This chapter describes how to build Web pages to administer, score, and provide feedback for self-report scales. Included in this chapter are samples of HTML and Perl code for displaying scale items on the Web, prompting participants who skip items, instantly checking for nonresponsiveness (responding inconsistently, randomly, or with strings of the same response category without regard to content), storing item responses for later analyses, computing scale scores, and producing feedback in the form of a narrative report. Shorter examples of code appear directly in this chapter; longer examples can be accessed at the supplementary Web site for this chapter as described in the Additional Resources section at the end of the chapter.

Self-reports sum a person’s responses over a set of items, resulting in a single aggregated score that measures the degree or level of a single psychological state or trait. Although self-reports are often referred to as “questionnaires,” they should not be confused with survey or opinion questionnaires (see chap. 12, this volume), which treat responses to each item separately rather than in aggregates. Self-reports are also often called “tests” but should not be confused with ability or aptitude tests (see chap. 9, this volume), whose items are problems to be solved rather than statements with which participants judge applicability or agreement.
Implementing a Self-Report Measure on the Web

Anyone who possesses the knowledge of Web page authoring described in chapters 2 through 6 of Fraley (2004) can create Web-based self-report scales through the following steps:

- Consider what you want to accomplish with Web-based assessment;
- choose (a) self-report scale(s) that will allow you to accomplish your goals;
- decide on recruiting strategies for obtaining participants;
- design the front page that invites participants to complete the scale;
- write and test files that present the items to participants;
- write and test script files for error checking, if desired;
- write and test script files that score the scale and record the results; and
- construct a routine for providing narrative feedback, if desired.

Each stage involves choosing from options according to one's purposes. Choices at each stage impact later stages. Next, I describe some of these options at each stage and how to implement them.

WHAT DO YOU WANT TO ACCOMPLISH?

Being clear about what you want to accomplish helps you to make good decisions in each of the remaining steps of Web-based measurement. One simple research goal for self-reports is to ascertain a single measure's psychometric properties, such as item endorsement frequencies, distribution of scores, scale reliability, and factor structure. Additional simple research goals include examining the scale's relation to demographic variables, such as age, sex, and ethnicity. If your goals fall into the category of simple research, your self-report scale is short, and you are not gathering a lot of other information about participants, you can present all of your items on a single Web page. However, if your goals are more complex (e.g. assessing construct validity by examining relations among several constructs with a longer, multiscale inventory or several different self-report scales), you will want to write scripts that spread the items across multiple Web pages. If you wish to have participants return to your site at a later time to retake your measure or to complete other measures, or if you are gathering information on the participants from informants (see chap. 11, this volume) or other sources, you will need to create an identifier for participants that allows you to match scores from the same participant while protecting his or her privacy.
Your goals will therefore determine the number of Web pages you will need to construct and what kind of identifying information to record. Your goals will also determine your recruiting strategy for inviting participants to complete your Web-based measure.

CHOOSE SELF-REPORT SCALES

Other than the standard concerns about conceptual relevance, reliability, and validity, the major consideration in choosing self-report scales is that they can be placed on the Web without violating copyright laws. Commercial test publishers rarely grant permission to reproduce their scales on the Web; authors of noncommercial measures are far more likely to do so. Researchers who wish to avoid altogether the bother of obtaining permission to place scales on the Web will find hundreds of scales in the public domain at the International Personality Item Pool (IPIP) Web site, http://ipip.ori.org (Goldberg, et al., 2006).

DECIDE ON RECRUITING STRATEGIES

Options for inviting participants fall on a continuum of active versus passive recruiting. Actively pursuing participants is useful when you are interested in collecting a modest amount of data in a relatively short period of time. A more passive approach involves waiting for Web surfers to discover your site and allowing knowledge of the site to spread by word of mouth. Researchers who are not in a hurry can recruit hundreds of thousands of participants in a few years’ time with this approach. A mixed strategy involves recruiting a certain number of participants at the outset and then leaving the site up for additional people to add their responses to the data pool.

