Financial Modeling

Simon Benninga

fourth edition

FINANCIAL MODELING

FINANCIAL MODELING

Simon Benninga

With a section on Visual Basic for Applications
by Benjamin Czaczkes

FOURTH EDITION

The MIT Press
Cambridge, Massachusetts
London, England

Print copies of this book include a card affixed to the inside back cover with a unique access code. Access codes are required to download Excel worksheets and solutions to end-of-chapter exercises. If you purchased an e-book, you may obtain a unique access code by emailing journals-access@mit.edu or calling 617-253-2889 or 800-207-8354 (toll-free in the U.S. and Canada).

This book was set in Times Roman by Toppan Best-set Premedia Limited. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Benninga, Simon.
Financial modeling / Simon Benninga.—Fourth edition.
pages cm
Includes bibliographical references and index.
ISBN 978-0-262-02728-1 (hardcover : alk. paper)
1. Finance—Mathematical models. 2. Microsoft Visual Basic for applications. I. Title.
HG173.B46 2014
332.01'5118—dc23

2013032409

10 9 8 7 6 5 4 3 2 1

To the memory of our parents:

Helen Benninga (1913–2008)
Groningen, Netherlands – Jerusalem, Israel

Noach Benninga (1909–1994)
Eenrum, Netherlands – Asheville, North Carolina

Esther Czaczkes (1931–2012)
Jerusalem, Israel – Jerusalem, Israel

Alfred Czaczkes (1923–1997)
Vienna, Austria – Jerusalem, Israel

Contents

Preface

The three previous editions of *Financial Modeling* have received a gratifyingly positive response from readers. The combination of a "cookbook," mixing explanation and implementation using Excel, has fulfilled a need in both the academic and the practitioner markets from readers who realize that the implementation of the finance basics typically studied in an introductory finance course requires another, more heavily computational and implementational approach. Excel, the most widely used computational tool in finance, is a natural vehicle for deepening our understanding of the materials.

In this fourth edition of *Financial Modeling*, I have added a section (Chapters 24–30) on Monte Carlo methods. The intention is to add a focus on the simulation of financial models. I have become convinced that a statistical understanding of modeling ("What is the mean and sigma of the portfolio return?") understates the impact of the uncertainty. Only by simulating the models and the return processes can we get a good feel for the dimensions of the uncertainty.

With the added section on Monte Carlo, *Financial Modeling* now consists of seven sections. Each of the first five sections of the book relates to a specific area of finance. These sections are independent of each other, though the reader should realize that they all assume some familiarity with the finance area—*Financial Modeling* is not an introductory text. Section I (Chapters 1–7) deals with corporate finance topics; Section II (Chapters 8–14) with portfolio models; Section III (Chapters 15–19) with option models; and Section IV (Chapters 20–23) with bond-related topics. Section V, as discussed above, introduces the reader to Monte Carlo methods in finance.

The last two sections of *Financial Modeling* are technical in nature. Section VI (Chapters 31–35) relates to various Excel topics which are used throughout the book. Chapters in Section VI can be read and accessed as necessary. Section VII (Chapters 36–39) deals with Excel's programming language, Visual Basic for Applications (VBA). VBA is used throughout *Financial Modeling* to create functions and routines which make life easier, but it is never intrusive—in principle the reader can understand the materials in all of the other chapters of *Financial Modeling* without needing the VBA chapters.

