**Instruction Manual**

**(Version 3.0)**

**NIH Database Pipeline Application**

**NIH-DPA v0.4.0-beta**

**Funded by NIMH R01MH116156**

**Principal Investigator: Jessica L. Nielson, PhD**

**Developed by: Thomas Kirsh, BS, MS**

Table of Contents

[Downloading/Installing the Pipeline 3](#_Toc68775695)

[GitHub Repository 3](#_Toc68775696)

[**Download Instructions** 3](#_Toc68775697)

[Using the Pipeline 3](#_Toc68775698)

[**MacOS** 3](#_Toc68775699)

[**Windows/PC** 4](#_Toc68775700)

[Fake Datasets for Practice 4](#_Toc68775701)

[Features of the Application 8](#_Toc68775702)

[**Main Page of the Application** 8](#_Toc68775703)

[**Collect NDA Data Dictionaries** 9](#_Toc68775704)

[**Collect FITBIR Data Dictionaries** 10](#_Toc68775705)

[**Preprocess NDA** 10](#_Toc68775706)

[**Preprocess FITBIR** 11](#_Toc68775707)

[**Plotting the Datasets (Visualize Dataset)** 12](#_Toc68775708)

[**Merge & Transform** 16](#_Toc68775709)

[**Merge** 17](#_Toc68775710)

[**Transform** 18](#_Toc68775711)

[**Get Stats** 20](#_Toc68775712)

[**Metrics** 20](#_Toc68775713)

[Upcoming Features 21](#_Toc68775714)

[Known Bugs 22](#_Toc68775715)

[Other Notes: 22](#_Toc68775716)

# **Downloading/Installing the Pipeline**

## **GitHub Repository**

### **Download Instructions**

* Location: https://github.com/Nielson-Lab/NIH-database-pipeline
* Click on "Releases" (underneath the "About" section) on the right side of the home page
* Find the most recent MacOS and Windows versions of the application, then click on "Assets" underneath those headers
* Only download the application ZIP file that is compatible with your system

# **Using the Pipeline**

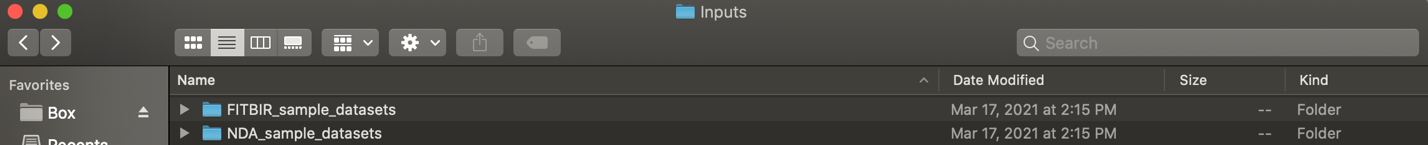
### **MacOS**

Graphical user interface

Description automatically generated

**Figure 1A.** The main folder for the application. This folder gets downloaded from Github. Please note that although this screenshot is for the Mac version of the application, the Windows version has the same files.

The downloaded file should contain folders for *Inputs* and *Outputs,* as well as the application (NFP) and an *Instructions.md* file*.* The user **should not** move the application out of the *MacOS\_app* folder.



A picture containing background pattern

Description automatically generated

**Figure 2A.** The *Inputs* folder in the application. Datasets that will be used in the application are uploaded here. The application only reads “.csv” or “.txt” files, so files stored in folders will not be read. Those files will need to be moved from the folder.

### **Windows/PC**

**Graphical user interface, application

Description automatically generated**

**Figure 1B.** The main folder for the application. This folder gets downloaded from Github. Please note that although this screenshot is for the Windows version of the application, the Mac version has the same files.

The downloaded file should contain folders for *Inputs* and *Outputs,* as well as the application (NFP) and an *Instructions.md* file*.* The user **should not** move the application out of the *Windows\_app* folder.

Table

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

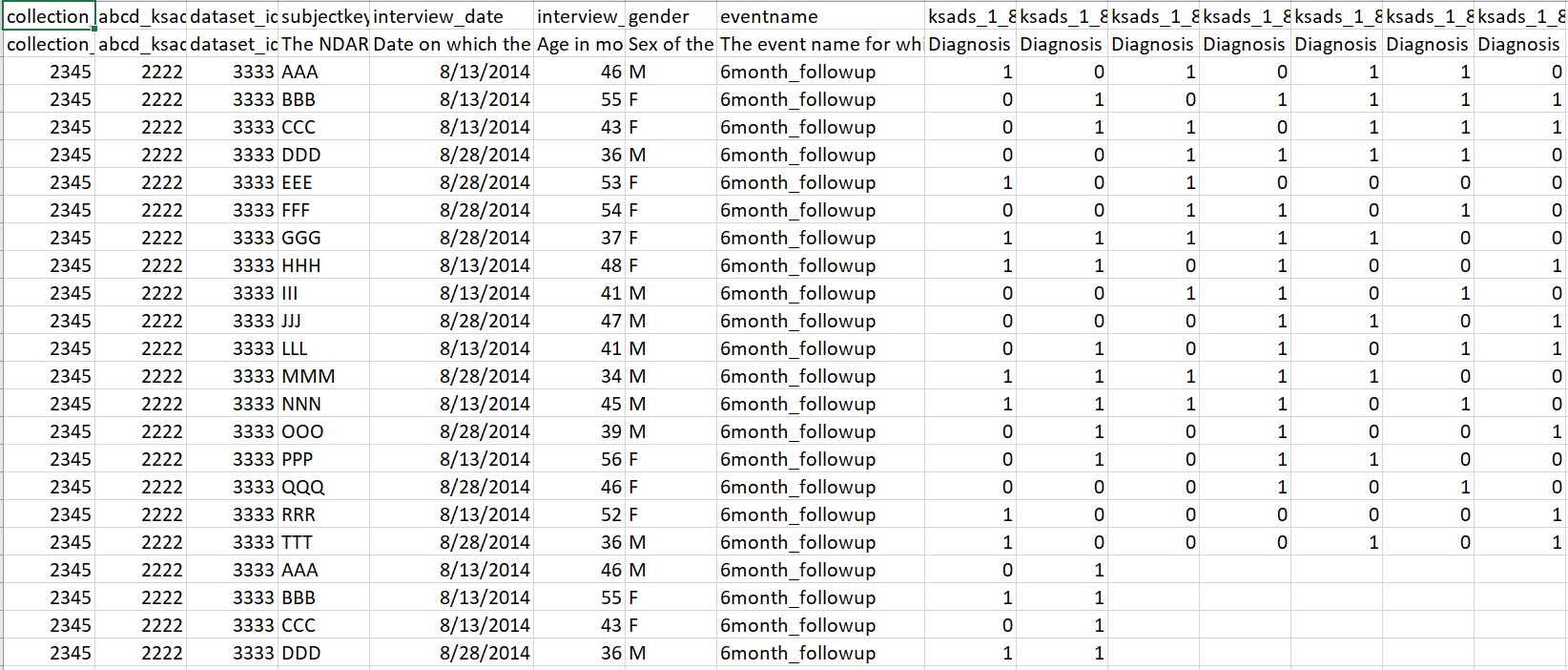
**Figure 2B.** The *Inputs* folder in the application. Datasets that will be used in the application are uploaded here. The application only reads “.csv” or “.txt” files, so files stored in folders will not be read. Those files will need to be moved from the folder.

