FIFTH EDITION

VBA FOR MODELERS

Developing Decision Support Systems with Microsoft® Office Excel®

S. CHRISTIAN ALBRIGHT

VBA FOR MODELERS

DEVELOPING DECISION SUPPORT SYSTEMS WITH MICROSOFT® OFFICE EXCEL®

VBA FOR MODELERS

DEVELOPING DECISION SUPPORT SYSTEMS WITH MICROSOFT® OFFICE EXCEL®

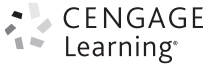
FIFTH EDITION

S. Christian Albright

Kelley School of Business, Indiana University

CENGAGE Learning®

Australia • Brazil • Mexico • Singapore • United Kingdom • United States



VBA for Modelers: Developing Decision Support Systems with Microsoft® Office Excel®, Fifth Edition
S. Christian Albright

Vice President, General Manager Science, Math, and Quantitative Business: Balraj Kalsi

Product Director: Joe Sabatino

Product Manager: Aaron Arnsparger

Associate Content Developer: Brad Sullender

Manufacturing Planner: Ron Montgomery

Marketing Manager: Heather Mooney

Art and Cover Direction, Production Management, and Composition: Lumina Datamatics, Inc.

Cover Image: © Awstok/Shutterstock

Intellectual Property

Analyst: Christina Ciaramella

Project Manager: Betsy Hathaway

Library of Congress Control Number: 2014958175

ISBN: 978-1-285-86961-2

Cengage Learning
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com**.

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

To learn more about Cengage Learning Solutions, visit **www.cengage.com**

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com**

Printed in the United States of America
Print Number: 01 Print Year: 2015

To my wonderful wife, Mary—she is my best friend and constant companion. To our talented son, Sam, his equally talented wife, Lindsay, and our two amazing grandsons, Teddy and Archer. And to Bryn, our dear Welsh corgi who still just loves to play ball.

About the Author

S. Christian Albright

Chris Albright got his B.S. degree in Mathematics from Stanford in 1968 and his Ph.D. degree in Operations Research from Stanford in 1972. Until his retirement in 2011, he taught in the Operations & Decision Technologies Department in the Kelley School of Business at Indiana University. His teaching included courses in management science, computer simulation, and statistics to all levels of business students: undergraduates, MBAs, and doctoral students. He has published over 20 articles in leading operations research journals in the area of applied probability and he has authored several books, including *Practical Management Science, Data Analysis and Decision Making, Data Analysis for Managers, Spreadsheet Modeling and Applications*, and *VBA for Modelers.* He jointly developed *StatTools*, a statistical add-in for Excel, with the Palisade Corporation. In "retirement," he continues to revise his books, he works as a consultant for Palisade, and he has developed a commercial product, Excel Now!, an Excel tutorial.

On the personal side, Chris has been married to his wonderful wife Mary for 43 years. They have a special family in Philadelphia: their son Sam, his wife Lindsay, and their two sons, Teddy and Archer. Chris has many interests outside the academic area. They include activities with his family (especially traveling with Mary), going to cultural events at Indiana University, power walking, and reading. And although he earns his livelihood from statistics and management science, his *real* passion is for playing classical music on the piano.

Contents

Preface

I wrote *VBA for Modelers* for students and professionals who want to create decision support systems (DSSs) using Microsoft Excel–based spreadsheet models. The book does *not* assume any prior programming experience. It contains two parts. Part I covers the essentials of VBA (Visual Basic for Applications) programming, and Part II provides many applications with their associated programming code. This part assumes that readers are either familiar with spreadsheet modeling or are taking a concurrent course in management science or operations research. There are many excellent books available for VBA programming, many others covering decision support systems, and still others for spreadsheet modeling. However, I have not found a book that attempts to unify these subjects in a practical way. *VBA for Modelers* is designed for this purpose, and I hope you will find it to be an important resource and reference in your own work.

Why This Book?

