BAYESIAN ANALYSIS WITH EXCEL AND R

CONRAD CARLBERG

Bayesian Analysis with Excel and R

Conrad G. Carlberg

Contents at a Glance

Downloadable Bonus Content

Excel Worksheets

Book: *Statistical Analysis: Microsoft Excel 2016* (PDF)

To access bonus materials, please register your book at informit.com/register and enter ISBN 9780137580989.

Bayesian Analysis with Excel and R

ISBN-13: 978-0-13-758098-9
ISBN-10: 0-13-758098-3

Library of Congress Control Number: 2022945423

ScoutAutomatedPrintCode

Editor-in-Chief
Mark Taub

Acquisitions Editor
Debra Williams Cauley

Development Editor
Chris Zahn

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Indexer
Timothy Wright

Proofreader
Donna E. Mulder

Technical Editors
Nick Cohron
Regina R. Monaco

Publishing Coordinator
Cindy Teeters

Cover Designer
Chuti Prasertsith

Compositor
CodeMantra

Credits

Cover image: ImageFlow/Shutterstock

Figure 2-1, Figure 2-2, Figure 2-4, Figure 2-6, Figure 3-1 through Figure 3-13, Figure 4-1 through Figure 4-12, Figure 6-2 through Figure 6-4, Figure 7-1 through Figure 7-5, Figure 7-8, Figure 8-1, Figure 8-3, Figure 8-4, Figure 8-6: Microsoft Corporation

Figure 8-5, Figure 8-7: The R Foundation

Contents

Downloadable Bonus Content

Excel Worksheets

Book: *Statistical Analysis: Microsoft Excel 2016* (PDF)

To access bonus materials, please register your book at informit.com/register and enter ISBN 9780137580989.

About the Author

Conrad Carlberg is a nationally recognized expert on quantitative analysis, data analysis, and management applications such as Microsoft Excel, SAS, and Oracle. He holds a Ph.D. in statistics from the University of Colorado and is a many-time recipient of Microsoft's Excel MVP designation. He is the author of many books, including *Business Analysis with Microsoft Excel*, Fifth Edition, *Statistical Analysis: Microsoft Excel 2016*, *Regression Analysis Microsoft Excel*, and *R for Microsoft Excel Users*.

Carlberg is a Southern California native. After college he moved to Colorado, where he worked for a succession of startups and attended graduate school. He spent two years in the Middle East, teaching computer science and dodging surly camels. After finishing graduate school, Carlberg worked at US West (a Baby Bell) in product management and at Motorola.

In 1995 he started a small consulting business (www.conradcarlberg.com), which provides design and analysis services to companies that want to guide their business decisions by means of quantitative analysis—approaches that today we group under the term "analytics." He enjoys writing about those techniques and, in particular, how to carry them out using the world's most popular numeric analysis application, Microsoft Excel.

Preface

This book has several aspects that I want to let you know about up front. If you're already comfortable with terminology and concepts such as Hamiltonian Monte Carlo sampling, conjugate pairs, and posterior distributions, then this book is probably not for you. You already know a lot about those topics, and if you need more you know where to find it.

On the other hand, if you don't feel quite at home with the purpose of random samples, R's user interface, and why you might want to work with mean-corrected instead of with raw values, then it's just possible that this book offers something that you might want to know about. Both this book and I assume that you have some background in statistical analysis—say, at the introductory college level, where you can expect to study some probability theory and how it applies to the assessment of sample means, variances, and correlations. Particularly if you have studied these problems in the past, you will be better placed to understand how Bayesian analysis differs from traditional approaches, and how it works out in the context of the functions and packages found in R. And if you feel as though you could use some refresher work in traditional statistical analysis, Pearson is making available to you for download an e-book titled *Statistical Analysis: Microsoft Excel 2016*. You'll find details on obtaining that book at the end of this Preface.

You're experienced. You probably have something close to the background in Bayesian analysis that I had in mind when I laid out the topics that I wanted this book to cover. It seemed to me that the world already has plenty of books about statistics and experimental methodology: one more isn't going to help much. Something similar can be said about using syntax and diction that R recognizes: we already have as many elementary to intermediate texts on R as we need.

What we did need, I thought, was a source of information that connected the simplistic capabilities of VBA (the programming language historically offered by Microsoft Excel to give the user more control over the application) with the more sophisticated capabilities of programming languages such as R and C.

Similarly, we were missing information about three basic types of sampling that range from the simplistic, univariate sort of categorical analysis that you find in undergraduate texts to the complex sampling methods used by techniques such as quadratic approximation and Markov Chain Monte Carlo (MCMC). Richard McElreath has written, and has supplied to R, helper functions that ease the task of designing, writing, and installing the code that does the heavy lifting for you.

I have done what I can in this book to leverage the Excel skills that you have already developed in the areas of managing functions, handling data, and designing graphs and plots. The point will come that you see that Excel too handles the necessary tools of calculus in the form of function arguments—albeit more slowly and awkwardly. Shortly thereafter you'll see how the three fundamental approaches to building posterior distributions by sampling are in fact wonderfully creative solutions to the same problem.

Now let's see how I propose to get us there.

Chapter 1: Bayesian Analysis and R: An Overview

When I first approached Pearson about writing this book, I came away from the discussions just a little discouraged. The editors and their advisors were polite and really good at listening, but I didn't think that I heard much in the way of encouragement. In particular, they wanted to know why I would want to write this book.

Good question. I had several reasons in mind, but it wasn't easy to articulate them. Still, I did so, and apparently I did so successfully because, well, look at what you're holding. And those reasons made sense as a place to start out, but I'll keep it to the first two that occurred to me:

- Why would you want to read it? There are several reasons, but if you are like most of us you use Microsoft Excel for most numeric purposes, even though Excel was designed as a general-purpose calculation engine. You might have stayed away from Bayesian analysis because you heard that Excel is comparatively slow. And you're right: because of both software problems and hardware issues, there was a time when you had to wait and wait for a solution to the problem that you posed to Bayesian software. No longer. Now you can get an answer in a reasonable length of time, and without making assumptions that you don't feel quite comfortable with.
- People I work with were using familiar words in unfamiliar ways. They were using terms like *prior*, *likelihood*, and *parameter* in contexts that they did not seem to fit. I wanted to find out more about what they were saying. But I needed a starting point, and because I was quite familiar with Excel's numeric capabilities, I decided to work from the platform of Excel and toward a platform based on R. It's true that Excel is comparatively slow and doesn't have many functions that you would like to have in a Bayesian-oriented platform. But for certain problems, Excel works great and returns accurate results in a short timeframe. Fine; I can work from there.

That's what's going on in Chapter 1. Let's move ahead.

Chapter 2: Generating Posterior Distributions with the Binomial Distribution

The basic idea behind a Bayesian analysis is to create a posterior distribution that informs you about the parameters that bring about the results of the simulation. You do not want to start a sequence with one family of distributions and then try to finish the sequence in another family, so you should aim for a situation in which the prior and the likelihood are from the same family.

That, of course, implies that you select the distributional family from which the product will stem. You have several families from which to choose, but your choice will almost inevitably depend on the specific questions that you want to answer, which in turn depend on the nature of the data that you want to analyze.

