1 INTRODUCTION

WHAT IS A FREE-LIST?

Free-list data elicitation is an invaluable ethnographic technique. At its most essential, the free-list method entails simply asking individuals what belongs in a given **domain** (e.g., morality, animals, foods) or **subdomain** (e.g., things that bad people do, all of the mammals you know, foods that your community enjoys, respectively). The task often comes in the form of "Please list all of the Xs you know about" or "Please tell me all the things that come to mind when you think of Y." Participants proceed to list items and you, the researcher, record what they say and the order in which they say each item.

The formal method of soliciting such discrete, serially listed information from individuals became popular in the 1960s, when psychology and anthropology were closely intertwined (Henley, 1969; Romney & D'Andrade, 1964). While free-listing has had lasting utility in folk-biological and folk-botanical research, where field researchers collect participants' knowledge of their environment (Quinlan, 2005), the method's utility has also spread into a wide range of fields, including cognitive science, ecology, health care, marketing and advertising, religious studies, and sociology.

In the past few years alone, researchers have used free-lists in far reaches of the human community in order to better understand a wide range of domains, including, but not limited to, knowledge of wild mushrooms among residents of Yunnan, China (Brown, 2019); what "honesty" means in the United States (Reynolds, Stokes, Jayawickreme, & Furr, 2023); various types of meat among Nigerians (Friant et al., 2019); cultural models of ritual postures in Mauritius (Kundtová Klocová, 2017); models of (un)healthy foods in Brazil and Germany (Moraes, Sproesser, & Alvarenga, 2023); what it means to be "moral" in Mongolia (Berniūnas, 2020); and conceptual associations with soils across groups in Morocco, Spain, and Tunisia (Topp et al., 2023). Across diverse samples and topics, such studies demonstrate just how profound an impact cultural knowledge, beliefs, and values have on our lives.

WHY FREE-LIST?

Why collect free-list data? There are at least six important reasons to collect such information. First and foremost, collecting free-list data satisfies the need to better understand things that people know, believe, think, care about, and do to and teach each other. The free-list method is an excellent tool to obtain this kind of data. Second, the method allows individuals to speak on their own terms rather than on researchers' terms. This maximizes **ecological validity**; such data really do consist of the kind of cultural data many recognize as important. For example, if you were interested in learning what different groups think constitutes a good partner, you could come up with some items you or your theory thinks are important and ask binary (e.g., yes/no) or Likert scale questions (e.g., "On a scale from 0 to 7 ..."). Imagine designing an instrument of, say, 10 questions asking how important honesty, fidelity, physical attractiveness, and so forth are important for an ideal partner to have. Now imagine asking these same questions to a few thousand people from different communities around the world. Assuming you don't get the same responses across each individual, you might get some interesting data and perhaps you can make some headway into explaining why different people prefer different partners along these 10 questions.

However, what if you also asked people to freely list their preferences in a partner and found that there are both universals in preference (i.e., listed items common to all groups) but also that each group had its own distinct cultural model of what it means to be a good partner? The 10-item scale might not detect any of these important values, and without any other method, you might not ever capture it. What happens if you find out that your questions were irrelevant in some places? Were they perhaps designed in a way that was relevant primarily to educated Americans or average Westerners (Boehm, 1980; Purzycki et al., 2018)? Free-lists are maximally useful when you're starting from scratch and you really need to know what and how people think about a given topic.

Third, then, free-list data are of particular importance in any field that is interested in the nature of **culture** and its role in human thought and behavior. Indeed, free-list data exhibit many important aspects of common theoretical definitions of culture, defined here as shared, socially transmitted information that is stored in human minds (Boyd & Richerson, 1988; D'Andrade, 1981; Sperber, 1996). One aspect of culture that free-list data has is that it has *content*; rather than information presumably codified into Likert scales, the free-list method is a naturalistic way to glean pools of interpretable and meaningful information from people. As it has lexical or semantic associations, this content often points to other aspects of social systems we are interested in, thus linking individuals to their social and natural environments in critical ways. Furthermore, analysis of free-list data highlights the fact that individuals have different knowledge from each other and that this information is stored and structured differently across individuals (D'Andrade, 1987). Additionally, free-list methods also allow for variation across groups; different populations will have different content in the same domain (e.g., favorite dishes, what spirits punish you for, animals you see on a daily basis, songs most people know, what it means to be a "good" person, etc.). Finally, when linked to other data, free-lists can tell us something about processes of cultural transmission. In sum, the free-listing method is a technique that is catered for especially *human* cultural phenomena, thus making it essential for structured and focused ethnographic inquiry.