The application will automatically look for the files the user wants to work on in the *Inputs* folder. Therefore, the user should put the files they want to work with or on in the *Inputs* folder. The files **must**  either be “.csv” or “.txt” files; “.xlsx” files are not supported at this time. Those files can easily be converted to either of the accepted formats. Having separate folders in the *Inputs* folder is fine because the application will ignore them when looking for files.

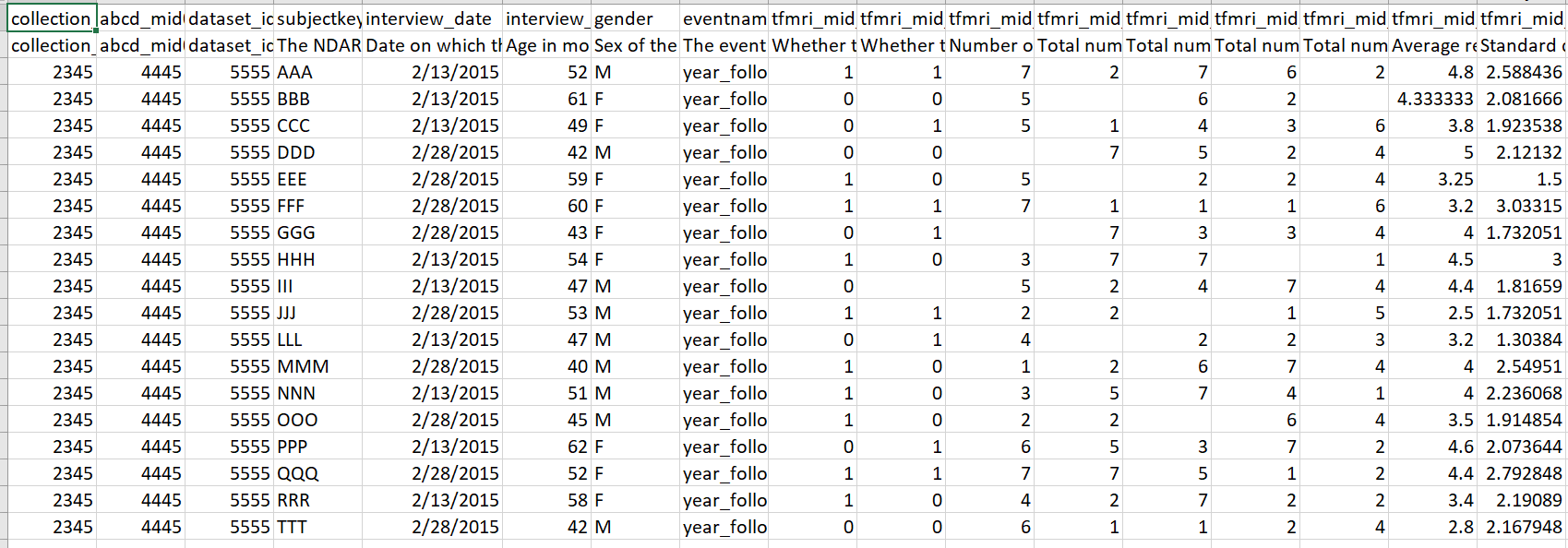
## **Fake Datasets for Practice**

The application comes with some sample FITBIR and NDA datasets. These datasets were generated by us to reflect the unique aspects about the data files from each database. All the continuous variables were randomly generated using either the RAND() function or the RANDBETWEEN() function in Excel. Categorical and date variables were assigned random variables that reflect values in a similar format to the format used by the databases. Certain time intervals were modeled after the “3 months” and “6 months” values found inside a selected dataset in FITBIR; however, the data in the sample files do not match any person inside the original dataset.

