How to automatically generate rich codebooks from study metadata

Author

Ruben C. Arslan
Center for Adaptive Rationality, Max Planck Institute for Human Development, Berlin
ruben.arslan@gmail.com

Abstract

Data documentation in psychology lags behind other disciplines and basic standards of usefulness. Codebooks are usually unstandardised, often stored in proprietary formats, and rarely properly indexed in search engines. Psychological scientists produce a wealth of data, but investing time and effort in documenting existing data well is often valued less than producing more data. So, rich datasets are sometimes used only once, by their creators, falling into oblivion over time. Given realistic amounts of data documentation, researchers may even find it frivolous to publish analyses based on data when they cannot be sure they understand it sufficiently well. The codebook package attempts to make it easier to generate rich metadata in human- and machine-readable codebooks. By automating some tedious tasks such as documenting psychological scales and reliabilities, summarising descriptives, and missingness patterns, we hope to encourage researchers to use the package for their own or their team's benefit. Over time, this should lead to more reusable and searchable data being available, reducing research waste, a benefit to the scientific community as a whole. I introduce an R package and a webapp which make it possible to generate rich codebooks in a few minutes and just three clicks.

Links
Preprint https://psyarxiv.com/5qc6h doi:10.31234/osf.io/5qc6h
Documentation and vignettes: https://rubenarslan.github.io/codebook/
Package on CRAN https://cran.r-project.org/web/packages/codebook/index.html
Webapp https://rubenarslan.ocpu.io/codebook/

Introduction

Psychologists rarely ensure that their data are findable, accessible, interoperable, and reusable (FAIR (Wilkinson et al., 2016). They make data reuse difficult for those who lack intimate knowledge of the dataset, which may even include their forgetful future selves. Data is

also rarely documented in standard formats palatable to search engines and other algorithmic approaches to data.

There are probably three main reasons for insufficient documentation. The curse of knowledge, meaning it is hard to imagine which descriptions will be difficult to understand to readers without intimate knowledge of the data. Further, lack of standards, meaning that there is little training in data documentation and if it happens, it is often done ad-hoc. Because many datasets in psychology are small and specific to certain research questions, there is also little room for professionalization. And lastly, the tragedy of the commons, meaning career incentives do not favour the large time investment in data documentation that would be most beneficial for advancing knowledge generation in the long term.

The codebook package is an attempt to give people something that is useful in the short term at a lower time cost than other solutions, in order to create a resource that is useful in the long term. It automates a few commonly performed data summary steps, such as computing reliabilities, generating descriptive statistics and plots, and making variable and value labels easily accessible in R, while at the same time also generating standardised metadata about a dataset that can be read by other researchers, search engines and other data processors.

Roles of a codebook

A good codebook fulfills several roles. By giving a high-level summary of the data, it can make it easier to **discover errors**, miscoded missings, oddly shaped distributions, and other cues for problems. This allows data creators to clean datasets more efficiently. At the same time, the high-level summary, ideally combined with text explaining the structure and nature of the dataset, helps explain unfamiliar datasets. This should lead to fewer errors where researchers analyze a dataset without understanding measurement specifics, which values encode missings, whether certain survey weights have to be used, and so on. Then there is **standardisation**. There are few exceptions (Gorgolewski et al., 2016) to the general lack of standards in psychological data description. Projects using integrated data analysis, i.e. performing the same analysis across multiple cohorts to boost power and generalisability (Leszko, Elleman, Bastarache, Graham, & Mroczek, 2016) currently have to spend much of their time to get many datasets into the same format. Human analysts benefit from standardisation, but they can make do with unstandardised data. However, **search** engines have a more difficult task. How can a search engine tell whether it is dealing with a dataset in which intelligence was measured or whether it simply contains a question whether elderly people would be willing to use an intelligent household robot? It gets even more difficult when it comes to structural aspects of the data: how do we find a study using peer reports, or dyadic data? And how do we filter by sample size to find only datasets which have measured at least one hundred individuals at least ten times? Repositories like the Open Science Framework (OSF) currently rely on user-supplied tags, a very limited approach. As a result, it is almost impossible to find a dataset on the OSF without knowing exactly where to look, or investing a lot of time.

The codebook package

Immediate benefits

The codebook package (Arslan, 2018a) makes it easy to share codebooks in four commonly required formats: in PDFs, HTML websites, spreadsheets, and in R data frames. These make it easy for a researcher to share information about a dataset with co-authors, reviewers, and readers. The codebooks generated by the package are richer than most current approaches. Not only do they include metadata such as variable names, variable and value labels, but also high-level summaries of the data, such as the means and standard deviations, plots of the distribution, number of missings and complete entries, missingness patterns in the data, and so on. If other metadata is also available, such as information about which items were shown to whom, the order in which questions were asked, and so on, this is also displayed.