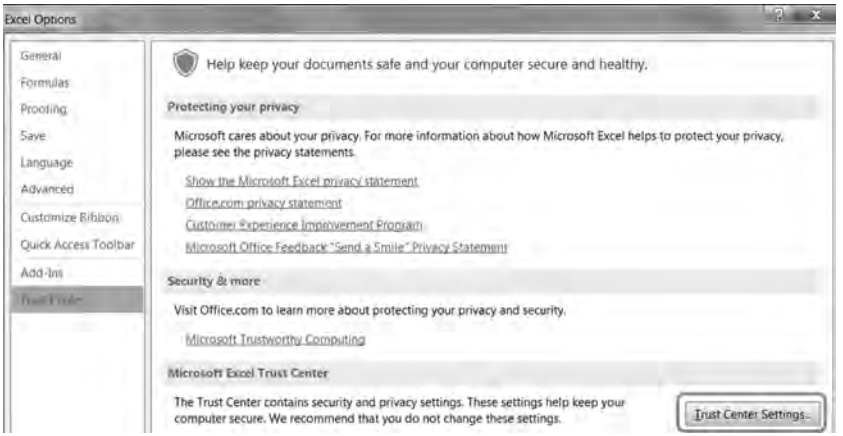
New Materials and Updates

This edition of *Financial Modeling* contains much new and updated material. We have already mentioned the new section on Monte Carlo methods. Also new are two chapters on valuation (Chapters 2 and 4) and a chapter on term structure modeling (Chapter 22). Much of the material has been tweaked

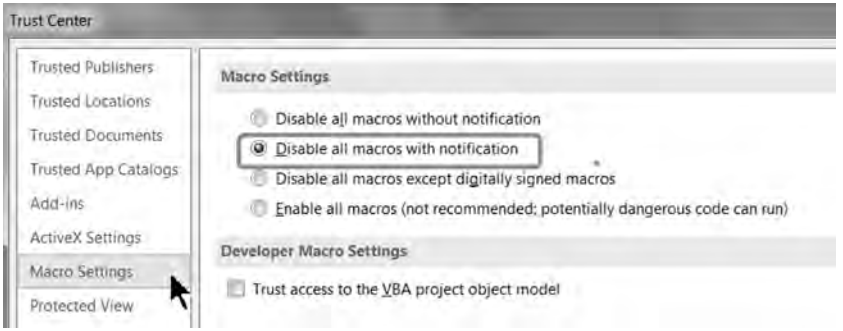
and improved. For example, the discussion of Excel financial functions now includes a discussion of XIRR and XNPV, including a fix for the bugs in these functions.

Getformula

The Excel files with this edition include a function called **Getformula** that enables the user to track cell contents. **Getformula** is discussed in Chapter 0 and also on a file on the disk that is included with *Financial Modeling*. To allow **Getformula** to work, go to **File|Options|Trust Center**:



In the **Trust Center** settings, I recommend the following setting:



If you have done this, then when opening an Excel notebook for the first time, you will be confronted by the following warning:

For notebooks that come with this book, you can safely click **Enable Content**, which enables the formulas on the notebook.

Excel Versions

In the examples throughout the book I have used Excel 2013. To the best of my knowledge, all of the spreadsheets work in Excel versions 2003, 2007, 2010, and 2011 (for Mac), although some minor and obvious adaptations by the reader may be called for.

Files for the Fourth Edition

Purchasers of *Financial Modeling* get access to all the Excel files for the chapters and exercises.

Using *Financial Modeling* in a University Course

Financial Modeling has become the book of choice in many advanced finance classes that stress the combination of modeling/Excel skills and a deeper understanding of the underlying financial models. The *Financial Modeling*–based courses are often a third- or fourth-year undergraduate or second-year MBA course. The courses are very different and include much instructor-specific input, but they seem to have a few general features in common:

- A typical course starts with two or three classes which stress the Excel skills needed for financial modeling. Often these courses are held in a computer lab. Though almost all business school students know Excel, they often do not know the finesses of data tables (Chapter 31), some of the basic financial functions (Chapters 1 and 33), and array functions (Chapter 34).
- Most one-semester courses then cover at most one of the *Financial Modeling* sections. If we assume that in a typical university course, covering one chapter per week is an upper limit (and many chapters will require two weeks), then a typical course might concentrate on either corporate finance (Chapters 1–7),

portfolio models (Chapters 8–14), or options (Chapters 15–19). At a stretch, the instructor could perhaps throw in the shorter bond section (Chapters 20–23).

