eyeris: A flexible, extensible, and reproducible pupillometry preprocessing framework in R

Shawn T Schwartz[1,2,3,4,5†*], **Haopei Yang**[1,2†], **Alice M Xue**[1,2,3,5†], **Mingjian He**[1,2,6†‡]

***For correspondence:** stschwartz@stanford.edu (STS)

Present address: †Department of Psychology, Stanford University, United States; ‡Department of Anesthesiology, Perioperative, and Pain Medicine, Stanford Medicine, United States

[1]Stanford University, Stanford, United States; [2]Stanford Memory Laboratory, Stanford, United States; [3]Stanford University Wu Tsai Neurosciences Institute, Stanford, United States; [4]Stanford University Wu Tsai Human Performance Alliance, Stanford, United States; [5]Stanford University Center for Mind, Brain, Computation and Technology, Stanford, United States; [6]Stanford University Medicine, Stanford, United States

Abstract Pupillometry provides a non-invasive window into the mind and brain, particularly as a psychophysiological readout of autonomic and cognitive processes like arousal, attention, stress, and emotional states. Pupillometry research lacks a robust, standardized framework for data preprocessing, whereas in functional magnetic resonance imaging and electroencephalography, researchers have converged on tools such as fMRIPrep, EEGLAB and MNE-Python; these tools are considered the gold standard in the field. Many established pupillometry preprocessing packages and workflows fall short of serving the goal of enhancing reproducibility, especially since most existing solutions lack designs based on Findability, Accessibility, Interoperability, and Reusability (FAIR) principles. To promote FAIR and open science practices for pupillometry research, we developed `eyeris`, a complete pupillometry preprocessing suite designed to be intuitive, modular, performant, and extensible (https://github.com/shawntz/eyeris). Out-of-the-box, `eyeris` provides a recommended preprocessing workflow and considers signal processing best practices for tonic and phasic pupillometry. Moreover, `eyeris` further enables open and reproducible science workflows, as well as quality control workflows by following a well-established file management schema and generating interactive output reports for both record keeping/sharing and quality assurance of preprocessed pupil data prior to formal analysis. Taken together, `eyeris` provides a robust all-in-one transparent and adaptive solution for high-fidelity pupillometry preprocessing with the aim of further improving reproducibility in pupillometry research.

Introduction

The measurement of pupil size and its relation to cognition dates back to the late 1800s with investigation of the pupillary light reflex (***Haab, 1886***). More than a century later, pupillometry underwent a present-day (re)discovery, establishing pupil size as an integrated psychophysiological readout of cognitive processes such as attention. More specifically, pupillometry offers a non-invasive window into the human mind and brain, and is sensitive to environmental (i.e., luminance, focal distance) and cognitive (i.e., alerting, orienting, attention, executive functioning) changes (***Huang and Clewett, 2024***; ***Strauch, 2024***; ***Strauch et al., 2022***).

The breadth of cognitive processes that have been associated with pupil size dynamics is expansive, including – but not limited to – emotional arousal (***Bradley et al., 2008***; ***Oliva and Anikin, 2018***), mental effort (***Kahneman and Beatty, 1966***; ***Kramer et al., 2013***), fatigue/task performance

(***Aston-Jones and Cohen, 2005a***; ***Eldar et al., 2013***; ***Knapen et al., 2016***; ***McGinley et al., 2015***; ***Murphy et al., 2011***), response preparation (***Einhäuser et al., 2008***; ***Reimer et al., 2014***), uncertainty/decision-making (***Gee et al., 2014***; ***Jepma and Nieuwenhuis, 2011***; ***Kawaguchi et al., 2018***), prediction error (***Alamia et al., 2019***), working memory (***Keene et al., 2022***; ***Robison and Unsworth, 2019***; ***Unsworth and Robison, 2015***, ***2017a***,***b***), and episodic memory (***Clewett et al., 2020a***; ***Madore et al., 2020***; ***Madore and Wagner, 2022***; ***Miller et al., 2019***; ***Miller and Unsworth, 2020***; ***Robison et al., 2022***; ***Schwartz et al., 2025***).[1]

As highlighted above, the majority of pupillometry studies use pupil size as an assay of attentional engagement or examine the impact of cognitive processes on pupil size (***Kristjansson et al., 2009***; ***Madore et al., 2020***; ***van den Brink et al., 2016***), using pupil diameter dynamics as a non-invasive (indirect) surrogate of locus coeruleus-noradrenergic (LC–NA) neuromodulatory activity (***He et al., 2020***; ***Joshi et al., 2016***; cf. ***Megemont et al., 2022***). Additionally, pupil diameter has been shown to covary with functional magnetic resonance imaging (fMRI) blood-oxygenation-level–dependent (i.e., BOLD) activity in human LC (***Murphy et al., 2014***). More recently, fMRI has been used to identify neural networks in human brain associated with pupil diameter dynamics in healthy young and older adults and revealed decreases in functional connectivity within a so-called pupil diameter network with healthy aging (***Wu et al., 2025***). In most studies, analyses of these pupil data are conducted offline following data collection, although there are also real-time applications providing causal evidence confirming the use of moment-to-moment pupil dynamics as an assay of attentional engagement (***Keene et al., 2022***; ***Meissner et al., 2024***; ***Schwartz et al., 2024***, ***2025***). Additionally, it is worth noting that given LC's central role in regulating autonomic function and arousal (***Poe et al., 2020***; ***Samuels and Szabadi, 2008***) and that LC is the first brain region to develop tau pathology in Alzheimer's disease (***Betts et al., 2019***; ***Braak and Del Tredici, 2011***; ***Holper et al., 2022***; ***Spires-Jones and Hyman, 2014***), pupillometry is also being used as a non-invasive window into cognitive aging (***El Haj, 2024***; ***Kim et al., 2024***; ***Piquado et al., 2010***).

Consequently, collecting high-quality pupil data and appropriately analyzing these data are critical for advancing both theories of cognition as well as translational applications of pupillometry. Moreover, data preprocessing is a fundamental prerequisite to high quality pupillometry data analysis (***Fink et al., 2024***; ***Hershman et al., 2024***; ***Kret and Sjak-Shie, 2019***; ***Laeng and Mathôt, 2024***; ***Mathôt and Vilotijević, 2023***; ***Steinhauer et al., 2022***; ***van Rij et al., 2019***).[2] In pupillometry research, experimenters deal with fewer time series, at most two for binocular recordings, compared to electroencephalography (EEG) or fMRI, but many of the same principles apply: *handling missing data*, *excluding artifacts*, and *smoothing out physiologically irrelevant noise*, to name a few.

Numerous researchers have contributed excellent software packages to the community over the years to aide others in preparing and analyzing pupillometry data without the need to implement core preprocessing functions themselves (e.g., ***Forbes, 2020***; ***Geller et al., 2020***; ***Hershman et al., 2019***; ***Kinley and Levy, 2021***; ***Tsukahara, 2020***). However, most of the existing tools only provide rigid preprocessing specifications, often have been factored out of particular analysis projects, and/or do not empower users to be deliberate in preprocessing choices and to compare visualizations of preprocessed pupil data to that of raw pupil data.

Thus, a critical gap in current pupillometry research is the absence of an integrated, standardized, and adaptable preprocessing pipeline – a point emphasized by ***Mathôt and Vilotijević*** (***2023***), who note:

> There is no commonly agreed-upon workflow for conducting cognitive-pupillometry experiments: design criteria, preprocessing steps, and statistical analyses all differ vastly between studies. Some attempts at standardization have been made, but these are not easily applicable to cognitive pupillometry [...] Furthermore, guidelines are often conceptual rather than specific implementations, and the lack of concrete examples and code makes it difficult for researchers to incorporate these guidelines into their own workflow.
>
> (p. 3055)

In contrast, tools developed for other neuroimaging modalities over the past few years, notably fMRIPrep for fMRI data (***Esteban et al., 2020***, ***2019***), have defined new standards for transparent and reproducible data preprocessing workflows. Popular software packages initially built for data analysis, such as EEGLAB (***Delorme and Makeig, 2004***) and MNE-Python (***Gramfort et al., 2013***) for M/EEG, are also incorporating preprocessing pipelines into their functionalities to facilitate more rigorous preprocessing with carefully prescribed default settings.

However, developing and sustaining robust preprocessing pipelines is not trivial. Many scientists with strong domain expertise lack formal training in computer science or software engineering (***Chen et al., 2025***; ***Segal, 2008***), a likely contributing factor to the inadvertent adoption of common programming "anti-patterns" when writing analytic code (***Wilson et al., 2014***). In fact, non-reproducibility in science often results from some combination of poor programming practices (***Hatton, 1997***; ***Hatton and Roberts, 1994***; ***Merali, 2010***; ***Miller, 2006***), suboptimal and inattentive code and data management and sharing practices (***Frank et al., 2025***; ***Gorgolewski and Poldrack, 2016***; ***Kovacs et al., 2024***), and/or insufficient transparency about the specific details underlying each step taken in between to transform raw data into a high quality format appropriate for inferential statistics (***Poldrack et al., 2019***). Over and above these pitfalls is the "garden of forking paths" problem and the culprit of Type I (i.e., false positive) errors in extant literature (***Gelman and Loken, 2013***; ***Poldrack et al., 2017***; ***Simmons et al., 2011***).

In the context of psychological science, flexibility in data analysis and reporting has been implicated as a prominent contributor to the replication and reproducibility crisis (***Hardwicke et al., 2021***, ***2018***, ***2022***; ***Ioannidis, 2005***; ***Open Science Collaboration, 2015***). Likewise, the garden of forking paths for psychophysiological data is a slippery slope: when researchers are afforded more preprocessing pathways and analytic degrees of freedom, even small adjustments to the way data are preprocessed (e.g., digital filter design, handling of missing data, artifact removal) can impact the outcomes and interpretations a researcher may draw from their data (***Šoški'c et al., 2024***).

With `eyeris`, we not only intend to provide users with an extensible toolkit for conducting robust pupillometry preprocessing, but critically aim to contribute to standardizing best practices in pupillometry research (especially for cognitive pupillometry; see ***Mathôt*** (***2018***) for a review discussing pupil responses, the neural pathways subserving these pupillary responses, and their relationship to high-level cognition). We should note that we are not the first group to put forth a recommended set of preprocessing guidelines for cognitive pupillometry. Here, we have built upon the comprehensive guide to cognitive pupillometry preprocessing put forth by ***Mathôt and Vilotijević*** (***2023***); their stated aim is to provide "guidelines [that] are meant to be illustrative, rather than prescriptive; that is, we show how things *could* be done – rather than how they *should* be done – in order to implement a workflow that is both easy to implement and that meets contemporary standards for good scientific practice" (p. 3056). In contrast, we adopt a **prescriptive**, rather than merely *illustrative*, approach to pupillometry preprocessing workflows.

Here, we argue that a standardized cognitive pupillometry preprocessing workflow is essential to narrow the number of possible analytic paths researchers might take through the garden. In other words, without consistent preprocessing, studies leveraging pupillometry to advance key theories in psychological science could be hindered by high degrees of freedom and vulnerable to unintended mistakes during preprocessing. As such, our accompanying R package is designed with "glass box" principles (see ***Esteban et al., 2019***; ***Poldrack et al., 2019***), thereby equipping both novice and experienced pupillometry researchers with key tools needed to appropriately transform raw pupillometry data into a format well-suited for downstream analysis, visualization, and statistical inference. Embracing a carefully designed pipeline is highly beneficial, as it enables researchers to focus on hypothesis testing with minimal worry about whether the data were preprocessed correctly. In this era of open-source software, it is also important to allow experienced users to inspect and modify steps in a preprocessing pipeline with minimal hurdles.

To fill this gap, we developed `eyeris`, a dedicated R package that aims to provide unified standards for pupillometry preprocessing (***Schwartz, 2025a***). `eyeris` strongly encourages users to fol-

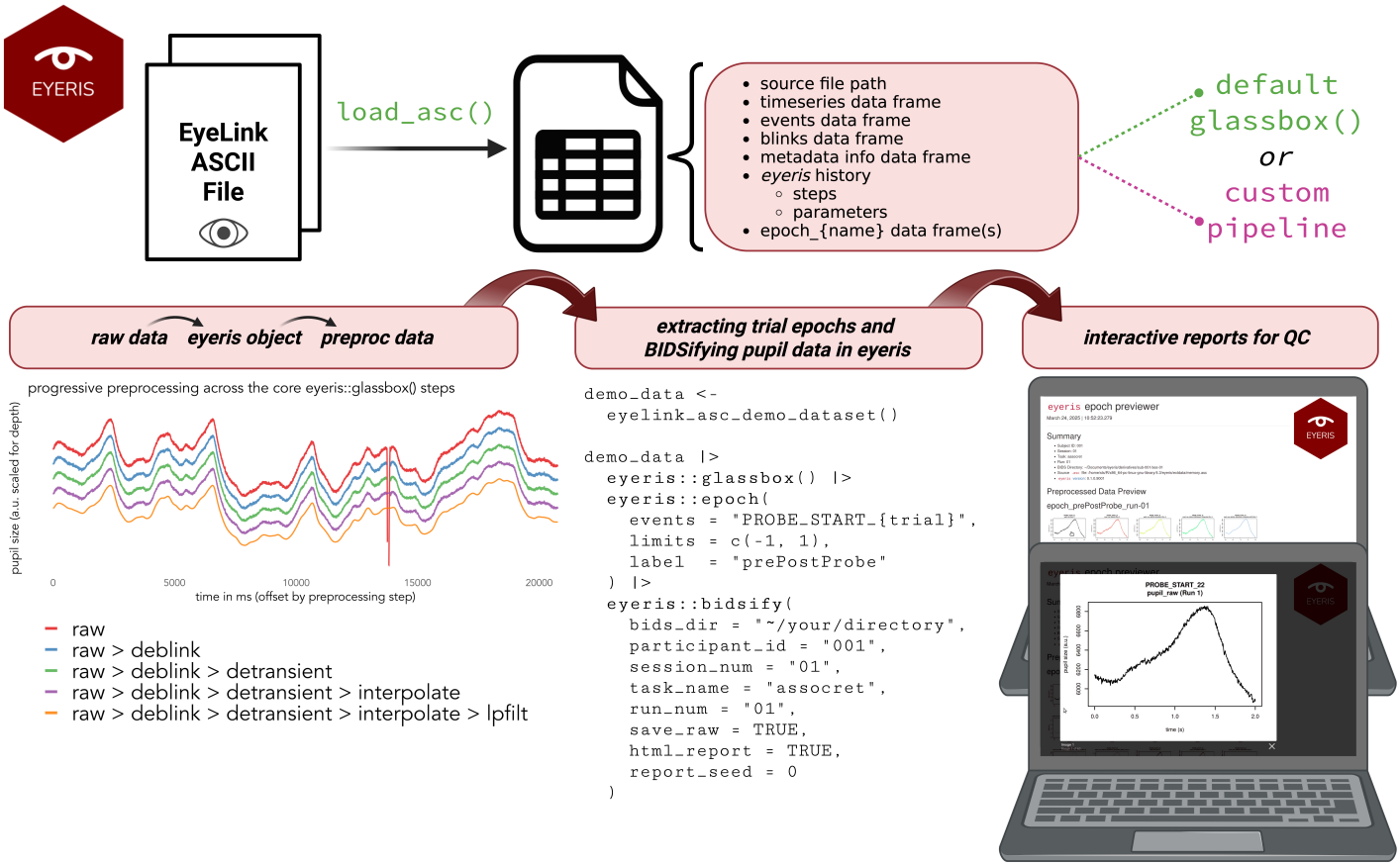


Figure 1. A sample workflow for the `eyeris` R package. The example code shows an end-to-end preprocessing pipeline, starting at raw data from the tracker to epoched and derived files ready for visualization and analysis. Here, we draw particular attention to the flexibility `eyeris` offers for segmenting (i.e., epoching) trials from the continuous pupil time series. In this demonstration, the ***epoch() function*** will find all event messages from the recording session that include the "PROBE_START_*" wildcard expression and extract 1-second prior to as well as 1-second following the onset timestamp from the matched message string (and will name the resulting tidy data frame of extracted pupil epochs: "prePostProbe", which will be saved to the resulting `eyeris` S3 class "list" object – *not shown here*). Note, the "{trial}" syntax is a feature that enables a user to tell `eyeris` to not only treat that portion of the string as a wildcard expression, *but also* to take whatever substring matched to that position of the string and place its content in a new column with the header title: `trial`. This feature was designed with the intention to streamline the extraction of metadata embedded within tracker event messages into the same tidy data frame as any corresponding pupil data extracted from these events. The `epoch()` function is quite flexible and can handle a wide range of event message-string configurations; an in depth walkthrough can be found in the package documentation. Created in BioRender. ***Schwartz*** (***2025b***) https://BioRender.com/vkahl3u.

low a prescribed set of steps, in a particular order, with carefully selected default parameters, aiming to avoid unnecessary preprocessing decisions that may lead to issues when made with incomplete knowledge of some of the nuances of signal processing. At the same time, `eyeris` provides transparent access and configurability to all preprocessing steps in a user-friendly fashion, which is supported by its modular, glass-box design (***Figure 1***, ***Box 1***). This design approach greatly reduces the burden of usage by removing the need to develop lengthy scripts for individual use cases (***Varoquaux, 2016***).

