

# The PIAAC Variable Finder: An interactive Shiny app for cleaning, interpreting and analyzing Programme for the International Assessment of Adult Competencies data

---

Nate Breznau, [breznau.nate@gmail.com](mailto:breznau.nate@gmail.com), 

German Institute for Adult Education – Leibniz Centre for Lifelong Learning

## Abstract

The release of an interactive app for searching through PIAAC documentation, variables and metadata. Details the process of building the app, using Gen AI and how the app can support researchers seeking to use PIAAC data more efficiently or build their own similar apps.

Keywords: PIAAC, data analysis, shiny, text scraping, app development

Funding: This project was partly funded from a German Science Foundation (Deutsches Forschungsgemeinschaft) project number 464546557

The most widely used data for comparative research on adult skills in rich and middle income countries is the *Programme for the International Assessment of Adult Competencies* (PIAAC). The various local and international surveys fielded as part of this Organisation for Economic Co-Operation and Development (OECD) project measure human capital in the form of literacy, numeracy and problem-solving in technology-rich environments. It covers adults aged 16-65, with some countries sampling from older populations. With the release of PIAAC Cycle 2 in 2025, there are now two waves of data from 24 countries with an additional 19 that participated in one of the two Cycles<sup>1</sup>. Since 2018 researchers produced over 150 publications per year using PIAAC data in some form (Maehler et al., 2025).

The PIAAC survey is large and complex. It has sophisticated item-response methods to predict adult skill scores, population weighting, and a detailed background questionnaire covering education, work, demographics, behaviors and attitudes. The data is available in open access Public Use Files (PUFs), requiring only name, email, and usage goals prior to downloading<sup>2</sup>. Changes between the two Cycles of PIAAC (roughly 2012 and 2023) increase the complexity of this massive data project. To fully understand all aspects of the data and the differences between the Cycles, a user needs at least four separate types of documents each in a different format. It can be challenging and confusing to have some or all of these documents open simultaneously, and to cycle through them searching for variables or other information use different software's native search functions for each type.

In my own attempts to understand and analyze PIAAC data, it became clear to me that these diverse documents could be integrated. They all contain stable variable names. This stable "ID" allows for the compilation of a dataset with metadata linked to each variable. With this in mind, I scraped the documents and combined their data, then built an interactive app for searching through the combined data that I could run locally to support my data cleaning, wrangling, analysis and visualization efforts. This allowed me to work much more efficiently, for example to quickly find any word that was used in the questionnaire during the survey, identify response categories and their values, identify if a variable was a trend variable between the Cycles or not, and check if a variable contained non-missing values (i.e., that it was not withheld due to data protection guidelines to protect the identities of survey respondents).

I relied consistently on Gen AI to develop the app, in this case ChatGPT about 95% of the time. It was crucial that I had strong pre-existing skills in the R statistical programming language, the Shiny package, and basic knowledge of working with text and scraping.

---

<sup>1</sup> There are also two predecessors to the PIAAC that measure similar adult skills across a handful of countries: The *International Adult Literacy Skills Survey* (IALS) and the *Adult Literacy and Life Skills Survey* (ALL).

<sup>2</sup> Access here: <https://www.oecd.org/en/about/programmes/piaac/piaac-data.html>. Note: There are Scientific Use Files (SUFs) as well which contain fewer redacted variables, but require minor security clearance and a usage agreement for access.

Assuming others could benefit from this app, I share the process of developing the app in this paper for others to follow. I also deployed the app on a Web-platform for those who want to use the app without running it in their own computing environment. The many teams of researchers and administrators producing PIAAC documentation did amazing work, and this app is simply an augmentation. Thus, this paper details both, A) the development of that tool – a demonstration of how any researcher with foundational R skills could build something similar, and B) a user’s guide to the tool, which I named the “PIAAC Variable Finder”.

Readers can use the interactive app via a Web browser<sup>3</sup>, access the entire workflow to construct the app on Github<sup>4</sup>, or simply download a zipped folder to run the app locally<sup>5</sup>. This open source approach allows users to expand and customize the app for their purposes. For example, I only scraped the British English and German versions of the background questionnaire, but more languages could easily be incorporated. It also shows how such apps could be developed in a general way potentially motivating practicing social scientists like myself to consider basic science tool development in their work.

