


How to Automatically Document Data With the *codebook* Package to Facilitate Data Reuse



Ruben C. Arslan 

Center for Adaptive Rationality, Max Planck Institute for Human Development

Advances in Methods and
 Practices in Psychological Science
 2019, Vol. 2(2) 169–187
 © The Author(s) 2019



Article reuse guidelines:
sagepub.com/journals-permissions
 DOI: 10.1177/2515245919838783
www.psychologicalscience.org/AMPPS



Abstract

Data documentation in psychology lags behind not only many other disciplines, but also basic standards of usefulness. Psychological scientists often prefer to invest the time and effort that would be necessary to document existing data well in other duties, such as writing and collecting more data. Codebooks therefore tend to be unstandardized and stored in proprietary formats, and they are rarely properly indexed in search engines. This means that rich data sets are sometimes used only once—by their creators—and left to disappear into oblivion. Even if they can find an existing data set, researchers are unlikely to publish analyses based on it if they cannot be confident that they understand it well enough. My *codebook* package makes it easier to generate rich metadata in human- and machine-readable codebooks. It uses metadata from existing sources and automates some tedious tasks, such as documenting psychological scales and reliabilities, summarizing descriptive statistics, and identifying patterns of missingness. The *codebook* R package and Web app make it possible to generate a rich codebook in a few minutes and just three clicks. Over time, its use could lead to psychological data becoming findable, accessible, interoperable, and reusable, thereby reducing research waste and benefiting both its users and the scientific community as a whole.

Keywords

individual differences, coding, data visualization, reliability, codebook, metadata, data dictionary, data documentation, open materials

Received 9/7/18; Revision accepted 2/25/19

Psychologists rarely ensure that their data are *findable*, *accessible*, *interoperable*, and *reusable* (FAIR; Borghi & Van Gulick, 2018; Hardwicke et al., 2018; Kidwell et al., 2016; Wilkinson et al., 2016). Reuse of a data set is therefore difficult for researchers who lack intimate knowledge of it—a group that may come to include the data collector and his or her collaborators after a few years. Furthermore, data sets are rarely documented in standard formats that can be read by search engines and other algorithmic approaches to data, whereas proprietary formats, such as *.sav* (SPSS), are widely used. Often only a processed subset of the data, without item-level information, is released.

There are at least three likely reasons for insufficient documentation of data sets in psychology. One is the curse of knowledge: It is difficult to imagine which descriptions will be cryptic to readers who are not familiar with the data. A second reason is a lack of standards

and training: Psychological scientists tend to receive little to no training in data documentation, and the training that is offered is often ad hoc. Because many data sets in psychology are small and specific to certain research questions, few departments hire specialized data librarians who can concentrate knowledge on the topic and support researchers in archiving useful data. However, even disciplines with a stronger history of data sharing, such as ecology, struggle to share reusable data (Roche, Kruuk, Lanfear, & Binning, 2015). The third likely reason stems from career incentives: They do not favor the significant time investment necessary for the kind of data documentation that would best advance

Corresponding Author:

Ruben C. Arslan, Center for Adaptive Rationality, Max Planck Institute for Human Development, 14195 Berlin, Germany
 E-mail: ruben.arslan@gmail.com

knowledge generation in the long term, but they do favor writing grant applications to collect more data before existing data sets have been fully used. Therefore, psychological scientists often have a surfeit of data and use rich data sets only once.

The *codebook* package gives psychological scientists a tool they can use in the short term, and that potentially even saves them time, in order to create a resource that is useful for everyone in the long term. It automates commonly performed data-summary steps, such as generating descriptive statistics and plots. It can compute state-of-the-art reliability indices automatically (Crutzen, 2014; McNeish, 2018; Peters, 2014). It makes variable and value labels easily accessible in R, while at the same time generating standardized metadata about a data set that can be read by other researchers and by search engines (e.g., Google Dataset Search, <https://toolbox.google.com/datasetsearch>) and other data processors.

The Roles of a Codebook

A good codebook¹ fulfills several roles. By giving a high-level summary of the data and providing meaningful labels, it can make it easier to discover errors, miscoded missing values, oddly shaped distributions, and other cues signaling problems, thereby allowing data creators to clean their data sets more efficiently and reproducibly. A high-level summary, ideally combined with text explaining the structure and nature of the data set, also helps to explain an unfamiliar data set to researchers who want to reproduce analyses or reuse the data. This should lead to fewer errors resulting from analyzing a data set without understanding measurement specifics, which values encode missingness, whether certain survey weights have to be used, and so on.

Codebooks also offer standardization. There are few exceptions (e.g., Gorgolewski et al., 2016) to the general lack of standards in descriptions of psychological data. Projects using integrated data analysis—that is, projects in which the same analysis is performed across multiple cohorts to boost power and generalizability (Leszko, Elleman, Bastarache, Graham, & Mroczek, 2016)—currently devote ample amounts of time to getting multiple data sets into the same format. Human analysts benefit from standardization, but they can make do with unstandardized data if need be. Search engines and other algorithmic approaches to data, however, have a more difficult task when data are not standardized—and psychological scientists frequently rely on search engines. Without a good codebook, how can a search engine tell whether a data set including the word *intelligence* consists of measures of intelligence or elderly people's responses to whether they would be willing to use an intelligent household robot? The task becomes even more challenging when it comes to structural

aspects of the data: How do search engines identify a study using peer reports or dyadic data? How do they differentiate experiments in which mood was manipulated from those in which it was the outcome? And how can researchers filter their searches by sample size, to find only data sets in which each of at least 100 individuals was measured at least 10 times? For example, the Open Science Framework (OSF) currently relies on user-supplied tags—a very limited approach—and is not indexed in Google Dataset Search. As a result, it is difficult to find a data set on OSF without either knowing exactly where to look or investing a lot of time.

Disclosures

The R code for the *codebook* package is open source. The most current version can be accessed on GitHub (<https://github.com/rubenarslan/codebook>), and major versions are archived permanently on Zenodo (doi:10.5281/zenodo.2574896) and in the CRAN package repository (<https://cran.r-project.org/web/packages/codebook/index.html>). A Web site for the *codebook* package documents all usable functions and presents further vignettes illustrating how to use the package (<https://rubenarslan.github.io/codebook/>); this information is also accessible within R. The Web app documented in Box 1 can be accessed at <https://codebook.formr.org>. A gallery of existing codebooks can be found at https://rubenarslan.github.io/codebook_gallery/. Google indexes all public codebooks produced with this package; to see how indexed codebooks look, go to <https://toolbox.google.com/datasetsearch/search?query=site%3Arubenarslan.github.io>.

The *codebook* Package

Immediate benefits

The *codebook* package (Arslan, 2018) makes it easy to share codebooks in many commonly used formats: .pdf files, .HTML Web sites, spreadsheets, R data frames, and proprietary data-set formats (e.g., SPSS and Stata files). These options facilitate sharing information about a data set with coauthors, reviewers, and readers. The codebooks generated by the package are more extensive and more portable than those produced by most other current approaches: They include not only metadata, such as variable names and variable and value labels, but also high-level summaries of the data, such as means and standard deviations, plots of distributions, counts of missing values and complete entries, and descriptions of patterns of missing data. Other available metadata, such as information about which items were shown to whom or the order in which questions were asked, are also displayed.

Box 1. Generate a Codebook in Three Clicks

The *codebook* Web app (<https://codebook.formr.org>) is an easy entry point to generating codebooks using SPSS or Stata files that are on your computer and already have properly labeled variables and values. It is especially convenient if you do not use R. This app allows you to turn SPSS and Stata files into codebooks that are in an open (i.e., nonproprietary) format and can be shared easily. The *codebook* Web app is a simple Web site^a (see Fig. 1) that shows an editable R Markdown document on the left and what initially appears as an empty space on the right. You can ignore the document for the moment; you will learn how to edit it in the remainder of this Tutorial. To generate a codebook, visit <https://codebook.formr.org> and choose a file from the bar at the top. This file can be in any format that can be read by the R package *rio*, such as *.rds* (R data serialization), *.sav* (SPSS), *.dta* (Stata), *.xlsx* (Excel), or *.csv* (comma-separated values).^b Then simply click the green button labeled “Generate codebook.” Depending on the size of the data set you have chosen, you may have to wait a few minutes. When it is ready, the codebook will appear on the right. You can browse the codebook in the app or download it for later using the second green button.

The Web app sets reasonable defaults and allows the user to edit the text and the R code to improve the resulting codebook. However, the Web app does not store edits, is less interactive than working in R, and requires the user to temporarily upload the data set to a server. This is not permissible for certain restricted-use data sets. Moreover, very large data sets may result in an error message because of resource limits imposed by the server. If you want to document large, private, or a number of data sets, or if you first need to add the metadata in R, I recommend installing the *codebook* package locally.