Key features & innovations

Here, we introduce `eyeris` with the goal of making pupillometry preprocessing more efficient, reproducible, and conducive for reliable inferences as well as providing researchers greater confidence when analyzing their pupillometry data.

1. Common pupillometry preprocessing steps performed in a prescribed order.

Pupillometry time series require a relatively small number of preprocessing steps compared to multivariate data (such as EEG and fMRI); yet, there is a more defined set of issues to be addressed when preprocessing pupil size data. Critically, these preprocessing steps must be completed in a certain order to avoid unexpected modifications of the data in a way that could compromise downstream analyses. In this section, we describe some of the most common preprocessing steps for pupillometry data, which are implemented in the current version of `eyeris`. While additional steps might be added in the future, some of the basic principles covered here will be applicable in general.

First, preprocessing of pupil sizes needs to handle artifactual or missing data due to eye blinking, movement, or simply incorrect pupil detection and measurement by the eye tracker. Most existing commercial eye trackers, including SR Research EyeLink[3], have built-in blink detection methods that will annotate blink periods in exported data. One can easily recognize blink periods when the pupil size becomes `0`/`NaN` for a brief period of time. However, it is important to recognize that there are often blink-related artifactual changes in pupil size surrounding an eye blink, which can be due to the eyelid rapidly covering the pupil area detected by the camera, resulting in rapidly diminishing pupil size before becoming zero. In contrast, as the eyelid opens, there is a brief period of rapidly increasing pupil size when the camera detects a partial pupil. Occasionally, eye trackers may incorrectly lock onto other structures in the camera's field of view, such as eyebrows and glasses frames, as they may become more probable pupil targets compared to partially occluded pupils during blinking. This can also result in fluctuations in recorded pupil sizes. As such, `eyeris` removes blink periods using the ***deblink() function*** with an `extend` duration (defaults to $50\,\mathrm{ms}$) bleeding both backward and forward in time to mask neighboring time periods that may contain such artifacts. Note, the *extend* duration parameter may need to be adjusted for different experimental setups, certain groups of participants, or even for individual participants, which can be determined on a case-by-case basis upon inspection of visual diagnostic reports.

While commercial eye trackers can mark blink periods for easy exclusion, there are also artifactual pupil size measures that could occur in recordings due to unstable pupil tracking or eye/-head movements. These non-physiological disruptions of pupillometry measurement often result in transient and abrupt changes in recorded pupil sizes, commonly visible as spikes in the time series. `eyeris` detects and removes such transient artifacts using the ***detransient() function***. Under the hood, `detransient()` computes an approximate pupil change velocity by taking the first-order difference divided by sampling interval and detects outliers using the Median Absolute Deviation (MAD; ***Arachchige and Prendergast, 2024***; ***Hampel, 1974***, ***1968***; ***Rousseeuw and Croux, 1993***; ***Rousseeuw et al., 1986***) procedure. In our experience, most artifacts in pupillometry data can be captured using the simple `detransient()` method, thereby excluding non-physiological measures. However, noisier recordings with different eye tracker hardware may require more elaborate artifact detection algorithms, such as to remove sustained periods of outlying pupil sizes instead of velocity. Users are strongly encouraged to expand on the function given their data and contribute to `eyeris` with additional methods to remove artifact periods (see ***Box 7*** and ***Appendix 1*** for more information about extensibility and developing custom extensions in `eyeris`).

Once data from blink and artifact periods are removed, the next step in pupillometry preprocessing is to interpolate over these periods to recover a contiguous time trace of pupil sizes (***Mathôt et al., 2013***). While not strictly required for certain types of pupillometry analysis – such as averaging across trial periods for a univariate measure – many recent analytic approaches for pupillometry data leverage each sampled time point to uncover temporal dynamics in changes of pupil sizes (***Clewett et al., 2020b***; ***Johansson et al., 2018***; ***Steinhauer et al., 2004***; ***Urai et al., 2017***; ***Van Slooten et al., 2018***; ***Verney et al., 2004***; ***Widmann et al., 2018***). With these forms of analysis, avoiding missing data in certain periods provides more evenly distributed statistical power across time points (even with "guessing" of missing data through interpolation). Accordingly, `eyeris` uses the

interpolate() function and takes the simplest approach of linear interpolation due to the relatively slow nature of pupil size fluctuations. However, users are strongly encouraged to adopt cutoff thresholds to exclude periods or trials when more than a certain percentage (such as 50%) of raw data is missing.

It is evident that the interpolation steps should occur after missing-data detection, but it is less clear if, when, and how smoothing of typically noisy pupillometry data should take place. Smoothing of pupil sizes over time is commonly employed in the literature (***Steinhauer et al., 2022***) both to provide cleaner time trace visualizations *and* to reduce sporadic temporal dynamics that may be unrelated to underlying neurocognitive processes of interest. Smoothing is justified for pupil sizes with relatively slow dynamics but should always occur *after* blink and artifact detection to avoid contaminating neighboring samples during smoothing. In addition, smoothing can be achieved in multiple different ways, such as with moving averages or low-pass filtering. `eyeris` uses the ***lpfilt() function*** to apply a low-pass Butterworth filter with passband starting at $4\,\mathrm{Hz}$ and stopband starting at $8\,\mathrm{Hz}$. Note, the `lpfilt()` function here uses a minimum-order Butterworth filter, whose order is obtained with the `buttord()` function from the `gsignal` package (***van Boxtel, 2021***) to meet the defined passband and stopband cutoff frequencies (i.e., `wp`, `ws`) and ripple/attenuation requirements (i.e., `rp`, `rs`). The filter created by the `gsignal::butter()` function is returned in the second-order sections format. Users can easily modify the filter design parameters to achieve different smoothing results, but should always inspect the frequency response of the filter to ensure sufficient signal attenuation at higher frequencies and minimal ripples in the passband. The frequency response can be visualized by passing the `plot_freqz = TRUE` parameter to `lpfilt()`.

Pupil sizes can fluctuate at a slow pace over the course of an entire recording; for example, the commonly reported decreasing trend during a cognitive task (a.k.a., the "time-on-task effect"; ***Beatty, 1982***; ***Hopstaken et al., 2015***; ***Martin et al., 2022***; ***Massar et al., 2016***; ***Steinhauer et al., 2022***; ***Unsworth and Robison, 2016***; ***Zhao et al., 2019***). Depending on the scientific questions at hand, removal of this time-on-task effect may be desirable. `eyeris` provides the ***detrend() function*** to remove a linear trend over the entire recording. However, it should be noted that pupillometry analysis often employs baseline correction in a trial-by-trial design; as such, detrending might be unnecessary and could also increase the risk of introducing artifacts in the data if an extreme trend gets extracted (although unlikely as these artifacts were likely removed in the preceding steps). Moreover, high-pass filtering is discouraged as a way to remove slow trends in pupillometry data because designing a filter with a sufficiently sharp cutoff at the lower frequency end is difficult; failing to implement a proper filter design could drastically change the waveforms of pupil sizes that contain frequency content at approximately $1\,\mathrm{Hz}$. Lastly, slow trends in pupil sizes might be physiologically meaningful, as tonic pupil sizes may reflect tonic arousal levels (***Aston-Jones and Cohen, 2005b***; ***Groot et al., 2021***; ***Joshi and Gold, 2020***; ***Madore et al., 2020***; ***Madore and Wagner, 2022***; ***Murphy et al., 2011***; ***Schwartz et al., 2025***; ***Smallwood et al., 2012***; ***Tsukahara et al., 2016***; ***Unsworth and Robison, 2018***). Depending on the target research question(s), such trends may need to be preserved for downstream analysis. Therefore, `eyeris` defaults to *not performing* detrending during preprocessing.

Finally, there has been an ongoing debate regarding z-scoring normalization of pupillometry data. Some have argued against doing so outside of MRI contexts (***Steinhauer et al., 2022***). Normalizing pupil sizes by the mean and variance can introduce additional challenges in interpreting derived measures. Yet, it can also be argued that in the absence of normalization, pupil data across individuals cannot be easily pooled together for statistical inference, even if the non-normalized pupil sizes faithfully reflect individuals' physiological responses. To illustrate, older adults and young adults may have differences in physical constraints that impact how much the pupil can dilate during cognitive performance. Not accounting for such differences may lead to the conclusion that older adults have weaker pupillary responses to certain stimuli (such as a loud tone). However, if different ranges of pupil fluctuations across individuals are accounted for by using z-scored pupil sizes, older adults may show comparable responses to such stimuli relative to their

Box 1. Example usage of the *glassbox* pipeline wrapper

```
library(eyeris)

path_to_demo_dataset <- eyelink_asc_demo_dataset()

cleandata <- glassbox(file = path_to_demo_dataset)
#> [  OK  ] - Running eyeris::load_asc()
#> [  OK  ] - Running eyeris::deblink()
#> [  OK  ] - Running eyeris::detransient()
#> [  OK  ] - Running eyeris::interpolate()
#> [  OK  ] - Running eyeris::lpfilt()
#> [ SKIP ] - Skipping eyeris::detrend()
#> [  OK  ] - Running eyeris::zscore()

plot(cleandata)
#> ! Plotting block 1 from possible blocks: 1
```

normal ranges of pupillary responses (***He et al., 2020***). Therefore, while older adults may show attenuated changes in pupil size due to physiological changes in muscles controlling pupil dilation, their cognitive arousal response to tone stimuli may not be preferentially reduced to suggest a selective deficit. Similar examples can be found even within a homogeneous group, since differences across individuals and across blocks/sessions are often present in each recording (e.g., ***Madore et al., 2020***). Averaging non-normalized pupillary responses produces a metric biased by absolute pupil size, potentially confounding the cognitive (pupil) signal of interest with physiological variability.

Hence, z-scoring pupillometry data allows one to focus on the relative dynamics of fluctuations in pupil sizes rather than the first-level characteristics such as mean pupil sizes or absolute changes. It should be emphasized that z-scoring automatically makes the assumption that the variances accountable by changes in the absolute value or range of pupil sizes are irrelevant to target scientific questions. In many modern applications of pupillometry within psychological sciences, this is not an unreasonable assumption (i.e., ***Bradshaw, 1969***; ***Gilzenrat et al., 2010***; ***He et al., 2020***; ***Peysakhovich et al., 2017***; ***Van Gerven et al., 2004***). Therefore, `eyeris` defaults to perform z-scoring normalization with the ***zscore() function***. One can easily turn off z-scoring during preprocessing, and we strongly encourage users to validate the robustness of their main findings as a consequence of choosing whether or not to perform normalization of pupil sizes across sessions, individuals, or cohorts.

2. A modular signal processing workflow with carefully chosen parameters ensures efficient and robust pupillometry preprocessing.

One key strength of the `eyeris` framework is its modularity. Each preprocessing function within `eyeris` is a wrapper around a core pipeline operation that automatically tracks, versions, and stores preprocessing states for any pupillometry dataset being processed in `eyeris`. This enables a "plug-and-play" experience when using `eyeris`: users have the option to use our recommended ***glassbox() method***, as well as the ability to construct novel extensions and custom pipelines. Critically, this modular architecture promotes both adaptability and extensibility, facilitating the streamlined integration of custom preprocessing methods directly into the default workflow (***Box 1,2,6,7; Appendix 1***).

Box 2. Tips & Tricks

Consider preprocessing your data with `eyeris::glassbox()` rather than manually calling individual steps.

Although we encourage most users to use the `glassbox()` function for simplicity and adherence to signal processing best practices, advanced users can create custom preprocessing steps that seamlessly integrate into the workflow (and thus will behave like native package functions without the need for hacky integrations). Therefore, it is imperative that the custom pipeline steps conform to the `eyeris` protocol to ensure maximum compatibility and reproducibility with downstream functions. We provide a thorough vignette that walks users through the process of creating custom extensions in the package documentation (https://shawnschwartz.com/eyeris/articles/custom-extensions.html) and in ***Advanced features for developers***.

We recommend that users start with the `glassbox()` function to reduce the probability of inadvertently introducing sub-optimal computations or errors that could arise when reimplementing the individual steps underlying the preprocessing pipeline – both within and between projects. The `glassbox()` function allows users to make flexible adjustments – such as disabling a particular step and/or changing the default settings for a step – by passing in parameter arguments (see ***Box 6***). A strength of the built-in `glassbox()` function compared to the many extant pupillometry packages is the framework it provides for rapidly preprocessing pupillometry data without requiring users to rely on copy-pasting complex step-by-step functions employed on a toy dataset from a vignette (which may not necessarily be optimized for the variety of forms that real-world pupillometry data come in). Furthermore, even when the instructions and functions supplied in a vignette of a package work "out-of-the-box" after pasting them into a custom script, it is still the user's responsibility to ensure that *every step* was not only run, but done so *in the correct order* (with respect to the best practices of signal processing). Failing to do so could lead to unexpected and possibly detrimental results for the conclusions researchers will ultimately draw from the data. Therefore, we aim to facilitate this process to enable users to more easily engage with preprocessing steps by using a package, rather than moving away from them. With the `eyeris glassbox()`, users can take a pupillometry recording file directly from a participant session, and in a short amount of time can go from raw to fully preprocessed data with detailed diagnostics for quality control in approximately three lines of code (***Box 1***,***2***; ***Figure 2***).

However, one critique of encapsulating a series of preprocessing steps into a single wrapper function – like `glassbox()` – is the regression towards a more black-box-like implementation. `eyeris` circumvents this pitfall in two key ways. First, the `glassbox()` function has two modes: *interactive* and *headless*. The interactive mode allows users to evaluate the consequences of the parameters used within each step on their data in real time, thereby facilitating an easy-to-use workflow for parameter optimization on any given dataset. In essence, this interactive process takes each recommended step and forces the user to evaluate both pre- and post-processing plots of time series for all steps in the default pipeline, with interjecting prompts to continue or modify parameters before moving forward. This stepwise approach to optimizing signal preprocessing parameters that yield satisfactorily de-noised pupil time series data reduces the opacity of the wrapper function to a transparent and engaging experience. The headless mode permits users to run the pipeline as-is on a dataset of any size – be it on their computer locally or on a cloud computing cluster – to preprocess large-scale pupillometry datasets using our recommended default parameters *without manual intervention*.

Second, `eyeris` stores the history of preprocessing steps and parameters for every action made on a raw dataset so that any derived data from the pipeline can be easily reproduced without ad-

ditional steps needed from the researcher. Specifically, all of these data are saved to interactive `HTML` files (***Figure 2***), which contain numerous diagnostic plots for every step of the pipeline, automatically parsed tracker metadata, and a snapshot of the actions performed at each step of the workflow[4]. Thus, using `eyeris` to deploy a large-scale preprocessing workflow on a headless computing architecture allows researchers to manually scrub their data sets for each run from every subject at every step of preprocessing *without additional custom scripting*. Critically, these interactive reports provided by `eyeris` go above and beyond within-study data quality control by automatically capturing and recording data and signal processing provenance in a standardized schema (***MacKenzie-Graham et al., 2008***) – an essential feature for analytic reproducibility, lacking in many existing pupillometry preprocessing tools.

3. User-friendly pattern matching and metadata parsing make extracting trial-level epochs both accurate and effortless.

Extant solutions for preprocessing pupil data lack intuitive workflows to extract trial-level epochs. For EEG researchers used to functions like `pop_epoch` in the MATLAB-based EEGLAB software (***Delorme and Makeig, 2004***), the lack of a comparable analogous function can lead to bottlenecks in a researcher's pupillometry preprocessing workflow. Naturally, it made sense to take inspiration from EEGLAB when designing the `epoch()` function within `eyeris`. In particular, extracting trial-level data epochs with `eyeris` is flexible and offers advanced tooling for the extraction of metadata from event message strings. To illustrate, a researcher may have event message tags that look something like: `PROBE_START_32_PIANO`, indicating the timestamp for the stimulus probe onset on trial 32 where the stimulus was the piano. While not all event message strings contain as much information as this contrived example (e.g., standard EyeLink trial markers `TRIALID 1` often provide helpful trial-number metadata), the additional sources of information in the former example can be leveraged by the researcher as an additional sanity check to ensure accurate linking of pupil data with behavioral and/or neuroimaging data prior to analysis. Unsurprisingly, with more complex event message strings comes more burden on the researcher to devise an accurate and reliable solution to handle their specific event message configuration, likely leading to inefficient one-off solutions for particular datasets. `eyeris` meets this common demand by providing a carefully crafted regular expression implementation designed to handle a wide variety of generic cases (***Box 3***).

