Pupillometric measures of cognitive load

Christopher Daniel Hyek
Computational Linguistics
Universität des Saarlandes
chhy00001@uni-saarland.de

Abstract

We present an implementation of Duchowski's pupillometric measurements known respectively as Index of Pupillary Activity (IPA) and Low/High Index of Pupillary Activity (LHIPA) which were previously only available through pseudo-code in Python 2.7 via updating and packaging them for future usage on the Github. Then we will then take these new implementations and apply them to the Mixed Reality (MR) datasets from Lindlbauer which previously used the original IPA implementation only to see if we can replicate it. Finally we also set up a proposal for the Real-Time Index of Pupillary Activity (RIPA) by Jayawardena and offer additional changes to the previous two pupillometric measurements that we present code for.

1 Introduction

Measuring ones cognitive load via their eyes has been a point of interest in Human Computer Interaction (HCI) research areas (Oviatt, 2006), eye tracking (Marshall, 2002), and even the military (Boehm-Davis et al., 2003) for quite some time and a lot of what we know on it started with Cognitive Load Theory (Sweller, 1988).

While eye tracking research is now finally hitting a point in feasibility via somewhat more portable eye tracking tools, we have the opportunity to implement two pupillometric measurements that are meant to reinvent Marshall's Index of Cognitive Activity (ICA) (Marshall, 2002). These two measurements in question are the Index of Pupillary Activity (IPA) (Duchowski et al., 2018) and Low/High Index of Pupillary Activity (LHIPA) (Duchowski et al., 2020) which improve upon the original patented measurement without being put behind a paywall.

Within this paper, we will go through an implementation and update of these two pupillometric measurements along with potential additions that can be done to them while testing them on the Mixed Reality datasets (Lindlbauer et al., 2019) which had previously only used the original IPA measurement within it. The primary focus for us will be to not only re-use IPA on these datasets, but to see how Low- and High-Frequency IPA performs on these same datasets as an extension to their original paper.

Then we will go through the most state of the art version of this pupillometric measurement known as Real-Time Index of Pupillary Activity (RIPA) (Jayawardena et al., 2022) and how we could potentially use that in future versions of the Lindlbauer experiments.

All of the programmatic portions of this can be found on an Open-Source Github page with a link below[1].

2 Related Work

There are many papers in the field of eye tracking that utilize the ICA, IPA, LHIPA, and RIPA in their work, with many being discussed throughout the original papers that created these implementations. While it is not necessarily the point of this paper to review what other works these pupillometric measurements are used for, I will note that there are well over two hundred papers that are citing and using these metrics for their eye tracking tasks as of 2020 and the number is ever increasing as the number of variations of the measurement come out.

3 Formula Implementation

Before we go into greater detail of what will be done through programmatic implementation, we should discuss the actual formulas involved in these pupillometric measurements. While it is possible to do all of the image transformations through hard coding the mathematical formulas, you could

[1] Github Link

do most of the actual math via libraries such as Python's PyWavelet (Lee et al., 2019) which covers the majority of what we will be doing for this paper. Without going over every single formula written throughout the three papers of focus, we will instead go over the ones that play a key role within each papers' respective formula, how they factor into the programming of it, and also note which ones will be changed in later iterations of this calculation.

3.1 Index of Pupillary Activity (IPA)

Duchowski's Index of Pupillary Activity (Duchowski et al., 2018) will have the most formulas discussed as this one is the original version of the measurement–taking in about five formulas within the paper. While it is not necessarily important to understand these formulas to be able to read the code that will be provided later, it is important to understand where things were changed and why they were changed, hence these formulas being highlighted.

3.1.1 Dyadic Wavelet Decomposition

The fundamental aspect of wavelet analysis and what most of the formulas will base themselves on from this point onward, this utilizes the mother wavelet function $\psi_{j,k}(t)$ and incorporating a scaling factor at $2^{j/2}$. This will equate to the PyWavelet code known as "$sym16$".