The original impetus for this book began about 20 years ago. Wayne Winston and I were experimenting with the spreadsheet approach to teaching management as we were writing the first edition of our *Practical Management Science (PMS)* book. Because I have always had an interest in computer programming, I decided to learn VBA, the relatively new macro language for Excel, and use it to a limited extent in my undergraduate management science modeling course. My intent was to teach the students how to wrap a given spreadsheet model, such as a product mix model, into an *application* with a "front end" and a "back end" by using VBA. The front end would enable a user to provide inputs to the model, usually through one or more dialog boxes, and the back end would present the user with a nontechnical report of the results. I found it to be an exciting addition to the usual modeling course, and my students overwhelmingly agreed.

The primary problem with teaching this type of course was the lack of an appropriate VBA textbook. Although there are many good VBA trade books available, they usually go into much more technical VBA details than I have time to cover, and their objective is usually to teach VBA programming as an end in itself. I expect that many adopters of our *Practical Management Science* book will decide to use parts of *VBA for Modelers* to supplement their management science courses, just as I have been doing. For readers who have already taken a management science course, there is more than enough material in this book to fill an entire elective course or to be used for self-study.

However, even for readers with no background or interest in management science, the first part of this book has plenty of value. We are seeing an increasing

number of our business students and graduates express interest in automating Excel with macros. In short, they want to become Excel "power users." After the first edition of this book appeared, I taught a purely elective MBA course covering the first part of the book. To my surprise and delight, it regularly attracted about 40 MBA students per year. Yes, it attracted *MBA students*, not computer science majors! (Since I have retired from teaching, the VBA course is still being taught, and it continues to attract these types of audiences.). The students see real value in knowing how to program for Excel. And it is amazing and gratifying to see how far these students can progress in a short 7-week course. Many find programming, especially for Excel, to be as addictive as I find it.

Objectives of the Book

VBA for Modelers shows how the power of spreadsheet modeling can be extended to the masses. Through VBA, complex management science models can be made accessible to nontechnical users by providing them with simplified input screens and output reports. The book illustrates, in complete detail, how such applications can be developed for a wide variety of business problems. In writing the book, I have always concerned myself with the following questions: How much will readers be able to do on their own? Is it enough for readers to see the completed applications, marvel at how powerful they are, and possibly take a look at the code that runs in the background? Or should they be taken to the point where they can develop their *own* applications, code and all? I suspect this depends on the audience, but I know I *can* get students to the point where they can develop modest but useful applications on their own and, importantly, experience the thrill of programming success.

With these thoughts in mind, I have written this book so that it can be used at several levels. For readers who want to learn VBA from scratch and then apply it, I have provided a "VBA primer" in Part I of the book. It is admittedly not as complete as some of the thick Excel VBA books available, but I believe it covers the basics of VBA quite adequately. Importantly, it covers coding methods for working with Excel ranges in Chapter 6 and uses these methods extensively in later chapters, so that readers will not have to use trial and error or wade through online help, as I had to do when I was learning VBA. Readers can then proceed to the applications in Chapters 19 through 35 and apply their skills. In contrast, there are probably many readers who do not have time to learn all of the details, but they can still *use* the applications in Part II of the book for demonstration purposes. Indeed, the applications have been developed for generality. For example, the transportation model in Chapter 24 is perfectly general and can be used to solve *any* transportation model by supplying the appropriate input data.

Approach

I like to teach (and learn) through examples. I have found that I can learn a programming language only if I have a strong motivation to learn it. I suspect that

most of you are the same. The applications in the latter chapters are based on many interesting management science models. They provide the motivation for you to learn the material. The examples illustrate that this book is not about programming for the sake of programming. Instead, it is about developing useful applications for business. You probably already realize that Excel modeling skills make you more valuable in the workplace. This book will help you develop VBA skills that make you much *more* valuable.

