.11

It's reasonable to wonder whether the deviations of the coefficients from the sample of size $n = 25$ result from a flaw in the modeling process or simply from sampling variation.

We cannot see sampling variation directly in the above result because there is only one trial. The sampling variation becomes evident when we run *many* trials. To accomplish this, run the above R code many times. That is, “run many trials.” Record the coefficient from each trial, storing it in one row of a data frame.

To avoid such tedium, the `{LSTbook}` R package includes a `'trials()'` function that automates the process, producing a data frame as output. Here, we run 500 trials. (Only the first three are displayed.)

```
Trials <-
  sim_02 |> sample(n = 25) |>
  model_train(y ~ x + a) |>
```

```
conf_interval() |>
  select(term, .coef) |>
  trials(500)
```

.trial	term	.coef
1	(Intercept)	5
1	x	3.2
1	a	-1.5
2	(Intercept)	5.2
2	x	2.9
2	a	-1.4
3	(Intercept)	5
3	x	3
3	a	-1.6

...:

Graphics provide a nice way to visualize the sampling variation. Figure 64 shows the results from the set of trials. The distributions are centered on the coefficient values used in the simulation itself.

```
Trials |>
  point_plot(.coef ~ term, annot="violin", point_ink = 0.2, size = 1)
```

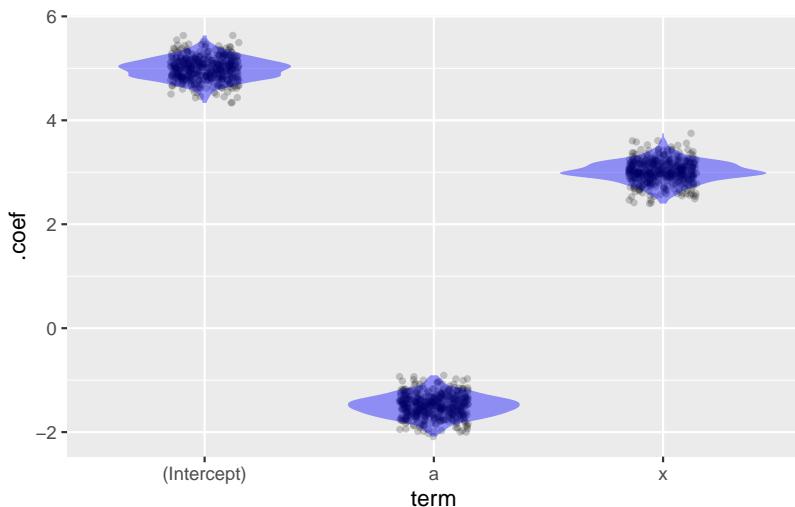


Figure 64: The sampling distribution as shown by 500 trials. Each dot is one trial where the model specification $y \sim x + a$ is fitted to a sample from `sim_02` of size $n = 25$.

Use `var()` to calculate the sampling variance for each of the two coefficients.

```
Trials |>
  summarize(sampling_variance = var(.coef),
            standard_error = sqrt(sampling_variance), .by = term)
```

term	sampling_variance	standard_error
(Intercept)	0.04623	0.215
x	0.04532	0.2129
a	0.04563	0.2136

Often, statisticians prefer to report the *square root of the sampling variance*. This has a technical name in statistics: the **standard error**. The “standard error” is an ordinary standard deviation in a particular context: the standard deviation of a sample of summaries. The words **standard error** should be followed by a description of the summary and the size of the individual samples involved. Here, the correct statement is, “The standard error of the Intercept coefficient from a sample of size $n = 25$ is around 0.2.”

⚠️ Confusion about “standard” and “error”

It is easy to confuse “standard error” with “standard deviation.” Adding to the potential confusion is another related term, the “margin of error.” We can avoid this confusion by using an interval description of the sampling variation. You have already seen this: the **confidence interval** (as computed by `conf_interval()`). The confidence interval is designed to cover the central 95% of the sampling distribution. (See Lesson -Chapter 20.)

19.5 SE depends on the sample size

The 500 trials of samples of size $n=25$ from `sim_02` revealed a *sampling variance* of about 0.045 for each of the three coefficients. (In general, different coefficients can have different

standard errors.) If we were to run 100 trials or 100,000 trials, we would get about the same result: 0.045. We ran 500 trials to get a reliable result without too much time computing.

In contrast, the sampling variance changes systematically with the sample size. We can see how the standard error depends on sample size by repeating the trials for several sizes, say, $n = 25$, 100, 400, 1600, 6400, 25,000, and 100,000.

The following command estimates the SE a sample of size 400:

```
Trials <-
  sample(sim_02, n = 400) |>
  model_train(y ~ x + a) |>
  conf_interval() |>
  trials(500)
Trials |>
  summarize(sampling_variance = var(.coef),
            standard_error = sd(.coef),
            .by = term)
```

term	sampling_variance	standard_error
(Intercept)	0.002732	0.05227
x	0.002733	0.05228
a	0.002732	0.05227

Whereas for a sample size $n = 25$, the standard error of the coefficients was about 0.2, for a sample size of $n = 400$, sixteen times bigger, the standard deviation is four times smaller: about 0.05.

We repeated this process for each of the other sample sizes. Table 72 reports the results.

Table 72: Results of repeating the sampling variability trials for samples of varying sizes.

term	n	sampling_variance	standard_error
(Intercept)	25	0.0437	0.209
(Intercept)	100	0.00985	0.0992
(Intercept)	400	0.00267	0.0517
(Intercept)	1600	0.000658	0.0257
(Intercept)	6400	0.000161	0.0127
(Intercept)	25000	4.41e-05	0.00664
(Intercept)	1e+05	9.63e-06	0.0031
a	25	0.0433	0.208
a	100	0.00965	0.0982
a	400	0.00237	0.0487
a	1600	0.000585	0.0242
a	6400	0.000154	0.0124
a	25000	3.9e-05	0.00625
a	1e+05	8.7e-06	0.00295
x	25	0.0482	0.22
x	100	0.011	0.105
x	400	0.00269	0.0519
x	1600	0.000649	0.0255
x	6400	0.00014	0.0118
x	25000	4.16e-05	0.00645
x	1e+05	1.03e-05	0.00321

There is a pattern in Table 72. Every time we quadruple n , the sampling variance decreases by a factor of four. Consequently, the standard error—which is just the square root of the sampling variance—goes down by a factor of 2, that is, $\sqrt{4}$. (The pattern is not exact because there is also sampling variation in the trials, which are themselves a sample of all possible trials.)

Conclusion: The larger the sample size, the smaller the sampling variance. For a sample of size n , the sampling variance will be proportional to $1/n$. Or, in terms of the standard error: For a sample size of n , the standard error will be proportional to $1/\sqrt{n}$.

20 Confidence intervals

Lesson 19 took a simulation approach to observing sampling variation: generate many trials from a source such as a DAG and observe how the same sample statistic varies from trial to trial. We quantified the sampling variation in the same way we usually quantify variation, taking the **variance** of the sample statistic across all the trials. We called this measure of variation the **sampling variance** as a reminder that it comes from repeated trials of sampling.

In this Lesson, we will examine a more informative format for reporting sampling variation: the **confidence interval**. We will also consider an important general concept for interpreting confidence intervals: **precision** of measurement. We will contrast *precision* with **accuracy** to help you avoid the common error of mistaking precision with accuracy.

Taking into consideration that precision is a general issue in any kind of quantitative reporting, not just statistical modeling, it might have been better if “precision interval” had been used instead of “confidence interval.” The word “confidence” in “confidence interval” has *nothing to do* with self-assuredness, boldness, or confidentiality. (When “confidence interval” was introduced in the 1930s, the word was chosen to avoid a once-bitter technical dispute in the philosophy of probability.)

Summary: The confidence interval is a measure of precision: the reproducibility from sample to sample. It tells us nothing about accuracy. Without understanding the difference between “precision” and “accuracy,” it is difficult to interpret confidence intervals appropriately.

20.1 Formats for confidence intervals

We have been looking at confidence intervals since Lesson 11, where we introduced the `conf_interval()` function for displaying model coefficients. To illustrate, consider the running time (in seconds, s) for Scottish hill races as a function of the race distance (in km) and overall height climbed (in meters, m):

```
Hill_racing |>
  model_train(time ~ distance + climb) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-533	-470	-407
distance	246	254	261
climb	2.49	2.61	2.73

As always, there is a model coefficient for each term mentioned in the model specification, `time ~ distance + climb`. Here, those terms give an intercept, a coefficient on distance, and a coefficient on climb. Each coefficient comes with two other numbers, called `.lwr` and `.upr` in the report, standing for “lower” and “upper.” The confidence interval runs from the lower number to the upper number.

Focus for the moment on the `distance` coefficient: 253.8 s/km. The confidence interval runs from 246 to 261 s/km. In previous Lessons about model values—the output of the model function when given values for the explanatory variables—we have emphasized the coefficient itself..

Statistical thinkers, knowing that there is sampling variation in any coefficient calculated from a data sample, like to use the word “**estimate**” to refer to the calculated value. Admittedly, the computer carries out the calculation of the coefficient without mistake and reports it with many digits. But those digits do not incorporate the uncertainty due to sampling variation. That’s the role of the confidence interval.

The meaning of a confidence interval such as the 246-to-261 s/km interval shown above is, “Any other estimate of the coefficient (made with other data) is consistent with ours so long as it falls within the confidence interval.”

An alternative, but entirely equivalent format for the confidence interval uses \pm (plus-or-minus) notation. The interval [246-261] s/km in \pm format can be written 254 ± 8 s/km.

i Significant digits?

Another convention for reporting uncertainty—legendarily emphasized by chemistry teachers—involves the number of digits with which to write a number: the “**significant digits**.” For instance, the **distance** coefficient reported by the computer is 253.808295 s/km. Were you to put this number in a lab report, you are at risk for a red annotation from your teacher: “Too many digits!”

According to the significant-digits convention, a proper way to write the **distance** coefficient would be 250 s/km, although some teachers might prefer 254 s/km.

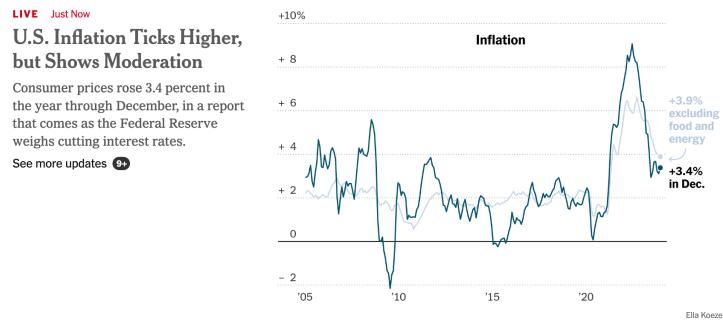
The situation is difficult because the significant-digit convention is attempting to serve three different goals at once. The first goal is to signal the precision of the number. The second goal is to avoid overwhelming human readers with irrelevant digits. The third goal is to allow human readers to redo calculations. These three goals sometimes compete. An example is the [246,261] s/km confidence interval on the **distance** coefficient reported earlier. For this coefficient, the width of the confidence interval is about 15 s/km. This suggests that there is no value to the human reader in reporting any digits after the decimal point. But a literal translation of [246-261] into \pm format would be 253.5 ± 7.5 . Now there is a digit being reported after the decimal point, a digit we previously said isn’t worth reporting!

As a general-purpose procedure, I suggest the following principles for model coefficients:

1. **Always** report an interval in either the [lower, upper] format or the center \pm spread format. It doesn’t much matter which one.
2. As a guide to the number of digits to print, look to the interval width, calculated as upper – lower or as $2 \times$ spread. Print the number using the interval width as a guide: only the first two digits (neglecting leading zeros) are worth anything.
3. When interpreting intervals, don’t put much stock

in the last digit. For example, is 245 km/s inside the interval [246, 261] km/s. Not mathematically. But remembering that the last digit in 246 is not to be taken as absolute, 245 is for all practical purposes inside the interval.

As I write (2024-01-11), a news notice appeared on my computer screen from the *New York Times*.



The “Inflation Ticks Higher” in the headline is referring to a change from 3.3% reported in November to 3.4% reported in December. Such reports ought to come with a precision interval. To judge from the small wiggles in the 20-year data, this would be about $\pm 0.2\%$. A numerical change from 3.3% to 3.4% is, taking the precision into account, no change at all!

20.2 Precision versus accuracy

In everyday language the words “precision” and “accuracy” are interchangeable; both describe how well a measurement has been made. Nevertheless there are two distinct concepts in “how well.” The easier concept has to do with *reproducibility* and *reliability*: if the measurement is taken many times, how much will the measurements differ from one another? This is the same issue as *sampling variation*. In the technical lingo of measurement, reproducibility or sampling variation is called **“precision”**. Precision is just about the measurements themselves.

In contrast, in speaking technically we use “**accuracy**” to re-

fer to a different concept than “precision.” Accuracy cannot be computed with just the measurements. Accuracy refers to something outside the measurements, what we might call the “true” value of what we are trying to measure. Disappointingly, the “true” value is an elusive quantity since all we typically have is our measurements. We can easily measure precision from data, but our data have practically nothing to say about accuracy.

An analogy is often made between precision and accuracy and the patterns seen in archery. Figure 65 shows five arrows shot during archery practice. The arrows are in an area about the size of a dinner plate 6 inches in radius: that’s the precision.



Figure 65: Results from archery practice

A dinner-plate’s precision is not bad for a beginner archer. Unfortunately, the dinner plate is not centered on the bullseye but about 10 inches higher. In other words, the arrows are inaccurate by about 10 inches.

Since the “true” target is visible, it is easy to know the accuracy of the shooting. The analogy of archery to the situation in statistics would be better if the target was shown in plane white, that is, if the “true” value were not known directly. In that situation, as with data analysis, the spread in the arrows’ locations could tell us only about the precision.

To illustrate the difference between precision and accuracy, let’s look again at the coefficient on `distance` in the Scottish hill racing model. Our original model was

```
Hill_racing |>
  model_train(time ~ distance + climb) |>
  conf_interval() |>
  filter(term == "distance")
```

term	.lwr	.coef	.upr
distance	246	254	261

Another possible model uses only `distance` as an explanatory variable:

```
Hill_racing |>
  model_train(time ~ distance) |>
  conf_interval() |>
  filter(term == "distance")
```

term	.lwr	.coef	.upr
distance	374	381	388

The second confidence interval, [374, 388] s/km, is utterly inconsistent with the earlier confidence interval [246, 261]. This is a matter of **accuracy**. The `distance` coefficient in the first model is aimed at a different target than the `distance` coefficient in the second model. In exploring hill-racing data, should we look at distance taking into account `climb` (the first model) or ignoring `climb` (the second model). The width of the confidence interval addresses only the issue of precision, not whether the model is accurate for the purpose at hand.

20.3 The confidence level

The confidence interval is designed to communicate to a human reader the influence of sampling variation as it plays out in the calculation of a model coefficient (or some other sample statistic such as the median or R^2). The two equivalent formats we use for the interval—for example, [374, 388] or equivalently 381 ± 7 —are intended to be easy to read and use for the intended purpose.

A more complete picture of sampling variation is provided by treating it as a noise model, as described in Lesson 15. We can

choose an appropriate noise model by looking at the distribution shape for sampling variation. Experience has shown that an excellent, general-purpose noise model for sampling variation is the *normal* noise model. To support this claim we can use a simulation of the sort reported in Figure 64, where the distribution of coefficients across the 500 sampling trials has the characteristic shape of the normal model.

To show how that normal noise model relates to confidence intervals, we can calculate a confidence interval from data and compare that interval to a simulation of sampling variation. We will stick with the `distance` coefficient in the model `time ~ distance + climb` trained on the Scottish hill racing data in the `Hill_racing` data frame. But any model of any other data set would show much the same thing.

Recall that the confidence interval on `distance` is 246 s/km to 261 s/km. We can construct individual trials of sampling variation through a technique called “**resampling**” that will be described in Chapter 19. In essence, the resampling technique takes a sample of the same size from a data frame. In the simulation, we will use resampling to generate a “new” sample, train a model on that new sample, then report the `distance` coefficient and its confidence interval. Each trial will look like this:

```
resample(Hill_racing) |>
  model_train(time ~ distance + climb) |>
  conf_interval() |>
  filter(term == "distance")
```

term	.lwr	.coef	.upr
distance	248.1	255.6	263

Let’s run 10,000 such trials and store them in a data frame we will call `Trials`:

```
Trials <-
  resample(Hill_racing) |>
  model_train(time ~ distance + climb) |>
```

```

conf_interval() |>
filter(term == "distance") |>
trials(10000)

```

Now, let's plot the 10,000 coefficients, one from each trial:

```

Trials |>
point_plot(.coef ~ 1, annot = "violin", point_ink = 0.1, size = 0.5) |>
gf_errorbar(246 + 261 ~ 1, color = "red") |>
add_plot_labels(y = "Coefficient on distance (s/km)")

```

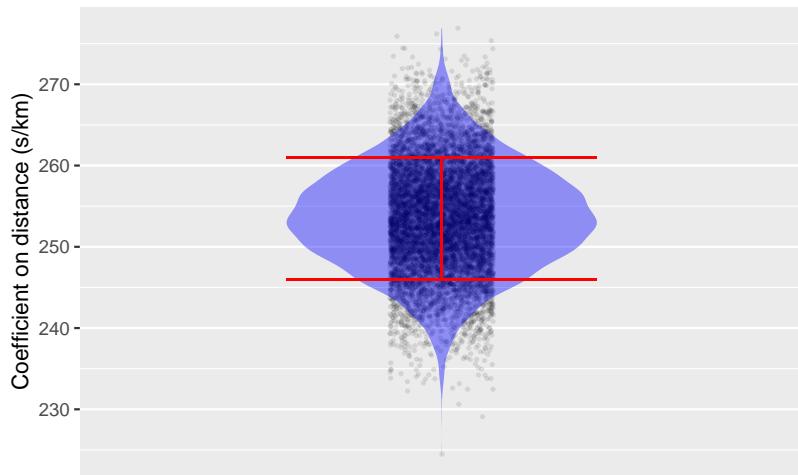


Figure 66: Five-hundred trials in which the `distance` coefficient in the model `time ~ distance + climb`. The $[246, 261]$ confidence interval from the actual data is drawn in red.

Some things to note from Figure 66:

1. The distribution of the `distance` coefficient from the resampling trials has the shape of the normal noise model.
2. The large majority of the trials produced a coefficient that falls inside the confidence interval found from the original data.
3. Some of the trials fall outside that confidence interval. Sometimes, if rarely, the trial falls far outside the confidence interval.

A complete description of the possible range in the `distance` coefficient due to sampling variation would be something like Figure 66. For pragmatic purposes, however, rather than report 10,000 (or more!) coefficients we report just two values: the bounds of the confidence interval.

By convention, the bounds of the confidence interval are selected to contain 95% of the coefficients. Thus, the confidence interval should more properly be called the “**95% confidence interval**” or “the confidence interval at a 95% level.” The confidence interval gives us a solid feel for the amount of sampling variation, but it can never encompass all of it.

To calculate a confidence interval at a level other than 95%, use the `level=` argument to `conf_interval()`. For instance, for an 80% level, use `conf_interval(level = 0.85)`.

20.4 Calculating confidence intervals (optional)

In Lesson 19, we repeated trials over and over again to gain some feeling for sampling variation. We quantified the repeatability in any of several closely related ways: the sampling variance or its square root (the “standard error”) or a “margin of error” or a “confidence interval.” Our experiments with simulations demonstrated an important property of sampling variation: the amount of sampling variation depends on the sample size n . In particular, the sampling variance gets smaller as n increases in proportion to $1/n$. (Consequently, the standard error gets smaller in proportion to $1/\sqrt{n}$.)

It is time to take off the DAG simulation training wheels and measure sampling variation from a *single* data frame. Our first approach will be to turn the single sample into several smaller samples: subsampling. Later, we will turn to another technique, resampling, which draws a sample of full size from the data frame. Sometimes, in particular with regression models, it is possible to calculate the sampling variation from a formula, allowing software to carry out and report the calculations automatically.

The next sections show two approaches to calculating a confidence interval. For the most part, this is background informa-

tion to show you how it’s possible to measure sampling variation from a single sample. Usually you will use `conf_interval()` or similar software for the calculation.

20.4.1 Subsampling

Although computing a confidence interval is a simple matter in software, it is helpful to have a conceptual idea of what is behind the computation. This section and Section 20.4.2 describe two methods for calculating a confidence interval from a single sample. The `conf_interval()` summary function uses yet another method that is more mathematically intricate, but which we won’t describe here.

To “subsample” means to draw a smaller sample from a large one. “Small” and “large” are relative. For our example, we turn to the `TenMileRace` data frame containing the record of thousands of runners’ times in a race, along with basic information about each runner. There are many ways we could summarize `TenMileRace`. Any summary would do for the example. We will summarize the relationship between the runners’ ages and their start-to-finish times (variable `net`), that is, `net ~ age`. To avoid the complexity of a runner’s improvement with age followed by a decline, we will limit the study to people over 40.

```
TenMileRace |>  
  filter(age > 40) |>  
  model_train(net ~ age) |>  
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	4015	4278	4542
age	22.83	28.14	33.44

The units of `net` are seconds, and the units of `age` are years. The model coefficient on `age` tells us how the `net` time changes for each additional year of `age`: seconds per year. Using the entire data frame, we see that the time to run the race gets

longer by about 28 seconds per year. So a 45-year-old runner who completed this year's 10-mile race in 3900 seconds (about 9.2 mph, a pretty good pace!) might expect that, in ten years, when she is 55 years old, her time will be longer by 280 seconds.

It would be asinine to report the ten-year change as 281.3517 seconds. The runner's time ten years from now will be influenced by the weather, crowding, the course conditions, whether she finds a good pace runner, the training regime, improvements in shoe technology, injuries, and illnesses, among other factors. There is little or nothing we can say from the `TenMileRace` data about such factors.

There's also sampling variation. There are 2898 people older than 40 in the `TenMileRace` data frame. The way the data was collected (radio-frequency interrogation of a dongle on the runner's shoe) suggests that the data is a census of finishers. However, it is also fair to treat it as a sample of the kind of people who run such races. People might have been interested in running but had a schedule conflict, lived too far away, or missed their train to the start line in the city.

We see sampling variation by comparing multiple samples. To create those multiple samples from `TenMileRace`, we will draw, at random, subsamples of, say, one-tenth the size of the whole, that is, $n = 290$

```
Over40 <- TenMileRace |> filter(age > 40)
# Run a trial
Over40 |> sample(n = 290) |>
  model_train(time ~ age) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	3232	4171	5110
age	15.48	34.14	52.8

```
# Run another trial
Over40 |> sample(n = 290) |>
```

```
model_train(time ~ age) |>
conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	3697	4509	5322
age	9.835	26.14	42.45

The age coefficients from these two subsampling trials differ one from the other by about 0.5 seconds. To get a more systematic view, run more trials:

```
# a sample of summaries
Trials <-
Over40 |> sample(290) |>
model_train(time ~ age) |>
conf_interval() |>
trials(1000)
```

There is a distribution of coefficients from the various trials. We can quantify the amount of variation with the variance of the coefficients. Here, we will use the standard deviation, which is (as always) simply the square root of the variance.

```
Trials |>
dplyr::summarize(sd(.coef), .by = term)
```

term	sd(.coef)
(Intercept)	445.2
age	9.069

The standard deviation of the variation induced by sampling variability is called the “**standard error**” (SE) of the coefficient. Calculating the standard error is one of the steps in traditional methods for finding confidence intervals. The SE is very closely related to the width of the confidence interval. For instance, here is the mean width of the CI calculated from the 1000 trials:

```

Trials |>
  mutate(width = .upr - .lwr) |>
  summarize(mean(width), sd(width), .by = term)

```

term	mean(width)	sd(width)
(Intercept)	1803	108.8
age	36.32	2.316

The SE is typically about one-quarter the width of the 95% confidence interval. For our example, the SE is 9 while the width of the CI is 36. The approximate formula for the CI is

$$CI = \text{coefficient} \pm \text{SE}.$$

As described in Lesson 19, both the width of the CI and the SE are proportional to $1/\sqrt{n}$, where n is the sample size. From the subsamples, know that the SE for $n = 290$ is about 9.0 seconds. This tells us that the SE for the full $n = 2898$ samples would be about $9.0\sqrt{\frac{290}{2898}} = 2.85$.

