Efficient `prASC.py` Manual

Michael Wilson

(Last updated December 16, 2019)

Contents

1. Introduction

Typically, analyzing data collected using eye-tracking while reading methodologies involves several intermediate steps. If using EyeLink, one must usually convert the EDF files output by the software to ASC files. Then, fixations must be corrected either manually using software like EyeDoctor, or automatically, using a script like `fix_align_v0p92.R` (Cohen 2013). In either case, the resulting output must be further analyzed using software like RoboDoc (if using EyeDoctor's output) or SideEye (if using `fix_align`) to eventually create a CSV containing results that are amenable to statistical analysis. Information about question accuracy must also be processed separately, and added to the results from the sentences.

It is clear that this data analysis pipeline is somewhat fragmented and time-consuming. The following manual explains how to use `prASC`.py,[1] a script written in Python to streamline data analysis of ASCII converted output from EyeLink systems (e.g., '`.asc`' files).

[1] So named because it facilitates **pr**ocessing **ASC** files.

It is in concept a combination of several pre-existing scripts created by Andrew Cohen (`fix_align_v0p92.R`, Cohen 2013) and Brian Dillon, with the goal of going from ASC files to usable CSV files in one fell swoop, sidestepping the intermediate steps in the process outlined above.

When run with the appropriate parameters set, `prASC` takes in raw, uncorrected ASC files and outputs a results CSV that is ready for statistical analysis. This results CSV includes information about reading times on user-specified sentence regions; question accuracy information; and, depending on whether you have an appropriately formatted stimuli file available, condition information about each trial. The author offers this software in hopes that it will help make data analysis of experiments run using EyeLink less time-consuming, allowing for greater productivity.

This manual is for the efficient version of `prASC`, so named because it is designed to only process results which have not yet been processed.

2. System Requirements

`PrASC` requires Python 3.8, along with the packages `sideeye` and `pandas`. You can install Python 3.8 from https://www.python.org/. To install the required packages, should you not already have them installed, open the command prompt (Windows) or Terminal (Mac), type the following, and hit enter:

- `pip install pandas sideeye`

If you have issues with this step, you might need to use the `--user` option when installing. If you already have these packages installed, you'll need to make sure you have their latest versions (at the time of writing this, these are `pandas 0.25.3` and `sideeye 1.0.0a13`, which you can ensure by running `pip install --upgrade pandas sideeye`. (You may also need to use the `--user` option with this command.)

3. Usage

3.1 Overview

`PrASC` is best run through the command prompt (Windows) or Terminal (Mac), since there are options the user can set when running it, though it is possible to double click on it to run it if default settings are desired. To run it through the command prompt/Terminal, navigate to the directory where it is located, type the following,[2] and hit enter:

- `python prASC.py`

After running, `prASC` creates a directory whose name and location you specify in the parameters file, which contains output files based on your settings. The following are possible output files:

- `results.csv`: a CSV containing information about reading times for sentences

[2]For this to work, Python must be in your system path. If this is not the case, then you will need to specify the directory where Python is located.

- `question_summary.txt`: a space-separated text file containing a summary of participants' overall performance on questions
- `subject_question_info.txt`: a space-separated text file containing information about participants' performance on each question
- `results_combined.csv`: a CSV combining the results from the above three files

The default behavior of `prASC` is to only output `results_combined.csv`. You may tell it to keep the intermediate files by using the argument `--keepall` or `-k`. You may tell it to not output the combined results file by using the argument `--nocombine` or `-nc`. If `--nocombine` is set, `--keepall` is automatically set.

In addition, `prASC` will output fix aligned ASCs to a directory whose name and location is specified separately from the output directory in the parameters file.

3.2 Command line arguments

`PrASC` may take the following command line arguments. This table shows the full version of the argument, the short version, and the default value. An explanation of what each argument does follows the table.