If you wish to actively recruit individuals with particular traits or interests (e.g., shy people; people who are passionate about politics), you can announce your research study in relevant Internet communities. Web sites for locating online communities are listed in the Additional Resources section at the end of this chapter (see also chap. 7, this volume). Ethical considerations (see chap. 16, this volume) suggest caution about intruding on the privacy of people who did not join a community to become a research participant. Researchers may do well to observe a community for a while to get a sense of whether its members would be receptive to an invitation to participate in research (Eysenbach & Till, 2001).

The more passive technique of waiting for participants to discover your Web-based questionnaire does not mean doing nothing. It is important to design your pages so that they can easily be indexed by major search engines such as Google. General good advice for optimizing your pages for search engines and specific guidelines for registering with
Google, Yahoo!, and MSN can be found at http://www.searchengines.com/. Web page optimization increases the chance that individuals who search for key phrases such as “anxiety research” will find your site because it was properly indexed by the major search engines. In addition to registering your Web site with the major search engines, you should also consider listing it with Krantz, who maintains an up-to-date list of online research studies at http://psych.hanover.edu/Research/exponnet.html.

Sometimes concerns are expressed about whether one can obtain “representative” samples from the Internet by any recruiting strategy. This concern is usually misguided. Aside from the fact that most Internet samples are at least as diverse as the frequently used college student samples (Gosling, Vazire, Srivastava, & John 2004), obtaining a representative sample is unnecessary for research that tests hypotheses about relations among variables, unless one has reason to believe that participants in the sample have unique characteristics that interact with the variables being studied (Leary, 2004). Sampling is more an issue for survey research (see chap. 12, this volume), when one wants to accurately describe the thoughts, feelings, or behavior of a particular population, in which case the sample needs to represent the population.

DESIGN THE FRONT PAGE

The front page (or home page) for your Web-based self-report serves two functions. The first function is to persuade potential participants to complete your measure. Persuasiveness is particularly important if you are using a passive recruitment strategy, but even if you are actively directing participants to your site, the front page should be welcoming rather than forbidding. Fraley (2004; pp. 285–289) offered good advice concerning effective design for a front page. Here I emphasize just two of Fraley’s points. First, you must strike a balance between making the page look professional and scholarly yet flashy enough to be eye-catching. The result should appeal to potential participants’ desire to learn more about themselves. A good example of such a site is the popular and successful http://www.outofservice.com. Also, your front page should be a base from which participants can branch to any of a number of online studies that you or your lab group is currently conducting. This way, as studies come to completion, participants will still be able to find your site.

The second important function of a front page is to obtain informed consent from participants. Requirements for obtaining informed consent differ across universities and agencies, so the details will depend on the regulations of the organization that is supporting your research. For example, for the self-report scales used at http://www.outofservice.com, the front page contains a link to the site’s privacy policy, and the pages for different scales contain a link labeled “Read our consent form” that
directs participants to a very short and simple explanation of risks and benefits. By completing the scales, participants are giving implied informed consent. Other organizations require researchers to channel participants through a more involved, legalistic Web page and have participants actively give informed consent by clicking a button before they can complete the scale. For examples, see Revelle’s form at http://test.personality-project.org/survey/consentform.html and one of my forms at the supplementary Web site for this chapter.

WRITE FILES FOR PRESENTING ITEMS

Two kinds of files can be used to present Web pages to your participants. The first is a file that contains fixed content and structure (defined by HTML tags) such that it always looks the same. I refer to these files as straight HTML files. The second type of file is a script file. Script files are written in programming languages such as Perl, PHP, and JavaScript. Script files can receive data, do calculations, and generate HTML files based on those calculations. This chapter shows examples of script files written in Perl, as this is the scripting language used in Fraley’s (2004) introduction to Internet research methods. Again, readers unfamiliar with HTML should read chapters 3 and 4 of Fraley’s book, and readers unfamiliar with Perl should read chapters 5 and 6.

Your front page, which represents the entry point for participants completing any of your self-report scales, should be a straight HTML page with a fixed URL so that search engines can find and index your site properly. Typically, this page will not contain self-report scale items. Rather, it will contain short descriptions of different scales with links that take you to other Web pages that contain the informed consent forms for the scales or to the scale itself (when informed consent is handled by a link to a separate page). For example, http://www.personal.psu.edu/~j5j/IPIP/ is a front page that directs people to either a 300- or 120-item IPIP representation of the Revised NEO Personality Inventory (NEO PI-R; Costa & McCrae, 1992); this representation is referred to as “IPIP-NEO” hereinafter.