So the interval summary of the `age` coefficient—the *confidence interval*—is

$$\begin{array}{ccc} 28.1 & \pm 2 \times 2.85 & = 28.1 \pm 5.6 \\ \text{age coef.} & \text{standard error} & \text{margin of error} \end{array} \quad \text{or, equivalently, } 22.6 \text{ to } 33.6$$

20.4.2 Bootstrapping

There is a trick, called “**resampling**,” to generate a random subsample of a data frame with the same n as the data frame: draw the new sample randomly from the original sample **with replacement**. An example will suffice to show what the “with replacement” does:

```

example <- c(1,2,3,4,5)
# without replacement
sample(example)

```

```
[1] 1 4 3 5 2
```

```
# now, with replacement  
sample(example, replace=TRUE)
```

```
[1] 2 4 3 3 5
```

```
sample(example, replace=TRUE)
```

```
[1] 3 5 4 4 4
```

```
sample(example, replace=TRUE)
```

```
[1] 1 1 2 2 3
```

```
sample(example, replace=TRUE)
```

```
[1] 4 3 1 4 5
```

The “with replacement” leads to the possibility that some values will be repeated two or more times and other values will be left out entirely.

The calculation of the SE using resampling is called “**bootstrapping**.”

⚠ Demonstration: Bootstrapping the standard error

We will apply bootstrapping to find the standard error of the `age` coefficient from the model `time ~ age` fit to the `Over40` data frame.

There are two steps:

1. Run many trials, each of which fits the model `time ~ age` using `model_train()`. From trial to trial, the data used for fitting is a resampling of the `Over40` data frame. The result of each trial is the coefficients

from the model.

2. Summarize the trials with the standard deviation of the `age` coefficients.

```
# run many trials
Trials <-
  Over40 |> sample(replace=TRUE) |>
  model_train(time ~ age) |>
  conf_interval() |>
  trials(500)

# summarize the trials to find the SE
Trials |>
  summarize(se = sd(.coef), .by = term)
```

term	se
(Intercept)	140.4
age	2.815

20.5 Decision-making with confidence intervals

Consider the situation of testing a new antibiotic “B” intended as a substitute for an antibiotic “A” that is already in use. The clinical trial involves 200 patients each of whom will be randomly assigned to take “A” or “B” as their treatment. The outcome for each patient will be the time from the beginning of treatment to the disappearance of symptoms. The data collected look like this:

patient	age	sex	severity	treatment	duration
ID7832	52	F	4	B	5
ID4981	35	F	2	A	3
ID2019	43	M	3	A	2

... and so on for 200 rows altogether.

Why random? See Lesson [21](#).

The outcome of the study is intended to support one of three clinical decisions:

- Continue preferring treatment A
- Switch to treatment B
- Dither, for instance, recommending that a larger study be done.

In the analysis stage of the study, you start with a simple model:

[In Lessons 25 through 25 we will see how to take `age`, `sex`, and `severity` into account as well.]

```
antibiotic_sim |> datasim_run(n=200) |>
model_train(duration ~ treatment) |>
conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	2.9	3.3	3.6
treatmentB	-0.88	-0.36	0.15

Figure 67 shows (in red) the confidence interval on `treatmentB`. The left end of the interval is in the region which would point to using treatment B, but the right end is in the treatment A region. Thus, the confidence interval for $n = 200$ creates an ambiguity about which treatment is to be preferred.

Which of the three decisions—continue with antibiotic A, switch to B, or dither—would be supported if only the $n = 200$ study results were available? Noting that the vast majority of the $n = 200$ confidence interval is in the “use B” region, common sense suggests that the decision should be to switch to B, perhaps with a caution that this might turn out to be a mistake. A statistical technique called “Bayesian estimation” ([[touched on in]] Lesson 28) can translate the data into a subjective probability that B is better than A, quantifying the “caution” in the previous sentence. Traditional statistical reasoning, however, would point to dithering.

With the larger $n = 400$ study, the confidence interval (blue) is narrower. The two studies are consistent with one another

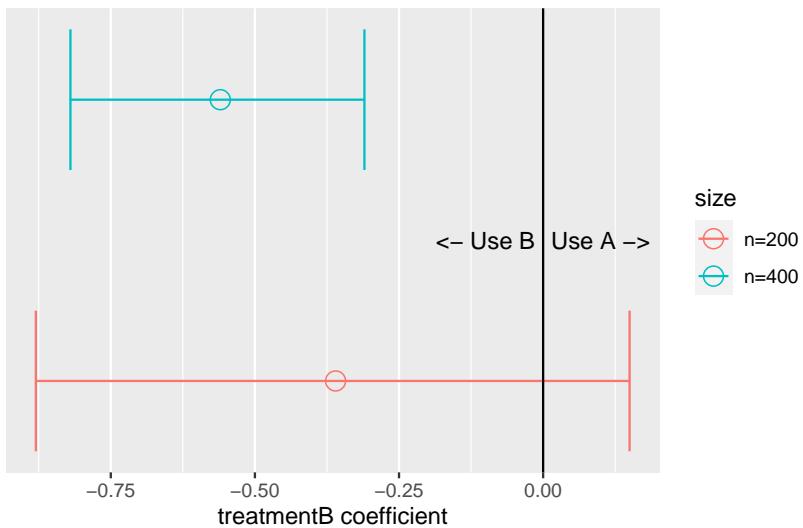


Figure 67: Confidence intervals from two differently sized studies.

in terms of the `treatmentB` coefficient, but the larger study results place both ends of the confidence interval in the “use B” region, removing the ambiguity.

Statistical analysis should support decision-making, but often there are other factors that come into play. For instance, switching to antibiotic B might be expensive so that the possible benefit isn’t worth the cost. Or, the option to carry out a larger study may not be feasible. Decision-makers need to act with the information that is in hand and the available options. It’s a happy situation when both ends of the confidence interval land in the same decision region, reducing the ambiguity and uncertainty that is a ever-present element of decision-making.

21 Measuring and accumulating risk

In everyday language, “risk” refers to a dangerous or unwelcome outcome. We talk about the “risk of heart disease,” “risk of bankruptcy,” or another unwelcome outcome. To apply risk to a positive outcome is non-idiomatic. For instance, for a person wanting to have a baby, we don’t talk about the “risk of pregnancy” but about the “chances of becoming pregnant.”

The statistics of risk and “chances of” are equivalent. It matters to the decision-maker whether the outcome referred to is positive or negative, but it makes no difference to the mathematics.

Usually, the outcomes described by “risk” or “chances of” are categorical. In these Lessons, we’ll consider only two-level categorical variables. Generically, the levels of the categorical variable might be “unwelcome” and “not unwelcome,” but they might be more specifically named, say, “death” and “survival,” or “lung disease” and “not.” Risk or “chances of” involve two categories, typically one unfavored and the other favored.

We have been building models of such categorical output variables from the start of these Lessons. For the zero-one categorical variables we have emphasized, the model output is in the form of a probability: the probability of the outcome of the event being “one” (or whatever actual level “one” corresponds to.) If we assign one for “death” and zero for “survival,” the probability which is the output of a model is a risk, but other than the choice of zero-one assignment, there is no mathematical difference (in statistics) between a risk and a probability.

Risk often depends on other factors, often called “risk factors.” In our modeling framework, such risk factors are merely explanatory variables. For instance, a study of the impact of smoking on health might use `outcome` represented by a categorical response variable with levels “death” or “survival.”

21.1 Risk vocabulary

In statistical terms, a **risk** is a probability associated with an outcome.

- A full description of risk looks much like a prediction: a complete list of possible outcomes, each associated with a probability, which we'll call a **risk level**.
- A *risk level* is properly measured as a pure number, e.g. 30 **percent**.
 - Being a probability, such numbers must always be between 0 and 1, or, equivalently, between 0 and 100 percent.
 - There are two ways of referring to percentages, e.g. 30 percent vs 30 percentage points. When talking about a single risk, these two are equivalent. However, “**percentage points**” should be reserved for a particular situation: Describing a change in absolute risk.
- *For simplicity*, we will focus on situations where there are only two outcomes, e.g. alive/dead, success/failure, cancer/not, diabetes/not.
 - Since there are only two outcomes, knowing the probability p of one outcome automatically sets the probability of the other outcome.
 - One of the outcomes is worse than the other, so we usually take the risk to be the worse outcome and its probability.
 - A **risk factor** is a condition, behavior, or such that changes the probability of the (worse) outcome. Just to have concise names, we will use this terminology:
 - i. baseline risk (level): the risk (level) *without* the risk factor applying.
 - ii. augmented risk (level): the risk (level) when the risk factor applies.
- A *risk ratio* is exactly what the name implies: the ratio of the augmented risk to the baseline risk.
 - For instance, suppose the baseline risk is 30% and the augmented risk is 45%. The risk ratio is $45/30 = 1.5 = 150$ percent. Risk ratios are often greater

than 1, which should remind us that a risk ratio is a different kind of beast from a risk, which can never be larger than 1.

- There are two distinct uses for risk factors:
 - i. Draw attention to a factor under our control (e.g. skiing, biking, using a motorcycle, smoking) so that we can decide whether the augmentation in risk is worth avoiding.
 - ii. Establish the *baseline* risk in a relevant way (e.g. our age, sex, and so on).
- For decision-making regarding a risk factor, it is most meaningful to focus on the **change in absolute risk**, that is, the difference between the augmented risk and the baseline risk.
 - Example: The risk ratio for the smoking risk factor is about 2.5/1 for ten-year, all-cause mortality. If the baseline risk is 3 percentage points, the augmented risk is 7.5%. Consequently, the augmentation in risk for smoking is $(2.5-1) \times 3\% = 4.5$ percentage points. On the other hand, if the baseline risk were 30 percentage points, the 2.5 risk ratio increases the risk by 45 percentage points.
 - Notice that we are describing the augmentation in risk as “percentage points.” Always use “percentage points” to avoid ambiguity. If we had said “45 percent,” people might mistake the augmentation in risk as a risk ratio of 1.45.

Question

Why bother to present risk factors in terms of risk ratios when for decision-making it's better to use the augmentation in risk in percentage points?

Answer: Because the same risk factor can lead to different amounts of augmentation depending on the baseline risk. If there are multiple risk factors, then adding up such augmentations can potentially lead to the risk level exceeding 100%.

21.2 Modeling risk

The *linear models* we have been using accumulate the model output as a linear combination of model inputs. Consider, for instance, a simple model of fuel economy based on the horsepower and weight of a car:

```
mpg_mod <- mtcars |> model_train(mpg ~ hp + wt)
mpg_mod |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	33.96	37.23	40.5
hp	-0.05024	-0.03177	-0.01331
wt	-5.172	-3.878	-2.584

The model output is a **sum** of the intercept and each of the other coefficients multiplied by an appropriate value for the corresponding variable. For instance, a 100 horsepower car weighting 2500 pounds has a predicted fuel economy of $37.2 - 0.032 \cdot 100 - 3.88 \cdot 2.5 = 24.3$ miles per gallon. If we're interested in making a prediction, we often hide the arithmetic behind a computer function, but it is the same arithmetic:

```
mpg_mod |> model_eval(hp = 100, wt = 2.5)
```

hp	wt	.lwr	.output	.upr
100	2.5	18.92	24.36	29.79

The arithmetic, in principle, lets us evaluate the model for any inputs, even ridiculous ones like a 10,000 hp car weighing 50,000 lbs. There is no such car, but there is a model output.

```
mpg_mod |> model_eval(hp=10000, wt = 50)
```

The **wt** variable in the training data **mtcars** is measured in units of 1000 lbs, so a 2500 pound vehicle has a **wt** value of 2.5.

A 10,000 hp, 50,000 lbs ground vehicle does have a name: a “tank.” Common sense dictates that one not put too much stake in a calculation of a tank’s fuel economy based on data from cars!

hp	wt	.lwr	.output	.upr
10000	50	-623.7	-474.4	-325.1

The prediction reported here means that such a car goes *negative* 474 miles on a gallon of gas. That's silly. Fuel economy needs to be non-negative; the output -474 mpg is *out of bounds*.

A good way to avoid out-of-bounds behavior is to model a *transformation* of the response variable instead of the variable itself. For example, to avoid negative outputs from a model of `mpg`, change the model so that the output is in terms of the logarithm of `mpg`, like this:

```
logmpg_mod <- mtcars |> model_train(log(mpg) ~ hp + wt)
logmpg_mod |> model_eval(hp = 100, wt = 2.5)
```

hp	wt	.lwr	.output	.upr
100	2.5	2.94	3.173	3.407

The reported output, 3.17, should **not** be interpreted as `mpg`. Instead, interpret it as `log(mpg)`. If we want output in terms of `mpg`, then we have to undo the logarithm. That's the original purpose of the exponential function, which is the *inverse* of the logarithm.

```
logmpg_mod |> model_eval(hp = 100, wt = 2.5) |>
  mutate(mpg = exp(.output), mpg.lwr = exp(.lwr), mpg.upr = exp(.upr))
```

hp	wt	.lwr	.output	.upr	mpg	mpg.lwr	mpg.upr
100	2.5	2.94	3.173	3.407	23.89	18.91	30.17

The logarithmic transform at the model-training stage does not prevent the model output from being negative. We can see this by looking at the tank example:

`exp()` is a mathematical function, often written e^x . We have also encountered a noise model with a similar `exp(.upr)` exponential noise model. `exp()` isn't a noise model; it's more like `cos()` or `tan()`.

```
mod_logmpg <- mtcars |> model_train(log(mpg) ~ hp + wt)
mod_logmpg |> model_eval(hp=10000, wt=50)
```

hp	wt	.lwr	.output	.upr
10000	50	-28.05	-21.63	-15.22

The model output is negative for the tank, but the model output corresponds to `log(mpg)`. What will keep the model from producing negative `mpg` will be the exponential transformation applied to the model output.

```
mod_logmpg |> model_eval(hp=10000, wt=50) |>
  mutate(mpg = exp(.output))
```

hp	wt	.lwr	.output	.upr	mpg
10000	50	-28.05	-21.63	-15.22	4.028e-10

The log transform fixes the out-of-bounds behavior but not the absurdity of modeling tanks based on the fuel economy of cars. The model's prediction of `mpg` for the tank is 0.0000000004 miles/gallon, but real-world tanks do much better than that. For instance, the M1 Abrams tank is reported to get approximately 0.6 miles per gallon.

21.3 Logistic regression

When modeling a *probability* (as opposed to, say, “miles per gallon”) The out-of-bounds problem applies to both sides of the zero-to-one probability scale. Figure 68 shows an example: modeling the probability that a person in the `Whickham` data was still alive at the 20-year follow-up. Notice that the model values go above 1 for a young person and below 0 for an old person.

There is a fix for the out-of-bounds problem when modeling probability. Straight-line models (if the slope is non-zero) must

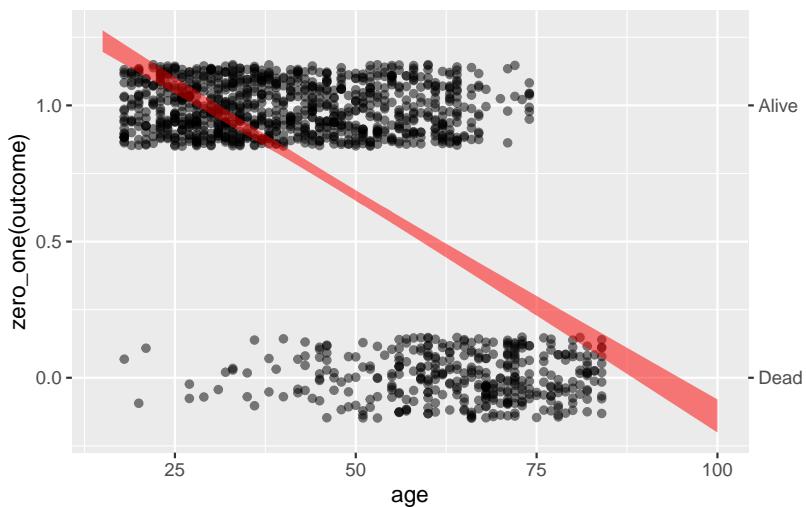


Figure 68: Using linear regression to model the probability of an outcome can lead to situations where the model values go out of the zero-to-one bounds for probability.

inevitably go out of bounds for very large or very small inputs. In contrast, **logistic regression** bends the model output to stay in bounds. (Figure 69) The mathematical means for this is similar in spirit to the way we used the logarithmic and exponential transformation to keep the miles-per-gallon model from producing negative outputs. The transformation is described in Section 21.5

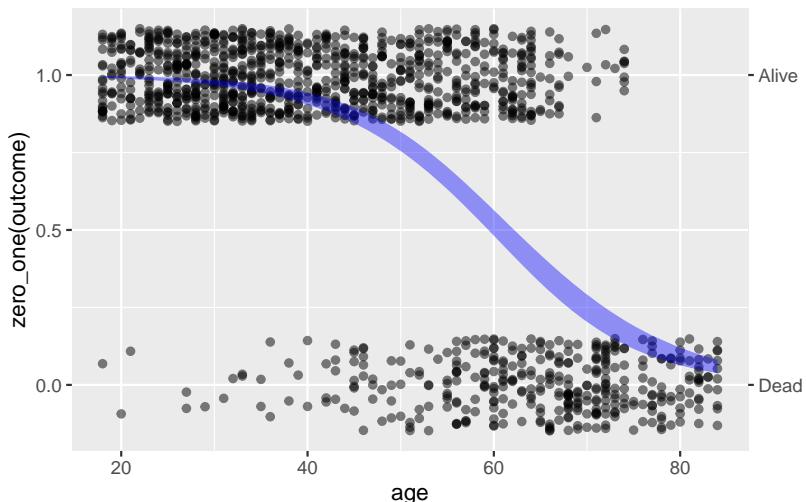


Figure 69: The output of a logistic regression model stays within the bounds zero to one.

`point_plot()` and `model_train()` recognize situations where the response variable is categorical with two levels and automatically use logistic regression.

21.4 Risk

To summarize, for statistical thinkers, a model of risk takes the usual form that we have used for models of zero-one categorical models. All the same issues apply: covariates, DAGs, confidence intervals, and so on. There is, however, a slightly different style for presenting effect sizes.

Up until now, we have presented effect in terms of an arithmetic difference. As an example, we turn to the fuel-economy model introduced at the beginning of this lesson. Effect sizes are about *changes*. To look at the effect size of, say, weight (`wt`), we would calculate the model output for two cars that differ in weight (but are the same for the other explanatory variables). For instance, to know the change in fuel economy due to a 1000 pound change in weight, we can do this calculation:

```
logmpg_mod |>
  model_eval(hp = 100, wt = c(2.5, 3.5)) |>
  mutate(mpg = exp(.output))
```

hp	wt	.lwr	.output	.upr	mpg
100	2.5	2.94	3.173	3.407	23.89
100	3.5	2.736	2.973	3.209	19.55

The lighter car is predicted to get 24 mpg, the heavier car to get 19.5 mpg. The arithmetic difference in output $19.5 - 24 = -4.5$ mpg is the effect of the 1000 pound increase in weight.

There is another way to present the effect, as a **ratio** or proportion. In this style, the effect of an addition 1000 pounds is $19.5/24 = 81\%$, that is, the heavier car can go only 81% of the distance that the lighter car will travel on the same amount of gasoline. (Stating an effect as a ratio is common in some fields. For example, economists use ratios when describing prices or investment returns.)

A change in risk—that is, a change in probability resulting from a change in some explanatory variable—can be expressed as either an arithmetic difference or an arithmetic ratio. A special

terminology that is used to name the two forms. “**Absolute** change in risk refers to the arithmetic difference. In contrast, a proportional change in risk is called a”**relative** risk.”

The different forms—absolute change in risk versus relative risk—both describe the same change in risk. For most decision-makers, the absolute form is most useful. To illustrate, suppose exposure to a toxin increases the risk of a disease by 50%. This would be a risk ratio of 1.5. But that risk ratio might be based on an absolute change in risk from 0.00004 to 0.00006, or it might be based on an absolute change in risk from 40% to 60%. The latter is a much more substantial change in risk and ought to warrant more attention from decision makers interested.

i Other ways to measure change in risk

It is important for measures of change in risk to be mathematically valid. But from among the mathematically valid measures, one wants to choose a form that will be the best for communicating with decision-makers. Those decision-makers might be the people in charge of establishing screening for diseases like breast cancer, or a judge and jury deciding the extent to which blame for an illness ought to be assigned to second-hand smoke.

Two useful ways to present a change in risk are the “**number needed to treat**” (NNT) and the “**attributable fraction**.” The NNT is useful for presenting the possible benefits of a treatment or screening test. Consider these data from the [US Preventive Services Task Force](#) which take the form of the number of breast-cancer deaths in a 10-year period avoided by mammography. The confidence interval on the estimated number is also given.

Age	Deaths avoided	Conf. interval
40-49	3	0-9
50-59	8	2-17
60-69	21	11-32
70-74	13	0-32

The table does not give the risk of death, but rather the absolute risk reduction. For the 70-74 age group this risk reduction is 13/100000 with a confidence interval of 0 to 32/100000.

The NNT is well named. It gives the number of people who must receive the treatment in order to avoid one death. Arithmetically, the NNT is simply the reciprocal of the absolute risk reduction. So, for the 70-74 age group the NNT is 100000/13 or 7700 or, stated as a confidence interval, [3125 to ∞].

For a decision-maker, NNT presents the effect size in a readily understood way. For example, the 40-49 year-old group has an NNT of 33,000. The cost of the treatment could be presented in terms of anxiety prevented (mammography produces a lot of false positives) or monetary cost. The US Affordable Care Act requires health plans to fully cover the cost of a screening mammogram every one or two years for women over 40. Those mammograms each cost about \$100-200. Consequently, the cost of mammography over the ten-year period (during which 5 mammograms might be performed) is roughly $5 \times \$100 \times 33000$ or about \$16 million per life saved.

The attributable fraction is a way of presenting a risk ratio—in other words, a relative risk—in a way that is more concrete than the ratio itself. Consider the effect of smoking on the risk of getting lung cancer. According to the [US Centers for Disease Control](#), “People who smoke cigarettes are 15 to 30 times more likely to get lung cancer.” This statement directly gives the confidence interval on the relative risk: [15 to 30].

The attributable fraction refers to the proportion of disease in the exposed group—that is, smokers—to be attributed to exposure. The general formula for attributable fraction is simple. If the risk ratio is denoted RR , the attributable fraction is

$$\text{attributable fraction} \equiv \frac{RR - 1}{RR}$$

For a smoker who gets lung cancer, the confidence interval on the attributable fraction is [93% to 97%].

For second-hand smoke, the CDC estimates the risk ratio for cancer at [1.2 to 1.3]. For a person exposed to second-hand smoke who gets cancer, the attributable fraction is [17% to 23%]. Such attributions are useful for those, such as judges and juries, who need to assign a level of blame for a bad outcome.

21.5 Probability, odds, and log odds

A probability—a number between 0 and 1—is the most used measure of the chances that something will happen, but it is not the only way nor the best for all purposes.

We use the word “odds” in everyday language. The phrase “What are the odds?” expresses surprise at an unexpected event. The setting for odds is an event that might happen or not: the horse Fortune’s Chance might win the race, otherwise not; it might rain today, otherwise not; the Red Sox might win the World Series, otherwise not. More generally, the setting for odds is an event with a two-level categorical outcome.

Odds are usually expressed as a ratio of two numbers, as in “3 to 2” or “100 to 1”, written more compactly as 3:2 and 100:1. Of course, a ratio of two numbers is itself a number. We can write odds of 3:2 simply as 1.5 and odds of 100:1 simply as 100.

The format of a *probability* assigns a number between 0 and 1 to the chances that Fortune’s Chance will win, or that the weather will be rainy, or that the Red Sox will come out on top. If that number is called p , then the chances of the “otherwise outcome” must be $1 - p$. The event with probability p would be reformatted into odds as $p : (1-p)$. No information is lost if we treat the odds as a single number, the result of the division $p/(1-p)$. Thus, when $p = 0.25$ the corresponding odds will be $0.25/0.75$, in other words, 1/3.