• I suggest that after the initial classes in a computer lab, the instructor move to a regular classroom. This enables the classroom emphasis to be on discussions of theory and implementation, with student homework concentrating on actual spreadsheets.

A major problem with a computer-based course is how to structure the final examination. Two solutions seem to work well. One alternative is to have students (whether alone or in teams) submit a final project; examples might be a corporate valuation if the course is based on Section I of the book, an event study for Section II, an option-based project for Section III, or the computation of a bond-expected return if the emphasis is on Section IV. A second alternative is to have students submit, by e-mail, a spreadsheet-based examination with severe time limits. One instructor using this book sends his class the final exam (a compendium of spreadsheet problems) at 9 in the morning and requires an e-mail with a spreadsheet answer by noon.

Acknowledgments

I thank a number of people who have made materially significant comments to this edition:

Meni Abudy, Zvika Afik, Javierma Bedoya, Lisa Bergé, Elizabeth Caulk, Sharon Garyn-Tal, Victor Lampe, Jongdoo Lee, Erez Levy, Warren Miller, Tal Mofkadi, Roger Myerson, Siddhartha Sarkar, Maxim Sharov, Permjit Singh, Sondre Aarseth Skjerven, Alexander Suhov, Kien-Quoc Van Pham, Chao Wang, Tim Wuu.

Finally, I would like to thank: my editor John Covell of MIT Press, Ellen Faran, the Director of MIT Press, and Nancy Benjamin and her editorial team at Books By Design. They have all been unfailingly helpful and patient.

Disclaimer

The materials in this book are intended for instructional and educational purposes only, to illustrate situations similar to those encountered in the real world. They may not apply directly to real-world situations. The author and MIT Press disclaim any responsibility for the consequences of implementation.

From the Preface to the Third Edition

The two previous editions of *Financial Modeling* have received a gratifyingly positive response from readers. The combination of a "cookbook," mixing explanation and implementation using Excel has fulfilled a need in both the academic and the practitioner markets from readers who realize that the implementation of the finance basics typically studied in an introductory finance course requires another, more heavily computational and implementational, approach. Excel, the most widely used computational tool in finance, is a natural vehicle for deepening our understanding of the materials.

Acknowledgments

I want to start by thanking a group of wonderful editors: John Covell, Nancy Lombardi, Elizabeth Murry, Ellen Pope, and Peter Reinhart. My next thanks go to a dedicated group of colleagues who read the typescripts for *Financial Modeling*: Michael Chau, Jaksa Cvitanic, Arindam Bandopadhyaya, Richard Harris, Aurele Houngbedji, Iordanis Karagiannidis, Yvan Lengwiler, Nejat Seyhun, Gökçe Soydemir, David Y. Suk.

Many of the changes in this edition of *Financial Modeling* are due to the comments of readers, who have been assiduous in offering suggestions and improvements in the book. I follow a tradition started with the first two editions of *Financial Modeling* by acknowledging those readers whose comments have been incorporated into this edition:

Meni Abudy, Zvika Afik, Gordon Alexander, Apostol Bakalov, Naomi Belfer, David Biere, Vitaliy Bilyk, Oded Braverman, Roeland Brinkers, Craig Brody, Salvio Cardozo, Sharad Chaudhary, Israel Dac, Jeremy Darhansoff, Toon de Bakker, Govindvyas Dharwada, Davey Disatnik, Kevin P. Dowd, Brice Dupoyet, Cederik Engel, Orit Eshel, Yaara Geyra, Rana P. Ghosh, Bjarne Jensen, Marek Jochec, Milton Joseph, Erez Kamer, Saggi Katz, Emir Kiamilev, Brennan Lansing, Paul Ledin, Paul Legerer, Quinn Lewis, David Martin, Tom McCurdy, Tsahi Melamed, Tal Mofkadi, Geoffrey Morrisett, Sandip Mukherji, Max Nokhrin, Michael Oczkowski, David Pedersen, Mikael Petitjean, Georgio Questo, Alex Riahi, Arad Rostampour, Joseph Rubin, Andres Rubio, Ofir Shatz, Natalia Simakina, Ashutosh Singh, Permjit Singh, Gerald Strever, Shavkat Sultanbekov, Ilya Talman, Mel Tukman, Daniel Vainder, Guy Vishnia, Torben Voetmann, Chao Wang, James Ward, Roberto Wessels, Geva Yaniv, Richard Yeh, and Werner Zitzman.