Fourth, beyond the cultural aspects of free-list data, these methods can also tell us some useful things about cognition, perception, and the structure of thought and memory (Friendly, 1977; Sousa, Soldati, Monteiro, Araújo, & Albuquerque, 2016). Despite the method being literally a list, memory is not structured like one. Yet, information about the ordering of listed items across individuals can tell us quite a bit about conceptual structures. Indeed, individuals search for and explore clusters of information in their minds much like one searches for things outside of their minds; we exploit particular bundles of information, then move on to others (Abbott, Austerweil, & Griffiths, 2015; Hills, Jones, & Todd, 2012). We can see these clusters with the right analyses.

Fifth, free-list data have pragmatic value because researchers and participants alike find them easy to do; described as "deceptively simple" (Bernard, 2017), executing these tasks is one of the most natural forms of systematic interviewing techniques. Even children can effortlessly convey their knowledge using the technique (Pretelli, Mulder, & McElreath, 2022). It is very easy for most people to participate, however limited their knowledge might be in a specific area, and participants tend to enjoy them. Further, non-numerate individuals can provide free-list data without struggling with scales or other quantitative measures. In terms of precision and front-end effort, free-list data management sits somewhere between standard Likert-style survey data and data from long qualitative interviews or text data; it requires more than merely entering numbers into a spreadsheet but is more focused and discrete than meandering and often free-wheeling interview material.

Finally, free-list data are also useful in making a difference in people's lives. To the extent that culture matters in health, well-being, and behavior, knowing what people think is an important step in intervention. Free-listing has been central to a wide range of applied projects. For example, with the ultimate goal of reducing sexual violence and supporting further health-bolstering initiatives, the University of California Speaks Up study used free-list data to better understand models of sexual consent and how individuals convey such models (Wagman et al., 2022). Others have deployed free-list tasks to pinpoint discrepancies between parents' and clinicians' conceptual models of attention-deficit disorder (Fiks, Gafen, Hughes, Hunter, & Barg, 2011). Other examples of applied work using freely listed interview data abound, including projects that identify animals used in local medicinal practices that are endangered or are source of zoonotic illnesses in Nigeria (Friant et al., 2022), detail where residents of Japan obtain information about mental health (Sunohara, Sasaki, Kogo, & Ryder, 2022), and provide important indicators of food stress in Brazil and Haiti (Owens et al., 2022). A critical first step in making a difference in human lives requires having a grasp of the culture involved, how widely knowledge is shared, and how central it is to action. Free-listing can be invaluable as this first step. Simply put, this method has many uses.

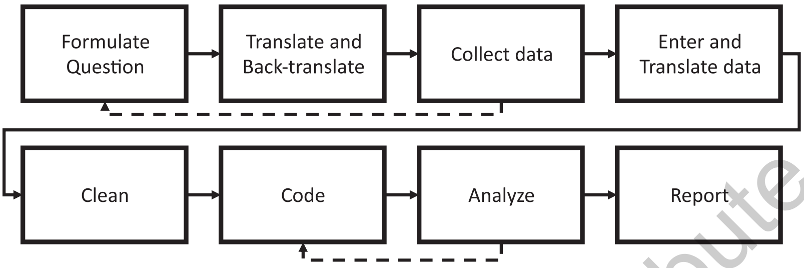
These uses can be means or ends. Free-lists are often used as means to other methodological ends; in order to design more precise instruments, using free-lists first can be indispensable, as we must first learn what people actually think and know before designing things that already presume some culturally specified information. Free-list data are most typically used as ends to describe cultural domains. However, this volume shows that when you wish to use listed items as predictors or classes of items as dependent variables, free-list data can also be used in a predictive statistical framework. If you have predictions about the content and structure of a given domain or anticipate that clusters of knowledge will look differently across groups of people, it is difficult to think of a better method than free-list data elicitation.

GETTING TO WORK

At a bare minimum, you need a domain you're interested in knowing more about and a target sample of people who already know it. Ideally, you have some guiding theory or practical application that motivates using the free-list task, data, and subsequent analyses (see Chapter 7). Once you have decided which domains you wish to ask about, you have to confront the practical reality of executing the task. Are your questions too leading? Are they intelligible in the local language? Do you want an open list or do you need to cap answers? What other data do you need to link to the free-list data? Will your free-list task affect or be affected by other tasks in your study? Do you need to counterbalance the questions (i.e., alternate their order to avoid priming answers across tasks)? Once the answers to these questions are sorted, executing a free-list task from beginning to end typically entails a few important steps modeled in the flowchart in Figure 1.1.

Once your questions are squared away, you might have to translate them into another working language. Ideally, you should also have them back-translated by an independent party to minimize loss of meaning. Once satisfactory, piloting the task with a few individuals might provide some

FIGURE 1.1 ■ Example Workflow of Free-List Data Project.



Note: Dotted lines indicate feedback.

insight and feedback that suggests reformulating the questions is a good idea (illustrated by the dotted feedback lines in the figure).