In doing so, the codebook package attempts to make it possible to decide whether to use a dataset for a particular research question without seeing the dataset itself. This is particularly useful in cases when the data cannot be openly shared because of e.g. privacy constraints, but it is of course also more convenient than downloading and examining a plethora of datasets.

But even when one has downloaded a dataset, the analyst frequently goes back and forth between the documentation of the data and the data itself, be that to refresh their memory or just to label axes accurately. By making metadata available in R via the attributes of vectors in a data frame, the codebook package puts this information at the fingertips of the analyst.

Long-term benefits

By encouraging analysts and would-be data sharers to use the codebook package to document their datasets, the codebook package will also make machine-readable metadata available. Hidden in the generated HTML files are JSON-LD (Javascript Object Notation - Linked Data) blocks. This format is an extensible ontology for datasets which will be supported by search engines such as Google. Of course, the codebook package could also be employed by larger organizations such as IPUMS[1] or the OSF to automatically generate codebooks and linked data for uploaded datasets. There are already solutions which cater to providers of large datasets, such as the Data Documentation Initiative (Rasmussen & Blank, 2007). My focus was on making this functionality available as a bottom-up tool that can be used by anyone, uses a modern metadata format, is open and freely available (unlike most DDI tools), and can be improved by its users.

Going beyond search engines, projects like WikiData (“Introduction to Wikidata,” n.d.) plan to link structured data from sources across the web (Google, 2017, 2018). If data was sufficiently complete and structured, Wikipedia could, for example, display information about the

[1] https://www.ipums.org/, Integrated Public Use Microdata Series

meta-analytic heritability of and sex differences in various traits on their respective pages. A Wikipedia for traits that arranges items and traits into ontologies (or nomological networks) could simply drop out of this for free instead of being painstakingly assembled as in the SAPA and metaBUS projects (Bosco, Steel, Oswald, Uggerslev, & Field, 2015; Condon, 2017). At the same time, structured data would enrich existing data for researchers, 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 re-used for purposes not imagined by those who released the data. Meta-analysis would of course also become much easier, by enabling data discovery through simpler channels than emails to mailing lists and browsing conference abstracts for clues. Currently, all of these applications are possible, but have no economy of scale.

Generate a codebook in three clicks

On https://rubenarslan.ocpu.io/codebook/ you'll find a simple OpenCPU web app (Ooms, 2014). On the left side, you see an editable rmarkdown document. You can ignore this document for now. On the right side, there is currently an empty space. Choose a file in the bar at the top. This can be an .rds (R data serialization) file, an .sav file (SPSS format), a .dta file (Stata format), or any other file format that can be read by the rio package (Chan & Leeper, 2018). After selecting a file here, simply click the green button saying "Generate codebook". Depending on the size of the dataset, this may take up to a few minutes. The codebook is then shown on the right. You can browse the codebook on the page, and download it using the second green button.