One basic family of distributions is the binomial distribution. The term *binomial* itself implies the nature of a binomial distribution: two names, such as win and loss, buys and doesn't buy, survives and fails to survive, and so on. Consider your lifetime experience with coins. You have almost surely come to expect that when you pull a coin at random from your pocket and flip it, the probability is 50% that it will come up heads and 50% that it will come up tails. That's a binomial distribution: two names, two outcomes, two results.

The distinctive feature of a binomial distribution is that its values are discrete rather than continuous. When you flip the coin, you do not anticipate that the flip could come up with any of an infinite number of results. You anticipate two and only two outcomes, heads and tails.

This can be a very different situation from that of a person's height or weight. Then, each measurement is just one of an *infinite* number of possible heights or weights. The beta distribution, discussed in Chapter 3, is an example of a continuous distribution as distinct from a discrete one, such as the binomial. When you set up your analysis using R, for example, you can specify that a given parameter should be distributed as binomial, or any of R's distributional families. This flexibility is one characteristic that makes R's structure, and its design, so useful in Bayesian analysis.

Right here's a good spot to stress that it's important to specify the distributional characteristics of the parameters you use in an analysis, but don't let them blind you to other aspects—aspects that you might well ignore if you were to ignore all the good reasons for adding Bayes to your toolkit.

It's all too easy to forget that one of the key assumptions underlying a binomial test is that any two tests in your experiment are independent of one another. Suppose that you are studying the distribution of political party membership; one of the questions you ask is therefore which party, if any, a respondent belongs to.

To make a valid inference regarding the probability of a participant's response, you must be sure that the response is independent of any other response in your survey. So, the value of George's response must be unaffected by the value of Ellen's response. If that is the case, you're able to add and subtract subtotals directly (for example, to derive cumulative totals) without having to adjust for some probably unknowable dependency in the data.

Chapter 2 discusses this sort of concern in greater detail.

Chapter 3: Understanding the Beta Distribution

The principal difference between the binomial and the beta distribution is the degree of granularity with which variables are measured. Both distributions show how numeric variables are distributed across a span of values, much like the normal curve shows how a y-variable is distributed across a range of x-values.

But a variable that follows a beta distribution does so in a continuous rather than an interrupted fashion. The heads and tails left by coin flips follow a binomial pattern. Sorted by their actual values (heads, tails on a coin; 1, 2, 3,..., 6 on a die), the values that you see are not distributed continuously but discretely. We do not act as though a third of a head is a legitimate coin flip value, any more than we do that 2 1/2 is a legitimate value for the roll of a die.

But both those values would be legitimate if the variable, instead of being a coin flip or the roll of dice, were a plant's weight or height. Weight and height are both legitimately continuous variables, and each can take on an infinite number of values. That's the distinction between the distributions: if a distribution can take on any number of numeric values it's a beta, whereas a binomial distribution is limited typically to a much smaller number of values, such as 2 for a coin flip, 11 for a dice roll, and 2 if an item is judged defective or acceptable in a quality control context.

Both R and Excel have functions used to explore and manipulate the binomial *and* the beta distributions. It's useful to keep in mind that there are times when it's more convenient and just as quick to use Excel and VBA for generating frequency distributions as it is to use R. Chapter 4 has more to say about this matter.

Keep in mind that both Bayesian and frequentist approaches often return results that are either very close to one another (due to rounding errors induced by nearly all applications of calculus) or identical.

Chapter 4: Grid Approximation and the Beta Distribution

At this point, the discussion has centered on frequency distributions, both discrete (binomial) and continuous (beta). It moves now to the use of approximation techniques with frequency distributions.

Bayesian methods depend on approximations of distributions. We can, literally by fiat, declare that there exists a frequency distribution that is defined by its location (its mean) and its spread (variance or standard deviation). We can pass those attributes—the mean and the variance—to software that with adequate speed and efficiency builds the distribution we're after, with the required location and spread.

VBA can do that. We can use VBA to structure an array of values that, when populated with enough values, looks and behaves like a beta distribution or a binomial distribution or a normal distribution, or any other recognizable distribution of data. So how is it that VBA has acquired a reputation for slow and clumsy code?

An important part of the answer is that VBA is only partly compiled at runtime. It's an interpreted language, which means the same code must be compiled repeatedly, again slowing matters down. Furthermore, VBA is not optimized for array management; newer languages such as Python manage arrays much more effectively by converting multi-row, multi-column arrays to single-row vectors, which some insist speeds up processing dramatically.

This chapter demonstrates how a posterior distribution changes in response to the act of modifying the likelihood. It's a useful place to provide that demonstration because it shows how the grid approximation technique results in simple modifications to the frequency distribution's structure—and the rationale for terming it a grid approximation.

Chapter 5: Grid Approximation with Multiple Parameters

Issues such as the speed with which hardware executes instructions, the efficiency with which code fills a distribution with simulated data, whether the computer in use is a vector machine, and other considerations are unquestionably important to the speed with which an analysis runs. But generally, a more important issue is the number of parameters and quantiles you ask the analysis to deal with.

When you expect to analyze only one parameter, even if it has as many as seven or eight meaningful levels, you could push likelihoods through a Bayesian analysis and have plenty of time left over. It might seem obvious, but as soon as you add a parameter to the design, you aren't just adding but multiplying design cells.

Start with six levels of a parameter, which even BASIC code could analyze before you finish your coffee. Now add another parameter that has five levels, and you're not simulating record counts for just 6 + 5 = 11, but 6 * 5 = 30 design cells. You might never have to put a simulated record in one of those multiple parameter cells, depending on matters such as size of the standard deviation, but your grid approximation code will need to attend to every one of them, when a quadratic approximation or a Markov Chain Monte Carlo instead could go flying past them.

Chapter 5 will give you a sense of how much time is spent needlessly dealing with design cells just because grid approximation requires that they be there.

Chapter 6: Regression Using Bayesian Methods

Most of us are familiar with the regression approach to solving problems that are presented in the context of the general linear model. We're familiar, even comfortable, with a page or two of printed output that includes figures such as

- Traditional correlation coefficients and regression constants
- Regression summaries such as R^2
- Inferential statistics such as F ratios and standard errors of estimate

This chapter begins to tie together concepts and techniques that in previous chapters have remained largely isolated from one another. In particular, difficulties imposed by the grid approximation method can be painful, especially when multiple predictor variables are involved. There are various reasons for this, particularly when the experimenter wants to assess the simultaneous effect of multiple variables. If one can't evaluate the combined effects of water and fertilization on a crop, it's at least that difficult to evaluate their separate effects. But just when the experiment becomes really interesting due to the addition of variables, the analysis starts to groan under the weight of that addition.

Chapter 6 starts to replace the use of grid approximation with that of an R function named `quap`, or *quadratic approximation*. The reason that so much ink is spent on discussing grid approximation is that it forms the basis for more sophisticated techniques such as speeding up the structuring and populating of posterior distributions, faster methods of approximating posterior distributions than grid approximation. Furthermore, the extra speed of quadratic approximation enables us to use multiple predictor variables simultaneously—and without that capability, grid approximation falls short.

Like grid approximation, `quap` approximates the posterior distribution density of the parameters we want to know about. To do so, the software uses a quadratic function, so we term it a *quadratic approximation*.

Chapter 7: Handling Nominal Variables

Often you'll have a variable whose values have been saved as numeric values but that should be analyzed as though the numeric values were in fact text values. This chapter discusses ways to handle them so that text values are managed as though they were in fact numeric. The opposite approach, in which numeric values are handled as though they were text, also exists. Dummy coding and index variables are discussed here, as is the use of the `quap` function to make conversion more straightforward.