Contents of the Book

The book is written in two parts. Part I, Chapters 1–18, is a VBA primer for readers with little or no programming experience in VBA (or any other language). Although all of these chapters are geared to VBA, some are more about general programming concepts, whereas others deal with the unique aspects of programming for Excel. Specifically, Chapters 7, 9, and 10 discuss control logic (If-Then-Else constructions), loops, arrays, and subroutines, topics that are common to all programming languages. In contrast, Chapters 6 and 8 explain how to work with some of the most common Excel objects (ranges, workbooks, worksheets, and charts) in VBA. In addition, several chapters discuss aspects of VBA that can be used with Excel and any other applications (Access, Word, PowerPoint, and so on) that use VBA as their programming language. Specifically, Chapter 3 explains the Visual Basic Editor (VBE), Chapter 4 illustrates how to record macros, Chapter 11 explains how to build user forms (dialog boxes), and Chapter 12 discusses the important topic of error handling.

The material in Part I is reasonably complete, but it is available, in greater detail and with a somewhat different emphasis, in several other books. The unique aspect of *this* book is Part II, Chapters 19–35. (Due to length, the last two chapters, Chapter 34, An AHP Application for Choosing a Job, and Chapter 35, A Poker Simulation Application, are available online only. You can find them at www.CengageBrain.com.) Each chapter in this part discusses a specific application. Most of these are optimization and simulation applications, and many are quite general. For example, Chapter 21 discusses a general product mix application, Chapter 23 discusses a general production scheduling application, Chapter 24 discusses a general transportation application, Chapter 25 discusses a stock-trading simulation, Chapter 29 discusses a multiple-server queue simulation, Chapter 30 discusses a general application for pricing European and American options, and Chapter 32 discusses a general portfolio optimization application. (Many of the underlying models for these applications are discussed in *Practical Management Science*, but I have attempted to make these applications stand-alone here.)

The applications can be used as they stand to solve real problems, or they can be used as examples of VBA application development. All of the steps in the development of these applications are explained, and all of the VBA source code is included. Using an analogy to a car, you can simply get in and drive, or you can open the hood and see how everything works.

Chapter 19 gets the process started in a "gentle" way. It provides a general introduction to application development, with an important list of guidelines. It then illustrates these guidelines in a car loan application. This application should be within the grasp of most readers, even if they are not yet great programmers. By tackling this application first, readers get to develop a simple model, with dialog boxes, reports, and charts, and then tie everything together. This car loan application illustrates an important concept that I stress throughout the book. Specifically, applications that really *do* something are often long and have a lot of details. But this does not mean that they are *difficult*. With perseverance—a word I use frequently—readers can fill in the details one step at a time and ultimately experience the thrill of getting a program to work correctly.

Virtually all management science applications require input data. A very important issue for VBA application development is how to get the required input data into the spreadsheet model. I illustrate a number of possibilities in Part II. If only a small amount of data is required, dialog boxes work well. These are used for data input in many of the applications. However, there are many times when the data requirements are much too large for dialog boxes. In these cases, the data are usually stored in some type of database. I illustrate some common possibilities. In Chapter 21, the input data for a product mix model are stored in a separate worksheet. In Chapter 31, the stock price data for finding the betas of stocks are stored in a separate Excel workbook. In Chapter 33, the data for a DEA model are stored in a text (.txt) file. In Chapter 24, the data for a transportation model are stored in an Access database (.mdb) file. Finally, in Chapter 32, the stock price data required for a portfolio optimization model are located on a Web site and are imported into Excel, *at runtime*. In each case, I explain the VBA code that is necessary to import the data into the Excel application.

New to the Fifth Edition

The impetus for writing the fifth edition was the release of Excel 2013. In terms of VBA, there aren't many changes from Excel 2010 to Excel 2013 (or even from Excel 2007 to Excel 2013), but I used the opportunity to incorporate changes that were made in Excel 2013, as well as to modify a lot of the material throughout the book.

- Programmers can never let well enough alone. We are forever tinkering with our code, not just to make it work better, but often to make it more elegant and easier to understand. So users of previous editions will see minor changes to much of the code throughout the book.
- The biggest change, which has nothing to do with the version of Excel, is the way information is passed between modules and user forms. In previous editions, I did this with global variables, a practice frowned upon by many professional programmers. In this edition, I pass the required information through arguments to "ShowDialog" functions in the user forms. This new method is explained in detail in Chapter 11 and is then used in later chapters where user forms appear.