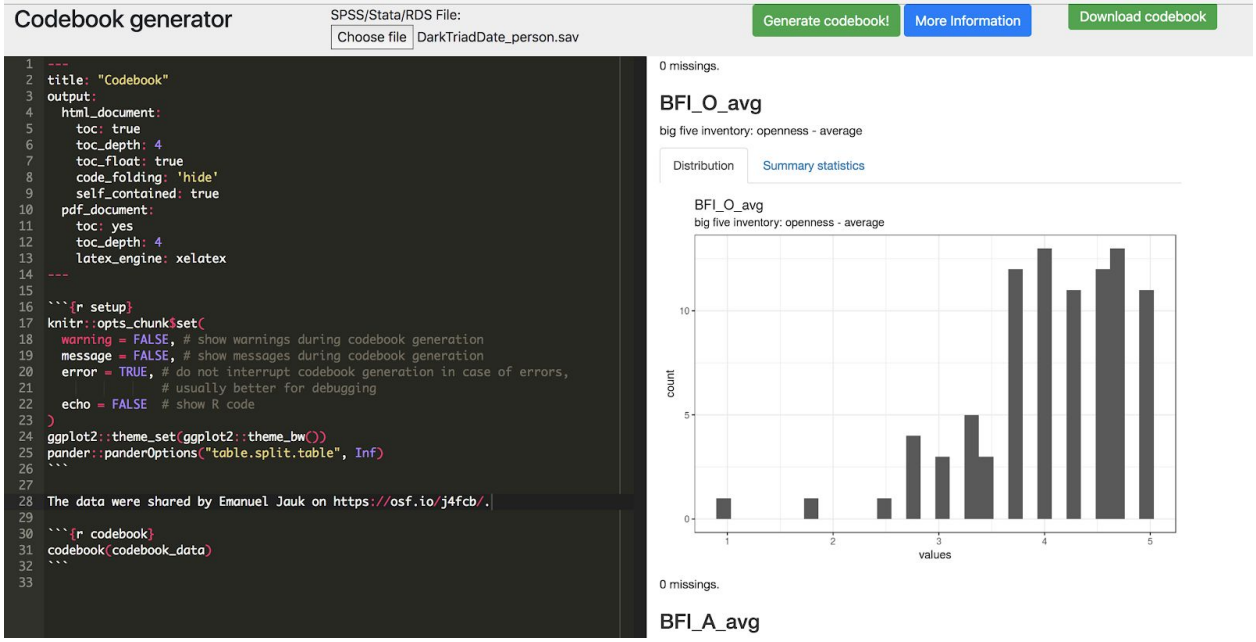

Figure 1. A screenshot of the codebook web app after a codebook has been generated.

Now, often a dataset imported from SPSS or Stata will not have all its missings set correctly. Especially SPSS users often simply code missings as 99 or 999, but don't actually label these as missing value placeholders in SPSS. For this, the webapp runs the function *detect_missings.* When its argument *ninety_nine_problems* is set to true, variables containing the value 99 or 999 (where that value is not in the plausible range of the other values for that variable) will be set as missing values. Depending on whether the argument *only_labelled_missings* is set to true, this will only happen, if the value 99 or 999 has a label. A similar option is available for negative values, a common convention used by Stata users.

The webapp is an easy entrypoint to generating codebooks, especially for researchers who do not use R, have properly marked up SPSS or Stata files, but want to share the metadata in an open format. The webapps tries to set reasonable defaults and it is possible to edit the text and the R code to improve the resulting codebook. However, the webapp does not store edits, is not as interactive as working in R, and it requires the user to upload the dataset to a server. This is not permissible for certain restricted-use datasets. Moreover, for very large datasets, you may get an error message, because the server limits the resources you can use. If one wants to document large, private, or many datasets, or if one needs to first add the metadata in R, it is easier to install the codebook package locally.

Using the codebook package locally in R and RStudio

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, rmarkdown editing, and addins), but it works independently of it. To install the package, simply run the following command in R

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

Then, you can use the package in several ways. For now, I will assume that you have a dataset where the relevant metadata (e.g. variable and value labels) is already set appropriately. I will discuss adding and changing metadata later.

Checking the codebook during analysis

Often, during data analysis, we want to confirm that a variable is the one we intend to use, that values are in the correct order, and so on. To this end, you can use the Addins menu in the top bar in the RStudio window and choose the "Static Label browser". The static label browser shows the variable and value labels for a dataset in the bottom right viewer pane of RStudio. It selects the dataset which is alphabetically first in the environment, or if text is selected in the editor, the dataset with the name of the selected text. You can also pick a data frame by typing *label_browser_static(data_frame_name)* in the console.

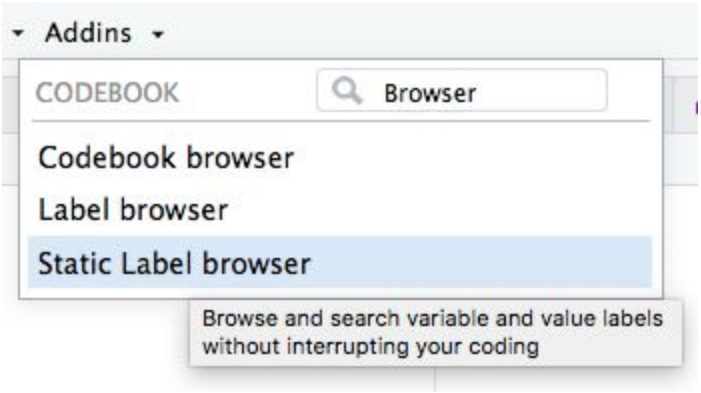

Figure 2. Opening an addin.

In this case, only the *bfi* dataset is loaded. In the viewer tab, you can now see the variable and value labels. The static label browser allows you to keep executing R code. The codebook browser and the dynamic label browser have the advantage of allowing you to select a dataset 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 only be executed after they have been closed.