Chapter 8: MCMC Sampling Methods

The final chapter in this book moves to a technique that for several years has been the gold standard for Bayesian sampling: Markov Chain Monte Carlo, or MCMC. Other and older approaches tend to get stuck in particular thickets of the posterior distribution, often because of autocorrelation built into the sampling logic. But MCMC manages to avoid that trap, and to simultaneously maintain its execution speed.

That characteristic—maintaining speed while increasing design complexity—is what allows MCMC to simulate large posterior distributions without slowing down unduly. In turn, that positions you to code predictor variables so that they behave in the best ways of both continuous and discrete variables, and in ways that ease their interpretation when it comes time to evaluate the results.

Who Are Those Guys?

Right about now you might well be asking yourself, "Why should I read this? What kind of statistical analysis is the author pushing, Bayesian or frequentist?" The best I can do by way of an answer to those questions is to tell you a little bit about my education and experience.

I took my first course in statistical analysis at a small, well-regarded liberal arts college in the Midwest. It was a miserable experience, and that might well have been due to the fact that it was taught out of the psychology department. I still have the textbook that was used in that course, and in the fashion of the day (this was in the 1970s) it told its readers what to do with a bunch of numbers and almost nothing about why it made sense to do that.

Nevertheless, I finished that course in statistics and took a couple more just for good measure. They were a bit better than the one I took from the psych department. After my undergrad degree I enrolled in grad school and started out under a professor who I knew I wanted to study with. He was a frequentist and was first author on a basic statistics text that broke new ground: It explained to the reader *why* it was desirable to include certain calculations in a given statistical analysis.

His book, as well as his classes, stressed the rationale for the kinds of analysis that were *de rigueur* during the late 1970s. You followed up carefully designed experiments with tests of statistical significance. You used t-tests (Gossett) to calculate that statistical significance with two groups. You used the analysis of variance (Fisher) to calculate that statistical significance with more than two groups. You used the product-moment correlation coefficient (Pearson) to measure the strength of the relationship between two ratio variables. You used factor analysis and multivariate analysis of variance (Green; Wilks) to reduce a data overload down to a few manageable factors and to test differences between groups measured on more than one outcome variable. You used multiple comparisons (Tukey) to pinpoint the location of statistically significant differences between group means.

Every one of these techniques belongs in the frequentist toolkit. I used each of them, in combination with an ad hoc technique called exponential smoothing, at a large telecommunications firm during the 1980s. We were able to reduce a bloated resale inventory from more than $14 million to less than $7 million in under a year, without write downs. (This was back when $14 million was a lot of money.)

So I have every possible reason in my educational and professional background to be grateful for the tools that frequentist statistics has offered me. And I am grateful. But...

I start to feel uneasy every time I read about a finding by the Reproducibility Project that contradicts the finding of another published study. That can happen, and does, for reasons that range from mis-specifying a design so that it treats a random factor as fixed, to something as commonplace as p-hacking.

I worry when I find that someone has applied Welch's correction or something similar in a situation where sample sizes are unequal and so are population variances: the Behrens-Fisher problem. There's something wrong with a scientific approach that allows such a problem to exist so long without a satisfactory solution.

The analysis of variance (ANOVA) has the principal purpose of determining whether any two population means are equal in an experiment consisting of at least three groups. There are at least six distinct procedures, collectively called *multiple comparisons*, intended to pinpoint which groups are responsible for a significant ANOVA outcome. One of them requires a standardized score difference of 7.5 for two means to be considered significantly different at the .05 level, and another requires a difference of 15. It is true that our choice of multiple comparison procedure differs according to the situation under which the data were collected and given the inferences we want to make. Still, we should be able to come up with methods that agree more closely than do the Scheffé and planned orthogonal contrasts.

Then there's multicollinearity, an issue that crops up in regression analysis. It can pose other problems for statistical analysis, and I touch on them briefly in Chapter 6. There are plenty of other similar issues, I promise you that. Some are solved by recourse to Bayesian methods, and some just aren't. My point is that I have no special reason to prefer frequentist methods to Bayesian or vice versa. I have tried in this book to avoid any bias toward frequentist methods, and I hope and think that I have succeeded.

Where to Find It

I suspect that you are someone who uses the R application with some level of experience. As such, I assume that you have probably installed R software on your computer, following the instructions provided by the CRAN website (cran.r-project.org). When you do so, quite a bit of default code and simple straightforward functions such as `max` and `read.csv` are automatically installed on your computer.

Other code takes the form of *packages*, and here there's nothing automatic about the installation. If you want to install a package, and you almost certainly will, the standard procedure is to identify a mirror site from the drop-down list that appears when you select the *Set CRAN mirror* item in R's *Packages* menu. After you have identified a mirror site, you can select one of the roughly 15,000 packages that CRAN offers in a drop-down.

Even though the packages are presented in alphabetical order, selecting one of 15,000 is more than most users look forward to doing. So you'll be glad to know that you do not need to go through that tedious process more than once in order to install the code discussed in this book.

NOTE

The R application, without any special assistance, recognizes most of the code discussed in this book. There are a few functions (notably, `quap` and `ulam`) that require you to install a package named *rethinking*. You do not use R's *Packages* menu to install *rethinking*. See Appendix A for detailed instructions on installing the *rethinking* package on a Windows machine.

And speaking of platforms, at the time that I'm writing this book, no provision is made to install *rethinking* on a Mac. For the time being, as far as we know, there is no version of *rethinking* that is compatible with the Mac.

At this point you should be good to go. Chapter 1, "Bayesian Analysis and R: An Overview," is coming right up.

Bonus Material

To access the downloadable worksheets and PDF of the book *Statistical Analysis: Microsoft Excel 2016* please

1. Got to informit.com/register.
2. Enter the ISBN 9780137580989.
3. Answer the proof of purchase challenge questions.
4. Click on the "Access Bonus Content" link in the Registered Products section of your account page, to be taken to the page where your downloadable content is available.

Regression Using Bayesian Methods

6

IN THIS CHAPTER

Statisticians use the term *regression* pretty loosely.

At its simplest, the term refers to the average of the products of the corresponding z-scores—a.k.a., the Pearson correlation coefficient. At its oldest, the term refers to the tendency of sons' heights to regress toward the mean of their fathers' heights. When applied to categories such as method of transportation, brand of car, or the presence of a defect in a manufactured product, it's usually called *logistic regression*. When particular types of coding schemes are applied to independent variables, which are manipulated by the researcher and not merely observed, it's often termed the *general linear model*. And in a true experimental design, the purpose of regression analysis is not simply to predict but, more typically, to explain. Depending on the context, then, *regression* can imply a variety of statistical and methodological purposes.

Regression à la Bayes

So it shouldn't be at all surprising that the Bayesian approach to regression looks very different from the frequentist approach. Suppose that you want to better understand the relationship between the amount of fat consumed by adults during a year and the amount of low density lipoproteins (LDL) cholesterol found in blood samples from similar adults at the year's end.