A big mathematical advantage to using odds is that the odds number can be anything from zero to infinity; it’s not bounded within 0 to 1. Even more advantageous for accumulating risk is to arrange the transform so that the output can be *any number*,

positive or negative. This is done by transforming the odds with the logarithm function. The end product of this two-stage, odds-then-log transformation is called the “**log odds**.” We will come back to this later.

The model coefficients in logistic regression (e.g. Figure 69) are in terms of log-odds. For example, consider the coefficients for the model `zero_one(outcome, one = "Alive") ~ age` trained on the Whickham data frame.

```
Whickham |>
  model_train(zero_one(outcome, one ="Alive") ~ age) |>
  conf_interval()
```

Waiting for profiling to be done...

term	.lwr	.coef	.upr
(Intercept)	6.6	7.4	8.2
age	-0.14	-0.12	-0.11

For a hypothetical 20-year old, the model output will be

$$7.403 - 0.1218 \times 20 = 4.967$$

Obviously, 5.05 is not a probability, and it’s not intended to be. Instead, 5.05 is the logarithm of an odds. To convert 5.05 to the corresponding probability involves two steps:

- 1) Undo the logarithm: `exp(4.967)` is 143.6. This is an odds, not yet a probability.
- 2) Convert the odds to a probability. The formula for this is $p = \frac{\text{odds}}{\text{odds}+1} = 143.6/144.6 = 0.993$.

Now consider a hypothetical 100-year-old. The model output is

$$7.490 - 1.22 \times 100 = -114.5.$$

As before, this is in terms of log odds. Using the method for conversion to probability, we get $odds = e^{-114.5} = 1.87 \times 10^{-50}$. This corresponds to a vanishingly small probability. In other words, according to the model, the probability of the 100-year-old being alive 20 years later is practically zero. (But not negative!)

The `model_eval()` function recognizes when its input is a logistic regression model and automatically renders the model output as a probability. (The default for `model_eval()` is to include a prediction interval. But there is no such thing when the model value is itself a probability.)

```
Whickham |>
  model_train(zero_one(outcome, one ="Alive") ~ age) |>
  model_eval(age = c(20, 100), interval = "none")
```

age	.output
20	0.993
100	0.0083

A simple, rough-and-ready way to interpret coefficients in a logistic regression model exists. The intercept sets the *baseline* risk. A positive intercept means a baseline probability greater than 0.5; a negative intercept corresponds to a baseline probability less than 0.5. For each of the other coefficients, a positive coefficient means an increase in risk, while a negative coefficient corresponds to a decrease in risk.

22 Effect size

Starting with this Lesson, we focus on issues surrounding the selection of explanatory variables. Up to now, we've taken a somewhat casual view, pointing out that models with multiple explanatory variables can be made. In discussing prediction models (Lesson 18) we went so far as to say that any set of explanatory (predictor) variables is valid so long as it leads to a good prediction.

This and the next several Lessons deal with statistical modeling methods that support **intervening** in a system. Such interventions occur on both grand scales and small: changes in government policies such as funding for preschool education or subsidies for renewable energy, closing a road to redirect traffic or opening a new highway or bus line, changing the minimum wage, etc. Before making such interventions, it is wise to know what the consequences are likely to be. Figuring this out is often a matter of understanding how the system works: what causes what. As interventions often affect multiple individuals, influencing the overall trend of the effect across individuals might be the goal instead of predicting how each individual will be affected.

Intervention versus prediction

The statistical thinker distinguishes between settings that call for a predictive model and settings that are fundamentally about the consequences of intervening in a system. In a predictive setting, we're interested in the outcome from a system that we do not plan to change fundamentally.

The need for predictions arises in both mundane and critical settings. For instance, an airline needs to consider what will be the demand for a particular route so that they can plan how many aircraft and of what type to serve the route. To set the route schedule, the airline needs a prediction about what will be the demand, which may vary based day of the week, time of day, season of the year, and special events (such as massive convention

or a solar eclipse). Another example: Merchants and social media sites must choose what products or posts to display to a viewer. Merchants have many products, and social media has many news feeds, tweets, and competing blog entries. The people who manage these websites want to promote the products or postings most likely to cause a viewer to respond. To identify viable products or postings, the site managers construct predictive models based on earlier viewers' choices.

In an intervention setting, there is a plan to change how the system works. Such changes can take many forms. For instance, an airline needs to set prices and the availability of different seat classes. Demand will depend on price but also on other factors, such as the requirement to make a connection through a third airport and the competition's prices. A useful model of demand versus price will include such other factors; those factors play a *causal role* in influencing demand.

22.1 Effect size: Input to output

This Lesson focuses on “**effect size**,” a measure of how changing an explanatory variable will play out in the response variable. Built into the previous sentence is an assumption that the explanatory variable *causes* the response variable. In this Lesson we focus on the calculation and interpretation of effect size. Lessons 23 through 26 take a detailed look at how to make responsible claims about whether a connection between variables is causal.

An intervention changes something in the world. Some examples are the budget for a program, the dose of a medicine, or the fuel flow into an engine. The thing being changed is the *input*. In response, something else in the world changes, for instance, the reading ability of students, the patient’s serotonin levels (a neurotransmitter), or the power output from the engine. The thing that changes in response to the change in input is called the “output.”

“Effect size” describes the change in the output with respect to

the change in the input. We will focus here on quantitative output variables. (For categorical output variables, the methods concerning “risk” presented in Lesson 21 are appropriate.)

An effect size (with a quantitative output variable) takes two different forms, depending on whether the explanatory variable is quantitative or categorical. We write “the explanatory variable” because effect sizes concern the response to **changes** in a single explanatory variable, even though there may be others in the model.

22.1.1 Effect size for *quantitative* explanatory variable

When the explanatory variable is *quantitative*, the effect size is a **rate**. Rates are always *rations*: the change in output divided by the change in input that caused the output to change. For instance, in the Scottish hill racing setting considered in Lesson 13.3 we modeled running **time** as a function of race **distance** and **climb**. Such a model will involve two effect sizes: the change in running time per unit change in distance; and the change in running time per unit change in climb.

Effect sizes typically have units. These will be the unit of the output variable divided by the unit of the explanatory variable. In the effect size of **time** with respect to **distance**, the effect-size unit will be seconds-per-kilometer. On the other hand, the effect size of **time** with respect to **climb** will have units seconds-per-meter.

Here is one way to calculate an effect size: change a single input by a known amount, measure the corresponding change in output, and take the ratio. For example:

```
race_mod <- Hill_racing |> model_train(time ~ distance + climb)
```

To calculate the effect size on **time** with respect to **distance**, we evaluate the model for two different distances, keeping **climb** at the same level for both distances.

```
race_mod |> model_eval(distance = c(5, 10), climb = 500)
```

distance	climb	.lwr	.output	.upr
5	500	395.1	2104	3813
10	500	1664	3373	5082

The output changed from 2104 seconds to 3373 seconds in response to changing the value of `distance` from 5 moving to 10 km. The effect size is therefore

$$\frac{3373 - 2104}{10 - 5} = \frac{1269}{5} = 253.8 \text{ s/km}$$

To calculate the effect size on `time` with respect to `climb`, a similar calculation is done, but holding `distance` constant and using two different levels of `climb`:

```
race_mod |> model_eval(distance = 10, climb = c(500,600))
```

distance	climb	.lwr	.output	.upr
10	500	1664	3373	5082
10	600	1925	3634	5343

The effect size is:

$$\frac{3634 - 3373}{100} = \frac{261}{100} = 2.6 \text{ s/m}$$

To see how effect sizes might be used in practice, put yourself in the position of a member of a committee establishing a new race. The new race will have a distance of 17 km and a climb of 600 m. The anticipated winning time in the new race will be a matter of *prediction*:

```
race_mod |> model_eval(distance = 17, climb = 600)
```

distance	climb	.lwr	.output	.upr
17	600	3701	5411	7120

Note how broad the prediction interval is: from about one hour up to two hours.

Debate ensues. One faction on the committee wants to shorten the race to 15 km and 500 m climb. How much will this lower the winning time?

On its own, the -2 km change in the race `distance` will lead to an approximately will lead to a winning time shorter by

$$-2 \text{ km} \times 253.8 \text{ s/km} = -508 \text{ s}$$

where 253.8 skm is the effect size we calculated earlier.

The previous calculation did not consider the proposed reduction in `climb` from 600 m to 500 m. On its own, the -100 m change in race `climb` will also shorten the winning time:

$$-100 \text{ m} \times 2.6 \text{ s/km} = -260 \text{ s}$$

Each of these two calculations of change in output looks at only a single explanatory variable, not both simultaneously. To calculate the overall change in race `time` when both `distance` and `climb` are changed, add the two changes associated with the variables separately. Thus, the overall change of the winning time will be

$$(-508 \text{ s}) + (-260 \text{ s}) = -768 \text{ s}.$$

i Comparing predictions?

The predicted winning race time for inputs of 17 km `distance` and 600 m `climb` was [3700 to 7100] seconds. What if we make a second prediction with the proposed changes in distance and time, and subtract the two predictions?

```
race_mod |> model_eval(distance = 15, climb = 500)
```

distance	climb	.lwr	.output	.upr
15	500	2933	4642	6351

The shorter race has a predicted winning time of [2900 to 6400] seconds.

Question: How do you subtract one *interval* from another? Should we look at the worst-case difference: [(6400 - 3700) to (2900 - 7100)], that is, [-4200 to 2700] seconds? Or perhaps we should construct the difference as the change between the lower ends of the two prediction intervals up to the change in the upper ends? That will be [(2900 - 3700) to (6400 - 7100)], that is, [-800 to -700] s.

A good perspective on this question of the difference between intervals is based on the distinction between the part of the **time** that is *explained* by **distance** and **climb**, and the part of **time** that remains unexplained, perhaps due to weather conditions or the rockiness of the course. If the committee decides to change the course **distance** and **time** it will not have any effect on the weather or course rockiness; these factors will remain random noise. The lower end of each prediction interval reflects one extreme weather/rockiness condition; the upper end reflect another extreme of weather/rockiness. Apples and oranges. The *change* in race **time** due to **distance** and **time** should properly be calculated at the same weather/rockiness conditions. Thus, the [-800 to -700] s estimate of the change in running **time** is more appropriate. The effect-size calculation does the apples-to-apples comparison.

22.1.2 Effect size for *categorical* explanatory variable

When an explanatory variable is categorical, the change in input must always be from one level to another. For example, an airline demand model might involve a day-of-week variable with levels “weekday” and “weekend.” To calculate the effect size on demand with respect to day-of-week, all you can do is measure the corresponding change in the model output when day-of-week is changed from “weekday” to “weekend.” The effect size will simply be this change in output, not a rate. Calculating a rate would mean quantifying the change in input, but weekday-to-weekend is not a number.

22.2 Model coefficients and effect size

For simplicity in these Lessons, we emphasize models where the explanatory variables contribute *additively*, as implicit in the use of `+` in model specifications like `time ~ distance + climb`. More generally, both additive and *multiplicative* contributions can be used in models. (Similarly, it's possible to use curvey transformations of variables.) In Section 22.3 we will investigate the uses of multiplicative contributions.

In models incorporating multiplicative or curvey contributions, effect size can be calculated using the `model_eval()`-based method described in Section 22.1.1. But, for models where explanatory variables contribute additively, there is an easy shortcut for calculating effect size: *the coefficient on each explanatory variable is the effect size for that variable*.

To illustrate, look at the coefficients on the `time ~ distance + climb` model:

```
race_mod |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-533.4	-470	-406.5
distance	246.4	253.8	261.2
climb	2.493	2.61	2.726

The `.coefficients` on `distance` and on `climb` are the same as we calculated using the `model_eval()` method!

Moreover, for additive models, the *confidence interval* on the coefficient also expresses the confidence interval on the corresponding effect size. So, when in Section 22.1.1 we said the effect size of `distance` on `time` was 253.8 s/km, a better statement would have been as an interval: [246 to 261] s/km.

22.3 Interactions

The model `time ~ distance + climb` combines the explanatory variables additively. Figure 70 shows the “shape” of

the model graphically in two different ways: with `distance` mapped to x and `climb` mapped to color (left panel) and with `climb` mapped to x and `distance` to color (right panel). The same model function is shown in both; just the presentation is different. In both panels, the model function appears as a set of **parallel** sloped lines. This is the hallmark of an additive model. (See Figure 24 for another example.)

```
Hill_racing |> filter(climb > 100) |>
  point_plot(time ~ distance + climb, annot = "model",
             model_ink = 1)
Hill_racing |> filter(climb > 200) |>
  point_plot(time ~ climb + distance, annot = "model",
             model_ink = 1)
```

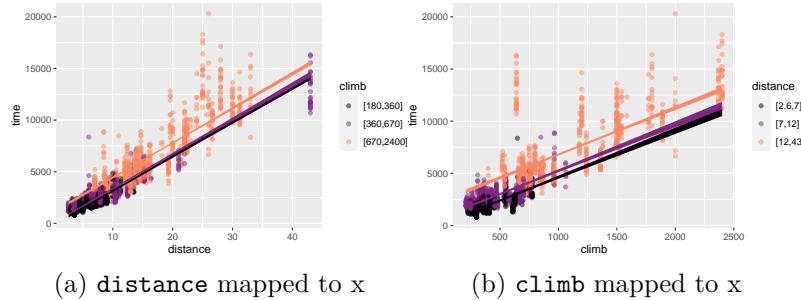


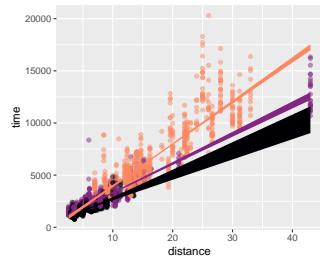
Figure 70: Two views of the additive model `time ~ distance + climb`. The lines for different colors are parallel.

The effect size of the variable being mapped to x appears as the slope of the lines. The effect size of the variable mapped to color appears as the vertical separation between lines. Figure 70 shows that the effect of `distance` and the effect of `climb` do not change when the other variable changes; the lines are parallel.

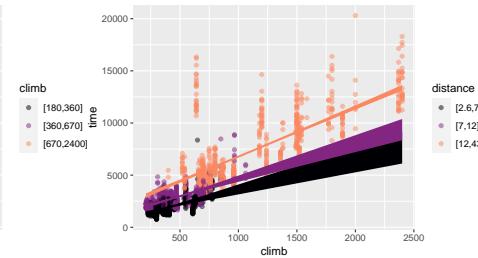
In contrast, Figure 71 gives views of the multiplicative model `time ~ distance * climb`. In Figure 71, the spacing between the different colored lines is not constant; the lines fan out rather than being parallel.

```
Hill_racing |> filter(climb > 100) |>
  point_plot(time ~ distance * climb, annot = "model",
             model_ink = 1)
```

```
Hill_racing |> filter(climb > 200) |>
  point_plot(time ~ climb * distance, annot = "model",
             model_ink = 1)
```



(a) distance mapped to x



(b) climb mapped to x

Figure 71: Two views of the **multiplicative** model `time ~ distance * climb`. The lines fan out.

Again, the effect size of the variable mapped to color appears as the vertical spacing between the different colored lines. Now, however, that vertical spacing changes as a function of the variable mapped to x. That is, the effect size of one explanatory variable depends on the other.

The model coefficients show the contrast between additive and multiplicative models. For the additive model, there is one coefficient for each explanatory variable. That variable's coefficient captures the effect size of the variable.

```
Hill_racing |> model_train(time ~ distance + climb) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-533.4	-470	-406.5
distance	246.4	253.8	261.2
climb	2.493	2.61	2.726

For the multiplicative model, there is a third coefficient. The model summary reports this as `distance:climb`. Generically, it is called the “**interaction coefficient**.” The interaction coefficient quantifies how the effect of each explanatory variable depends on the other.

```
Hill_racing |> model_train(time ~ distance * climb) |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-165.7	-59.18	47.38
distance	214.5	224.1	233.7
climb	1.576	1.784	1.992
distance:climb	0.03494	0.04426	0.05358

You can't read the effect size for an explanatory variable from a single coefficient. Instead, arithmetic is required. For instance, the effect size of `distance` is not just the quantity reported as the `.coef` on `distance`: 224 s/m. Instead, the effect size of `distance` is a function of `climb`:

Effect size of `distance` : $224 + 0.044 \times \text{climb}$

Effect size of `climb` : $1.78 + 0.044 \times \text{distance}$

Each of the above formulas is for an effect size: how the model output changes when the corresponding explanatory variable changes. In contrast, the model function gives `time` as a function of `distance` and `climb`. The model function is:

Model function: `time(distance, climb) = -59.2 + 224 × distance + 1.78 × climb + 0.044 × distance × climb`

i For the reader who has already studied calculus:

The effect sizes are the partial derivatives of the model function. The interaction coefficient is the “mixed partial derivative” of the function with respect to both `distance` and `climb`.

$$\text{Effect size of } \text{distance} : \frac{\partial \text{time}}{\partial \text{distance}}$$

$$\text{Effect size of } \text{climb} : \frac{\partial \text{time}}{\partial \text{climb}}$$

23 Directed acyclic graphs

As a verb, **to influence** means to affect a person, object, or condition. Examples: The shortening days of autumn influence my mood. The teacher influences the student's education, that is, the assimilation of facts, concepts, and methods. Education influences later job prospects.

A thing being influenced is called a **consequence**.

As a noun, **influence** refers to a capacity to influence a consequence. There are degrees of influence. At one extreme, the influence may completely determine the consequence. On the other hand, a particular influence is just one of multiple factors that shape the overall consequence. Randomness—noise—may also contribute to the consequence.

A “**network**” is a set of elements and links that tie them together. For instance, the internet is a vast set of computers and communication channels that connect them. A **causal network** is a set of consequences and influences that connect them.

Causal networks provide an excellent way to envision and understand the mechanisms at work in the real world. They are essential to decision-making since decision-making aims to direct action that will have a desired consequence in the real world.

This Lesson is about the representation of causal networks by diagrams. The technical term for such diagrams is “**directed acyclic graphs**” (DAGs). A less offputting name is “**influence diagrams**.”

23.1 Influence diagrams

The first paragraph of this lesson contains three sentences describing influences. Each sentence has the form, “X influences Y.” Part of translating such a form into an influence diagram involves replacing “influences” with the symbol →.

Diagrams are easier to read if we use short names for the consequences on either side of \rightarrow . With an eye toward our eventual use of influence diagrams to interpret data, using *variable names* for the consequences is helpful. But it is often desirable to include in an influence diagram a consequence that is not recorded as a variable. In the jargon of causal networks, such an unrecorded variable is called a “**latent variable**,” the word “latent” coming from the Latin for “hidden.”

It’s time to simplify a little. We now have three words being used for things that influence or things that are influenced: consequence, variable, and latent variable. Let’s use the short word “**node**” to stand for any of these three.

Here are possible translations of the sentences in the first paragraph into influence diagrams:

Sentence	Influence diagram
“The shortening days of autumn influence my mood.”	<code>daylight_trend → teachers_mood</code>
“The teacher influences the student’s education.”	<code>teachers_mood → student_skills</code>
“Education influences later job prospects.”	<code>student_skills → student_job_prospects</code>

Notice that the influence diagrams given above are not complete translations of the English sentence. Starting at the bottom, `student_skills` are not the only component of “education.” The other components of education may also influence job prospects directly or indirectly. The `teachers_mood` is hardly the only attribute of the teacher that contributes to `student_skills`. There is also the teacher’s experience, knowledge, sympathy, enthusiasm, articulateness.

Influence diagrams are assembled from smaller influence diagrams. For instance, a larger diagram can incorporate all three small diagrams into which we translated the sentences.

```
daylight_trend → teachers_mood → student_skills →
student_job_prospects
```

The above diagram is a *chain* of nodes. Other network shapes are also possible. To run with the daylight/mood/skills/prospects example, what about the student's mood, which may also be influenced by daylight and influence the assimilation of skills and job prospects? Figure 72 shows one possible arrangement.

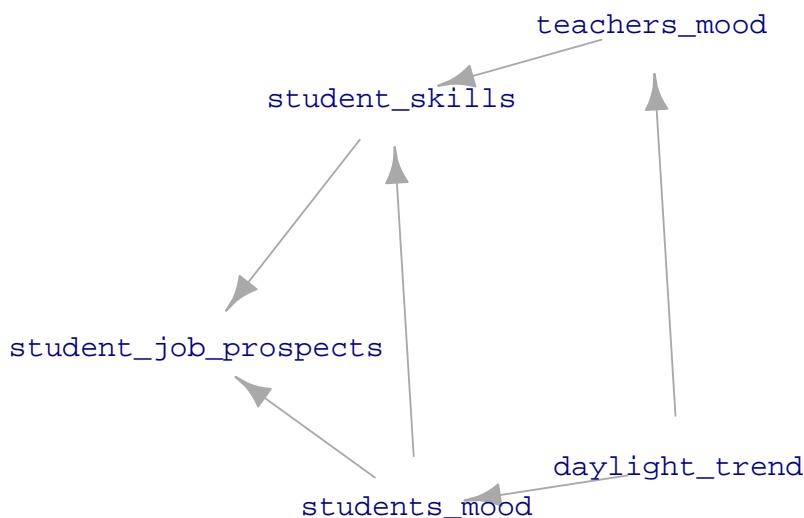


Figure 72: An influence diagram linking seasonal trends in daylight length to a student's job prospects.

The word “influence” comes from the Latin word for “flow into,” as in fluids flowing through pipes or streams flowing into rivers and lakes. The arrows in influence diagrams show the sources, destinations, and flow directions. The diagram itself doesn’t describe what substance is flowing. I like to think of it as “causal water.” By tinting with dye the causal water coming from a node, one could track the flow from that node to the other nodes in the diagram. In Lesson 24 we will come back to the issue of such flow paths see how the choice of explanatory variables in modeling can effectively block or unblock a flow pathway. Similarly, scientific experiment can be thought of as taking control over a node, cutting off its natural inflow.

Remember that an influence diagram is a *drawing*; it is not the

real world. At best, we can say that an influence diagram is a **hypothesis** about real-world connections. It's usually best to entertain *multiple hypotheses* (as in Lesson 16) to help you think carefully about the paths and directions of the flow of causation. As well, many debates in science, government, and commerce can be represented as reflecting different hypotheses about the causal connections in the real world.

23.2 Nodes

In an influence diagram, each node can have zero or more inputs. For example, the `student_skills` node in Figure 72 has two inputs: `students_mood` and `teachers_mood`. The `daylight_trend` node has *no* inputs shown in the diagram. This is just a convention for saying that we are not interested in the inputs upstream from `daylight_trend`; it might as well be pure noise so far as we are concerned. The `teachers_mode` has just one input, coming from `daylight_trend`.

Contrary to how the diagrams are drawn, every node has precisely **one output**. A node such as `daylight_trend` may be drawn with two or more outward-pointing arrows, but all the arrows originating from a node carry the same thing to their respective destinations. Sometimes, nodes are drawn with no outputs. Again, this convention says we are not concerned with any of those influences.

The node itself is drawn as a name: a label for the node. But there is something else in the node, even though it is not usually shown in the influence diagram. Every node has a **mechanism** that puts together the inputs (and often some noise) to produce the output.

The **simulations** introduced in Lesson 14 are a list of node names along with the mechanism for that node. The mechanism is expressed using R expressions. Each input to the mechanism is identified by the name of the node from which the input originates.

Consider `sim_07`, one of the simulations packaged with the `{LSTbook}` package that comes along with these Lessons. To

see the nodes and the mechanism within each node, just print the simulation:

```
print(sim_07)
```

Simulation object

```
-----
[1] a <- rnorm(n)
[2] d <- rnorm(n)
[3] b <- rnorm(n) - a
[4] c <- a - b + rnorm(n)
```

`sim_07` has four nodes, uncreatively labelled `a`, `b`, `c`, and `d`. Nodes `a` and `d` do not have any inputs; they are pure noise. (The particular noise model here is `rnorm()`, the normal noise model. But other noise models could have been used.)

In contrast, node `b` has one input. The mechanism is `rnorm(n) - a`, which says that the output will be noise minus the value of node `a`. The mechanism of node `c` is somewhat richer; it has `a` and `b` as inputs and some random noise.

The symbol `n` in a simulation object is unique. It is neither a node nor an input to the mechanism. `n` is there just for compatibility of the simulation system with the built-in R random number generators.

To draw the influence diagram for `sim_07`, use the `dag_draw()` function.