#### **A. Building an interactive app, circa 2025**

Background knowledge was necessary to fuel my idea of having an interactive app. This included how document text scraping works, and the software that could perform this scraping. In this case I know R (R Core Team, 2024), and use it in the R Studio IDE (Posit Team, 2025) which has several advantages: The IDE was specifically designed to run R statistical software and is compatible with Git allowing for easy version control and open source sharing. The programming language R is also ideal because it was developed by scientists for scientists.

Secondly, knowledge of the Shiny package was necessary. Shiny is a powerful set of tools designed to enable interactive app development and deployment with minimal data science knowledge. I used Shiny tools in R previously, and therefore had a strong understanding of how they work, and what a Shiny app can and cannot do, in addition to how much (how little) coding is required to produce an app (Chang et al., 2025).

Third, based on my previous knowledge and experiences with Gen AI, I had enough motivation to build an app. My experience learning and deploying Shiny apps prior to AI was positive, but also a hurdle to future endeavors given the amount of time it could take to develop and troubleshoot them. My familiarity with how powerful Gen AI tools like ChatGPT are, and how effectively they help a user build applications in languages like R and Python that are well known and widely discussed on the Internet was a necessary condition to build

---

<sup>3</sup> [https://nate-brezna.shinyapps.io/PIAAC\\_Variable\\_Finder/](https://nate-brezna.shinyapps.io/PIAAC_Variable_Finder/)

<sup>4</sup> [https://github.com/nbreznau/PIAAC\\_Variable\\_Finder](https://github.com/nbreznau/PIAAC_Variable_Finder)

<sup>5</sup> [https://github.com/nbreznau/PIAAC\\_Variable\\_Finder/blob/main/Shiny.zip](https://github.com/nbreznau/PIAAC_Variable_Finder/blob/main/Shiny.zip)

this app. My assumptions were correct: It took one day of dedicated work to build a rudimentary but working app, and 2-3 more to have something that already could improve my PIAAC data analysis goals. This would not have been possible prior to the rise of Gen AI in 2021 (Breznau & Nguyen, 2025).

Fourth, of course, was that I had enough familiarity with both Cycles of PIAAC data to know where to find the documentation, and experience with survey data analysis to identify what I needed for optimizing PIAAC work.

### *Building the database*

There are four types of information contained in four document types that I needed to best perform scientific analyses using PIAAC Cycles 1 and 2 in combination. These are the *questionnaires*, *variable labels*, *longitudinal trend identification*, and *derived variable* details for each Cycle.

Any skilled survey data analyst will tell you that the *questionnaire* is essential because it gives verbatim questions and response categories from the survey. A social scientist cannot determine the face validity or meaning of questions without reading them. For all major languages in all countries surveyed there is a different questionnaire document provided in HTML format<sup>6</sup>. As I operate in English and German, I built the app using the German and United Kingdom (British English) questionnaire documents which are available as HTML files. Others could easily follow this workflow and add more languages.

When working with the PIAAC data there are variable names and variable labels. After an analyst downloads the PUFs as CSV files, they only contain variable names. Thus, they need a second type of documentation for *variable labels*. This comes in Stata (Cycle 1 and 2) and R (Cycle 2 only) routines<sup>7</sup>. Although it is tempting to simply rely on variable labels to understand the variables, they can be misleading without the questionnaires. An example taken from PIAAC sheds light on this. In both Cycles there is a “political efficacy” question. The variable label suggests that the question asks about influence on government. But only by looking at the questionnaire is it possible to identify exactly what this means.

In Cycle 1 the question is “To what extent do you agree or disagree with the following statements? People like me don't have any say about what the government does?” and in Cycle 2 it is “How much would you say the political system in [country] allows people like you to have a say in what the government does?”. These questions both ask about political self-efficacy, but in different ways. We have known for most of the history of survey research that even small wording differences in survey questions tap different ranges of meanings among

---

<sup>6</sup> For each country's questionnaire <https://www.oecd.org/en/data/datasets/piaac-2nd-cycle-database.html>

<sup>7</sup> Or download the files in SPSS format and convert them.

respondents (Bishop et al., 1978). In this case the questions are completely different in wording – they lack content validity in measuring an identical thing.