Option	Short version	Default
`filename`	`N/A`	`parameters.py`
`--overwrite`	`-o`	`False`
`--keepall`	`-k`	`False`
`--refix`	`-rf`	`False`
`--nofix`	`-nf`	`False`
`--resentences`	`-rs`	`False`
`--nosentences`	`-ns`	`False`
`--verbose`	`-v`	`False`
`--requestions`	`-rq`	`False`
`--noquestions`	`-nq`	`False`
`--nocombine`	`-nc`	`False`
`--reeverything`	`-re`	`False`

Table 1: Command line arguments and default values

3.3 Option explanations

The following is an explanation of each option:

- `filename`: the location of your parameters file. The default assumes a file in the executing directory named `parameters.py`.
- `--overwrite`: Set this to overwrite existing results files. The default is to not overwrite existing results files; if files exist and you haven't specified this option, processing that would result in any existing files will be skipped.
- `--keepall`: Set this to keep all results files generated, including separate files for sentence and question information. If `--nocombine` is set, this is automatically set.

- `--refix`: Set this to rerun `fix_align` on ASC files in your ASC files directory that have existing fix aligned counterparts in the specified fix align output directory. The default will only run `fix_align` on files without existing counterparts in the fix align output directory.
- `--nofix`: Set this to process non-fix aligned ASC files. Note that if this is set, `prASC` will assume that the ASC files directory is the location of the ASC files you want to process for sentence and question information. It will not look in the fix aligned files directory (as it does if this is not set). Note that the default behavior is to only fix align ASC files without existing fix aligned counterparts, so there is little reason to set this unless you want to process sentence and question information from raw ASC files (which is not recommended). A better reason to set this might be if you have deleted the raw ASC files from your hard drive, and retained the fix aligned versions. In this case, you can set the ASC files directory to the location of your fix aligned ASCs, and use this option.
- `--resentences`: Set this to reprocess all sentences from ASC files in the ASC files directory. The default only processes sentences from ASC files not already represented in an existing results file.
- `--nosentences`: Set this to skip processing information about sentence trials using SideEye. The default processes sentence information.
- `--verbose`: Set this to print information about subjects' overall performance on questions to the console as the script executes. The default does not print information to the console.
- `--requestions`: Set this to reprocess all questions from ASC files in the ASC files directory. The default only processes questions from ASC files not already represented in an existing results file.
- `--noquestions`: Set this to skip processing question accuracy and response time information from ASCs. The default processes question information.
- `--nocombine`: Set this to not combine sentence and question information into a single CSV. The default combines results into a single CSV.
- `--reeverything`: Set this to set all of `--refix`, `--resentences`, and `--requestions` at once. Essentially a way to reprocess the whole experiment from ASC files in the ASC files directory.

Some of these arguments may have undesired interactions; for instance, setting `--nosentences` and `--noquestions` will still lead to `prASC` attempting to combine existing results. If existing results are present and have the names `prASC` would assign to them, this will work provided information about the same subjects are present in the sentence and question files. Similar behavior occurs when setting `--nosentences` or `--noquestions` separately without setting `--nocombine`. Use caution with these options to avoid unwanted output. The safest thing to do is to not use the `--nosentences` or `--noquestions` options, but these are designed for the benefit of those who may have good reasons for processing these separately. In order for these to work correctly when `--nocombine` is not set, it is recommended that you not rename output files; if you do, `prASC` will not know where to find

them, and will not be able to combine information with them as a result. Usually, though, `--keepall` is the best option in a scenario where the user wants to keep sentence and question information separate, since separate files for sentence and question information will be retained if that option is used.

3.4 Processing only new results

By default, the efficient version of `prASC` only processes new results. For this to work, results files must have the names `prASC` assign them (‘`results.csv`’ for sentences, ‘`subject_question_info.txt`’ and ‘`question_summary.txt`’ for questions, and ‘`results_combined.csv`’ for the CSV combining sentence and question results), and these files must be in the user-specified output directory. The script first looks for the uncombined results files, and then in the combined results file if no uncombined results file exists.

It uses this information to construct a list of filenames already in the results files, and removes files with those names from the list of files to process for sentences and/or questions. This means that ASC files you have not yet processed should not have the same name as any ASC file in the existing results file(s), in order for them to be properly processed as new results. In addition, `prASC` will ignore directory information when comparing files in the directory that is to be processed with those specified in the results file(s), and only compare file names. Take this into account when setting your directories in the parameters file.