```
dag_draw(sim_07)
```

Let's track the calculations for a sample of $n = 1$, that is, one row from a data frame produced by `sim_07`.

```
set.seed(429)
sim_07 |> sample(n=1)
```

	a	d	b	c	
:	---	-----	-----	-----	:
	0.5	0.1754	-1.863	4.352	

You don't need to use the `print` function explicitly as was done here. Just `sim_07` would accomplish the same thing.

d

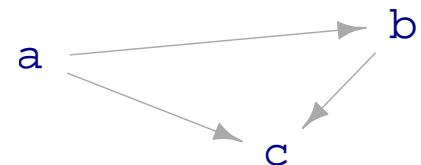


Figure 73: The influence diagram for `sim_07`. Note that node `d` has no connections to or from the other nodes.

In forming this output row, `sample()` looks at its input (`sim_07`). It evaluates the mechanism for the first node in the list. But the special symbol `n` is replaced by 1, like this

```
rnorm(1)
```

```
[1] 0.4999627
```

This value is stored under the name `a`, for future reference.

The simulation goes on to the next node in the list. For `sim_07` this is node `d`. The mechanism happens to be the same as for node `a`, but it's the nature of random number generators to give a different result each time the generator is used.

```
rnorm(1)
```

```
[1] 0.1753615
```

This value is stored under the name `d`.

On to the next node, `b`. The mechanism is evaluated to produce a value:

```
rnorm(1) - a
```

```
[1] -1.8632
```

Storing this result unde the name `b`, the simulation engine goes on to the next node. That's the last node in `sim_07`, but other simulations may have more nodes, each identified by name and given a mechanism.

If we had asked `sample()` to generate more than one row of data, it would have repeated this process anew for each additional row, independently of the rows that have already been generated or the rows that are yet to be generated.

Because each row is independent of every other row, there is no way for a node's mechanism to refer to the node itself. For instance, we might imagine a feedback relationship like this:

```
Cycle_sim <- datasim_make(  
  a <- 2 - b, # Illegal!  
  b <- a + b # Illegal!  
)
```

The `datasim_make()` function is designed to recognize self-referential situations and cycles where a path of arrows circles back on itself. Here's what happens when there is such an issue:

```
Error in igraph::topo_sort(datasim_to_igraph(sim, report_hidden = TRUE)): At core/properties/d
```

As is often the case, the error message contains much information that might be valuable only to a programmer. For an end-user, the critical part of the message is “The graph has cycles.” Not allowed

Directed Acyclic Graphs

The standard name used in the research literature, instead of “influence diagram,” is “**directed acyclic graph**” (DAG for short.) From now on, we will mostly say DAG instead of influence diagram. This will help you form the habit of using the name “DAG” yourself.

DAGs, despite the G for “graph,” are not about data graphics. The “graph” in DAG is a mathematical term of art; a suitable synonym is “network.” Mathematical graphs consist of a set of “nodes” and a set of “edges” connecting the nodes. For instance, Figure 74 shows three different graphs, each with five nodes labeled A, B, C, D, and E.

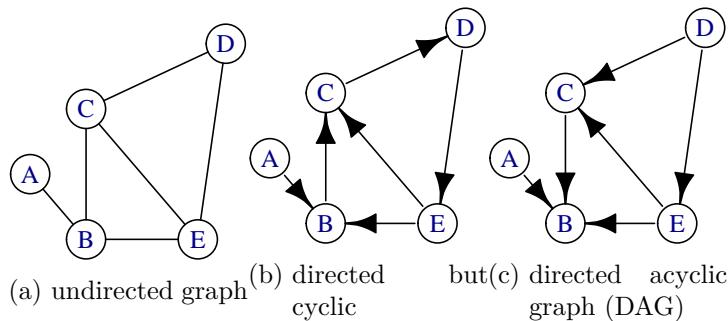


Figure 74: Graphs of various types

The nodes are the same in all three graphs of Figure 74, but each is different. It is not just the nodes that define a graph; the edges (drawn as lines) are part of the definition as well.

The left-most graph in Figure 74 is an “**undirected**” graph; there is no suggestion that the edges run one way or another. In contrast, the middle graph has the same nodes and edges, but the edges are **directed**. As mentioned earlier, an excellent way to think about a directed graph is that each node is a pool of water; each directed edge shows how the water flows between pools. This analogy is also helpful in thinking about causality: the causal influences flow like water.

Look more carefully at the middle graph. There are a couple of loops; the graph is **cyclic**. In one loop, water flows from E to C to D and back to E. The other loop runs B, C, D, E, and back to B. Such a flow pattern cannot exist, at least, not without pumps pushing the water back uphill! There is nothing in a DAG that corresponds to a pump.

The rightmost graph reverses the direction of some of the edges. This graph has no cycles; it is **acyclic**. Using the flowing and pumped water analogy, an acyclic graph needs no pumps; the pools can be arranged at different heights to create a flow exclusively powered by gravity. The node-D pool will be the highest, E lower. C has to be lower than E for gravity to pull water along the edge from E to C. The node-B pool is the lowest, so water can

flow in from E, C, and A.

Directed acyclic graphs represent causal influences; think of “A causes B,” meaning that causal “water” flows naturally from A to B. In a DAG, a node can have multiple outputs, like D and E, and it might have multiple inputs, like B and C. In terms of causality, a node—like B—having multiple inputs means that more than one factor contributes to the value of that node. A real-world example: the rising sun causes a rooster to crow, but so can a fox approaching the chicken coop at night.

Often, nodes do not have any indicated inputs. These are called “**exogenous factors**,” at least by economists. The “genous” means “originates from.” “Exo” means “outside.” The value of an exogenous node is determined by something, just not something that we are interested in (or perhaps capable of) modeling. No edges are directed into an exogenous node since none of the other nodes influence its value.

24 Causal influence and DAGs

Lesson 23 introduced Directed Acyclic Graphs (DAGs) to express our ideas about what causes what. Our first use for DAGs was to visualize the connections between the variables in a *simulation*.

This Lesson covers an important application for DAGs when building useful models with data: DAGs can help us choose covariates for models in a way that respects what we want to find out about a particular causal connection between an explanatory variable and the response variable.

As you saw in Lessons 14 and 23, we can define a DAG by listing all the direct causal connections between nodes. Using the simulation notation to illustrate, here is a simple simulation of the influence of a medical `drug` on `mortality`, where the medical drug is presumed to reduce mortality, but where the only patients prescribed the drug are those who are unhealthy.

```
drug_mortality_sim <- datasim_make(  
  health <- rnorm(n),                                     ①  
  drug    <- - health + 0.5*rnorm(n),                      ②  
  mortality <- - drug - 2 * health + rnorm(n)           ③  
)
```

- ① Outside factors determine the `health` of the person. Presumably, these relate to risk factors such as age, genetics, lifestyle, and so on, but the simple model doesn't go into detail; a person's health is assigned randomly. A bigger number indicates better health.
- ② In the simulation, whether or not the person takes the `drug` depends on the subject's `health`. People who are in good health are not prescribed the medication. This is why there is a negative sign in the `drug <- - health` mechanism.
- ③ The general `health` of the patient influences `mortality`: the better the health, the lower the mortality. This is why there is a negative sign in front of `health` for the `mortality` variable. Similarly, if the patient is prescribed and takes the `drug`, the patient's mortality is lower.

Figure 75 presents the simulation as a DAG. Every variable on the left-hand side of \leftarrow has an input from all the variables mentioned on the right-hand side. For instance, `mortality` has two inputs: from `drug` and from `health`. But `health` has no inputs because no variables are mentioned on the right-hand side of the \leftarrow for `health`.

Suppose the question of interest is whether the drug improves mortality. A simple design for this study is to sample many people, noting for each person whether they take the `drug` and whether the person died in the next, say, five years. With the data in hand, we'll build a model, `mortality ~ drug` and check whether the effect size of `drug` on `mortality` is positive (harmful drug!) or negative (drug reduces mortality).

Our intention with `mortality ~ drug` is to probe the direct causal link between `drug` and `mortality`. But will we achieve our intention with the model `mortality ~ drug`? There is a difficulty in this example: the direct `drug → mortality` causal link is only one of the *pathways* between `drug` and `mortality`. The other pathway is `drug ← health → mortality`. When we model `mortality ~ drug`, will we be studying just the direct link, or will the other pathway get involved? This is the sort of question we will address in this Lesson, as well as how to *block* a pathway that we don't intend to study because we are interested in the pharmacological action of the drug, not the role of `health` in determining mortality.

i A DAG is a hypothesis

The simulation and DAG picture in the above example are made up. They reflect our idea of how a drug might be related to mortality. The word we will prefer for such ideas is “**hypothesis**,” although “theory”, “speculation,” or “assumption” would serve almost as well.

A hypothesis is a statement that might or might not be true. A common purpose for a hypothesis is to organize thinking around an **assumed** state of the world. But just because we *assume* a hypothesis, does not mean the hypothesis is true. Any conclusions that we draw from hypothetical thinking are **conditioned** on the assumption that the hypothesis is true.

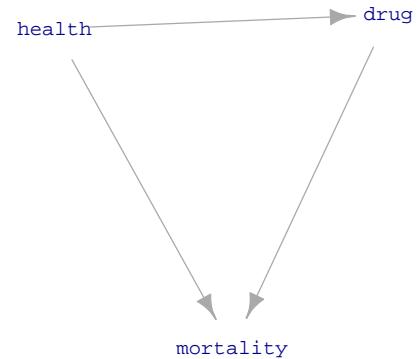


Figure 75: The `drug_mortality_sim` drawn as a DAG.

tion; they should not be regarded as a definitive statement about the world.

The author F. Scott Fitzgerald (1896-1940) famously wrote:

“The test of a first rate intelligence is the ability to hold two opposed ideas in the mind at the same time, and still retain the ability to function.”

The statistical thinker is often in the position of having to keep in mind two or more hypotheses about the causal connections in a system. Different hypotheses may lead to different conclusions. DAGs provide a way to write down a causal hypothesis and, as we will see, choose appropriate covariates for uncovering a particular causal link or pathway. When different hypotheses lead to different choices of covariates, the statistical thinker has to retain the ability to function, recognizing the conditional truth of the conclusions derived from each hypothesis. Ideally, when faced with contradictory conclusions conditioned on different hypotheses, the statistical thinker can use a clever choice of covariates or an experiment (Lesson 26) to resolve the issue.

24.1 Pathways

A DAG is a network. In a complicated roadway network like the street grid in a city or the highway system, there is usually more than one way to get from an origin to a destination. In the language of DAGs, we use the word “**pathway**” to describe a particular route between the origin and destination. Even a simple DAG, like that in Figure 75, can have multiple pathways, like the two we identified in the previous section between `drug` and `mortality`.

A good metaphor for a DAG is a network of *one-way* streets. On a one-way street, you can drive your car in one direction but not the other. In a DAG, *influence* flows in only one direction for any given link.

The one-way street network metaphor fails in an important way. Influence is not the only thing we need to keep track of in a DAG. *Information* is another entity that can seem to “flow” through a DAG. To illustrate, consider this simple DAG:

$$Y \leftarrow C \rightarrow X$$

In this DAG, there is no way for influence to flow from X to Y; the one-way links don’t permit it. We have used water as an analogy for causal influence. For *information*, we need another analogy. Consider *ants*.

We will allow the ants to move in only one direction along a link. So in $Y \leftarrow C \rightarrow X$, ants can move from C to Y. They can also move from C to X. But an individual ant is powerless to move from X to Y or vice versa.

A particular property of *ants* is that we don’t usually consider them individuals but a collective, a colony. When we see ants in two different places, we suspect that those two places are connected by some ant pathway, even if we can’t figure out whether the ants originated in one of the places or the other.

In the $Y \leftarrow C \rightarrow X$ network, an ant colony at C can generate ant sightings at both X and Y even though an individual ant can’t move from X to Y. That is, $Y \leftarrow C \rightarrow X$ has an ant connection between Y and X and vice versa.

Our data consists only of ant sightings. Two variables being connected is indicated by simultaneous ant sightings at each of the two nodes representing the variables. We will call the type of connection where ants can show up at two nodes a **“correlating pathway”** between the two nodes.

$Y \leftarrow C \rightarrow X$ is a correlating pathway between X and Y. So is $Y \leftarrow C \leftarrow X$. But $Y \leftarrow C \leftarrow X$ is also a **causal pathway**. When an individual ant can travel from X to Y, we have a causal pathway. But we have a *correlating pathway* when ants from the same colony can appear at X and Y. Every causal pathway is also a correlating pathway because if an individual ant can travel from X to Y, then ants from the same colony can be sighted at both X and Y.

There is another kind of pathway: the **non-correlating pathway**. With a non-correlating pathway between X and Y, there is no way for ants from the colony to show up at both X and Y. For example

$$\text{Non-correlating pathway: } Y \rightarrow C \leftarrow X$$

Try it out. Is there any single node where you can place an ant colony and get ant sightings at both X and Y? If not, you've got a non-correlating pathway.

Correlating pathways create connections between two variables, X and Y, even when there is no causal influence that runs from X to Y. This becomes a problem for the data analyst, who is often interested in causal connections but whose tools are rooted in detecting *correlations* between variables.

i Correlation *is* causation

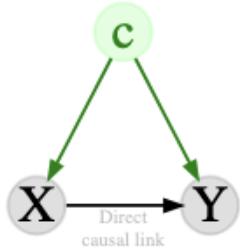
Conventional statistics courses emphasize this motto: “Correlation **is not** causation.” This is true to the same extent that ants and water are different things: ants **are not** water.

Suppose we detect a correlation between X and Y, e.g. a non-zero coefficient on X in the model $Y \sim X$. In that case, a causal connection provides a reasonable explanation for the correlation. But we can't say what the direction of causation is just by analyzing X and Y data together. Even a correlating pathway is constructed out of causal segments. The challenge for the statistical thinker is to figure out the nature of the causal connections from the available data, that is, the flow of an appropriate DAG.

If our data include only X and Y, the situation is hopeless. A non-zero coefficient for the $Y \sim X$ model might be the result of a causal path from X to Y, or a causal path from Y to X, or even a correlating pathway between X and Y mediated by some other variable C (or multiple other variables, C, D, E, etc.). Similarly, a zero coefficient for the $Y \sim X$ model is no guarantee that there is no causal connection between them.

24.2 Blocking correlating and non-correlating pathways using covariates

Here is a DAG with links drawn in different colors to help distinguish the **direct link** between X and Y, which is drawn in black, and the **backdoor pathway** involving node C is drawn in green.



Our interest in DAGs relates to a question: should a covariate C be included in a model when the purpose is to study *specifically the direct relationship from X to Y*? The answer, to be demonstrated experimentally below, is simple.

If the backdoor pathway is **correlating**, include covariate C to **block** that pathway. On the other hand, if the backdoor pathway is **non-correlating**, including covariate C will **unblock** the pathway. Consequently, for non-correlating backdoor pathways, **do not** include covariate C.

In this section, we will conduct a numerical experiment to look at three simple arrangements of backdoor X-C-Y pathways. For

each pathway, the experiment consists of 1) making a simulation, 2) generating data from that simulation, and 3) modeling the data in two ways: $Y \sim X$ and $Y \sim X + C$. We can then check whether including or excluding the covariate C in the model reveals any connection between X and Y .

In each of the three cases, the direct causal link between X and Y will have an X multiplier of -1 . This makes it easy to check whether the model coefficient on X is correct or whether the backdoor pathway interferes with seeing the direct $X \rightarrow Y$ pathway.

i Experiment A: Mediated causal backdoor pathway

$$X \rightarrow C \rightarrow Y$$

::: {.cell}

```
pathA <- datasim_make(
  X <- rnorm(n),
  C <- 1 * X + rnorm(n),
  Y <- 2 * C + rnorm(n) - X
)
```

①

1. Note: The $- X$ is the direct causal connection between X and Y .

```
pathA |> sample(n=10000) -> dataA
dataA |> model_train(Y ~ X) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	0.02828
X	1.018

$Y \sim X$ gives the *wrong* answer. The coefficient on X should be -1 .

```
dataA |> model_train(Y ~ X + C) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	0.01877
X	-1.012
C	2.013

Adding the covariate C to the model produces the correct -1 coefficient on X. :::

i Experiment B. Common cause backdoor pathway

$$X \leftarrow C \rightarrow Y$$

::: {.cell}

```
pathB <- datasim_make(
  C <- rnorm(n),
  X <- 1 * C + rnorm(n),
  Y <- 2 * C + rnorm(n) - X
)
```

①

1. Again, the direct influence of X on Y is the $- X$ term.

```
pathB |> sample(n=10000) -> dataB
dataB |> model_train(Y ~ X) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	-0.001716
X	0.01966

Incorrect result. The coefficient on X should be -1 .

```
dataB |> model_train(Y ~ X + C) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	-0.002309
X	-0.9863
C	1.982

Correct result: X coefficient is -1 . :::

i Experiment C. Common consequence backdoor pathway

$$X \rightarrow C \leftarrow Y$$

```
pathC <- datasim_make(
  X <- rnorm(n),
  Y <- rnorm(n) - X,
  C <- 1 * X + 2 * Y + rnorm(n)
)
```

①

① Again, note the $- X$ in the mechanism for Y

```
pathC |> sample(n=10000) -> dataC
dataC |> model_train(Y ~ X) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	0.01171
X	-0.9976

The word “collider” is preferred by specialists in causality to describe the situation I’m calling a “common consequence.”

Correct result. The coefficient on X is -1 .

```
dataC |> model_train(Y ~ X + C) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	-0.0008783
X	-0.601
C	0.3997

Incorrect result: X should be -1 .

To summarize the three experiments:

Experiment	Pathway	Correlating pathway?	Include covariate?
A	$X \rightarrow C \rightarrow Y$	Yes	Yes
B	$X \leftarrow C \rightarrow Y$	Yes	Yes
C	$X \rightarrow C \leftarrow Y$	No	No

24.3 DAGs and data

People often disagree about what causes what. Ideally, you could use data to resolve such disputes. Under what conditions is this possible?

The question arises because there can be situations where it can be impossible to resolve a dispute purely through data analysis. We can illustrate a very simple system: $Y \leftrightarrow X$. By this, we mean any of the following three systems:

- i. $Y \leftarrow X$. Let's suppose this is Ava's view.
- ii. $Y \rightarrow X$. Let's suppose Booker holds this view.
- iii. No connection at all between X and Y. Cleo holds this view.

Simulation allows us to create a world in which the causal connections are exactly known. For example, here is a simulation in which Y causes X.

```
XYsim <- datasim_make(  
  Y <- rnorm(n), # exogenous  
  X <- 2 * Y + rnorm(n)  
)
```

Imagine three people holding divergent views about the nature of variables X and Y. They agree to resolve their disagreement by collecting data from X and Y, collecting many specimens, and measuring X and Y on each.

In a real-world dispute, concerns might arise about how to sample the specimens and the details of the X and Y measurement techniques. Such concerns suggest an awareness that factors other than X and Y may be playing a role in the system. The correct course of action in such a case is to be explicit about what these other factors might be, expand the DAG to include them, and measure not just X and Y but also, as much as possible, other covariates appearing in the DAG.

Nevertheless, we will show what happens if the parties to the dispute insist that only X and Y be measured. Let's play the role of Nature and generate data for them:

```
XYdata <- XYsim |> sample(n=1000)
```

Ava goes first. “I think that X causes Y. I’ll demonstrate by fitting $Y \sim X$ to the data.

```
Ava_model <- XYdata |> model_train(Y ~ X)  
Ava_model |> conf_interval() |> select(term, .coef)
```

term	.coef
(Intercept)	0.005304
X	0.3963

“You can tell from the X coefficient that X influences Y,” says Ava.

Unexpectedly, Cleo steps in to point out that the coefficient of 0.4 might just be due to accidental alignments of the unconnected X and Y variables.

Ava, who has already read Lesson 20, points out an accepted way to assess whether the 0.4 coefficient might be an accident: look at the confidence intervals.

```
Ava_model |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-0.022	0.0053	0.033
X	0.38	0.4	0.41

The ends of the confidence interval on the X coefficient are far from zero; the interval refutes any claim that the X coefficient is actually zero. “Moreover,” Ava gloats, “my model’s R^2 is 78%, very close to 1.”

```
Ava_model |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
1000	1	0.7966	3909	0.7964	0	1	998

Now Booker speaks up. “I don’t understand how that could be right. Look at my X Y model. My R^2 is just as big as yours (and my coefficient is bigger).”

```
Booker_model <- XYdata |> model_train(X ~ Y)
Booker_model |> conf_interval()
```

Those with previous exposure to statistics methods might be inclined to say that the “p-value is small.” This is equivalent to saying that the confidence interval is far from zero. In general, as Lesson 29 discusses, it’s preferable to talk about confidence intervals rather than p-values.

term	.lwr	.coef	.upr
(Intercept)	-0.07246	-0.01026	0.05193
Y	1.947	2.01	2.073

```
Booker_model |> R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
1000	1	0.7966	3909	0.7964	0	1	998

Neither Booker's nor Ava's models can resolve the dispute between them. Data can't speak for themselves about the direction of influence. Model-building methods (with a large enough sample size) are helpful in showing whether there is a connection. For instance, either Booker's or Ava's results refute Cleo's hypothesis that there is no connection between X and Y. But models, on their own, are powerless to show the direction of influence.

For more than a century, many statisticians did not carry the issue beyond the simple $Y \leftrightarrow X$ example. It became dogma that the only way to establish causation is to experiment, that is, for the researcher to intervene in the system to sever causal influences. (See Lesson 26.) You will still see this statement in statistical textbooks, and news reports will endorse it by identifying "**Random controlled trials**" as the "Gold Standard" of causal relationships.

Although $Y \leftrightarrow X$ systems provide no fulcrum by which to lever out the truth about the direction of influence, richer systems sometimes present an opportunity to resolve causal disputes with data. The choice of covariates via DAGs provides the necessary key.

See [this article](#) in the prestigious British journal *The Lancet* to appreciate the history and irony of "gold standard."

Causal nihilism and smoking

Often, but not always, our interest in studying data is to reveal or exploit the causal connections between variables. Understanding causality is essential, for instance, if we are planning to intervene in the world and want to anticipate the consequences. Interventions are things like “increase the dose of medicine,” “stop smoking!”, “lower the budget,” “add more cargo to a plane (which will increase fuel consumption and reduce the range).”

Historically, mainstream statisticians were hostile to using data to explore causal relationships. (The one exception was **experiment**, which gathers data from an actual intervention in the world. See Lesson 26.) Statistics teachers encouraged students to use phrases like “associated with” or “correlated with” and reminded them that “correlation is not causation.”

Regrettably, this attitude made statistics irrelevant to the many situations where intervention is the core concern and experiment was not feasible. A tragic episode of this sort likely caused millions of unnecessary deaths. Starting in the 1940s, doctors and epidemiologists saw evidence that smoking causes lung cancer. In stepped the most famous statistician of the age, Ronald Fisher, to insist that the statement should be, “smoking is associated with lung cancer.” He speculated that smoking and lung cancer might have a common cause, perhaps genetic. Fisher argued that establishing causation requires running an experiment where people are randomly assigned to smoke or not smoke and then observed for decades to see if they developed lung cancer. Such an experiment is unfeasible and unethical, to say nothing of the need to wait decades to get a result.

Fortunately, around 1960, a researcher at the US National Institutes of Health, Jerome Cornfield, was able to show mathematically that the strength of the association between smoking and cancer ruled out any genetic mechanism. Cornfield’s work was a key step in developing a new area in statistics: “**causal inference**.”

Causal inference is not about proving that one thing

causes another but about formal ways to say something about how the world works that can be used, along with data, to make responsible conclusions about causal relationships.