Assuming that you have no insurmountable difficulties with the acquisition of good data, you're set up to quantify the relationship between LDL and fat consumption. Just about any application designed to return numeric analyses will provide you with the summary statistics you're after:

- **Correlation coefficient.** A number between –1.0 and +1.0 that expresses the direction and the strength of relationship between two variables. A correlation of 1.0 describes a perfect and positive relationship, such as height in inches with height in centimeters. A correlation of –1.0 describes a perfect negative relationship. An example of a perfect negative relationship is the correlation between the number of correct answers on a test with the number of incorrect answers on that same test.
- **R^2.** The square of the correlation between a predicted variable and one or more predictor variables. I believe that usage calls for the abbreviation to be capitalized (R^2) with more than one predictor, and lowercase (r^2) with just one predictor.
- **Slope or regression coefficient.** The gradient of a line that shows where x-values, such as golf score, connect with predicted y-values, such as years playing golf (see Figure 6.1). You may recall this concept as taught in middle school as "the rise over the run."

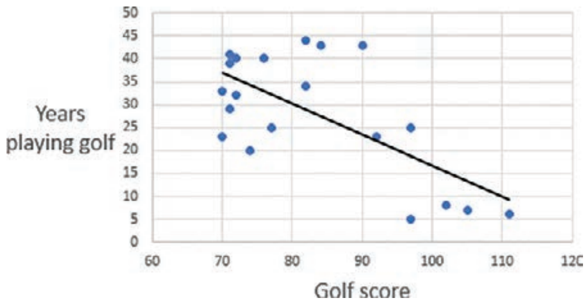


Figure 6.1
A regression line slopes up when the correlation is positive, such as calories consumed and weight. It slopes down, as here, when the correlation is negative, such as number of years playing golf and average golf score.

All of the just-named statistics—and more—are returned by any credible statistics package, certainly various packages supplied by R and even the venerable BMD and Lotus 1-2-3. What distinguishes the Bayesian approach to regression analysis is that it does not maximize or minimize the value of some function such as R^2 to arrive at a solution; that is the goal of frequentist approaches. Bayesian methods seek to maximize the probabilities of particular outcomes.

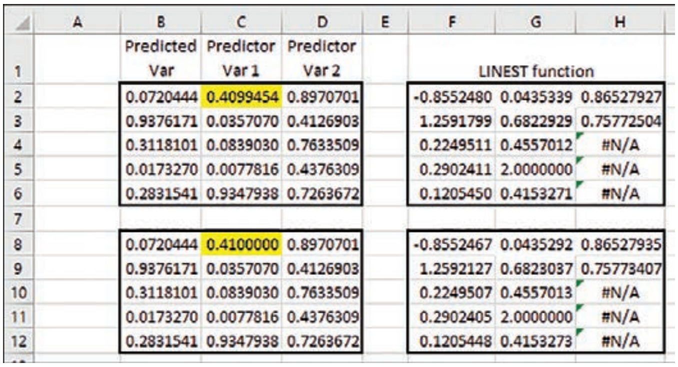
One of the names for frequentist regression is *least squares analysis*. The frequentist algorithms calculate the combination of predictors that minimizes the squared deviations of the observed predictor variable's values from the predicted values. The values of the remaining statistics flow from that finding: R^2, the F ratio, the standard errors of the intercept and the coefficients, the standard error of estimate, and so on.

The least squares approach to regression analysis works with one, two, three, or more predictor variables. Regression's job is to combine those predictors to create a new variable. They are combined by multiplying each predictor by its own coefficient, then summing the products of the predictors and their coefficients. Regression does the heavy lifting when it optimizes those coefficients.

Then, regression calculates the correlation between, on one hand, the observed or outcome variable, and on the other hand, the combined predictor variables. Make one tiny change to the value of one of the predictor variables—say, change it from 5.00 to 5.01—and typically all the other variables change in response: their regression coefficients, the standard errors of the regression coefficients, R^2, the F ratio, the sums of squares—anything except the degrees of freedom.

Figure 6.2 shows an example.

Figure 6.2
The values in the range B2:D6 are identical to those in B8:D12 with one exception: the value in C2 has been changed from 0.4099 to 0.4100 in cell C8. But the regression statistics in F2:H6 are all different from those in F8:H12, with the exception of the degrees of freedom regression.



	A	B	C	D	E	F	G	H
1		Predicted Var	Predictor Var 1	Predictor Var 2		LINEST function		
2		0.0720444	0.4099454	0.8970701		-0.8552480	0.0435339	0.86527927
3		0.9376171	0.0357070	0.4126903		1.2591799	0.6822929	0.75772504
4		0.3118101	0.0839030	0.7633509		0.2249511	0.4557012	#N/A
5		0.0173270	0.0077816	0.4376309		0.2902411	2.0000000	#N/A
6		0.2831541	0.9347938	0.7263672		0.1205450	0.4153271	#N/A
7								
8		0.0720444	0.4100000	0.8970701		-0.8552467	0.0435292	0.86527935
9		0.9376171	0.0357070	0.4126903		1.2592127	0.6823037	0.75773407
10		0.3118101	0.0839030	0.7633509		0.2249507	0.4557013	#N/A
11		0.0173270	0.0077816	0.4376309		0.2902405	2.0000000	#N/A
12		0.2831541	0.9347938	0.7263672		0.1205448	0.4153273	#N/A

Sample Regression Analysis

To lay the groundwork for a comparison of Bayesian regression analysis with traditional least squares, Figure 6.2 shows the basics of a very small analysis, rendered in Excel. It includes

- Values in B2:D6, which are used as inputs to Excel's `LINEST` function.
- Values in the range C2:D6, which contains two predictor variables in columns C and D.
- Values in cells B2:B6, which contain a predicted variable.
- The `LINEST` function, in the range F2:H6, which contains and displays the results of the function. For example, the contents of each cell in F2:H6 are computed with the dynamic formula that's repeated here:

```
=LINEST(B2:B6,C2:D6,,TRUE)
```

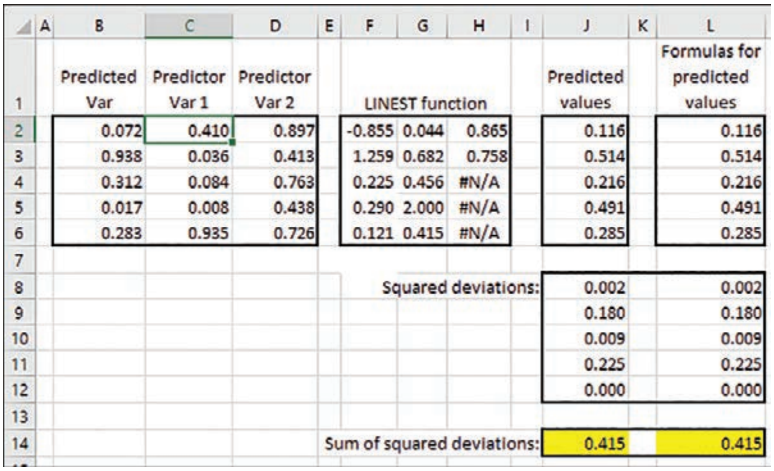
NOTE

That formula is known as a *dynamic* array formula in more recent versions of Excel, released in the 2021 timeframe. Earlier versions of Excel use a *legacy* array formula, which requires that the user begin by selecting the entire range to be occupied by the array formula, and to enter the formula via Ctrl+Shift+Enter rather than via Enter alone. One of the results of the changes made to the way that Excel handles formulas is that you can now enter a `LINEST` formula without either having to begin by selecting the full target range or having to enter the formula via Ctrl+Shift+Enter. If you prefer, you can start by selecting a single cell and end with Enter instead of Ctrl+Shift+Enter. The legacy array formula appears on the worksheet surrounded by curly braces. The dynamic array formula appears on the worksheet without those braces.