When you are ready to do the final study, there are a few pointers in interviewing that might be useful to consider. First, there is some evidence that face-to-face free-lists yield more data than online surveys (Weller et al., 2018). Of course, face-to-face interviews also allow you to record particular points of discussion an interviewee might offer or note any important behavioral observations you make while conducting the interview. Direct interviews also provide immediate feedback and details otherwise unobtainable from online methods (e.g., timing oral responses, reactions, etc.). With some online tools, individuals might be able to revise and restructure their free-lists in ways that disrupt the natural process of listing. If this is advantageous for your study, make sure it is a consistent part of the protocol. Otherwise, take necessary steps to minimize participant-led data manipulation.

Second, there are a few techniques for when participants stall or don't provide much data such as probes and prompts (Brewer, 2002). Simple verbal nudges can help along a participant (e.g., "Can you think of anything else?"), but so can one technique called the "silent probe"; after a participant stops listing items, just look at them without saying anything. This mildly unsettling feeling might prompt them to list more items as well. Whichever technique(s) you choose, it helps to plan and stick to your method as consistently across participants as possible. Note that even if you allow participants to exhaust themselves with the task, they still might not exhaust their knowledge of a given domain (Quinlan, 2005; Zambrana et al., 2018). Complementary, mixed methods are always optimal if one wishes to get a more complete, data-driven idea of what participants know, believe, and value, but on their own, free-list data give us plenty to work with.

Third, your target **sample size** will depend on many factors. As you'll see, many of the methods in Chapters 2 through 4 describe properties of the data and sample quite well. If all you want to do is examine free-list frequencies, content, and structure, a general rule of thumb for sample sizes is around 20 to 30 (Stausberg, 2021; Borgatti & Halgin, 1999; Weller & Romney, 1988). You might collect up to 20 to examine the "saturation point" (see Chapter 2) and collect 10 more to see if things changed. If the most commonly listed things have changed with the bigger sample, you probably have some diversity to address. If you want to compare subgroups, having a comparable number for each targeted group will certainly be more informative. In studies where you'll use the content of free-lists to predict other things or use other things to predict free-lists (Chapters 5 through 7), the standard assumptions hold for each analysis type and how hefty your statistical model is. Power analysis will help target a sample size (Gelman & Hill, 2006; Gelman, Hill, & Vehtari, 2020).

DATA MANAGEMENT

Once your data are collected (ideally on paper for archival purposes), you'll want to enter, clean, structure, and analyze them using software. Before we dig into these steps, we should introduce the working software this volume uses.

Software

There are a few resources specifically designed to analyze free-list data. Common resources to use are the DOS-based `ANTHROPAC` (Borgatti, 1989) or the Excel-based FLAME (Pennec, Wencelius, Garine, Raimond, & Bohbot, 2012). Among other issues, however, this software is not as readily equipped to handle larger data sets. It's also not as amenable to quick data transformations necessary for other analytical techniques, techniques of which are only available elsewhere (Borgatti, 2015; Purzycki & Jamieson-Lane, 2017). Ideally, the fewer programs we need to use, the less clutter we have to endure. We will therefore work primarily in `R` (R Core Team, 2023).

`R` is a free, open-source, and widely used programming language and now standard in data analysis. Here, I will assume that you have installed `R` and know how to perform the most basic of operations. The popular program RStudio (RStudio Team, 2020) is a nice interface for `R`, and if you decide to use RStudio, you'll still need to have `R` installed on your computer. While this book assumes some basic familiarity with `R`, there are countless resources available get you going. `R` Code Box 1.1 contains some of the most useful—though not necessarily the most commonly used—commands.

R CODE BOX 1.1: BASICS IN R

```
# comment out
<- # define; object creation
setwd() # set working directory
getwd() # see working directory
str() # structure of object
c() # combine elements into a vector
cbind() # bind columns
rbind() # bind rows
tolower() # convert all letters to lowercase
toupper() # convert all letters to uppercase
summary() # summary of object
View() # view object; note capital V
seq(x, y, z) # sequence from, to, by
plot() # plot
install.packages() # obvious!
library() # load package
merge(x, y, by = "var") # merge two sets by "var"
read.csv() # load .csv file
read.delim() # load .txt file
write.table() # save data as text file
write.csv() # save as .csv text file
d[i, j] # coordinates on a dataset d
object$subobject # $ navigates structures
data.frame() # turn object into a data frame
t() # transpose
par() # set graph parameters
rm(list = ls()) # completely clear workspace
```

These might be some of the most commonly used and/or useful commands in R. To see the full options for each command, run the command with `?` in front of it (e.g., `?plot()`). See accompanying code for some examples.

Throughout the chapters, there are snippets of R code. In many cases, these are included to show how mathematical formulas correspond to programming and to facilitate your own analyses and function writing. The companion materials—available at `https://github.com/bgpurzycki/free-list_QASS`—provide step-by-step examples of how to perform each of the analyses and construct the kinds of graphs included in this book. These materials also include analyses and graphs that go beyond those discussed in the text.