25 Confounding

*“Economics’s reputation for dismality is a bad rap. Economics is as exciting as any science can be: the world is our lab, and the many diverse people in it are our subjects. The excitement in our work comes from the opportunity to learn about cause and effect in human affairs.”—Joshua Angrist and Jorn-Steffen Pischke (2015), *Mastering Metrics: The path from cause to effect**

Many people are concerned that the chemicals used by lawn-greening companies are a source of cancer or other illnesses. Imagine designing a study that could confirm or refute this concern. The study would sample households, some with a history of using lawn-greening chemicals and others who have never used them. The question for the study designers: What variables to record?

An obvious answer: record both chemical use and a measure of health outcome, say whether anyone in that household has developed cancer in the last five years. For simplicity in the presentation, we will suppose that the two possible levels of grass treatment are “organic” or “chemicals.” As for illness, the levels will be “cancer” or “not.”

Here are two simple DAG theories:

$$\text{illness} \leftarrow \text{grass treatment} \quad \text{or} \quad \text{illness} \rightarrow \text{grass treatment}$$

The DAG on the left expresses the belief among people who think chemical grass treatment might cause cancer. But belief is not necessarily reality, so we should consider alternatives. If only two variables exist, the right-hand DAG is the only alternative.

Section 24.3 demonstrated that it is not possible to distinguish between $Y \leftarrow X$ and $X \rightarrow Y$ purely by modeling data. Here, however, we are constructing theories. We can use the theory to guide how the data is collected. For example, one way to avoid the possibility of $\text{illness} \rightarrow \text{grass treatment}$ is to include

only households where cancer (if any) started *after* the grass treatment. Note that we are not ignoring the right-hand DAG; we are using the study design to disqualify it.

The statistical thinker knows that covariates are important. But which covariates? Appropriate selection of covariates requires knowing a lot about the “domain,” that is, how things connect in the real world. Such knowledge helps in thinking about the bigger picture and, in particular, possible covariates that connect plausibly to the response variable and the primary explanatory variable, grass treatment.

For now, suppose that the study designers have not yet become statistical thinkers and have rushed out to gather data on illness and grass treatment. Here are a few rows from the data (which we have simulated for this example):

grass	illness
organic	not
chemicals	not
chemicals	not
chemicals	not
organic	not
chemicals	cancer
organic	not

Analyzing such data is straightforward. First, check the overall cancer rate:

```
# overall cancer rate
Cancer_data |>
  mutate(illness = zero_one(illness, one="cancer")) |>
  model_train(illness ~ 1, family = "lm") |>
  conf_interval()
```

We are using linear regression where the intercept for `illness ~ 1` equals the proportion of specimens where the illness value is one.

term	.lwr	.coef	.upr
(Intercept)	0.01612	0.026	0.03588

In these data, 2.6% of the sampled households had cancer in the last five years. How does the grass treatment affect that rate? First, train a model...

```
mod <- Cancer_data |>
  mutate(illness = zero_one(illness, one="cancer")) |>
  model_train(illness ~ grass, family = "lm")
mod |> model_eval(skeleton = TRUE)
```

grass	.lwr	.output	.upr
organic	-0.277	0.03506	0.3472
chemicals	-0.2998	0.01247	0.3247

For households whose lawn treatment is “organic,” the risk of cancer is higher by 2.3 percentage points compared to households that treat their grass with chemicals. We were expecting the reverse, but the data seemingly disagree. On the other hand, there is sampling variability to take into account. Look at the confidence intervals:

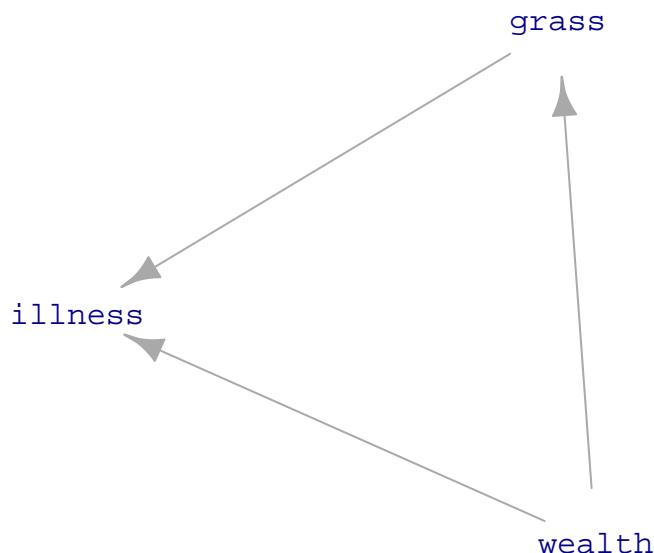
```
mod |> conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-0.003103	0.01247	0.02804
grassorganic	0.002469	0.02259	0.04271

The confidence interval on `grassorganic` does not include zero, but it comes close. So, might the chemical treatment of grass be protective against cancer? Not willing to accept what their data tell them, the study designers finally do what they should have from the start: think about covariates.

This is not an endorsement of the “keep searching until you find what you expected” research style. We will return to the negative consequences for the reliability of results from adopting this style in Lesson 29.

One theory—just a theory—is this: Green grass is not a necessity, so the households who treat their lawn with chemicals tend to have money to spare. Wealthier people also tend to have better health, partly because of better access to health care. Another factor is that wealthier people can live in less polluted neighborhoods and are less likely to work in dangerous conditions, such as exposure to toxic chemicals. Such a link between wealth and illness points to a DAG hypothesis where “wealth” influences how the household’s **grass** is treated and **wealth** similarly influences the risk of developing **cancer**. Like this:



A description of this causality structure is, “The effect of grass treatment on illness is **confounded** by wealth.” The [Oxford Languages](#) dictionary offers two definitions of “confound.”

1. *Cause surprise or confusion in someone, especially by acting against their expectations.*
2. *Mix up something with something else so that the individual elements become difficult to distinguish.*

This second definition carries the statistical meaning of “confound.”

The first definition seems relevant to our story since the protagonist expected that chemical use would be associated with

higher cancer rates and was surprised to find otherwise. Nevertheless, the statistical thinker does not throw up her hands when dealing with mixed-up causal factors. Instead, she uses modeling techniques to untangle the influences of various factors.

Using covariates in models is one such technique. Our wised-up study designers go back to collect a covariate representing household wealth. Here is a glimpse at the updated data.

wealth	grass	illness
1.428	organic	not
0.06286	chemicals	not
0.4383	chemicals	not
0.6084	chemicals	not
0.8034	organic	not
-0.9367	organic	not
0.6664	organic	not
-1.245	organic	not
-1.319	chemicals	cancer
-1.616	organic	not

Having measured `wealth`, we can use it as a covariate in the model of `illness`. We use *logistic regression* for this model with multiple explanatory variables to avoid the out-of-bounds problem introduced in Section 21.3.

```
Cancer_data |>
  mutate(illness = zero_one(illness, one="cancer")) |>
  model_train(illness ~ grass + wealth) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-5.95	-4.75	-3.82
grassorganic	-2.16	-1.02	0.222
wealth	-2.92	-2.25	-1.67

With `wealth` as a covariate, the model shows that (all other things being equal) “organic” lawn treatment reduces cancer

risk. However, we do not see this directly from the `grass` and `illness` variables because all other things are not equal: wealthier people are more likely to use chemical lawn treatment. (Remember, this is **simulated data**. Do not conclude from this example anything about the safety of the chemicals used for lawn greening.)

i Example: The flu vaccine

As you know, people are encouraged to get vaccinated before flu season. This recommendation is particularly emphasized for older adults, say, 60 and over.

In 2012, *The Lancet*, a leading medical journal, published a [systematic examination and comparison of many previous studies](#). The *Lancet* article describes a hypothesis that existing flu vaccines may not be as effective as originally found from modeling mortality as a function of vaccination.

A series of observational studies undertaken between 1980 and 2001 attempted to estimate the effect of seasonal influenza vaccine on rates of hospital admission and mortality in [adults 65 and older]. Reduction in all-cause mortality after vaccination in these studies ranged from 27% to 75%. In 2005, these results were questioned after reports that increasing vaccination in people aged 65 years or older did not result in a significant decline in mortality. Five different research groups in three countries have shown that these early observational studies had substantially overestimated the mortality benefits in this age group because of unrecognized confounding. This error has been attributed to a healthy vaccine recipient effect: reasonably healthy older adults are more likely to be vaccinated, and a small group of frail, undervaccinated elderly people contribute disproportionately to deaths, including during periods when influenza activity is low or absent.

Such a study of earlier studies is called a *meta-analysis*.

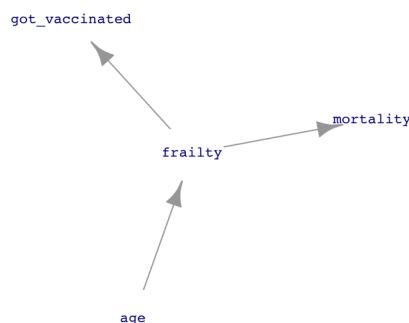


Figure 76: A DAG diagramming the “healthy vaccine recipient” effect

Figure 76 presents a network of causal influences that could shape the “healthy vaccine recipient.” People are more likely to become frail as they get older. Frail people are *less* likely to get vaccinated but more likely to die in the next few months. The result is that vaccination is associated with reduced mortality, even if there is no direct link between vaccination and mortality.

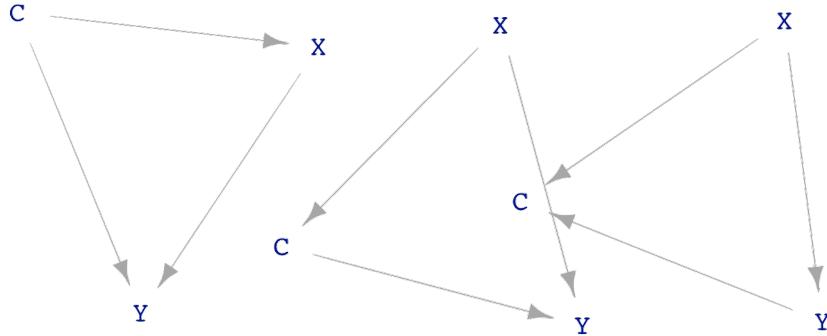
25.1 Block that path!

Let us look more generally at the possible causal connections among three variables: X, Y, and C. We will stipulate that X points causally toward Y and that C is a possible covariate. Like all DAGs, there cannot be a cycle of causation. These conditions leave three distinct DAGs that do not have a cycle, as shown in Figure 77.

C plays a different role in each of the three dags. In sub-figure (a), C causes both X and Y. In (b), part of the way that X influences Y is *through* C. We say, in this case, “C is a mechanism by which X causes Y. In sub-figure (c), C does not cause either X or Y. Instead, C is a consequence of both X and Y.

Chemists often think about complex molecules by focusing on sub-modules, e.g. an alcohol, an ester, a carbon ring. Similarly, there are some basic, simple sub-structures that often appear

In any given real-world context, good practice calls for considering each possible DAG structure and concocting a story behind it. Such stories will sometimes be implausible, but there can also be surprises that give the modeler new insight.



(a) C is a confounder. (b) C is a mechanism. (c) C is a consequence.

Figure 77: Three different DAGs connecting X, Y, and C.

in DAGs. Figure 78 shows four such structures found in Figure 77.

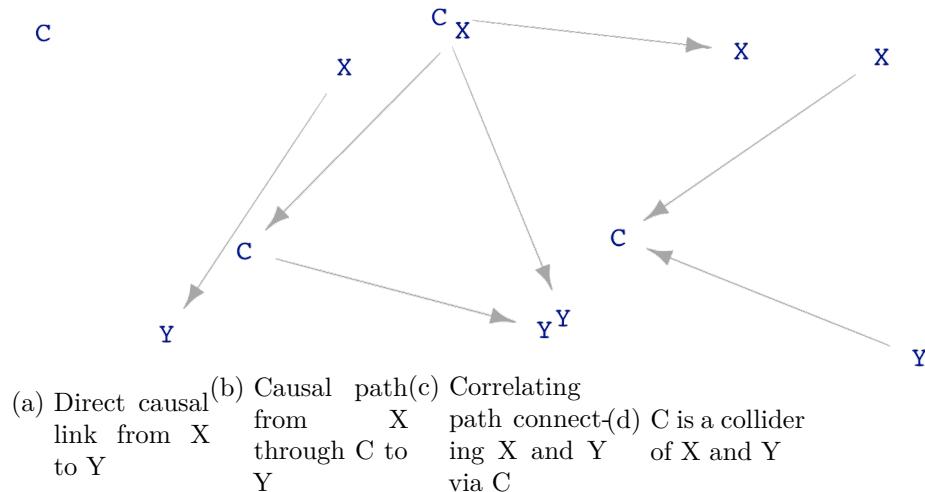


Figure 78: Sub-structures seen in Figure 77.

- A “**direct causal link**” between X and Y. There are no intermediate nodes.
- A “**causal path**” from X to C and on to Y. A causal path is one where, starting at the originating node, flow along the arrows can get to the terminal node, passing through all intermediate nodes.

- A “**correlating path**” from Y through C to X. Correlating paths are distinct from causal paths because, in a correlating path, there is no way to get from one end to the other by following the flows.
- A “**common consequence**,” also known as a “**collider**”. Both X and Y are causes of C and there is no causal flow between X and Y.

Look back to Figure 77(a), where `wealth` is a confounder. A confounder is always an intermediate node in a *correlating path*.

Including a covariate either blocks or opens the pathway on which that covariate lies. Which it will be depends on the kind of pathway. A causal path, as in Figure 78(b), is blocked by including the covariate. Otherwise, it is open. A correlating path (Figure 78(c)) is similar: the path is open unless the covariate is included in the model. A colliding path, as in Figure 78(d), is blocked *unless* the covariate is included—the opposite of a causal path.

i Where do the blocking rules come from?

To understand these blocking rules, we need to move beyond the metaphors of ants and flows. Two variables are correlated if a change in one is reflected by a change in the other. For instance, if a specimen with large X tends also to have large Y, then across many specimens there will be a correlation between X and Y. There is a correlation as well if specimens with large X tend to have *small* Y. It's only when changes in X are not reflected in Y, that is, specimens with large X can have either small, middle, large values of Y, that there will *not* be a correlation. We will start with the situation where C is not used as a covariate: the model $y \sim x$.

Perhaps the easiest case is the *correlating path* (Figure 78(c)). A change in variable C will be propagated to *both* X and Y. For instance, suppose an increase in C causes an increase in X and separately causes an increase in Y. Then X and C will tend to rise and fall together from

For simplicity, we'll walk through those situations where specimens with large X tend to have large Y. The other case, specimens with large X having small Y, is much the same. Just change “large” to “small” when it comes to Y.

specimen to specimen. This is a correlation; the path $X \leftarrow C \rightarrow$ is not blocked. (We say, “an increase in C *causes* an increase in X ” because there is a direct causal link from C to X .)

For the *causal path* (Figure 78(b)), we look to changes in X . Suppose an increased X causes an increased C which, in turn, causes an increase in Y . The result is that specimens with large X tend to have large Y : a correlation and therefore an open causal path $X \rightarrow C \rightarrow Y$.

For a *common consequence** (Figure 78(c)) the situation is different. C does not cause either X or Y . In specimens with large X , Y values can be small, medium, or large. No correlation; the path $X \rightarrow C \leftarrow Y$ is blocked.

Now turn to the situation where C is included in the model as a covariate: $y \sim x + c$. As described in Lesson Chapter 12, to include C as a covariate is, through mathematical means, to look at the relationship between Y and X *as if C were held constant*. That’s somewhat abstract, so let’s put it in more concrete terms. We use modeling and adjustment because C is not in fact constant; we use the mathematical tools to make it seem constant. But we wouldn’t need the math tools if we could collect a very large amount of data, then select only those specimens for analysis that have *the same value of C*. For these specimens, C would in fact be constant; they all have the same value of C .

For the *correlating path*, because C is the same for all of the selected specimens, neither X nor Y vary along with C . Why? There’s no variation in C ! Any increase in X from one specimen to another would be induced by other factors or just random noise. Similarly for Y . So, when C is held constant, the up-or-down movements of X and Y are unrelated; there’s no correlation between X and Y . the $X \leftarrow C \rightarrow Y$ path is blocked.

For the *causal path* $X \rightarrow C \rightarrow Y$, because C has the same value for all specimens, any change in X is *not reflected* in C . (Why? Because there is no variation in C ! We’ve picked only specimens with the same C value.) Likewise, C and Y will not be correlated; they can’t be because there

is no variation in C even though there is variation in Y. Consequently, among the set of selected specimens where C is held constant, there is no evidence for synchronous increases and decreases in X and Y. The path is blocked. Look now at the *common consequence* (Figure 78(c)). We have selected only specimens with the same value of C. Consider the back-story for each specimen in our selected set. How did C come to be the value that it is in order to make it into our selection? If for the given specimen X was large, then Y must have been small to bring C to the value needed to get into the selected set of specimens. Or, *vice versa*, if X was small then Y must have been large. When we look across all the specimens in the selected set, we will see large X associated with small Y: a correlation. Holding C constant *unblocks* the pathway that would otherwise have been blocked.

Often, covariates are selected to block all paths except the direct link between the explanatory and response variable. This means *do* include the covariate if it is on a correlating path and *do not* include it if the covariate is at the collision point.

As for a causal path, the choice depends on what is to be studied. Consider the DAG drawn in Figure 77(b), reproduced here for convenience:

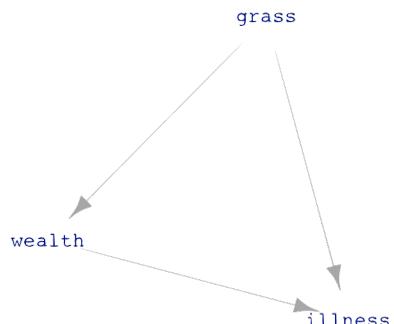
`grass` influences `illness` through two distinct paths:

- i. the direct link from `grass` to `illness`.
- ii. the causal pathway from `grass` through `wealth` to `illness`.

Admittedly, it is far-fetched that choosing to green the grass makes a household wealthier. However, for this example, focus on the topology of the DAG and not the unlikeliness of this specific causal scenario.

There is no way to block a direct link from an explanatory variable to a response. If there were a reason to do this, the modeler probably selected the wrong explanatory variable.

But there is a genuine choice to be made about whether to block pathway (ii). If the interest is the purely biochemical



link between grass-greening chemicals and illness, then block pathway (ii). However, if the interest is in the *total* effect of **grass** and **illness**, including both biochemistry and the sociological reasons why **wealth** influences **illness**, then leave the pathway open.

25.2 Don't ignore covariates!

In 1999, a **paper** by four pediatric ophthalmologists in *Nature*, perhaps the most prestigious scientific journal in the world, claimed that children sleeping with a night light were more likely to develop nearsightedness. Their recommendation: “[I]t seems prudent that infants and young children sleep at night without artificial lighting in the bedroom, while the present findings are evaluated more comprehensively.”

This recommendation is based on the idea that there is a causal link between “artificial lighting in the bedroom” and nearsightedness. The paper acknowledged that the research “does not establish a causal link” but then went on to imply such a link:

“[T]he statistical strength of the association of night-time light exposure and childhood myopia does suggest that the absence of a daily period of darkness during early childhood is a potential precipitating factor in the development of myopia.”

“Potential precipitating factor” sounds a lot like “cause.”

The paper did not discuss any possible covariates. An obvious one is the eyesight of the parents. Indeed, ten months after the original paper, *Nature* printed a **response**:

“Families with two myopic parents, however, reported the use of ambient lighting at night significantly more than those with zero or one myopic parent. This could be related either to their own poor visual acuity, necessitating lighting to see the child more easily at night, or to the higher socio-economic level of myopic parents, who use more child-monitoring devices. Myopia in

children was associated with parental myopia, as reported previously.”

Always consider possible alternative causal paths when claiming a direct causal link. For us, this means thinking about that covariates there might be and plausible ways that they are connected. Just because a relevant covariate wasn’t measured doesn’t mean it isn’t important! Think about covariates *before* designing a study and measure those that can be measured. When an essential blocking covariate wasn’t measured, don’t fool yourself or others into thinking that your results are definitive.

26 Experiment and random assignment

In its everyday meaning, the word “experiment” is similar to the word “experience.” As a verb, to experiment means to “try out new concepts or ways of doing things.” As a noun, an experiment is a “course of action tentatively adopted without being sure of the outcome.” Both quotes are from the [Oxford Languages](#), which provides examples of each: “the designers experimented with new ideas in lighting” or “the farm is an ongoing experiment in sustainable living.”

From movies and other experiences, people associate experiments with science. Indeed, one of the dictionary definitions of “experiment” is “a scientific procedure undertaken to make a discovery, test a hypothesis, or demonstrate a known fact.”

Almost all the knowledge needed to perform a scientific experiment relates to the science itself: what reagents to use, how to measure the concentration of a neurotransmitter, how to administer a drug safely, and so on. This is why people who carry out scientific procedures are trained primarily in their area of science.

i Example: Malaria and bed nets

In many parts of the world, malaria is a major cause of disability and death. Economists who study ways to relieve poverty have a simple, plausible theory: reducing the effect of illnesses such as malaria will impact poverty rates since healthier people are more productive. Reduced uncertainty about illness can help them amass capital to invest to increase production further.

There are many possible ways to reduce the burden of malaria. Vaccination (although effective vaccines have been hard to develop), insect control using pesticides (which can cause environmental problems), etc. One simple intervention is using **bed nets**, screens deployed at night by draping over the bed and its occupant. Still, there are reasons why distributing bed nets may not be effective; people might use them incorrectly or for other purposes such as fishing. People might not be able to af-

ford them, but giving them away might signal that they have no value.

To find out, try it: experiment. For instance, run a trial program where nets are given away to everyone in an area and observe whether and to what extent rates of malarial illness go down.

Such a trial is undoubtedly an experiment. However, it may not be the best way to get meaningful information.

26.1 Replication

To understand some of the contribution that statistical thinking can make to experiment, recall our earlier definition:

Statistic thinking is the explanation/description of variation in the context of what remains unexplained/undescribed.

A key concept that statistical thinking brings to experiment is the idea of **variation**. Simply put, a good experiment should involve some variation. The simplest way to create variation is to repeat each experimental trial multiple times. This is called “**replication**.”

26.2 Example: Replicated bed net trials

One way to improve the simple experiment bed net described above is to conduct many trials. One reason is that the results from any single trial might be shaped by accidental or particular circumstances: the weather in the trial area was less favorable to mosquito reproduction; another government agency decided to help out by spraying pesticides broadly, and so on. Setting up trials in different areas can help to balance out these influences.

Replicated trials also allow us to estimate the size of the variability caused by accidental or particular factors. To illustrate, suppose a single trial is done. Result: the rate of malarial illness goes down by five percentage points. What can we conclude?

The result is promising, but we can't rule out that it is due to accidental factors other than bed nets. Why not? Because we have no idea how much unexplained variation is in play.

Table ?? shows data from ten imagined trials on the effect of bed nets; one for each of ten different sites. (Reduction by a negative number, like reduction by -1, is an *increase*.) The mean reduction is three percentage points, but this number is not much use unless we can put it in the context of sampling variation. Conducting multiple trials introduces *observed* variation in results and thereby gives us a handle on the amount of sampling variation.

Using the regression framework makes estimating the amount of sampling variation easy. The mean reduction corresponds to the coefficient from the model `reduction ~ 1`.

```
Bed_net_data |>
  model_train(reduction ~ 1) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	1	3	5

The observed three percentage point mean reduction in malaria incidence does stand out from the noise: the confidence interval does not include zero. In these (imagined) data, we have confidence that we have seen a signal.

26.3 Control

However, there is still a problem with the design of the imagined bed-net experiment. What if the year the experiment was done was arid, reducing the mosquito population and, with it, the malaria infection rate? Then we don't know whether the observed 3-point reduction is due to the weather or the bed nets, or even something else, e.g., better nutrition due to a drop in international prices for rice.

Table 108: Imagined bed-net data

site	reduction
A	5
B	8
C	2
D	-1
E	3
F	1
G	4
H	0
I	2
J	6

We need to measure what the change in malarial infection would have been without the bed-net intervention. Care needs to be taken here. If the trial sites were rural, comparing their malarial rates to urban areas as controls is inappropriate. We want to compare the trial sites with non-trial sites where the intervention was not carried out, the so-called “**control**” sites. The `With_controls` data frame imagines what data might look like if in half the sites no bed-net program was involved.

The proper regression model for the `With_controls` data is `reduction ~ nets`:

```
With_controls |>
  model_train(reduction ~ nets) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-2.2	0.2	2.6
netstreatment	0.058	3.4	6.7

The effect of the bed nets is summarized by the `netstreatment` coefficient, which compares the `reduction` between the `treatment` and `control` groups. In this new (imagined) data frame, the confidence interval on `netstreatment` touches close to zero; the signal is barely discernible from the noise.