- Chapter 15 contains a brief discussion of the new PowerPivot tool introduced in Excel 2013. This tool can actually be automated with VBA, but because of its advanced nature, I don't discuss the details. Maybe this will appear in the *next* edition of the book, by which time Excel's online help will hopefully be improved.

How to Use the Book

I have already discussed several approaches to using this book, depending on how much you want to learn and how much time you have. For readers with very little or no computer programming background who want to learn the fundamentals of VBA, Chapters 1–12 should be covered first, in approximately that order. (I should point out that it is practically impossible to avoid "later" programming concepts while covering "early" ones. For example, I admit to using a few If statements and loops in early chapters, *before* discussing them formally in Chapter 7. I don't believe this should cause problems. I use plenty of comments, and you can always look ahead if you need to.) After covering VBA fundamentals in the first 12 chapters, the next six optional chapters can be covered in practically any order.

Chapter 19 should be covered next. Beyond that, the applications in the remaining chapters can be covered in practically any order, depending on your interests. However, some of the details in certain applications will not make much sense without the appropriate training in the management science models. For example, Chapter 34 discusses an AHP (Analytical Hierarchy Process) application for choosing a job. The VBA code is fairly straightforward, but it will not make much sense unless you have some knowledge of AHP. I assume that the knowledge of the models comes from a separate source, such as *Practical Management Science*; I cover it only briefly here.

Finally, readers can simply use the Excel application files to solve problems. Indeed, the applications have been written specifically for nontechnical end users, so that readers at all levels should have no difficulty opening the application files in Part II of the book and using them appropriately. In short, readers can decide how much of the material "under the hood" is worth their time.

Premium Web Site Content

The companion Web site for this book can be accessed at www.cengagebrain.com. There you will have access to all of the Excel (.xlsx and .xlsm) and other files mentioned in the chapters, including those in the exercises. The Excel files require Excel 97 or a more recent version, but they are realistically geared to Excel 2007 and later versions. Many of the files from Chapter 17 and later chapters "reference" Excel's Solver. They will not work unless the Solver add-in is installed and loaded. Chapters 14 and 24 uses Microsoft's ActiveX Data Object (ADO)

model to import the data from an Access database into Excel. This will work only in Excel 2000 or a more recent version. Finally, Chapter 13 uses the Office FileDialog object. This works only in Excel XP (2002) or a more recent version.

The book is also supported by a Web site at www.kelley.iu.edu/albrightbooks. The Web site contains errata and other useful information, including information about my other books.

Acknowledgments

I would like to thank all of my colleagues at Cengage Learning. Foremost among them are my current editor, Aaron Arnsbarger, and my former editors, Curt Hinrichs and Charles McCormick. The original idea was to develop a short VBA manual to accompany our *Practical Management Science* book, but Curt persuaded me to write an entire book. Given the success of the first four editions, I appreciate Curt's insistence. I am also grateful to many of the professionals who worked behind the scenes to make this book a success:

- Brad Sullender, Content Developer; Heather Mooney, Marketing Manager; Kristina Mose-Libon, Art Director; and Sharib Asrar as the Project Manager at Lumina Datamatics.

Next, I would like to thank the reviewers of past editions of the book. Thanks go to

- Gerald Aase, Northern Illinois University; Ravi Ahuja, University of Florida; Grant Costner, University of Oregon; R. Kim Craft, Rollins College; Lynette Molstad Gorder, Dakota State University; and Jim Hightower, California State University-Fullerton; Don Byrkett, Miami University; Kostis Christodoulou, London School of Economics; Charles Franz, University of Missouri; Larry LeBlanc, Vanderbilt University; Jerry May, University of Pittsburgh; Jim Morris, University of Wisconsin; and Tom Schriber, University of Michigan.

Finally, I want to thank my wife, Mary. She continues to support my book-writing activities, even when it requires me to work evenings and weekends in front of a computer. I also want to thank our Welsh corgi Bryn, who faithfully accompanies her daddy when he goes upstairs to do his work. She doesn't add much technical assistance, but she definitely adds a lot of motivational assistance.