^aThis Web site was constructed using OpenCPU (Ooms, 2014), a way of using R on the Web that is similar but not identical to Shiny (Chang, Cheng, Allaire, Xie, & McPherson, 2018). ^bFor more information, read *rio*'s documentation (Chan & Leeper, 2018).

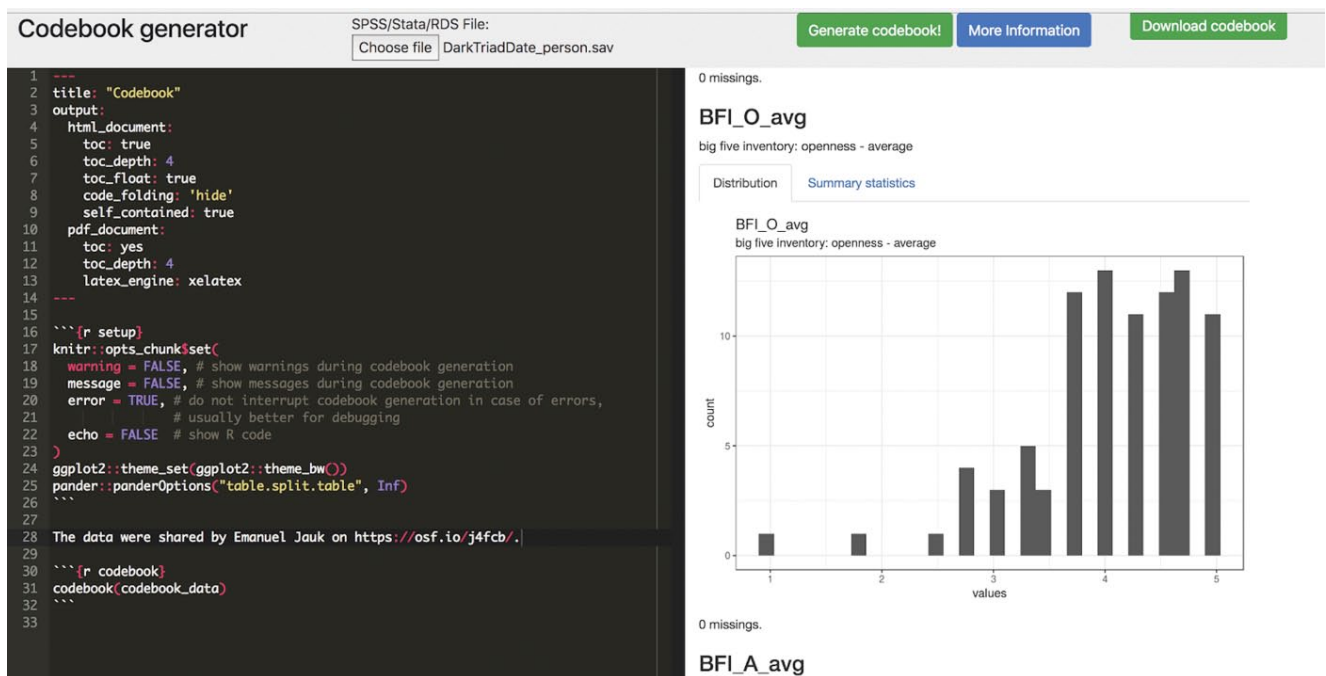


Fig. 1. A screenshot of the *codebook* Web app after a codebook has been generated.

By making data-set contents and structure more transparent, the *codebook* package makes it easier for researchers to search for data sets that might be useful for answering their research questions and to decide whether to reuse a data set for a particular research

question without seeing the data. This is particularly useful when the data cannot be openly shared and access needs to be requested (e.g., because of privacy constraints); it is also more convenient than downloading and examining a plethora of data sets when data

are open. It is especially useful when the literature of adjacent scientific fields is not typically shared between those fields, such that researchers may not easily become aware of data sets available in the other discipline, even though the data can often be repurposed for different questions in their own discipline.

Even after downloading a data set, analysts will go back and forth between the metadata and data frequently in order to find relevant variables, refresh their memory, or simply label axes accurately. By making metadata available in R, the *codebook* package puts this information at analysts' fingertips.

Long-term benefits

When analysts and would-be data sharers use the *codebook* package to document their data sets, they also create machine-readable metadata. Hidden in the HTML files that can be generated are JSON-LD (JavaScript Object Notation for Linked Data) blocks. This format is an extensible shared vocabulary for data sets that is supported by search engines, including Google. Large-scale providers of public data, such as IPUMS (Integrated Public Use Microdata Series, <https://www.ipums.org/>), already generate JSON-LD for data sets. Some providers of large data sets use other, older solutions, such as the Data Documentation Initiative (Rasmussen & Blank, 2007), but it is difficult to learn about and use these solutions with open and free tools. My focus in creating the *codebook* package was on making this functionality available as a bottom-up tool that is suitable for nonspecialists, uses a modern metadata format, is open source and freely available (both in the sense of cost and in not being tied to one platform), and can be improved by its users.

Going beyond search engines, projects such as Wikidata ("Wikidata: Introduction," 2018) link structured data from sources across the Web (Google, 2018; Noy & Brickley, 2017). If data sets were sufficiently complete and structured, Wikipedia could, for instance, automatically display information about the sex differences in various traits in their respective entries, or synthesize the heritability of traits studied using meta-analytic and genetic methods. A Wikipedia for traits that arranges items and traits into ontologies (or nomenclological networks) by collecting bivariate correlations could simply emerge for free instead of being painstakingly assembled, as in the metaBUS and SAPA projects (Bosco, Steel, Oswald, Uggerslev, & Field, 2015; Condon, 2018). Structuring data would also enrich existing data for researchers, for example, by tying locations recorded in psychological data to geographic or administrative metadata, or by tying the time of study participation to current events and news. There are

many ways in which existing data could be reused for purposes not imagined by the researchers who released the data. Findability and accessibility of data sets are crucial for this expanded utility. Meta-analysis would also become much easier. In particular, relevant unpublished effect sizes could be discovered more easily if data sets could be found via search engines by querying the constructs measured. This approach compares favorably with sending requests out through mailing lists and browsing conference abstracts for clues. Including more unpublished and unpredicted effects in meta-analyses could reduce selection bias in estimated effect sizes (Bosco, Aguinis, Field, Pierce, & Dalton, 2016). Even published work can be hard to find using keyword searches and requires a lot of human filtering. Currently, ontologies and meta-analysis databases can be built by determined teams, but have no economy of scale and little potential for automation because every step requires the efforts of qualified researchers and research assistants.

Alternatives

Several other ways to create codebooks exist (see Table 1 for a summary of features). The closest relative to the *codebook* package is the *dataspice* R package (Boettiger et al., 2018), which also generates machine-readable metadata but does not provide an overview of distributions, reliabilities, and missing data; nor does it allow users to easily reuse existing metadata in SPSS and Stata files (i.e., labels for variables, values, and missing values). However, it excels at helping inexperienced R users interactively enter metadata that do not yet exist in a structured form. Conceivably, the *codebook* and *dataspice* packages could be integrated in the future by working with the same metadata substrate; the *dataspice* R package is currently available only on GitHub.

The *dataMaid* R package (Helby Petersen & Thorn Ekstrøm, 2018) also offers a codebook function. It generates a similar, but pdf-focused, overview document, but neither computes reliabilities nor generates machine-readable metadata. The *summarytools* R package (Comtois, 2019) generates overviews, in HTML format, that are similar to those of *dataMaid* and have similar limitations. These packages focus on helping users find errors in their own data and do not prioritize sharing metadata.

As does *codebook*, the Dataset Nutrition Label (Holland, Hosny, Newman, Joseph, & Chmielinski, 2018) project generates high-level data-set overviews for machine learning, but it does not yet offer a public-facing product; furthermore, it does not yet generate machine-readable metadata. There are also several

Table 1. Comparison of the Features of Various Codebook Solutions

Feature	<i>codebook</i>	<i>dataspice</i>	<i>summarytools</i>	Dataset Nutrition Label	<i>dataMaid</i>	DataWiz	Other online providers
Machine-readable metadata	Yes, for data sets and variable labels	Yes, for data sets and variable labels	No	Not yet	No	Yes, but only citation-related metadata	Yes, but usually only citation-related metadata
Distribution plots of the data	Yes	No	Yes	Yes	Yes	No	Varies
Reliability computation	Yes	No	No	No	No	No	No
Missingness patterns	Yes	No	No	Yes	No	No	No
Web interface	Yes	Yes (Shiny)	No	No	Not yet	Yes	Yes
Interactive entry of metadata	No	Yes (locally)	No	No	No	Yes (online)	Yes (online)
Independent storage	Yes	Yes	Yes	Yes	Yes	Yes	No
Metadata available locally during analysis	Yes	No	Yes	No	No	No	No

online interfaces in which one can enter metadata. The DataWiz platform (<https://datawiz.leibniz-psychology.org>; Kerwer, Bölter, Dehnhard, Günther, & Weichselgartner, 2017) helps users generate data-set documentation that complies with funders' statutes, but lacks a dedicated outlet for sharing machine-readable metadata that can be indexed by search engines and instead focuses on administrative information that is not particularly interesting for other researchers, such as information on funders and data-management plans.