Conceptually, there are numerous ways that a researcher might want to extract their trial epochs for downstream analysis. In the simplest scenario, the user may be interested in testing a hypothesis about the tonic pupil size in the 1-second period just before the onset of the stimulus. In this case, the user would need to provide the string pattern and fixed start/stop duration of the epoch they would like to extract. For example, `c(-1,0)` for the 1-second just before the onset of the trial specifies a start/stop time array using the matched event message strings as the reference point (i.e., time=`0`).

4. Interactive visual reports and a BIDS-like data scheme enable intuitive quality control and transparent data reporting.

To date, most tools for preparing pupillometry data lack suitable features for data quality control up to the individual trial level. While some packages have integrated helpful plotting functions (e.g., ***Forbes, 2020***; ***Hershman et al., 2019***; ***Mittner, 2020***; ***Tsukahara, 2020***), these approaches only get the user part of the way there, as the onus falls on the user to aggregate such plots in a meaningful way for quality control. Moreover, this significant hurdle could be an arduous task even for researchers with programming experience. *We strongly recommend that researchers carefully inspect the data at each stage of the preprocessing pipeline before aggregating trials and/or running inferential statistics*, as is common in other human neuroscience data modalities such as EEG with the EEGLAB interactive viewer (***Delorme and Makeig, 2004***), M/EEG with MNE-Python (***Gramfort et al., 2013***), fMRI with fMRIPrep (***Esteban et al., 2019***), and structural MRI with MRIQC (***Esteban et al., 2017***; ***Provins et al., 2023***) visual reports.

Box 3. Tips & Tricks

Extracting pupil data from trial epochs with varying durations.

Users may also be interested in extracting the pupil time series for each stimulus/probe from start-to-finish. To illustrate, if each probe was displayed for a variable duration – perhaps because the participant's response designates the end of that trial – then the user cannot rely on a fixed-duration epoch extraction method. To demonstrate the power and flexibility of our implementation for this case in `eyeris`, take, for instance, an expansion of the researcher's event message tags for the probe onsets we discussed in the main text above and suppose that the researcher also included event message tags to signal the end of any given epoch, such as: `PROBE_END_32_PIANO`. With `eyeris`, the user can specify their starting boundary to be `PROBE_START_{trial}_{stimulus}` and ending boundary to be `PROBE_END_{trial}_{stimulus}`, which tells `eyeris` to grab all samples between each of these boundaries, and additionally, to create new metadata columns in the tidy output data frame containing the third and fourth extracted values in between the underscore separator with column names of `trial` and `stimulus`, respectively. See ***Function:*** *`epoch()`* for additional details.

Given the lack of a generic plug-and-play quality control solutions to date – in addition to the amount of effort required to implement a such solution – we suspect that many published pupillometry studies may not have raw data readily inspected end-to-end before and after preprocessing and running statistics. We therefore provide users with visualization tools of the pupil time series before *and* after each preprocessing operation for both an **entire** block/task *and* every trial-level epoch (i.e., as delineated by event messages/epochs). With this key feature, we hope to lower the barrier required for researchers to visually inspect entire recording sessions for anomalies in an efficient manner, as is necessary to ensure reproducible research results.

For the above reasons, we made visual reporting the default behavior in `eyeris` (***Figure 2***). Similar to the goals put forth by the team behind fMRIPrep, we anticipate that automatically generating diagnostic reports per participant will (i) maximize shareability and transparency between researchers, (ii) make this data surfing processing less cumbersome and more efficient and hence increase the probability that thorough data surfing occurs before submitting data to statistical tests, (iii) increase statistical power of future pupillometry investigations by enabling researchers to purge poor data at the trial level without excluding entire runs or participants altogether, and (iv) contribute to best practices in open and reproducible science by recording all preprocessing decisions and settings for any given dataset that makes its way through `eyeris` by default (***Esteban et al., 2020***, ***2019***; ***Poldrack et al., 2019***).

Importantly, all derived data and reports from the foregoing preprocessing workflows are automatically sorted into a BIDS-like format (***Gorgolewski et al., 2016***), facilitating separation of source (i.e., original or raw) data from derived (i.e., preprocessed, trimmed, epoched, etc.) data, in a directory structure that is intuitive to navigate, automatically organized, and therefore essentially ready to share on an appropriate data sharing outlet/repository like OSF, dryad, and/or GitHub (see ***Box 4***).

Discussion

The bells and whistles detailed throughout this manuscript and the documentation website[5] for the `eyeris` R package aim to standardize the way pupillometry data are preprocessed and used in downstream analysis. To this end, we designed the architecture underlying `eyeris` to support the building of a robust suite of pupillometry tools analogous to the current offerings for fMRI analysts with fMRIPrep (***Esteban et al., 2020***, ***2019***). At its core, `eyeris` enables standardized and

Box 4. BIDS-like derivatives data structure

```
eyeris/
|- derivatives/
   |- sub-001/
      |- eye/
      |  |- sub-001_task-ret_run-01_desc-timeseries_pupil.csv
      |  |- sub-001_task-ret_run-01_epoch-probe_desc-preproc_pupil.csv
      |- source/
      |  |- figures/
      |     |- run-01/
      |        |- epoch_probe/
      |        |  |- run-01_PROBE_22_1.png
      |        |  |- run-01_PROBE_22_2.png
      |        |  |- run-01_PROBE_22_3.png
      |        |  |- ...
      |        |- run-01_fig-1_desc-histogram.jpg
      |        |- run-01_fig-1_desc-timeseries.jpg
      |- sub-001_epoch-probe_run-01.html
      |- sub-001.html
```

flexible preprocessing of the pupil assay.

Moreover, `eyeris` was designed with FAIR[6] principles in mind. To fulfill this promise, `eyeris` uses a built-in protocol to carry out and manage the record keeping of the entire preprocessing pipeline on a raw dataset (see ***Figure 1***). This protocol is more or less a "grammar", where each operation is a "verb" that can be chained one after the other (see ***Box 7***, which draws inspiration from familiar `tidyverse` "pipe" workflows; ***Wickham et al., 2019***). This approach allows us to take advantage of syntactic sugar such that one could take a raw pupillometry dataset and within seconds obtain (*i*) a step-by-step record of all preprocessing steps (and parameters used), (*ii*) how each step sequentially impacted the raw signal, (*iii*) default diagnostic reports that organize relevant preprocessing metadata for any particular dataset passed through `eyeris`, and (*iv*) an intuitive gallery viewing functionality for efficiently scrolling through hundreds-to-thousands of raw data epochs for quality control prior to running inferential statistics (***Figure 2***).

To this end, our hope is that enabling access to intuitive, interactive visual inspection tools to efficiently assess trial/epoch-level data quality – *at scale* – will support researchers in their efforts to perform reproducible and rigorous pupillometry research in a more accessible manner than afforded by extant tools.

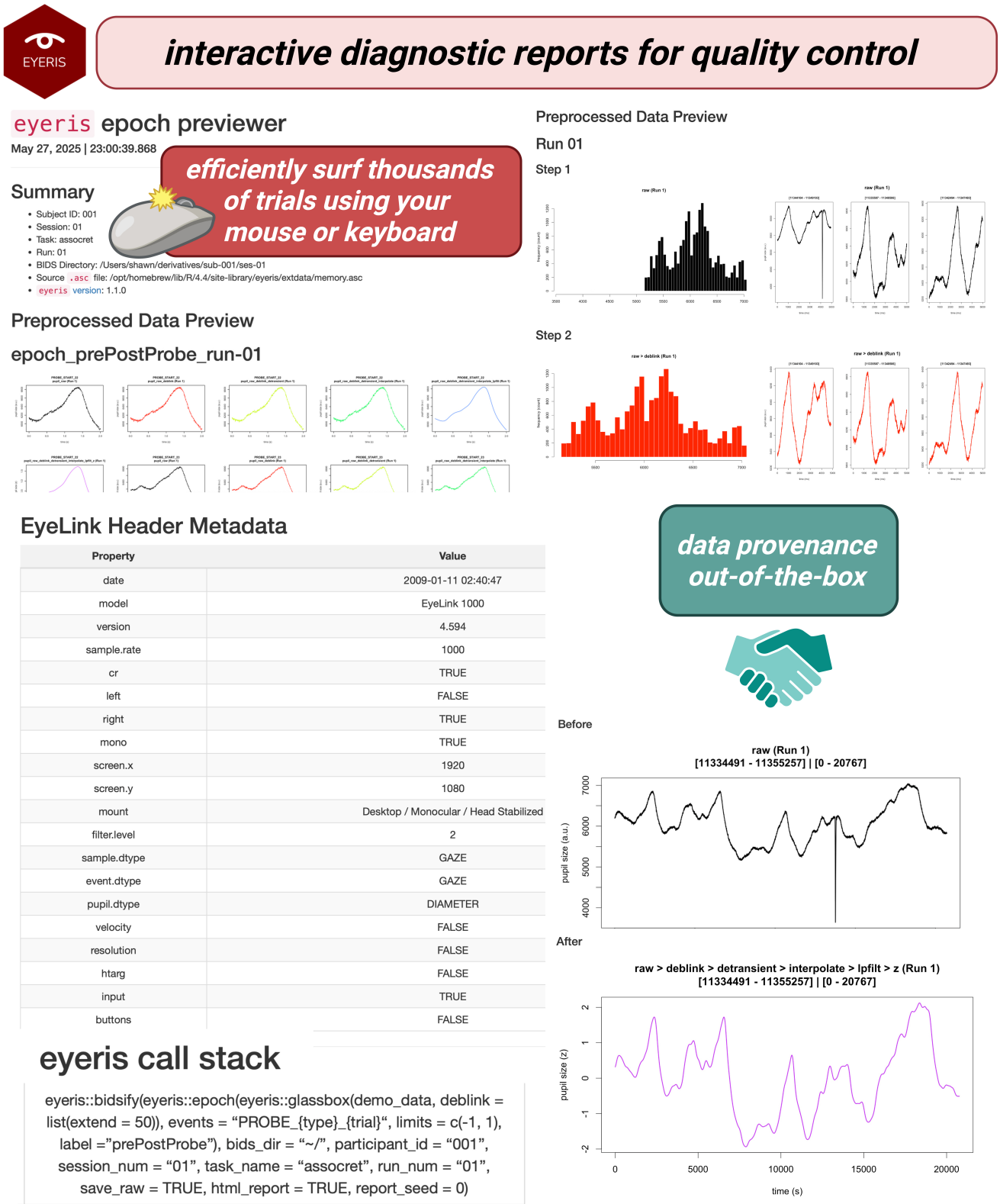


Figure 2. Overview of data and signal processing provenance provided out-of-the-box. `eyeris` provides automatically generated summary reports with diagnostic plots, recording metadata, and all preprocessing operations/parameters – and the resulting effects of each operation on the pupil time series – for every file/run for every participant. `eyeris` reports facilitate quality control by enabling surfing through thousands of trials worth of pupil data efficiently and interactively. Furthermore, these reports provide data provenance out-of-the-box and can be shared alongside any publication to support best practices in open science and reproducibility. Created in BioRender. ***Schwartz*** (***2025c***) https://BioRender.com/j2h2hzs.

Box 5. *eyeris* R package installation options

```
# (a) to install the latest stable release from CRAN:
install.packages("eyeris")

# (b) using Posit's pak approach to install packages interactively:
# install.packages("pak")
pak::pak("eyeris")

# (c) to install the bleeding edge development version:
# install.packages("devtools")
devtools::install_github("shawntz/eyeris")
```

Methods

eyeris R package

The `eyeris` R package is available on both CRAN and GitHub (see ***Code availability***). ***Box 5*** describes the three recommended methods for installing `eyeris`.

Entry-point function: `glassbox()`

(`eyeris`' recommended "glass-box" pipeline)

Glass-box pipeline implementation and usage

`eyeris` is built on a fully transparent workflow which can be executed all at once (using the `glassbox()` function) or individually in a chained pipeline command (see ***Box 1***). Either of these approaches provides users with the advantage of being certain that their pupil data are being processed correctly. In the following, we demonstrate the core arguments of the `glassbox()` function to demonstrate both its flexibility and its syntactic simplicity.

Arguments

`file`	An SR Research EyeLink `.asc` file generated by the official EyeLink `edf2asc` command.[7]
`interactive_preview`	A logical flag to indicate whether to run the `glassbox` pipeline autonomously all the way through or to interactively provide a visualization after each pipeline step, where you must also indicate "(y)es" or "(n)o" to either proceed or cancel the current `glassbox` pipeline operation (set to `FALSE` by default).
`preview_n`	The number of random example "epochs" to generate for previewing the effect of each preprocessing step on the pupil time series.
`preview_duration`	Time (in seconds) of each randomly selected preview.
`preview_window`	The start and stop raw timestamps used to subset the preprocessed data from each step of the `eyeris` workflow for visualization (set to `NULL` by default, which means randomly selected epochs as defined by `preview_n` and `preview_duration` will be plotted). To override the random epochs, the user must set `preview_window` here to a vector containing the relative start and stop times (e.g., `c(5000, 6000)` to indicate the raw data from 5-6 seconds for data recorded at $1000\,\mathrm{Hz}$). Note that the start/stop

Box 6. Overriding default parameters in the `eyeris::glassbox()`

```
output <- glassbox(
  rawdata,
  confirm = FALSE,
  deblink = list(extend = 40),
  detransient = FALSE,
  lpfilt = list(plot_freqz = FALSE)
)
#> [  OK  ] - Running eyeris::load_asc()
#> [  OK  ] - Running eyeris::deblink()
#> [ SKIP ] - Skipping eyeris::detransient()
#> [  OK  ] - Running eyeris::interpolate()
#> [  OK  ] - Running eyeris::lpfilt()
#> [ SKIP ] - Skipping eyeris::detrend()
#> [  OK  ] - Running eyeris::zscore()
```

---

	time values indicated here relate to the raw index position of each pupil sample from 1 to n (which will need to be specified manually by the user depending on the sampling rate of the recording; i.e., 5000-6000 for the epoch positioned from 5-6 seconds after the start of the time series, sampled at $1000\,\mathrm{Hz}$).
`verbose`	A logical flag to indicate whether to print status messages to the console (set to `TRUE` by default). To suppress informational messages about the current preprocessing step and, therefore, run the `glassbox()` function silently, set to `FALSE`.
`...`	Additional arguments to override the default settings prescribed for each preprocessing step within the `glassbox()` function (see ***Box 6*** for an example).

Core functions called by the `eyeris` glass-box pipeline

In the following, we outline the modular functions utilized by the glass-box pipeline to clean and transform pupil data using `eyeris`.

Function: `load_asc()`

(Load and parse SR Research EyeLink `.asc` files)

https://shawnschwartz.com/eyeris/reference/load_asc.html

This function builds on the `eyelinker::read.asc()` function to parse messages and metadata within the EyeLink `.asc` file. After loading and additional processing, this function returns an S3 `eyeris` class for use in all subsequent `eyeris` pipeline steps and functions.

Arguments

`file`	An SR Research EyeLink `.asc` file generated by the official EyeLink `edf2asc` command.
`block`	Optional block number specification. The following are valid options:

- `"auto"` (default): Automatically handles multiple recording segments embedded within the same `.asc` file. We recommend using this default, as this is likely the safer choice than assuming a single-block recording.
  - Note, by "block" we mean distinct recording sessions (e.g., a study phase followed by a memory test in an episodic memory experiment will likely be two separate recordings (even if stored in the same recording file) – as these are two distinct periods of the task – and thus would be considered "two blocks".
  - As such, please be sure to double-check the recording structure of your experiment, as correctly identifying blocks here is necessary for accurate downstream operations (such as z-scoring, where you would want each block to be z-scored separately (*which is the default behavior in* `eyeris`).
- `NULL`: Omits block column. Suitable for single-block recordings.
- Numeric value: Manually sets block number on the basis of the value provided here.

Returns

An object of the S3 class `eyeris` with the following attributes:

1. `file`: Path to the original `.asc` file.
2. `timeseries`: A list of data frames (1 data frame per identified recording block per file) which contains the following columns:
   - `block`: block number
   - `time_orig`: raw tracker time (ms)
   - `eye_x`: eye position x-coordinate
   - `eye_y`: eye position y-coordinate
   - `eye`: which eye (Left or Right) the recorded data are sourced from
   - `hz`: tracker sampling rate (Hz)
   - `type`: whether source data were recorded using the diameter or area method
   - `pupil_raw`: raw recorded pupil source data in arbitrary units (a.u.)
3. `events`: Data frame of all event messages and their timestamps.
4. `blinks`: Data frame of all blink events.
5. `info`: Data frame of various metadata parsed from the file header.
6. `latest`: `eyeris` variable for tracking the pipeline run history.

Function: `deblink()`

(NA-pad blink events/missing data)

https://shawnschwartz.com/eyeris/reference/deblink.html

Deblinking (a.k.a. NA-padding) of time series data. The intended use of this method is to remove blink-related artifacts surrounding periods of missing data. For instance, when an individual blinks, there are usually rapid decreases followed by increases in pupil size, with a chunk of data missing in-between these 'spike'-looking events. The deblinking procedure here will NA-pad each missing data point by your specified number of ms.

