Chapter 1

Introduction

During the last few years, an increasing number of behavioral scientists have begun to use the World Wide Web as a tool for conducting psychological research. It is easy to understand the appeal of using "the web" for research purposes. Just about any study that can be conducted via traditional pencil-and-paper methods can be implemented on-line, but without the hassles of explicit transcription or data entry, the scheduling of research participants, and paper costs. Moreover, researchers who use computers in their experiments for manipulating visual or narrative stimuli, randomizing trials, or creating customized assessments can easily implement their studies on-line. Finally, although researchers can use the web simply as an efficient way to collect data from undergraduates in their departmental subject pools (as is often done in psychology), the web allows us to open our laboratory doors to participants from across the world.

Unfortunately, there are very few resources available to the behavioral scientist who wishes to create on-line research studies. One of the best books on the market, *Introduction to Behavioral Research on the Internet* by Michael Birnbaum, does an outstanding job at broadly reviewing many of the tools that one may use for Internet research, but does not focus on CGI programming—a technique that is useful for designing dynamic and interactive on-line research applications. Other texts, such as *CGI Programming 101* by Jacqueline Hamilton or *Perl and CGI for the World Wide Web* by Elizabeth Castro, are wonderful introductions to CGI programming but are not written with the research scientist in mind.

My objective in writing this book was to create a basic, step-by-step guide for behavioral researchers who are interested in using the Internet to conduct research. This book has been

written to be a one-stop shop, if you will, for moving from "square one" to the point at which you can create innovative and dynamic studies online. Any researcher with access to nothing more than a desktop computer, an internet connection, a healthy dose of curiosity and patience, and this book (of course) should be able to conduct on-line research within two weeks or less.

I thought it would be advantageous for researchers to have a thorough discussion of *one* way to do Internet research, a way that was simple, required as few monetary investments as possible, wouldn't assume excessive programming knowledge, and wouldn't require asking favors from your local computer guru. Therefore, in this book I focus on **server-side CGI programming in Perl**. (Exactly what this means will be explained in more depth below.) It is not my intent to review all the possible tools that one could use to conduct on-line research (e.g., Java, JavaScript, ASP, Perl, PHP, C). I have chosen a way that works for me, and have presented it here as simply as possible so that other researchers can take advantage of it. As a consequence, some of the programming code we discuss will not be the most beautiful or efficient code in the world. (If you don't know what it means to describe code as "beautiful," that is a good thing—you have not been corrupted yet.) The code we discuss, however, will be explained thoroughly and it will get the job done.

The intended audience for this book is anyone who currently conducts or is interested in conducting research in the social and behavioral sciences (e.g., psychology, sociology, epidemiology, economics, anthropology, marketing). As a psychologist, I have found it easiest to write this book as if I were writing it for my colleagues and students in psychology. However, many of the techniques and applications I discuss are not limited to psychology, and researchers from other disciplines should have no trouble understanding how these tools can be used to serve their purposes.

What You Will Be Able to Accomplish

It is my intention to provide you with the skills and knowledge you need to design a wide array of online research studies. To accomplish this, we will focus on many examples that illustrate what I consider to be the *generic components* of research projects—techniques that are common to a wide variety of research designs and areas of investigation. Here is a brief sampling of some of the generic skills that you will learn from this book:

- You will learn how to create web-based questionnaires involving rating scales, free responses, pull-down menus, and check lists.
- You will learn how to write programs that will store participants' data automatically in a text file—a file that can be imported easily into commonly used statistical packages and spreadsheets, such as Excel, S-Plus, SAS, or SPSS.
- You will learn how to provide response-specific stimuli or specially tailored stimuli for your participants. Moreover, you will learn how to write programs that process a participant's data and provide him or her with immediate feedback (e.g., whether he or she was correct, how he or she scored on a personality inventory).
- You will learn how to randomize the order of stimuli, whether those stimuli are images, text, or questionnaire items.
- You will learn how to randomly assign participants to conditions of an on-line experiment.
- You will learn how to carry a participant's responses forward from one web page to the next in studies that use multiple web pages.

- You will learn how to create online, Internet-grading quizzes for students. Also, I will show you how to create an online discussion forum with which you and your students or colleagues can exchange ideas.
- You will learn how to implement some basic graphing and data analytic techniques online.
- You will learn how to implement logins, personal identification numbers (PINs), and passwords so that the same participants can be studied over multiple sessions.
- You will learn how to assess reaction times online.