S. Christian Albright
(e-mail at albright@indiana.edu,
Web site at www.kelley.iu.edu/albrightbooks)
Bloomington, Indiana
January 2015

Part I

VBA Fundamentals

This part of the book is for readers who need an introduction to programming in general and Visual Basic for Applications (VBA) for Excel in particular. It discusses programming topics that are common to practically all programming languages, including variable types and declarations, control logic, looping, arrays, subroutines, and error handling. It also discusses many topics that are specific to VBA and its use with Excel, including the Excel object model; recording macros; working with ranges, workbooks, worksheets, charts, and other Excel objects; developing user forms (dialog boxes); and automating other applications, including Word, Outlook, Excel's Solver add-in, and Palisade's @RISK add-in, with VBA code.

Many of the chapters in Part I present a business-related exercise immediately after the introductory section. The objective of each such exercise is to motivate you to work through the details of the chapter, knowing that many of these details will be required to solve the exercise. The finished files are included in the online materials, but I urge you to try the exercises on your own, before looking at the solutions.

The chapters in this part should be read in approximately the order they are presented, at least up through Chapter 12. Programming is a skill that builds upon itself. Although it is not always possible to avoid referring to a concept from a later chapter in an earlier chapter, I have attempted to refrain from doing this as much as possible. The one small exception is in Chapters 6 (on ranges) and 7 (on control logic and loops). It is almost impossible to do any interesting programming in Excel without knowing about ranges, and it is almost impossible to do any interesting programming in general without knowing about control logic and loops. I compromised by putting the chapter on ranges first and using some simple control logic and loops in it. I don't believe this should cause any problems.

1 Introduction to VBA Development in Excel

1.1 Introduction

My books *Practical Management Science* (PMS) and *Business Analytics: Data Analysis and Decision Making* (DADM), both co-authored with Wayne Winston, illustrate how to solve a wide variety of business problems by developing appropriate Excel models. If you are familiar with this modeling process, you probably do not need to be convinced of the power and applicability of Excel. You realize that Excel modeling skills will make you a valuable employee in the workplace. This book takes the process one giant step farther. It teaches you how to develop applications in Excel by using Excel's programming language, Visual Basic for Applications (VBA).

In many Excel-modeling books, you learn how to model a particular business problem. You enter given inputs in a worksheet, you relate them with appropriate formulas, and you eventually calculate required outputs. You might also optimize a particular output with Solver, and you might create one or more charts to show outputs graphically. You do all of this through the Excel interface, using its ribbons (as of Excel 2007), menus, and toolbars, entering formulas into its cells, using the chart tools, using the Solver dialog box, and so on. If you are conscientious, you document your work so that other people in your company can understand your completed model. For example, you clearly indicate the input cells so that other users will know which cells they should use for their own inputs and which cells they should leave alone.

Now suppose that your position in a company is to *develop* applications for other less-technical people in the organization to use. Part of your job is still to develop spreadsheet models, but the details of these models might be incomprehensible to many users. These users might realize that they have, say, a product mix problem, where they will have to supply certain inputs, and then some computer magic will eventually determine a mix of products that optimizes company profit. However, the part in between is beyond their capabilities. Your job, therefore, is to develop a user-friendly application with a model (possibly hidden from the user) surrounded by a "front end" and a "back end." The front end will present the user with dialog boxes or some other means for enabling them to define their problem. Here they will be asked to specify input values and possibly other information. Your application will take this information, build the appropriate model, optimize it if necessary, and eventually present the back end to the user—a nontechnical report of the results, possibly with accompanying charts.

This application development is possible with VBA, as I will demonstrate in this book. I make no claim that it is easy or that it can be done quickly, but I do claim that it is within the realm of possibility for people like yourself, not just for professional programmers. It requires a logical mind, a willingness to experiment and take full advantage of online help, plenty of practice, and, above all, perseverance. Even professional programmers seldom accomplish their tasks without difficulty and plenty of errors; this is the nature of programming. However, they learn from their errors (and their colleagues), and they refuse to quit until they get their programs to work properly. Computer programming is essentially a process of overcoming one small hurdle after another. This is where perseverance is so important. But if you are not easily discouraged, and if you love the feeling of accomplishment that comes from getting something to work, you will love the challenge of application development described in the book.