Arguments

eyeris	An object of class `eyeris` derived from `load_asc()`.
extend	Either a single number indicating the number of milliseconds to pad forward/backward around each missing sample, or, a vector of length two indicating different numbers of milliseconds to pad forward/backward around each missing sample, in the format `c(backward, forward)`.

Returns

An `eyeris` object with a new column in `timeseries`: `pupil_raw_{...}_deblink`, where {…} holds the name(s) of each preceding preprocessing step.

Usage

This function is automatically called by `glassbox()` by default. If needed, customize the parameters for `deblink` by providing a parameter list (e.g., `glassbox(deblink = list(extend = c(40, 50)))`. Use `glassbox(deblink = FALSE)` to disable this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to manually customize the pipeline.

Function: `detransient()`

(Remove pupil samples that are physiologically unlikely)

https://shawnschwartz.com/eyeris/reference/detransient.html

The intended use of this method is to remove pupil samples that emerge more quickly than would be physiologically expected. This is accomplished by rejecting samples that exceed a "speed"-based threshold (i.e., the median absolute deviation (MAD) from sample-to-sample). This threshold is computed based on the constant `n`, which defaults to the value `16`.

Arguments

eyeris	An object of class `eyeris` derived from `load_asc()`.
n	A constant used to compute the MAD threshold.
mad_thresh	Default `NULL`. This parameter provides alternative options for handling edge cases where the computed properties here within `detransient()` mad_val and $median_speed$ are very small. For example, if $$mad_val = 0 \;\; and \;\; median_speed = 1,$$ then, with the default multiplier $n = 16$, $$mad_thresh = median_speed + (n \times mad_val) = 1 + (16 \times 0) = 1.$$ In this situation, any speed $p_i \geq 1$ would be flagged as a transient sample, which could be too sensitive. To reduce this sensitivity, two possible adjustments are available: 1. If $mad_thresh = 1$, the transient detection criterion is modified from $$p_i \geq mad_thresh$$ to $$p_i > mad_thresh.$$ 2. If mad_thresh is very small, the user may manually adjust the sensitivity by supplying an alternative threshold value here directly via this `mad_thresh` parameter.

Computed properties

`pupil_speed` Compute speed of pupil by approximating the derivative of x (pupil) with respect to y (time) using finite differences.

- Let $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ be two numeric vectors with $n \geq 2$; then, the finite differences are computed as:

$$\delta_i = \frac{x_{i+1} - x_i}{y_{i+1} - y_i} \quad i = 1, 2, \ldots, n-1.$$

- This produces an output vector $p = (p_1, p_2, \ldots, p_n)$ defined by:
  - For the first element: $p_1 = |\delta_1|$,
  - For the last element: $p_n = |\delta_{n-1}|$,
  - For the intermediate elements $(i = 2, 3, \ldots, n-1)$:

$$p_i = \max\{|\delta_{i-1}|,\ |\delta_i|\}.$$

`median_speed` The median of the computed `pupil_speed`:

$$median_speed = median(p)$$

`mad_val` The median absolute deviation (MAD) of `pupil_speed` from the median:

$$mad_val = median(|p - median_speed|)$$

`mad_thresh` A threshold computed from the median speed and the MAD, using a constant multiplier n (default value: 16):

$$mad_thresh = median_speed + (n \times mad_val)$$

Returns

An `eyeris` object with a new column in `timeseries`: `pupil_raw_{...}_detransient`, where $\{\ldots\}$ holds the name(s) of each preceding preprocessing step.

Usage

This function is automatically called by `glassbox()` by default. If needed, customize the parameters for `detransient` by providing a parameter list (e.g., `detransient = list(n = 16)`). Use `glassbox(detransient = FALSE)` to disable this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to manually customize the pipeline.

Function: `interpolate()`

(Interpolate missing pupil samples)

https://shawnschwartz.com/eyeris/reference/interpolate.html

Linear interpolation of time series data. The intended use of this method is to fill in missing pupil samples (NA) in the time series. This method uses the `na.approx()` function from the `zoo` package (***Zeileis and Grothendieck, 2005***), which implements linear interpolation using the `approx()` function from the `stats` package (***R Core Team, 2025***). Currently, the NAs at the beginning and end of the data are replaced with values at either end, respectively, using the `rule = 2` argument in the `approx()` function.

Arguments

`eyeris` An object of class `eyeris` derived from `load_asc()`.

`verbose` A logical flag to indicate whether to print informational messages to the console. Defaults to `TRUE`. Set to `FALSE` to suppress messages about the current processing step and run silently.

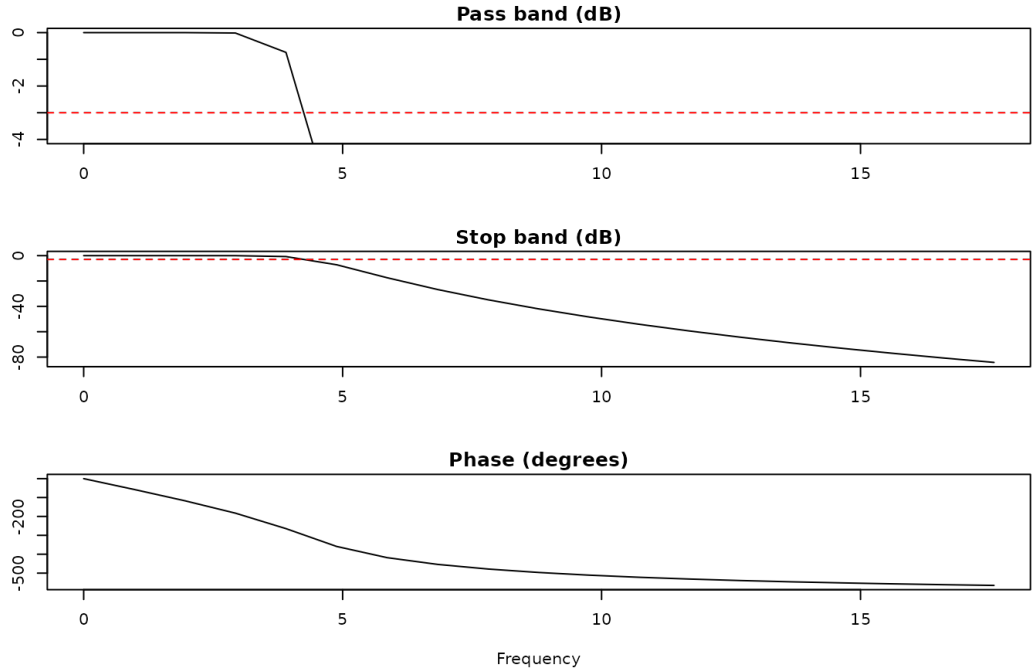


Figure 3. Frequency response for the low-pass filter; cutoff (4 Hz), stopping (8 Hz).

Returns

An `eyeris` object with a new column in `timeseries`: `pupil_raw_{...}_interpolate`, where {…} holds the name(s) of each preceding preprocessing step.

Usage

This function is automatically called by `glassbox()` by default. Use `glassbox(interpolate = FALSE)` to disable this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to manually customize the pipeline.

Function: `lpfilt()`

(Low-pass filtering of time series data)

https://shawnschwartz.com/eyeris/reference/lpfilt.html

The intended use of this method is for smoothing, although by specifying `wp` and `ws` differently one can achieve high-pass or band-pass filtering. **However, only low-pass filtering should be done on pupillometry data.**

Arguments

`eyeris`	An object of class `eyeris` derived from `load_asc()`.
`wp`	The end of the pass-band frequency in Hz (desired low-pass cutoff).
`ws`	The start of the stopband frequency in Hz (required low-pass cutoff).
`rp`	Required maximal ripple within the passband in dB.
`rs`	Required minimal attenuation within the stopband in dB.
`plot_freqz`	A logical flag for displaying the filter frequency response (see ***Figure 3*** for an example).

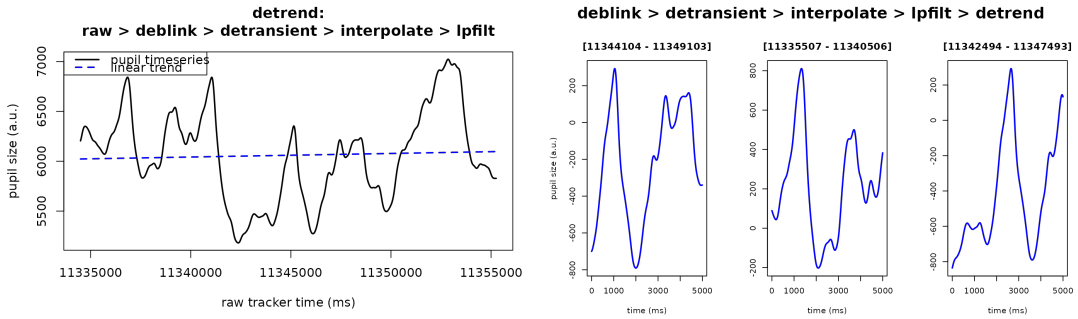


Figure 4. *Left,* example diagnostic plot of the entire recording's time series and the linear trend of those data provided by the `eyeris::detrend()` function. *Right,* example diagnostic plot of three randomly sampled 5-second epochs from the recording's time series *after* detrending the pupil time series.

Returns

An `eyeris` object with a new column in `timeseries`: `pupil_raw_{...}_lpfilt`, where {…} holds the name(s) of each preceding preprocessing step.

Usage

This function is automatically called by `glassbox()` by default. If needed, customize the parameters for `lpfilt` by providing a parameter list (e.g., `lpfilt = list(plot_freqz = FALSE)`). Use `glassbox(lpfilt = FALSE)` to disable this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to manually customize the pipeline.

Function: `detrend()`

(Detrend the pupil time series)

https://shawnschwartz.com/eyeris/reference/detrend.html

Linearly detrend pupil data by fitting a linear model of `pupil_data ~ time`, and return the fitted betas and the residuals (`pupil_data - fitted_values`).

Arguments

`eyeris`	An object of class `eyeris` derived from `load_asc()`.

Returns

An `eyeris` object with two new columns in `timeseries`: `detrend_fitted_betas` and `pupil_raw_{...}_detrend`, where {…} holds the name(s) of each preceding preprocessing step.

Usage

By default, the `detrend()` function is *disabled* within `glassbox()`.

Use `glassbox(detrend = TRUE)` to *enable* this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to manually customize the pipeline. See ***Figure 4*** for an example diagnostic plot provided by `eyeris` when invoking the `detrend()` function.

Function: `zscore()`

(z-score pupil time series data)

https://shawnschwartz.com/eyeris/reference/zscore.html

The intended use of this method is to scale the arbitrary units of the pupil size time series to have a mean of `0` and a standard deviation of `1`. This is accomplished by mean centering the data points and then dividing them by their standard deviation (that is, z-scoring the data, similar to `base::scale()`). Opting to z-score your pupil data helps with trial-level and between-subjects analyses where arbitrary units of pupil size recorded by the tracker do not scale across participants and therefore make analyses that depend on data from more than one participant difficult to interpret.

Arguments

`eyeris`	An object of class `eyeris` derived from `load_asc()`.

Returns

An `eyeris` object with a new column in `timeseries`: `pupil_raw_{...}_z`, where {...} holds the name(s) of each preceding preprocessing step.

Usage

This function is automatically called by `glassbox()` by default. Use `glassbox(zscore = FALSE)` to disable this step as needed. Users should prefer using `glassbox()` rather than invoking this function directly unless they have a specific reason to customize the pipeline manually.

In general, it is common to z-score pupil data within any given participant, and furthermore, z-score that participant's data as a function of block number (for tasks/experiments where participants complete more than one block of trials) to account for potential time-on-task effects across task/experiment blocks.

As such, if you use the eyeris package as intended, you should NOT need to specify any groups for the participant/block-level situations described above. This is because `eyeris` is designed to preprocess a single block of pupil data for a single participant, one at a time. Therefore, when you later merge all of the preprocessed data from `eyeris`, each individual, preprocessed block of data for each participant will have already been independently scaled from the others.

Additionally, if you intend to compare the mean z-scored pupil size between task conditions, such as memory successes vs. memory failures, then do NOT set your behavioral outcome (i.e., success/failure) variable as a grouping variable within your analysis. If you do, you will consequently obtain a mean pupil size of 0 and standard deviation of 1 within each group (since the scaled pupil size would be calculated on the time series from each outcome variable group separately). Instead, you should compute the z-score on the entire pupil time series (before epoching the data) and then split and take the mean of the z-scored time series as a function of condition variable.

Additional functions for epoching, exporting & visualizing pupil data

Built-in methods to conveniently extract tidy trial-based epochs with optional baseline correction, as well as to export BIDS-style derivatives and diagnostic HTML reports with interactive plot galleries to efficiently browse tens of thousands of trials of your pupil size data at each preprocessing step.

Function: `epoch()`

(epoch – and baseline correct – pupil data based on custom event message structures)
https://shawnschwartz.com/eyeris/reference/epoch.html

Intended to be used as the final preprocessing step. This function creates data epochs of either fixed or dynamic durations with respect to supplied `events` and time `limits`, and also includes an intuitive metadata parsing feature where additional trial data embedded within event messages can easily be identified and joined into the resulting epoched data frames. The three main *event-modes* that `eyeris` supports for epoching are: (*i*) a single string representing the event message to perform trial extraction around, using specified limits to center the epoch around or no limits (which then just grabs the data epochs between each subsequent event string of the same type); (*ii*) a vector containing both start and end event message strings – here, limits will be ignored and the duration of each trial epoch will be the number of samples between each matched start and end event message pair; and (*iii*) a list of 2 data frames that manually specify start/end event timestamp-message pairs to pull out of the raw time series data – here, it is required that each raw timestamp and event message be provided in the following format of a list of data frames and block number identifier:

- **start events data frame:** `data.frame(time = c(...), msg = c(...))`
- **end events data frame:** `data.frame(time = c(...), msg = c(...))`
- **block number integer:** e.g., `1`

Arguments

`eyeris` An object of class `eyeris` derived from `load_asc()`.

`events`

1. a single string representing the event message to perform trial extraction around, using specified `limits` to center the epoch around or no `limits` (which then just grabs the data epochs between each subsequent event string of the same type);
2. a vector containing both `start` and `end` event message strings – here, `limits` will be ignored and the duration of each trial epoch will be the number of samples between each matched `start` and `end` event message pair; or
3. a list of 2 data frames that manually specify start/end event timestamp-message pairs to pull out of the raw time series data – here, it is required that each raw timestamp and event message be provided in the following format:

```
list(
   # start events
   data.frame( time = c(...),  msg = c(...) ),
   # end events
   data.frame( time = c(...),  msg = c(...) ),
   # block number
   1
)
```

where the first data.frame indicates the `start` event timestamp and message string pairs, and the second data.frame indicates the `end` event timestamp and message string pairs. Additionally, manual epoching only words with 1 block at a time for event-modes `2` and `3` herein; thus, please be sure

to explicitly indicate the block number in your input list.

For event-modes `1` and `2`, the way in which you pass in the event message string must conform to a standardized protocol so that `eyeris` knows how to find your events and (optionally) parse any included metadata into the tidy epoch data outputs. You have two primary choices:

(a) **specify a string followed by a wildcard:** (e.g., `"PROBE_START*"`), which will match any messages that have "PROBE_START ..." (... referring to potential metadata, such as trial number, stim file, etc.); or

(b) **specify a string using the `eyeris` syntax:** (e.g., `"PROBE_{type}_{trial}"`), which will match the messages that follow a structure like this "PROBE_START_1" and "PROBE_STOP_1", and generate two additional metadata columns: `type` and `trial`, which would contain the following values based on these two example strings: `type`: `("START", "STOP")`, and `trial:(1, 1)`.

`limits` A vector of 2 values (start, end) in seconds, indicating where trial extraction should occur centered around any given `start` message string in the `events` parameter.

`label` An (optional) string you can provide to customize the name of the resulting `eyeris` class object containing the epoched data frame. If left as `NULL` (default), then list item will be called `epoch_xyz`, where `xyz` will be a sanitized version of the original `start` event string you provided for matching. If you choose to specify a `label` here, then the resulting list object name will take the form: `epoch_label`. **Warning: if no `label` is specified and there are no event message strings to sanitize, then you may obtain a strange-looking epoch list element in your output object (e.g., `$epoch_`, or `$epoch_nana`, etc.). The data should still be accessible within the nested lists, however, to avoid ambiguous list objects, we recommend you provide an `epoch` label here to be safe.**

`calc_baseline` A flag indicated whether to perform baseline correction. Note, setting `calc_baseline` to `TRUE` alone will only compute the baseline period, but will not apply it to the preprocessed time series unless `apply_baseline` is also set to `TRUE`.