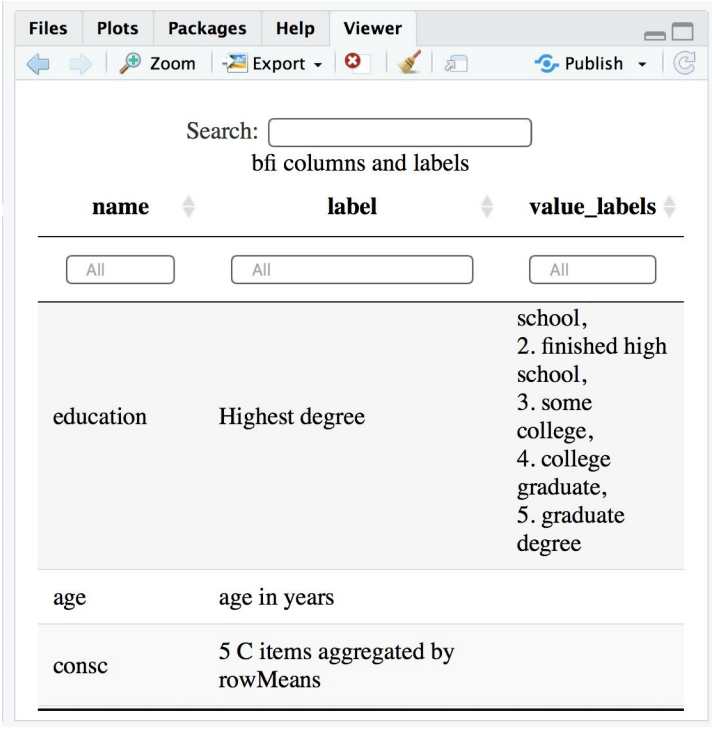

Figure 3. The variable and value labels as seen in the RStudio viewer tab.

Generating a codebook for future reference

To get the same codebook generated by the webapp locally, you need to work with rmarkdown (Allaire et al., 2018). In RStudio select File -> New file -> R Markdown[2] and edit the resulting file. The document in the webapp can serve as a useful starting point, so copy that into the window. Now, all we need is data. Rmarkdown documents have to be reproducible and independent of the current session state, so it is not enough for a dataset to be loaded locally. Assuming you have a file called *esm.sav* in your project folder, write the following on line 31 to import it:

```
codebook_data <- rio::import("esm.sav")
```

Now, you can click the Knit 

button at the top of your document. The codebook should be rendered in the viewer tab. In the viewer tab, you can click a button with a window and an arrow on it to view it in a bigger browser window. A file was also generated in your project directory now. You can open this file to see the codebook.

Adding and changing metadata

Currently, the codebook package focuses on metadata stored in R attributes and can export this data to PDF, HTML, spreadsheets, and JSON. So, what are attributes and how does metadata end up there? Attributes in R are most commonly used to store the type of a variable. A date in R is just a number with attributes (a time zone and a class). However, these attributes can just as easily store other metadata. The Hmisc (Harrell, 2018), haven (Wickham & Miller, 2018), and rio (Chan & Leeper, 2018) packages, for example, use them to store labels. The haven and rio packages set these attributes when importing data from SPSS or Stata files. However, it is also easily possible to add metadata ourselves:

```
attributes(bfi$BFIK_consc_2R)$label <- "I am lazy"
```

Here, we assigned a new label to a variable. Because it is inconvenient to write repeatedly, the *labelled* package (Larmarange, 2018) adds a few convenience functions.

```
var_label(bfi$BFIK_consc_2R) <- "I am lazy.")
```

or to label multiple items in a data frame

```
var_label(bfi) <- list(BFIK_consc_2R = "I am lazy", BFIK_consc_1
= "I do tasks properly.")
```

We can also label values in this manner:

```
val_labels(bfi$BFIK_consc_2R) <- c("totally agree" = 1, "don't
care" = 3, "totally agree" = 5)
```

Many already have a codebook in the form of a spreadsheet and want to import this. An example of this can be found in the manual labelling vignette for the codebook package. Further information about adding labels with the labelled package can be found in their vignette[3].

---

[2] Please note that codebooks will not display properly in R *notebooks*.

[3] https://cran.r-project.org/web/packages/labelled/vignettes/intro_labelled.html

Other attributes can be added too. For example, the formr R package (Arslan, 2018b) and the qualtRics R package (Ginn, 2018) both add an "item" attribute which contains metadata about the widget used to ask the question. Unfortunately, there is a lack of standards for attributes going beyond simple labels. Please confer the Qualtrics vignette in the package for an example.