1.2 VBA in Excel 2007 and Later Versions

As you are probably aware, Excel went through a major face lift in 2007. The look of Excel, especially its menus and toolbars, is now much different than in Excel 2003 and earlier. Unfortunately, some users have not converted to Excel 2007 or a later version, so book authors, including myself, are in the uncomfortable position of having to write simultaneously for several audiences. Fortunately, not much about VBA changed in the transition from 2003 to 2007 or from 2007 to 2010 or from 2010 to 2013. I will try to point out the differences as necessary throughout the book, hopefully without interrupting the flow too much.

Perhaps the main difference is in the file extensions you will see. In Excel 2003 and earlier, all Excel files (except for add-ins, not covered here) ended in .xls. It didn't matter whether they contained VBA code or not; they were still .xls files. In Excel 2007 and later versions, there are two new extensions. Files without VBA code now have .xlsx extensions, whereas files with VBA code *must* use .xlsm extensions. If you try to save a file with VBA code as an .xlsx file, you won't be allowed to do so. There is one exception: you can save your new files in the old Excel 2003 format, which is still an option (with Save As), in which case they will have .xls extensions. Why would you do this? The probable reason is that you want to share a file you created in Excel 2007 or a later version with a friend who still uses Excel 2003. Of course, if your file includes features new to Excel 2007 or a later version, your friend won't be able to see them.

I have been using Excel 2007, 2010, and now 2013 since their original releases, and I personally think they are great improvements over earlier versions, at least in most respects. So I will provide my example files in .xlsx and .xlsm formats. If you are using Excel 2003, you will be able to open these if you first install a free Office Compatibility Pack from Microsoft (just search the Web for it). Without this compatibility pack, Excel 2003 users cannot read files in the new .xlsx or .xlsm formats (although users of Excel 2007 and later versions can always read files in the old .xls format).

The fortunate part is that VBA has changed very little. I will usually *not* include new features of Excel 2007 or later versions in my example files that Excel 2003 users (even those with the compatibility pack) could not see. And in the few cases where I need to do so, I will make it clear that these examples are for users of Excel 2007 or later versions only.

1.3 Example Applications

If you have used my PMS or DADM books, you probably understand what a spreadsheet model is. However, you might not understand what I mean by spreadsheet *applications* with front ends and back ends. In other words, you might not understand what this book intends to teach you. The best way to find out is to run some of the applications that will be explained in Part II of the book. At this point, *you* can become the nontechnical user by opening any of the following files that accompany this book: **Product Mix.xlsm, Scheduling.xlsm, Stock Options.xlsm,** and **Transportation.xlsm**. Simply open any of these files and follow instructions. It should be easy. After all, the purpose of writing these applications is to make it easy for a nontechnical user to run them and get results they can understand. Now step back and imagine what must be happening in the background to enable these applications to do what they do. This is what you will be learning in the book. By running a few applications, you will become anxious to learn how to do it yourself. These sample applications illustrate just how powerful a tool VBA for Excel can be.

Security Settings and Trusted Locations

You might encounter annoying messages when you try to open these applications. Microsoft realizes that viruses can be carried in VBA code, so it tries to protect users. First, it sets a macro security level to High by default. This level disallows *any* VBA macros to run. Obviously, this is not good when you are trying to learn VBA programming. The fix is easy.

- For users of Excel 2010 and 2013, open Excel, click the File button, then Options, then the Trust Center tab, then Trust Center Settings, then the Macro Settings tab, and check the "Disable all macros with notification" option.
- For users of Excel 2007, it is the same as for Excel 2010 and 2013 except that you click the Office button, not the File button. (The Office button was replaced by the File button in 2010.)
- For users of Excel 2003 or earlier, open Excel, select the **Tools → Macro → Security** menu item, and select Medium.
- You should need to do this only once. However, even with this macro security setting, you are always asked whether you want to enable macros when you open a file that contains VBA code. Of course, you should typically enable macros. Otherwise, you will be safe from viruses, but none of the VBA code will run!