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \quad where \quad j,k \in Z \tag{1}$$

3.1.2 Integral Transformation of Wavelet Coefficients

This part is where we will get our first actual line of code from PyWavelet called $pywt.wavedec()$, but this will span formulas 2 through 7 for us. These will only lightly be discussed now on what they do and why they are important to know.

The formula below is important because this is the baseline composition for the Integral Transformation, we will take the sum over all possible combinations of j and k integers through a square-integrable function–meaning that it will not become infinitely large when squared or integrated–through all possible whole numbers.

$$\mathrm{x} \in L^2(R) : x(t) = \textstyle\sum_{j,k=-\infty}^{\infty} c_{j,k}\psi_{j,k}(t), \quad where \quad j,k \in Z \tag{2}$$

Now we must make sure that the values we find are a complex conjugate–this ensures that the result is a real value–and plug in our Mother Wavelet Function from before.

$$\mathrm{c}_{j,k} = \textstyle\int_{-\infty}^{\infty} x(t)\overline{\psi_{j,k}(t)}dt, \quad where \quad x \in L^2(R), \quad j,k \in Z \tag{3}$$

With our formula plugged into the equation and our integrand is accounted for over the sigma summation, we now have almost everything we want in one spot. We replace the ψ with the Dyadic Wavelet Function that we had from before and we move the Scaling Factor to the outside of the integrand.

$$\mathrm{c}_{j,k} = 2^{j/2}\textstyle\int_{-\infty}^{\infty} x(t)\overline{\psi(2^j t - k)}\,dt, \quad where \quad x \in L^2(R), \quad j,k \in Z \tag{4}$$

If everything is done correctly, this should result a multi-resolution signal analysis process that uses a scaling function. The remaining functions that describe this signal analysis and the Discrete Wavelet Transformation (DWT) on a finer detail will be omitted as these will stay relatively the same throughout IPA and LHIPA. With the formula below being the cleaner version of the wavelet coefficients.

$$= \{W_\psi x(t)\}(j,k) = \langle x(t), \psi_{j,k}(t)\rangle \tag{5}$$

Now what is important to focus on for this and the other two pupillometric measurements is the formula listed below which is known as the Modulus Maxima Detection, and arguably the only part of the formula that remains relatively intact throughout every iteration.

This is a technique used in wavelet analysis, the wavelet modulus is used to identify significant changes within the wavelet coefficients that are obtained–these points are locally maximal. This is where the naming convention for this come from and the purpose of this is to identify important events within the signal for us to know when there may be statistically significant variations within the pupil diameter.

$$\begin{aligned} &|\langle x(t_0 - 1), \psi_{j,k}\rangle| \\ &\leq |\langle x(t_0), \psi_{j,k}\rangle| \\ &\geq |\langle x(t_0 + 1), \psi_{j,k}\rangle| \end{aligned} \tag{6}$$

At each of these points, we determine if the magnitude is larger than its neighbouring coefficients and if it is greater than both of its immediate neighbours, it is considered locally maximal, and it is

strictly larger than at least one of its neighbours then it is officially a modulus maximum. The modulus maximal detection is the primary focus of the IPA and all future iterations and it is imperative that you understand how it is determined.

$$\begin{aligned} &\text{and} \quad \{|\langle x(t_0), \psi_{j,k}\rangle| \\ &> |\langle x(t_0 - 1), \psi_{j,k}\rangle| \\ &\quad \text{or} \quad |\langle x(t_0), \psi_{j,k}\rangle| \\ &> |\langle x(t_0 + 1), \psi_{j,k}\rangle| \end{aligned} \tag{7}$$

After the modulus maximal detection, the only remaining step before determining the IPA is to threshold the signal to remove any additional noise. This formula will remain relatively the same throughout the iterations but the IPA version in particular will use a "hard" threshold.

Then we simply count the coefficients that remain and a higher count indicate that there is more significant and frequent changes in the pupil diameter and vice-versa for lower counts. Which is how we will determine if it may reflect stronger arousal levels within a person.