Adding scales

The codebook package also relies on a simple convention to be able to summarise psychological scales, such as the Big Five dimension extraversion, which are aggregates across several items. We can set the relevant attributes in R by manually documenting scales through commands such as

```
bfi$extra <- aggregate_and_document_scale(bfi %>% select(item1, item7, item9))
```

However, it is even easier when the item names follow a simple convention, namely the format scale_number_R (e.g. bfi_extra_1R for a reverse-coded extraversion item, bfi_neuro_2 for a neuroticism item). If variables are (re)named in this way, we can simply run a function that finds items with different numbers but identical stems

```
bfi <- detect_scales(bfi)
```

This function assumes that high item values reflect a high value on the scale. If this is not the case, i.e. for reverse-coded items, a command using dplyr (Wickham, François, Henry, & Müller, 2018) functions and the *reverse_labelled_values* function can easily remedy this.

```
library(dplyr)
bfi <- bfi %>%
mutate_at(vars(matches("_R$")), reverse_labelled_values)
```

All this statement does is find variable names which end with _R and reverse them. Please note that this function can only work automatically if the highest and lowest possible values are encoded in the labels or levels attribute of the variable. If this is not the case, codebook cannot infer the range of the values. For further reading, see the manual labelling vignette in the package.

Losing and rescuing attributes

Attributes in R have been around for a long time, but they can easily be lost during data processing. Whereas the tidyverse (Wickham, 2017) suite of packages usually preserves them, many operations in base R can delete attributes. Similar problems occur when reshaping data with tidyr (Wickham & Henry, 2018). To this end, the codebook functions simply has the function *rescue_attributes.*

```
restored_df <- rescue_attributes(df_lost_attributes, df_with_attributes)
```

It will recover attributes for variables that were not renamed. Sometimes, we may also want to get rid of attributes, because some R functions expect vectors without attributes. To this end, we can use *zap_* functions to destroy value *labels* or the variable *label*, the *labelled* class

or all custom *attributes*. These functions can be applied to single variables or the entire data frame.

Core features of the generated codebook

The codebook package attempts to make full use of all metadata that it can find. Sometimes this may mean that variables have to be renamed.

Survey response rates, durations

If a data frame has the column *session* to identify participants, and the column *created, modified, ended*, and *expired* in the form of datetimes, the codebook package can calculate a few commonly desired summaries about participation. It can give the number of participants, and the number of rows per participant. It can show on which dates and times people enrolled, and by subtracting *created* from *ended* how long it took participants to fill out the survey. By checking whether *modified* is missing, it can differentiate people who filled out any information in the survey from those who did not. By checking whether *expired* is missing, it can determine how many did not finish the survey in time. These column names reflect the standard names in the survey software formr.org (Arslan, Walther, & Tata, 2018), but of course columns from other survey providers can be renamed accordingly if they contain the same information. Pending agreement on standard names for psychological variables, the codebook package will respect those.

Reliability estimates and Likert plots

The codebook package automatically calculates an estimate of reliability for all defined scales. By default, this will be done using the *alpha* function in the psych package (Revelle, 2018) if there is just one row per participant. If there are exactly one or two rows per participant (*session* column), and the *created* column can be used to order them, the package will calculate internal consistencies for both time points and a retest reliability. If there is a variable number of rows per participant, the multilevel.reliability function in psych reports the generalizability of change, the person average, and so on (Shrout & Lane, 2012). The scale summary also includes a Likert plot (Figure 4) generated using the likert package (Bryer & Speerschneider, 2016).