To overcome some of the aforementioned limitations with extant software, we developed the `AnthroTools` package (Jamieson-Lane & Purzycki, 2016; Purzycki & Jamieson-Lane, 2017). Among other things, this package specifically handles free-list data and includes a wide range of free-list–specific instructions, warnings, and error messages to alert users to common issues found in free-lists. We designed `AnthroTools` specifically to perform many of the operations detailed herein. R Code Box 1.2 includes code that will install it on your computer.

R CODE BOX 1.2: INSTALLING ANTHROTOOLS

```
install.packages("devtools")
library("devtools")
install_github("alastair-JL/AnthroTools")
library(AnthroTools)
```

This code installs the `devtools` and `AnthroTools` packages in R. The former is a package that allows you to download developers' packages from the Internet (Wickham, Hester, Chang, & Bryan, 2022), and the latter is one such package designed specifically for free-list data processing and analysis (Jamieson-Lane & Purzycki, 2016; Purzycki & Jamieson-Lane, 2017). Once `AnthroTools` is installed, you only need to load it on subsequent R sessions. If there are any updates to the package, reinstalling it will include any and all updates.

Programming languages like R make it possible to write functions that perform particular tasks. In general, the book avoids relying too heavily on external software packages and provides quite a few functions written specifically for it. Some of the included functions already exist elsewhere in other forms, often with more options. I include all of this to bring focus on the mechanics behind the methods, provide some insight into programming and function writing, and ensure that everything will be reproducible without having to download much. Plus, base R is not likely to change as much as packages might. The same goes for graphs; by keeping things stark while providing enough code to get you going, you'll learn a lot about crafting and customizing graphs to suit your needs. As such, most of the graphs used in this volume are hard-coded. As the book moves forward, there is less and less in-text code simply because operations get too cumbersome fairly quickly. The supplementary code nevertheless includes all that is required to reproduce everything in this volume.

R CODE BOX 1.3: LOADING AND SAVING DATA

```
### Reading Data
FL <- read.csv("FL.csv") # comma separated text file
FL <- read.delim("FL.txt") # tab delimited text file

library(readxl) # load package "readxl"
FL <- read_excel("FL.xlsx", na = "NA") # excel file
FL <- as.data.frame(FL) # turn into a data.frame

### Saving Data
write.table(FL, file = "FL.txt", sep = "\t",
      row.names = TRUE, col.names = NA)
write.table(FL, file = "FL.csv", sep = ";",
      row.names = TRUE, col.names = NA)
```

These bits of code load or save files called "FL" in various formats. Note that to load an .xls or .xlsx file, `R` needs to load a package that specifically handles that task. If you use `AnthroTools`, you might need to convert .xls or .xlsx files into data frames first as importing such files might be automatically converted into a "tibble." If your missing data are denoted with something other than "NA," you need to specify that. In this case, all of my missing data have "NA" in the cells, hence `na = "NA"`. By default, `R` treats cells with "NA" as missing values. To save data in a tab-delimited format, use the `write.table` function, and specify `sep = "\t"`. The other saves the file in a .csv format using a semicolon.

File Formats

In terms of data, `R` can import just about any file format in which you'd have free-list data stored. I typically enter and clean data using standard spreadsheet software (e.g., Calc or Excel) because it's easier to enter, sort, and code everything in it. Among other general good practices (Broman & Woo, 2018), coding consistently and making sure cells in your set without data in them have "`NA`" in them are important guidelines. When the data are finalized (or near-finalized), I work exclusively in `R` primarily because it won't alter the data and it performs so many other operations. The next section addresses what your data might look like.

I used to save data files as .csv files (comma-separated files) until I started using European computers. Some currencies use commas instead of decimals, and local computers tend to default to creating semicolon-separated files instead of comma-separated files as a result. I therefore tend to save all ready-to-use files as tab-delimited text files (.txt). Text files are useful across software, and the tab-delimited format makes it easier to load into R across different international contexts. R Codebox 1.3 shows options how to load and save various file types. The accompanying code uses various file types, but the remainder of the book doesn't worry about it because once data are in R, it is no longer an issue.

Data Structure

One way to store data is the "wide" format. This might be the most typical form of data storage in the social sciences and follows the individual-by-variable format (i.e., individuals are rows and variables are columns). It can be useful to keep all of your data in a single place. The structure of our data can often play a decisive role in what we can do with it, and a lot of software requires particular formats. With a programming language, we just need to know how to get it in the right format for analyses.[1]

To illustrate, let's say that in addition to asking participant age and sex, you asked people to list breakfast foods. In the wide format, your data might resemble something like Table 1.1. All freely listed items (from Item 1 to Item *n*) flow from left to right, with the first-listed item in the Item 1 column, the second-listed item in the Item 2 column, and so forth. This data structure requires that all participants have the same number of Item columns (up to Item *n*); as such, some individuals like ID004 might not list many items, so some cells will have no data. The default value for no data in R is "NA" (see R Code Box 1.3). In the wide format, the number of columns of free-list data corresponds to the longest list in your set. If the longest list in your sample is 50 items, you'll need 50 free-list data columns for everyone, even if most people listed a few items. In such cases, there will be many NA values.