The reader might wonder why, in moving to the controlled design, the ten sites were not all treated with nets and another ten or so sites selected to use as the control. The control sites could be chosen as villages near the bed net villages.

One reason is pragmatic: the more extensive project would require more effort and money. The more extensive project might be worthwhile; larger n would presumably narrow the confidence interval. Another reason, to be expanded on in the next section, is that the treatment and control sites should be as similar as possible. This can be surprisingly hard to achieve. Other factors, such as the enthusiasm or skepticism of the town leaders toward public-health interventions might be behind the choice of the original sites for the bed-net program.

Table 110: `With_controls`,
imagined data from
a new study where
five sites were used
as controls.

site	reduction	nets
K	2	control
L	8	treatment
M	4	treatment
N	1	treatment
O	-1	control
P	-2	control
Q	0	control
R	2	treatment
S	3	treatment
T	2	control

The control sites might be towns that turned down the original offer of the bed-net program and, accordingly, have different attitudes toward public health.

26.4 Example: Testing the Salk polio vaccine

Today, most children are vaccinated against polio, though a smaller fraction than in previous years. This might be because symptomatic polio is rare, lessening the perceived urgency of protecting against it. Partly, the reduction reflects the growth in the “anti-vax” movement, which became especially notable with the advent of COVID-19.

The first US polio epidemic occurred in 1916, just two years before the COVID-like “Spanish flu” pandemic. Up through the early 1950s, polio injured or killed hundreds of thousands of people, particularly children. Anxiety about the disease was similar to that seen in the first year of the COVID-19 pandemic.

There were many attempts to develop a vaccine against polio. Jonas Salk created the first promising vaccine, the promise being based on laboratory tests. To establish the safety and effectiveness of the Salk vaccine, it needed to be tried in the field, with people. Two organizations, the US Public Health Service and the National Foundation for Infantile Paralysis, got together to organize a clinical field trial which, all told, involved two-million students in grades 1 through 3.

The two studies involved both a treatment and a control group. In some school districts, students in grades 1 and 3 were held as controls. The treatment group was students in grade 2 whose parents gave consent. We will call this “Study 1.” In other school districts, the study design was different: the parents of all students in all three grades were asked for consent. The students with parental consent were then randomly split into two groups: a treatment and a control. Call this “Study 2.”

The Study 2 design might seem inefficient; it reduced the number of children receiving the vaccine because half of the children with parental consent were left unvaccinated. On the other hand, it might be that children from families who consent to be

“Spanish” is in quotes because Spain was not the source of the pandemic.

given a vaccine are different in a systematic way from children whose families refuse, just as today's anti-vax families might be different from "pro-vax" families.

As reported in Freedman (1998)¹, the different risks of symptomatic polio between children from consenting versus refusing families became evident in the study. `?@tbl-polio1` shows a difference between the treatment and "no consent" groups: 25 per 100,000 in the treatment group got polio versus 44 per 100,000 in the "no consent" group. But we can't untangle the effects of the vaccine itself from the effects associated with different families' decisions. Confounding is a possibility.

`?@tbl-polio2` shows the results from the school districts that used half the consent group as controls. The difference between treatment and control groups is evident: a reduction from 71 cases per 100,000 children to 28 cases per 100,000. The no-consent children had a rate between the two, 46 per 100,000. Since both the "control" and "no consent" groups did not get the vaccine, one might expect those rates to be similar. That they are not demonstrates the confounding between consent and vaccine; the "no-consent" children are systematically different from those children whose parents gave consent.

The results from Study 2 demonstrate that the estimated effect of the vaccine from Study 1 understated the biological link between vaccination and reduction of polio risk. The confounding between consent and vaccine in Study 1 obscured the positive effect of the vaccine.

26.5 Random assignment

The example of the Salk vaccine trial is a chastening reminder that care must be taken when assigning `treatment` or `control` to the units in an experiment. Without such care, confounding enters into the picture. Merely the possibility of confounding damages the experiment's result; it invites skepticism and doubt.

It is illuminating to look at the vaccine trial as a DAG. The essential situation is diagrammed in Figure 79. The `socio_economic` node represents the idea that socio-economic

¹ D. Freedman, R Pisani, R Purves, *Statistics 3/e*, p.6

Table 112: Results from polio Study 1

vaccine	size	rate
Treatment	225000	25
No consent	125000	44

Table 113: Results from polio Study 2

vaccine	size	rate
Treatment	200000	28
Control	200000	71
No consent	350000	46

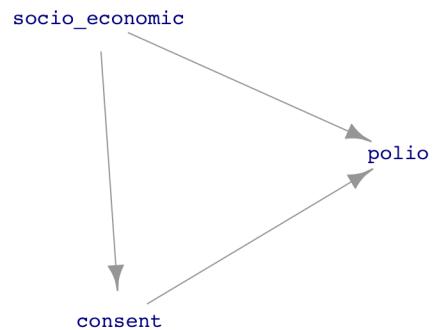


Figure 79: A simulation of the polio vaccine experiment.

status has an influence on susceptibility to symptomatic polio and also is a factor in shaping a family's decision about giving consent. (In contrast to the usual expectation that lower socio-economic status is associated higher risk of disease, with polio the opposite holds true. The explanation usually given is that children who are exposed to the polio virus as infants do not become sick but do gain immunity to later infection. People later in childhood and in adulthood are at risk of a severe, symptomatic response to exposure. Polio is transmitted mainly via a fecal-oral route. Conditions favoring this route are more common among those of low socio-economic status. Consequently, infants of well-to-do families are less exposed to the virus and do not develop immunity. When they are eventually exposed to polio as children or adults, the well-to-do are at greater risk of developing disease.)

The DAG in Figure 79 has two pathways between **treatment** and **polio** that can produce confounding:

- $\text{treatment} \leftarrow \text{consent} \rightarrow \text{polio}$
- $\text{treatment} \leftarrow \text{consent} \leftarrow \text{socio.economic} \rightarrow \text{polio}$

The approach emphasized in Lesson 25 to avoid such confounding is blocking the relevant pathways. Both can be blocked by including **consent** as a covariate. However, in Study 1, assignment to vaccine was purely a matter of consent; **consent** and **treatment** are essentially the same variable. Figure 80 shows the corresponding DAG, where **consent** and **treatment** are merged into a single variable. Holding **consent** constant deprives the system of the explanatory variable and still introduces confounding through **socio_economic**.

In Study 2, all the children participating had parents give consent. This means that **consent** is not a variable; it doesn't vary! The corresponding DAG, without **consent** as a factor, is drawn in Figure 81. This Study 2 DAG is unfolded; there are no confounding pathways! Thus, the model $\text{polio} \sim \text{treatment}$ is appropriate.

The assignment to treatment or control in Figure 81 is made by the people running the study. Although the DAG doesn't show any inputs to **assignment**, the involvement of people in making the assignment opens up a possibility that other factors, such as

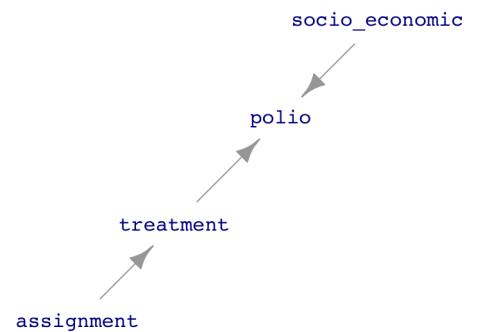


Figure 80: The DAG when **consent** \equiv **vaccine**.

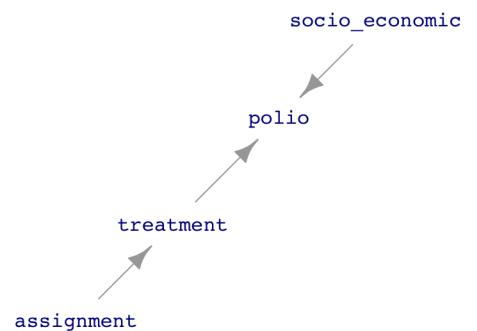


Figure 81: The Study 2 DAG.

socio-economic status, might have influenced their assignment of treatment or control. To guard against this, or even skepticism raised by the possibility, experimentalists have developed a simple safeguard: “**random assignment**.” In random assignment, assignment is made by a computer generating random numbers. Nobody believes that the computer algorithm is influenced by socio-economic status or any other factor that might be connected to polio in any way.

27 Hypothetical thinking

Many people presume that the logical process of drawing conclusions from data is **inductive reasoning**, unlike the **deductive reasoning** involved in mathematical proof. **Inductive reasoning** involves constructing general statements based on observed *facts*. A famous historical example involves Robert Boyle (1627-1691), whose name will be familiar to chemistry students.

Boyle's law, from 1662, says that the pressure of a given amount of a gas is inversely proportional to its volume at a constant temperature.

We have access to Boyle's lab notebooks and publications. Some of his experimental equipment is pictured in Figure 82. Here are the data from his 1662 treatise *A Defence Of the Doctrine touching the Spring and Weight Of the Air*.

A Table of the Condensation of the Air.					
A	B	C	D	E	
48	12	00	29 $\frac{1}{2}$	29 $\frac{1}{2}$	
46	11 $\frac{1}{2}$	01 $\frac{1}{2}$	30 $\frac{1}{2}$	30 $\frac{1}{2}$	
44	11	02 $\frac{1}{2}$	31 $\frac{1}{2}$	31 $\frac{1}{2}$	
42	10 $\frac{1}{2}$	04 $\frac{1}{2}$	33 $\frac{1}{2}$	33 $\frac{1}{2}$	
40	10	06 $\frac{1}{2}$	35 $\frac{1}{2}$	35 $\frac{1}{2}$	
38	9 $\frac{1}{2}$	07 $\frac{1}{2}$	37-	36 $\frac{1}{2}$	
36	9	10 $\frac{1}{2}$	39 $\frac{1}{2}$	38 $\frac{1}{2}$	
34	8 $\frac{1}{2}$	12 $\frac{1}{2}$	41 $\frac{1}{2}$	41 $\frac{1}{2}$	
32	8	15 $\frac{1}{2}$	44 $\frac{1}{2}$	43 $\frac{1}{2}$	
30	7 $\frac{1}{2}$	17 $\frac{1}{2}$	47 $\frac{1}{2}$	46 $\frac{1}{2}$	
28	7	21 $\frac{1}{2}$	50 $\frac{1}{2}$	50 $\frac{1}{2}$	
26	6 $\frac{1}{2}$	25 $\frac{1}{2}$	54 $\frac{1}{2}$	53 $\frac{1}{2}$	
24	6	29 $\frac{1}{2}$	58 $\frac{1}{2}$	58 $\frac{1}{2}$	
23	5 $\frac{1}{2}$	32 $\frac{1}{2}$	61 $\frac{1}{2}$	60 $\frac{1}{2}$	
22	5 $\frac{1}{2}$	34 $\frac{1}{2}$	64 $\frac{1}{2}$	63 $\frac{1}{2}$	
21	5 $\frac{1}{2}$	37 $\frac{1}{2}$	67 $\frac{1}{2}$	66 $\frac{1}{2}$	
20	5	41 $\frac{1}{2}$	70 $\frac{1}{2}$	70 $\frac{1}{2}$	
19	4 $\frac{1}{2}$	45 $\frac{1}{2}$	74 $\frac{1}{2}$	73 $\frac{1}{2}$	
18	4 $\frac{1}{2}$	48 $\frac{1}{2}$	77 $\frac{1}{2}$	77 $\frac{1}{2}$	
17	4 $\frac{1}{2}$	53 $\frac{1}{2}$	82 $\frac{1}{2}$	82 $\frac{1}{2}$	
16	4	58 $\frac{1}{2}$	87 $\frac{1}{2}$	87 $\frac{1}{2}$	
15	3 $\frac{1}{2}$	63 $\frac{1}{2}$	93 $\frac{1}{2}$	93 $\frac{1}{2}$	
14	3 $\frac{1}{2}$	71 $\frac{1}{2}$	100 $\frac{1}{2}$	99 $\frac{1}{2}$	
13	3 $\frac{1}{2}$	78 $\frac{1}{2}$	107 $\frac{1}{2}$	107 $\frac{1}{2}$	
12	3	88 $\frac{1}{2}$	117 $\frac{1}{2}$	116 $\frac{1}{2}$	

AA. The number of equal spaces in the shorter leg, that contained the same parcel of Air diversly extended.
 B. The height of the Mercurial Cylinder in the longer leg, that comprehend'd the Air into those dimensions.
 C. The height of a Mercurial Cylinder that counterbalanc'd the pressure of the Atmosphere.
 D. The Aggregate of the two last Columns B and C, exhibiting the pressure sustainted by the included Air.
 E. What that pressure shou'd be according to the Hypothesis, that supposes the pressures and expansions to be in reciprocal proportion.

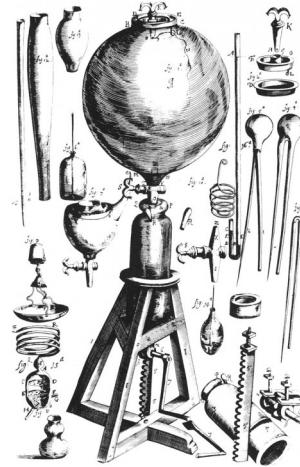


Figure 82: Robert Boyle's air pump and associated equipment, 1661.
 Source

Based on this data, Boyle formulated his law. Boyle's Law still appears in textbooks, even though successive generations of scientists have tweaked it to be more precise (replace "amount" with "mass") or to account for contrary observations (replace "gas" with "ideal gas.")

Boyle is regarded as among the founders of modern scientific method, one description of which, arising in the mid-20th century, is called the **hypothetico-deductive model**. The pro-

cess of the hypothetical-deductive model consists of formulating a hypothesis to describe the world, deducing consequences from these hypothesis, and carrying out experiments to look for those consequences. If the consequences are experimentally observed, the experiment corroborates the hypothesis. If not, the experiment refutes the hypothesis. In the progress of science, refuted hypotheses are replaced with alternatives that are compatible with the assembled experimental data. And the cycle of experiment, corroboration or refutation, and hypothesis generation begins again.

Notice that the word **inductive** does not appear in the name of the hypothetical-deductive model. That model does not attempt to explain where the refined hypotheses come from. If induction is the process of generating hypotheses from data, the hypothetical-deductive model does not involve induction.

Similarly, most statistical method is not about induction because it is not about generating hypotheses. All statistical quantities in these Lessons—coefficients, R^2 , effect sizes, and so on—are *deduced* from data. That’s why we can program a computer to calculate for us: creativity is not required.

The creative elements in statistical thinking are in the decisions about what systems to study, how to collect data from the system, and specification of models, e.g. `time ~ distance * climb`. Every set of such decisions is a hypothesis: “This approach will be helpful for my purposes.” If the purpose involves intervening in a system, the DAG methodology can guide in selecting covariates. If the purpose is prediction, other criteria for selecting covariates are appropriate.

This part of the Lessons is about how to choose among a set of hypotheses on the basis of data. I call this “hypothetical thinking” because it is not deductive. Neither is it inductive; the hypothetical thinking methods don’t necessarily point toward the formation of hypotheses. Rather, hypothetical thinking, in my formulation, is about the logic of evaluating hypotheses based on data.

The next two Lessons deal with closely related methods of hypothetical thinking. Lesson 28 covers **Bayesian inference**,

a universally accepted and mathematically demonstrated approach to evaluating hypotheses which was introduced in the 1700s. However, starting about 1900 an important group of statistical pioneers rejected Bayesian methods, claiming that they were logically inadmissible for drawing completely objective conclusions from data. These statistical pioneers introduced new procedures for hypothetical reasoning called “**Null hypothesis testing**” (NHT), which we cover in Lesson 29.

NHT has been the dominant paradigm in applied statistics for many decades. It’s also widely accepted by statisticians that most people who use NHT do not understand it and often use it in ways that are invalid. You need to learn about NHT because it is dominant and in order to avoid its pitfalls. But you also need to learn about Bayes because it is indisputably the right approach in many contexts, particularly those that relate to decision-making. And, in my view, Bayes is a valuable perspective for learning what NHT is ... and what it isn’t.

27.1 Where do hypotheses come from?

“Dans les champs de l’observation, le hasard ne favorise que les esprits préparés.” (“In the field of observation, chance favors only the prepared mind.”)
- Louis Pasteur (1822-1895)

“Genius is one per cent inspiration, ninety-nine per cent perspiration.” - Thomas Edison (1847-1931)

It’s uncontroversial to say that hypotheses come from imagination, inspiration, creativity, dreaming, metaphor, and analogy. True though this may be, it’s helpful to have a more concrete model to draw on to describe the relationship between hypothesis and data.

Consider Robert Boyle and his hypothesis that gas pressure is inversely proportional to volume (at constant temperature). Boyle did not grow up in a vacuum. For instance, he would have been educated in geometry and familiar with the classic geometrical shapes: circle, ellipse, line, parabola, hyperbola, etc. In 1641, Boyle lived in Florence for a year, studying the work of the then-elderly Galileo. In *The Assayer* (1623), Galileo

famously wrote: ““Philosophy is written in this grand book, the universe ... It is written in the language of mathematics, and its characters are triangles, circles, and other geometric figures;....”

We can suppose that Boyle had these mathematical “characters” in mind when looking for a quantitative relationship in his data. Perhaps he went through the list of characters looking for a match with his observations. The hyperbola was the most likely and corresponds to the “inversely proportional” description in his Law.

The astronomer Johannes Kepler (1571-1660) similarly worked through a succession of possible geometrical models, matching them with data assembled by Tycho Brahe (1546-1601). An elliptical orbit was the best match. And Boyle’s lab assistant, Robert Hooke (1635-1703), would have had access to the same set of geometrical possibilities in framing his theory—called Hooke’s Law—that the relationship between the extension of a spring and the force exerted by the spring is one of direct proportionality. Hooke also proposed, before Isaac Newton, that the relationship between gravitational force and distance is an inverse-square proportion.

All of us have access to a repertoire or library of theoretical forms or patterns. This repertoire is bigger or smaller depending on our experience and education. For instance, in these Lessons you have seen a framework for randomness in the form of the various named noise models and their parameters. Another important framework in these Lessons is the scheme of regression modeling with its response variable and explanatory variables. Lesson 13.3 included a new pattern to add to your repertoire: interaction between variables.

Another important source for hypotheses is **analogy**. For instance, in the 1860s, James Clerk Maxwell (1831-1879) noted a close similarity between the mathematics of electricity and magnetism and the mathematics of propagation of waves in water or air. He offered the hypothesis that light is also a wave.

The “interaction” pattern is the same as the “law of mass action” in chemistry.

27.2 Hypotheses and deduction

Imagine your repertoire of patterns in book form: a listing of all the relationship patterns you have encountered in your life and education. Each of these patterns can be entertained hypothetically as an explanation of observations. The book provides a springboard to reasoning *inductively* from data to general pattern. You can work through the book systematically, assessing each possible pattern as a model for the data. The comparison can be done by intuition or calculation. For instance, you might graph the data and compare it against graphs of the various patterns.

Likelihood, introduced in Lesson 16 provides a particular form of calculation for comparison of hypothesis and data. Understanding likelihood involves comprehending several of the frameworks introduced earlier in these Lessons: considering data as a combination of signal and noise, modeling the signal using regression to find coefficients, various noise models and their parameters. The quantity used to do the comparison is **likelihood**: the relative probability of the data given the signal and noise models. To the extent that likelihood is high, the data corroborates the hypothesis.

This picture of *inductive* reasoning is based on the *deduction* (or calculation) of likelihood from a hypothesis and data. The mysterious part of induction—where does the hypothesis come from—has been reduced to a book of patterns. This idea of how we learn from data is consistent with a name often associated with the scientific method: the hypothetico-deductive model. The pattern book is the “hypothetico” part, likelihood is the deductive part.

Let’s look closer at the above statement: “To the extent that likelihood is high, the data corroborates the hypothesis.” When you encounter words like “high,” “big,” “small,” etc., an excellent mental habit is to ask, “Compared to what?” The number that results from a likelihood calculation does not come on a scale marked with the likelihood values of other successful or unsuccessful hypotheses. Instead, determining whether likelihood is high or low depends on comparing it to the likelihood calculated (on the same data) based on *other hypotheses*. Each

individual hypothesis generates a likelihood number, judging whether that number is high or low depends on comparing the likelihoods for multiple hypotheses.

In the following two Lessons, we will mostly focus on the comparison between *two* hypotheses, but the methods can be generalized to work for any number of hypotheses. In Bayesian reasoning, one calculates the *relative probability* of each of the hypotheses under consideration. Proponents of Null hypothesis testing (NHT), usually called “**frequentists**,” consider just a single hypothesis: the eponymous Null. Naturally, this rules out any comparison of the likelihood of different hypotheses. But NHT is nevertheless based on comparing likelihoods. The likelihoods compared in NHT relate to different data rather than different hypotheses. (It will be easier to understand this when we specify what “different data” means in NHT.)

Whatever the differences between the two primary schools of hypothetical thinking, frequentists and Bayesians, they both agree that likelihood is a valuable quantity to consider when drawing conclusions. So, in these Lessons, *hypothetical thinking* will center on the concept of likelihood.

Instructors who have taught hypothesis testing in a conventional framework might find this [blog post](#) informative.

27.3 Planets and hypotheses

In thinking about abstractions, it can be helpful to have a concrete mental representation. For *hypotheses*, the interplanetary travel of science fiction provides a good representation. In the science-fiction genre, each planet is a place where new things are going on: new life forms, new forces, new social organizations, and so on. Each hypothesis corresponds to a planet; the hypothesis tells how things work on that planet. In hypothetical thinking, you travel to the planet where things work according to the hypothesis under consideration. So whatever the hypothesis is, things work just that way.

We use hypotheses as mental stepping stones to inform conclusions about how things work in our own world: Earth. Our data are collected on Earth, although we do not know precisely how everything works.



Figure 83: Planet Earth, where we collect data.

When we turn to working with the sample of data that we have collected, we are operating in the world of our data, which is much simpler than Earth. Let's call this Planet Samp. It presumably resembles Earth, but it is potentially subject to sampling bias, and it necessarily lacks detail, like a low-resolution photograph. The sample may also be lacking essential covariates, so conclusions drawn on Planet Samp may deviate systematically from Planet Earth.

Some of our statistical operations take place on a Planet Samp. For instance, *resampling* is the process of taking a new sample on Planet Samp. No amount of resampling is going to acquire data from Planet Earth. Even so, this work on Planet Samp can let us estimate the amount of sampling variation that we would see had we been back on Earth.

In Lesson 28 we will work with two additional planets, one for each of the two hypotheses we are placing in competition. These planets are custom made to correspond to their respective hypothesis. In one example in Lesson 28, we will construct a Planet Sick and a Planet Healthy. All the people on Planet Sick genuinely have the disease, and all the people on Planet Healthy genuinely do not. When we carry out a medical screening test on Planet Sick, every *negative* result is therefore an error. Likewise, on Planet Healthy, every *positive* screening result is an error.

In another example in Lesson 28, we will consider the rate of car accidents. To compute a likelihood, we construct a planet where every car has the specified accident rate, then observe the action on that planet to see how often a sample will correspond to the data originally collected on Earth.

Lesson 29 is about a form of hypothetical reasoning centered on the **Null Hypothesis**. This, too, is a planet, appropriately named Planet Null. Planet Null is in many ways like Planet Samp, but with one huge exception: there are no genuine patterns, all variables are unrelated to one another. Any features we observe are merely accidental alignments.