This is a case where an analyst might try to compare these variables over time without checking the questionnaire, and generate non-sensical output. Thus, a third document is necessary to easily identify *longitudinal trends*. This allows a user to only look at the questionnaire for understanding rather than identifying trends. The PIAAC released a PDF document “PIAAC Cycle 2 BQ Draft Conceptual Framework” (ROA & GESIS, 2025) which details the comparability of variables across the two Cycles. It identifies items that are identical, or so close that they constitute a “strict” (‘hard’) trend – in other words that they have both face and content validity, or instead constitute a “soft” trend with only face validity on top of slight changes to the contents, such as question wording, response wordings or both. Furthermore, this document identifies which variables are non-trends, like the political efficacy variable just mentioned. Finally, it identifies which questions and corresponding variables were dropped after Cycle 1. Of course, an analyst could use the questionnaire to compare between Cycles to identify trend variables, but this is tedious work.

Another issue is that there are many derived variables, some with extremely complex coding schemes and information that is not fully in the questionnaire. One example of this is participation in adult education and learning (ALE). The coding rules for derived variables are in a third document to which I will return to shortly. Measurement of ALE derives from various combinations of 20 questions. For example, the variable “NFE12” appears in Cycle 1, which refers to participation in any organized work or non-work related non-formal education (‘NFE’), meaning that is not part of initial educational qualifications (primary, secondary, tertiary) even if these are being sought as an adult. The interviewer gives a show card to demonstrate a concept of what qualifies as ALE for the respondent. Then they are asked to report on four types of ALE (distance, on-the-job, seminars or workshops, and private or other courses not mentioned in the previous three). In each of the four cases a follow-up question is asked if they answered “yes” to determine the number of courses. Then after determining the most recent, several other follow ups are asked about the format, such as whether it related directly to their work and their motivation for participating. The FET12 variable itself is “yes”, “no” or “missing”, but the question cannot be understood without digging into all the derived variables.

In Cycle 2 a nearly identical variable appears “NFE12C2”. For many trend variables a “C2” is added to denote the same variable in Cycle 2. However, in this case the variable is not considered a trend variable according to the PIAAC team. To unpack why requires tracing the complex question wording and variable derivation in Cycle 2. To determine that they should not be compared as a trend, in other words cannot easily be understood by reading casually through the questionnaire. Thus, understanding derived variables such as NFE12, requires

the *codebooks* which are in Word DOCX formats. Each Cycle as a derived variables codebook with corresponding coding rules to generate the derived variables (OECD, 2015, 2025). The only information I extracted from these for app development was to identify if a variable was derived or not.

My workflow uses scraping methods to get verbatim question wording from the HTML files into a dataframe (files *parse\_questionnaires\_cycle1.R* and *parse\_questionnaires\_cycle2.R*). I converted the Stata (Cycle 1 saved as *label\_data\_PIAAC\_Cy1.csv*) and R (Cycle 2 saved as a dataframe (*label\_data\_PIAAC.RDS*) variable label documents into CSV files which are imported and merged with the dataframe (routing *parse\_labels.R*), conversion of the longitudinal trend PDF into a Word document using Word which preserved the tables containing the trend data with minor conversion bugs requiring hand editing and then saving as CSV (*trend\_vars.csv* imported and cleaned via *trend\_var.R*), and finally copying a list from the derived variables document and saving it as a CSV file (*derived\_vars.csv*) and putting everything together with the main routine *prep\_shiny.R*.

### *Building the Shiny app*

I had two general goals with the app. To have the most information for any given variable in one place, and to find any information in PIAAC by searching through all the documentation text. I imagined a search bar on the left-hand side, where users could put in variable names or any text that appears anywhere in variable labels, question wording and responses in English or German. I also imagined a list on the right-hand side for the search results with the variable name, label, and which Cycle it appeared in (Cycle 1, Cycle 2 or both).