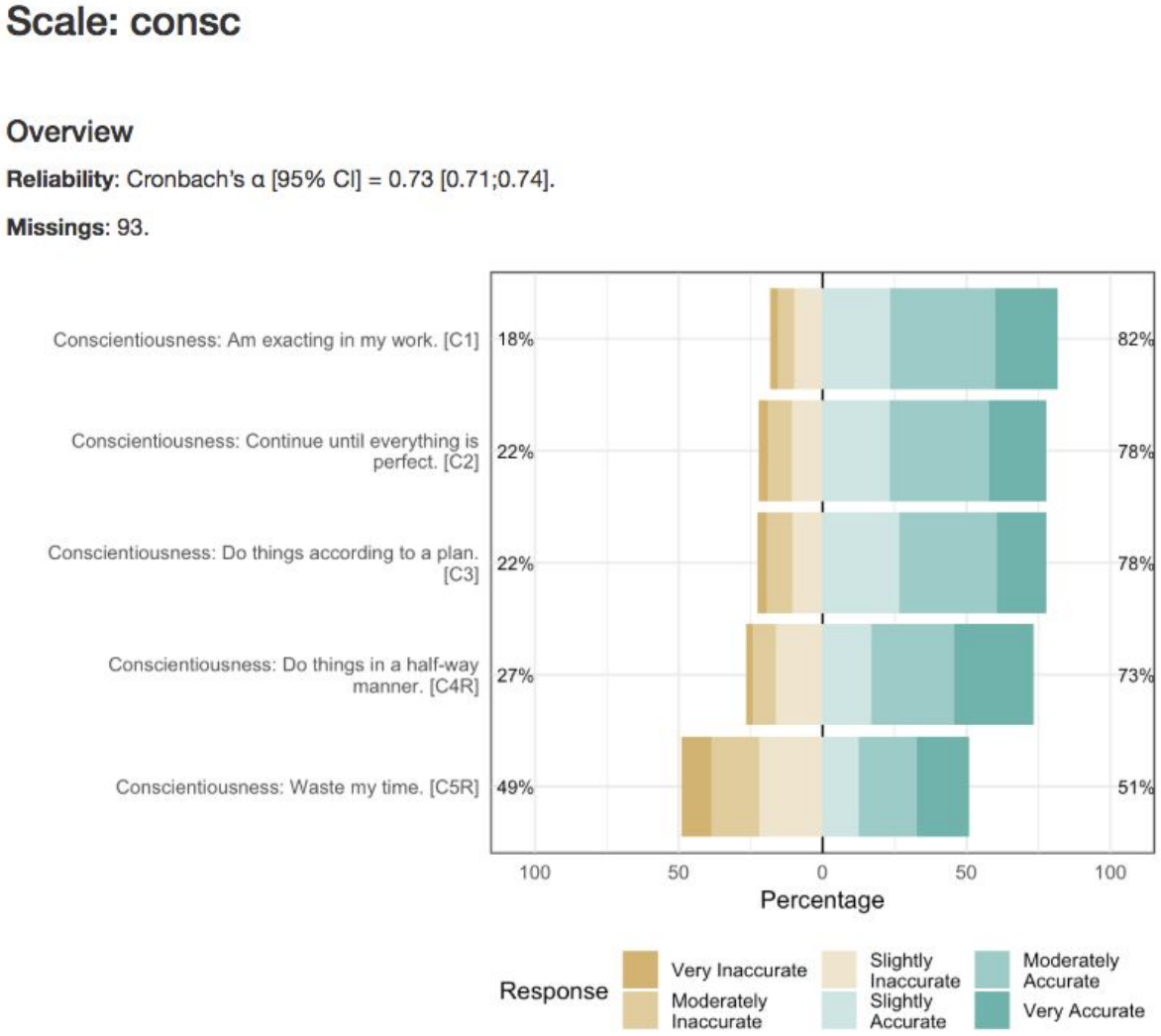

Figure 4. A Likert plot for the conscientiousness scale in the bfi dataset in the psych package.

Distribution plots and descriptives

The codebook package also shows a plot of the distribution for all individual items and all scales, except for large numbers of unique values (e.g. free text responses). These plots are labelled using the variable name, label, and value labels. They can be generated in isolation using the `plot_labelled` function. Further, using the skimr package (McNamara, Arino de la Rubia, Zhu, Ellis, & Quinn, 2018), each item or scale is accompanied by a compact summary of the data, i.e. missings, means, ranges, standard deviations for numeric data types, top counts for categorical types, and so on for dates, texts, and other data types. If there are labelled 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 missings in a separate plot.

Missingness patterns

Whereas the number and types of missings (if missings are labelled) is always summarised for each item, this does not give a prospective analyst the answer to the question how many data points have non-missing data for a bivariate or multivariate analysis. To this end, the codebook package displays a table of missingness patterns towards the end. Here, we can see the number of complete cases, cases with missings in one variable, frequent culprits for missings, but also whether there are common patterns of missings (e.g. a filter question and its dependents).

Parallelisation

The sequential summary of all items and scales in graphs and tables can take some time. Computing multilevel reliabilities in large data frames is even more demanding and may even fail on desktop computers. To solve this, the codebook package implements the *future* API (Bengtsson, 2018b). Simply by calling

```
knitr::opts_chunk$set(dev = "CairoPNG") # use an image device that works in parallel
library(future)
plan(multicore) # execute codebook functions on multiple cores if available
```

in the chunk preceding the call to codebook, the generation can be sped up significantly (i.e. from circa 24 hours for the General Social Survey to under three hours on a Macbook Pro with 8 cores). Using the *future* API and the future.batchtools package (Bengtsson, 2018a) it is even possible to pre-compute the multilevel reliabilities on a remote computing cluster.

```
reliabilities_diary %<-% {
future::value(future({codebook::compute_reliabilities(s3_daily, 'repeated_many')}) # the future statements are doubly nested so that compute_reliabilities is processed by a node, not the cluster scheduler
```

The result then has to be passed to the codebook function. Please confer the documentation of the future package for more information.

Codebook table

At the end of the codebook, we can see a rich codebook in tabular form (Figure 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., 2018)*. You can also directly create this table by calling the function *codebook_table*, but to share codebooks with others, we

recommend also making the HTML document available which contains the metadata in a search-engine-friendly format.