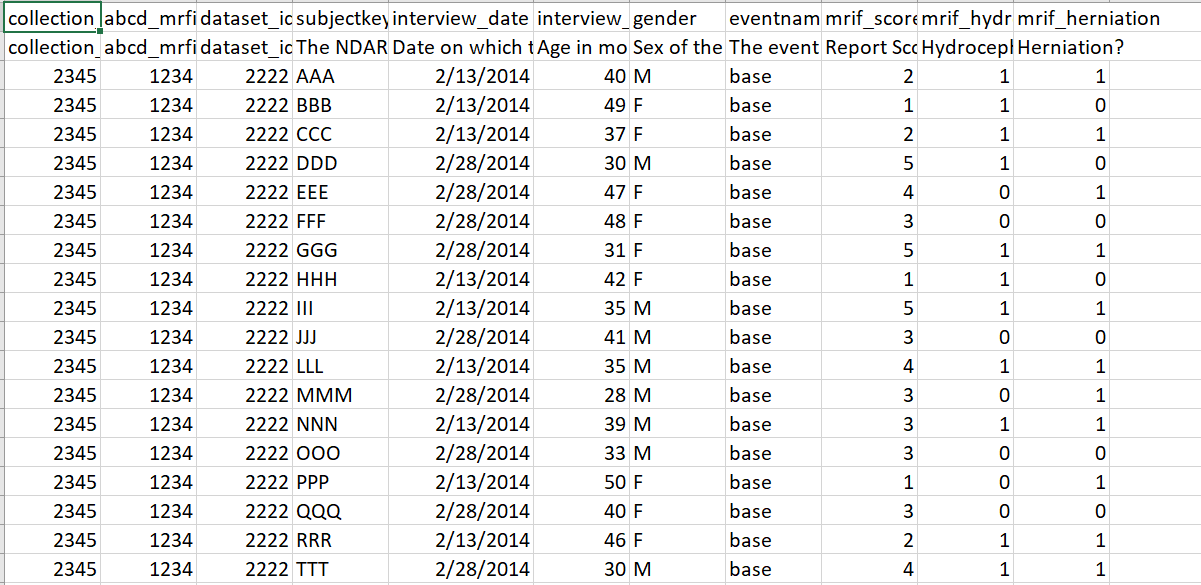
Legitimate names of forms in NDA, and data elements in both NDA and FITBIR are used and necessary for the user to be able to play with the data dictionary scraping features.



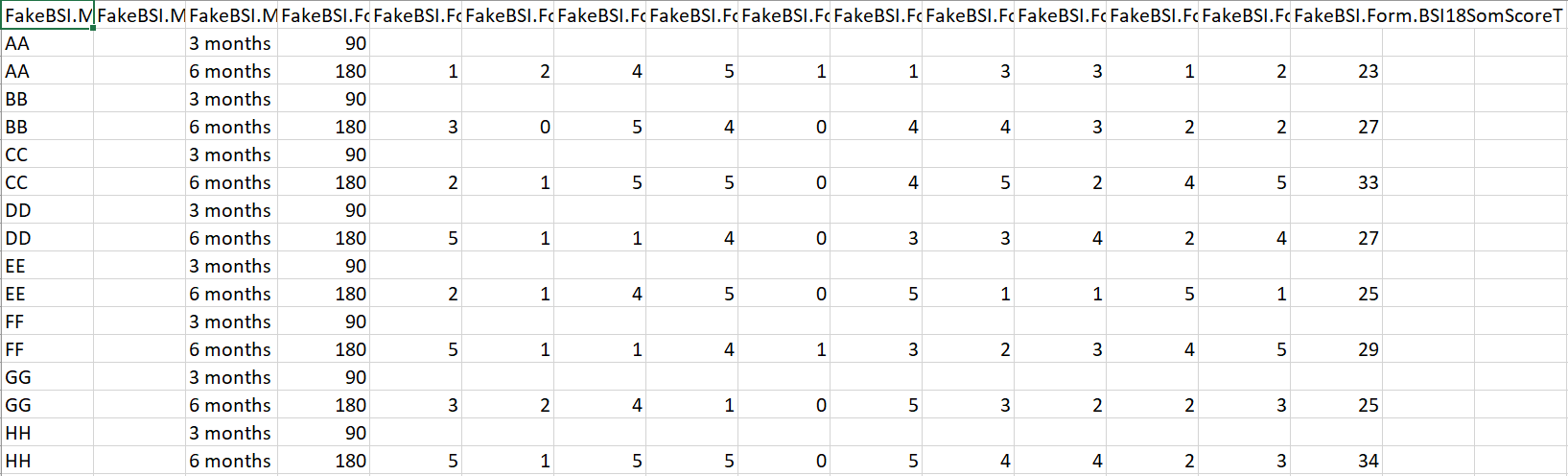
**Figure 3.** Sample NDA dataset called “abcd\_ksads01.txt” in order for users to be able to test the web scraping functionality. Please see the above description for how the datasets were generated.