This all assumes that you have not modified your JSON configuration between runs of `prASC`; if you modify configuration settings related to column names in your configuration file between runs, `prASC` may not be able to detect which files have already been processed. The same holds if you have modified the column names or deleted columns in the results files themselves and saved over the original file. If you do either of these things, things may break in unexpected ways, since the config file is the only way for `prASC` to determine how to combine new results with existing results. If you change your config file, it is recommended that you reprocess everything anew by using the ‘`--reeverything`’ (‘`-re`’) option.

If you don’t have any files to process, but just want to generate the separate results files from an existing combined results CSV, just run the script using the ‘`--keepall`’ (‘`-k`’) and ‘`--nocombine`’ (‘`-nc`’) or ‘`--overwrite`’ (‘`-o`’) options.[3]

[3]It is recommended to use the ‘`--nocombine`’ option for this behavior rather than the ‘`–overwrite`’ option because of how `prASC` recreates the results. First, `prASC` recreates the individual results files from the combined results CSV. Then, if `--nocombine` is not set, it will attempt to combine them. However, if `--nocombine` is not set and `--overwrite` is not set, `prASC` will throw an error before doing anything because of the potential of overwriting an existing file. If `--nocombine` is set, `prASC` will not check whether it might overwrite it, since that is not even possible with `--nocombine` set. On the other hand, if `--overwrite` is set, then `prASC` will combine the individual results files (which it generated from the combined results CSV) back into a combined results CSV. This new combined results CSV will be identical in content to the old one, even though it technically overwrites it. Setting `--nocombine` rather than `--overwrite` avoids this extra step of overwriting a file with an identical file.

4. The parameters file

The parameters file is a `.py` file that specifies certain information. By default, `prASC` will look in the current directory for a file named `parameters.py`; you can set this to a different location when running using the command prompt/Terminal by putting a file location after the `python prASC.py` command.

The following list explains what values may be set in the parameters file, and what the default value for omitted parameters are. See also the included example parameters file.

- `asc_files_dir`: the location where your raw ASC files are located. The default assumes they are in a subdirectory of the executing directory named "ASC."
- `fa_output_dir`: the location where you want fix aligned ASC files to go. The default assume they are in a subdirectory of the `asc_files_dir` named "Fix Aligned."
- `script_loc`: optional argument, indicating the location of an EyeTrack script file. If your script file was generated using `scriptR`, you have the option to include a `fix_align start_pts` matrix in the header of your script. `PrASC` can pull the `start_pts` information from a script file that includes it, provided it's included in the way `scriptR` does. If this is not included in your script file, or you do not have such a script file, you must specify the `start_pts` matrix manually (see section (**??**)).
- `fix_align_loc`: the location where you have `fix_align.r` (Cohen 2013) on your computer. `fix_align` can be downloaded here. The default assumes it is in the executing directory, and is named `fix_align_v0p92.r` (as this is the version that is current as of writing this manual). If you are not fix aligning files, this is not required.
- `config_json_loc`: the location of a JSON configuration file to be used with SideEye. You will need this even if you are not processing sentences with SideEye, since `prASC` uses this to determine column names for question files and when combining results. The default assumes a file named `config.json` in the executing directory.
- `sentences_txt_loc`: the location of a `sentences.txt` file, which specifies analysis regions when running SideEye. You do not need to specify this if you are not analyzing sentences with SideEye. The default assumes a file named `sentences.txt` in the executing directory.
- `stimuli_loc`: the location of a stimuli file you want to join with your results files. In order for this to work, your stimuli file must have an `item_id` column and an `item_condition` column, which match the values used in your EyeTrack script. `ScriptR` outputs formatted stimuli files that can be joined with results files in the correct way.
- `file_encoding`: when combining results, `prASC` uses the `pandas` package to read in intermediate files as CSVs. The kind of file encoding these files use must be specified. The default is `latin1`. File encoding can be determined automatically to some extent, but results in a much longer run time, so this is set manually; even if you have issues with it, it should still be faster than attempting to do it automatically.
- `output_dir`: the location where you want your results file(s) to go. The default is a file named `prASCed results` in the executing directory.