Given my skill set, I was confident I could build this; however, I wanted to prompt Gen AI before starting for two reasons. The first is simply to save time. I am not an expert R Shiny programmer, and getting going each time is painfully slow. I can work my way through it, as I have in other apps which I built prior to using Gen AI<sup>8</sup>, but if Gen AI can build it through prompting, I could save potentially days if not weeks of work. Second, prompting Gen AI could teach me about features in Shiny app design that I am not aware of, and that might contribute to a better tool. Appendix Prompt 1 lists the exact prompt and corresponding output. Although the entire Gen AI prompting history is quite long, this single prompt might have saved the most time, because it almost instantly built a framework for a working Shiny app that could then be de-bugged and tweaked.

I faced many further issues for development, but my motivation to push forward was very strong after such a successful first prompt. I discovered that R does not yet have a package specifically designed to deploy a Boolean search. Therefore, I added two search boxes, the

---

<sup>8</sup> <https://nate-breznau.shinyapps.io/shiny/>

second of which searches within the results of the first search. I added the most important meta-data based on my needs, so that the user can find the variable label, question text and responses in English or German, identify if it is a trend variable, and get a list of related variables for example the name of the trend in the other Cycle, variables measuring similar things or those used to construct the variable.

As I am a practicing social scientist rather than a developer, I simply kept a list of bugs that I encountered or ideas that would improve the app while using it for my own research. This means that I simply added lines of code to the script that would recode values in the dataframe (in *prep\_shiny.R*). A developer would probably go back and adjust the scraping routine or improve automation, but I have much simpler goals and am not interested in using up my time for such activities. Moreover, I wanted to show that even non-expert R users can build practical tools.

Some notable benefits of working with Shiny are fluid layouts. The infrastructure of Shiny means that all the heavy development lifting is done for me. Simple packages like `fluidPage()` generate outputs that seamlessly adjust to screen sizes and move text and fields around to fit. Using base R and tidyverse commands allows filtering the dataset reactively. Thus, as soon as users type anything in the search or tick boxes, the filtering starts. Also, it allows the users to click (or toggle on a phone) on a single result in the results list. Doing this activates that particular variable which populates the display in a lower panel with all the meta-data about that variable.

I was able to add some basic instructions and an attribution, along with a logo for my institute into the left-hand side. With the help of Gen AI, I embedded CSS for improved wrapping on screen without the need for horizontal scrolling. Advanced filtering options with check boxes were added to allow users to view only trend variables and filter out results for variables that have no non-missing values in either of the two countries or Cycles individual-level data – this occurs due to data protection policies that got much stricter with Cycle 2. This information was extracted from the actual PIAAC PUFs in the routine *prep\_shiny.R*.

The app runs locally on any recent R version so long as the user installs all the necessary packages. They can simply download and extract the Shiny folder, open the file `app.R` in R Studio and a button “Run App” automatically appears. Click and it runs. At the same time I thought that for the benefit of non-R-users, or people on the go, it is helpful to have a publicly accessible tool. For this, shinyapps.io provides a low-cost solution to host the app so that users can access it via the Internet and a Web browser. The shinyapps.io service builds a virtual computing environment ensuring that the app will always function even if updates to packages or R itself reduce or remove functionality. This has the benefit that if I no longer maintain the app on Github, or have not updated it recently, users can still access it online.

## B. The PIAAC Variable Finder, a user's guide

This section is a condensed version of the “User’s Guide” which can be accessed by the identically named link in the left-hand panel of the Shiny app. The link is dynamic, so it will work either on the Web-based version by linking directly to the guide on Github, or on the local version by opening the PDF file in the Shiny/www folder inside the project folder.

**Figure 1. PIAAC Variable Finder User Interface**

PIAAC Variable Finder

Main Search 1

Search Within Results 3

☐ Show Trend Variables 2

☐ Show Only PUF Non-Missings 5

Search in German or English (British) through the questionnaires from Cycle 1 and Cycle 2. You can also search variable names. Click on result to see more info.

[User's Guide](#) 6

License CC BY4.0  
Please cite as:  
Brezna, Nate. 2025. PIAAC Variable Finder.  
German Institute for Adult Education - Leibniz Center for Lifelong Learning.