Finally, I want to thank my very patient wife, Terry, who has maintained her own and my equilibrium through two books and a business school deanship in the past five years.

From the Preface to the Second Edition

The purpose of this book remains to provide a "cookbook" for implementing common financial models in Excel. This edition has been expanded by six additional chapters, covering financial calculations, cost of capital, value at risk (VaR), real options, early exercise boundaries, and term-structure modeling. There is also an additional technical chapter containing a potpourri of Excel hints.

I am indebted to a number of people (in addition to those mentioned in the previous preface) for help and suggestions: Andrew A. Adamovich, Alejandro Sanchez Arevalo, Yoni Aziz, Thierry Berger-Helmchen, Roman Weissman Bermann, Michael Giacomo Bertolino, John Bollinger, Enrico Camerini, Manuel Carrera, Roy Carson, John Carson, Lydia Cassorla, Philippe Charlier, Michael J. Clarke, Alvaro Cobo, Beni Daniel, Ismail Dawood, Ian Dickson, Moacyr Dutra, Hector Tassinari Eldridge, Shlomy Elias, Peng Eng, Jon Fantell, Erik Ferning, Raz Gilad, Nir Gluzman, Michael Gofman, Doron Greenberg, Phil Hamilton, Morten Helbak, Hitoshi Hibino, Foo Siat Hong, Marek Jochec, Russell W. Judson, Tiffani Kaliko, Boris Karasik, Rick Labs, Allen Lee, Paul Legerer, Guoli Li, Moti Marcus, Gershon Mensher, Tal Mofkadi, Stephen O'Neil, Steven Ong, Oren Ossad, Jackie Rosner, Steve Rubin, Dvir Sabah, Ori Salinger, Meir Shahar, Roger Shelor, David Siu, Maja Sliwinski, Bob Taggart, Maurry Tamarkin, Mun Hon Tham, Efrat Tolkowsky, Mel Tukman, Sandra van Balen, Michael Verhofen, Lia Wang, Roberto Wessels, Ethan Weyand, Ubbo Wiersema, Weiqin Xie, Ke Yang, Ken Yook, George Yuan, Khurshid Zaynutdinov, Ehud Ziegelman, and Eric Zivot. I also want to thank my editors, who again have been a great help: Nancy Lombardi, Peter Reinhart, Victoria Richardson, and Terry Vaughn.

From the Preface to the First Edition

Like its predecessor *Numerical Techniques in Finance,* the aim of this book is to present some important financial models and to show how they can be solved numerically and/or simulated using Excel. In this sense this is a finance "cookbook;" like any cookbook, it gives recipes with a list of ingredients and instructions for making and baking. As any cook knows, a recipe is just a starting point; having followed the recipe a number of times, you can think of your own variations and make the results suit your tastes and needs.

Financial Modeling covers standard financial models in the areas of corporate finance, financial statement simulation, portfolio problems, options, portfolio insurance, duration, and immunization. The aim in each case has been to explain clearly and concisely the implementation of the models using Excel. Very little theory is offered except where necessary to understand the numerical implementations.