I have tried describing the same data set on various well-known commercial, nonprofit, and governmental platforms, including OSF, Figshare, DataDryad, ReShare, the Inter-university Consortium for Political and Social Research (ICPSR), Dataverse, PsychData, and Zenodo. I found that none make use of metadata that are already stored in a data file. Some, such as OSF, do not use metadata related to a data set's contents at all. With others, such as ReShare, I had to give up after the sign-up process led to errors. Yet others, such as Dataverse, allow for metadata, but they have to be entered in a cumbersome online interface, even if they already exist in a structured format, and a researcher does not derive any selfish benefit from entering the metadata. These platforms are thus better suited to storing data; the rich description of the data for other humans and search engines might be housed more comfortably elsewhere.

The most mature platform for social-science research appears to be openICPSR (<https://www.openicpsr.org>). In sum, the codebook solutions I found, with the exception of *dataspice*, do not provide rich metadata for humans and machines, can make heavy demands on researchers' time, and give little in return.

Using the *codebook* Package Locally in RStudio

In this section, I provide an introduction to how to use the *codebook* package on your local computer within R. It is also possible to use the Web app if you are not an R user or want to see results quickly (Box 1).

One-time preparatory work and assumed knowledge

RStudio is an integrated development environment for R. The *codebook* package can make use of some of RStudio's features (such as the Viewer tab, R Markdown editing, and addins), but it works independently of it. I strongly recommend using RStudio with the *codebook* package in order to prevent problems resulting from misspecified paths and to simplify the transition to self-contained reproducible project management, and in the remainder of this Tutorial, I assume that *codebook* is

used in combination with RStudio. You can generate a codebook without previous experience with R and RStudio with the help of this Tutorial. For an introduction to R and RStudio, readers can consult the excellent Tutorial for the *apaTables* package (Stanley & Spence, 2018).

Installing R Markdown. The *codebook* package makes use of the *rmarkdown* package (Allaire et al., 2018). R Markdown creates living documents that let you intermesh text, graphics, and code in a fully reproducible manner. R Markdown documents are simple plain-text documents that can be *knit* into rich HTML, pdf, or Word documents. To install R Markdown, simply click on “File,” then select “New File,” and then “R Markdown” in the RStudio menu. RStudio will prompt you to install the necessary packages at this point. The resulting document serves as a succinct introduction to R Markdown. Verbal explanations in plain text surround code blocks, which start with ````{r}` and end with `````, each delimiter appearing on a new line. Click the “Knit” button at the top of the document and select a file name. Within seconds, the document will be turned into an HTML document with formatting and graphics, shown in the bottom-right viewer panel.

Installing codebook. Once R and R Markdown are installed, run the following command in the RStudio console to install the *codebook* package:

```
install.packages("codebook")
```

This command will automatically install the *codebook* package, as well as several other R packages, some of which are discussed in this Tutorial. Another way to install the packages is by clicking the “Install” button in the bottom-right Packages tab in RStudio, then typing in “codebook” and hitting “Enter.”

Creating your codebook

Now you need an R Markdown file to serve as the basis for your codebook. Because codebooks look best with a few defaults already set, load a template by executing the following command in the RStudio console:

```
codebook::new_codebook_rmd()
```

The file that just opened is the template you will be working with. It has been saved as the file “codebook.Rmd” in your working directory. For now, it is just an empty template without data. Try clicking on “Knit” at the top of the document. In the RStudio viewer pane on the bottom right, a codebook for a mock data set

included with the package will appear.² The codebook and its table of contents may look a little squished depending on the size of your screen. You can expand them to a full browser window by clicking on the little window with an arrow in the Viewer tab.

Loading data

It is time to load some data. I describe this process using the bfi data set made available in the *psych* R package (Goldberg, 1999; Revelle et al., 2017; Revelle, Wilt, & Rosenthal, 2010). The bfi data set is already very well documented in the *psych* package, but by using the *codebook* package, one can add automatically computed reliabilities, graphs, and machine-readable metadata to the mix. The data set is available within R, but because this will not usually be the case when you are working with the *codebook* package, I have uploaded it to OSF, which also features many other publicly available data sets. A new package in R, *rio* (Chan & Leeper, 2018), makes loading online data in almost any format as easy as loading local data. You can import the bfi data set directly from OSF³ by replacing the line

```
codebook_data <- codebook::bfi
```

with

```
codebook_data <- rio::import(
  "https://osf.io/s87kd/download",
  "csv")
```

on line 34 in the template. R Markdown documents have to be reproducible and self-contained, so it is not enough for a data set to be loaded locally; you must load the data set at the beginning of the document. You can also use the document interactively, although this will not work seamlessly for the *codebook* package.⁴ To see how this works, execute the line you just added by pressing “Command + Enter” (on a Mac) or “Ctrl + Enter” (on other platforms).

RStudio has a convenient data viewer you can use to check whether your command worked. In the Environment tab on the top right, you should see “codebook_data.” Click that row to open a spreadsheet view of the data set in RStudio. As you can see, it is not particularly informative—just long columns of numbers with variable names like “A4.” Does “A” refer to aggressiveness, agreeableness, or the German industrial norm for paper size? The lack of useful metadata is obvious. Click on “Knit” again to see what the *codebook* package can do with this. It will take time for the result to appear in the Viewer tab, but when it does, scroll through it. You can see a few warnings stating that the package

saw items that might form part of a scale, but there was no aggregated scale. You will also see graphs of the distributions for all the items and summary statistics.

Adding and changing metadata

Variable labels. The last codebook you generated could already be useful if the variables had meaningful names and self-explanatory values. Unfortunately, they do not, which is typically the case. Generally, you will need more metadata: labels for variables and values, a data-set description, and so on. The *codebook* package can use metadata that are stored in R attributes. Attributes in R are most commonly used to store variable types; for instance, *datetime* in R is just a number with two attributes (a time zone and a class marking it as a date and time). However, R attributes can just as easily store other metadata; the *Hmisc* (Harrell, 2019), *haven* (Wickham, Miller, & RStudio, 2018), and *rio* (Chan & Leeper, 2018) packages, for example, use attributes to store labels. The benefit of storing variable metadata in attributes is that even data sets that are the product of merging and processing raw data retain the necessary metadata. The *haven* and *rio* packages set these attributes when importing data from SPSS or Stata files. However, it is also easy to add metadata yourself, as with the following code:

```
attributes(codebook_data$C5)$label
  <- "Waste my time."
```

You have just assigned a new label to a variable (i.e., the variable C5 in the bfi data set). Because this is a lot to type over and over again as you label more variables, you may want to use a few convenience functions in the *labelled* package (Larmarange, 2019) instead. Load the *labelled* package by writing the following in your codebook.Rmd file:

```
library(labelled)
```

Now you can label the C5 item using the following shorthand:

```
var_label(codebook_data$C5) <- "Waste
  my time."
```

Write one of these labeling commands after loading the data set and click on “Knit” again. As you can see in the viewer pane, the graph for the C5 variable now has a label at the top.⁵ If the prospect of adding labels for every single variable seems tedious, do not fear. Many researchers already have a codebook in the form of a spreadsheet, and this can be used to avoid entering

labels one by one. The bfi data set in the *psych* package is a good example of this because it comes with a tabular dictionary. After loading the bfi data, instead of labeling variables one at a time as just illustrated, type the following to import this data dictionary:

```
dict <- rio::import(
  "https://osf.io/cs678/download",
  "csv")
```

To see what you just loaded, click on the “dict” row in the Environment tab in the top right panel. You will see that the dictionary has information on the construct on which each item loads and on the direction with which it should load on the construct. You can make the metadata in the dictionary usable through the *codebook* package, but working on the data frames will often help you do this; to make this easier, use the *dplyr* package (Wickham, François, Henry, Müller, & RStudio, 2019). Load it by typing the following:

```
library(dplyr)
```

To label more than one variable at once, you need a list of variable labels. Each element of the list is one item that you want to label. For example, you could label variables C5 and C1 at once by using the following code:

```
var_label(codebook_data) <- list(
  C5 = "Waste my time.",
  C1 = "Am exacting in my work."
)
```

However, there is already a list of variables and labels in your data dictionary that you can use, so you do not have to perform the tedious task of writing out the list. You do have to reshape it slightly, though, because it is currently in the form of a rectangular data frame, not a named list. To do so, you will use a convenience function from the *codebook* package called *dict_to_list*. This function expects to receive a data frame with two columns: The first should contain the variable names, and the second should contain the variable labels. To select these columns, you will use the *select* function from the *dplyr* package. You will also need to use a special operator, `%>%`. This operator, called a *pipe*, allows you to read and write R code from left to right, almost as you would write an English sentence. To label the variables using the dictionary, you need to take the dict data set, select the variable and label columns, and use the *dict_to_list* function. You also need to assign the result of this operation to become the variable labels in *codebook_data*. You can do all this in a single line using pipes:

```
var_label(codebook_data) <- dict %>%
  select(variable, label) %>%
  dict_to_list()
```

Click on “codebook_data” in the Environment tab again. You should now see the variable labels below the variable names. If you click on “Knit” again, you will see that your codebook now contains the variable labels. They are both part of the plots and part of the codebook table at the end of the document. They are also part of the metadata that can be found using, for example, Google Dataset Search, but this will not be visible to you.