If any parameters are set to disallowed values, prASC will prompt you to correct them until they are set permissibly. If prASC prompts you and you wish to use the default value, just push enter without typing anything, and it will use the default value for any parameter that has one.

4.1 Parameters passed to fix_align

PrASC executes fix_align when it is run if the user is fix aligning ASC files. The following values are also set in the parameters file, and are passed to the function call to fix_align. Here, I just provide the names and default values for the parameters; more information can be found in Cohen (2013). Note that there is one small difference between the default settings here and those in Cohen (2013): when running prASC, the default setting of trial_plots is FALSE.

Note that these values are set as **strings**; i.e., rather than setting keep_y_var = True, you should set keep_y_var = ``TRUE''.

Setting name	Default value
start_pts	(none; see next section)
xy_bounds	NULL
keep_y_var	FALSE
use_run_rule	TRUE
trial_plots	FALSE
save_trial_plots	FALSE
summary_file	TRUE
show_image	FALSE
start_flag	TRIALID
den_sd_cutoff	Inf
den_ratio_cutoff	1
k_bounds	c(-.1, .1)
o_bounds	c(-50, 50)
s_bounds	c(1, 20)

Table 2: fix_align settings and default values

If any settings are set to impermissible values, prASC will prompt you until you enter a permissible value. If you wish to use the default setting when prASC prompts you, just hit enter without typing anything, and it will use the default value for any parameter that has one.

4.2 Determining values for the start_pts *matrix*

Fix align corrects alignment by fitting a linear regression to each trial's data. In order to determine the starting x and y position of the fitted line, the start_pts matrix is used. This is a $2 \times n$ matrix, where n is the number of separate lines sentences are displayed on in your experiment. The format of this is an expression that looks like the following:

- rbind(c(x, y)[, c(x, y), ...])

where x and y are the starting x and y positions of your lines. If you have more than one line, you need to provide separate x and y positions for each. These values can be determined by opening your experiment in (Mini) EyeTrack, clicking on a sentence trial, clicking "Change", and clicking near the beginning of your sentence.

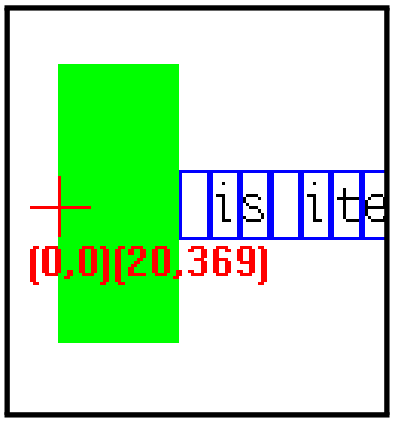

Figure 1: How to manually determine values for the `start_pts` matrix

In this case, you would set the value of `start_pts` to "`rbind(c(20, 369))`".

If you are using `scriptR` to set up your EyeTrack script, you will have the option of generate a `start_pts` matrix when generating your script. You can use this `start_pts` matrix either by manually copying and pasting it from the header of your script file into the parameters file in the appropriate place, or by directing `prASC` to look in your script file for it by setting `script_loc` to its location.

5. SideEye Required Files

5.1 JSON Configuration File

`PrASC` requires a JSON configuration file, which it uses in the call to SideEye when processing sentences. Details about the format of the configuration file can be found at https://sideeye.readthedocs.io/en/latest/configFile.html.

5.2 `sentences.txt` *file*

`PrASC` requires a `sentences.txt` region file, which it passes to SideEye when processing sentences. Details about the format of the configuration file can be found at https://sideeye.readthedocs.io/en/latest/start.html#text-region-file-formats. In brief, a `sentences.txt` file is a space separated file, where each row is a condition of a single item. The first column is the item number as in the EyeTrack script, the second column is the item condition number as in the EyeTrack script, and the final column is the sentence. The final column is used to specify regions, which are designated by forward slashes, "/". Anything between one forward slash and the next defines an analysis region.

6. Contact

If you have any questions or suggestions regarding `prASC` you may contact the author at mawilson@linguist.umass.edu.

References

Cohen, A. (2013). Software for the automatic correction of recorded eye fixation locations in reading experiments. *Behavior Research Methods*, *45*(3), 679–683.