While Excel is often not the tool to use for high-level, industrial-strength calculations (portfolios are an example), it is an excellent tool for understanding the computational intricacies involved in financial modeling. It is often the case that the fullest understanding of the models comes by calculating them, and Excel is one of the most accessible and powerful tools available for this purpose.

Along the way a lot of students, colleagues, and friends (these are nonexclusive categories) have helped me with advice and comments. In particular I would like to thank Olivier Blechner, Miryam Brand, Elizabeth Caulk, John Caulk, Benjamin Czaczkes, John Ferrari, John P. Flagler, Dan Fylstra, Kunihiko Higashi, Julia Hynes, Don Keim, Anthony Kim, Ken Kunimoto, Rick Labs, Adrian Lawson, Philippe Nore, Isidro Sanchez Alvarez, Nir Sharabi, Edwin Strayer, Robert Taggart, Mark Thaler, Terry Vaughn, and Xiaoge Zhou.

Finally, my thanks go to a wonderful set of editors: Nancy Lombardi, Peter Reinhart, Victoria Richardson, and Terry Vaughn.

0 Before All Else

0.1 Data Tables

Financial Modeling makes extensive use of data tables. I advise readers of the book to first make sure that they understand data tables (read Chapter 31, sections 1–5). Data tables are absolutely critical in the sensitivity analysis that is part of most financial models. They are a little bit complicated, but an invaluable addition to the modeling arsenal of the financial modeler.

In the remainder of this short chapter, I discuss **Getformula**.

0.2 What Is Getformula?

The Excel notebooks in *Financial Modeling*, fourth edition, contain a function called **Getformula** that aids in annotating your spreadsheets. In the example below, cell C5 shows the formula contained in cell B5; the formula in question computes the annual repayment of a loan of 165,000 for 7 years at 8%. Cell C5 contains the function **=Getformula(B5)**.

	A	B	C
2	Principal	165,000	
3	Interest	8%	
4	Term	7	<-- years
5	Annual payment	31,691.95	<-- =PMT(B3,B4,-B2)

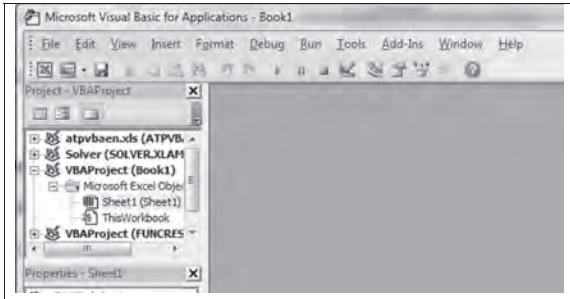
In this short chapter, we describe how to add this formula to your Excel notebook. Mac users: This works only in Excel 2011.

0.3 How to Put Getformula into Your Excel Notebook

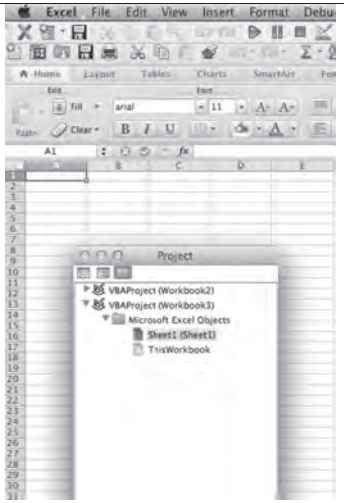
1. Open the Excel workbook in which you want the formula to work.
2. Open the VBA editor:

- On Windows computers: Press [Alt] + F11.
- On Mac (Excel 2011): Choose **Tools|Macro|Visual Basic Editor**