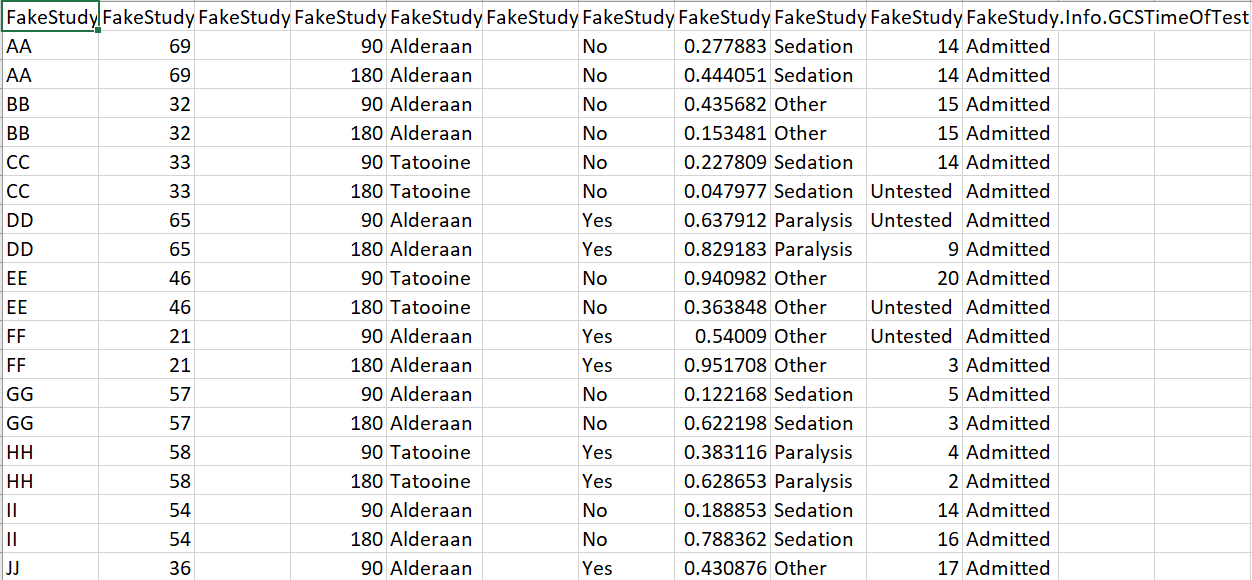
**Figure 4.** Sample NDA dataset called “abcd\_mid02.txt” to match the name of a data dictionary users could scrape. Please see the above description for how the datasets were generated.



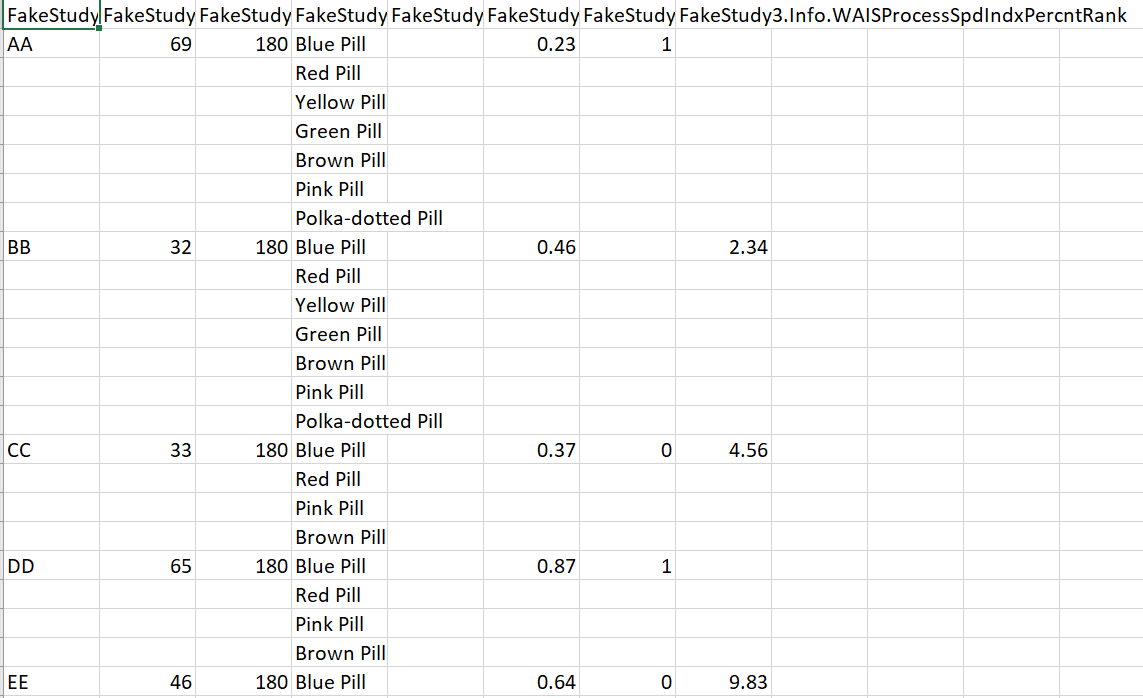
**Figure 5.** Sample dataset titled “abcd\_mrifindings01.txt”. Please see the above description for how the dataset was generated.



**Figure 6.** Sample dataset from FITBIR. The column names contain parts separated by periods. The “time” column here contains strings as time points.



**Figure 7.** A second FITBIR sample dataset.



**Figure 8.** An example of a fake FITBIR dataset with a similar format to unflattened CSV files. The last part of the column names match an actual data element in FITBIR so the user can practice scraping data dictionaries from FITBIR.

All files that the application processes and returns will be saved to the *Outputs* folder.

## **Features of the Application**

All files downloaded from NDA and FITBIR contain the name of the form to which they correspond. The Data Dictionary collection pages of this application rely on the downloaded dataset names to know which dictionaries to collect! Do not change the names until after you are done collecting the relevant data dictionaries.

### **Main Page of the Application**

**A screenshot of a red background

Description automatically generated**

**Figure 9.** **The main page for the web application**. Each button takes the user to a different function in the application. The buttons are displayed in the order we recommend users to use the options.

The main page displays all the options the users have for working with their data. Each option describes its function. The top-down order of the options should be similar to the order the user would expect to work with their data.

### **Collect NDA Data Dictionaries**

A screenshot of a yellow box

Description automatically generated

**Figure 10. The page for the collecting NDA data dictionaries for all files in the *Inputs* folder.** There are options to collect only the data dictionaries that correspond to the files in the *Inputs* folder and an option to collect all the data dictionaries in NDA. The second option takes more time and is useful if the user is studying the whole of NDA.