In contrast, the "long" format is more catered to the structure and coding of free-list data (Table 1.2). In this format, individuals get multiple rows and a column is specified for the order[2] in which an item was listed alongside the items listed. This format is arguably easier when having to translate and process your data; it's nicer to view the raw data (in this example, "animals you know" collected in Danish) alongside the translated data. And, when it's

[1]AnthroTools has built-in software that will convert data already in the ANTHROPAC format, and with a little effort, it is quite easy to convert data in the FLAME—or any other—format to be used in R. See the accompanying code for some examples.

[2]In the event that you need to add an order numbers to your data set, you can do this in R quite easily *as long as your data are already sorted correctly*. In other words, you should only do this if your free-list data are in the order individuals listed them and the identifying values for each individual are correct. Assuming your data set is called `data` and your participant IDs are in a column called `ID`, you first create a **vector**—a column or list of information of the same type in R—made of the lengths of each individual's list: `runs <- rle(data$ID)$lengths`. Then, you take this vector and apply a sequence of values that go up to that number for each individual: `data$order <- sequence(runs)`. See the accompanying code for a toy example.

time to code the data, it's much easier and enriching to see the data process all together. It's also much more intuitive to sort data vertically rather than horizontally.

TABLE 1.1 ■ Example Spreadsheet in the Wide Format.

PARTID	Age	Sex	Item 1	Item 2	Item 3	...	Item *n*
ID001	22	1	cereal	apples	oranges	...	NA
ID002	32	0	milk	eggs	cereal	...	fruit
ID003	34	1	milk	bacon	eggs	...	juice
ID004	19	0	juice	milk	cereal	...	NA
...	...	...	...	...	...	...	...
ID*n*	42	1	muesli	milk	bacon	...	eggs

Note: PARTID refers to unique participant number, Sex is participant sex (1 = male, for instance), and items are listed. NA refers to "not available" or missing data.

TABLE 1.2 ■ Example Spreadsheet in the Long Format.

PARTID	Order	Danish	English	Age	Sex
DAN001	1	får	sheep	22	1
DAN001	2	ged	goat	22	1
DAN001	3	yak	yak	22	1
DAN002	1	ged	goat	54	0
DAN002	2	får	sheep	54	0
DAN002	3	ko	cow	54	0
DAN002	4	kamel	camel	54	0
DAN002	5	trane	crane	54	0
DAN002	6	yak	yak	54	0

Note: PARTID refers to unique participant number, Danish represents the raw data, and English is the translated data.

R CODE BOX 1.4: RESHAPING DATA

```
FLlong <- reshape(data,
        varying = c("Item1", "Item2", "Item3", "Itemn"),
        timevar = "order", idvar = "id",
        direction = "long", sep = "")
```

This code creates an object called `FLlong` by running `data` through the `reshape` function. It specifies which variables are the target of reshaping (here, called "Item1", "Item2", and so forth), which variable is the participant ID number (here, the ID variable is called "id"), and which direction the data are to be reshaped (`direction = "long"`). If the variables in the wide format have numerical values indicating their order number (e.g., "Item 1", "Item 2", etc.), the `sep = ""` command will pick out those numbers and use them as order values. A succinct way of defining the `varying` command is to find the column numbers of the variables, say, from columns 2 to 31, and use `varying =names(data[,2:31])`.

```
FLwide <- reshape(FLong,
         timevar = "order",
         idvar = "id",
         v.names = "Item",
         direction = "wide")
```

This code turns the data back into the wide format. In this case, we're declaring that each column with free-list data should be called "Item" and they will be stored as "Item.1", "Item.2", and so on. You can see a working example in the accompanying R code.

For many analyses, even if you decide to *store* your free-list data in the wide format, you'll still need to *transform* the data into the long format anyway. When the maximum number of items someone listed is relatively low and data processing is relatively straightforward, it pays to keep everything in the wide format. However, if data processing is going to be a bit more involved (e.g., with translation, coding with multiple coders, etc.), I recommend keeping the free-list data and all of its processing steps in the long format. In R, it's very easy to navigate between long and wide formats; you can see how to reshape your free-list data in R Code Box 1.4. You can always store data in different sheets; as long as there is a linking variable, such as PARTID that connects these different structures together, merging data is easy.