My IPIP Web site is currently designed to educate visitors about personality assessment rather than to test research hypotheses, so following either link from my front page takes people to a disclaimer page rather than an informed consent form. Nonetheless, this disclaimer page illustrates two features that one might use on an informed consent Web page. One is a pair of checkboxes that must be ticked to access the actual self-report scale. The first checkbox requests the participant to accept responsibility for the time invested in completing the inventory, regardless of the results. The second checkbox asks the participant to acknowledge an understanding of the limitations of the results. The precise wording adjacent to the checkbox for your own informed consent page
would be whatever is required by your human subjects committee (see chap. 16, this volume). The checkbox is a useful tool to increase the probability that participants will actually read a consent form or any instructions or information you would like them to read.

The second feature I use on my disclaimer page that can also be used in an informed consent Web page is the HTML interactive form, which will send the status of the checkboxes to a script file named "shortipipneo1.cgi," which will either begin presenting the self-report scale items (if the boxes are checked) or halt and send a warning to the participants that they must give consent by checking the boxes to begin the self-report scale.

To see the disclaimer page for the 120-item IPIP measure, please visit the supplementary Web site for this chapter. You can see the warnings sent to participants by clicking the "Send" button without ticking the checkboxes. To see the underlying HTML code that generates the checkboxes and interactive form, use the "View Source" command in your browser.

Once your script file has verified that the person has given informed consent or has read the preliminary instructions, it will next generate HTML to present self-report items. Choices for presenting the items and possible uses include

- text boxes or text areas (for entering a code name or for responding to open-ended items),
- checkboxes (for yes–no items such as those on a mood or activity checklist),
- pull-down menus (for choosing one item from a list, such as ethnicity or country of origin), and
- radio buttons (for responding to items on a scale of, say, 1–5).

The HTML generated by shortipipneo1.cgi includes text boxes for recording a nickname and the person's age, radio buttons for indicating whether the person is male or female, and a pull-down menu for indicating one's country. The Perl code for generating this HTML can be examined at the supplementary Web site for this chapter.

This chapter focuses on the presentation of self-report items where responses are recorded on a 1–5 scale with radio buttons. The format I describe here presents a limited number of items (no more than 60) per Web page in an HTML table. Presenting more than 60 items at a time may overtax some systems, causing a program crash (Johnson, 2001).

Some Web designers have criticized the use of tables to control the layout of content on a Web page, arguing that this should be accomplished instead with Cascading Style Sheets (CSS). Nonetheless, the ease of using tables can outweigh the alleged advantages of CSS (Budd, 2004). This is especially true when the table is simple, as in the case of laying out items and response options in a self-report scale. Furthermore, whereas
Web-Based Self-Report Personality Scales

Three possible error checking by Web-based self-report personality scales: (a) table critics are usually referring to borderless tables used to control layout, for self-report scales I use borders to visually connect each item to its radio button responses. With this visual connection, participants are less likely to accidentally use the radio buttons for the previous or following item. Items appear in the leftmost column of the table, and five radio buttons with the response anchors appear in the five adjacent columns. Placing the anchors immediately above all radio button rather than only at the beginning of the scale ensures that the response option meanings will remain visible as the participant scrolls down the page.

If you look at the HTML in shortipipneo1.cgi for generating items, on the supplementary Web site for this chapter, you will see that all five radio buttons for the first item have the same name, Q1, but that the value assigned to the buttons varies from 1 to 5 to indicate how accurately the item describes the participant. Reverse-scored items can be assigned decreasing values from 5 to 1, if desired. Alternatively, responses can be reversed later when scale scores are computed. One advantage to not reverse-scoring items at the level of collecting data is that it facilitates the counting of consecutive identical responses, which can be a sign of non-attentiveness to content (Johnson, 2005). This is explained further in the next section on error checking.

After the HTML for the 60th item, the table is closed with the table end tag, "</table>,” and followed by HTML for a “Submit” button with a message that clicking the button will take the user to the next 60 items. The beginning of shortipipneo2.cgi, the script file to which the participant’s responses are sent, contains the first of several error-checking routines that are described in the next section. The Perl code in shortipipneo2.cgi can be examined on the supplementary Web site for this chapter.