Value labels. You may have noticed that the values for the education variable in the bfi data set are shown as numbers. Do they indicate the number of years of education? The average is 3, so that seems unlikely. In fact, these numbers signify levels of education. In the dict data frame, you can see that there are value labels for the levels of this variable. However, these levels of education are abbreviated, and you can probably imagine that it would be difficult for an automated program to understand how they map to the values in your data set. You can do better, using another function from the *labelled* package: not `var_label` this time, but `val_labels`. Unlike `var_label`, `val_labels` expects not just one label, but a named vector,⁶ with a name for each value to be labeled. Named vectors are created using the `c()` function (labels go in quotation marks before the equal sign, and values go after the equal sign. Add the following lines at the end of the code you have entered thus far:

```
val_labels(codebook_data$gender) <-
  c("male" = 1, "female" = 2)

val_labels(codebook_data$education)
<- c(
  "in high school" = 1,
  "finished high school" = 2,
  "some college" = 3,
  "college graduate" = 4,
  "graduate degree" = 5)
```

Click on the “Knit” button. The bars in the graphs for education and gender should now be labeled.

Now consider the data set’s many Likert items, which all have the same value labels. You could assign these labels in the same way you did for gender and education, entering the same lines for each variable over and over, or you could create a function to do the job for you instead. We will call this function `add_likert_labels`. In the code defining this function, the

keyword function is followed by parentheses and then braces. Inside the parentheses is an `x` that serves as a placeholder for the many variables you will use the function for in the next step. The code inside the braces shows what you plan to do with the variable `x`; use the `val_labels` function and assign a named vector. The last statement determines the resulting value of the function. You should explicitly return `x` by writing it out on its own line:

```
add_likert_labels <- function(x) {
  val_labels(x) <- c("Very Inaccurate" = 1,
    "Moderately Inaccurate" = 2,
    "Slightly Inaccurate" = 3,
    "Slightly Accurate" = 4,
    "Moderately Accurate" = 5,
    "Very Accurate" = 6)
  x
}
```

A function is just a tool and does nothing on its own; you have not used the `add_likert_labels` function simply by entering this code. To use this function only on the Likert items, you need a list of them. An easy way to create a list is to use the `filter` and `pull` functions from the *dplyr* package to select the desired variables from the dict data frame (in the present case, the Big Six items from the bfi data set):

```
likert_items <- dict %>% filter(Big6
  != "") %>% pull(variable)
```

To apply your new function to these items, use another function from the *dplyr* package called `mutate_at`. It expects a list of variables and a function that applies to each. You have both! You can now add value labels to all the Likert items in `codebook_data`:

```
codebook_data <- codebook_data %>%
  mutate_at(likert_items,
    add_likert_labels)
```

Click on “Knit” again. The graphs for all the items should now have value labels. However, this display is quite repetitive. How about grouping the items by the factors that they are supposed to load on? And while you are at it, how can the metadata about keying (or reverse coding) in your dictionary become part of the data set?

Adding scales. The *codebook* package relies on a simple convention to summarize psychological scales, which are aggregates across several items. Your next step will be to assign a new variable, extraversion, to the result of selecting all extraversion items in the data set and passing them to the `aggregate_and_document_scale` function. This function takes the mean of its inputs and assigns a label to the result, so that you can still tell which variables it is an aggregate of. The code for creating the extraversion scale is as follows:

```
codebook_data$extraversion <- codebook_data
  %>% select(E1:E5) %>%
  aggregate_and_document_scale()
```

Try knitting now. In the resulting codebook, the items for extraversion have been grouped in one graph. In addition, several internal-consistency coefficients have been calculated. However, they are oddly low. You need to reverse-code items, such as “Don’t talk a lot,” that load negatively on the extraversion factor.

To do so, I suggest following a simple convention early on, when you come up with the names for the items in your study. Specifically, use `scale_numberR` as the format for reverse-coded items (e.g., `bfi_extra_1R` for a reverse-coded extraversion item, `bfi_neuro_2` for a non-reverse-coded neuroticism item). That way, analysts who use the codebook will know how items relate to their scales. For the present exercise, though, you can keep the names already encoded in the `bfi` data dictionary you imported, but simply rename the reverse-coded items so that you cannot forget about their direction. First, you need to grab from the dictionary all the items with a negative keying, by adding the following line above the `aggregate_and_document_scale()` line of code you just entered:

```
reversed_items <- dict %>% filter
  (Keying == -1) %>% pull(variable)
```

You can see in the Environment tab that names such as `A1`, `C4`, and `C5` are now stored in the `reversed_items` vector. You can now refer to this vector in the `rename_at` function, which applies a function to all variables you list. In the immediately following line of code, use the very simple function `add_R`, which does exactly what its name indicates:

```
codebook_data <- codebook_data %>%
  rename_at(reversed_items, add_R)
```

Click on “codebook_data” in the Environment tab, and you will see that some variables have been

renamed: `A1R`, `C4R`, and `C5R`, and so on. This could lead to an ambiguity: Does the suffix “R” mean “should be reversed before aggregation” or “has already been reversed”? With the help of metadata in the form of labeled values, there is no potential for confusion. You can reverse the underlying values, but keep the value labels right. So with these Likert items, if somebody responded “very accurate,” that label remains, but the underlying value switches from 6 to 1 for a reverse-scored item. The data you generally import will rarely include labels that remain correct regardless of whether underlying values are reversed, but the *codebook* package makes it easy to bring the data into this shape. In the next line of code, a command using *dplyr* functions and the `reverse_labelled_values` function can easily remedy this problem:

```
codebook_data <- codebook_data %>%
  mutate_at(vars(matches("\\dR$")),
    reverse_labelled_values)
```

All this statement does is find variable names that end with a number (`\d` is the regular expression for a number; a dollar sign denotes the end of the string) and “R” and reverse them.⁷ Because the extraversion items have been renamed, we have to amend our scale-aggregation line slightly:

```
codebook_data$extraversion <-
  codebook_data %>% select(E1R:E5) %>%
  aggregate_and_document_scale()
```

Try knitting again. The reliability for the extraversion scale should be much higher, and all items should load positively. Adding further scales is easy: Just repeat the last line of code, changing the names of the scale and the items. Adding scales that integrate smaller scales is also straightforward. The data dictionary mentions the Giant Three—try adding one, plasticity, which subsumes extraversion and openness:

```
codebook_data$plasticity <- codebook_data
  %>% select(E1R:E5, O1:O5R) %>%
  aggregate_and_document_scale()
```

Note that writing `E1R:E5` works only if the items are all in order in your data set. If items from different constructs are intermixed, you will need a different way to select them. One option is to list all items, writing `select(E1R, E2R, E3, E4, E5)`. This can get tedious when many items need to be listed. Another solution is to write `select(starts_with("E"))`. Although this is quite elegant, it will not work in this case because you have more than one label that starts

with “E”; this command would include education items along with the extraversion items you want. This is a good reason to give items descriptive stems, such as `extraversion_` or `bfi_extra`. Longer stems not only make confusion less likely, but also make it possible for you to refer to groups of items by their stems, and ideally to refer to their aggregates by only their stems. If you have already named an item too minimally, another solution is to use a regular expression, as in the earlier code for matching reversed items. In this scenario, `select(matches("^E\\dR?$"))` would work.⁸

Metadata about the entire data set

Finally, you might want to sign your work and add a few descriptive words about the entire data set. If you simply edit the R Markdown document to add a description, this information will not become part of the machine-readable metadata. Metadata (or attributes) about a data set as a whole are much less persistent than metadata about variables. Hence, you should add your description of the data set right before calling the `codebook` function, which actually begins the generation of your codebook. Adding metadata about the data set is very simple: Just wrap the `metadata` function around `codebook_data` and assign a value to a field. The name and description fields are required, so if you do not edit them, they will be automatically generated using the name of the data frame and its contents. To overwrite these values and describe the bfi data set more clearly, enter the following lines above the call `codebook(codebook_data)`:

```
metadata(codebook_data)$name <- "25
  Personality items representing 5
  factors"
metadata(codebook_data)$description <-
  "25 personality self report items
  taken from the International
  Personality Item Pool (ipip.ori.org)
  [...]"
```