For everything that the original Index of Pupillary Activity (IPA) does right, there are still some issues that it struggles with and we will note them here. Whereas it was thought that the original IPA was immune to the effects of lighting because of the sampling rate found within it, the second iterative (Duchowski et al., 2020) paper discovered that it was not quite the case. IPA struggled with the effects of lighting and the law of initial value (Lacey, 1956), which make it less than optimal for the task at hand. This means that pupil may already be somewhat dilated in low-light (*E.g.* <70 lux) environments. And this will in turn effect the other pupillometric measures as well. On top of that, the IPA also suffers from prohibitive restrictions on movement and noise from even minor changes in axis distortions which make for a bad testing environment.

3.2 Low/High IPA (LHIPA)

The Low- and High-frequency Index of Pupillary Activity (LHIPA) has several changes that just improve upon the original iteration–such as more forgiving axis distortion–and also changes up the overall calculation in other unique ways that are noted below.

3.2.1 LF/HF Ratio

The first major distinction between the two IPA measurements, whereas IPA computed over all discontinuities in the entire signal, LHIPA focuses on a low- and high-frequency component as two separate entities. These values are then put through the same Discrete Wavelet Transformation (DWT) that we had from IPA (Duchowski et al., 2018) and then we will create the LF/HF Ratio from there as noted below:

$$\frac{(x_{\psi}^{\frac{1}{2}\log_2(n)}(t)}{x_{\psi}^{1}\left(2^{\frac{1}{2}\log_2(n)}t\right)} \tag{8}$$

This is calculated at the mid-level frequency octave of $j = \frac{1}{2}\log_2(n)$ through to the high frequency octave of $j = 1$ and is meant to capture the changes in pupil diameter before going through the same Modulus Maxima Detection. And the last notable change from there for the LHIPA is that it will take on a "less" approach for its threshold compared to the IPA versions' "hard" approach because LHIPA expects smaller responses from even greater cognitive loads than the original IPA. And to note from here, the LHIPA also considered smaller responses as good which is inverse to the original IPA where a higher count is indicative of higher cognitive load.

Similar to IPA, there are limitations to this iteration of the pupillometric measurement and these come in the form of light sensitivity issues just like the original IPA (Lacey, 1956) but in general it is lacking sensitivity in certain task conditions.

3.3 Real-Time IPA (RIPA)

Whereas LHIPA took on some modifications to fix the previous iteration of Index of Pupillary Activity (IPA), RIPA goes for something completely different and attempts to introduce real-time implementation into the analysis process over the offline approach that was currently dominating the other two versions. While this in turn is very useful for future iterations of the measurement, there is not much that will be discussed here as there is little implementation of this in the actual Github.

3.3.1 Savitzky-Golay filtering

But the macro-view of the changes come in the form of replacing the Discrete Wavelet Transformation (DWT) that we used in both other iterations with a Savitzky-Golay filter that allows for more seamless measuring because it factors in many items that the original just did not account for–mainly real-time processing, feature preservation, smoothing techniques, and finite impulse response

filtering. All of these sound very fancy, but what this means for the wavelet analysis is that we are 1) more efficient, 2) able to reduce noise and maintain even more peaks/valleys, 3) reduce noise through even more means, and 4) allow for a finite filter to be placed over the input signal.

3.3.2 Ring Buffering

And the final major change to the RIPA to the offline iterations of the IPA, is the way in which the data is stored. A ring buffer allows for a finite window of storage for the data that allows for a continuous stream of pupil diameter measurements over a specific timeline. This allows for real-time processing and a general increase in efficiency for the model to make it even more capable of doing real-time analysis.

4 Datasets

The datasets that will be used throughout the experiments are the eighteen datasets found within the 2019 Lindlbauer paper (Lindlbauer et al., 2019). These will go through various activities that–for the sake of brevity–are classified as 1) VR Tasks, 2) VR Music Tasks, 3) VR Miscellaneous Tasks, and 4) VR Room Search Tasks.

Specifically we will be using the datasets found in the first three and omitting 4) VR Room Search Tasks from the four studies provided. The data dictionary for these tasks if provided below, [2] but in general there is very little of this data that we will actually be able to use as the IPA/LHIPA/RIPA only necessarily need two inputs in the form of the pupil diameter and the time.