In Lessons 28 and 29, we will often calculate likelihoods. As you know, a likelihood always refers to a specific hypothesis. The likelihood number is the relative probability of the

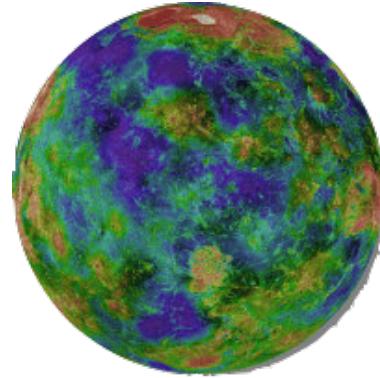


Figure 84: Planet Samp, composed solely of the data in our sample.



Figure 85: Planet Null, where there are no genuine patterns, just random alignments.

observed data **given** that specific hypothesis. Synonyms for “given that specific hypothesis” are “**under** that hypothesis” or “**conditioned on** that hypothesis.” Or, more concretely, think of “given,” “under,” and “conditioned on” as all meaning the same thing: you have *travelled* to the planet where the specific hypothesis holds true.

Now you can have a new mental image of a likelihood calculation. First, travel to the planet corresponding to the specific hypothesis under consideration. On this planet, things always work exactly according to the hypothesis. While on that planet, make many observations; collect many data samples. For each of these samples, calculate the summary statistic. The likelihood is the fraction of samples for which the summary statistic is a match to the summary statistic for the sample we originally took on Earth.

Planet Null is a boring place; nothing ever happens there. One reason for the outsized popularity of Planet Null in statistical tradition is that it is very easy to get to Planet Null. As you’ll see in Lesson 29, to collect a sample on Planet Null simply shuffle the sample you collected on Earth.

Finally, as you will see, in the Neyman-Pearson configuration of hypothetical reasoning, there is an “**alternative hypothesis**”. The alternative hypothesis—that is, Planet Alt—is conjured from your own imagination, experience, and expertise. It is a cartoon planet, drawn to reflect a hypothesis about the world that originally prompted you to undertake the work of collecting and analyzing data.

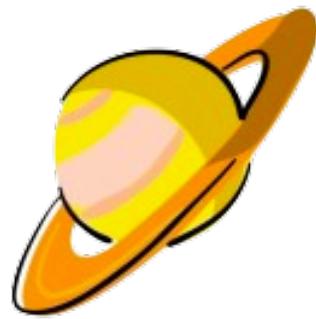


Figure 86: Planet Alt, where things happen the way you imagined they should.

28 Competing hypotheses with Bayesian reasoning

We can start working with Bayesian reasoning even before introducing it as a system; the Bayesian rules are simple consequences of counting and comparison by ratios. To illustrate, consider this familiar situation:

You have undergone a medical testing procedure to figure out if you have a particular illness.

For simplicity, we will call this illness “Sick.” The alternative possibility is that you are “Healthy.”

The two possible results of the testing procedure are “Positive” and “Negative.” By convention, a *positive* test result points toward the subject of the test being sick and a *negative* tests points toward being healthy.

To avoid being long-winded, we’ll denote a positive test result by \mathbb{P} and a negative result by \mathbb{N} . Similarly, we will abbreviate “Sick” by S and “Healthy” by H .

Suppose your result is \mathbb{P} . What is the probability that you are S ?

It’s commonplace to assign too much significance to a test result, reasoning like this: $\mathbb{P} \implies S$. Experience shows, however, that there can be healthy people who, nevertheless, get a positive test result. This is why we introduced the issue of “probability.” Might you be one of the healthy people with a \mathbb{P} result?

Even educated people tend to assume that this probability is almost 1, that a \mathbb{P} means that you are almost certainly S . But for some familiar tests and illnesses, this is far from being the case.

To calculate the probability that you are S given a \mathbb{P} result we need some background information. This information will have been collected by the test developers in the course of making sure their test works before it is released for general use. The information describes two situations:

1. Among S patients, how well does the test work. Specifically, what is the probability of a \mathbb{P} result in the group of S patients. The test developers might establish this by working with a group of clinics. The clinics refer patients who have been diagnosed with S . The test is administered to each of these referred patients and the test results tallied up. Let's suppose, for the purposes of illustration, that 90% of the S patients had a positive result.

Back in Lesson 16 we introduced the word “**likelihood**” to refer to probability of observed data in a world where a given hypothesis is true. In our case, the observed data is \mathbb{P} . The hypothesis is that each of the patients is S . We will write the probability of a \mathbb{P} in a world where the patient is S as $L_S(\mathbb{P})$. The L is a reminder that the quantity is a likelihood and applies only when the patient is S .

2. The test developers need to make sure that, among H people, the \mathbb{P} result is rare. Healthy people should get a \mathbb{N} result! In other words, $L_H(\mathbb{P})$ should be low. To estimate $L_H(\mathbb{P})$, the test developers will work with a completely different group of people than in (1). For instance, they might recruit the neighbors of the people in (1), and send them to a clinic to confirm that they really are H . So, group (2) will consist only of H people. Each is given the test and the results tallied. The fraction of the H people who test \mathbb{P} is $L_H(\mathbb{P})$. Let's suppose that the developer's work shows that $L_H(\mathbb{P}) = 0.20$. That is, a H person is pretty likely to get a \mathbb{N} result, but not certain.

These two pieces of information in (1) and (2) are both in the form of likelihoods. But each is relevant only to the given situation. $L_S(\mathbb{P})$ applies only to people who are known to be sick. $L_H(\mathbb{P})$ is applicable only to healthy people.

One important use for tests such as the one we described is “**medical screening**.” Screening is applied to members of the general population who display no relevant symptoms and have no particular reason to believe they might be S . Familiar examples of tests used for screening: mammography for breast cancer, PSA for prostate cancer, Pap smears for cervical cancer. Screening also occurs in non-medical settings, for instance

drug tests or criminal background checks required by employees for current or prospective workers.

Medical tests are also used in non-screening settings. For example, a person who is feeling flu-like symptoms will often take a COVID test. Similarly, a person going to a wedding might take a COVID test even though she is not feeling any symptoms.

The difference between screening settings and non-screening settings is a matter of degree. The number used to quantify the setting is called the “**prevalence**,” which is the fraction of people in the test-taking group who are S .

The test developers applied the test to two different groups of people. In the S group the prevalence is 100%. In the H group the prevalence is 0.

Now we come to you and your \mathbb{P} result. You are a member of a group. Which group that is depends on your circumstances, for instance your age, sex, and nationality. Other risk factors may also come into the definition of your group, for example, fitness or whether you drink alcohol regularly. Whatever risk factors define your group, the number you need to know is the prevalence of S in your group.

It is usually impractical to measure prevalence precisely in a large, general group of people. Doing so requires a random sample of a large size and then subjecting each person in the sample to a diagnostic procedure. In practice, the stated prevalence of S in a group will only be an estimate based on how frequently people are diagnosed with S . For instance, by looking retrospectively at insurance and medical records, one can find the risk of developing S over a 5-year period. We will leave such important issues to public health specialists, since our goal here is to show you the logic of hypothetical thinking.

Suppose, for the purposes at hand, that the prevalence of S in your group is 2%. When you walked into the medical clinic your risk of S was therefore 2%. What is your risk once you have been handed your \mathbb{P} test result?

To help you see how the calculation is organized, let’s look at your group graphically. Imagine they are assembled in a sports field and a picture has been taken from an overhead drone, as

in Figure 87(a). The S people are drawn as triangles and the H as a circle.

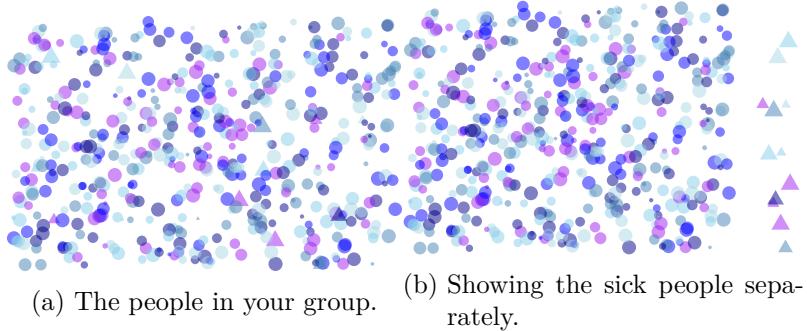


Figure 87: The members of your group, gathered on a playing field.

Naturally, the member of your group differ from one another, shown by size and color in Figure 87(a). Since the prevalence in your group is 1 percent, about 1 in 100 of the people are S , even though they don't know it yet. In Figure 87(b), we have moved the S people off to the right, just for display purposes.

Imagine that everyone in your group takes the medical test. Most will test \mathbb{N} , since the prevalence is small. As for the few who test \mathbb{P} , we will change their color to red.

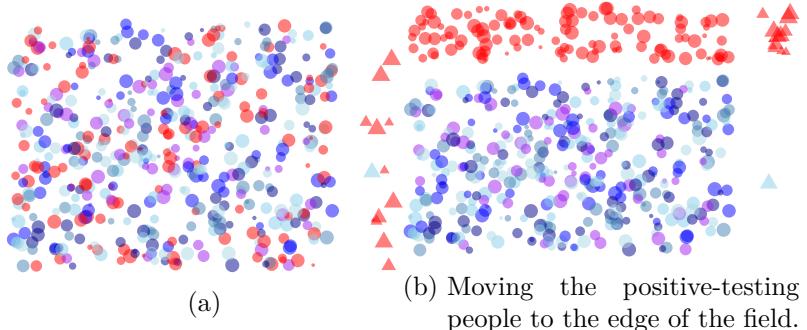


Figure 88: The same people as in Figure 87, but showing those who tested positive in red.

Now we can answer the original question:

Suppose your result is \mathbb{P} . What is the probability that you are S ?

We don't know which dot in Figure 88 is you, but we do know that you are one of the red ones. The probability you seek is the fraction of red people who are at the S end of the field. We can answer the question by counting the dots. By eye, the S are about 10% of the \mathbb{P} .

We could also answer the probability question by simple wrangling. The (simulated) data behind Figure 88 are called `Your_group`. The wrangling:

```
Your_group |>
  filter(test == "P") |>                               ①
  summarize(mean(sick=="S"))                           ②
```

```
| mean(sick == "S") |
| :-----:|
| 0.09677 |
```

i Arithmetic calculation

We demonstrated using a data simulation how to compute the probability that you are S given your \mathbb{P} .

In constructing the simulation, we used the relevant information:

- $L_S(\mathbb{P}) = 0.90$
- $L_H(\mathbb{P}) = 0.20$
- Prevalence in your group is 0.02.

We don't actually need the simulation. We can carry out the calculation of the probability that a \mathbb{P} person is S with arithmetic.

- The proportion of people in your group who are S is the prevalence: $p(S) = 0.02$.
 - Of these S people, the proportion who will test positive is $L_S(\mathbb{P})$.
 - So, the proportion of the whole group who are both S and \mathbb{P} is $L_S(\mathbb{P})p(S) = 0.9 \times 0.02 = 0.018$.

This corresponds to the red dots in the upper right quadrant of Figure 88(b).

- The proportion of people in your group who are H is 1 minus the prevalence: $p(H) = 1 - 0.02 = 0.98$
 - Of these H people, the proportion who will test positive is $L_H(\mathbb{P}) = 0.9 \times 0.01 = 0.009$.
 - So, the proportion of the whole group who are both H and \mathbb{P} is $L_H(\mathbb{P})p(H) = 0.20 \times 0.98 = 0.196$.
- Putting these two proportions together, we get $0.196 + 0.018 = 0.216$ have a \mathbb{P} .

We want the proportion of sick \mathbb{P} people out of all the \mathbb{P} people, or:

$$p(S \mid \mathbb{P}) = \frac{0.018}{0.198 + 0.018} = 0.084 .$$

Even though you tested \mathbb{P} , there is less than a 10% chance that you are S !

28.1 Bayesian thinking

We used the specific, concrete situation of medical testing to illustrate Bayesian thinking, the result of which was the probability that you are S given your \mathbb{P} result. In this section we will describe Bayesian thinking in more general terms.

Bayesian thinking is analogous to deductive reasoning in geometry. The purpose of both is to generate new statements (e.g. “the two lines are not parallel”) from existing statements (e.g. “the two lines cross at a point”) that are posited to be true. In geometry, statements are about lengths, angles, areas, and so on. In Bayesian thinking, the statements are about a set of hypotheses, observations, and likelihoods.

Bayesian thinking involves two or more hypotheses that you want to choose between based on observations. In the medical testing example, the two hypotheses were S and H .

This claim that S and H are hypotheses may surprise you. Aren't S and H two different objective states of being, one of which is true and the other one isn't? In the Bayesian system, however, such states are *always uncertain*. We quantify the uncertainty by relative probabilities.

For instance, a possible Bayesian statement about S and H goes like this, "In the relevant instance, S and H have relative probabilities of 7 and 5 respectively." (Many people prefer to use "belief" instead of "statement.")

The book-keeping for Bayesian statements is easiest when there are only two hypotheses in contention. In this section, we will stick to that situation. Since there are only two hypotheses, any statement about them can be translated from relative probabilities into "odds." For instance, "relative probabilities of 7 and 5 respectively" is equivalent to "the odds of S are 7 to 5, that is 1.4. (The odds of the other hypothesis, H in the example, are just the reciprocal of the odds of the first hypothesis.)

As mentioned previously, Bayesian thinking is a way of generating new statements out of old ones that are posited to be true. The words "new" and "old" suggest that *time* is in play, and that's a good way to think about things. Conventionally the words **prior** and **posterior** are used to indicate "old" or "new." From prior statements we will deduce posterior statements.

Observations are the thing that drive the derivation from prior statements to posterior statements. For instance, in the medical testing example, a good prior statement about S for you relates to the prevalence of S in your relevant reference group. We stipulated before that this is 0.02. In terms of odds, this amounts to saying that the odds of S on the day before the test were $2/98 = 0.02041$. Or, better, your *prior* for S is 0.02041.

Now new information comes along: your test result: \mathbb{P} . We will use this to transform your prior into a posterior informed by the test result. Like this:

$$\text{posterior for } S \xleftarrow{\mathbb{P}} \text{prior for } S$$

Keep in mind that both the prior and posterior are in the form of "odds of S .

How do we accomplish the transformation? This is where the likelihoods come in. There is one \mathbb{P} likelihood for each of the two hypotheses. We will write them as a fraction:

$$\text{Likelihood ratio}(\mathbb{P}) \equiv \frac{L_S(\mathbb{P})}{L_H(\mathbb{P})}$$

Note that the likelihood for the S hypothesis is on the top and H is on the bottom. This is because we are framing our prior and posterior in terms of the odds of S . Also, both likelihoods involve the same observation, in this case the \mathbb{P} result from your test.

Here is the formula for the transformation:

$$\text{posterior for } S = \text{Likelihood ratio}(\mathbb{P}) \times \text{prior for } S$$

i Example calculation

We assumed your reference group has a prevalence of 2%. Translating this probability into the form of odds gives:

$$\text{prior for } S = \frac{2}{98} = 0.02041$$

The relevant likelihoods were established, as described in the previous section, by the test developer's study of S patients and H individuals.

$$\text{Likelihood ratio}(\mathbb{P}) \equiv \frac{L_S(\mathbb{P})}{L_H(\mathbb{P})} = \frac{0.90}{0.20} = 4.5$$

Consequently, the posterior (driven by the observation \mathbb{P}) is

$$\text{posterior for } S = 4.5 \times 0.02041 = 0.09184 .$$

This posterior is stated as odds. In terms of probability, it corresponds to $\frac{0.09184}{1+0.09184} = 0.084$, exactly what we got when we counted red circles and red triangles in Figure 88!

28.2 Bayes with multiple hypotheses

The previous section showed the transformation from prior to posterior when there are only two hypotheses. But Bayesian thinking applies to situations with any number of hypotheses.

Suppose we have N hypotheses, which we will denote H_1, H_2, \dots, H_N .

Since there are multiple hypotheses, it's not clear how odds will apply. So instead of stating priors and posteriors as odds, we will write them as *relative probabilities*. We'll write the prior for each hypothesis as $prior(H_i)$ and the posterior as $posterior(H_i)$.

Now an observation is made. Let's call it \mathbb{X} . This observation will drive the transformation of our priors into our posteriors. As before, the transformation involves the likelihood of \mathbb{X} under the relative hypotheses. That is, $L_{H_i}(\mathbb{X})$. The calculation is simply

$$posterior(H_i) = L_{H_i}(\mathbb{X}) \times prior(H_i) \text{ in relative probability form}$$

If you want to convert the posterior from a *relative probability* into an ordinary probability (between 0 and 1), you need to collect up the posteriors for all of the hypotheses. The notation $p(H_i | \mathbb{X})$ is conventional, where the posterior nature of the probability is indicated by the $| \mathbb{X}$). Here's the formula:

$$p(H_i | \mathbb{X}) = \frac{posterior(H_i)}{posterior(H_1) + posterior(H_2) + \dots + posterior(H_N)}$$

... { .callout-note } ## Example: Car safety

Maybe move the example using the exponential distribution from the Likelihood Lesson to here.

...

28.3 Accumulating evidence

THE CYCLE OF ACCUMULATION

Note: There are specialized methods of Bayesian statistics and whole courses on the topic. An excellent online course is [*Statistical Rethinking*](#).

29 Hypothesis testing

“**Standard operating procedures**” (SOPs) are methods, rules, or actions established for performing routine duties or in designated situations. SOPs are usually associated with organizations and bureaucratic operations such as hiring a new worker or responding to a report of broken equipment or a spilled chemical. SOPs are intended to routinize operations and help coordinate different components of the organization.

There are SOPs important to scientific work. For example, research with human subjects undergoes an “institutional review” SOP that ensures safety and that subjects are thoroughly informed about risk.

This Lesson is about an SOP called “**hypothesis testing**,” a statistical procedure intended to inform decision-making by researchers, readers, and editors of scientific publications. For example, an individual researcher or research team needs to assess whether the data collected in an investigation is adequate to serve the intended purpose. A reader of the scientific literature needs a quick way to assess the validity of claims made in a report. A journal editor, needs a straightforward means to screen submitted manuscripts to check that the claims are supported by data and that the claims are novel to the journal’s field. The hypothesis testing SOP is designed to serve these needs. The words “adequate,” “quick,” and “straightforward” in the previous sentences correctly reflect the tone of hypothesis testing.

Science is often associated with ingenuity, invention, creativity, deep understanding, and the quest for new knowledge. Perhaps understandably, “SOP” rarely appears in reports of new scientific findings. Unfortunately, failing to see “hypothesis testing” as an “SOP” results in widespread misunderstanding and inappropriate use of the results from hypothesis testing.

29.1 The Null hypothesis

One of the best descriptions of hypothesis tests comes [from the 1930s](#), when they were just starting to gain acceptance. Ronald Fisher, who can be credited as the inventor, wrote this description, using the name he preferred: “**significance test**.

[Significance testing] is a technical term, standing for an idea very prevalent in experimental science, which no one need fail to understand, for it can be made plain in very simple terms. Let us suppose, for example, that we have measurements of the stature of a hundred Englishmen and a hundred Frenchmen. It may be that the first group are, on the average, an inch taller than the second, although the two sets of heights will overlap widely. ... [E]ven if our samples are satisfactory in the manner in which they have been obtained, the further question arises as to whether a difference of the magnitude observed might not have occurred by chance, in samples from populations of the same average height. If the probability of this is considerable, that is, if it would have occurred in fifty, or even ten, per cent. of such trials, the difference between our samples is said to be "insignificant." If its probability of occurrence is small, such as one in a thousand, or one in a hundred, or even one in twenty trials, it will usually be termed "significant," and be regarded as providing substantial evidence of an average difference in stature between the two populations sampled. In the first case the test can never lead us to assert that the two populations are identical, even in stature. We can only say that the evidence provided by the data is insufficient to justify the assertion that they are different. In the second case we may be more positive. We know that either our sampling has been exceptionally unfortunate, or that the populations really do differ in the sense indicated by the available data. The chance of our being deceived in the latter conclusion may be very small and, what is more important, may be calcu-

lable with accuracy, and without reliance on personal judgment. Consequently, while we require a more stringent test of significance for some conclusions than for others, no one doubts, in practice, that the probability of being led to an erroneous conclusion by the chances of sampling only, can, by repetition or enlargement of the sample, be made so small that the reality of the difference must be regarded as convincingly demonstrated.” (Emphasis added.)

The possibility that “a difference of the magnitude observed ... occurred by chance” came to be called the **“Null hypothesis.”**

The hypothesis testing SOP centers around the Null hypothesis. To understand what the Null is, it may help to start by pointing out what it is *not*. Consider this mainstream definition of the scientific method:

“a method of procedure that has characterized natural science since the 17th century, consisting in systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses.” - Oxford Languages

It would be easy to conclude from the definition that “testing ... of hypotheses” is central to science. However, the “hypotheses” involved in the scientific method are not the “Null hypothesis” that is implicated in the statistical “hypothesis testing” SOP.

A famous example of a scientific hypothesis is Newton’s law of gravitation from 1666. This hypothesis was tested in various ways: predictions of the orbits of the planets and moons around planets, laboratory detection of minute attractions in experimental apparatus, and so on. In the mid-1800s, it was observed that the movement of Mercury was not entirely in accord with Newton’s law. This is an example of scientific *testing* of a hypothesis; Newton’s law failed the test. In response, various theoretical modifications were offered, such as the presence of a hidden planet called Vulcan. These were ultimately unsuccessful. However, in 1915, Einstein published a modification of Newton’s gravitation called “the theory of general relativity.”

This correctly accounts for the motion of Mercury. Additional evidence (such as the bending of starlight around the sun observed during the 1919 total eclipse) led to the acceptance of general relativity. The theory has continued to be tested, for example looking for the actual existence of “black holes” predicted by general relativity.

The Null hypothesis is different. The same Null hypothesis is used in diverse fields: biology, chemistry, economics, geology, clinical trials of drugs, and so on, more than can be named. This is why the Null is taught in statistics courses rather than as a principle of science.

A common sort of question in widely ranging fields is whether one variable is related to another. To be concise, we will call the two variables Y and X. The statistical method often used to address this question is regression modeling: $Y \sim X$. The Null hypothesis is that Y and X are unrelated, that is, that the X coefficient is zero. Throughout these Lessons, you have the Null tested using confidence intervals: Does the confidence interval on the X coefficient contain zero? If not, your data provide evidence that X and Y are related.

i Relationships, differences, and effects

We have been using the general term “relationship” to name the connection between Y and X. Other words are also used.

For example, when X is categorical, it effectively divides the data frame into **groups** of specimens. A relationship between Y and X can then be stated in everyday terms: “Are the groups different in terms of their Y values?”

When X is quantitative, a relationship between Y and X can be phrased, “Does X have an effect on Y?”

The Null hypothesis in statistics is the presumption that there are no group differences due to categorical X or, similarly, that there is no effect of X on Y. We can test the Null. To illustrate, consider Y to be the **height** of the children represented in the **Galton** data frame and X to be the **sex** of the child. The Null is that the sexes do not differ by height. Here’s the test:

```
Galton |>
  model_train(height ~ sex) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	63.87	64.11	64.35
sexM	4.79	5.119	5.448

The confidence interval on `sexM` does not contain zero. The `Galton` data refute the presumption that the two groups do *not* differ in height. In the language of statistical hypothesis testing, one “**rejects the Null hypothesis.**” The hypothesis testing process is identical when X is quantitative. For instance, does the mother’s height have a non-zero effect on the child’s height?

```
Galton |>
  model_train(height ~ mother) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	40.3	46.69	53.09
mother	0.2134	0.3132	0.4129

The confidence interval on `mother` does not include zero, so one “rejects the Null hypothesis.”

If a confidence interval includes zero, then we can’t rule out (based on our data) that there might be no-difference/no-effect. The language used in hypothesis testing is: “We fail to reject the Null.”

It might have been better if the Null hypothesis were called the “null presumption.” That would properly put more distance between the statistical test of a presumption and the sort of genuine, contentful hypotheses used to define the “scientific method.”

The phrase, “We fail to reject the Null,” however, hits the nail right on the head. When you take a fair test in school, the failed test indicates that your understanding or knowledge is inadequate. A failed test says something about the student, not the truth or falsity of the contents of the test itself.