R CODE BOX 1.5: MERGING DATA

```
FL1 <- read.delim("FL1.txt")
FL2 <- read.delim("FL2.txt")
FL3 <- read.delim("FL3.txt")

mr1 <- merge(FL1, FL2, by = c("PARTID", "Order"),
      all.x = TRUE, all.y = TRUE)
mr2 <- merge(mr1, FL3, by = c("PARTID", "Order"),
      all.x = TRUE, all.y = TRUE)
```

After loading your data sets (here, there are three different hypothetical free-list sheets) and assigning them different names, you just need to merge all of the sets together by both participant ID and order number (variables called `PARTID` and `Order`, respectively). The `all.x` and `all.y` commands tell `R` to include *all* of the data from both objects when merging. It automatically recognizes your longest `Order` number and assigns that number of rows to each individual, filling all missing values with `NA`. If you set this option to `FALSE` for both cases, then the resulting data set will have only those variables common to both data sets. The accompanying `R` code includes an example of merging free-list data sets.

Data Merging

If you have multiple free-list data sets entered into different spreadsheets and you want to merge them together so that they're all in one handy spot, you can use the `merge()` function. This convenient function will link disparate data sets by participant IDs and listed order number, so be sure that (a) the variable for participant ID is called the same thing across sheets, and (b) the participant IDs actually refer to the same individuals. This operation also conveniently accounts for the fact that sometimes, the same individual will list different numbers of items across different free-list tasks. In a way, this operation will create an item-by-domain matrix where the maximum number of rows for any individual will be the maximum number of items they listed in any of the target tasks. So, if an individual did three free-list tasks and listed 5 items for one, 18 items for the second, and 7 items for the third, the final merged data set will have 18 rows for that individual. The rest of the values will be `NA`. `R` Code Box 1.5 gives an example of how to merge a few disparate data sheets.

Data Cleaning

Once your data are entered and, if necessary, translated into your working language, it helps to make sure that they are then properly cleaned for analysis.

Cleaning is probably best done in standard spreadsheet software using the long format described above. That way, whoever is cleaning and coding the data can sort them quickly (e.g., by country, alphabetically by item, order number, etc.). But be warned: This software often sorts the data in ways you aren't anticipating (e.g., it will only sort selected or highlighted data rather than the entire data set) and can often reformat your data because it is programmed to detect particular patterns. My advice is to *carefully* clean and sort in a standard spreadsheet, save as a tab-delimited text file, and then do all subsequent transformations and analyses in `R`. Aside from perhaps encoding strange characters that you might have in your file (e.g., æ, ø, etc.),[3] `R` will not alter your data. Furthermore, instead of pointing and clicking, sorting data in `R` requires writing code (see `R` Code Box 2.4). This is itself a form of error control; it's much easier to accidentally sort data incorrectly by pointing and clicking, whereas writing code is relatively more deliberate.

Cleaning your data means getting them in shape to analyze. IDs and item order values have to be correct. The same items have to be coded the same way so that your analysis is meaningful. This means (a) spelling the same items the same way; (b) making sure similar items are coded in the same way, even though they might be worded differently; (c) all words and letters are in the same case; and (d) making sure any spaces are in the same place. Let's say you ask people to list farm animals and one person says "cow" and another says "cattle." Should you treat these as the same? If so, your computer certainly won't appreciate this, so you have to do something about it.

Similarly, most programming languages like `R` appreciate capitalization; your computer will treat "Goat" differently from "goat." If you didn't do it during the cleaning stage, you can do this in `R`. To convert all of the letters in a vector to lower- or uppercase, use the `tolower()` or `toupper()` functions. For mysterious reasons, researchers or coders sometimes enter spaces before or after items (e.g., " goat" or "goat " instead of just "goat"). Here, too, `R` and other programs will treat these as different from the target item. Cleaning minimizes these issues so that your analyses are streamlined and avoid obnoxious clerical problems. `AnthroTools` includes the `CleanFreeList()` function that will scan and correct your data for more mundane issues like order number sequencing, when individuals list multiples of the same item, and remove any individuals with such issues (see supplementary code for example).

Once they're entered, *don't change the raw data directly!* Best practice dictates that however visually unpleasant, all raw data should be stored with

[3]Note that if you use characters beyond the standard Latin alphabet, many functions won't work in `R`. When recoding free-list data, it makes sense to use the standard Latin alphabet when performing analyses. Usually, you can import files with a more general encoding by adding `encoding = "UTF-8"` to your code when loading your data. If you need special characters for tables and graphs, you'll have to check for any issues with encoding.

the cleaned data so that anyone can examine the changes that took place between data entry and analysis. In other words, enter them raw, and provide separate columns or rows for the cleaned-up version. You can see a mock example—including errors—in Table 1.3.