Study	Dataset
study01	p02-pupil_positions-02_trimmed
study01	p02-pupil_positions_trimmed
study01	p02-pupil_positions
study01	p03-pupil_positions
study01	p04-pupil_positions
study01	p05-pupil_positions
study01	p06-pupil_positions
study02	p01-pupil_positions
study03	pupil_positions

4.1 Dataset Limitations

The dataset dictionary is unfortunately unclear in its ability to distinguish timestamps for the data to be assessed. The two major assumptions made here are found with whether we believe "pupil_timestamp" from the data dictionary link provided is corresponding to "world_timestamp" which in turn is a very large number with no indication of measurement or if we believe "world_index" is the correct one.

In order to compensate for potential misunderstandings, both are considered within the code-base, we have one where "world_timestamp" is used as is and the other where "world_index" is used and the four frames are grouped and averaged amongst each other to equate to one second. In both cases, this seems to be questionable at best in practice and begs the question of if this data is suitable for the task at hand to begin with.

The other primary issue with the datasets are the general lack of information on which ones are suitable or if they are all suitable for the IPA/LHIPA/RIPA calculations. In order to qualify for assessment, each of the eighteen datasets were filtered on whether or not they had "world_timestamp", "world_index", and "diameter". There was also ones that did not start at "world_index" 0, but we ignored that for the time being.

Study	Dataset
study03	blink_detection_report
study03	blinks
study03	export_info
study03	gaze_positions_info
study03	world_timestamps
study04	recording01
study04	recording02
study04	recording03
study04	recording04

5 Experiment

The original experiment for the Lindlbauer paper was to take a short sliding window of the data in the form of 60 seconds and to take the IPA of that. There was tests done at both 30 seconds and 180 seconds as well, but the data to support that was not included so we have incorporated it into the final version of the code-base.

5.1 Premise

What we will do before the experiment is preprocess the datasets down to only those that qual-

[2] Pupil Player Core

ify with the PyPil_Datapreprocessor class (PyPil being short for Python Pupil as an adage to the PyWavelet library that so graciously carried so much of this project) so that all of our datasets are treated with the exact same cleaning and trimming process. While this was not necessary for the code implementation, it is important to have a clear and easily reproducible data cleaning process so it was incorporated into the final iteration.

5.2 Program-Wide Implementation Changes

Before going into what was specifically changed with each individual pupillometric measurement, there is the items that were changed within every single implementation that should be addressed all in one spot instead of repeated several times. These changes are within every part of the code-base and as such are noted here.

The entire pseudo-code has been given the "pythonic" treatment with packaging, classes, optimization, re-usability, modularity, and above all else well-documented. The original pseudo-code was just a few lines of code with no real functions, classes, or methods being utilized, meaning that if I were interested in doing this many times over, I would have to either rewrite the code, or reuse a lot of it through many different dataframes. So it was packaged to properly account for this and given classes and optimization via "pythonic" good practices.

On top of that, there was a great amount of the code-base that could be re-used between the three versions of the IPA, so instead of repeating it several times over, the code was refactored with subclasses to allow for the referencing of previous methods that would otherwise fit perfectly with what we needed. And to better improve upon this, modularity was included in several locations where a single word change in the calling of these classes will allow for us to do entirely different setups of the original code in situation such as dataframe, wavelet, periodization, level, and even specification of column for timestamps.

The entirety of the code-base is also meant to be very well documented and easily readable because while you could just use the pseudo-code provided by Duchowski, there is little understanding of the formulas used unless you are already familiar with Wavelet Analysis–so a great amount of effort has been put into it to allow for improved readability. This should help with future iterations of the code-base and allow for seamless interactions with others who may want to further improve the code themselves.