WRITE SCRIPT FILES FOR ERROR CHECKING

One elementary error researchers will want a script to catch is when a participant fails to provide required information. The IPIP-NEO produces a narrative report based on sex and age norms and may eventually use norms based on nationality, so responses to these demographic items are required. The line of Perl code in the shortipipneo2.cgi script that checks whether participants have identified themselves as male or female reads:

```perl
if ($Sex ne "Male" && $Sex ne "Female").
```

If the statement is true (i.e., the person has not identified as male or female), the program stops and prints a message indicating the need to provide that information on the previous page in order to continue. Similar checks are conducted for the age and nationality variables. One
could also check to see that a participant has completed each item in the scale, although you might only advise participants about missed items rather than requiring them to complete all items, as human subjects committees sometimes want to give research participants the option of not responding to individual items.

An alternative to checking for missing responses in a subsequent Perl script is to use JavaScript within the original Perl script. When a participant clicks the “Submit” button, a JavaScript form validation routine checks for missing responses and presents a message either on the page or in a pop-up window informing the participant about the incomplete items. Fraley (2004) advised against using JavaScript because of incompatibilities across different browsers and because many people disable JavaScript to avoid annoying pop-up ads. Also, there are ways to get past JavaScript checking, so if you are serious about catching missing responses, you will have to use techniques such as the Perl illustration above in addition to using the JavaScript. Nonetheless, interested readers can search the Web for “JavaScript form validation” for more information on this topic.

If you have given participants unique identifiers to match data over multiple sessions or with other sources of information (e.g., from informants; see chap. 11, this volume), you will want to ensure that the identifiers (and passwords, if you are using them) are the ones that you have assigned participants. This can be accomplished with a script provided by Fraley (2004, chap. 12). If you do not need to actually track participants but you want to associate some kind of nickname or identifier with them, you can use a subroutine written by Hunt (1999) that generates a semi-random but pronounceable string of any length. My script, shortipipneo2.cgi, calls this subroutine to create a 23-letter identifier when a participant leaves the nickname box blank. This allows me to check for duplicate submissions (Johnson, 2005). The script also shortens user-generated nicknames longer than 23 characters to a 23-character identifier, and it pads nicknames shorter than 23 characters to a 23-character string. This maintains lines of uniform length in the file where data are saved.

Other kinds of error checking can take place only after all of the items in the self-report have been answered. In my own research, I have used these error-checking techniques after downloading completed protocols from all participants (Johnson, 2005), but one could easily use them as part of the standard scoring procedures, to which I turn next.

WRITE SCRIPT FILES FOR SCORING SCALE AND RECORDING THE RESULTS

Although Perl was created primarily for the manipulation and formatting of text, it does contain sufficient computational features to score a self-report scale, test for statistical anomalies, standardize scores, and
graph the results. To accomplish these activities, the script files that present the scale items must also pass the responses entered by participants to subsequent script files until the final script file does the scoring and statistical analyses. Passing inputted data to subsequent scripts is achieved with the “Input Hidden” tag (Fraley, 2004).

After shortipipneo2.cgi receives the input from shortipipneo1.cgi, it generates the HTML for the final 60 items. When the participant responds to these items and clicks the “Submit” button, shortipipneo2.cgi passes, by way of “Input Hidden” tags, the demographic information and responses to all 120 items to shortipipneo3.cgi, which reads the information, stores responses to all 120 items in a data file for creating norms, scores the IPIP-NEO, standardizes the scores, and provides feedback to the participant. This script, which can be examined at the supplementary Web site for this chapter, uses essentially the same process described in chapter 5 of Fraley (2004) for saving data to a file, so that procedure is not described here. The remainder of this section describes the process of scoring and standardization. It also describes some final error-checking routines that could be implemented, if desired. The production of narrative feedback is discussed in the final section of this chapter. Space limitations preclude the reproduction of code from shortipipneo3.cgi in this chapter, so the reader may wish to print a copy of the code from the supplementary Web site to more easily follow the discussion that follows.