`apply_baseline` A flag indicating whether to apply the calculated baseline to the pupil time series. The baseline correction will be applied to the pupil from the latest preprocessing step.

`baseline_type` Whether to perform *subtractive* (`sub`) or *divisive* (`div`) baseline correction. Defaults to `sub`.

`baseline_events` Similar to `events`, for `baseline_events` you can supply either

1. a single string representing the event message to center the baseline calculation around, as indicated by `baseline_period`; or
2. a single vector containing both a `start` and an `end` event message string – here, `baseline_period` will be ignored and the duration of each baseline period that the mean will be

calculated on will be the number of samples between each matched `start` and `end` event message pair, as opposed to a specified fixed duration (as described in 1).

Please note, providing a list of trial-level start/end message pairs (like in the `events` parameter) to manually indicate unique start/end chunks for baselining is currently unsupported. Though, we intend to add this feature in a later version of `eyeris`, given it likely will not be a heavily utilized/in-demand feature.

`baseline_period` A vector of 2 values (start, end) in seconds, indicating the window of data that will be used to perform the baseline correction, which will be centered around the single "start" message string provided in `baseline_events`. Again, `baseline_period` will be ignored if both a "start" *and* "end" message string are provided to the `baseline_events` argument.

`hz` Data sampling rate. If not specified, will use the value contained within the tracker's metadata.

`verbose` A logical flag to indicate whether to print informational messages to the console. Defaults to `TRUE`. Set to `FALSE` to suppress messages about the current processing step and run silently.

Returns

An updated `eyeris` object with data frames containing the epoched data (`epoch_xyz`).

Here, the first data.frame indicates the start event timestamp and message string pairs, and the second data.frame indicates the end event timestamp and message string pairs. Additionally, manual epoching only works with one block at a time for event modes 2 and 3; thus, users must be sure to explicitly indicate the block number in the input list as shown above.

For event modes 1 and 2, the way in which you pass in the event message string must conform to a standardized protocol so that `eyeris` knows how to find these events and (optionally) parse any included metadata into the tidy epoch data outputs. Users have two primary choices: either (a) specify a string followed by a wildcard expression (e.g., `PROBE_START`), which will match any messages that have `PROBE_START ...` (*where ... refers to potential metadata, such as trial number, stimulus file, etc.*); or (b) specify a string using the `eyeris` syntax: (e.g., `PROBE_{type}_{trial}`), which will match the messages that follow a structure like this `PROBE_START_1` and `PROBE_STOP_1`, and generate two additional metadata columns: `type` and `trial`, which would contain the following values based on these two example strings:

- **type column values:** `("START", "STOP")`
- **trial column values:** `(1, 1)`

Function: `bidsify()`

(Save out pupil time series data in a bids-like structure)
https://shawnschwartz.com/eyeris/reference/bidsify.html

This method provides a structured way to save out pupil data in a BIDS-like structure. The method saves out epoched data as well as the raw pupil time series, and formats the directory and filename structures based on the metadata you provide.

Arguments

`eyeris`	An object of class `eyeris` derived from `load_asc()`.
`save_all`	Logical flag indicating whether all epochs are to be saved vs only a subset of them. Defaults to `TRUE`.
`epochs_list`	List of epochs to be saved. Defaults to `NULL`.
`merge_epochs`	Logical flag indicating whether epochs should be saved as one file or as separate files. Defaults to `FALSE` (no merge).
`bids_dir`	Base BIDS-derivatives directory.
`participant_id`	BIDS subject ID.
`session_num`	BIDS session ID.
`task_name`	BIDS task ID.
`run_num`	BIDS run ID. For single files without blocks (i.e., runs), `run_num` specifies which run this file represents. However, for files with multiple recording blocks embedded within the *same* `.asc` file, this parameter is ignored and the blocks are automatically numbered as runs (block 1 = run-01, block 2 = run-02, etc.) in the order they appeared/were recorded.
`merge_runs`	Logical flag indicating whether multiple runs (either from multiple recording blocks existing within the *same* `.asc` file (see above) or manually specified) should be combined into a single output file. When `TRUE`, adds a 'run' column to identify the source run. Defaults to FALSE (i.e., separate files per block/run, the standard BIDS-like behavior).
`save_raw`	Logical flag indicating whether to save raw source pupil data in addition to preprocessed, epoched data. Defaults to `TRUE`.
`html_report`	Logical flag indicating whether to save out the `eyeris` preprocessing summary report as an HTML file. Defaults to `FALSE`.
`pdf_report`	Logical flag indicating whether to save out the `eyeris` preprocessing summary report as a PDF file. Note that a valid TeX distribution must already be installed. Defaults to `FALSE`.
`report_seed`	Random seed for the plots that will appear in the report. Defaults to 0. See ***Function:*** `plot()` for a more detailed description.
`report_epoch_grouping_var_col`	String name of grouping column to use for epoch-by-epoch diagnostic plots in an interactive rendered HTML report. The column name must exist (i.e., it must be a custom grouping variable name set within the metadata template of your `epoch()` call). Defaults to `"matched_event"`, which all epoched data frames have as a valid column name. To disable these epoch-level diagnostic plots, set to `NULL`.
`verbose`	A flag to indicate whether to print informational messages to the console. Defaults to `TRUE`. Set to `FALSE` to suppress messages about the current processing step and run silently.

Returns

Invisibly returns `NULL`. Called for its side effects.

Notes

In the future, we intend for this function to save the data in an official BIDS format for eye tracking data (see the proposal currently under review here). However, at this time, this function instead takes a more BIDS-inspired approach to organizing the output files for preprocessed pupil data.

Function: `plot()`

(Plot preprocessed pupil data from `eyeris`)

https://shawnschwartz.com/eyeris/reference/plot.eyeris.html

S3 plotting method for objects of class `eyeris`. Plots a single-panel time series for a subset of the pupil time series at each preprocessing step. The intended use of this function is to provide a simple method for qualitatively assessing the consequences of the preprocessing recipe and parameters on the raw pupillary signal.

Arguments

`x`	An object of class `eyeris` derived from `load_asc()`.
`...`	Additional arguments to be passed to `plot`.
`steps`	Which steps to plot; defaults to `all` (i.e., plot all steps). Otherwise, pass a vector containing the index of the step(s) you want to plot, with index `1` being the original raw pupil time series.
`num_previews`	Number of random example "epochs" to generate for previewing the effect of each preprocessing step on the pupil time series.
`preview_duration`	Time in seconds of each randomly selected preview.
`preview_window`	The start and stop timestamps used to subset the preprocessed data from each step of the `eyeris` workflow for visualization. Defaults to `NULL`, meaning random epochs as defined by `num_previews` and `preview_duration` will be plotted. To override the random epochs, set `preview_window` here to a vector with relative start and stop times (e.g., `c(5000, 6000)` to indicate the raw data from 5-6 seconds on data that were recorded at $1000\,\mathrm{Hz}$). Note that the start/stop time values indicated here relate to the raw index position of each pupil sample from 1 to `n` (which will need to be specified manually by the user depending on the sampling rate of the recording; i.e., 5000-6000 for the epoch positioned from 5-6 seconds after the start of the time series, sampled at $1000\,\mathrm{Hz}$).
`seed`	Random seed for current plotting session. Leave `NULL` to select `num_previews` number of random preview "epochs" (of `preview_duration`) each time. Otherwise, choose any seed-integer as you would normally select for `base::set.seed()`, and you will be able to continue replotting the same random example pupil epochs each time – which is helpful when adjusting parameters within and across `eyeris` workflow steps.
`block`	For multi-block recordings, specifies which block to plot. Defaults to 1. When a single `.asc` data file contains multiple recording blocks, this parameter determines which

	block time series to visualize. Must be a positive integer not exceeding the total number of blocks in the recording file.
`plot_distributions`	Logical flag to indicate whether to plot both diagnostic pupil time series and accompanying histograms of the pupil samples at each processing step. Defaults to `FALSE`.

Returns

No return value; iteratively plots a subset of the pupil time series from each preprocessing step run.

Advanced features for developers

We provide developers the flexibility to extend upon `eyeris`' default offerings by writing their own custom pipeline operations that can integrate seamlessly with `eyeris` primitives (see ***Box 7***).

Function: `pipeline_handler()`

(Build generic extensions for your *eyeris* pipeline)

https://shawnschwartz.com/eyeris/reference/pipeline_handler.html

`pipeline_handler` enables flexible integration of custom data processing functions into the `eyeris` pipeline.

Arguments

`eyeris`	An object of class `eyeris` containing time series data in a list of data frames (one per block), various metadata collected by the tracker, and `eyeris` specific pointers for tracking the preprocessing history for that specific instance of the `eyeris` object.
`operation`	The name of the function to apply to the time series data. This custom function should accept a data frame `x`, a string `prev_op` (i.e., the name of the previous pupil column, which you do not need to supply as a literal string, as this is inferred from the `latest` pointer within the `eyeris` object), and any custom parameters you would like.
`new_suffix`	A character string indicating the suffix you would like to be appended to the name of the previous operation's column, which will be used for the new column name in the updated preprocessed data frame(s).
`...`	Additional (optional) arguments passed to the `operation` method.

Returns

An updated `eyeris` object with the new column added to the `timeseries` data frame and the `latest` pointer updated to the name of the most recently added column plus all previous columns (i.e., the history "trace" of preprocessing steps from start to present). For instance: `pupil_raw` → `pupil_raw_deblink` → `pupil_raw_deblink_detransient` → `pupil_...`

Notes

Following the `eyeris` protocol also ensures:

- all operations follow a predictable structure, and
- that new pupil data columns based on previous operations in the chain are able to be dynamically constructed within the core time series data frame.

Box 7. Deconstructing `glassbox()`: Using the flexible `eyeris` primitives

```
system.file("extdata", "memory.asc", package = "eyeris") |>
  load_asc(block = "auto") |>
  deblink(extend = 50) |>
  detransient(n = 16) |>
  interpolate() |>
  lpfilt(wp = 4, ws = 8, rp = 1, rs = 35, plot_freqz = TRUE) |>
  # eyeris::detrend() |>  # optional (read docs before enabling)
  zscore()
```

Building your own custom pipeline extensions

One key strength of the preprocessing framework within `eyeris` is its modularity. While we encourage most users to use the `glassbox()` function for simplicity and reproducibility, advanced users can create custom preprocessing steps that seamlessly integrate into the pipeline.

How the pipeline works

Under the hood, each preprocessing function in `eyeris` is a wrapper around a core operation that gets tracked, versioned, and stored using the `pipeline_handler()`. Custom pipeline steps must conform to the `eyeris` protocol for maximum compatibility with the downstream functions we provide (see ***Function:*** *`pipeline_handler()`*). Below, we demonstrate the flexibility of `eyeris` for developers to build custom extensions as needed. Please note that in the example that follows, we are *not* recommending that one performs this particular statistical transformation on their pupil data. Rather, this is just for demonstration purposes only. See ***Appendix 1*** for all source code for this demo.

Creating a custom extension for `eyeris`

Suppose you want to write a new `eyeris` extension function called `winsorize()` to apply winsorization[8] to extreme pupil values.

Step 1: Write the core operation function logic

This function should accept a data frame `x`, a string `prev_op` (i.e., the name of the previous pupil column), and any custom parameters.

Step 2: Create the wrapper using the *pipeline_handler()* function

The `eyeris::pipeline_handler()` enables your function to automatically:

- track your function within the `eyeris` list object's `params` field,
- append a new column to each block within the `timeseries` list of data frames, and
- update the object's `latest` pointer.

To ensure compatibility with `eyeris`, we impose the following requirements for any custom extension functions:

1. The first argument is always the `eyeris` class object.
2. Second is the name of the function (i.e., the one you created in *step 1* above).
3. Third is the internal label you want `eyeris` to refer to (i.e., the one for the column name, plots, etc.).
4. Fourth position++ are the parameters you created after `prev_op` when creating the function `winsorize_pupil` in *step 1* above.

More information about customizing `eyeris` can be found in the online package documentation[9].

Acknowledgments

Development of this software library was supported by an agility grant from the Stanford University Wu Tsai Human Performance Alliance (to Prof. Anthony D Wagner & Shawn T Schwartz); the Stanford University Wu Tsai Neurosciences Institute Center for Mind, Brain, Computation and Technology (to Shawn T Schwartz & Alice M Xue); a Stanford University Ric Weiland Graduate Fellowship in the Humanities & Sciences (to Shawn T Schwartz); the McKnight Brain Research Foundation Clinical Translational Research Scholarship in Cognitive Aging and Age-Related Memory Loss; the American Brain Foundation; the American Academy of Neurology (to Haopei Yang); National Science Foundation Graduate Research Fellowship DGE-2146755 (to Alice M Xue); and National Institutes of Health Grants R01-AG065255 (to Prof. Anthony D Wagner) and R01-AG079345 (to Prof. Patrick L Purdon). The funders had no role in study design, data collection, and interpretation, or the decision to submit the work for publication. Furthermore, we are grateful to Ryan Yan for providing helpful comments on this manuscript. We are also grateful to Anthony Wagner for supporting the development of this software. We are thankful to Gustavo Santiago-Reyes for debugging and contributing to an earlier version of the source code, as well as to Natalie Wang for beta-testing the software during its early stages of development. We additionally thank Brian Knutson for helpful theoretical and practical discussions about pupillometry and arousal mechanisms. Lastly, we extend our gratitude to the past and present members of the Stanford Memory Lab[10] who have influenced the work discussed herein.

Notes

[1] See ***Fink et al.*** (***2024***) for a detailed table further outlining documented associations between cognitive processes and pupil size.

[2] Throughout this manuscript, "preprocessing" means the specific prescription of steps necessary to derive transformed pupil data ready for visualization and analysis from the raw pupil data recorded from a participant using an eye tracking device.

[3] https://www.sr-research.com

[4] See ***4. Interactive visual reports and a BIDS-like data scheme enable intuitive quality control and transparent data reporting***.

[5] https://shawnschwartz.com/eyeris

[6] Findability, Accessibility, Interoperability, and Reusability (see ***Hahn et al., 2024***; ***Lazar, 2023***; ***Wilkinson et al., 2016***).

[7] At this time, `eyeris` only supports EyeLink acquired pupillometry datasets, given its prevalence in this research space. However, we acknowledge that accessibility to EyeLink hardware is limited due to equipment costs, and thus are now working on building in a porting mechanism for other eye tracking data files. To achieve this goal, we will first create a generic solution that can take a `CSV` file of recording data and will specify a mapping between key column names in that file with the expected labels utilized by `eyeris`. A more ambitious goal is native support for proprietary formats from other trackers; this step will require external contributions and requests from users with access to these data formats that we could use to implement the compatibility layer, which we will evaluate on a case-by-case basis via submissions to the issues tab on GitHub (see ***Code availability*** for more information on how to access the source code and/or contribute directly to the development of `eyeris`).

[8] https://en.wikipedia.org/wiki/Winsorizing

[9] https://shawnschwartz.com/eyeris/articles/custom-extensions.html

[10] https://memorylab.stanford.edu

Additional information

Competing interests

We declare that there are no conflicts of interest with respect to the authorship or publication of this article.

Author contributions

Shawn T Schwartz, Conceptualization, Formal Analysis, Methodology, Project Administration, Resources, Software, Supervision, Validation, Visualization, Writing - Original Draft Preparation, Writing - Review & Editing; **Haopei Yang**, Data Curation, Resources, Validation, Writing - Review & Editing; **Alice M Xue**, Data Curation, Resources, Validation, Writing - Review & Editing; **Mingjian He**, Formal Analysis, Methodology, Resources, Supervision, Validation, Writing - Review & Editing.

Author ORCIDs

Shawn T Schwartz https://orcid.org/0000-0001-6444-8451
Haopei Yang https://orcid.org/0000-0002-1207-3245
Alice M Xue https://orcid.org/0009-0006-3176-8786
Mingjian He https://orcid.org/0000-0002-6688-8693

Code availability

Please use the `issues` tab on GitHub to note bugs, comments, suggestions, and/or feedback.

CRAN https://cran.r-project.org/package=eyeris
GitHub repository https://github.com/shawntz/eyeris
GitHub issues https://github.com/shawntz/eyeris/issues
Package documentation https://shawnschwartz.com/eyeris/

References

Alamia A, VanRullen R, Pasqualotto E, Mouraux A, Zenon A. Pupil-Linked Arousal Responds to Unconscious Surprisal. The Journal of Neuroscience. 2019; 39(27):5369–5376. https://doi.org/10.1523/jneurosci.3010-18.2019.

Arachchige CN, Prendergast LA. Confidence intervals for median absolute deviations. Communications in Statistics-Simulation and Computation. 2024; p. 1–10. https://doi.org/10.1080/03610918.2024.2376198.

Aston-Jones G, Cohen JD. An Integrative Theory Of Locus Coeruleus-Norepinephrine Function: Adaptive Gain and Optimal Performance. Annual Review of Neuroscience. 2005; 28(1):403–450. https://doi.org/10.1146/annurev.neuro.28.061604.135709.