Similarly, a hypothesis test is not really about the Null hypothesis. Instead, it is a test of the researcher’s method for experiment, measurement, data collection (e.g. sample size), analysis of the data (e.g. consideration of covariates), and so on. The test determines whether these methods are fit for the purpose of scientific discovery. The passing grade is called “reject the Null.” The failing grade is “fail to reject the Null.”

It’s common sense that your research methods should be fit for the purpose of scientific discovery. If you can’t demonstrate this, then there is no point in continuing down the same road in your research. You might decide to change your methods, for instance increasing the sample size or guarding more carefully against contamination or other experimental pitfalls. Or, you might decide to follow another avenue of research.

Few readers or journal editors are interested in a report that your research methods are not fit for purpose. Consequently, the demonstration of methodological fitness—rejecting the Null—is often a requirement for publication of your work.

There are rare occasions when there is genuine interest in demonstrating no difference between groups or no effect of one variable on another. On these occasions, a hypothesis test is misplaced. Even if your research methods are sound, you would properly fail to reject the Null. Taken literally, the test results would (wrongly) show that your methods are unsound. Instead, it’s appropriate to demonstrate the fitness off your methods in a setting where there is an actual difference or effect to detect. (This issue will come up again when we look at Neyman-Pearson hypothesis testing.)

29.2 Formats for NHT results

Hypothesis tests were originally (and still are in some cases) called “**significance tests**.” They are also called “**Null**

hypothesis tests" or even "**Null hypothesis significance tests.**" We will use the abbreviation **NHT**.

Only two qualitative statements are allowed for conveying the results of NHT: "reject the Null" or "fail to reject the Null."

Confidence intervals provide a valid quantitative statement of the NHT result: an interval excluding zero corresponds to "reject the Null," an interval incorporating zero indicates that the work "fails to reject the Null." Of course, the primary role of confidence intervals is to indicate the precision of your measurement of the difference/effect-size. It's a bonus that confidence intervals fit in with the NHT SOP.

However, for historical reasons, the use of confidence intervals to quantify NHT has become common only in the last few decades. This may be because NHT was introduced before confidence intervals were invented.

A widespread way to quantify the result of NHT is a number called a "**p-value**" that is between zero and one. A small p-value, near zero, signifies rejection of the Null hypothesis. Typically, "small" means less than 0.05, but other values are preferred in some fields. The numerical value of "small" is called the "**significance level**." Often, instead of "reject the Null," reports state that the results are "significant at the 0.05" level or at whatever significance level is used for the field of research. Even more consisely, in place of "reject the Null," many researchers like to say that their results are "significant." Such researchers also tend to replace "fail to reject the Null" with "non-significant."

There have been persistent calls by statisticians to stop using the word "significant" in NHT because it can easily mislead. The ordinary, everyday meaning of "significance" tricks people into thinking that "statistically significant" results are also "important," "useful," or "notable" in practice. NHT is merely an SOP for documenting that research methods are fit for purpose, not a reckoning that the results have practical importance. Understandably, scientists are flattered by the misleading implications of "significance." For journalists, quoting a scientist's claim of "significance" is a magic wand to charm the unaware

reader into concluding that a news item is worth reading. Consider, for instance, a clinical trial of a drug where the confidence interval points to a reduction in high blood pressure by 0.5 to 1.5 mmHg. This reduction is so trivial that the drug has no medical use. However, since the confidence interval excludes zero, the reduction can be reported as “significant” in the technical sense of NHT. A genuine demonstration of practical significance requires a large effect size, not merely a narrow confidence interval.

This confusing situation could be avoided entirely by switching from “significant” to a word that conveys the correct meaning. For example, statistician [Jeffrey Witmer](#) has proposed the word “discernible” be used in place, as in, “The difference between groups is statistically discernible,” or, “We found a discernible difference.”

29.3 Calculating “significance”

Let’s return to Ronald Fisher’s account of “significance testing” given in Section 29.1. In the paragraph quoted there, he wrote:

“The chance of our being deceived [by sampling variation] may be calculable with accuracy, and without reliance on personal judgment.”

How is this calculation to be performed? Fisher gives this description, which follows the paragraph quoted in Section 29.1.

“The simplest way of understanding quite rigorously, yet without mathematics, what the calculations of the test of significance amount to, is to consider what would happen if our two hundred actual measurements were written on cards, shuffled without regard to nationality, and divided at random into two new groups of a hundred each. This division could be done in an enormous number of ways, but though the number is enormous it is a finite and a calculable number. We may suppose that for each

of these ways the difference between the two average statures is calculated. Sometimes it will be less than an inch, sometimes greater. If it is very seldom greater than an inch, in only one hundredth, for example, of the ways in which the sub-division can possibly be made, the statistician will have been right in saying that the samples differed significantly.”

Fisher wrote before the availability of general-purpose computers. Consequently, for his technical work he relied on algebraic formulas. Standard statistical textbooks will offer half-a-dozen formulas, which misleadingly suggests that the p-value is technically difficult and highly precise. However, the underlying logic is straightforward and the assumed precision of formula-based methods is misleading. Or, as Fisher continued,

“Actually, the statistician does not carry out this very simple and very tedious process, but his conclusions have no justification beyond the fact that they agree with those which could have been arrived at by this elementary method.”

With software, the “tedious process” can easily be carried out. First, we’ll imagine Fisher’s two hundred actual measurements in the form of a modern data frame, which, lacking Fisher’s actual playing cards, we’ll simulate:

height	nationality
68	French
68.5	English
71.5	French
68	French
68	English
69	English

The calculation of the difference in average heights between the nationalities is computed in the way we have used so often in these Lessons:

```
Height_data |>
  model_train(height ~ nationality) |>
  conf_interval() |>
  select(term, .coef)
```

term	.coef
(Intercept)	69.21
nationalityFrench	-0.7844

In our sample, the Frenchmen are shorter than the Englishmen by -0.8 inches on average.

Let's continue with the process described by Fisher, and "shuffle without regard to nationality, and divide at random into two new groups of a hundred each."

```
Height_data |>
  model_train(height ~ shuffle(nationality)) |> ①
  conf_interval() |> ②
  select(term, .coef)
```

- ① This is "shuffling without regard to nationality" and "dividing at random", all in one step!
- ② And calculate the mean difference in heights.

term	.coef
(Intercept)	69
shuffle(nationality)French	-0.072

You can see that the shuffling has created a much smaller coefficient that we got on the actual data. Also, note that the confidence interval now includes zero, as expected when the "nationality" is randomized.

Fisher instructed us to do this randomization "in an enormous number of ways." In our language, this means to do a large number of trials in each of which random shuffling is performed and the coefficient calculated. Like this:

```
Trials <-
  Height_data |>
  model_train(height ~ shuffle(nationality)) |>    ①
  conf_interval() |>
  trials(500) |>
  filter(term == "shuffle(nationality)French")
```

What remains is to compare the results from the shuffling trials to the coefficient `nationalityFrench` that we got with the non-randomized data: -0.8 inches.

```
Trials |>
  filter(abs(.coef) >= abs(-0.8)) |>
  nrow()
```

```
[1] 10
```

In only 7 of 500 trials, did the shuffled data produce a coefficient as large in magnitude than observed in the non-randomized data. Again, to quote Fisher, “If it is very seldom greater than an inch [0.8 inches in our data], in only one hundredth of the [trials], the statistician will have been right in saying that the samples differed significantly.” More conventionally, nowadays, and at Fisher’s recommendation, the threshold of one-in-twenty (or, equivalently, 25 in 500 trials) is reckoned adequate to declare “significance.”

Of course, remember that “[significance] is a technical term.” There is nothing in the calculation to suggest that the “significant” result is important for any practical purpose. For instance, knowing that Frenchmen are on average 0.8 inch shorter than Englishmen would not enable us to predict from a man’s height whether he is French or English.

29.4 The p-value

In the previous section, we calculated that in 7 of 500 trials the shuffled coefficient is at least as big in magnitude as the 0.8 difference seen in the actual data. This fraction—7 of 500—is now

called the p-value. For regression modeling, there are formulas to find the p-values for coefficients without conducting many random trials. `conf_interval()` will show these p-values, if you request it.

```
Height_data |>
  model_train(height ~ nationality) |>
  conf_interval(show_p = TRUE)
```

term	.lwr	.coef	.upr	p.value
(Intercept)	68.78	69.21	69.64	1.938e-269
nationalityFrench	-1.423	-0.7844	-0.1462	0.01625

The p-value on the intercept is effectively zero. As it should be. No amount of shuffling the height data will produce an average English height of 0 inches! The p-value on `nationalityFrench` is $p=0.016$. That's a little bigger than 5 out of 700. But simulation results are always random to some extent.

The p-value calculated from the formula seemingly has no such random component, yet we know that every summary statistic, even a p-value, has sampling variation. To paraphrase Fisher, “[p-values have] no justification beyond the fact that they agree with those [produced by simulation].”

A p-value can be calculated for any sample statistic by the shuffling method. There are also formulas for them when dealing with R^2 :

```
Height_data |>
  model_train(height ~ nationality) |>
  R2()
```

n	k	Rsquared	F	adjR2	p	df.num	df.denom
200	1	0.02882	5.875	0.02391	0.01625	1	198

It is not an accident that the p-value on R^2 reported from the model is identical to the p-value calculated on the only non-intercept coefficient term in a model.

i Hypothesis test with confidence interval

Confidence intervals had not come into widespread use when Fisher wrote the material quoted above. But confidence intervals provide a shortcut to hypothesis testing, at least when it comes to model coefficients. Simply check whether the confidence interval includes zero. If so, the conclusion is “failure to reject the Null hypothesis.” But if zero is outside the range of the confidence interval, “reject the Null.”

The confidence interval hypothesis test does not always agree exactly with the test as done using a p-value. But the precision of formula-based p-values is illusory. Many statisticians recommend using confidence intervals instead of p-values, particularly because they provide information about the effect size. That’s been our practice throughout these Lessons.

As it happened, Fisher denigrated the idea of confidence intervals. In this, he is utterly out of step with mainstream statistics today.

29.5 Power and the alternative hypothesis

NHT was introduced early in the 1920s. By the end of the decade an extension was proposed that incorporated into the reasoning a second, scientific hypothesis called the **“Alternative hypothesis.”** We will call the extended version of hypothesis testing NP, after the two statisticians Jerzy Neyman (1894-1981) and Egon Pearson (1895-1980).

The point of NP was two-fold:

- i. To provide some guidance in interpreting a “fail to reject the Null” result.
- ii. To guide scientists in designing studies, for example, deciding on an appropriate sample size.

Recall, in the context of $Y \sim X$, that the Null hypothesis is the presumption that Y and X are not connected to one another. In modeling terms, the Null is that the coefficient on X is zero.

The alternative hypothesis can also be framed in terms of the coefficient on X . In its simplest form, the alternative is a specific, non-zero, numerical value for the coefficient on X . One

purpose of the alternative is to provide an idea about what motivates the research. For instance, a study of a drug that reduces blood pressure might have an alternative that the drug reduces pressure, on average, by 10 mmHg. In many cases, the alternative is set to be an effect size or difference between groups of the smallest that would be of interest in the application of the research. (You'll see the logic behind "smallest" in a bit.)

Another purpose for the alternative hypothesis is to deal with situations where the Null or something like it might actually be true. In such situations, the result of NHT will be to "fail to reject the Null." It would be nice to know, however, whether the failure should be ascribed to inadequate methods or to the Null being true.

Stating an alternative hypotheses draws, ideally, on expertise in the subject matter and the hoped-for implications of the research if it is successful.

The alternative framed *before* data are collected. It is part of the SOP of study design. For our purposes here, it suffices to think of the alternative being implemented as a simulation of the sort discussed in Lesson 14 built to implement the smallest effect of interest and incorporating what is known of subject to subject variation.

Setting an appropriate sample size is an important part of the study design phase. For the sake of economy, the sample size should be small. But to have better precision—i.e., tighter confidence intervals—a larger sample size is better. One way to resolve this trade-off is in the spirit of NHT: aim for a precision that is just tight enough to make it likely that the Null will be rejected.

Likelihood, as we saw in Lesson 16, is a probability calculated given a stated hypothesis. The relative hypothesis here is the alternative hypothesis. To find the likelihood of rejecting the Null for a proposed sample size, run many trials of the simulation and carry out the NHT calculations for each. Then count the proportion of trials in which the Null is rejected. This fraction of successfully rejected trials. This fraction, a number between zero and one, is called the "**power**."

i Example: Get out the vote!

Consider the situation of the [political scientists](#) who designed the study in which the `Go_vote` data frame was assembled.

To carry out the study, they needed to decide how many postcards to send out. To inform this decision they looked at existing data from the 2004 primary election to determine the voting rate:

```
Go_vote |> count(primary2004) |>  
  mutate(proportion = n / nrow(Go_vote))
```

`Go_vote` looked at whether postcards sent to registered voters led to an increase in the rate of voting.

primary2004	n	proportion
abstained	183098	0.5986
voted	122768	0.4014

A turn-out rate of 40%.

Next, the researchers would speculate about the effect of the postcards might be. Such speculation can be informed by previous work in the field. This is one reason that research reports often contain a “literature survey.” Here’s an excerpt from the literature survey in the journal article reporting the experiment and its results:

“Prior experimental investigation of publicizing vote history to affect turnout is extremely limited. Our work builds on two pilot studies, which appear to be the only prior studies to examine the effect of providing subjects information on their own vote history and that of their neighbors (Gerber et al. 2006). These two recent experiments, which together had treatment groups approximately [2000 voters], found borderline statistically significant evidence that social pressure increases turnout.”

Such experiments indicated an increase in turnout of approximately 1-2 percentage points. For demonstration purposes, let’s set the alternative hypothesis to be an increase by 1 percentage point from the baseline voting level of 40% observed in the 2004 primary. This gives us the essential information to build the simulation implementing the alternative hypothesis.

```
Alternative_sim <- datasim_make(  
  postcard <- bernoulli(n, prob=0.33, labels = c("control", "card")), ①  
  vote <- bernoulli(n, prob = ifelse(postcard == "card", 0.41, 0.40), ②  
    labels = c("abstained", "voted"))  
)
```

- ① Send a postcard to one-third of households in the experiment.
- ② For postcard recipients, simulated voting rate will be 0.41. For the control group, 0.40 is the rate.

A single trial of the simulation and the follow-up analysis of the data looks like this. We will start with an overall sample size of n=1000.

A simulation isn’t the only way to calculate the power. In this simple setting it can also be done using algebra.

```
set.seed(102)
Alternative_sim |> sample(n=1000) |>
  model_train(zero_one(vote, one="voted") ~ postcard) |>
  conf_interval()
```

term	.lwr	.coef	.upr
(Intercept)	-0.4965	-0.273	-0.05165
postcardcontrol	-0.4778	-0.2075	0.06365

We are interested only in the coefficient on `postcard`, specifically whether the confidence interval excludes zero, corresponding to rejecting the Null hypothesis. Let's run 500 trials of the simulation:

```
Sim_results <- Alternative_sim |> sample(n = 1000) |>
  model_train(zero_one(vote, one = "voted") ~ postcard) |>
  conf_interval() |>
  trials(500) |>
  filter(term == "postcardcontrol")
```

.trial	term	.lwr	.coef	.upr
1	postcardcontrol	-0.43	-0.16	0.1
2	postcardcontrol	-0.25	0.013	0.28
3	postcardcontrol	-0.084	0.19	0.46
4	postcardcontrol	-0.19	0.077	0.35
5	postcardcontrol	-0.54	-0.28	-0.0098

We can count the number of trials in which the confidence interval on `postcardcontrol` *excludes* zero. Multiplying `.lwr` by `.upr` will give a positive number if both are on the same side of zero. The power for the simulated sample size is the fraction of trials that exclude zero.

```
Sim_results |>
  mutate(excludes = (.lwr * .upr) > 0) |>
  summarize(power = mean(excludes))
```

$$\begin{array}{c} \hline \text{power} \\ \hline 0.072 \end{array}$$

This is a power of about 7%.

Ideally, the *power* of a study should be close to one. This can be accomplished, for any alternative hypothesis, by making the sample size very large. However, large sample sizes are expensive or impractical, so researchers have to settle for power

less than one. A power of 80% is considered adequate in many settings. Why 80%? SOP.

In the voting simulation with $n=1000$, the power is about 7%. That's very small compared to the target power of about 80%. In Exercise 29.5 you can explore how big a sample size is needed to reach 80%.

29.6 False discovery

The screenshot shows a research paper page. At the top left, there is a box indicating 'Not Available for Download'. To the right, there are sharing icons for Facebook, Twitter, Email, and a link. Below these, there is a blue star icon labeled 'Add Paper to My Library'. The main title of the paper is 'Partisan Mail and Voter Turnout: Results From Randomized Field Experiments', published in 'Electoral Studies, Vol. 22, pp. 563-579, 2003'. The date posted is '19 Dec 2008'. The authors listed are Alan Gerber, Donald P. Green, and Matthew Green. The paper was written on December 17, 2008. On the right side of the page, there is a sidebar with a section titled 'Do you have negative results from your research you'd like to share?' containing a 'Submit Negative Results' button. Below this, there is a 'Paper statistics' section showing 'ABSTRACT VIEWS 460' and a 'PlumX Metrics' section featuring a purple asterisk icon.

Figure 89: Negative results request

29.7 Hypothesis testing interpreted by a Bayesian

Null hypothesis testing (NHT) and Neyman-Pearson (NP) have similarities.

- i. Both are centered on the Null hypothesis, and for both that hypothesis amounts to a claim that the coefficient on X is zero in the model $Y \sim X$.
- ii. Both produce a result that has two possible values: “reject the Null” or “fail to reject the Null.” Often, this result is stated as a p-value.

- iii. In both, the test result refers *only* to the Null hypothesis. Once the alternative and sample size has been selected, NP works the same as Bayes. At this point, only the Null is involved in the calculations.
- iv. NP involves an alternative hypothesis which is used only to assess the “power” of the null hypothesis test. The concept of power doesn’t apply in NHT since there is no alternative hypothesis in NHT. In NP, the calculation of power does not refer to the data actually collected. Power is calculated in the setup to the study, prior to the collection of data. Power is typically used to guide the selection of sample size.

There are similarities and difference between both NHT and NP compared to Bayesian reasoning.

- i. All three forms involve a *summary* of the data. This summary might be a model coefficient, or an R^2 , or sometimes something analogous to these two.
- ii. Bayesian analysis always involves (at least) two hypotheses. It is perfectly reasonable to use the Null as one of the hypotheses and the alternative as the other. For comparison to NHT and NP, we will use those two hypotheses.
- iii. The output of the Bayesian analysis is the posterior odds of the Alternative hypothesis. The posterior odds of the Null come for free, since odds of the Null is simply the reciprocal of the odds of the Alternative. The odds refer to *both* of the hypotheses.

Consider the Bayes formula for computing the posterior probability in odds form:

$$\text{posterior odds for Alternative} = \text{Likelihood ratio}_{A/N}(\text{summary}) \times \text{prior odds for Alternative}$$

Neither NHT or NP makes any reference to a prior odds. NHT doesn’t even involve an Alternative hypothesis. NP does involve an Alternative, but this contributes not at all to the outcome of the test.

Any statement of prior odds necessarily refers to the beliefs of the researchers. NHT and NP are often regarded as more

objectives, since the beliefs don't enter in to the calculation. Or, at least, the beliefs don't enter *explicitly*. Presumably the reason the researchers took on the study in the first place is some level of subjective belief that the Alternative is a better description of the real-world situation than the Null.

Even though the prior odds are subjective, the likelihood ratio is not. The likelihood ratio multiplies the prior odds to produce the posterior odds. In the realm of medical diagnosis, a likelihood ratio of 10 or greater is considered "strong evidence" in favor of the Alternative. The bigger the likelihood ratio, the stronger the claim of the Alternative hypothesis.

Since the likelihood ratio encodes what the data has to say, irrespective of prior beliefs, it's tempting to look at NHP and NP with an eye to a possible analog of the likelihood ratio. For reference, let's write the likelihood ratio in terms of the individual likelihoods:

$$\text{Likelihood ratio}_{A/N}(\text{summary}) = \frac{L_{Alt}(\text{summary})}{L_{Null}(\text{summary})}$$

It turns out that the p-value is in the form of a likelihood. The assumed hypothesis is the Null.

$$\text{p-value} = L_{Null}(??)$$

The $??$ has been put in $L_{Null}(??)$ to indicate that the quantity does not exactly refer to the actual data. Instead, for NHT and NP, the $??$ should be replaced by "the summary *or more extreme*." For simplicity, let's refer to this as $\geq \text{summary}$, with the p-value being

$$\text{p-value} = L_{Null}(\geq \text{summary})$$

For NP, we can also refer to another likelihood: $L_{Alt}(\geq \text{summary})$. The NHT/NP analog to the likelihood ratio is

$$\frac{L_{Alt}(\geq \text{summary})}{L_{Null}(\geq \text{summary})} = \frac{L_{Alt}(\geq \text{summary})}{\text{p-value}} \approx \frac{0.5}{\text{p-value}} .$$

Deeks, J. J., & Altman, D. G. (2004). "Statistics Notes: Diagnostic tests 4: Likelihood ratios." *British Medical Journal* 329(7458) 168-169. [link](#)

In NP, it would be straightforward to calculate $L_{Alt}(\geq summary)$. The calculation would be just like how power is calculated: many trials of generating data from the simulation and summarizing it. For the power, NP summarizes the trial by checking whether the Null is rejected. To find $L_{Alt}(\geq summary)$ summarize the trial by comparing it to the value stated for the Alternative. Typically this will be a number near 0.5. (In the hypothetical world where the Alternative is true, a model coefficient is about equally likely to be greater or less than the value set for the Alternative.)

To draw an inference in favor of the Alternative hypothesis, we want the likelihood ratio to be *large*, say 10 or higher. This can happen if the p-value is small. In both NHT and NP, a standard threshold is $p < 0.05$. Plugging $p = 0.05$ into the NHT/NP analog to the likelihood ratio gives $0.5/0.05 = 10$.

Thus, the p-value in NHT and NP is closely related in form to a likelihood ratio, with the standard cutoff of $p < 0.05$ corresponding to a likelihood ratio of 10.

An NHT is always straightforward to carry out, since the form of the Null doesn't involve any knowledge of the area of application. NP forces the researcher to state an Alternative hypothesis and have a way to simulate it (either with random number generators or, in some cases, with algebra). The Alternative is a stake in the ground, a statement of the effect size that would be interesting to the researchers. NHT lacks this statement. But in either NHT or NP, the p-value translates to an approximate odds ratio.

In engineering-like disciplines, it's often possible to make a reasonable statement about the prior odds. In basic research, the situation is sketchier. All three forms of hypothetical reasoning—Bayes, NP, and NHT—produce something very much like a likelihood ratio, with a ratio of 10 corresponding to a p-value of 0.05.

i The textbook version of the alternative hypothesis

Almost all introductory textbook cover hypothesis testing. Formally, they cover the NP style, in that they bring an

Alternative hypothesis into the discussion. But there is a serious shortcoming. To state a meaningful Alternative requires knowledge of the field of study, but textbooks prefer to make mathematical discussions, not field-specific ones. Perhaps this reflects the background of many introductory statistics instructors: mathematics.

As a substitute for a genuine, field-specific Alternative, it's conventional to offer an "anything but the Null" Alternative. For instance, if the Null is that the relevant model coefficient is zero, the Alternative is stated as "the coefficient is non-zero." This is regrettable in two ways. First, it misleads students into thinking that an Alternative hypothesis in science is something mathematical rather than field-specific. Second, it's not possible to calculate a power when the Alternative is "anything but the Null."

Also regrettable is the attempt made by such textbooks to create a role for the Alternative other than the calculation of power. After all, why would one mention an Alternative if it has nothing to do with the calculations? So textbooks have created an alternative to the anything-but-the-Null Alternative. This is that the Alternative is "anything greater than the Null." In other words, the made up, pseudo-useful Alternative amounts to "a model coefficient greater than zero." This sort of Alternative translates easily into the corresponding p-value calculation. Take the p-value from the "anything but the Null" situation and divide it by two.

Giving researchers a license to divide at whim their p-values by two distorts the meaning of a p-value. Better to report a real p-value: no division by two.