The 120-item IPIP-NEO is scored for 30 facet scales similar to the facets of the original NEO PI-R. These facet scores are then combined to yield five domain scores representing the major factors of the five-factor model (Costa & McCrae, 1992). Fraley (2004) showed how summing or averaging items into a single scale score is a very simple matter. One need only compute a total score variable with a Perl statement such as

\[ \$\text{total} = \$q1 + \$q2 + \$q3 + \$q4 + \$q5; \]

and the new variable, $\text{total}$, will represent the sum of the five items. If any of the items (say, $\$q4$) is a reversed-keyed item, and the values assigned to the radio buttons were not reversed, the proper value for such items is derived by subtracting them from the quantity $(1 + \text{the value of the highest possible response})$, which in this case would be $(6 - \$q4)$.

For a multiscale inventory such as the IPIP-NEO, however, one can take advantage of looping in a script to score all of the scales with one short set of instructions when the items on all scales are separated by the same number of items. Five simple lines of code from shortipipneo3.cgi, labeled “# Score facet scales” can score all 30 facets because items on the same facet scale occur every 30 items on the inventory. For example, items 1, 31, 61, and 91 form the Friendliness facet scale. Values for these items are summed into the variable $\text{ss}[\text{1}]$ by incrementing the subscript of the array @Q (which contains all 120 item responses as variables Q[1])
to $Q(120)$) by 30 three times (as $j$ increases from 0 to 3). The full loop is repeated 30 times, as $i$ increases from 1 to 30, generating 30 facet scores, $ss1$ to $ss30$, in the array, @ss. (For a review of scalar and array variables, see Fraley, 2004.)

For mnemonic convenience, shortpipneo3.cgi uses a similar looping procedure labeled “# Number each facet set from 1–6” to assign the 30 facet scores to new array variables, @NF, @EF, @OF, @AF, and @CF, referring to the Neuroticism, Extraversion, Openness, Agreeableness, and Conscientiousness facets, respectively. Finally, the facets scales are summed to yield the domain scales.

The standardization of the facet and domain scale scores is achieved in shortpipneo3.cgi under the label “# Standardize scores” by including several arrays of means and standard deviations for males and females of different age groups. The norm array chosen for an individual is based on an if-statement. For example, the if-statement that begins “if ($Sex eq "Male" && $Age < 21) {@norm = “ defines the @norm array for male participants under 21 years of age. To make later computations simpler, a zero is placed as the first element in the array because Perl indices begin with 0 rather than 1, viz, $norm[0], $norm[1], $norm[2], and so on. The 10 elements in the first line of the array following the zero are means for the N, E, O, A, and C domain scores followed by the respective standard deviations for those scores. The next line contains means and standard deviations for the six facets of Neuroticism, and so forth. The variable “$id” is created to identify the comparison group (e.g., males under 21 years of age) for the narrative report.

The arrays for norms in shortpipneo3.cgi are based on an Internet sample of over 20,000 individuals (Johnson, 2005). One can create dynamic norms, however, by having a scoring script open a continuously updated data file, compute means and standard deviations based on the most recent data, and place these values into a @norm array. Fraley (2004, chap. 6) explained how this can be accomplished.

Next, shortpipneo3.cgi standardizes the raw scores into $t$ scores by applying the usual formula, using the means and standard deviations in @norm. Again, the Perl script takes advantage of the systematic ordering of facets by using a loop. To present feedback to participants, the shortpipneo3.cgi script next estimates percentile scores from the standard $t$ scores, on the basis of a cubic relation between the two distributions under normality. Discussion of the estimation is beyond the scope of this chapter. However, one could also compute actual percentiles on the basis of accumulated data—dynamically, if desired—and report results in those terms.

Before turning to the production of narrative feedback, I describe additional error-checking measures that one could implement, if deemed appropriate. In my own research (Johnson, 2005) I normally apply these algorithms with statistical software after data are downloaded to
screen out duplicate and otherwise problematic protocols. For demonstration purposes, I describe below how these procedures can be accomplished in real time with Perl so that problematic cases are not even added to one’s database.