Aston-Jones G, Cohen JD. An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. Annu Rev Neurosci. 2005; 28(1):403–450. https://doi.org/10.1146/annurev.neuro.28.061604.135709.

Beatty J. Phasic Not Tonic Pupillary Responses Vary With Auditory Vigilance Performance. Psychophysiology. 1982; 19(2):167–172. https://doi.org/10.1111/j.1469-8986.1982.tb02540.x.

Betts MJ, Kirilina E, Otaduy MC, Ivanov D, Acosta-Cabronero J, Callaghan MF, Lambert C, Cardenas-Blanco A, Pine K, Passamonti L, et al. Locus coeruleus imaging as a biomarker for noradrenergic dysfunction in neurodegenerative diseases. Brain. 2019; 142(9):2558–2571. https://doi.org/10.1093/brain/awz193.

van Boxtel G, gsignal: Signal processing; 2021. https://doi.org/10.32614/CRAN.package.gsignal.

Braak H, Del Tredici K. Alzheimer's pathogenesis: is there neuron-to-neuron propagation? Acta neuropathologica. 2011; 121:589–595. https://doi.org/10.1007/s00401-011-0825-z.

Bradley MM, Miccoli L, Escrig MA, Lang PJ. The pupil as a measure of emotional arousal and autonomic activation. Psychophysiology. 2008; 45(4):602–607. https://doi.org/10.1111/j.1469-8986.2008.00654.x.

Bradshaw JL. Background light intensity and the pupillary response in a reaction time task. Psychonomic Science. 1969; 14(6):271–272. https://doi.org/10.3758/BF03329118.

Chen A, Wong C, Sharif B, Peruma A. Exploring Code Comprehension in Scientific Programming: Preliminary Insights from Research Scientists. arXiv preprint arXiv:250110037. 2025; https://doi.org/10.48550/arXiv.2501.10037.

Clewett D, Gasser C, Davachi L. Pupil-linked arousal signals track the temporal organization of events in memory. Nature Communications. 2020; 11(1):4007. https://doi.org/10.1038/s41467-020-17851-9.

Clewett D, Gasser C, Davachi L. Pupil-linked arousal signals track the temporal organization of events in memory. Nature communications. 2020; 11(1):4007. https://doi.org/10.1038/s41467-020-17851-9.

Delorme A, Makeig S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. Journal of neuroscience methods. 2004; 134(1):9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009.

Einhäuser W, Stout J, Koch C, Carter O. Pupil dilation reflects perceptual selection and predicts subsequent stability in perceptual rivalry. Proceedings of the National Academy of Sciences. 2008; 105(5):1704–1709. https://doi.org/10.1073/pnas.0707727105.

El Haj M. Pupillometry as tool to assess cognitive and affective processing in aging. Brain Disorders. 2024; p. 100129. https://doi.org/10.1016/j.dscb.2024.100129.

Eldar E, Cohen JD, Niv Y. The effects of neural gain on attention and learning. Nature Neuroscience. 2013; 16(8):1146–1153. https://doi.org/10.1038/nn.3428.

Esteban O, Birman D, Schaer M, Koyejo OO, Poldrack RA, Gorgolewski KJ. MRIQC: Advancing the automatic prediction of image quality in MRI from unseen sites. PloS one. 2017; 12(9):e0184661. https://doi.org/10.1371/journal.pone.0184661.

Esteban O, Ciric R, Finc K, Blair RW, Markiewicz CJ, Moodie CA, Kent JD, Goncalves M, DuPre E, Gomez DEP, Ye Z, Salo T, Valabregue R, Amlien IK, Liem F, Jacoby N, Stojić H, Cieslak M, Urchs S, Halchenko YO, et al. Analysis of task-based functional MRI data preprocessed with fMRIPrep. Nature Protocols. 2020; 15(7):2186–2202. https://doi.org/10.1038/s41596-020-0327-3.

Esteban O, Markiewicz CJ, Blair RW, Moodie CA, Isik AI, Erramuzpe A, Kent JD, Goncalves M, DuPre E, Snyder M, Oya H, Ghosh SS, Wright J, Durnez J, Poldrack RA, Gorgolewski KJ. fMRIPrep: a robust preprocessing pipeline for functional MRI. Nature Methods. 2019; 16(1):111–116. https://doi.org/10.1038/s41592-018-0235-4.

Fink L, Simola J, Tavano A, Lange E, Wallot S, Laeng B. From pre-processing to advanced dynamic modeling of pupil data. Behavior Research Methods. 2024; 56(3):1376–1412. https://doi.org/10.3758/s13428-023-02098-1.

Forbes SH. PupillometryR: An R package for preparing and analysing pupillometry data. Journal of Open Source Software. 2020; 5(50):2285. https://doi.org/10.21105/joss.02285.

Frank MC, Braginsky M, Cachia J, Coles NA, Hardwicke TE, Hawkins RD, Mathur MB, Williams R. Replication. In: *Experimentology: An Open Science Approach to Experimental Psychology Methods* Stanford University; 2025.https://doi.org/10.25936/3JP6-5M50, also published by MIT Press, ISBN 978-0-262-55256-1.

Gee JWd, Knapen T, Donner TH. Decision-related pupil dilation reflects upcoming choice and individual bias. Proceedings of the National Academy of Sciences. 2014; 111(5):E618–E625. https://doi.org/10.1073/pnas.1317557111.

Geller J, Winn MB, Mahr T, Mirman D. GazeR: A Package for Processing Gaze Position and Pupil Size Data. Behavior Research Methods. 2020; 52(5):2232–2255. https://doi.org/10.3758/s13428-020-01374-8.

Gelman A, Loken E. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. Department of Statistics, Columbia University. 2013; 348(1-17):3. https://osf.io/n3axs.

Gilzenrat MS, Nieuwenhuis S, Jepma M, Cohen JD. Pupil diameter tracks changes in control state predicted by the adaptive gain theory of locus coeruleus function. Cognitive, Affective, & Behavioral Neuroscience. 2010; 10(2):252–269. https://doi.org/10.3758/CABN.10.2.252.

Gorgolewski KJ, Auer T, Calhoun VD, Craddock RC, Das S, Duff EP, Flandin G, Ghosh SS, Glatard T, Halchenko YO, Handwerker DA, Hanke M, Keator D, Li X, Michael Z, Maumet C, Nichols BN, Nichols TE, Pellman J, Poline JB, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. Scientific Data. 2016; 3(1):160044. https://doi.org/10.1038/sdata.2016.44.

Gorgolewski KJ, Poldrack RA. A practical guide for improving transparency and reproducibility in neuroimaging research. PLoS biology. 2016; 14(7):e1002506. https://doi.org/10.1371/journal.pbio.1002506.

Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L, et al. MEG and EEG data analysis with MNE-Python. Frontiers in Neuroinformatics. 2013; 7:267. https://doi.org/10.3389/fnins.2013.00267.

Groot JM, Boayue NM, Csifcsák G, Boekel W, Huster R, Forstmann BU, Mittner M. Probing the neural signature of mind wandering with simultaneous fMRI-EEG and pupillometry. NeuroImage. 2021; 224:117412. https://doi.org/10.1016/j.neuroimage.2020.117412.

Haab O. Vortrag gesellschaft der ärzte in zürich am 21. November 1885. Korrespondenzblatt für Schweizer Ärzte. 1886; 16:153.

Hahn S, Allgaier N, Garavan H. Methods for Analyzing Large Neuroimaging Datasets. Neuromethods. 2024; p. 255–287. https://doi.org/10.1007/978-1-0716-4260-3_12.

Hampel FR. The influence curve and its role in robust estimation. Journal of the american statistical association. 1974; 69(346):383–393.

Hampel FR. Contributions to the theory of robust estimation. University of California, Berkeley; 1968.

Hardwicke TE, Bohn M, MacDonald K, Hembacher E, Nuijten MB, Peloquin BN, DeMayo BE, Long B, Yoon EJ, Frank MC. Analytic reproducibility in articles receiving open data badges at the journal Psychological Science: An observational study. Royal Society open science. 2021; 8(1):201494. https://doi.org/10.1098/rsos.201494.

Hardwicke TE, Mathur MB, MacDonald K, Nilsonne G, Banks GC, Kidwell MC, Hofelich Mohr A, Clayton E, Yoon EJ, Henry Tessler M, et al. Data availability, reusability, and analytic reproducibility: Evaluating the impact of a mandatory open data policy at the journal Cognition. Royal Society open science. 2018; 5(8):180448. https://doi.org/10.1098/rsos.180448.

Hardwicke TE, Thibault RT, Kosie JE, Wallach JD, Kidwell MC, Ioannidis JP. Estimating the prevalence of transparency and reproducibility-related research practices in psychology (2014–2017). Perspectives on Psychological Science. 2022; 17(1):239–251. https://doi.org/10.1177/1745691620979806.

Hatton L. The T-experiments: errors in scientific software. Quality of Numerical Software: Assessment and Enhancement. 1997; p. 12–31. https://doi.org/10.1007/978-1-5041-2940-4_2.

Hatton L, Roberts A. How accurate is scientific software? IEEE Transactions on Software Engineering. 1994; 20(10):785–797. https://doi.org/10.1109/32.328993.

He M, Heindel WC, Nassar MR, Siefert EM, Festa EK. Age-related changes in the functional integrity of the phasic alerting system: a pupillometric investigation. Neurobiology of aging. 2020; 91:136–147. https://doi.org/10.1016/j.neurobiolaging.2020.02.025.

Hershman R, Henik A, Cohen N. CHAP: Open-source software for processing and analyzing pupillometry data. Behavior Research Methods. 2019; 51(3):1059–1074. https://doi.org/10.3758/s13428-018-01190-1.

Hershman R, Milshtein D, Henik A. Processing and Analyzing of Pupillometry Data. In: *Modern Pupillometry, Cognition, Neuroscience, and Practical Applications* Springer; 2024.p. 431–466. https://doi.org/10.1007/978-3-031-54896-3_15.

Holper S, Watson R, Yassi N. Tau as a biomarker of neurodegeneration. International Journal of Molecular Sciences. 2022; 23(13):7307. https://doi.org/10.3390/ijms23137307.

Hopstaken JF, van der Linden D, Bakker AB, Kompier MAJ. A multifaceted investigation of the link between mental fatigue and task disengagement. Psychophysiology. 2015; 52(3):305–315. https://doi.org/10.1111/psyp.12339.

Huang R, Clewett D. The Locus Coeruleus: Where Cognitive and Emotional Processing Meet the Eye. In: *Modern Pupillometry, Cognition, Neuroscience, and Practical Applications* Springer; 2024.p. 3–75. https://doi.org/10.1007/978-3-031-54896-3_1.

Ioannidis JP. Why most published research findings are false. PLoS medicine. 2005; 2(8):e124. https://doi.org/10.1371/journal.pmed.0020124.

Jepma M, Nieuwenhuis S. Pupil Diameter Predicts Changes in the Exploration–Exploitation Trade-off: Evidence for the Adaptive Gain Theory. Journal of Cognitive Neuroscience. 2011; 23(7):1587–1596. https://doi.org/10.1162/jocn.2010.21548.

Johansson R, Parnamets P, Bjernestedt A, Johansson M. Pupil dilation tracks the dynamics of mnemonic interference resolution. Scientific reports. 2018; 8(1):4826. https://doi.org/10.1038/s41598-018-23297-3.

Joshi S, Gold JI. Pupil size as a window on neural substrates of cognition. Trends in cognitive sciences. 2020; 24(6):466–480. https://doi.org/10.1016/j.tics.2020.03.005.

Joshi S, Li Y, Kalwani RM, Gold JI. Relationships between pupil diameter and neuronal activity in the locus coeruleus, colliculi, and cingulate cortex. Neuron. 2016; 89(1):221–234. https://doi.org/10.1016/j.neuron.2015.11.028.

Kahneman D, Beatty J. Pupil Diameter and Load on Memory. Science. 1966; 154(3756):1583–1585. https://doi.org/10.1126/science.154.3756.1583.

Kawaguchi K, Clery S, Pourriahi P, Seillier L, Haefner RM, Nienborg H. Differentiating between Models of Perceptual Decision Making Using Pupil Size Inferred Confidence. The Journal of Neuroscience. 2018; 38(41):8874–8888. https://doi.org/10.1523/jneurosci.0735-18.2018.

Keene PA, deBettencourt MT, Awh E, Vogel EK. Pupillometry signatures of sustained attention and working memory. Attention, Perception, & Psychophysics. 2022; 84(8):2472–2482. https://doi.org/10.3758/s13414-022-02557-5.

Kim J, Lee J, Jun SB, Sung JE. Pupillometry as a window to detect cognitive aging in the brain. Biomedical Engineering Letters. 2024; 14(1):91–101. https://doi.org/10.1007/s13534-023-00315-6.

Kinley I, Levy Y. PuPl: an open-source tool for processing pupillometry data. Behavior Research Methods. 2021; p. 1–24. https://doi.org/10.3758/s13428-021-01717-z.

Knapen T, Gee JWd, Brascamp J, Nuiten S, Hoppenbrouwers S, Theeuwes J. Cognitive and Ocular Factors Jointly Determine Pupil Responses under Equiluminance. PLoS ONE. 2016; 11(5):e0155574. https://doi.org/10.1371/journal.pone.0155574.

Kovacs M, Varga MA, Dianovics D, Poldrack RA, Aczel B. Opening the black box of article retractions: exploring the causes and consequences of data management errors. Royal Society Open Science. 2024; 11(12):240844. https://doi.org/10.1098/rsos.240844.

Kramer SE, Lorens A, Coninx F, Zekveld AA, Piotrowska A, Skarzynski H. Processing load during listening: The influence of task characteristics on the pupil response. Language and Cognitive Processes. 2013; 28(4):426–442. https://doi.org/10.1080/01690965.2011.642267.

Kret ME, Sjak-Shie EE. Preprocessing pupil size data: Guidelines and code. Behavior Research Methods. 2019; 51(3):1336–1342. https://doi.org/10.3758/s13428-018-1075-y.

Kristjansson SD, Stern JA, Brown TB, Rohrbaugh JW. Detecting phasic lapses in alertness using pupillometric measures. Applied Ergonomics. 2009; 40(6):978–986. https://doi.org/10.1016/j.apergo.2009.04.007.

Laeng B, Mathôt S. Methodological Aspects of Pupillometry. In: *Modern Pupillometry: Cognition, Neuroscience, and Practical Applications* Springer, Cham; 2024.p. 375–400. https://doi.org/10.1007/978-3-031-54896-3_13.

Lazar NA. Functional neuroimaging in the era of Big Data and Open Science: A modern overview. Wiley Interdisciplinary Reviews: Computational Statistics. 2023; 15(5). https://doi.org/10.1002/wics.1609.

MacKenzie-Graham AJ, Van Horn JD, Woods RP, Crawford KL, Toga AW. Provenance in neuroimaging. Neuroimage. 2008; 42(1):178–195. https://doi.org/10.1016/j.neuroimage.2008.04.186.

Madore KP, Khazenzon AM, Backes CW, Jiang J, Uncapher MR, Norcia AM, Wagner AD. Memory failure predicted by attention lapsing and media multitasking. Nature. 2020; 587(7832):87–91. https://doi.org/10.1038/s41586-020-2870-z.

Madore KP, Wagner AD. Readiness to remember: predicting variability in episodic memory. Trends in Cognitive Sciences. 2022; 26(8):707–723. https://doi.org/10.1016/j.tics.2022.05.006.

Martin JT, Whittaker AH, Johnston SJ. Pupillometry and the vigilance decrement: Task-evoked but not baseline pupil measures reflect declining performance in visual vigilance tasks. European Journal of Neuroscience. 2022; 55(3):778–799. https://doi.org/10.1111/ejn.15585.

Massar SA, Lim J, Sasmita K, Chee MW. Rewards boost sustained attention through higher effort: A value-based decision making approach. Biological Psychology. 2016; 120:21–27. https://doi.org/10.1016/j.biopsycho.2016.07.019.

Mathôt S. Pupillometry: Psychology, physiology, and function. Journal of cognition. 2018; 1(1):16. https://doi.org/10.5334/joc.18.

Mathôt S, Aarts E, Verhage M, Veenvliet J, Dolan C, van der Sluis S, A simple way to reconstruct pupil size during eye blinks; 2013.

Mathôt S, Vilotijević A. Methods in cognitive pupillometry: Design, preprocessing, and statistical analysis. Behavior Research Methods. 2023; 55(6):3055–3077. https://doi.org/10.3758/s13428-022-01957-7.

McGinley M, David S, McCormick D. Cortical Membrane Potential Signature of Optimal States for Sensory Signal Detection. Neuron. 2015; 87(1):179–192. https://doi.org/10.1016/j.neuron.2015.05.038.

Megemont M, McBurney-Lin J, Yang H. Pupil diameter is not an accurate real-time readout of locus coeruleus activity. Elife. 2022; 11:e70510. https://doi.org/10.7554/eLife.70510.