- After entering the formula in F2:H6 of Figure 6.2, I copied and saved it as the result *values* in F8:H12. That is because I want you to be able to use Excel's Solver, or to change the regression coefficients manually, so you can compare the results of that change with the original results. By doing so, you can demonstrate for yourself what happens when you try to maximize regression's accuracy by adjusting the returned coefficients and intercept. (You cannot change just part of an array formula; it's all or nothing at all. But if you have saved the results of `LINEST` as values, you are free to change any of those values as you please.)
- The regression equation's predicted value for the first of the five records is shown in cell L2 of Figure 6.3. It is calculated with this equation:

```
=$H$2+$G$2*C2+$F$2*D2
```

which is then copied and pasted into L3:L6 of Figure 6.3.



	A	B	C	D	E	F	G	H	I	J	K	L
1		Predicted Var	Predictor Var 1	Predictor Var 2		LINEST function				Predicted values		Formulas for predicted values
2		0.072	0.410	0.897		-0.855	0.044	0.865		0.116		0.116
3		0.938	0.036	0.413		1.259	0.682	0.758		0.514		0.514
4		0.312	0.084	0.763		0.225	0.456	#N/A		0.216		0.216
5		0.017	0.008	0.438		0.290	2.000	#N/A		0.491		0.491
6		0.283	0.935	0.726		0.121	0.415	#N/A		0.285		0.285
7												
8								Squared deviations:		0.002		0.002
9										0.180		0.180
10										0.009		0.009
11										0.225		0.225
12										0.000		0.000
13												
14								Sum of squared deviations:		0.415		0.415

Figure 6.3
A slight change in the input data or in a regression coefficient can result in a dramatic change in the results.

]

TIP You can get the same effect using Excel's `TREND` function. I used the `LINEST` approach because I wanted to show the steps explicitly.

- I also entered the formulas in L2:L6 *as values* in J2:J6 by first copying the formulas, and then pasting them into J2:J6, choosing one of the Paste Values options.
- Finally, I entered formulas for the sum of the squared deviations in J8:J12 and L8:L12, and their sums in J14:L14.

Open the Excel workbook for this chapter and activate the worksheet named *Fig 6.3*. Verify that the sums of the squared deviations are both 0.415.

Now, change the value of one or both the regression coefficients in cells F2, G2, or H2. Make your entry a numeric value. Notice that the values displayed in cells J14 and L14 no longer equal one another. While you're at it, you might note that the value in L14 is now larger than the value shown in cell J14.

The value in cell J14 is unchanged from its original value. That's why I saved the results of the `LINEST` function in F2:H6—so that it would be unaffected by your selection of a different value for the regression coefficient in cell F2, G2, or H2. Either way, the sum of the squared deviations in L14 has increased above its value when the `LINEST` results were undisturbed. And that means the regression equation is not doing as accurate a job of forecasting outcomes as when you left its coefficients alone.

What's the point of all this? It's that traditional, least squares techniques for regression analysis do not necessarily tell you what you need or want to know about the relationship between an outcome variable and one or more predictor variables. Of course, you don't want to ignore the traditional point estimate that's returned by the traditional R^2 calculations, but neither should you ignore the results of calculations that return an R^2—and associated statistics—that don't quite meet or exceed the criterion of maximized R^2.

To keep some flexibility in your analytic tools, it's a good idea to view the results of a regression analysis through both a frequentist and a Bayesian lens. I've already discussed some of the issues surrounding the frequentist approach in this chapter—in particular, the worksheet function `LINEST`—so let's now take a look first at regression methods that rely heavily on matrix algebra, and then on one alternative from the Bayesian toolbox, R's `quap` function.

Matrix Algebra Methods

Suppose that you took regression's job as your own, in a situation that called for you to predict the value of an outcome variable given knowledge of three predictor variables, named *Var 1*, *Var 2*, and *Var 3*. You decide to declare, by fiat, that each predictor variable should be multiplied by a regression coefficient of 1. Then the regression equation would look like this:

```
(1 * Var 1) + (1 * Var 2) + (1 * Var 3) = Predicted variable
```

There is nothing to prevent you from doing that, but it's wildly unlikely that the regression coefficients you chose, a sequence of 1s, will work better than any other coefficients that you might choose. Nevertheless, you will have completed a basic requirement of regression analysis: a sequence of variables, each multiplied by its regression coefficient and added together to create a new, composite variable.

NOTE This is the meaning of the term *multiple* regression. You have multiple predictor variables and one outcome variable. Other kinds of analysis, such as multivariate ANOVA, employ multiple outcome variables. But in the case of regression, the word *multiple* belongs to the predictor variables, not the predicted variable. This leads to confusion in many basic to intermediate statistics classes.

For years, statistical packages such as Systat, and even more generalized applications such as Excel, used matrix algebra to solve regression's normal equations. These processes failed to operate successfully when they were presented with data sets that involved severe multicollinearity. Multicollinearity comes about when two or more predictor variables in a regression equation are strongly or even perfectly correlated.

When this situation occurs, it can throw the results of the matrix algebra off course. Taking apart the matrix components of a multiple regression, you find that the process involves calculating the sums of squares and cross products matrix (SSCP). Then the inverse of the SSCP is calculated. If the values of one of the fields in the original data matrix is a linear function of another one of those fields, then the inverse of the SSCP cannot be calculated. (This is usually because the determinant of the SSCP is zero.)

This problem was known in the waning years of the previous century, but it went unfixed, largely because it took an unusual sequence of events for the problem to arise. Furthermore, the user who encountered the problem got an error warning, sometimes in the form of a lengthy text message, sometimes in the form such as Excel's #NUM! cell value. So an opportunity existed for the user to recognize that an infrequent error had occurred, and to fix it in the data file.

But users did not like knowing of a remaining problem, however unusual, in their software, so developers applied an approach called *QR decomposition* in place of the existing matrix algebra. It's the approach that you find in Excel and other numeric analysis packages even as late as this book's publication in 2022.

However, QR decomposition does not truly fix the multicollinearity problem, which is not a strictly either/or situation. When one field is a nearly perfect linear function of another, problems can arise with rounding errors, and those errors can reduce the accuracy of the analysis results.

Some software publishers have adopted the reasonable solution of displaying a zero instead of a calculated regression coefficient when QR decomposition detects the presence of multicollinearity. This has the effect—possibly useful, possibly disastrous—of eliminating the associated field from the regression equation. Depending on the nature of the linear function, the regression software might set both the regression coefficient and its standard error to zero.

For the time being, though, let's shift our attention to some of the critical elements of the `quap` function.

Understanding `quap`

R's `quap` function occupies a position between the simpler (but often awkward) grid approximation and the more sophisticated (but often murky) MCMC. We might as well begin with the function's name: `quap` is an abbreviation of *qu*adratic *ap*proximation. (The functionality is also referred to as *Laplace approximation*.)

Behind the scenes, the software makes an *approximation* of the posterior distribution density (the product of the priors and the observations) of the parameter we want to know about; for example, a regression coefficient in a multiple regression equation. To do so, the software uses a *quadratic* function; hence the term *quadratic approximation*.