It is good practice to give data sets a canonical identifier. This way, if a data set is described in multiple locations, it can still be identified as the same data set. For instance, when I set a canonical identifier for the bfi data set, I did not want to use the URL of the R package from which I took it because URLs can change; instead, I generated a persistent document object identifier (DOI) on OSF and specified it as follows:

```
metadata(codebook_data)$identifier <-
  "https://dx.doi.org/10.17605/OSF.IO/
  K39BG"
```

In order to let other people know whom they can contact about the data set, how to cite it, and where to find more information, we will set the attributes creator, citation, and URL as follows:

```
metadata(codebook_data)$creator <-
  "William Revelle"
metadata(codebook_data)$citation <-
  "Revelle, W., Wilt, J., & Rosenthal, A.
  (2010). Individual differences in
  cognition: New methods for examining
  the personality-cognition link. In A.
  Gruszka, G. Matthews, & B. Szymura
  (Eds.), Handbook of individual
  differences in cognition: Attention,
  memory, and executive control (pp.
  27-49). New York, NY: Springer."
metadata(codebook_data)$url <-
  "https://CRAN.R-project.org/
  package=psych"
```

Finally, it is useful to note when and where the data were collected, as well as when they were published. Although I could not find information as specific as would be ideal, here is the code for entering some further information about the bfi data:

```
metadata(codebook_data)$datePublished <-
  "2010- 01 - 01"
metadata(codebook_data)$
  temporalCoverage <- "Spring 2010"
metadata(codebook_data)$
  spatialCoverage <- "Online"
```

The attributes that can be assigned are documented in more depth at <https://schema.org/Dataset> (Schema.org, n.d.). You can also add attributes that are not documented by Schema.org, but they will not become part of the machine-readable metadata. Click on “Knit” again. In the Viewer tab, you can see that the metadata section of the codebook has been populated with your additions.

Exporting and Sharing the Data With Metadata

Having added all the variable-level metadata, you might want to reuse the marked-up data elsewhere or share it with collaborators or the public. You can most easily export the data and metadata using the *rio* package (Chan & Leeper, 2018), which permits embedding the variable metadata in the data-set file for those formats that support this. The only way to keep all the metadata

in one file is by staying in R, as in the following line of code:

```
rio::export(codebook_data, "bfi.rds")
# to R data structure file
```

The variable-level metadata can also be transferred to SPSS and Stata files, as follows:

```
rio::export(codebook_data, "bfi.sav")
# to SPSS file

rio::export(codebook_data, "bfi.dta")
# to Stata file
```

Note that this export is based on reverse-engineering the SPSS and Stata file structure, so the resulting files should be tested before sharing them.

Releasing the Codebook Publicly

If you want to share your codebook with other people, you can use the `codebook.html` file in the project folder you created at the start. You can e-mail it to collaborators or upload it to OSF file storage. However, if you want Google Dataset Search to index your data set, this is not sufficient. For security reasons, OSF will not render your HTML files, and Google will not index the content of your e-mails (at least not publicly). You need to post your codebook online. If you already have your own Web site⁹, uploading the HTML file to your own Web site should be easy. The simplest way I found for publishing the HTML for the codebook is as follows. First, rename the `codebook.html` file as `index.html`. Then create an account on netlify.com. Once you're signed in, drag and drop the folder containing the codebook to the Netlify Web page (make sure the folder does not contain anything you do not want to share, such as raw data). Netlify will upload the files and create a random URL, such as `estranged-armadillo.netlify.com`. You can change this to something more meaningful, such as `bfi-study.netlify.com`, in the settings. Next, visit the URL to check that you can see the codebook. The last step is to publicly share a link to the codebook so that search engines can discover it; for instance, you could tweet the link with the hashtag `#codebook`.¹⁰ If you can, you should also link to the codebook from the repository where you have shared the data, so that researchers who find your data will also find your codebook. For instance, I added a link to my codebook for the bfi data set on OSF (<https://osf.io/k39bg/>), where I have also shared the data. Depending on the speed of the search-engine crawler, the data set, including its contents, should be findable on Google Dataset Search in anywhere from 3 to 21 days.

When and Why You Should Generate a Codebook

You have just created a public good, but there are also personal benefits to generating codebooks this way. Codebooks are useful not only for other researchers, but also for the majority of us who struggle to keep all the details about our own data sets in mind at all times. Properly annotated data can help us complete rote tasks faster and help us make fewer errors. Usually, it will be easier to create a codebook, and the codebook will be more accurate, if one creates it right after data collection, when the study is still fresh in mind, rather than waiting until later. I hope the increased convenience of having codebooks at hand during analysis might motivate you to create them early on.

Designing studies so that the collected data can be automatically turned into a codebook should lead to more meaningful variable names and labels, and reusable data sets. Currently, pending agreement on naming conventions for psychological variables, the *codebook* package picks up on variables that carry meaning regarding the structure of the data set according to the conventions used in the formr survey framework (formr.org; Arslan, Walther, & Tata, in press), but columns from other survey providers can be renamed according to these conventions if they carry the same meaning.¹¹ To provide useful codebooks, the *codebook* package draws on functionality supplied by many other R packages. I discuss them in this section to give their authors credit, but you do not need to learn about all of them in order to use the *codebook* package. Part of the benefit of putting forethought into metadata (such as variable names) is that automated data summaries can be more meaningful and require less additional user input and interpretation. For example, the variable labels you made for your bfi codebook will be reused by default in the plots and model summaries generated using the *sjPlot* package (Lüdtke, 2018). However, nobody should avoid generating machine-readable codebooks because one of the automated summaries does not look right. Therefore, all the sections of the codebook you just generated can be turned off via arguments of the main *codebook* function. The following features of the *codebook* package may save you time or prevent errors in your work with data sets.

Checking the codebook during analysis

Often, during data analysis, we want to confirm that a variable is the one we intend to use or that values are in the correct order, or we may need to find a

variable but not remember what we called it. To solve these and similar problems, go to the “Addins” menu in the top bar in the RStudio window and choose “Static Label browser” (Fig. 2). This browser shows the variable and value labels for a data set in the bottom right viewer pane of RStudio. It selects the data set that is alphabetically first in the environment or the data set with the name of any text that is selected in the editor. You can also pick a data frame by typing `label_browser_static(data_frame_name)` in the console.

Because you have loaded only the `codebook_data` data set and the `dict` data set, the browser will choose `codebook_data`. In the Viewer tab, you can now see the variable names and the variable and value labels (Fig. 3). When the static label browser is open, you can keep working on and executing your R code. The `codebook` browser and the dynamic label browser have the advantage of allowing you to select a data set conveniently via a dropdown instead of via text selection, but because they are implemented as Shiny apps (Chang, Cheng, Allaire, Xie, & McPherson, 2018), code can be executed only after they have been stopped (using the red stop button).

Automatically making sense of metadata when you have preprocessed the data file elsewhere

Dealing with miscoded missing values. Sometimes not all the missing values in a data set imported from SPSS or Stata will be set correctly. For example, SPSS users often code missing values as 99 or 999, but fail to actually label these as missing-value placeholders. To correct for this, the `codebook` Web app (Box 1) runs the function `detect_missing`. When its argument `ninety_nine_problems` is set to true, values of 99 or 999 will be set as missing values (when 99 or 999 is not in the plausible

range for the variable in question). When the argument `only_labelled` is set to true, values of 99 or 999 will be set as missing values only if they have a label. A similar option is available for negative values, a convention commonly used in Stata. The `detect_missing` function will not do anything for the `bfi` data set because it has no labeled missing values, but calling it by default should be harmless.

Detecting scales that have been aggregated outside of R. If your items have been aggregated outside of R, the function `detect_scales` is helpful. Calling this function on the entire data set will link items and scales. This linkage is a precondition for the Likert plots and reliability computations to work. The function is called by default when data are uploaded into the Web app (Box 1) or when the default `codebook` template is used. It will also warn you if it finds numbered items with no apparent aggregate, or if an apparent aggregate is not perfectly correlated with the sum of the items (which often indicates a missing item or ad hoc reverse-coded item).

Survey response rates and durations

If a data frame has a “session” column to identify participants and datetimes in “created,” “modified,” “ended,” and “expired” columns, the `codebook` package can calculate a few commonly desired summaries about participation in a survey. It can give the number of participants and the number of rows per participant. It can show the dates and times people enrolled and, by subtracting the “created” value from the “ended” value, how long it took participants to fill out the survey. By checking for missing values in the “modified” column, it can differentiate people who filled out information in the survey from those who did not. By checking for missing values in the “expired” column, it can determine how many participants did not finish the survey in time. The resulting values will be summarized in the beginning of the codebook, if the necessary variables exist. The manual for the `codebook` package (<https://rubenarslan.github.io/codebook/articles/codebook.html>) provides an example.