Meissner SN, Bächinger M, Kikkert S, Imhof J, Missura S, Carro Dominguez M, Wenderoth N. Self-regulating arousal via pupil-based biofeedback. Nature Human Behaviour. 2024; 8(1):43–62. https://doi.org/10.1038/s41562-023-01729-z.

Merali Z. Error: why scientific programming does not compute. Nature. 2010; 467(7317):775–777. https://doi.org/10.1038/467775a.

Miller AL, Gross MP, Unsworth N. Individual differences in working memory capacity and long-term memory: The influence of intensity of attention to items at encoding as measured by pupil dilation. Journal of Memory and Language. 2019; 104:25–42. https://doi.org/10.1016/j.jml.2018.09.005.

Miller AL, Unsworth N. Variation in Attention at Encoding: Insights From Pupillometry and Eye Gaze Fixations. Journal of Experimental Psychology: Learning, Memory, and Cognition. 2020; 46(12):2277–2294. https://doi.org/10.1037/xlm0000797.

Miller G, A scientist's nightmare: software problem leads to five retractions. American Association for the Advancement of Science; 2006. https://doi.org/10.1126/science.314.5807.1856.

Mittner M, pypillometry: A Python package for pupillometric analyses. Open Journals; 2020. https://doi.org/10.21105/joss.02348.

Murphy PR, O'connell RG, O'sullivan M, Robertson IH, Balsters JH. Pupil diameter covaries with BOLD activity in human locus coeruleus. Human brain mapping. 2014; 35(8):4140–4154. https://doi.org/10.1002/hbm.22466.

Murphy PR, Robertson IH, Balsters JH, O'connell RG. Pupillometry and P3 index the locus coeruleus–noradrenergic arousal function in humans. Psychophysiology. 2011; 48(11):1532–1543. https://doi.org/10.1111/j.1469-8986.2011.01226.x.

Oliva M, Anikin A. Pupil dilation reflects the time course of emotion recognition in human vocalizations. Scientific reports. 2018; 8(1):4871. https://doi.org/10.1038/s41598-018-23265-x.

Open Science Collaboration. Estimating the reproducibility of psychological science. Science. 2015; 349(6251):aac4716. https://doi.org/10.1126/science.aac4716.

Peysakhovich V, Vachon F, Dehais F. The impact of luminance on tonic and phasic pupillary responses to sustained cognitive load. International Journal of Psychophysiology. 2017; 112:40–45. https://doi.org/10.1016/j.ijpsycho.2016.12.003.

Piquado T, Isaacowitz D, Wingfield A. Pupillometry as a measure of cognitive effort in younger and older adults. Psychophysiology. 2010; 47(3):560–569. https://doi.org/10.1111/j.1469-8986.2009.00947.x.

Poe GR, Foote S, Eschenko O, Johansen JP, Bouret S, Aston-Jones G, Harley CW, Manahan-Vaughan D, Weinshenker D, Valentino R, et al. Locus coeruleus: a new look at the blue spot. Nature Reviews Neuroscience. 2020; 21(11):644–659. https://doi.org/10.1038/s41583-020-0360-9.

Poldrack RA, Baker CI, Durnez J, Gorgolewski KJ, Matthews PM, Munafo MR, Nichols TE, Poline JB, Vul E, Yarkoni T. Scanning the horizon: towards transparent and reproducible neuroimaging research. Nature reviews neuroscience. 2017; 18(2):115–126. https://doi.org/10.1038/nrn.2016.167.

Poldrack RA, Gorgolewski KJ, Varoquaux G. Computational and informatic advances for reproducible data analysis in neuroimaging. Annual Review of Biomedical Data Science. 2019; 2(1):119–138. https://doi.org/10.1146/annurev-biodatasci-072018-021237.

Provins C, MacNicol E, Seeley SH, Hagmann P, Esteban O. Quality control in functional MRI studies with MRIQC and fMRIPrep. Frontiers in Neuroimaging. 2023; 1:1073734. https://doi.org/10.3389/fnimg.2022.1073734.

R Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria; 2025, https://www.R-project.org/.

Reimer J, Froudarakis E, Cadwell C, Yatsenko D, Denfield G, Tolias A. Pupil Fluctuations Track Fast Switching of Cortical States during Quiet Wakefulness. Neuron. 2014; 84(2):355–362. https://doi.org/10.1016/j.neuron.2014.09.033.

Robison MK, Trost JM, Schor D, Gibson BS, Healey MK. Pupillary correlates of individual differences in long-term memory. Psychonomic Bulletin & Review. 2022; 29(4):1355–1366. https://doi.org/10.3758/s13423-022-02081-5.

Robison MK, Unsworth N. Pupillometry tracks fluctuations in working memory performance. Attention, Perception, & Psychophysics. 2019; 81(2):407–419. https://doi.org/10.3758/s13414-018-1618-4.

Rousseeuw PJ, Croux C. Alternatives to the median absolute deviation. Journal of the American Statistical association. 1993; 88(424):1273–1283.

Rousseeuw PJ, Hampel FR, Ronchetti EM, Stahel WA, Robust statistics: the approach based on influence functions. Wiley, New York; 1986.

Samuels ER, Szabadi E. Functional neuroanatomy of the noradrenergic locus coeruleus: its roles in the regulation of arousal and autonomic function part I: principles of functional organisation. Current neuropharmacology. 2008; 6(3):235–253. https://www.ingentaconnect.com/content/ben/cn/2008/00000006/00000003/art00005.

Schwartz ST, Nguyen KK, Yang H, deBettencourt MT, Tran TT, Madore KP, Wagner AD. Real-time reorienting of preparatory sustained attention lapsing during episodic retrieval using closed-loop pupillometry. In: *31st Annual Meeting of the Cognitive Neuroscience Society* Toronto, Ontario, Canada; 2024. https://drive.google.com/file/d/1axi_MGuLCd6OG_XRAuM7_BratDAzmxcR/view?usp=sharing, conference presentation, https://drive.google.com/file/d/1rNRkWZMQCZfSoYS-JHWL2wB3Kh0UX4VZ/view.

Schwartz ST. eyeris: Flexible, Extensible, & Reproducible Processing of Pupil Data; 2025, https://doi.org/10.32614/CRAN.package.eyeris, r package version 1.1.0.

Schwartz ST, eyeris R package; 2025. Created in BioRender. https://BioRender.com/vkahl3u.

Schwartz ST, eyeris reports; 2025. Created in BioRender. https://BioRender.com/j2h2hzs.

Schwartz ST, Yang H, Xue AM, Wagner AD. Attending to Remember: Recent Advances in Methods and Theory. Current Directions in Psychological Science. 2025; https://doi.org/10.1177/09637214251339452.

Segal J. Scientists and software engineers: A tale of two cultures. In: Buckley J, Rooksby J, Bednarik R, editors. *Proceedings of the 20th Annual Meeting of the Psychology of Programming Interest Group (PPIG 2008)* Lancaster University, Lancaster, UK; 2008. https://oro.open.ac.uk/17671/4/PPIG_08Segal.pdf.

Simmons JP, Nelson LD, Simonsohn U. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. Psychological science. 2011; 22(11):1359–1366. https://doi.org/10.1177/0956797611417632.

Smallwood J, Brown KS, Baird B, Mrazek MD, Franklin MS, Schooler JW. Insulation for Daydreams: A Role for Tonic Norepinephrine in the Facilitation of Internally Guided Thought. PLOS ONE. 2012 04; 7(4):1–5. https://doi.org/10.1371/journal.pone.0033706.

Šoški'c A, Styles SJ, Kappenman ES, Kovi'c V. Garden of forking paths in ERP research–Effects of varying preprocessing and analysis steps in an N400 experiment. Psychophysiology. 2024; 61(10):e14628. https://doi.org/10.1111/psyp.14628.

Spires-Jones TL, Hyman BT. The intersection of amyloid beta and tau at synapses in Alzheimer's disease. Neuron. 2014; 82(4):756–771. https://doi.org/10.1016/j.neuron.2014.05.004.

Steinhauer SR, Bradley MM, Siegle GJ, Roecklein KA, Dix A. Publication guidelines and recommendations for pupillary measurement in psychophysiological studies. Psychophysiology. 2022; 59(4):e14035. https://doi.org/10.1111/psyp.14035.

Steinhauer SR, Siegle GJ, Condray R, Pless M. Sympathetic and parasympathetic innervation of pupillary dilation during sustained processing. International journal of psychophysiology. 2004; 52(1):77–86. https://doi.org/10.1016/j.ijpsycho.2003.12.005.

Strauch C. The forgotten wave of early pupillometry research. Trends in Neurosciences. 2024; 47(8):571–572. https://doi.org/10.1016/j.tins.2024.06.002.

Strauch C, Wang CA, Einhäuser W, Stigchel SVd, Naber M. Pupillometry as an integrated readout of distinct attentional networks. Trends in Neurosciences. 2022; 45(8):635–647. https://doi.org/10.1016/j.tins.2022.05.003.

Tsukahara JS, Harrison TL, Engle RW. The relationship between baseline pupil size and intelligence. Cognitive Psychology. 2016; 91:109–123. https://doi.org/10.1016/j.cogpsych.2016.10.001.

Tsukahara J, Pupillometry: An R package to preprocess pupil data (v0.8.1); 2020. https://github.com/dr-jt/pupillometry.

Unsworth N, Robison MK. Individual differences in the allocation of attention to items in working memory: Evidence from pupillometry. Psychonomic Bulletin & Review. 2015; 22(3):757–765. https://doi.org/10.3758/s13423-014-0747-6.

Unsworth N, Robison MK. Pupillary correlates of lapses of sustained attention. Cognitive, Affective, & Behavioral Neuroscience. 2016 Aug; 16(4):601–615. https://doi.org/10.3758/s13415-016-0417-4.

Unsworth N, Robison MK. The Importance of Arousal for Variation in Working Memory Capacity and Attention Control: A Latent Variable Pupillometry Study. Journal of Experimental Psychology: Learning, Memory, and Cognition. 2017; 43(12):1962–1987. https://doi.org/10.1037/xlm0000421.

Unsworth N, Robison MK. A locus coeruleus-norepinephrine account of individual differences in working memory capacity and attention control. Psychonomic Bulletin & Review. 2017; 24(4):1282–1311. https://doi.org/10.3758/s13423-016-1220-5.

Unsworth N, Robison MK. Tracking arousal state and mind wandering with pupillometry. Cognitive, Affective, & Behavioral Neuroscience. 2018 Aug; 18(4):638–664. https://doi.org/10.3758/s13415-018-0594-4.

Urai AE, Braun A, Donner TH. Pupil-linked arousal is driven by decision uncertainty and alters serial choice bias. Nature communications. 2017; 8(1):14637. https://doi.org/10.1038/ncomms14637.

van den Brink RL, Murphy PR, Nieuwenhuis S. Pupil Diameter Tracks Lapses of Attention. PLOS ONE. 2016 10; 11(10):1–16. https://doi.org/10.1371/journal.pone.0165274.

Van Gerven PW, Paas F, Van Merriënboer JJ, Schmidt HG. Memory load and the cognitive pupillary response in aging. Psychophysiology. 2004; 41(2):167–174. https://doi.org/10.1111/j.1469-8986.2003.00148.x.

van Rij J, Hendriks P, Rijn Hv, Baayen RH, Wood SN. Analyzing the Time Course of Pupillometric Data. Trends in Hearing. 2019; 23:2331216519832483. https://doi.org/10.1177/2331216519832483.

Van Slooten JC, Jahfari S, Knapen T, Theeuwes J. How pupil responses track value-based decision-making during and after reinforcement learning. PLoS computational biology. 2018; 14(11):e1006632. https://doi.org/10.1371/journal.pcbi.1006632.

Varoquaux G. Beyond computational reproducibility, let us aim for reusability. IEEE CIS Newsletter on Cognitive and Developmental Systems. 2016; 13(2). https://inria.hal.science/hal-01592368v1, what is Computational Reproducibility?

Verney SP, Granholm E, Marshall SP. Pupillary responses on the visual backward masking task reflect general cognitive ability. International journal of psychophysiology. 2004; 52(1):23–36. https://doi.org/10.1016/j.ijpsycho.2003.12.003.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, et al. Welcome to the Tidyverse. Journal of open source software. 2019; 4(43):1686. https://doi.org/10.21105/joss.01686.

Widmann A, Schröger E, Wetzel N. Emotion lies in the eye of the listener: Emotional arousal to novel sounds is reflected in the sympathetic contribution to the pupil dilation response and the P3. Biological psychology. 2018; 133:10–17. https://doi.org/10.1016/j.biopsycho.2018.01.010.

Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, Blomberg N, Boiten JW, Santos LBdS, Bourne PE, Bouwman J, Brookes AJ, Clark T, Crosas M, Dillo I, Dumon O, Edmunds S, Evelo CT, Finkers R, Gonzalez-Beltran A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016; 3(1):160018. https://doi.org/10.1038/sdata.2016.18.

Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, Haddock SH, Huff KD, Mitchell IM, Plumbley MD, et al. Best practices for scientific computing. PLoS biology. 2014; 12(1):e1001745. https://doi.org/10.1371/journal.pbio.1001745.

Wu J, Toporek A, Lin Q, Goldstein FC, Loring DW, Kelberman MA, Weinshenker D, Levey AI, Lah JJ, Qiu D. Probing locus coeruleus functional network in healthy aging and its association with Alzheimer's disease biomarkers using pupillometry. Alzheimer's Research & Therapy. 2025; 17(1):1–12. https://doi.org/10.1186/s13195-025-01701-1.

Zeileis A, Grothendieck G. zoo: S3 Infrastructure for Regular and Irregular Time Series. Journal of Statistical Software. 2005; 14(6):1–27. https://doi.org/10.18637/jss.v014.i06.

Zhao S, Bury G, Milne A, Chait M. Pupillometry as an Objective Measure of Sustained Attention in Young and Older Listeners. Trends in Hearing. 2019; 23:2331216519887815. https://doi.org/10.1177/2331216519887815.

Appendix 1

1) Write core operation function logic.

```
winsorize_pupil <- function(x, prev_op, lower = .01, upper = .99) {
  vec <- x[[prev_op]]
  q <- quantile(vec, probs = c(lower, upper), na.rm = TRUE)
  vec[vec < q[1]] <- q[1]
  vec[vec > q[2]] <- q[2]
  vec
}
```

2) Create wrapper using the *pipeline_handler()* function.

```
#' Winsorize pupil values
#'
#' Applies winsorization to extreme pupil values within
#' each block.
#'
#' @param eyeris An ‘eyeris‘ object created by [load_asc()].
#' @param lower Lower quantile threshold. Default is 0.01.
#' @param upper Upper quantile threshold. Default is 0.99.
#'
#' @return Updated ‘eyeris‘ object with new winsorized
#' pupil column.
winsorize <- function(eyeris, lower = .01, upper = .99) {
  pipeline_handler(
    eyeris,
    winsorize_pupil,
    "winsorize",
    lower = lower,
    upper = upper
  )
}
```

3) Invoke custom function in *eyeris* pipeline call.

```
system.file("extdata", "memory.asc", package = "eyeris") |>
  eyeris::load_asc(block = "auto") |>
  eyeris::deblink(extend = 50) |>
  winsorize()
```

EYERIS

EyeLink ASCII File

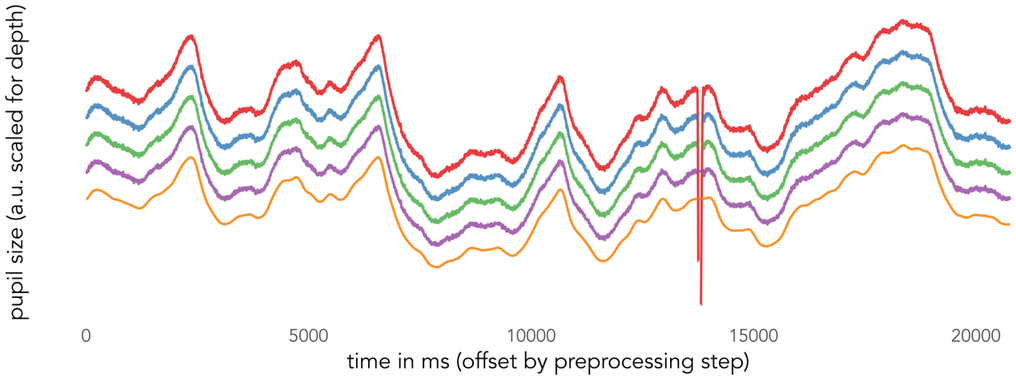
`load_asc()`

- source file path
- timeseries data frame
- events data frame
- blinks data frame
- metadata info data frame
- *eyeris* history
  - steps
  - parameters
- epoch_{name} data frame(s)