5.3 Experiment 1: Implement IPA

The pseudo-code provided in the Duchowski paper (Duchowski et al., 2018) is quite well written and there is very little changes that were needed to be done with the original code despite it being written in Python 2. But there were changes to the code-base nonetheless, the code for this can be found between the classes PyPil and IPA. The list of changes for IPA are quite short as the only notable differences are in the timestamps() and computer_threshold() methods in which modularity was provided to allow for future changes to the code without having to rewrite the entire thing.

5.3.1 Problems with Experiment 1

The code for IPA was quite easy to implement, but the sliding window that is discussed in the Lindlbauer paper (Lindlbauer et al., 2019) is not quite as clear as one would hope. There is a sliding window, and via the shifting_window_processing() function we were able to find a difference of 60.0 seconds to shift over at index 7000, but this took brute forcing to find and the results for it were quite poor with IPA calculations within the 0.15 to 0.20 range (in which case a high IPA is good and a low one is bad). This would lead us to believe that at least for the VR Task dataset found in study01 p02 did not perform very well.

5.3.2 Changes with Experiment 1

Due to the dataset composition, it would probably be best for us to present data that is clearly defined or with a clear 60 second or any number of second difference between them within the dataset to better find this difference as it is quite difficult for this to be found. Another key discussion point is to include a well defined data dictionary as it only was possible for us to understand that "world_index" was not the frames per second measurement only because it made no logical sense upon testing.

5.4 Experiment 2: Implement LHIPA

This one is arguably the most changed of the two that are fully implemented and can be found within PyPil and LHIPA classes. The two largest points of contention within the classes are the LF/HF Ratio and the timestamps() method were very difficult to get done correctly. The pseudo-code for this paper (Duchowski et al., 2020) is far less extensive and it

needed far more fixes as Python 2 had several errors pop up under the hood with just the base code on its own. After these revisions were made, there needed to be several renditions of the lhipa_ratio as it kept failing to converge to a number that was not infinite despite having to be a square-integrable function.

5.4.1 Problems with Experiment 2

The majority of the issues for Experiment 2 mirror the first experiment, the dataset itself proved to be far harder to work with than the actual methods, but the difference of 60 seconds needed to be brute force found and when we did the sliding window of 60 seconds on the datset, it did not turn out very well either.

5.4.2 Changes with Experiment 2

There would be many changes within the LF/HF Ratio calculation for the LHIPA, but many of them would be debugging error catches as that is what the majority of time I had to deal with for this one.

5.5 Experiment 3: Implement RIPA

This one will only remain on the Github as one that could only be done in theory, after the struggles that were involved with properly getting the other two to work with timestamps alone, the ability to get the Savitzky-Golay filtering and Ring Buffers to properly work with the dataset and existing methods were just going to be far too outside the scope of the experiment. Please review the code-base section referring to the RIPA if you would like to learn more about it.

6 Results

Due to the sheer need to brute force the index difference, we did not continue to do other datasets at this point in time, but we did do some preliminary testing of each of the datasets that were provided to us and they all score well below 0.5 for their IPA measurements. On top of that when looking over the LHIPA measurements through the same means, there were not that many good scores in that section as well with almost the entire LHIPA for study01 p02 that was tested being well over 4.0 for the entire time that it was assessed. And when you check over the other dataframes on a cursory search, none of them performed very well despite their global LHIPA in the initial check being closer to 1.5.

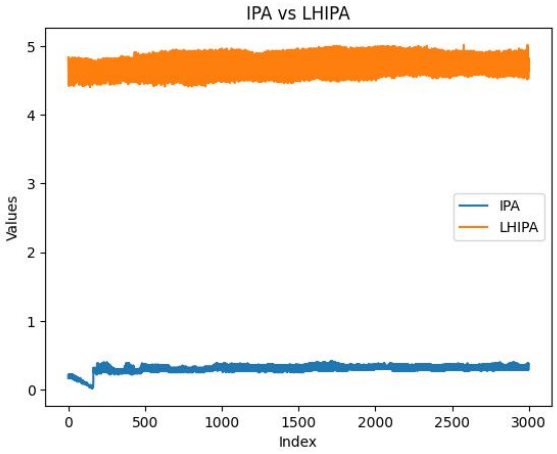

Figure 1: Example output for study01_p02-pupil_positions-trimmed-02.