What You'll Need

The only "thing" you will need in order to get up-and-running is a simple desktop computer in your lab, office, or home. I will assume that your computer is connected to the Internet, and that you can easily use it to surf the web or check e-mail. I will also assume that your computer is running Microsoft Windows (95, 98, 2000, Me, XP) as its operating system. This, of course, is not a prerequisite for conducting research over the web, but this book would be more difficult to read if I were trying to explain how to do things for both PC users and Macintosh users. If you're an experienced Mac user, you should have no trouble adapting the instructions discussed in this book to a Macintosh environment. You will also need some special software that can be downloaded for free online, and I'll show you how to do so in the next two chapters. It should go without saying that you should already have some experience using the Internet. If you use the web to order books, check e-mail, or search for interesting articles, you'll be fine. I will also assume that you have a research background that includes a minimum of an undergraduate research methods and statistics course.

Some (Very) Basic Things You Need to Know about the Internet

There are some recurring terms and concepts you'll need to know in order to get the most out of this book. A **browser** is a program used to view web pages. The two most commonly used browsers are Netscape's Navigator and Microsoft's Internet Explorer. When you view a web page, several complex things take place beneath the surface that make this possible. First, when you type in a **URL** (Universal Resource Locator) or web address (e.g., http://www.web-research-design.net/index.htm), your computer sends a request to another computer "located" at that address. This computer is often called a **server**, and its job is to receive such requests, and then "serve" the requested information back to you—the **user**.

More often than not, the kind of information that is sent to your browser is in the form of **hypertext markup language** or **HTML**. HTML has become a standard way of sharing information over the internet. Your browser translates the HTML code that it receives from a server into the kinds of web pages with which we are familiar. The **World Wide Web** (WWW), by the way, is nothing more than the network of computers across the world that participate in this process.

In your typical day-to-day experience with the web, you probably do little more than view your favorite web pages or link from one page to another in hopes of discovering something new (or at least something entertaining). Sometimes, however, your web experience might be more complex than this. You might, for example, use the web to manage your e-mail accounts or you might order a book or a CD from on on-line retailer. In this later case, it should be clear that the server is doing something much more complex than simply "serving" you the same old HTML files that it serves everyone. It might, for example, be storing your shipping address or tracking items you've purchased in the past in order to make recommendations for other products you might enjoy. The pages you see in these cases are typically *created* "on the fly," just for you.

In short, the sever can be used to perform a number of tasks that make a user's web experience highly dynamic and interactive. In this book, I'm going to show you how to make the server behave in these dynamic ways. Therefore, it is necessary to introduce some useful sever-specific terminology from the get-go. The programs that we will write for the server are often called **CGI scripts**. "CGI" is an acronym for **Common Gateway Interface**, a method or protocol by which the server interacts with other software on the server (e.g., databases), as well as other computers on the web. There are a number of programs and programming languages that can be used for CGI programming (e.g., Cold Fusion, PHP, ASP, C+). In this book, we're going to focus on Perl. **Perl** (Practical Extraction and Report Language) is a highly versatile programming language and it is one of the most popular languages for CGI scripting.

There is one final distinction that I should make because it will help you better understand how the things we'll be doing fit into the broader Internet context. In this book we'll be focusing on **server-side** programming—writing CGI programs that run on the server. **Client-side** programming, in contrast, involves writing programs that run on the user's (sometimes called the **client's**) browser. You've probably heard of some common client-side programming languages before, such as JavaScript. **JavaScript** is a special kind of code that can be embedded in an HTML document. When that file is delivered to the user's browser, the JavaScript program is executed by the browser and all the relevant computations (e.g., tallying up a total price) are performed by the user's computer. The advantage of using JavaScript is that all the computation is done by the user's computer. This frees up the server from having to perform resource-consuming functions for multiple users at once. The downside of JavaScript is that is different browsers do not always interpret the same JavaScript code in the same way. (In fact, different versions of the same browser do not even interpret the same JavaScript code in the same

manner!) Sometimes you might need to write separate JavaScript programs depending on whether the user is working with Netscape Navigator or Internet Explorer. Moreover, some users have the JavaScript option turned off on their browser. Most importantly, JavaScript cannot be used to save data easily, making it virtually useless for serious research purposes. Having worked with both server-side and client-side programs, I believe that server-side programming is better tailored to the needs of the behavioral scientist.