Reliability estimates and Likert plots

The `codebook` package automatically calculates an estimate of reliability for all defined scales. By default, this is done using the internal-consistency indices computed by the `scalediagnosis` function in the `userfriendlyscience` package (Crutzen, 2014; Crutzen & Peters, 2017; Peters, 2014) if there is just one row per participant. If there are up to two rows per participant, the package will calculate internal consistencies for each

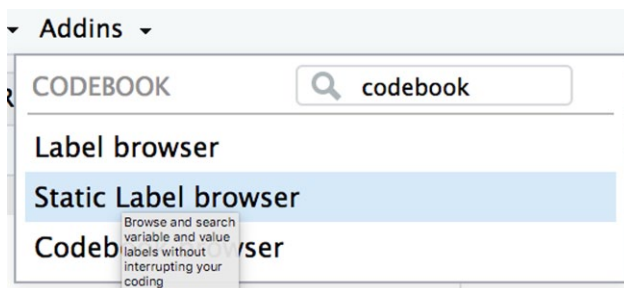


Fig. 2. Screenshot showing the three addins that the `codebook` package supplies to make metadata available during analysis. In this case, the “Static Label browser” has been selected so that the user can view the variable names and labels.

codebook_data columns and labels		
name	label	value_labels
All	All	All
O5R	Will not probe deeply into a subject.	6. Very Inaccurate, 5. Moderately Inaccurate, 4. Slightly Inaccurate, 3. Slightly Accurate, 2. Moderately Accurate, 1. Very Accurate
gender	gender	1. male, 2. female
education	education	1. in high school, 2. finished high school, 3. some college, 4. college graduate, 5. graduate degree

Fig. 3. Screenshot illustrating how variable names and variable and value labels are shown in the RStudio Viewer tab.

time point and a retest correlation between time points (data have to be sorted by time), again using the *user-friendlyscience* package. If there is a variable number of rows per participant and there are more than two rows in some cases, the `multilevel.reliability` function in the *psych* package (Revelle, 2019) reports the generalizability of changes over time, of the person average, and more (Shrout & Lane, 2012). For ordinal variables, the scale summary also includes a Likert plot (Fig. 4) generated using the *likert* package (Bryer & Speerschneider, 2016). A boon of defining metadata up front is that you do not have to get your data into the shape expected by these various functions; the *codebook* package can handle this for you, because it understands how scales and items relate to each other. Given the right metadata, the package could also be extended to automatically compute measures of precision suitable for reaction times or other psychological data.

Distribution plots and descriptive summaries

The *codebook* package also shows a plot of the distribution for each individual item and scale, except when there are large numbers of unique values (e.g., for free text responses, it shows the distribution of number of

characters instead). These plots are labeled using the variable names and variable and value labels. They can be generated in isolation by calling the function `plot_labelled(codebook_data$E1R)` for specific variables. Further, you can use the *skimr* package (McNamara, Arino de la Rubia, Zhu, Ellis, & Quinn, 2019) so that each item or scale will be accompanied by a compact summary of the data, such as the number of missing values, the mean, the range, the standard deviation (for numeric data), and the top count (for categorical data, dates, text, and other data types). If there are labeled missing values (e.g., “user did not do this part of the survey” vs. “user did not respond to this item”), the summary includes the count of the types of missing values in a separate plot.

Missingness patterns

Although the number and types of missing values are always summarized for each item, this does not tell a prospective analyst how many data points have nonmissing data that can be used in the planned bivariate or multivariate analysis. The *codebook* package therefore displays a table of missingness patterns that shows the number of complete cases, cases with missing data for one variable, and variables for which values are frequently missing, as well as whether there are common

Scale: conscientiousness

Overview

Reliability: ω_{ordinal} [95% CI] = 0.77 [0.76;0.78].

Missing: 93.

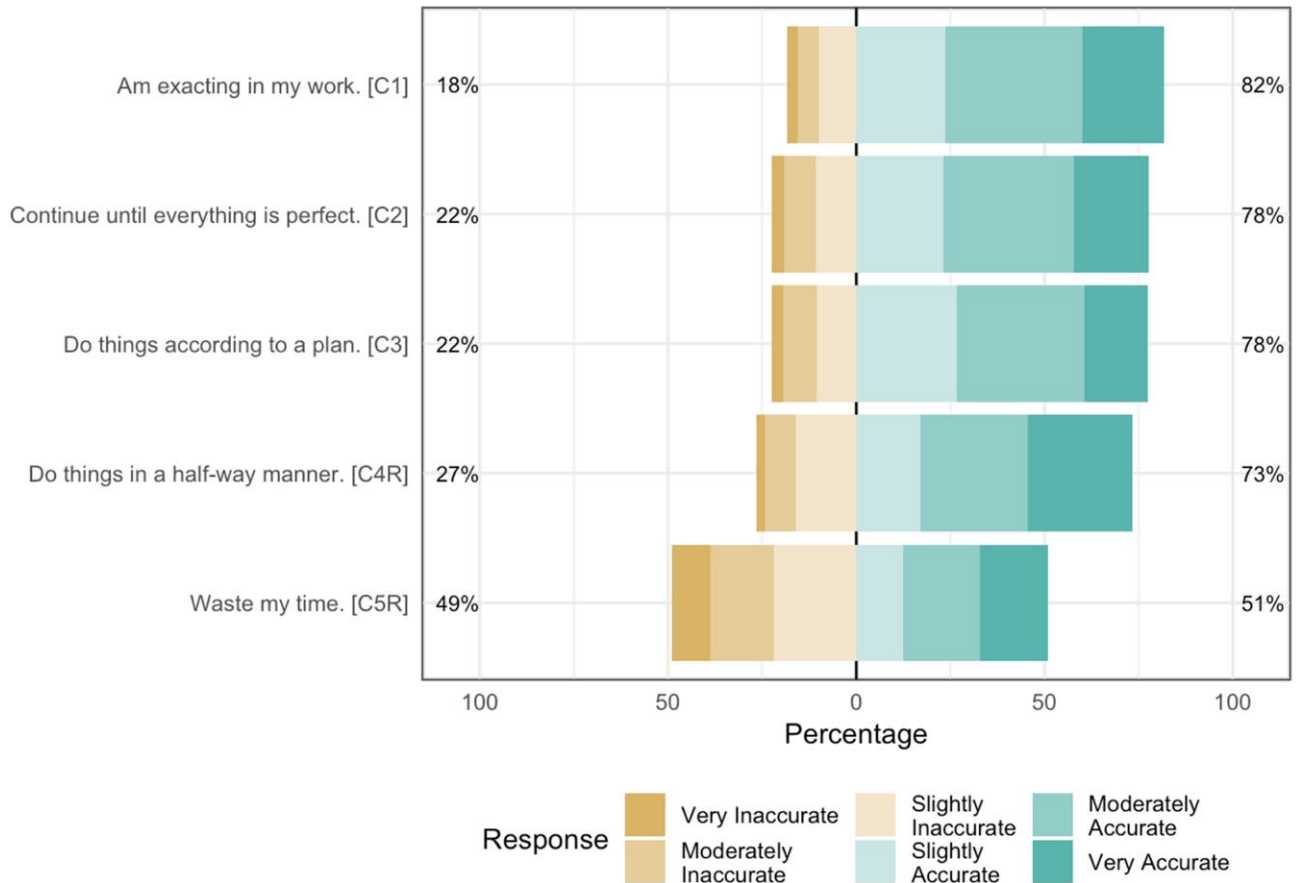


Fig. 4. A Likert plot for the conscientiousness scale in the *bfi* data set in the *psych* package.

patterns of missingness (e.g., all questions about relationships are missing for single people). This information is useful mainly for reusability. If other analysts need to request access through a cumbersome procedure, such as that required by the National Institutes of Health's Data and Specimen Hub, they might like reassurance that the combination of variables they are interested was actually measured often enough for the data set to be useful to them before they immerse themselves in forms.

Codebook table

At the end of a *codebook* document is a rich tabular data dictionary (Fig. 5). This searchable table is made possible through the *DT* package (Xie, 2018). It can be exported

to Excel, csv, and other formats from the browser. The variable names in the table are linked to the details in the HTML codebook. The table also includes variable and value labels, as well as the compact data summaries generated by *skimr* (McNamara et al., 2019). You can create this table directly without making a full codebook by calling the function `codebook_table`, but to share a codebook with other people, I recommend also making available the HTML version of the codebook, which contains the metadata in a search-engine-friendly format.