The `quap` function is capable of returning a variety of analyses that support Bayesian methods. However, its principal purpose is to build a posterior distribution from samples that conform to requirements that you supply. These often include the location and spread of Gaussian distributions from which priors are assembled. Another purpose that the `quap` function serves is to define the relationships among the variables in your analysis.

Let's take a look at how those processes might support a `quap` function that supports the Bayesian version of simple (that is, single-predictor) regression analysis. We start with a little housekeeping:

```
library(rethinking)
setwd("C:/Users/Smith/Documents")
PropTaxes <- read.csv("Assessments.csv")
```

The `quap` function is part of the `rethinking` package, so begin by loading `rethinking`. You'll need to install `rethinking` first, if you haven't done so already.

You don't need to set the working directory by means of the `setwd` function if your data file is already stored there; otherwise, use `setwd` to point R in the right direction, or copy the file into the current working directory.

The third line of R code above assumes that your data is in a csv file named `Assessments.csv`, so read the data into R's workspace from that file and assign it the name `PropTaxes`. Keep in mind that the `read.csv` function results in a data frame, so you now have a data frame named `PropTaxes`. (Don't forget that names in R, including file names, are case sensitive.)

Here's the next line in the R code:

```
MeanValue <- mean(PropTaxes$Value)
```

This statement establishes a new variable named `MeanValue` from the variable named `Value`. It is the arithmetic mean of the variable for all the cases in the `PropTaxes` data frame. The code goes on to subtract that mean value from each observed value, changing the nature of the variable from a raw observation to a mean-corrected value. At that point, `MeanValue` is no longer an assessment measured in dollars but a deviation from the mean assessment, measured in dollars. There are some good analytic reasons to shift the meaning of the `Value` variable in this way, but the principal purpose here is to clarify the meaning of the resulting regression coefficient.

We'll take a look at that shortly. In the meantime, notice that when the `setwd` function creates the new data frame from the `Assessments.csv` data file, it attaches the `Value` field as a variable. You can address that variable directly by providing the data frame's name, followed by the dollar sign `$`, followed by the variable's name. For example:

```
PropTaxes$Value
```

Housekeeping's over, and now it's time to build the model for the analysis. The first step is to name the model, which here will have the name `AssessModel`. The specifications that follow the function name in the code will be used to structure `AssessModel`. Those specifications are assigned to the model by means of the assignment operator, which in R is indicated by the less-than symbol followed by a dash: `<-`. (Sometimes, although rarely, the equal sign is used instead of `<-`.)

```
AssessModel <- quap(
  alist( . . .
```

Here, the result returned by the `quap` function is saved to a new object (a model) named `AssessModel`. The model is made in the form of a list created by the `alist` function. The elements that belong to the list are formulas and as such might include references to variables and parameters that aren't yet ready for use. For example:

```
Tax ~ dnorm( mu , sigma ) ,
```

This is a *model formula*, and it can be used as a component of the list assembled by the `alist` function. A list created by `alist` has some important differences from a list that results from the `c` or the `list` function; for example, elements of the list are not necessarily evaluated immediately. In the prior example, the value of `Tax` can be read as dependent on the purpose of the `dnorm` function and the parameters `mu` and `sigma`. If we don't yet know what values to use for `mu` and `sigma`, we can't yet evaluate `dnorm` or its results. But no worries: we'll get around to evaluating them shortly.

So, that's the first component of the list. Here's where we left off:

```
Tax ~ dnorm( mu , sigma ) ,
```

That tilde operator is used frequently in `quap` formulas, and its effect can depend on the context. Here, it means roughly that `Tax` will be distributed as a normal curve with `mu` and `sigma` as its parameters. In English, `Tax` depends on the result returned by `dnorm` when it gets `mu` and `sigma` as its arguments.

As I just noted, the code doesn't have those values yet. While waiting for them, let's get a handle on `dnorm`. That's an abbreviation of *density normal*. It tells R to look in the normal curve and return the density (in this context, density means probability) when values have been assigned to both `mu` and `sigma`.

CASE STUDY: INVENTORYING TYPES OF DISTRIBUTION

R provides support for 17 types of distribution (plus several less common ones), including the beta, binomial, chi-squared, F, gamma, log-normal, Poisson, t, and uniform.

Each type of distribution can be accessed to return that distribution's probability, cumulative probability, quantiles, and random values. The first letter of the function denotes the type of information to return. The four letters used are *d, r, p, and q*.

So, for example:

- The letter *d*, prepended to *norm* to produce *dnorm*, returns the probability (density) of the normal distribution at a given x-value.
- Prepending *r* to *binom* returns random numbers from the binomial distribution via *rbinom*.
- The function *pf* returns the cumulative probability from the F distribution.
- The function *qlnorm* returns quantiles from the *log-normal* distribution.

Continuing the Code

The R statement that I was about to discuss before introducing the topic of R's distributional function syntax is

```
Tax ~ dnorm( mu , sigma ) ,
```

That statement establishes that `Tax` comprises the parameters `mu` and `sigma`, but we don't yet know how they are involved. For all we know, `Tax` could be the sum of `mu` and `sigma`, or their difference, or their ratio—it's just too soon to know. But we do know that you can specify the normal, Gaussian distribution with only two parameters:

- The mean of the distribution, usually termed *mu*. The mu parameter locates the distribution along the horizontal axis. So, the mean of a population's IQ scores might be 100; the mean of a population's HDL cholesterol score might be 65 mg/dl. It is the normal curve's central tendency.
- The standard deviation of the distribution, usually termed *sigma*. In a Gaussian distribution, about 34% of the cases fall between the mean and one sigma above the mean, and another below it; another 13.6% falls between one and two sigmas above (and another below) the mean; and 2.1% falls three sigmas above and below the mean. It's a measure of the distribution's spread: the width of the distribution, relative to its height.

Software that actually performs Bayesian statistical analysis needs some way of knowing what the underlying distributions look like, and the arguments to the quap function in R provide that capability. Because the Gaussian distribution requires so little information to structure—that is, the mean and the standard deviation—it's straightforward to code.

Furthermore, many topics of interest to all life forms follow the template of a standard normal distribution, and they do so intrinsically. Consequently it's not usually necessary to provide code that takes into account anomalous distributions, such as bimodal curves, highly skewed shapes, and fits that require some grappling.

A Full Example

Let's put some meat on these bones. Suppose that you're interested in the relationship between body weight and LDL cholesterol levels. You have a simple, straightforward hypothesis that, other things being equal, there is a direct relationship between a person's body weight and his or her LDL level. To take advantage of the tools that `quap` gives you, you'll need an `alist`, one that looks something like the following code:

```
library(rethinking)
adult.weight <- read.csv("Sample Weight Data.csv")
```

The `read.csv` statement attempts to open the file named `Sample Weight Data.csv` in the working directory. You can include the file's path in the argument to `read.csv`; if you handle it that way, keep in mind that R uses forward slashes, not back slashes, to delimit folder names in file addresses. Or, you could save the data file in what you know to be R's current working directory.

```
sample.mean.wt <- mean(adult.weight$Weight)
  ldl.model <- quap(
    alist(
        LDL ~ dnorm( mu , st.dev.wt ) ,
        mu <- alpha + beta *( adult.weight$Weight - sample.mean.wt ) ,
```

I have given these two variables in `mu`'s definition the names of `alpha` and `beta`, because that's how they are normally referred to in the literature on simple (i.e., not multiple: only one predictor) regression: `alpha` is the intercept and `beta` is the regression coefficient.