**Missing Values**

Entire books (e.g., Little & Rubin, 2002) have been written about options for handling missing data, some of them quite complex. The optimal strategy may be to preserve the encoding of all missing values as zero (the default in HTML forms when no data are entered) for your data file, to be handled later when you conduct data analyses, while using a simple method for dealing with missing data (e.g., replacing missing values with the midpoint of the rating scale) when you score a scale for feedback. A suggested first step is to simply count how many items were skipped with an incremental counter. You can use your own judgment to determine how many items can be skipped before you refuse to score the scale. For the 300-item version of the IPIP-NEO, 75.6% of the 20,000+ protocols had fewer than 3 missing responses, and the frequency curve showed a sharp decrease in cases after 10 missing responses, suggesting a cutoff at that point (Johnson, 2005). For protocols with fewer than 11 missing responses, the missing values can be replaced with the rating scale midpoint, 3, and a warning about the replaced values can be included in the narrative report. (Remember that the script should save the entire protocol to the data file before the missing values are replaced.) The demonstration script file at the supplementary site for this chapter contains a routine to handle missing values following the label “# Count missing values.”

**Excessive Use of the Same Response Category**

Especially for longer, multiscale measures, a respondent will occasionally ignore the content of items, either across the entire measure or for a set of items, and will simply choose the same response option repeatedly to get through the measure quickly. For my large Internet sample (Johnson, 2005), I plotted frequency curves for each response category, showing the number of respondents who used the same response category twice in a row, three times in a row, and so on, and identified breaks where the frequency dropped off sharply. This suggested maximal strings of 9, 9, 8, 11, and 9 consecutive uses of each of the five response categories. These values are identical or similar to the values observed by Costa and McCrae (in press) for their NEO-PI-R. Researchers with access to data from a large sample who completed their scale can use either method to set cutoffs; otherwise, they must set some arbitrary limits. Perl code that could be used to flag excessive use of the same
response category appears in the demonstration script file at the supplementary Web site for this chapter under the label "# P array unreverse-scores the reverse-keyed items." (Note that because my Likert scales were set up to code reverse-scored items as 5-1 rather than 1-5, I had to create a new array of item responses, $P[1]$ to $P[120]$, with the reverse-scored items recoded to reflect the actual response category used by participants.)

Protocol Inconsistency

Without some minimal level of consistent responding to items, a protocol is invalid and uninterpretable. Item response theory (Reise, 1999) offers models of response consistency that could, in principle, be used to examine response consistency on Web-based self-report scales. However, these models may be too complex and computationally intensive to implement in Perl as a real-time screening mechanism. In Johnson (2005), I described two simpler methods for quantifying consistency of self-expression on self-report scales, one described by Jackson (1976) and one described by L. R. Goldberg (personal communication, June 20, 2000). Both methods require relatively long (more than 30 items) measures, and the Jackson method further requires that the measure be a multiscale inventory. Because the methods cannot be used for short scales and because the program code for implementing them in the IPIP-NEO is lengthy, these measures are described only briefly here. Further details can be found in Johnson (2005). I currently apply these consistency algorithms to my data after they are downloaded rather than online. However, for demonstration purposes, Perl code for computing these consistency indices for the 300-item IPIP measure can be found at the supplementary Web site for this chapter.

Jackson's (1976) method involves forming two half-scales from every scale. Scores on the half-scales are correlated with each other across all half-scale pairs and the result corrected with the Spearman–Brown formula. Although it takes many lines of program code to compute a correlation coefficient from the half-scale scores, sums of squares, and sums of products, the process is straightforward. The range of Jackson coefficients representing acceptable internal consistency is discussed in Johnson (2005) and can be used to issue warnings in the narrative report or to set flags in one's data file.

L. R. Goldberg's (personal communication, June 20, 2000) method requires that one first intercorrelate all item responses to one's measure from a relatively large existing data set. The 30 unique item pairs with the highest negative correlations are identified, preferably items that are not forward-scored and reverse-scored items from the same scale. These items therefore represent item pairs that people generally answer in opposite directions. To calculate an individual's tendency to be consistent
with general response patterns, program code computes the correlation across item pairs for an individual. A highly negative correlation indicates consistency with general response patterns. Again, interpreting the magnitude of this correlation to identify insufficiently consistent responding is discussed in Johnson (2005).