JSON-LD data

As mentioned earlier, the HTML codebook contains JSON-LD blocks, although these are unseen¹² by the

Codebook table

Copy CSV Excel PDF Print Search:

name	label	data_type	value_labels	scale_item_names	missing	complete	n	mean
All	A	A	All	All		A		
A1R	Am indifferent to the feelings of others.	numeric	6. Very Inaccurate, 5. Moderately Inaccurate, 4. Slightly Inaccurate, 3. Slightly Accurate, 2. Moderately Accurate, 1. Very Accurate		16	2784	2800	4.59

Fig. 5. The first row of a codebook table.

researcher. JSON-LD follows a flexible and extensible standard for metadata and is how search engines, such as Google, “see” your data. Although it is presently limited to basic descriptive functionality, efforts are underway to extend it for use in biology and psychology. If you publish a codebook generated using the package online, so that Google can index it, it will appear in the Google Dataset Search after some time (3–21 days). Unlike many other data platforms, the *codebook* package ensures that the data-set description contains all variable names and labels, thereby making it much easier to find relevant data. A heavily abbreviated example of JSON-LD data on the bfi data set would look like this:

```
{
"@context": "http://schema.org/",
"@type": "Dataset",
"name": "25 Personality items
representing 5 factors",
"description": "25 personality self
report items taken from the
International Personality Item Pool
(ipip.ori.org)[...]",
"identifier": "https://CRAN.R-project
.org/package=psych",
"datePublished": "2010-01-01",
"creator": {
"@type": "Person",
```

```
"givenName": "William",
"familyName": "Revelle",
"email": "revelle@northwestern.edu",
"affiliation": {
"@type": "Organization",
"name": "Northwestern University"
}
},
"citation": "Revelle, W., Wilt, J., &
Rosenthal, A. (2010). Individual
differences in cognition: New methods
for examining the personality-
cognition link. In A. Gruszka, G.
Matthews, & B. Szymura (Eds.),
Handbook of individual differences in
cognition: Attention, memory, and
executive control (pp. 27-49). New
York, NY: Springer.",
"url": "https://cran.r-project.org/web/
packages/psych/index.html",
"temporalCoverage": "Spring 2010",
"spatialCoverage": "Online",
"keywords": ["E1R", "E2R", "E3", "E4",
"E5", "gender", "education", "age",
"extra"],
"variableMeasured": [
```

```

{
  "name": "E1R",
  "description": "Extraversion: Don't
    talk a lot.",
  "value": "6. Very Inaccurate,\n5.
    Moderately Inaccurate,\n4.
    Slightly Inaccurate,\n3. Slightly
    Accurate,\n2. Moderately
    Accurate,\n1. Very Accurate",
  "maxValue": 6,
  "minValue": 1,
  "@type": "propertyValue"
},
{
  "name": "gender",
  "description": "Self-reported
    gender",
  "value": "1. male,\n2. female",
  "maxValue": 2,
  "minValue": 1,
  "@type": "propertyValue"
},
{
  "name": "extraversion",
  "description": "5 extraversion items
    aggregated by rowMeans",
  "@type": "propertyValue"
}
]
}

```

Note that this block of code is basically a nested list of properties—some rather technical, but most with an obvious meaning. (Interested readers may want to refer to Schema.org, n.d., for documentation of possible attributes, but typical users of the *codebook* package can assume that the package creates correct attributes automatically.)

In the future, psychologists could extend the list of possible attributes to document certain psychological measurement scales; whether a variable is a self- or informant-report item or a measured behavior; the type of data set or research design, and even single items such as demographic questions. Extending the schema is an open and community-driven process. Other research communities, such as those in the health and life sciences and in the biological sciences, have started schema-extension processes on Web sites such as <https://health-lifesci.schema.org/> and <http://bioschemas.org/>. Discussions about extending schemas often take place on GitHub. The Society for the Improvement of Psychological Science has formed a work group for the specification of psychological data.¹³

Summary

Standardized, metadata-rich codebooks are useful to data creators, their teams, and the scientific community. The inconvenience and effort involved in creating such codebooks may have contributed to the current state of affairs in psychology: Those codebooks that exist are frequently unstandardized and lack information that is essential to understanding the data, and data sets are not always available in open formats and are rarely machine readable—and are therefore undiscoverable via Web searches. In short, data are rarely easily findable, accessible, interoperable, and reusable. The *codebook* package makes some common tasks easier: It speeds up the data-cleaning and -summary process, and makes data findable and accessible using tools such as Google Dataset Search, independently of where the data are stored or whether they are even publicly available. Thanks to a public standard vocabulary, the metadata are interoperable. And because the package creates codebooks that are rich, descriptive, and interpretable by other researchers, the data become more reusable. The metadata are also portable; structured metadata can be imported to and exported from many formats. A working codebook can be generated by an inexperienced user within minutes. If researchers follow certain conventions or use specific survey providers when generating a data set, or if they reuse metadata available in a closed-source format such as .sav files, they can save even more time, letting the *codebook* package take over graphing distributions, computing descriptive statistics, describing missingness patterns, and estimating reliabilities. It is my hope that the *codebook* package will encourage researchers to generate rich codebooks that benefit themselves and the scientific community as a whole.


Action Editor

Alex O. Holcombe served as action editor for this article.

Author Contributions

R. C. Arslan is the sole author of this article and is responsible for its content.

ORCID iD

Ruben C. Arslan  <https://orcid.org/0000-0002-6670-5658>

Acknowledgments

I am grateful to Martin Brümmer for help setting up the proof of concept for JSON-LD and to early users, including Christoph Schild, Caroline Zygar, Daniël Lakens, Matti Heino, Lisa DeBruine, and Mark Brandt, who tested earlier versions of the *codebook* package. I also thank Bill Revelle, who publicly released the bfi data set and gave me permission to use it for this Tutorial. I particularly thank Ioanna Iro Eleftheriadou, who tested the *codebook* package by generating a gallery from several publicly available data sets on the Open Science Framework. I thank Deborah Ain and Michele Nathan for their scientific editing. All remaining errors are mine.

Declaration of Conflicting Interests

The author(s) declared that there were no conflicts of interest with respect to the authorship or the publication of this article.

Open Practices



Open Data: not applicable

Open Materials: https://zenodo.org/record/2574896#.XH_Z9iJKjb0

Preregistration: not applicable

All materials have been made publicly available via Zenodo and can be accessed at https://zenodo.org/record/2574896#.XH_Z9iJKjb0. The complete Open Practices Disclosure for this article can be found at <http://journals.sagepub.com/doi/suppl/10.1177/2515245919838783>. This article has received the badge for Open Materials. More information about the Open Practices badges can be found at <http://www.psychologicalscience.org/publications/badges>.

Prior Versions

An earlier version of this manuscript was posted as a preprint on PsyArXiv (<https://psyarxiv.com/5qc6h>).

Notes

1. Other widely used related terms are *data dictionary*, which tends to indicate a tabular format, and the broader terms *data documentation* and simply *metadata*.

2. The generated document is named “codebook.html.” You can open this file in your project directory any time to view the codebook or share it with other people.

3. Loading local data is just as easy; remember to put the data set you want to use in the same directory as the *codebook.rmd* file in order to avoid having to think about paths. For example, to load the bfi data set from a local file, go to OSF at <https://osf.io/s87kd/>, download the .csv file and put it in the directory with the *codebook.rmd* file, and then type `codebook_data <- rio::import("bfi.csv")` on line 33 in the template. Note that the package will automatically use a file's extension to select how to import the file, and almost all common standard file extensions for tabular data are supported, including SPSS and Stata.

4. The output generated by the *codebook* package does not fit inside the interactive results box that RStudio uses.

5. Further information about adding labels with the *labelled* package can be found in Larmarange (n.d.).

6. In R, vectors are variables that contain one or many values of one type (e.g., numbers or text). All variables in normal data frames are vectors.

7. This function can work automatically only if the highest and lowest possible values are both encoded in the labels or levels attribute of the variable. Otherwise, *codebook* cannot infer the possible range of the values and will not know how to translate the highest into the lowest value.

8. This code means, “match only variables whose names start with (^) the letter *E*, continue with one digit (\\d) optionally followed by the letter *R* (R?), and then end (\$)”.

9. If you want to learn how to make a personal Web site using GitHub or GitLab, there are several guides available (e.g., University of Glasgow Institute of Neuroscience and Psychology, n.d.).

10. I would happily share the first 10 codebooks published this way with my Twitter followers and also give feedback on them.

11. An example of renaming columns from a Qualtrics survey can be found in the documentation for the *codebook* package (<https://rubenarslan.github.io/codebook/>).

12. By clicking on “JSON-LD metadata,” in small text at the bottom of the generated codebook, you can see a copy of what search engines see.