3. This will open the VBA editor.

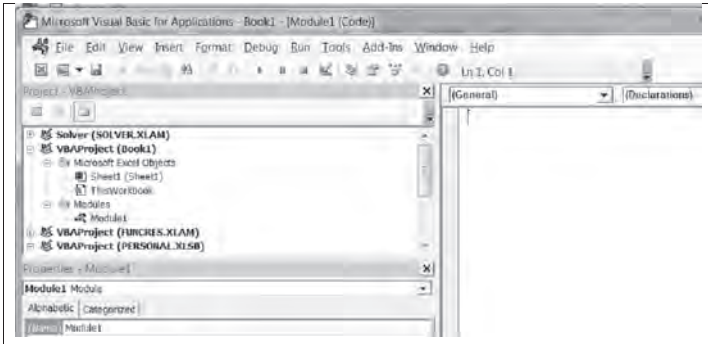
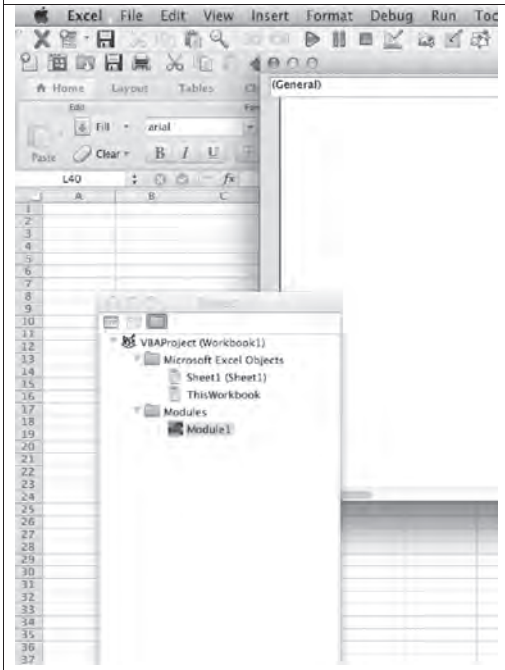


Windows screen



Mac screen

4. Select **Insert|Module** at the top of the screen.

	← Windows screen
	← Mac screen

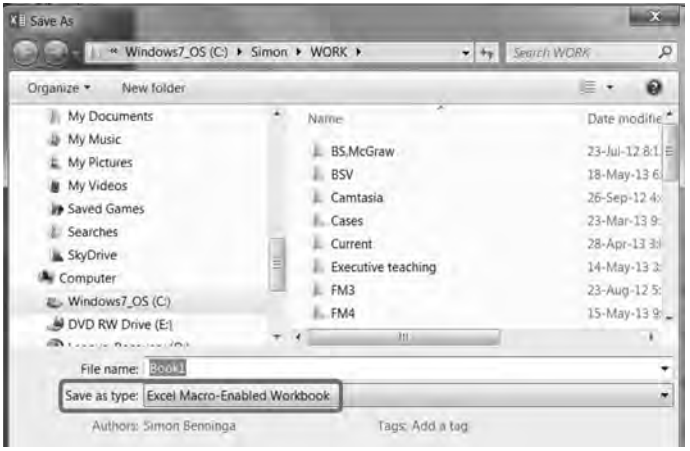
5. Now insert the following text into the Module window (where it says **General**). Just copy/paste the text below.

```
'8/5/2006 Thanks to Maja Sliwinski and
'Beni Czaczkes
Function getformula(r As Range) As String
     Application.Volatile
     If r.HasArray Then
     getformula = "<-- " & _
       " {" & r.FormulaArray & "}"
     Else
     getformula = "<-- " & _
       " " & r.FormulaArray
     End If
End Function
```

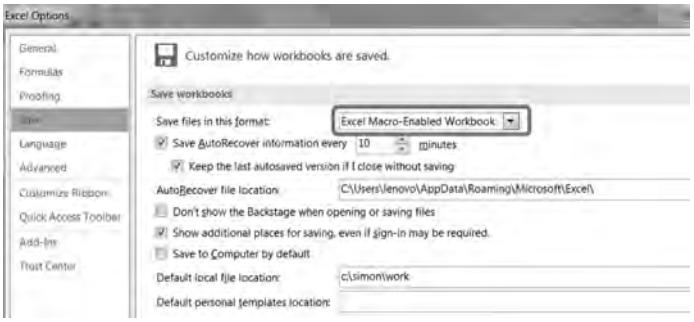
In Windows, close the VBA window (no need to save). On the Mac, just continue to work on the spreadsheet. The formula is now part of the spreadsheet and will be saved along with it.