There is another option, at least in Excel 2007 and later versions, which avoids the security settings altogether. If you find that most of the Excel files with VBA code are in a particular folder on your hard drive, you can add this folder to the list of *trusted locations* on your computer. To do this, a one-time task on a given computer, go to the Trust Center Settings, as explained in the first bullet above, then Trusted Locations, then "Add new location," and browse for the folder you want to add. (In the resulting dialog box, you will probably want to check the "Subfolders of this location are also trusted" option.) From then on, any .xlsm files in this folder (or its subfolders) will open without any warning about enabling macros.

I will make one final comment about enabling macros that pertains to Excel 2007 or later versions only. If you open a file that contains macros, that is, an .xlsm file, and it isn't in a trusted location, you sometimes see the message in Figure 1.1 and you sometimes instead see the button in Figure 1.2 (right above the formula bar). Thanks to John Walkenbach, a fellow VBA author, I finally learned the pattern. If the VB editor (discussed in Chapter 3) is already open when you open the file, you will see the message in Figure 1.1. If it isn't open, you will see the button in Figure 1.2. Why did Microsoft do it this way? I have no idea.

Figure 1.1 Enable Macro Message with VB Editor Open

Figure 1.2 Enable Macro Button with VB Editor Not Open

1.4 Decision Support Systems

In many companies, programmers provide applications called **decision support systems (DSSs).** These are applications, based on Excel or some other package, that help managers make better decisions. They can vary from very simple to very complex, but they usually provide some type of user-friendly interface so that a manager can experiment with various inputs or decision variables to see their effect on important output variables such as profit or cost. Much of what you will be learning, especially in Part II of the book, is how to create Excel-based DSSs. In fact, if you ran the applications in the previous section, you should already understand what decision support means. For example, the Transportation application helps a manager plan the optimal shipping of a product in a logistics network, and the Stock Options application helps a financial analyst price various types of financial options. The value that you, the programmer, provide by developing these applications is that other people in your company can then run them—easily—to make better decisions.

1.5 Required Background

Readers of this book probably vary widely in their programming experience. At one extreme, many of you have probably never programmed in VBA or any other language. At the other extreme, a few of you have probably programmed in Visual Basic but have never used it to automate Excel and build Excel applications. In the middle, some of you have probably had some programming experience in another language such as C or Java but have never learned VBA. This book is intended to appeal to all such audiences. Therefore, a simplified answer to the question, "What programming background do I need?" is "None." You need only a willingness to learn and experiment.

If you ran the applications discussed in Section 1.2, you are probably anxious to get started developing similar applications. If you already know the fundamentals of VBA for Excel, you can jump ahead to Part II of the book. But most of you will have to learn how to walk before you can run. Therefore, the chapters in Part I go through the basics of the VBA language, especially as it applies to Excel. The coverage of this basic material will provide you with enough explanations and examples of VBA's important features to enable you to understand the applications in Part II—and to do some Excel development on your own.

If you want more detailed guidance in VBA for Excel, you can learn from Microsoft's online help or the many user groups on the Web. Indeed, this is perhaps the best way to learn, especially in the middle of a development project. If you need to know one specific detail to overcome a hurdle in the program you are writing, you can look it up quickly in online help or do an online search for it. A good way to do this will be demonstrated shortly.

Part II of the book does presume some modeling ability and general business background. For example, if you ran the Product Mix application, you probably realize that it develops and optimizes a product mix model, a classic

linear programming model. One (but not the only) step in developing this application is to develop a product mix model exactly as in Chapter 3 of PMS. As another example, if you ran the Stock Options application, you realize the need to understand option pricing, explained briefly in the second simulation chapter of PMS. Many of the applications in this book are based on examples (product mix, scheduling, transportation, and so on) from PMS or DADM. You can refer to these books as necessary.