13. Interested parties can find more information at <https://github.com/mekline/psych-DS>.

References

- Allaire, J. J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., . . . Iannone, R. (2018). *rmarkdown: Dynamic documents for R* (R package Version 1.11) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=rmarkdown>
- Arslan, R. C. (2018). *Cook codebooks from survey metadata encoded in attributes in R*. doi:10.5281/zenodo.1326520
- Arslan, R. C., Walther, M., & Tata, C. (in press). *formr: A study framework allowing for automated feedback generation and complex longitudinal experience sampling studies using R*. *Behavior Research Methods*.
- Boettiger, C., Chamberlain, S., Fournier, A., Hondula, K., Krystalli, A., Mecum, B., . . . Woo, K. (2018). *dataspice: Create lightweight schema.org descriptions of dataset*. Retrieved from <https://github.com/ropenscilabs/dataspice>
- Borghi, J. A., & Van Gulick, A. E. (2018). Data management and sharing in neuroimaging: Practices and perceptions

- of MRI researchers. *PLOS ONE*, *13*(7), Article e0200562. doi:10.1371/journal.pone.0200562
- Bosco, F. A., Aguinis, H., Field, J. G., Pierce, C. A., & Dalton, D. R. (2016). HARKing's threat to organizational research: Evidence from primary and meta-analytic sources. *Personnel Psychology*, *69*, 709–750. doi:10.1111/peps.12111
- Bosco, F. A., Steel, P., Oswald, F. L., Uggerslev, K., & Field, J. G. (2015). Cloud-based meta-analysis to bridge science and practice: Welcome to metaBUS. *Personnel Assessment and Decisions*, *1*(1), Article 2. doi:10.25035/pad.2015.002
- Bryer, J., & Speerschneider, K. (2016). likert: Analysis and visualization of Likert items (R package Version 1.3.5) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=likert>
- Chan, C.-H., & Leeper, T. J. (2018). rio: A Swiss-army knife for data I/O (R package Version 0.5.16) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=rio>
- Chang, W., Cheng, J., Allaire, J. J., Xie, Y., & McPherson, J. (2018). shiny: Web application framework for R (R package Version 1.2.0) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=shiny>
- Comtois, D. (2019). summarytools: Tools to quickly and neatly summarize data (R package Version 0.9.2) [Computer software]. Retrieved from <https://cran.r-project.org/web/packages/summarytools/>
- Condon, D. M. (2018). The SAPA Personality Inventory: An empirically-derived, hierarchically-organized self-report personality assessment model. *PsyArXiv*. doi:10.31234/osf.io/sc4p9
- Crutzen, R. (2014). Time is a jailer: What do alpha and its alternatives tell us about reliability? *European Health Psychologist*, *16*, 70–74.
- Crutzen, R., & Peters, G.-J. Y. (2017). Scale quality: Alpha is an inadequate estimate and factor-analytic evidence is needed first of all. *Health Psychology Review*, *11*, 242–247. doi:10.1080/17437199.2015.1124240
- Goldberg, L. R. (1999). A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In I. Merviele, I. Deary, F. De Fruyt, & F. Ostendorf (Eds.), *Personality psychology in Europe* (Vol. 7, pp. 7–28). Tilburg, The Netherlands: Tilburg University Press.
- Google. (2018). *Dataset search*. Retrieved from <https://developers.google.com/search/docs/data-types/dataset>
- Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., . . . Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, *3*, Article 160044. doi:10.1038/sdata.2016.44
- Hardwicke, T. E., Mathur, M. B., MacDonald, K., Nilsson, G., Banks, G. C., Kidwell, M. C., . . . Frank, M. C. (2018). Data availability, reusability, and analytic reproducibility: Evaluating the impact of a mandatory open data policy at the journal *Cognition*. *Royal Society Open Science*, *5*(8), Article 180448. doi:10.1098/rsos.180448
- Harrell, F. E., Jr. (2019). Hmisc: Harrell miscellaneous (R package Version 4.2-0) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=Hmisc>
- Helby Petersen, A., & Thorn Ekstrøm, C. (2018). dataMaid: A suite of checks for identification of potential errors in a data frame as part of the data screening process (R package Version 1.2.0) [Computer software]. Retrieved from <https://cran.r-project.org/web/packages/dataMaid/index.html>
- Holland, S., Hosny, A., Newman, S., Joseph, J., & Chmielinski, K. (2018). The Dataset Nutrition Label: A framework to drive higher data quality standards. *ArXiv*. Retrieved from <https://arxiv.org/abs/1805.03677>
- Kerwer, M., Bølter, R., Dehnhard, I., Günther, A., & Weichselgartner, E. (2017). *Projekt DataWiz*. Retrieved from https://e-science-tage.de/sites/default/files/2017-04/est_talk_kerwer_17-03-2017.pdf
- Kidwell, M. C., Lazarević, L. B., Baranski, E., Hardwicke, T. E., Piechowski, S., Falkenberg, L.-S., . . . Nosek, B. A. (2016). Badges to acknowledge open practices: A simple, low-cost, effective method for increasing transparency. *PLOS Biology*, *14*(5), Article e1002456. doi:10.1371/journal.pbio.1002456
- Larmarange, J. (2019). labelled: Manipulating labelled data (R package Version 2.1.0) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=labelled>
- Larmarange, J. (n.d.). *Introduction to labelled*. Retrieved from https://cran.r-project.org/web/packages/labelled/vignettes/intro_labelled.html
- Leszko, M., Elleman, L. G., Bastarache, E. D., Graham, E. K., & Mroczek, D. K. (2016). Future directions in the study of personality in adulthood and older age. *Gerontology*, *62*, 210–215. doi:10.1159/000434720
- Lüdecke, D. (2018). *sjPlot: Data visualization for statistics in social science*. doi:10.5281/zenodo.1308157
- McNamara, A., Arino de la Rubia, E., Zhu, H., Ellis, S., & Quinn, M. (2019). skimr: Compact and flexible summaries of data (R package Version 1.0.5) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=skimr>
- McNeish, D. (2018). Thanks coefficient alpha, we'll take it from here. *Psychological Methods*, *23*, 412–433. doi:10.1037/met0000144
- Noy, N., & Brickley, D. (2017, January 24). Facilitating the discovery of public datasets [Blog post]. Retrieved from <http://ai.googleblog.com/2017/01/facilitating-discovery-of-public.html>
- Ooms, J. (2014). The OpenCPU system: Towards a universal interface for scientific computing through separation of concerns. *arXiv*. Retrieved from <http://arxiv.org/abs/1406.4806>
- Peters, G.-J. Y. (2014). The alpha and the omega of scale reliability and validity: Why and how to abandon Cronbach's alpha and the route towards more comprehensive assessment of scale quality. *European Health Psychologist*, *16*, 56–69.
- Rasmussen, K. B., & Blank, G. (2007). The data documentation initiative: A preservation standard for research. *Archival Science*, *7*, 55–71. doi:10.1007/s10502-006-9036-0
- Revelle, W. (2019). psych: Procedures for psychological, psychometric, and personality research (R package Version 1.8.12) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=psych>

- Revelle, W., Condon, D. M., Wilt, J., French, J. A., Brown, A., & Elleman, L. G. (2017). Web- and phone-based data collection using planned missing designs. In N. G. Fielding, R. M. Lee, & G. Blank (Eds.), *The SAGE handbook of online research methods* (pp. 578–594). London, England: Sage.
- Revelle, W., Wilt, J., & Rosenthal, A. (2010). Individual differences in cognition: New methods for examining the personality-cognition link. In A. Gruszka, G. Matthews, & B. Szymura (Eds.), *Handbook of individual differences in cognition: Attention, memory, and executive control* (pp. 27–49). New York, NY: Springer.
- Roche, D. G., Kruuk, L. E. B., Lanfear, R., & Binning, S. A. (2015). Public data archiving in ecology and evolution: How well are we doing? *PLOS Biology*, *13*(11), Article e1002295. doi:10.1371/journal.pbio.1002295
- Schema.org. (n.d.). *Dataset*. Retrieved from <https://schema.org/Dataset>
- Shrout, P., & Lane, S. P. (2012). Psychometrics. In T. S. Conner & M. R. Mehl (Eds.), *Handbook of research methods for studying daily life* (pp. 302–320). New York, NY: Guilford Press.
- Stanley, D. J., & Spence, J. R. (2018). Reproducible tables in psychology using the apaTables package. *Advances in Methods and Practices in Psychological Science*, *1*, 415–431. doi:10.1177/2515245918773743
- University of Glasgow Institute of Neuroscience and Psychology. (n.d.). *Academic webpages*. Retrieved from <https://gupsych.github.io/acadweb/>
- Wickham, H., François, R., Henry, L., Müller, K., & RStudio. (2019). dplyr: A grammar of data manipulation (R package Version 0.8.0.1) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=dplyr>
- Wickham, H., Miller, E., & RStudio. (2019). haven: Import and export 'SPSS', 'Stata' and 'SAS' files (R package Version 2.1.0) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=haven>
- Wikidata: Introduction. (2018). Retrieved from <https://www.wikidata.org/wiki/Wikidata:Introduction>
- Wilkinson, M. D., Dumontier, M., Aalbersberg, IJ. J., Appleton, G., Axton, M., Baak, A., . . . Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*, Article 160018. doi:10.1038/sdata.2016.18
- Xie, Y. (2018). DT: A wrapper of the JavaScript library 'DataTables' (R package Version 0.5) [Computer software]. Retrieved from <https://CRAN.R-project.org/package=DT>