Data Coding

In some cases, you might have a standardized coding rubric to help guide coding. Ideally, a theoretical motivation drives one's coding scheme, but such schemes also have the practical advantage of appropriately reducing unnecessarily precise resolution in your data. For example, if you are really only interested in how people list ungulates, birds, and house pets, it doesn't help to have your data coded with specific examples (e.g., tapirs, blackbirds, gerbils, etc.). Instead, use general codes that capture all of the appropriate items at the appropriate level of resolution.

If you have a general coding scheme, it often pays to hire two or more assistants to use it to code the free-list data. If these coding assistants are not familiar with your research goals, all the better; it is a testament to your coding scheme if independent coders can usefully code your data and agree with each other. But you have to give them good data. As you can see in Table 1.3, some of the raw data points for EX001 and EX003 were cleaned incorrectly; yaks are decidedly *not* sheep, and giraffes are *not* dogs. In such cases, your coders might only work with the "cleaned data," thus making it a little more difficult to track down any downstream errors. I suggest stressing the point that if they have a coding rubric, they need to be as *consistent as possible* as they go through the data. If the same items are coded differently, this can be disastrous for downstream analyses. That said, it also helps to get the job done in one chunk of time rather than sporadically over a long span of time.

Coders are likely to disagree on some of their codes. You can easily calculate how much they (dis)agree (i.e., **intercoder reliability**) by looking at the proportion of agreement across the coders, but this runs the risk of overestimating agreement since coders could have agreed by chance in some instances. A commonly used metric that overcomes this problem is Cohen's κ (Cohen, 1960), a value that ranges between 1 (perfect agreement) and 0 (perfect disagreement).[4] There are conventional thresholds of how to interpret intercoder reliability metrics like Cohen's κ (Landis & Koch, 1977), but like all rules of thumb, they are just conventions and should be treated with care. It never hurts to thoroughly check all disagreements anyway. For instance, it might be the case that the disagreement is systematic; sometimes, particular

[4]Cohen's $\kappa = \frac{p_o - p_c}{1 - p_c}$, where p_o is the proportion of agreements out of the total number of decisions (i.e., how many times coders agreed and disagreed), and p_c is the proportion of chance agreement, the sum of the joint probabilities of the marginal proportions. See the accompanying code for a walk-through example and a function that computes Cohen's κ.

TABLE 1.3 ■ Example of Working Spreadsheet in the Long Format.

ID	Order	RAW	CLEAN	C1	C2	DIS	CODE.G	TIME	EAT
EX001	1	YAK	sheep	ungulate	ungulate	0	ungulate	00:05	Y
EX001	2	GOLDFISH	goldfish	pet	pet	0	pet	00:10	Y
EX001	3	COWS	cow	ungulate	ungulate	0	ungulate	00:15	N
EX002	1	pigs	pig	ungulate	ungulate	0	ungulate	00:20	N
EX002	2	parrot	parrot	pet	bird	1	bird	00:25	Y
EX002	3	goldfish	goldfish	pet	pet	0	pet	00:30	N
EX002	4	parakeet	parakeet	pet	bird	1	bird	00:35	Y
EX003	1	Tapir	tapir	ungulate	bird	1	ungulate	00:40	N
EX003	2	Dogs	dog	pet	pet	0	pet	00:45	N
EX003	3	Giraffes	dog	pet	pet	0	pet	00:50	Y
EX004	1	dolphins	dolphin	ungulate	ungulate	0	ungulate	00:55	Y
EX004	2	deer	deer	ungulate	ungulate	0	ungulate	01:00	N
EX004	3	elks	elk	ungulate	ungulate	0	ungulate	01:05	N
EX004	4	cats	cat	pet	pet	0	pet	01:10	N
EX004	5	gerbils	gerbil	ungulate	pet	1	pet	01:15	N
EX004	6	guppies	guppy	pet	pet	0	pet	01:20	Y
...	...	...	...	...	...	...	...	...	...
EX*n*	1	quokka	quokka	pet	pet	0	pet	00:05	N!

Note: ID refers to unique participant number, RAW represents the raw data, and CLEAN is the translated data. C1 and C2 are data coded by Coders 1 and 2, DIS denotes whether or not there was disagreement, and CODE is the final data to be analyzed.

items or codes are ambiguous, so coders disagree more often. Resolving these kinds of disagreements are important because they can help improve the coding rubric (Hruschka et al., 2004).

In the event of unsystematic disagreement, some method of resolution is also required. A lead researcher can choose the worst or best of the two or completely override the coders. Either way, having a clear reason and being transparent about the decision can help others reproduce your work and lead to helpful guidelines for future efforts. The spreadsheet example in Table 1.3 includes variables for Coders 1 and 2 (`C1` and `C2`, respectively), counts where the coders disagreed (`DIS`), and the resolved and final items used for analysis (`CODE.G`).