Please report bugs or ideas:  
Email

**PIE** German Institute for Adult Education  
Leibniz Centre for Lifelong Learning

Show 7 entries

Search:

	variable	display_label	cycle
1	AETPOP	Adult education/training population – excludes youths 16-24	Cycle 1
2	AETPOPC2	Adult education/training population – excludes youths 16-24	Cycle 2
3	AGEG10LFS	Age in 10 year bands (derived)	both
4	AGEG5LFS	Age groups in 5-year intervals based on LFS groupings (derived)	both
5	AGRE	Agreeableness dimension z-score based on 3 items/BFI-2-XS (derived)	Cycle 2
6	BIRTHRGN	Country of birth (9 regions - derived)	both
7	BORNLANG	Interactions between place of birth and language status (derived)	both

Showing 1 to 7 of 1,840 entries

Previous 1 2 3 4 5 ... 263 Next

VARIABLE: AETPOP  
LABEL: Adult education/training population – excludes youths 16-24  
CYCLE: Cycle 1  
NON-MISSING in PUF: YES  
TREND STATUS: Derived TREND VAR: AETPOPC2

QUESTION TEXT EN: [derived var] 5  
RESPONSES EN: 0 Excluded from AET population, 1 AET population

QUESTION TEXT DE: NA  
RESPONSES DE: Skala: binär (kategorial); 0 Aus der AET-Population ausgeschlossen; 1 AET-Population; .N Nicht angegeben oder abgeleitet.  
Related variables: AETPOPC2, AGE\_R, FE12, EDAT8, B\_Q02a, B\_Q02b, B\_Q03a, B\_Q03b, B\_Q05a

The user interface (UI) contains three primary regions. The left-hand sidebar has two search fields (Figure 1.1). The Main Search filters the entire database directly. It is not Boolean, so words and phrases must be exact. As of version 1.0, most fields have both English and German meta data. The app does not engage in any translation however, so in some cases the ostensibly same word in both languages could yield different results. The Search Within Results field searches within those results already showing.

Next in the sidebar are two check boxes that default as unticked (Figure 1.2). If the user ticks the Show Trend Variables box, it will filter the results to include only strict and soft trend variables. Corresponding to this is the “TREND STATUS” variable in Figure 1.5 which can have “NA” which means ‘not applicable’ and indicates no trend, “Derived” which indicates it is most likely a trend variable but the variable from the other Cycle has a different name (usually with “C2” at the end, “Cy 1 only” or “Cy 2 only” which is self-evidence, and “soft” or “strict”



indicating face (roughly speaking) or both face and content validity (strictly speaking) in the measure across the two Cycles. If the user ticks the Show Only PUF Non-Missings box, it will only show variables that have non-missing data in at least one cycle and at least one country (Germany or the United Kingdom). In other words, when this box is not ticked it will show many variables that exist in PIAAC but are all missing values in the PUFs. Corresponding to this variable in Figure 1.5 is the field for “NON-MISSING in PUF:”, which will populate with the value “No” for every variable when the box is ticked.

Figure 1.3 is a link to the User’s Guide which contains the same basic information relayed here, but with a bullet point structure and more details as part of the app’s technical documentation. Directly below this is the preferred citation. The DOI for the app works through Zenodo. This is a free to use service that has a Github plugin. It automatically generates a DOI and keeps up with newer versions of the repository supporting the app. Zenodo works stand alone and with many other workflow and repository services. Thus, Zenodo is an ideal open science tool.

Figure 1.4 is the main results panel in the center-right region of the app. It displays the first 7 variable results in alphabetical order (the order of the dataframe). I elected to display 7 per page, because on most devices this seemed to maximize space allowing the other two panels to be viewable on most single monitor screens. These features are all easily customizable. Once a user is familiar with Shiny, it is incredibly easy to use, especially by asking Gen AI. My point is that user’s are encouraged to customize their experience. In addition, feedback is welcome, therefore the final line at the bottom of Figure 1.3 (the side panel) is a link to my email to suggest modifications and report any bugs.