The underlying script here takes advantage of the Python API that NDA provides. Using that API, we scrape the data dictionaries, which are publicly available, that match the names of the data text files. These text files are the forms used in a study (e.g. “Beck Symptoms Inventory”). The variables are the individual *data elements* in each form (e.g. “feels sad”). From there, the script combines all the data dictionaries into one big data dictionary, with information about the variable name, its description, type, and possible values. The possible values are the values that the standard data element can have, as agreed upon by NDA, but do not necessarily reflect the actual values in your dataset. For example, the data dictionary could say that the values [1,2,3,4,5] are possible, but your dataset only has people who have values [1,2,4].

In the application, we have tried to account for when some datasets are not public. If a data dictionary cannot be found, the application will return an error. Check to make sure your files can be found in NDA’s Data Dictionary search tool.

### 

### **Collect FITBIR Data Dictionaries**

**A yellow box with black text

Description automatically generated**

**Figure 11. The page for the collecting FITBIR data dictionaries for all files in the *Inputs* folder.** There are options to collect only the data dictionaries that correspond to the files in the *Inputs* folder and an option to collect all the data dictionaries in FITBIR. The second option takes more time and is useful if the user is studying the whole of FITBIR.

In principle, the underlying script is the same as the one that collects NDA data dictionaries. The big difference is that FITBIR does not have a Python API that facilitates web scraping. Additionally, FITBIR data dictionaries (like NDA dictionaries) do not display statistics for each study in which a form was used in. This means that the value range given in FITBIR data dictionaries denotes the *possible values* that could be in this data element. A quick note about terminology: *data files* refers to the collection of forms used in a study. *forms* refer to the different assessments used in a study (i.e. Beck’s Depression Inventory, PTSD Checklist, PHQ-9, etc.) while *data elements* refer to the variables measured in each assessment (i.e. “feels sad”, “feels alone”, etc.).

The data dictionaries can be merged with the “Get Stats” output file to create a more complete data dictionary for the user’s specific dataset.

### 

### **Preprocess NDA**

**A screenshot of a computer

Description automatically generated**

**Figure 12.** The page for the user to decide how to process their NDA datasets.

Options to process NDA datafiles are:

1. Remove empty columns
2. Drop spurious columns and other columns
3. Create missing data indicator columns and for which columns.
4. **New:** convert and scale data elements to their FITBIR names and scaling. An internal database is used to map NDA data elements to their FITBIR counterparts. Only works if the data element is found in both NDA and FITBIR.

The default missing data indicator value is an empty string (or an empty cell in Excel). If you want to tell the application that another value indicates missingness, enter that value or a list of values separated by a ‘;’. Examples of other values that indicate missingness are -777, -999, NA, NaN (as a string), etc. **NOTE:** If you add missing value indicator columns prior to merging, there could be new missing values that are unaccounted for. It is recommended to use “Preprocess FITBIR” after merging to add missing value indicator columns.

Because NDA datafiles have the first row of each file as metadata, the script automatically removes them and saves those to separate files in the *Outputs* folder.

### **Preprocess FITBIR**

****

**Figure 13.** The page for the user to decide how they want to process their FITBIR files.

Options to process FITBIR files are:

1. Split the column names by period and which portion of the column name should be retained
2. Dropping specified columns. These are separated by a ‘;’.
3. Remove all empty columns.
4. Create missing value indicator columns for specific variables. **NOTE:** If you add missing value indicator columns prior to merging, there could be new missing values that are unaccounted for. It is recommended to use “Preprocess FITBIR” after merging to add missing value indicator columns.
5. **New:** convert and scale data elements to their NDA names and scaling. An internal database is used to map FITBIR data elements to their NDA counterparts. Only works if the data element is found in both NDA and FITBIR.
6. Fix unflattened files (by either removing the offending columns and saving them to a separate file, or by joining their values into one cell separated by a ‘;’.

Splitting the column names means separating the column names by period, so each name will be split into three parts. The option for how many parts of the name to keep tells the application to keep the last *n* parts. For example, if one part is requested to be kept in the column name “FakeStudy.Info.GUID” , the new column name will be “GUID”. If two parts are requested, the new column name will be “Info.GUID”. If zero or more than 3 parts are requested, the new column name will be “FakeStudy.Info.GUID”.

When downloading the datasets from FITBIR, users have the option to download the files as “flattened” or “unflattened”. We **strongly** recommend you always download the files as “flattened”. This option creates binary variables for each value in the list columns. It is a much cleaner method for working with the data. An example of an unflattened file can be found in Figure 6.

If, however, you are given an unflattened file, you can handle it in two ways:

1. Remove them and copy the GUID column and store them in a separate file.

2. Merge all the cells by group and combine the values in the list column into one string separating values by a ';'.

The "group columns by" columns should be columns that don't have more than one value per row (like the list column). A good example is "GUID;AgeYrs;GeneralNotesTxt", etc.

### 

### **Plotting the Datasets (Visualize Dataset)**

**Graphical user interface, application, table

Description automatically generated**

**Figure 14. First page displaying the datasets in tabular form.** The application provides 2 default datasets for the user to familiarize themselves with the capabilities of the visualization. The visualization page of the application no longer requires that your dataset be located in the *Inputs* folder. Instead, you can click on the “Select File” or drag and drop the file you want to look at into the dashed box. The row underneath the column names is for filtering the data. Text, numbers, and dates can be filtered. For example, you can only look at data from women by typing “Female” underneath “sex”. Using the command “= Female” (without quotes) also works. The command “contains” searches for all text values that contain the specified substring (e.g. “contains Fem”). All columns can be filtered using “=, >, <, != (not equal), >=, <=”. Datetime columns take the format of YYYY-MM-DD (e.g. “< 2020-01” or “= 2020-01-01”). Filtering the data table does not change the dataset that is used for plots.