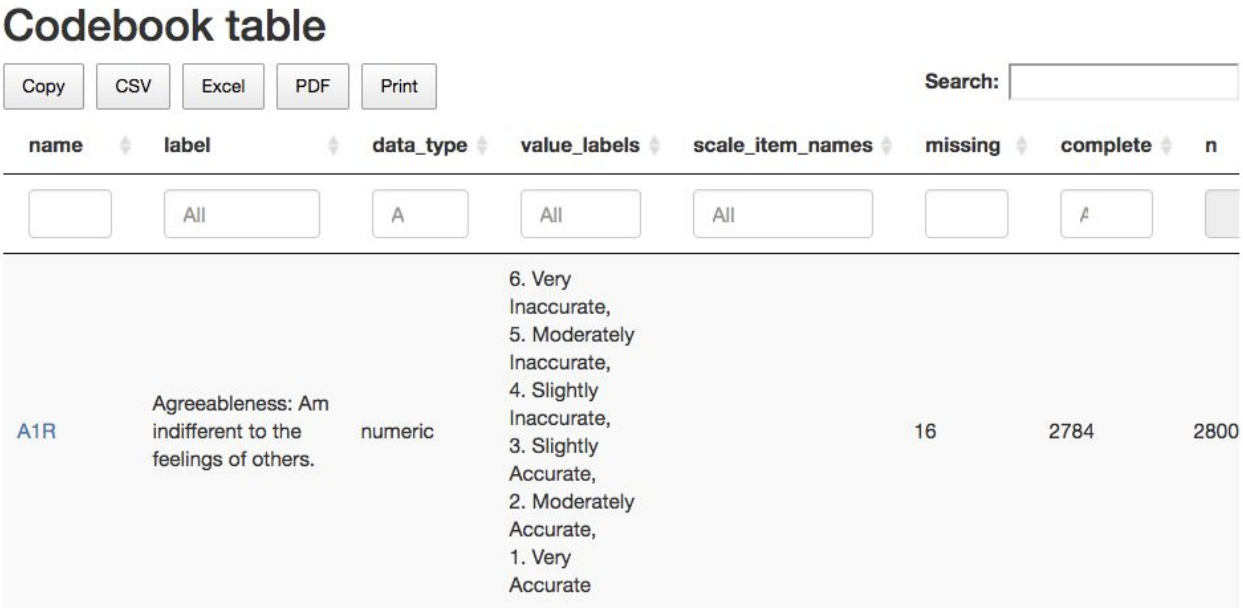

Figure 5. The first row of a codebook table.

JSON-linked data (JSON-LD)

Invisibly to the researcher, the codebook package also writes JSON-LD into the HTML documents. This data follows a flexible and extensible standard for metadata. Although it is presently quite limited, efforts are underway to extend it for use in biology, and soon psychology. Currently, codebook's implementation is a proof-of-concept. An abbreviated example on a Big Five dataset could look like this.

```
      {
        "name": "bfi",
         "keywords": ["A1R", "A2", "A3", "A4", "A5", "C1", "C2", "C3",
"C4R", "C5R", "E1R", "E2R", "E3", "E4", "E5", "N1R", "N2R", "N3R",
"N4R", "N5R", "O1", "O2R", "O3", "O4", "O5R", "gender", "education",
"age", "consc", "extra", "open", "agree", "neuro"],
         "variableMeasured": [
           {
             "name": "A1R",
             "description": "Agreeableness: Am indifferent to the
feelings of others.",
             "value": {
               "Very Inaccurate": 6,
               "Moderately Inaccurate": 5,
               "Slightly Inaccurate": 4,
               "Slightly Accurate": 3,
```

```
            "Moderately Accurate": 2,
            "Very Accurate": 1
          },
          "maxValue": 6,
          "minValue": 1,
          "@context": {
            "@vocab": "http://pending.schema.org/"
          },
          "@type": "propertyValue"
        }
...
}
```

In the future, psychologists could extend schema.org to document certain psychological measurement scales, types of datasets and research designs, but even single items such as demographic questions. This is an open and community-driven process. Other research communities such as health and life sciences or biological sciences have started these processes on websites such as https://health-lifesci.schema.org/ or http://bioschemas.org/. Discussions about extending schemas often take place on Github. The Society for the Improvement of Psychological Science has formed a workgroup for a psychological data specification. Interested parties should contact Melissa Kline.