1.6 Visual Basic Versus VBA

Before going any further, I want to clarify one common misconception. Visual Basic (VB) is *not* the same as VBA. VB is a software development language that you can buy and run separately, without the need for Excel or Office. Actually, there are several versions of VB available. The most recent is called VB.NET, which comes with Microsoft's Visual Studio software development suite. (The .NET version of VB has many enhancements to the VB language.) Before VB.NET, there was VB6, still in use in thousands of applications. In contrast, VBA comes with Office. If you own Microsoft Office, you own VBA. The VB language is very similar to VBA, but it is not the same. The main difference is that VBA is the language you need to manipulate Excel, as you will do here.

You can think of it as follows. The VBA language consists of a "backbone" programming language with typical programming elements you find in all programming languages: looping, logical If-Then-Else constructions, arrays, subroutines, variable types, and others. In this respect, VBA and VB are essentially identical. However, the "A" in VBA means that any application software package, such as Excel, Access, Word, or even a non-Microsoft software package, can "expose" its *object model* to VBA, so that VBA can manipulate it programmatically. In short, VBA can be used to develop applications for any of these software packages. This book teaches you how to do so for Excel.

Excel's objects are discussed in depth in later chapters, but a few typical Excel objects you will recognize immediately are ranges, worksheets, workbooks, and charts. VBA for Excel knows about these Excel objects, and it enables you to manipulate them with code. For example, you can change the font of a cell, name a range, add or delete a worksheet, open a workbook, and change the title of a chart. Part of learning VBA for Excel is learning the VB backbone language, the elements that have nothing to do with Excel specifically. But another part, the more challenging part, involves learning how to manipulate Excel's objects in code. That is, it involves learning how to write computer programs to do what you do every day through the familiar Excel interface. If you ever take a course in VB, you will learn the backbone elements of VBA, but you will not learn how to manipulate objects in Excel. This requires VBA, and you *will* learn it here.

By the way, there are also VBA for Access, VBA for Word, VBA for PowerPoint, VBA for Outlook, and others. The difference between them is that each

has its own specific objects. To list just a few, Access has tables, queries, and forms; Word has paragraphs and footnotes; PowerPoint has slides; and Outlook has mail. Each version of VBA shares the same VB backbone language, but each requires you to learn how to manipulate the objects in the specific application. There is certainly a learning curve in moving, say, from VBA for Excel to VBA for Word, but it is not nearly as difficult as if they were totally separate languages. In fact, the power of VBA, as well as the relative ease of programming in it, has prompted many third-party software developers to license VBA from Microsoft so that they can use VBA as the programming language for their applications. One example is Palisade, the developer of the @RISK and PrecisionTree add-ins for Excel, as will be discussed briefly in Chapter 17. In short, once you know VBA, you know a lot about what is happening in the programming world—and you can very possibly use this knowledge to obtain a valuable job in business.

1.7 Some Basic Terminology

Before proceeding, it is useful to clarify some very basic and important terminology that will be used throughout the book. First, whenever you program in any language, your basic building blocks are lines of **code**, short for **programming code**. Any line of code is intended to accomplish something, and it must obey the rules of syntax for the programming language being used. This book is all about coding in VBA.

Typically, a set of logically related lines of code that accomplishes a specific task is called a **subroutine**, a **procedure**, or a **macro**. In fact, one of the first keywords you will learn in VBA is `Sub`. This keyword begins all subroutines. The terms *subroutine*, *procedure*, and *macro* are essentially equivalent, although programmers tend to use the terms *subroutine* and *procedure*, whereas spreadsheet users tend to use the term *macro*. I tend to refer to any of these as a sub.

Finally, the term **program** is typically used to refer to all of the subs in an application. When you explore the more complex applications in Part II of the book, you will see that they often include many subs, where each sub is intended to perform one specific task in the overall program. (Chapter 10 discusses why this division of a program into multiple subs makes a lot of sense.)

1.8 Summary

VBA is the programming language of choice for an increasingly wide range of application developers. The main reason for this is that VBA uses the familiar Visual Basic programming language and then adapts it to many Microsoft and even non-Microsoft application software packages, including Excel. In addition, VBA is a relatively easy programming language to master. This makes it accessible to a large number of nonprofessional programmers in the business world—including you. By learning how to program in VBA, you will definitely enhance your value in the workplace.