Graphical user interface

Description automatically generated

**Figure 15.** Example of a histogram for a nominal/categorical variable, plotting the count of the days that people tipped on, colored by “sex”. The application can produce vertical (shown here) or horizontal histograms/barplots. The counts are stacked by color (Mode can be stacked, overlay (default), or group) and not normalized (Bar Normalization can be none, fraction or percent).

Chart

Description automatically generated with medium confidence

**Figure 16.** Example of a histogram for a continuous variable. Here the “total\_bill” distribution is plotted and grouped by “sex”. Instead of plotting the count for each bin, which are automatically calculated, the percent is chosen. A marginal subplot (also called a “rug plot”) is plotted above, using marks to show the distribution of the data. The user also has the option to rename the axes labels and the title of the plot by clicking on the current text.

The histograms will only plot non-missing data. If your data has a column that has all its values missing, an error will be thrown.

Graphical user interface

Description automatically generated

**Figure 17.** Example of a lineplot using the “gapminder” dataset (see the Data Table example datasets). Plots two continuous variables to search for trends.

Graphical user interface, chart, scatter chart

Description automatically generated

**Figure 18.** Example of a scatter plot. Markers can be colored by the hue column, vary in sizes specified in the dataset, and stylized.

Graphical user interface

Description automatically generated

**Figure 19.** Example of a heatmap. Best for plotting categorical variables but numerical variables can be used as well if there aren’t too many. Multiple coloring options are included and the option to center the color scale at a specific value is called “Centering Value”.

Chart, box and whisker chart

Description automatically generated

**Figure 20.** Example of the boxplot feature for visualizing data. These can be oriented vertically (shown) or horizontally. The columns will need to be adjusted appropriately. There is also the option to exclude outliers, show outliers, or show all data as points in addition to the boxplot.

Graphical user interface, application, Word

Description automatically generated

**Figure 21.** Example of the violin plot feature for visualizing data. These can be oriented vertically (shown) or horizontally. The columns will need to be adjusted appropriately. There is also the option to exclude outliers, show outliers, or show all data as points in addition to a boxplot.

### **Merge & Transform**

**A screenshot of a computer

Description automatically generated**

**Figure 22.** The page that prompts the user for the various inputs necessary to merge and convert files from longitudinal to wide format.

The merge and transform page merges all the files in the *Inputs* folder and then converts the resultant dataset from longitudinal to wide format. This is a streamlined process for users who know that they want to merge and transform all their data files and how they want to do it. However, each time the files are merged first and then transformed. If the user wants to transform first and then merge, they can run the ‘Transform’ option first from the main page and then run the ‘Merge’ option. See the ‘Merge’ and ‘Transform’ options for more details.

### **Merge**

A screenshot of a computer screen

Description automatically generated

**Figure 23.** The page prompts the user for the inputs necessary to merge files.

The merge option runs the same merging script that “merge and transform” uses, the only difference is that this script will output the file merged file. The files are automatically read into the script from the *Inputs* folder; the user does not need to select the files in the application.

The ‘time’ column does not have to be a ‘Time’ column (likewise the ‘GUID’ column doesn’t need to be a ‘GUID’ columns), it can be all column that you want to use to direct the merge.

If your time column has empty strings either: remove the rows with empty strings, find another column to use as your time column, or figure out what dates those empty strings indicate, or just merge on the GUID column and leave the time column option to the default value. The default value tells the application to just merge on the first column.

### **Transform**

A screenshot of a computer

Description automatically generated

**Figure 24.** The page that prompts the user for the necessary inputs to convert a longitudinal dataset to a wide dataset (only for NDA and FITBIR. NIDA is another beast).

The conversion from longitudinal to wide format requires that a column that contains the time points will be used. In NDA and FITBIR (and NIDA), there are three types of ‘time’ columns: dates (like 4/3/2017), strings (‘3 months’, ‘6 months’), or numbers (30, 90, 180). The application can handle all of these. For the dates, the application converts to days from the earliest time point by default. If you have a time column with specific dates (e.g. ‘03/14/15’), sort this column from earliest to latest.

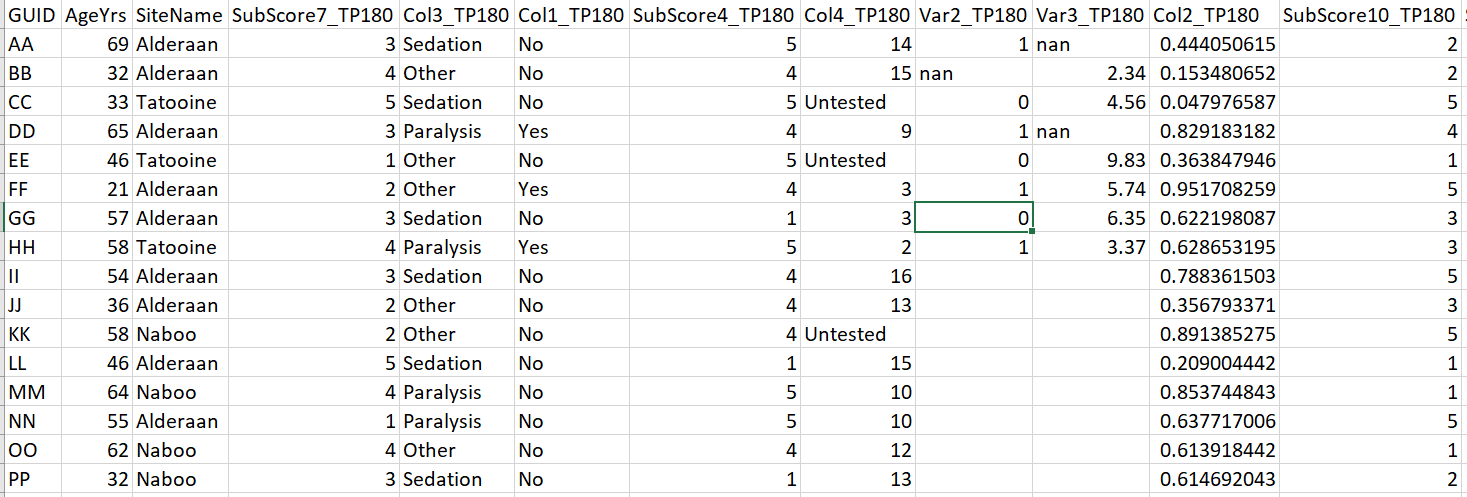
If you want the output to be in months or year or weeks, you can enter a number for the interval between time points in days (i.e. 30 days for intervals of a month), and the application will divide the raw days by the interval to convert times. To illustrate, imagine your starting date is 3/14/15 and the second time point is 6/14/15. The time interval is 3 months. If you enter the time interval as 30 (for 30 days) and a prefix of “M” (for month), the application will add “M3” to the column names. If you instead enter 90, the application will add “M1” to the column names at the second time point, because these times are the “M1” times, times before them are “M0”.