In the list for Figure 1.4 the user sees the variable name verbatim as it is in the PIAAC data in the column “variable”. Then under “display\_label” is the value for variable\_label\_en which is the metadata variable for the English label of the variable taken mostly from the PIAAC’s own labelling. There are exceptions because some countries have their own questions, especially when it comes to the education system. For example, the UK as a variable B\_Q01a3UK from the question, “Can you indicate which level in our national education system corresponds most closely with the level of this qualification?”. In Germany a similarly unique variable B\_Q01aDE1 is measured with the question, “Welchen höchsten allgemeinbildenden Schulabschluss haben Sie? Bitte sagen Sie es mir anhand dieser Liste.” These completely country-unique variables as a rule have the country’s two-digit alpha International Standards Organization (iso2c) classification in the variable name. Because these variables do not have labels, I use the question wording from the questionnaires as replacement – I programed it this way, so that if there is no label, then the question wording appears. The last column “cycle” lists from which Cycle the variable derives, or if it is in “both”.

Figure 1.5 is a more detailed breakdown of the highlighted variable. This appears blank when the app is launched, and is populated with values only after the user selects (clicks or taps) one of the seven rows in the list so that it is highlighted, i.e., active. Then nearly all available metadata for that variable appears below. Table 1 lists all variables in the metadata that I extracted from the PIAAC documentation and the PIAAC data itself. This provides an explanation for the metadata results in this section.

**Table 1. Metadata Variables in the Dataframe Behind the PIAAC Variable Finder**

meta variable	purpose	values	displayed?
variable	variable name in PIAAC data	verbatim from PUF files	yes
question_text_de	Question wording German	Verbatim from questionnaire <sup>a</sup>	yes
question_text_en	Question wording English	Verbatim from questionnaire <sup>a</sup>	yes
responses_de	Response wording German	Verbatim from questionnaire <sup>a</sup>	yes
responses_en	Response wording German	Verbatim from questionnaire <sup>a</sup>	yes
generic_label_de	Variable label German	From PIAAC documentation	no
generic_label_en	Variable label English	From PIAAC documentation	yes
constructed_vars	Rules for variable construction	From PIAAC documentation	no
ref_variables	A list of related variables, including those used to construct this variable <sup>b</sup>	If/then rules	no
cycle	Cycle identifier	<ul style="list-style-type: none"> <li>• Cy 1</li> <li>• Cy 2</li> <li>• both</li> </ul>	yes
trend	Identifies a trend variable	<ul style="list-style-type: none"> <li>• Strict</li> <li>• Soft</li> <li>• Derived<sup>c</sup></li> <li>• Cy1 only</li> <li>• Cy2 only</li> <li>• NA (not a trend)</li> </ul>	yes
trend_var	Identifies trend pair	If different between Cycles, the name of the corresponding trend variable	yes
soft_trend_explanation	Differences between Cycles	PIAAC's documented explanation	yes, if trend = "Soft"
c_vars	Related variables	Compiles base variables for derived measures, and similar variables	yes
notin	If all missing in PUF for DE & UK Cycle 1	1 = Yes	no
notin2	If all missing in PUR for DE & UK Cycle 2	1 = Yes	no
none	If missing for both Cycles and countries	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>	yes

<sup>a</sup>Except for derived variables which use the PIAAC documentation descriptive wording

<sup>b</sup>There uncertainty here (room for development) because of the diverse sources for this information

<sup>c</sup>"Derived" indicates a constructed variable that exists in both Cycles, thus theoretically a trend, but measurement can differ. Should be investigated on a case-by-case basis for comparison

### **C. Summary**

I described here the possibility to design an interactive app to support large scale assessment and survey data research. As an applied social scientist with only basic computer science and statistical software coding skills, but augmented by Gen AI, I was able to develop a working app in a few days. Within a few weeks' worth of work tweaking and refining the app, I had something that can support diverse researcher projects working with PIAAC data. Now I present a relatively clean version of the app to the scientific community.

The app allows a user to access information from diverse sources that otherwise requires manually searching through HTML questionnaires in different languages (one for each language and wave), the variable labels provided in either an Excel file or as integrative code (Stata and R) to automatically add them to the combined PIAAC data (one for each cycle), longitudinal trend identification via a PDF, and derived variable information in a Word document (one for each cycle). As such, the app can economize this otherwise complicated research process. It not only saves time and effort, but it also can help researchers interactively explore the PIAAC data. Researchers looking for ideas, or curious about what was asked will benefit from this app.