Summary

Standardised, metadata-rich codebooks are useful to the data creators, their teams, and the scientific community. The inconvenience and time investment needed to make such codebooks may have contributed to the current state of affairs in psychology, where codebooks are frequently unstandardised, lack information that is essential to understand the data. Datasets are not always made available in open formats, and usually not machine-readable, disabling search capabilities. The codebook package makes some common tasks easier, speeding up the data cleaning and summary process. We hope this encourages researchers to generate rich codebooks, which then benefit the scientific community. A working codebook can be generated by inexperienced users within minutes. If certain conventions are followed or specific survey providers used, the package may even save researchers time on net, by taking over the tasks of graphing distributions, computation of descriptives, and estimating reliabilities.

Acknowledgements

I am grateful to Martin Brümmer for help setting up the proof-of-concept for JSON-LD and to early users like Christoph Schild, Caroline Zygar, Daniël Lakens, and Mark Brandt who reported bugs in earlier versions of the package.

References

Allaire, J. J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., … Chang, W. (2018). *rmarkdown: dynamic documents for R*. Retrieved from https://CRAN.R-project.org/package=rmarkdown

Arslan, R. C. (2018a). *Cook codebooks from survey metadata encoded in attributes in R*. doi:10.5281/zenodo.1326520

Arslan, R. C. (2018b). *formr.org survey framework utility R package*. doi:10.5281/zenodo.1326523

Arslan, R. C., Walther, M., & Tata, C. (2018). formr: A study framework allowing for automated feedback generation and complex longitudinal experience sampling studies using R. doi:10.31234/osf.io/pjasu

Bengtsson, H. (2018a). *future.batchtools: A Future API for Parallel and Distributed Processing using "batchtools."* Retrieved from https://CRAN.R-project.org/package=future.batchtools

Bengtsson, H. (2018b). *future: Unified Parallel and Distributed Processing in R for Everyone*. Retrieved from https://CRAN.R-project.org/package=future

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), 2. doi:10.25035/pad.2015.002

Bryer, J., & Speerschneider, K. (2016). *likert: analysis and visualization of Likert items*. 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*. 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. Retrieved from https://CRAN.R-project.org/package=shiny

Condon, D. M. (2017). The SAPA Personality Inventory: An empirically-derived, hierarchically-organized self-report personality assessment model.

Ginn, J. (2018). *qualtRics: download Qualtrics survey data*. Retrieved from https://jasperhg90.github.io/qualtRics/

Google. (2017, January 24). Facilitating the discovery of public datasets. Retrieved August 2, 2018, from http://ai.googleblog.com/2017/01/facilitating-discovery-of-public.html

Google. (2018, August 2). Dataset Search. Retrieved August 2, 2018, 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*, 160044. doi:10.1038/sdata.2016.44

Harrell, F. E., Jr. (2018). *Hmisc: Harrell Miscellaneous*. Retrieved from https://CRAN.R-project.org/package=Hmisc

Introduction to Wikidata. (n.d.). Retrieved September 4, 2018, from https://www.wikidata.org/wiki/Wikidata:Introduction

Larmarange, J. (2018). *labelled: manipulating labelled data*. Retrieved from https://CRAN.R-project.org/package=labelled

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*(2), 210–215. doi:10.1159/000434720

McNamara, A., Arino de la Rubia, E., Zhu, H., Ellis, S., & Quinn, M. (2018). *skimr: compact and flexible summaries of data*. Retrieved from https://CRAN.R-project.org/package=skimr

Ooms, J. (2014). *The OpenCPU system: towards a universal interface for scientific computing through separation of concerns*. *arXiv [stat.CO]*. Retrieved from http://arxiv.org/abs/1406.4806

Rasmussen, K. B., & Blank, G. (2007). The data documentation initiative: a preservation standard for research. *Archival Science*, *7*(1), 55–71. doi:10.1007/s10502-006-9036-0

Revelle, W. (2018). *psych: procedures for psychological, psychometric, and personality research*. Retrieved from https://CRAN.R-project.org/package=psych

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: Guilford Press. Retrieved from https://nyu.pure.elsevier.com/en/publications/psychometrics

Wickham, H. (2017). *tidyverse: easily install and load the "tidyverse."* Retrieved from https://CRAN.R-project.org/package=tidyverse

Wickham, H., François, R., Henry, L., & Müller, K. (2018). *dplyr: a grammar of data manipulation*. Retrieved from https://CRAN.R-project.org/package=dplyr

Wickham, H., & Henry, L. (2018). *tidyr: Easily Tidy Data with "spread()" and "gather()" Functions*. Retrieved from https://CRAN.R-project.org/package=tidyr

Wickham, H., & Miller, E. (2018). *haven: import and export "SPSS", "Stata" and "SAS" files*. Retrieved from https://CRAN.R-project.org/package=haven

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A., … Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*, 160018. doi:10.1038/sdata.2016.18

Xie, Y. (2018). *DT: a wrapper of the JavaScript library "DataTables."* Retrieved from https://CRAN.R-project.org/package=DT