For columns with strings and/or numbers, the application can use those values as the new times. Note that this requires the user to know these things about their data prior to converting.

If your time column has empty strings either: remove the rows with empty strings, find another column to use as your time column, or figure out what dates those empty strings indicate. The application will not transform the way you expect if your column has empty strings.

Because there could be multiple measurements made at the same time point (for whatever reason), the application can aggregate over these measurements in various standard ways: using the mean, median, mode, first value, last value, or no aggregation (use only if you have one measurement per time point). The aggregation will be used to return one value per GUID.

Finally, the user can enter a prefix to denote the different time points. The default prefix is “TP”, so “TP” plus the number indicating the time will be added on to the end of each column name.



**Figure 25.** Example output after using the transform options in the previous figure and the merged dataset from the ‘merge’ example. The example prefix is the default “TP”. The columns “AgeYrs” and “SiteName” were excluded from the transformation because those variables do not change over time.

### **Get Stats**

A screenshot of a yellow box

Description automatically generated

**Figure 26.** The page that prompts the user to decide what they want to call the file that collects various statistics for a single dataset in the *Inputs* folder.

The statistics that are calculated per column are:

* % missing
* # unique values
* Mean
* Median
* Min
* Max
* Mode
* Variance
* Standard Deviation
* 5th percentile
* 95th percentile
* Skewness
* Kurtosis
* Value Range (for up to 10 unique values)

### **Metrics**

This section measures the time it takes the application to perform the merging and transforming functions independently, so that users will have a rough idea how long they can expect their tasks to last. A range of dataset file sizes were chosen to show how these functions perform for small and large files. As expected, larger datasets and more datasets took longer to merge and transform. The datasets used in these tests were taken from the sample FITBIR datasets provided with the application (see Figures 2A and 2B), and sample datasets from NDA. These datasets were chosen from the files we had downloaded for our own analysis and they are specified for users to compare results with (specifically for the sample datasets).

* Sample FITBIR datasets: provided with the application
* Sample NDA datasets: provided with the application
* ABCD datasets (not provided with the application, but accessible to qualified users in NDA):
  + The forms “fmriresults01”, “mri\_rsi\_p102”, and “mri\_rsi\_p202” from the ABCD study in NDA
* Suicidality datasets (not provided with the application, but accessible to qualified users in NDA):
  + The forms “abcd\_ksad01”, “abcd\_ksad501”, “cssrs01”, “ctq01”, “fmriresults01”, “freesqc01”, “freesvol01”, “freesvol201”, “freesvol301”, “hrsd01” and “ndar\_subject01” from the ABCD study in NDA

The datasets used to test the transform function were the datasets that resulted from the merging function.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Source** | **Method** | **No. of Files** | **File Sizes** | **Time** |
| Sample FITBIR datasets | Merge | 3 | 3 KB, 1 KB, 1 KB | < 1 second |
| Transform | 1 | 6 KB | < 1 second |
| Sample NDA datasets | Merge | 3 | 2 KB, 5 KB, 8 KB | < 1 second |
| Transform | 1 | 15 KB | <1 second |
| ABCD datasets | Merge | 3 | 90 MB, 40 MB, 40 MB | 51 minutes |
| Suicidality datasets | Merge | 11 | 113 KB - 34 MB | 19 minutes |
| Transform | 1 | 267 MB | 13 minutes (output file size 63 KB) |

**Table 1:** Metrics table showing the time it took to merge and transform certain selected files. This table can be used as a reference for users to estimate how long their operations will take.

**Walkthrough Example**

In this example, we’ll walkthrough how to merge datasets from NDA and FITBIR together. We’ll use NDA’s data element scales and names. Start by moving the data files from “Input > NDA\_sample\_datasets” to the main Inputs folder.

A screenshot of a computer

Description automatically generated

Collect the data dictionaries.

A screenshot of a yellow box

Description automatically generated

A screenshot of a spreadsheet

Description automatically generated

To preprocess the NDA files, we notice that there are other ID columns for each form. We’ll drop those

since they add no information to our analysis. They are

* collection\_id
* abcd\_mrfindings01\_id
* dataset\_id
* abcd\_ksad01\_id
* abcd\_mid02\_id

Since we want to keep NDA data element structure, we won’t rename to match FITBIR names and scales. At this point, we won’t add missing value indicator variables since merging will likely introduce more.

A screenshot of a computer screen

Description automatically generated

# **A table of numbers with black text Description automatically generated with medium confidence**

Processed\_abcd\_ksad01 data file.