The app is uniquely designed for the PIAAC data, but the workflow here demonstrates that such an app could be built for any large-scale assessment survey (e.g., Programme for International Student Assessment) or any survey data with many waves, variations and complexities (e.g. International Social Survey Program or World Values Survey). This project demonstrates that social scientists can all be developers of tools. It also represents a transition in thinking toward metascience and meta-searching. Moreover, such activities can help shift our thinking away from publications as the only means for evaluating research activities and knowledge. Tool development, like generating public data or public science activities could be a welcome addition to evaluation criteria.

My limited experience with the world of app development, and contact with development communities, is that open source is of the highest value. Like FAIR data practices (Findable, Accessible, Interoperable, Reusable) open source app development with high quality workflows and documentation materials follows the ideals of open science and reproducibility.

## D. References

- Bishop, G. F., Oldendick, R. W., & Tuchfarber, A. J. (1978). Effects of Question Wording and Format on Political Attitude Consistency. *Public Opinion Quarterly*, 42(1), 81–92.
- Breznau, N., & Nguyen, H. H. V. (2025). *An Introduction to Generative Artificial Intelligence for Academics* (No. 14:655). F1000Research. <https://doi.org/10.12688/f1000research.166513.1>
- Chang, W., Cheng, J., Allaire, J. J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2025). *shiny: Web Application Framework for R* (v1.11.1). R Package Suite. <https://shiny.posit.co/>
- Maehler, D. B., Hernández Torrano, D., Courtney, M. G. R., Fischer, F. P., Hollricher, L. F., & Gorges, J. (2025). PIAAC Survey of Adult Skills: A review of the research landscape. *International Review of Education*. <https://doi.org/10.1007/s11159-024-10123-4>
- OECD. (2015). *Codebook for Derived Variables for PIAAC Public Database (with SAS code)*. Organisation for Economic Co-Operation and Development. <https://www.oecd.org/content/dam/oecd/en/about/programmes/edu/piaac/data-materials/Codebook-for-derived-Variables-16March2015.docx>
- OECD. (2025). *Codebook and SQL Code for Simple Derived Variables (PIAAC Cycle 2)*. Organisation for Economic Co-Operation and Development. <https://www.oecd.org/content/dam/oecd/en/about/programmes/edu/piaac/data-materials/cycle-2/piaac-cy2-codebook-and-sql-codes-for-derived-variables.pdf>
- Posit Team. (2025). *RStudio: Integrated Development Environment for R*. Posit Software, PBC. <http://www.posit.co/>
- R Core Team. (2024). *R: A Language and Environment for Statistical Computing* (R version 4.4.1-- “Race for Your Life”). R Foundation for Statistical Computing. <https://www.R-project.org/>
- ROA & GESIS. (2025). *PIAAC Cycle 2 BQ Draft Conceptual Framework*. Organisation for Economic Co-Operation and Development. [https://www.oecd.org/content/dam/oecd/en/about/programmes/edu/piaac/background-and-questionnaire/cycle-2/PIAAC\\_CY2\(2018\\_11\)BQ\\_Draft\\_Conceptual\\_Framework.pdf](https://www.oecd.org/content/dam/oecd/en/about/programmes/edu/piaac/background-and-questionnaire/cycle-2/PIAAC_CY2(2018_11)BQ_Draft_Conceptual_Framework.pdf)

## E. Appendix

### Prompt 1

My first go at a Shiny app:

Ok I have my final dataframe "df" which I will use to build the shiny app. I want a shiny app with two panels, side-by-side. The one on the left will be much smaller and a user interactive panel. It will have a Boolean search bar. In this search bar, users can write Boolean search terms in English or German. The app will then search through the values in the variables "variable", "question\_text\_en", "question\_text\_de", "responses\_en", "responses\_de", "generic\_label\_en", "trend\_var" and "generic\_label\_de". If any hits are found in any of these variables, the corresponding row should be included in the output. The right panel will be all output and be split into two halves and upper half and a lower half. It will report the results of the search in the upper half, which could be more than one hit. Results will be displayed in rows and the top row will be selected by default, but the user themselves can click on different rows. The rows will consist of the values for "variable", "generic\_label\_en" (which will be replaced by "question\_text\_en" if "generic\_label\_en" is NA, and will be replaced by "question\_text\_de" in case both generic\_label\_en and question\_text\_en are NA) and "cycle". The lower half of the right panel will be a report of the selected row (top row by default, or user clicked row). This report will consist of "Variable:" which will output the value for df\$variable, "Cycle:" which will output df\$cycle, "Trend status:" which will output df\$trend, "Question text EN:" which will output df\$question\_text\_en, "Question text DE:" which will output df\$question\_text\_de, "Responses:" which will output df\$responses\_en (but if this is NA then it will output df\$responses\_de), "Related variables:" which will combine all values from the following separated by a comma df\$trend\_var, df\$ref\_variables and df\$constructed\_vars.

ChatGPT 3.5o results:

```
library(shiny)
library(dplyr)
library(DT)
library(stringr)

# Load Data
df <- readRDS("piaac_final_df.RDS") # Ensure this is the correct file name

# Define UI
ui <- fluidPage(
  titlePanel("PIAAC Questionnaire Search"),
  sidebarLayout(
    sidebarPanel(
      width = 3, # Smaller panel
      textInput("search_term", "Boolean Search (English/German)", value = ""),
      actionButton("search_btn", "Search")
    ),
    mainPanel(
      width = 9,
      fluidRow(
        column(12, DTOutput("search_results"))
      ),
      fluidRow(
        column(12, verbatimTextOutput("selected_row"))
      )
    )
  )
)
```

```

# Define Server
server <- function(input, output, session) {

  # Reactive: Perform Boolean Search
  filtered_data <- reactive({
    req(input$search_btn) # Only search when button is clicked
    search_term <- input$search_term

    if (search_term == "") {
      return(df) # Show all if empty
    }

    # Filter: Search across multiple columns
    df %>%
      filter(
        str_detect(variable, regex(search_term, ignore_case = TRUE)) |
        str_detect(question_text_en, regex(search_term, ignore_case = TRUE)) |
        str_detect(question_text_de, regex(search_term, ignore_case = TRUE)) |
        str_detect(responses_en, regex(search_term, ignore_case = TRUE)) |
        str_detect(responses_de, regex(search_term, ignore_case = TRUE)) |
        str_detect(generic_label_en, regex(search_term, ignore_case = TRUE)) |
        str_detect(generic_label_de, regex(search_term, ignore_case = TRUE)) |
        str_detect(trend_var, regex(search_term, ignore_case = TRUE))
      )
    })

  # Output: Display search results in a table
  output$search_results <- renderDT({
    df_filtered <- filtered_data()

    df_filtered %>%
      mutate(
        display_label = coalesce(generic_label_en, question_text_en, question_text_de)
      ) %>%
      select(variable, display_label, cycle) %>%
      datatable(selection = "single", options = list(pageLength = 5))
  })

  # Reactive: Get selected row data
  selected_row_data <- reactive({
    req(input$search_results_rows_selected)
    selected_index <- input$search_results_rows_selected
    df_filtered <- filtered_data()

    df_filtered[selected_index, ]
  })

  # Output: Display details of selected row
  output$selected_row <- renderPrint({
    row <- selected_row_data()

    if (nrow(row) == 0) {
      return("No row selected.")
    }

    cat("***Variable:**", row$variable, "\n")
    cat("***Cycle:**", row$cycle, "\n")
    cat("***Trend status:**", row$trend, "\n")
    cat("***Question text EN:**", row$question_text_en, "\n")
    cat("***Question text DE:**", row$question_text_de, "\n")
    cat("***Responses:**", coalesce(row$responses_en, row$responses_de), "\n")
  })
}

```

```
# Combine related variables
related_vars <- paste(na.omit(c(row$trend_var, row$ref_variables, row$constructed_vars)), collapse = ", ")
cat("***Related variables:***", related_vars, "\n")
})
}

# Run the application
shinyApp(ui = ui, server = server)
```