How to Get the Most Out of This Book

In Chapter 2 I will show you how to get a web server up and running. We'll discuss two distinct ways for getting access to a web server and you will need to chose which method is best for you. One approach is to transform one of your spare PCs into a web server. This is much easier than it may sound, and I'll walk you through each step to help ensure that the process goes smoothly for you. One of the advantages of this approach is that, if you have a spare PC, transforming the computer into a server is cost-free. Another approach is to find a professional web hosting service and use their web server for your research. It is possible that your university or department has a server that can be used for research purposes, but, if not, there are many professional web hosting companies that will allow you to build your research web pages on their computers either free of charge (if you agree to allow those companies to display small advertisements on your web pages) or for a trivial fee (e.g., anywhere from $5 to $20 a month, depending on your needs). The primary advantage of this approach is that it will require no server maintenance on your part. In fact, if you have the extra money to spend, I would strongly recommend this approach for the beginner. I would encourage you to skim through Chapter 2, decide which approach seems best suited to your needs, and then follow the steps for that method in order to get your server up and running.

After you successfully set up your server in Chapter 2, I encourage you *to work through each example* as you read the chapters. Learning how to write programs is a bit like learning how to drive a car. Although you can read "Driving for Dummies" and become quite knowledgeable about stick shifts and parallel parking, you won't actually acquire the necessary skills until you're sitting in the driver's seat. Designing web experiments is a skill, and your ability to perform this skill well will depend less on memorizing the right kinds of facts and details (you can always look those up in a reference manual) and more on acquiring a certain degree of familiarity and intuition concerning the "logic" of programming. This book is a "beginner's guide" not because it shows you how to accomplish easy tasks in an easy way; the book is a "beginner's guide" because I have done my best to walk you through a highly complex terrain without assuming you know your way around. With a fair amount of practice, you'll come to know this terrain well.

I encourage you to work through one chapter a day. As you read each chapter, you should enter the programming code and see what it looks like on an actual web browser, such as Microsoft's Internet Explorer. To make this easy for you, all of the code that is presented in this book can be copy-and-pasted from the web site for this book: http://www.web-research-design.net. Moreover, "live" demonstrations of the various studies and applications that we discuss can be accessed at that site. Be sure to tinker with the code a bit and see if things change in the way in which you expect. Finally, after working your way though a chapter, take a break and then reread the chapter. It is easy to lose sight of the Big Picture when you're wading through the minutia of programming code. Be sure to take a step back, reread the chapter, and reflect on the skills you're learning and how they may be applied to different contexts.

At the end of each chapter I have included, where relevant, a table that contains all of the new HTML or Perl/CGI code that was introduced in that chapter. As you'll observe from thumbing through the book, I introduce the majority of the programming code early in the book, in Chapters 3 through 7. In fact, the latter half of the book uses very little new code. One of the things you'll discover as you begin to learn HTML and Perl is that you don't need to know much in order to accomplish a lot. Many of the complex techniques that we'll discuss later in the book, such as tracking a user's responses over multiple sessions, involve finding novel and creative ways to use the code we already know rather than learning new commands. With this in mind, as you read the book I encourage you to think "outside of the box" and try to envision some of the ways in which the skills you are learning can be reconfigured to address research scenarios that are not discussed here.

I have also created some on-line quizzes that you can take to test your learning (see the web page for this book: http://www.web-research-design.net). These quizzes, which are discussed in detail in Chapter 14, will automatically score your answers and give you feedback on your performance. Unlike the quizzes described in Chapter 14, however, they will not tell you the correct answer when you get a question wrong. At the end of each chapter, you may want to take the corresponding quiz multiple times until you have maximized your performance.

With the exception of Chapter 15, I have deliberately organized the chapters in a manner that will allow you to build on skills that you have acquired earlier in the book. Thus, I recommend that you read through the chapters *sequentially*. Even if you're not interested in randomly assigning your participants to different conditions (Chapter 8), the skills you learn in that chapter will be critical to understanding things that may be of interest to you in later chapters. In short, it may seem that you can pick and choose among the chapters in this book, but

I strongly discourage you from approaching the book in that manner. I have written the chapters in a fashion that allows you to gradually build your programming skills, so you should tackle each chapter in turn—even if it focuses on a research technique that seems tangential to your interests.

I will adopt a few conventions throughout this book that should make your reading and programming experience easier. First, all code, whether it be HTML or Perl code, will be printed in `Courier typeface`. Moreover, whenever a sizable portion of code (e.g., a CGI script) is being presented, I'll place it within a box so it can be easily separated from the rest of the text. Finally, I'll often make parenthetical comments or important qualifications throughout the text. Instead of placing these in footnotes (where they are easily overlooked), I have placed them in gray boxes.