**Identifying Duplicate Protocols**

Although not a problem for producing meaningful feedback to an individual, duplicate protocols are undesirable in the database one is building for research. Duplicate protocols can be prevented by insisting that participants obtain a user ID and password before participating, but this disallows for anonymous participation. Fraley (2004) suggested using an initial item in which participants indicate whether they have participated before. This method is used at http://www.outofservice.com. Unfortunately, it will not detect multiple submissions that occur with repeated clicks of the final “Submit” button or reloading previous pages and resubmitting (Johnson, 2005). My preference has been to detect duplicates after the fact by identifying protocols submitted with the same nickname within a short period of time, with a suspiciously large number of identical responses, or both. This method is described in detail in Johnson (2005). Nonetheless, researchers may decide to block duplicate protocols from entering the database with a Perl screening routine that opens the data file, compares all information in the current protocol to all existing cases, and decides whether the information is similar enough to reject the protocol as a duplicate.

**Social Desirability and General Considerations About Invalid Responding**

One long-standing question in assessment by self-reports is whether people respond to items honestly or whether they respond in a way that they consider to be socially desirable (Paulhus, 2002). Researchers eventually realized that there are individual differences in the motive to present oneself in a socially desirable fashion. Thus, social desirability scales were constructed as checks on protocol validity. Such scales are freely available at the IPIP Web site, http://ipip.ori.org, and can be incorporated into one’s substantive research scales if social desirability is a concern. Evidence suggests, however, that social desirability distortion represents an insignificant threat to validity in most contexts (Johnson & Hogan, 2006) and that invalid responding of any sort on Internet measures is a trivial problem (Johnson, 2005).

The script’s final action will be to produce feedback for the participant after the final script file scores a protocol; examines it for missing values, nonresponsive patterns, and possibly duplications with previous
protocols; and stores the information in a data file. This chapter concludes with that topic.

WRITE PROGRAMMING CODE FOR PROVIDING FEEDBACK

As far as one can tell, a primary motive that encourages people to complete Web-based psychological measure is a desire to learn more about themselves, to compare themselves with other people, or to think about themselves in new ways. Providing meaningful feedback to research participants is therefore one of the prime incentives researchers can offer potential participants. Possibly the most important factor in designing appropriate feedback is that, unlike many popular quizzes on the Internet, it is research based. A second factor that makes feedback useful is when it goes beyond the trivially obvious. Telling participants who strongly agree with many items such as “I love parties” and “I enjoy being around a lot of people” that they are sociable may be accurate, but it is hardly enlightening. More useful would be to explain some of the consequences of being highly sociable, as revealed by research.

The IPIP-NEO measures individual differences along the heavily researched five factors of personality. This allows the narrative feedback report to include relatively rich and definitive descriptions of the five factors and their correlates. Much of this information goes beyond lay knowledge of personality. If you are using a relatively new measure, your feedback will necessarily be less extensive and more suggestive than definitive. Regardless of the amount of available research findings for a scale, the professional integrity of the report depends on making research-based statements about the implication of scores (Johnson, 1996).

With this in mind, the IPIP-NEO narrative report begins with an explanation of personality terms such as trait and percentiles. It then stresses that high scores are not better than low scores and that low, average, or high levels for different traits can be helpful, neutral, or detrimental in different contexts. It also discusses measurement error and how to resolve questions of accuracy. This general explanation of measurement by self-report is followed by descriptions of what research has revealed about persons who score high or low on each trait. Only after a personality trait is explained thoroughly is the participant presented with his or her standing on the trait. Most of the narrative focuses on the nature of the traits rather than on personal description, such that the report reads almost like a personality textbook, instructing the participant about each trait. See a sample report for the IPIP-NEO or complete the measure for your own report at the supplementary Web site for this chapter.

The general introduction to personality assessment and the description of the five factors are the same in everyone’s report. The portion of
the report that is tailored to the individual participant's results is generated by Perl if-statements, as described in chapter 6 of Fraley (2004). For example, the variable $SE represents the t-standardized Extraversion domain score in shortipipneo.cgi, and the if-statements “if ($SE < $LO),” “if ($SE >= $LO && $SE <= $HI),” and “if ($SE > $HI)” determine whether a description of low, average, or high Extraversion, respectively, is printed. The variables $LO and $HI were set previously in the script at 45 and 55, such that scores within a half standard deviation of the mean are considered average. Hofstee (1994) explained why it is advisable to use only three categories along a continuum for feedback.