0.4 Saving the Excel Workbook: Windows

To save the notebook with the **Getformula** macro in VBA, you will have to save it as a **Macro-enabled workbook**.

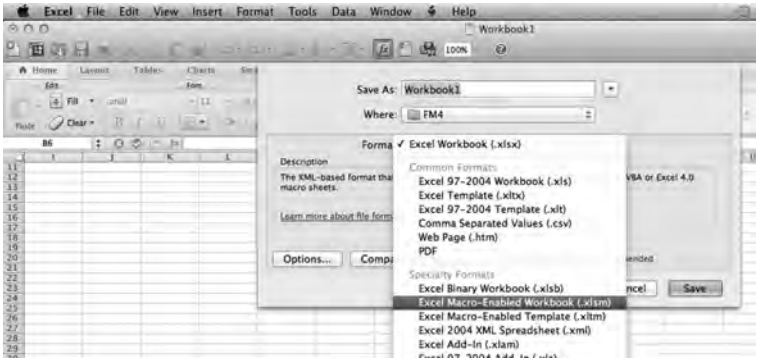


Macro-enabled workbooks have the extension .xlsm, whereas regular Excel workbooks have the extension .xlsx. Your users will never know the difference. We have changed our Excel settings (**File|Options|Save**) to make the Macro-enabled workbook our default:



0.5 Saving the Excel Workbook: Mac

The Mac screen for saving as a Macro-enabled workbook looks like this:

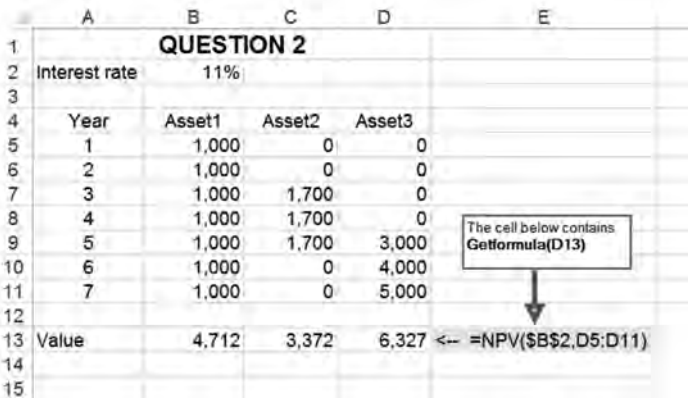


0.6 Do You Have to Put Getformula into Each Excel Workbook?

The short answer is "yes." You could create an add-in to Excel (see Chapter 39) that contains **Getformula**, but this will make it more difficult for you to share your workbooks. We prefer to put **Getformula** in each new spreadsheet we create.

0.7 A Shortcut to Use Getformula

Once you have put **Getformula** into your Excel workbook, you will have to use it! Ninety percent of our uses of this function point to the cell to the left of the formula itself:



We've put a short macro into our **Personal workbook** that automates this procedure. The remainder of this section describes how to automate the **Getformula** procedure.

Automating the Procedure

We want to automate this procedure of putting **Getformula** into a cell:

- Turn it into a macro.
- Attach a key sequence (in our case, [Ctrl] + t) to the macro.
- Make the macro and key sequence available in your Excel spreadsheets.

We will save the macro to our **Personal.xlsb** file. This file activates each time you start Excel. It's yours only—other readers of your spreadsheets won't see it. Below we describe the steps, for both Windows and the Mac.