Now we need to establish the central tendency and the spread of the `alpha` and `beta` priors. We can tell `quap` that `alpha`, the equation's intercept, has a mean of 20 and a standard deviation of 20:

```
alpha ~ dnorm( 20 , 20 ) ,
```

and that `beta`, the regression coefficient, has a mean of 0 and a standard deviation of 1:

```
beta ~ dnorm( 0 , 1 ) ,
```

and that the standard deviation of body weight follows a uniform distribution with a mean of 0 and its own standard deviation of 50:

```
        st.dev.wt ~ dunif( 0 , 50 )
    ) , data = adult.weight)
precis(ldl.model)
```

And you'll need a set of observations that are stored in the csv file `Sample Weight Data.csv`. They are temporarily stored by R in the structure named `adult.weight`. Here are the first few observations in `adult.weight`. Note that the first row of the csv file contains field names. The `read.csv`'s header argument is set to `True` when the table's first row contains one fewer field name than the table's number of columns. (This is often true when the first column contains row numbers but is not the case here.)

Weight	LDL
165	37
118	48
114	61
117	55
108	33

And here are the results of running the code, shown in `precis` form:

> precis(ldl.model)

	mean	sd	5.50%	94.50%
alpha	58.27	1.63	55.6	60.8
beta	0.06	0.07	-0.03	0.2
st.dev.wt	11.53	1.16	9.68	13.38

You can compare R's results to Excel's by running `LINEST`. The `LINEST` function (entered normally in current Excel versions, by selecting a single cell and using Enter rather than Ctrl+Shift+Enter) is

```
=LINEST(B2:B51,A2:A51-AVERAGE(A2:A51),,TRUE)
```

Here are the results of running the Excel `LINEST` function:

0.064249	58.52
0.070996	1.655082
0.016775	11.7032
0.818952	48.0
112.1677	6574.312

The `LINEST` results require some mapping:

- The value in `LINEST`'s first row and rightmost column (here, 58.52) is always the equation's intercept. Notice that the `quap` model returns a value of 58.27 (second row, second column of the `precis` summary). The two values are quite close, and the difference is easily attributable to sampling error in the `quap` model.

- The value in `LINEST`'s first row and leftmost column (here, 0.064) is always the final regression coefficient. In this case, because we have called for one coefficient only, it is also the equation's first and only coefficient.
- The value in `LINEST`'s second row and rightmost column (here, 1.655) is the standard error of the intercept. It is always in that cell, of `LINEST`'s results, directly below the intercept. Its value is quite close to that returned by `precis` in its third column, second row, 1.63.
- The value in `LINEST`'s second row and leftmost column (here, 0.071) is the standard error of the regression coefficient. Values in the second row of `LINEST` results are always the standard error of the statistic in the same column, first row.
- The value in `LINEST`'s third row and rightmost column (here, 11.7) is the standard error of estimate, and it is quite close to the `precis` estimate of 11.53. Suppose that you took all the observations at a given value of the predictor and found the standard deviation of the difference between their actual and the predicted values on the predicted variable. That's the standard error of estimate, and it helps you decide whether the prediction equation is more accurate at some levels of the predictor than at others.

Compare the results of the regression analysis as returned by Excel with those returned by `quap` via `precis`. It's clear that where the two approaches return the same analyses (e.g., intercepts, coefficients, standard errors), the Bayesian approach and the frequentist approach are either identical or very nearly so.

And you can get those results without risking the slippery slope of multicollinearity. Which makes this a good point to go further into multiple regression.

Designing the Multiple Regression

Suppose that you have data on 50 cars, including each car's weight in pounds, mean speed at which it has been driven, and mean miles per gallon (MPG). You're interested in the effect that a car's weight and average speed have on the miles per gallon of fuel that the car achieves.

One way to approach the problem is with one analysis using Weight as the sole predictor variable and another using Speed as the sole predictor. You could choose the analysis that returns the greater R^2 value as the one to use in assessing a car's predicted MPG.

One problem with running and comparing the two analyses is that the two predictor variables, Speed and Weight, might not be independent of one another; that is, they might be correlated and therefore share variance. In that case, you can't tell how much of the shared variance is shared by Speed and MPG and how much is shared by Weight and MPG. But it's very likely that running two analyses and summing the R^2 values will double count some amount of the variance (because it's shared by the two predictor variables) and therefore mislead you as to the strength of the relationships.

Only in the limiting cases in which the predictor variables share no variance with one another (so they're independent) or in which they're perfectly correlated (so they share all their own variance) can you tell what's going on. Of course, that sort of complete independence or dependence appears only in samples handed out in stats class. (An exception occurs when regression is used in preference to the analysis of variance and the categorical predictor variables are designed to be independent of one another.)

Whether your primary interest is in the total variance in the outcome variable that's associated with variance in both the predictor variables, or the total amount that's shared with each of the predictors, you're going to need to arrange things to combine the predictors without double counting the variance shared with the outcome. Multiple regression does that for you, whether by means of matrix algebra or by means of QR decomposition, and I wouldn't have spent so much ink on the topic if Bayesian methods didn't do it too.

Arranging a Bayesian Multiple Regression

Earlier in this chapter I described how to provide arguments to a `quap` function that support a single-predictor regression. I'll review it briefly here. You supply these arguments:

- A variable that represents the outcome for each case, such as a car's MPG, usually the name of the outcome variable. For example:

  ```
  MPG <- dnorm ( mu, sigma)
  ```

  specifies that MPG's density is normally distributed (`dnorm`) with a mean of `mu` and a standard deviation of `sigma`. This outcome variable is usually input in a data frame along with the predictor (see below).
- A parameter, often but not necessarily termed `mu`, that represents the result of the regression equation. For example:

  ```
  mu <- alpha + beta ( predictor )
  ```

- Parameters, usually but not necessarily named `alpha` and `beta`, that represent the constant (or the intercept) and the coefficient (or the slope) in the regression equation.
- A parameter, often but not necessarily termed *sigma*, which represents the standard deviation of the outcome variable. This determines the spread of the outcome variable's distribution across its x-axis.
- A data frame that contains, at a minimum, the values for the outcome variable (in this example, `MPG`) and for a predictor variable such as `Speed`. The data frame might be named `CarData`.

Here's how the quap function might appear for an analysis of MPG given a single predictor variable, Speed:

```
CarQuap <- quap(
       alist(
               MPG ~ dnorm ( mu, sigma )
               mu <- alpha + beta ( Speed )
               alpha ~ dnorm ( 0, 1 )
               beta ~ dnorm ( 0, 1 )
               sigma ~ dexp (1)
       ), data = CarData )
```

A few comments about the arguments to the quap function:

- As I mentioned earlier in this chapter, it's usually a good idea to standardize the values that you supply for the outcome variable and the predictor variable(s) before passing them along to quap. Doing so minimizes the effects that numeric overflows can have on the results of the analysis. You can use an R function, standardize, to handle this for you, or you can subtract the mean value of a variable from each actual value and divide each result by the variable's standard deviation. (The results are often termed *z-scores*.)
- One result of this standardization is that the z-scores will have a mean of 0 and a standard deviation of 1. It often works out well, especially if you have standardized the predictors and the outcome variable, to use 0 and 1 as the mean and sigma of the dnorm arguments that describe the distributions of alpha and beta.
- Notice the use of the tilde instead of an assignment operator in several lines of the quap code. This simply indicates that a parameter is to be distributed as the density of, in this case, a normal curve.
- In this example, sigma is specified as sigma ~ dexp(1). The dexp function returns the density of the exponential distribution, which is the parent for a variety of other continuous distributions such as the Gaussian-normal, the Gamma, the Poisson, and the Binomial.