Next, we’ll process the FITBIR datasets. Sample datasets 1 and 2 are flattened, while sample dataset 3 is unflattened. We cannot process flattened and unflattened datasets together, so we’ll start with the flattened set. Move “sample\_fitbir\_1.csv” and “sample\_fitbir\_2.csv” to Inputs and then switch to “Process FITBIR”. To match NDA, we’ll need to split all columns and keep the last part. There are no specific columns we want to drop but we want to remove any empty columns. Since our goal is to match the NDA datasets, we’ll scale and match the NDA column names. Like NDA, we won’t add missing value indicator columns at this stage.

A screenshot of a yellow box

Description automatically generated

A white calendar with numbers

Description automatically generated

You can see how GUID was changed to “subjectkey” and the corresponding BSI data elements have been modified.

Next, we’ll work on the unflattened dataset. Move “sample\_fitbir\_3.csv” to Inputs and then switch to “Process FITBIR”. We’ll keep everything the same as the unflattened datasets, but we’ll want to us unflattened CSV files, save them separately, and specify columns to group by. IMPORTANT: the order of operations is the same as the order on the screen. That means the names will be modified to NDA names prior to grouping the flattened columns. So we have to use the NDA names after that moment.

A screenshot of a computer

Description automatically generated

A screenshot of a table

Description automatically generated

Running the preprocessing produced no errors but dropped AgeYrs. What’s the issue? It’s because we input “AgeYrs” but the NDA data element is “interview\_age”. A complete list of data element name equivalency is in the “NDA\_FITBIR\_alignment.csv” file. Using the correct data element name yields

A screenshot of a computer

Description automatically generated

A screenshot of a data

Description automatically generated

Looking through the datasets, NDA and FITBIR use separate methods to mark time points. NDA uses raw dates while FITBIR uses days since baseline. Therefore, if we tried to merge and transform these together, we’d have two different time point formats and the application cannot handle that. Instead, we’ll merge and transform them separately before merging them together. Let’s start by putting all the processed FITBIR files into the Inputs folder.

To merge the FITBIR datasets, we’re going to use the “subjectkey” and “DaysSinceBaseline” columns as keys to merge on. SiteName doesn’t change with time so we’ll exclude it from the conversion. If there are more than one value at a time point and GUID, we’ll take the mean of them. Since there are two time points (90 days and 180 days), we’ll just keep them in the new column names and do not need to calculate a time interval.

A screenshot of a computer

Description automatically generated

# **A screenshot of a computer Description automatically generated**

Now we’ll merge the NDA datasets. Replace the processed FITBIR datasets with the processed NDA datasets. Then open the “Merge” tab. In NDA, we’ll merge on the “subjectkey” and the “interview\_date”.

A screenshot of a yellow box

Description automatically generated

A screenshot of a spreadsheet

Description automatically generated

You’ll notice that after the merging and transforming a statistics file is automatically generated of the output dataset. That can be helpful to understand what’s in your datasets.

Take the “nda\_merged\_file.csv” from the Outputs folder and move it to Inputs. Move the “processed” datasets out of Inputs so that only “nda\_merged\_file.csv” is in there. Let’s see what the distribution of the dates look like in NDA. Go to “Home” and “Visualize Dataset”. Select the “nda\_merged\_file.csv” file to load into the application. Then go to the “Histogram” tab and Plot “interview\_date”

A screenshot of a graph

Description automatically generated

From the histogram, it looks like data is taken every 6 months (180 days) from the participants and maybe they started on different days. We’ll use an interval of 180 days to transform on. Go back to the main page and select “Transform”.

Since “gender” doesn’t change over time, we’ll exclude it from the time conversion. We don’t want to use the values in the column in the new column names, we want to calculate them from a list of dates. If the second measurement is a redo, we’ll use that (choose “last”). To match what we did for FITBIR, we’ll keep the “Day” suffix.

A yellow screen with black text

Description automatically generated

A table with numbers and lines

Description automatically generated

Now we’re ready to combine the NDA and FITBIR files! Take both the “fitbir\_merged\_and\_transformed.csv” and “nda\_transformed.csv” files and put only them in the Inputs folder. Remove any other files. Since we removed the time column, we only need to merge on the “subjectkey” column.

If you want to add missing value indicator columns, now is the time to go back to the “Preprocess FITBIR” tab and add them.

A screenshot of a yellow box

Description automatically generated

A screenshot of a spreadsheet

Description automatically generated

A stats file is generated at the end but you can also use the “Get Stats” tab to generate one for any file you put into Inputs.

If at any time you run into an error, you can find the log file using the directions on the Github README. If you need help, please open an Issue in the same Github repository after searching to see if anyone has the same Issue as you.

To close the application, close the tab and stop the process by going to “Activity Monitor” on Mac or “Task manager” on Windows, finding “NFP” and stopping the process.

# **Upcoming Features**

* Create a new Issue on the Github repository to request features

# **Known Bugs**

* Data Table can’t read comma-separated .txt files
* The MacOS version needs to be at least Monterey 12.7.4 to use the application
* PC version runs on Windows 10

# **Other Notes:**

This application is not intended for use with imaging data.

You can merge using 1-2 columns as identifiers at a time. While the purpose of this application is to facilitate data mining and processing data from NDA, FITBIR, and NIDA, you can use datasets from other sources and use this application on them in the same way. For other datasets (or even NDA, FITBIR, or NIDA), your “GUID” column doesn’t *have* to be an actual GUID column, it can be whatever column you want to merge on. Same with the “time” column for merging. For transforming, your “time” column does need to actually be a “time” column and match formatting with NDA or FITBIR.

If you use a dataset that is not from NDA, FITBIR, or NIDA, that is comma-separated (CSV), enter the information into the application as though it were a FITBIR dataset.