7 Criticism

Overall we think that the IPA and LHIPA implementation had gone quite well but the dataset that was provided for it just did not live up to the task conditions set up for it. However it was done in the original Lindlbauer paper is unclear enough that we were not able to properly replicate everything that was done.

Acknowledgments

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Deborah A Boehm-Davis, Wayne D Gray, Leonard Adelman, Sandra Marshall, Robert Pozos, and GEORGE MASON UNIV FAIRFAX VA DEPT OF PSYCHOLOGY. 2003. Understanding and measuring cognitive workload: A coordinated multidisciplinary approach. *George Mason University, Fairfax, VA2003*.

Andrew T Duchowski, Krzysztof Krejtz, Nina A Gehrer, Tanya Bafna, and Per Bækgaard. 2020. The low/high index of pupillary activity. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Andrew T Duchowski, Krzysztof Krejtz, Izabela Krejtz, Cezary Biele, Anna Niedzielska, Peter Kiefer, Martin Raubal, and Ioannis Giannopoulos. 2018. The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13.

Gavindya Jayawardena, Yasith Jayawardana, Sampath Jayarathna, Jonas Högström, Thomas Papa, Deepak Akkil, Andrew T Duchowski, Vsevolod Peysakhovich, Izabela Krejtz, Nina Gehrer, et al. 2022. Toward a real-time index of pupillary activity as an indicator of cognitive load. *Procedia Computer Science*, 207:1331–1340.

John I Lacey. 1956. The evaluation of autonomic responses: Toward a general solution. *Annals of the New York Academy of Sciences*, 67(5):125–163.

Gregory Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. 2019. Pywavelets: A python package for wavelet analysis. *Journal of Open Source Software*, 4(36):1237.

David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 147–160.

Sandra P Marshall. 2002. The index of cognitive activity: Measuring cognitive workload. In *Proceedings of the IEEE 7th conference on Human Factors and Power Plants*, pages 7–7. IEEE.

Sharon Oviatt. 2006. Human-centered design meets cognitive load theory: designing interfaces that help people think. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 871–880.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285.

A Appendix: Hard Coded IPA

Version 1 of the code-base found in the Unmaintained Codebase folder on the Github is where we were previously doing all of the Wavelet Decomposition and Discrete Wavelet Transformations (DWT) by hand before this was no longer considered important to the task at hand. It will remain unmaintained, but for those who may be interested in working on the finer details of what PyWavelet are doing under the hood, they should go check that out as well as PyWavelets' website found here.

B Appendix: ChatGPT/ Github CoPilot Utilization

As part of the full disclosure agreement for the project, as discussed with advisors, both ChatGPT and Github CoPilot have been utilized in the creation of this project for various tasks. The primary tasks that they handled were broken down in the table listed below.

ChatGPT	**GitHub Copilot**
Spelling corrections	Autocompletion of functions
Creating LaTeX formulas	Noting missing variables/items
Formatting tables and images	Try/except error blocks
As a debugger	As a debugger

C Appendix: PyPil_Datapreprocessor Class

The code-base includes an additional tool called the Pypil_Datapreprocessor, which is mentioned previously within §Datasets. This was created in part because of the sheer size and number of datasets that needed the exact same pre-processing steps. Its purpose is to mainly do the following actions:

- Allows you to search through all existing folders in the chosen directory.
- Allows you to limit how deep you go through this folder chain.
- Removes Null/NaNs.
- Trims columns to a preset list.
- Provides suffixes and prefixes for each CSV change.
- Marks all CSVs that provided an error.

- Marks all CSVs that were disqualified via column trimming.
- Creates a relevant folder for you to store them.
- Stores all of them in said folder.

D Appendix: PyPil Class

The second part of the code-base and arguably the focus of it. This did not need to be nearly as polished as it was, but the original intention of this was to get it to the point where it can be a functional Python Package with a bit more modifications put into it. This is why there is so much documentation within each of the Classes and Methods.