  The exponential distribution has one parameter, rate (or lambda); by contrast, the Gaussian distribution has two: the mean and the standard deviation. In R syntax, the exponential distribution's rate parameter is 1 by default, and the dexp function returns the density probability for the associated quantile, x (or 1 as here). Among other reasons, the exponential distribution is handy for specifying sigma, because the exponential is constrained to positive returns, and the standard deviation is, by definition, a positive quantity.

That's all you need for a simple regression of one outcome variable on one predictor. To add a predictor and analyze the simultaneous effect of two on one outcome variable, you need four items omitted from the single-predictor analysis:

1. The additional predictor named Weight should be added to the input data frame named CarData above.

2. The additional regression coefficient, for `Weight`, must be specified by the addition of this line of code:

   ```
   Weight_beta ~ dnorm ( 0, 1 )
   ```

3. In addition, for clarity it makes sense to edit the existing specification for the `Speed` coefficient to this:

   ```
   Speed_beta ~ dnorm ( 0, 1 )
   ```

4. The `Weight` predictor and its coefficient should be added to the regression equation. In the single-variable example, that equation looks like this:

   ```
   mu <- alpha + beta ( Speed )
   ```

   In the two-variable example the equation looks like this:

   ```
   mu <- alpha + Speed_beta ( Speed ) + Weight_beta (Weight)
   ```

   The full code example might look like this:

   ```
   library(rethinking)
   setwd("C:/Users/conra/Documents/Pearson Bayes/Drafts/Ch 6")
   CarDataFrame <- read.csv("Cars.csv")
   #You may need to adjust the path to the .csv file on your computer
   #The three variables are named Spd, Wt and Mileage
   #in the csv file. They are saved as newly standardized data
   #with new names (Speed, Weight, and MPG) in the
   #same steps that standardize them.
   CarDataFrame$Speed <- standardize( CarDataFrame$Spd )
   CarDataFrame$Weight <- standardize( CarDataFrame$Wt )
   CarDataFrame$MPG <- standardize( CarDataFrame$Mileage )
   regmodel <- quap(
   alist(
   MPG ~ dnorm( mu , sigma ) ,
   mu <- a + ( Speed_beta * Speed ) + ( Weight_beta * Weight ) ,
   a ~ dnorm( 0 , 1 ) ,
   Weight_beta ~ dnorm ( 0, 1 ) ,
   Speed_beta ~ dnorm ( 0, 1 ) ,
   sigma ~ dexp( 1 )
   ) , data = CarDataFrame )
   ```

You can get a smattering of summary information using the rethinking library's `precis` function. Simply supply it with the name of the quap model you just created, and specify the number of significant figures if you wish:

```
precis(regmodel, digits=6)
```

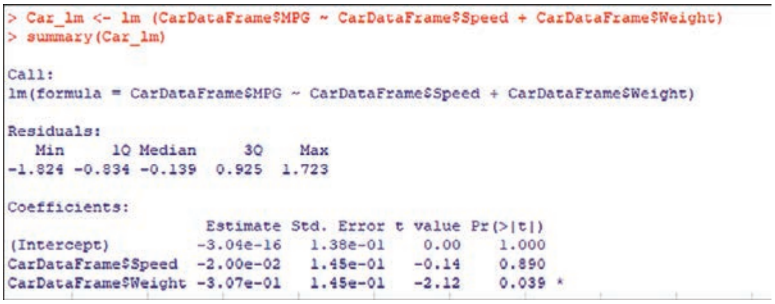
Here's what `precis` returns:

	mean	sd	5.50%	94.50%
a	-1.1E-05	0.131111	-0.20955	0.20953
Weight_beta	-0.30059	0.137421	-0.52021	-0.08096
Speed_beta	-0.01806	0.137417	-0.23768	0.201556
sigma	0.93517	0.092242	0.78775	1.08259

(The 5.50% and 94.50% limits are how the developer of the rethinking package chooses to protest the conventional and arbitrary criteria of, for example, 5% and 95% confidence intervals.)

To check your work, consider running a true regression package on the data that this section has analyzed. One convenient way, using continuous predictors and an outcome as here, is to use the `lm` package. If you do so after running your Bayesian analysis you can take advantage of the data frame you just created. For example, you can get quite a bit of summary information from these two statements, which return the results shown in Figure 6.4:

```
Car_lm <- lm (CarDataFrame$MPG ~ CarDataFrame$Speed + CarDataFrame$Weight)
   summary(Car_lm)
```



```
> Car_lm <- lm (CarDataFrame$MPG ~ CarDataFrame$Speed + CarDataFrame$Weight)
> summary(Car_lm)

Call:
lm(formula = CarDataFrame$MPG ~ CarDataFrame$Speed + CarDataFrame$Weight)

Residuals:
   Min     1Q Median     3Q    Max
-1.824 -0.834 -0.139  0.925  1.723

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -3.04e-16   1.38e-01    0.00    1.000
CarDataFrame$Speed  -2.00e-02   1.45e-01   -0.14    0.890
CarDataFrame$Weight -3.07e-01   1.45e-01   -2.12    0.039 *
```

Figure 6.4
The `lm` function performs a traditional multiple regression analysis.

Notice first that the intercept and coefficients returned by `lm` are close to the a (alpha) and Speed and Weight (betas) returned by `quap` and `precis`, but do not duplicate them precisely. This is largely due to traditional regression's use of the maximum R^2 as its criterion that a solution has been reached.

NOTE

In making your comparisons, bear in mind that `lm` and `precis` might each display the regression coefficients differently than the other.

Furthermore, `lm` by default returns only three significant figures, but you can choose the number of digits with `quap`'s `digits` argument. You might want to compare as many as, say, eight digits in the regression coefficients. One way to do so is via the `options` function. For example, these functions:

```
options(digits=4)
coef(Car_lm)
```

return these results:

```
(Intercept)  CarDataFrame$Speed CarDataFrame$Weight
 -3.036e-16          -2.005e-02          -3.066e-01
```

but these functions:

```
options(digits=6)
coef(Car_lm)
```

return these results:

```
(Intercept)  CarDataFrame$Speed CarDataFrame$Weight
 -3.03642e-16        -2.00472e-02        -3.06564e-01
```

(In the latter two examples I've used the `coef` function instead of the `summary` function to save space by showing only the coefficients.)

There are lots of ways to specify numeric formats in R. The `options` statement, just discussed, belongs to R's base functions, whereas the `digits` specification belongs, among many others, to the `quap` function. This situation tends to make matters more confused rather than less.

Summary

And that's the main point of this chapter: to clarify the aspects of Bayesian analysis without confusing you with abstruse details such as the physics of sampling in an MCMC context. It's my intention, and I believe the intention of `quap`'s author, Richard McElreath, to provide a steppingstone from an oversimplified discussion of grid approximation to an overcomplex essay on multilevel regression. It's important to understand how and why Bayesian techniques require a definition of your variables' distribution. Then, you're much better placed to also understand the workings of nominal variables and MCMC, the topics of the final two chapters in this book.

This page intentionally left blank

Index