Cleaning and coding are important tasks and serve as a critical bridge between participants and reporting. In fact, careful and thoughtful coders will probably wind up knowing aspects of your data better than anyone else as they often observe oddities, outliers, and other important facets of participants' thoughts that would otherwise go unnoticed. All in all, the task of coding is one of the more important stages of the workflow as poring over each data point inevitably facilitates familiarization with participants' knowledge and attitudes, but it also can play a large role in how your analyses pan out. Treat your coders well.

Consultations and Other Meta-Data

Sometimes, specific data points or large swathes of data will be mysterious to both coders and the researcher. Especially in cross-cultural projects, culturally idiosyncratic terms might befuddle translation or require more context. In such cases, consultation with others or other material is necessary to properly code the items. For example, one major cross-cultural project asked individuals to list what pleases local spirits (Bendixen et al., 2024). In one group, many individuals listed "milk." In a coding rubric with around a dozen different categories, one option was FOOD and the coders coded all instances of "milk" as FOOD. This might have been interpreted as the local spirit enjoys drinking milk. In this case, however, it was more appropriate to code "milk" as RITUAL because it is a standard ritualized offering at shrines where this group lived. In such cases, even if coders are confident they have coded it correctly, it sometimes makes sense to override such codes after a consultation with an expert. Such maneuvers are important for interpretation and should be explicitly documented as part of your workflow. That way, anyone can follow what you've done and why.

After analysis, you might see other issues that warrant recoding. This isn't ideal; your coding rubric should be informed by something theoretically useful. At the same time, the qualitative aspects of free-list data really do lend themselves to interpretation and engagement. Furthermore, as science is an iterative process, we should be learning and evolving our knowledge. But if the coding scheme did not work because the theory has problems, then revisions should not have anything to do with advancing or disconfirming a theory; we shouldn't have our codes altered to get the answer we thought was correct. However, if it is a matter of contextualization, interpretation, coding, or problems of translation, then post-analysis recoding might be justified. Again, it never hurts to add a variable in your spreadsheet that details the kinds of and stage at which feedback took place and report what you did and why. This way, all data alterations are logged and transparent, thus making reproduction easier. Transparency and good bookkeeping are especially important at this stage. Keeping track of such **meta-data** is crucial as it often tells us important factors that went into the production of data.

Reporting and Open Science

The remainder of the volume offers various ways of reporting analyses using tables, graphs, and discussion. In my view, these three aspects of conveying your results should be as straightforward as possible. Tables should include enough information to summarize the verbal discussion of results and their interpretation. Graphs should not overwhelm (or underwhelm) readers. In my opinion, many graphs used in the social sciences come across as tacky and gimmicky. On the other hand, in advertisement design and other tasks where the goals include attracting attention and/or selling a product, flashy graphs designed more to excite and/or convey an impression rather than inform might be perfectly suitable (Frankfurt, 2005). Again, to provide a baseline of graphical techniques, I'll keep things stark.

As a final note, I strongly encourage researchers using free-list data to seriously consider making their data publicly available! The burgeoning open-science movement calls for more scientific transparency; providing all data and code for all scientific works is increasingly recognized as a prerequisite for contemporary practice (Hardwicke et al., 2020; Minocher, Atmaca, Bavero, McElreath, & Beheim, 2021). Unfortunately, compared to how often researchers use free-list methods, there are very little data available for public use. Moreover, even when they are available, data aren't always in a state ready to facilitate reproduction and reuse. This needs to change. Given that much of free-list data are collected in fieldwork, research like this is often difficult to reproduce and replicate as it is, so making this precious data available is an important service to the public and to the field.

As text files, data and code can be uploaded to personal websites or public repositories such as `https://github.com` or `https://osf.io`. These repositories keep track of any and all alterations to your materials, thus maximizing transparency for many of the important data-processing steps. In addition to data, you can make your analytical code and its development public as well.

The remainder of the volume provides the kind of data and code for analyses that you would include to follow along step-by-step.

Finally, while it has never been easier to publicly deposit and reuse data, the sizable moral and ethical dimensions of doing so are not so clear-cut. Many now push publishing data as an obligation, but living up to this expectation might violate the way participants in our studies see things. In my view, our primary priority is *always* and *unambiguously* the participants of our studies. If our research would in no way bring negative consequences to them, and their anonymity would be in no way impaired by the release of data, then we should be able to make a case for their public availability, especially if the research was funded by a public institution. Among some communities, however, there are also compelling calls for the increased assurance that participants have rights over the use and availability of data collected among them (Walter, Kukutai, Carroll, & Rodriguez-Lonebear, 2021). We should ensure that data and their public availability are handled ethically and responsibly (Carroll et al., 2020). Considering that benefits to researchers typically outweigh those accrued by participants, we should always consider how our projects might serve the communities in which we work (Broesch et al., 2020). If we're unsure, we should ask them. At the end of the day, we should be fully transparent with participants about what data will be made public, and they should decide whether or not it is worth their participation.