Currently, the IPIP-NEO scripts at http://www.personal.psu.edu/~j5j/IPIP/ produce simple ASCII graphs of the participants' results, calibrated in percentile scores. A revision of the graphing routines is underway, using Fraley's (2004, chap. 11) method for creating pictorial bar graphs. The needed bar width is created by taking the gray image file of pixel size 1 x 1 (Fraley, 2004, p. 182) and redefining its width as the standardized score multiplied by an experimentally determined constant. The demonstration script at the supplementary Web site for this chapter includes the Perl code for generating the pictorial bar graphs. The multiplier constant that produces bars of appropriate width was determined to be 4.1. Therefore, under the label "# Create graphs," one can see, for example, a width variable for Extraversion ($WEP) defined by multiplying the standardized percentile for Extraversion ($SEP) by this constant: ($WEP = 4.1 * $SEP). This width variable is then to determine the width of the image file, bargray.jpg, in the table row for Extraversion:

<img src='bargray.jpg' width=$WEP height='20'>

The production of graphs as well as verbal summaries can be an effective way to satisfy the desire of participants to know where they stand on traits relative to other people. They may already know from their own experience that they are relatively extraverted, but a bar graph scaled in percentiles can show them whether their level of extraversion puts them in the top 50% or top 10% of a reference group.

Summary and Conclusion

Web-based self-report scales are an efficient way to gather data on psychological traits and states for large numbers of research participants. The one disadvantage of this method, compared with paper-and-pencil measures, is lack of control over the assessment environment, but this is offset by the ability to detect nonresponsive behavior with routines in
the scoring script. (Also, note that one can bring research participants into a computer laboratory to provide Web-based self-reports to enjoy the benefits of efficient data collection and scoring without losing control over the assessment setting.) With a Web-based self-report scale, scoring the scale, checking for errors, recording data, and providing customized feedback are completely automated and instantaneous processes. Purported drawbacks such as lack of diversity in samples and participants responding inappropriately turn out to be nonissues. Properly constructed front pages, registered with the major search engines, will draw large numbers of participants, and scripts can be written to detect nearly every kind of problematic protocol. The payoff for care and effort put into authoring good Web pages and CGI scripts for online self-reports will be large samples of high-quality data.

Additional Resources

RECRUITING FROM INTERNET COMMUNITIES

With the Internet in constant flux, it is impossible to list definitive, permanent sites for locating online communities, but the following three sites can be a good starting point:

- Google Groups: http://groups.google.com,
- Yahoo! Groups: http://groups.yahoo.com/, and
- LiveJournal Communities: http://www.livejournal.com/community/

To give a sense of what these resources will provide, a search on the keyword *shyness* at the time of writing this chapter turned up 140 groups on Google Groups, 116 groups on Yahoo!, and 113 communities on LiveJournal. Representative groups from each site with the number of members are, respectively, alt.support.shyness (848 members), shyness (1122 members), and social anxiety (948 members).

LEARNING SCRIPTING LANGUAGES

Professional programmers almost invariably write as if the reader already understands programming. Consequently, novices should always begin with Fraley's (2004) book. People usually find that once they have learned one programming language, it is much easier to learn additional languages. People also find that modifying existing scripts to their own uses facilitates their learning. Therefore, if after learning Perl you want to learn PHP, you might consider working with a generic PHP script for processing forms, made available by Göritz on her Web site,
RESOURCES ON THE SUPPLEMENTARY WEB SITE, HTTP://WWW.APA.ORG/BOOKS/RESOURCES/GOSLING

- 120- and 300-item IPIP representations of the Revised NEO Personality Inventory (IPIP-NEO)
- Version of 120-item IPIP-NEO that incorporates built-in error-checking and pictorial bar graphs
- Plain text copy of the Perl code in shortipipneo2.cgi for error-checking
- Plain text copy of the Perl code in shortipipneo3.cgi for scoring, standardizing, saving data, and providing feedback
- Script for checking for response consistency with Jackson’s method
- Script for checking for response consistency with Goldberg’s method
- Sample IPIP report

References