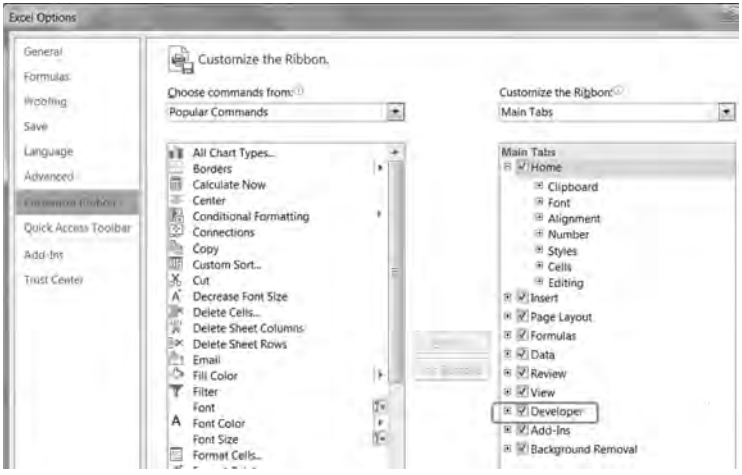
0.8 Recording Getformula: The Windows Case

Here are the steps to recording the macro in Windows:

- Activate the **Developer** tab on the menu bar.
- Use **Record Macro** to save a macro as a personal notebook.

Activate the Developer Tab

Go to **File|Options|Customize Ribbon** and activate the **Developer** tab as shown below:

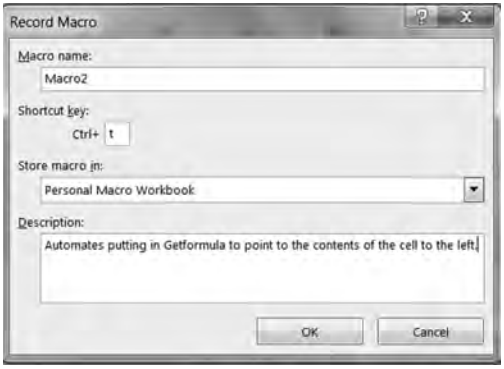


Use Record Macro

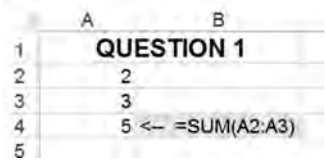
The Developer tab allows you to record a macro and save it as part of the Personal.xlsb notebook. We will illustrate with the copy as picture feature.

1. Open a blank Excel notebook and click on the **Developer** tab and then on **Record Macro**:

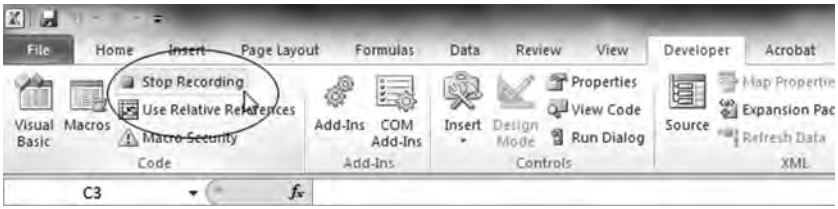
Excel will ask for details of the recording. Here's what I wrote. We will save this as a **Personal Macro Workbook** and then use the shortcut [Ctrl] + t:



2. Now go to your spreadsheet and use **Getformula**, pointing to the cell to the left of where you want **Getformula** to appear. In the spreadsheet below, we have typed =Getformula(A3) into cell B4:



3. Go back to the **Developer** tab and stop the recording:



4. Close down Excel. Excel will ask you if you want to save the Personal workbook. The answer is, of course, positive:

This creates the following file ("simon benninga" is of course my user name on my computer—you will substitute your user name).

C:\Users\simon benninga\AppData\Roaming\Microsoft\Excel\XLSTART\PERSONAL.XLSB

Using the Macro

From now on, whenever you open a file on *your computer*, you can use [Ctrl] + t to copy a region as a picture. Cool!