`default glassbox()` *or* `custom pipeline`

raw data → eyeris object → preproc data

progressive preprocessing across the core eyeris::glassbox() steps



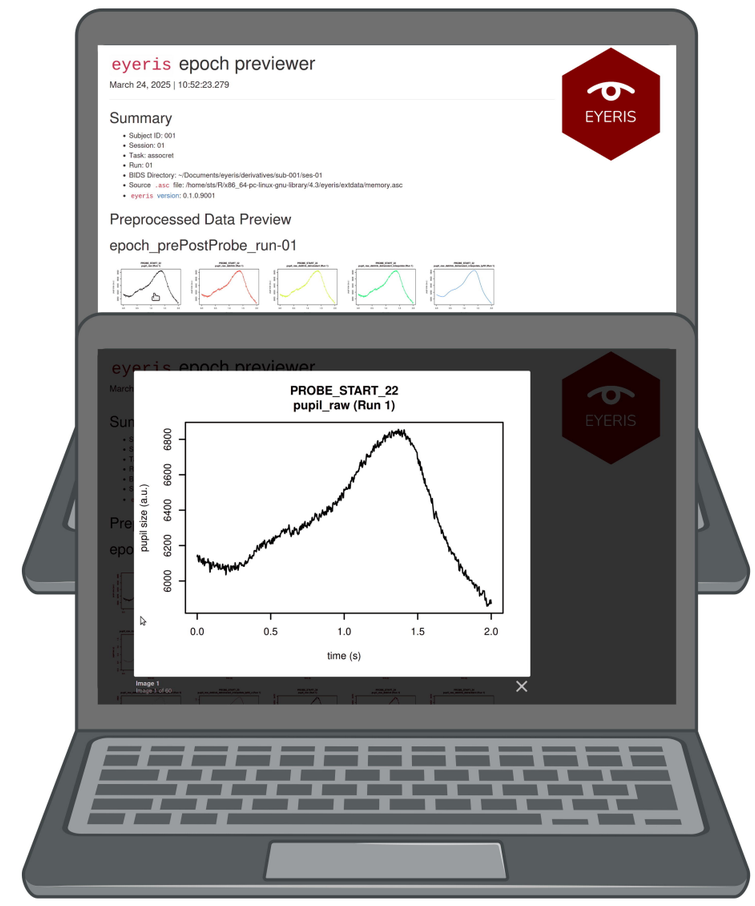
- raw
- raw > deblink
- raw > deblink > detransient
- raw > deblink > detransient > interpolate
- raw > deblink > detransient > interpolate > lpfilt

extracting trial epochs and BIDSifying pupil data in eyeris

```
demo_data <-
  eyelink_asc_demo_dataset()

demo_data |>
  eyeris::glassbox() |>
  eyeris::epoch(
    events = "PROBE_START_{trial}",
    limits = c(-1, 1),
    label  = "prePostProbe"
  ) |>
  eyeris::bidsify(
    bids_dir = "~/your/directory",
    participant_id = "001",
    session_num = "01",
    task_name = "assocret",
    run_num = "01",
    save_raw = TRUE,
    html_report = TRUE,
    report_seed = 0
  )
```

interactive reports for QC



interactive diagnostic reports for quality control

`eyeris` epoch previewer

May 27, 2025 | 23:00:39.868

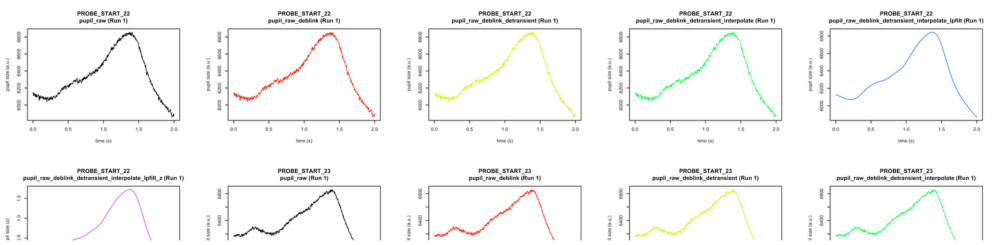
efficiently surf thousands of trials using your mouse or keyboard

Summary

- Subject ID: 001
- Session: 01
- Task: assocret
- Run: 01
- BIDS Directory: /Users/shawn/derivatives/sub-001/ses-01
- Source `.asc` file: /opt/homebrew/lib/R/4.4/site-library/eyeris/extdata/memory.asc
- `eyeris` version: 1.1.0

Preprocessed Data Preview

epoch_prePostProbe_run-01



EyeLink Header Metadata

Property	Value
date	2009-01-11 02:40:47
model	EyeLink 1000
version	4.594
sample.rate	1000
cr	TRUE
left	FALSE
right	TRUE
mono	TRUE
screen.x	1920
screen.y	1080
mount	Desktop / Monocular / Head Stabilized
filter.level	2
sample.dtype	GAZE
event.dtype	GAZE
pupil.dtype	DIAMETER
velocity	FALSE
resolution	FALSE
htarg	FALSE
input	TRUE
buttons	FALSE

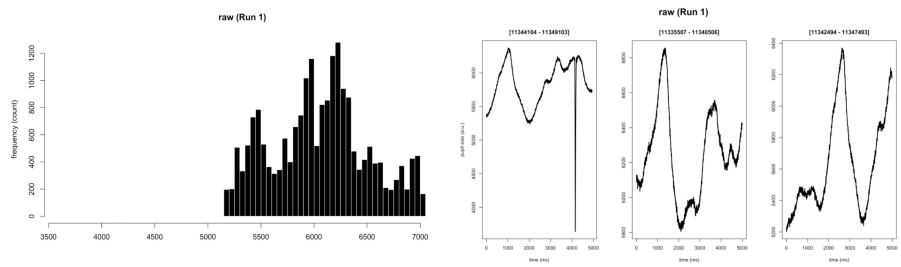
eyeris call stack

```
eyeris::bidsify(eyeris::epoch(eyeris::glassbox(demo_data, deblink =
list(extend = 50)), events = “PROBE_{type}_{trial}“, limits = c(-1, 1),
label =”prePostProbe”), bids_dir = “~/”, participant_id = “001”,
session_num = “01”, task_name = “assocret”, run_num = “01”,
save_raw = TRUE, html_report = TRUE, report_seed = 0)
```

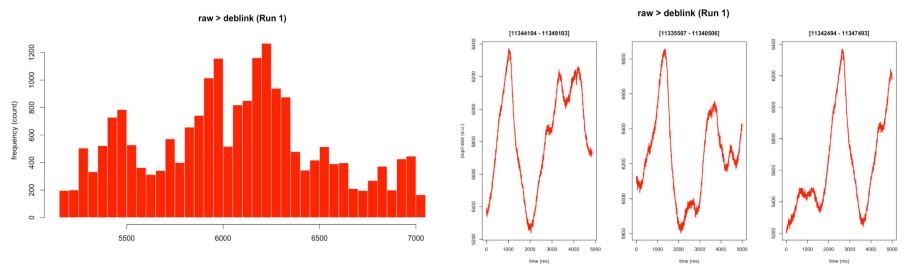
Preprocessed Data Preview

Run 01

Step 1

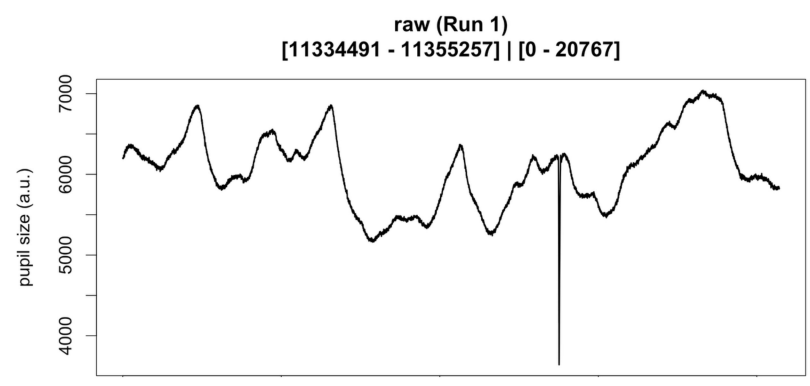


Step 2

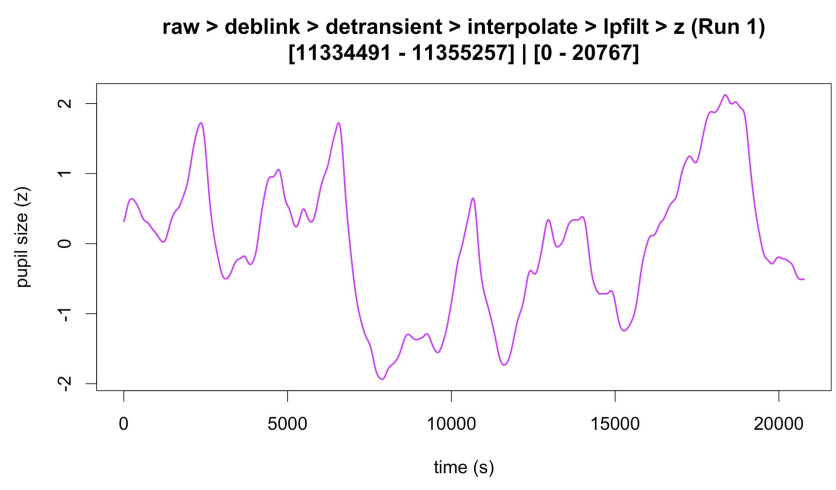


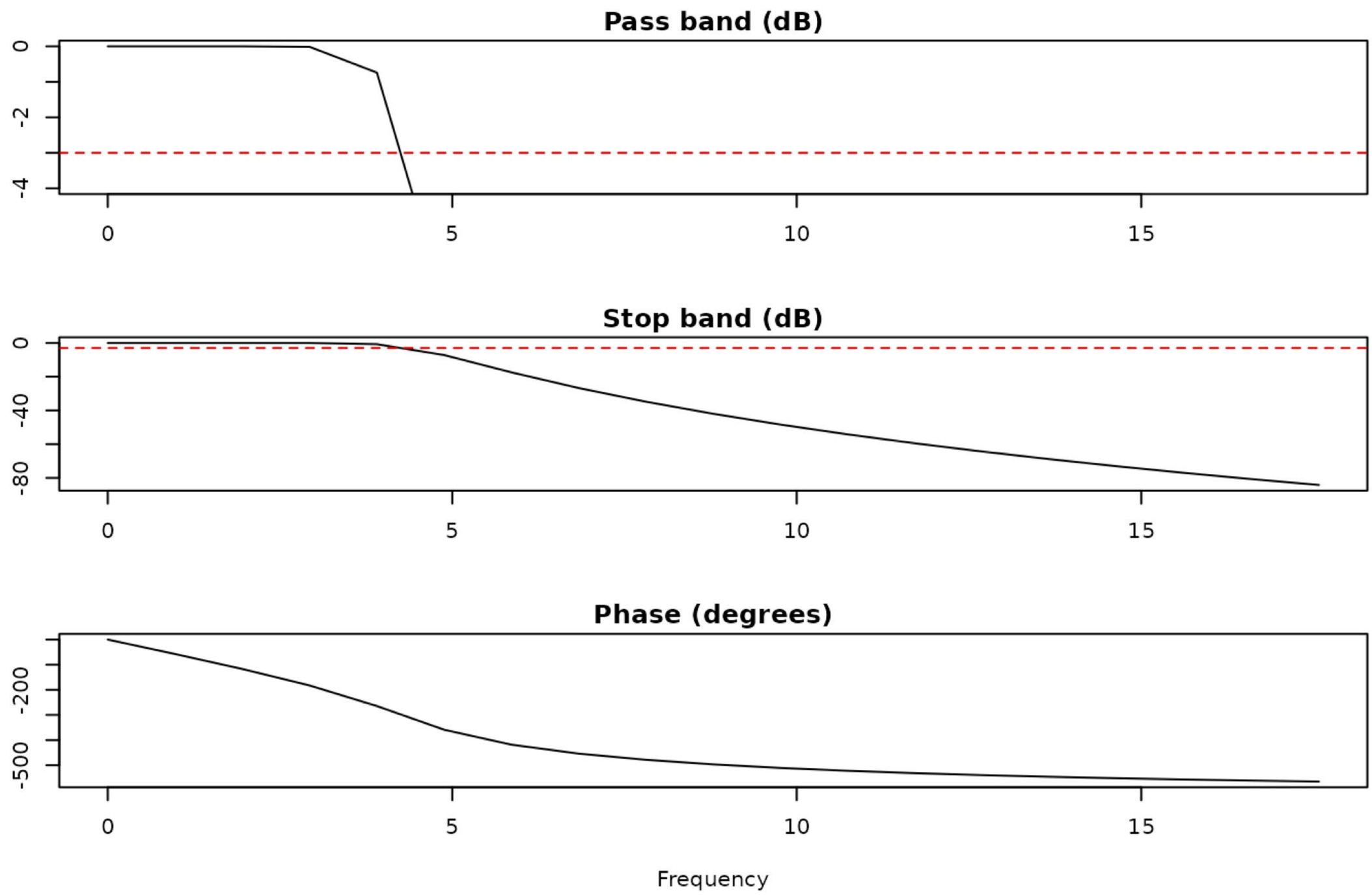
data provenance out-of-the-box

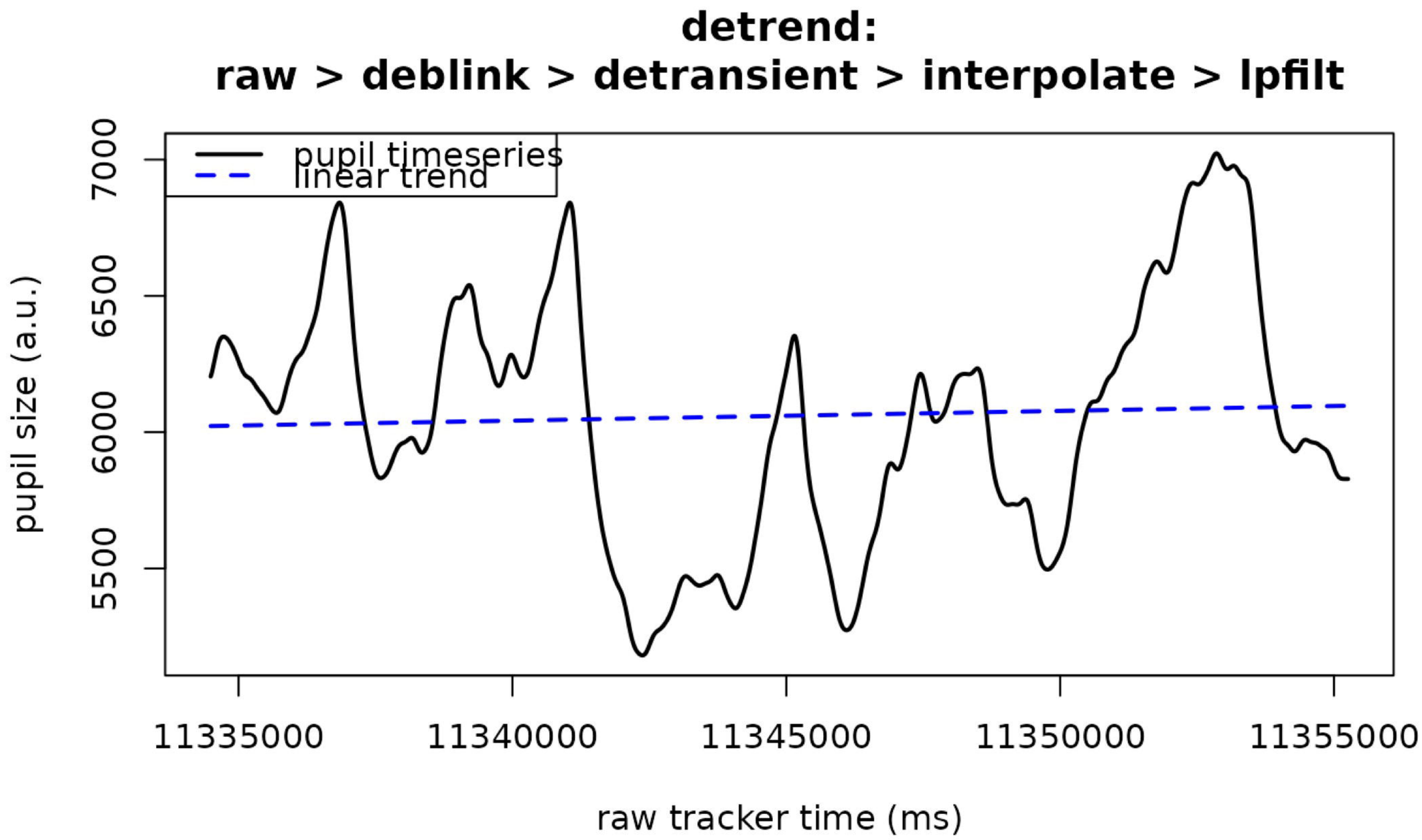
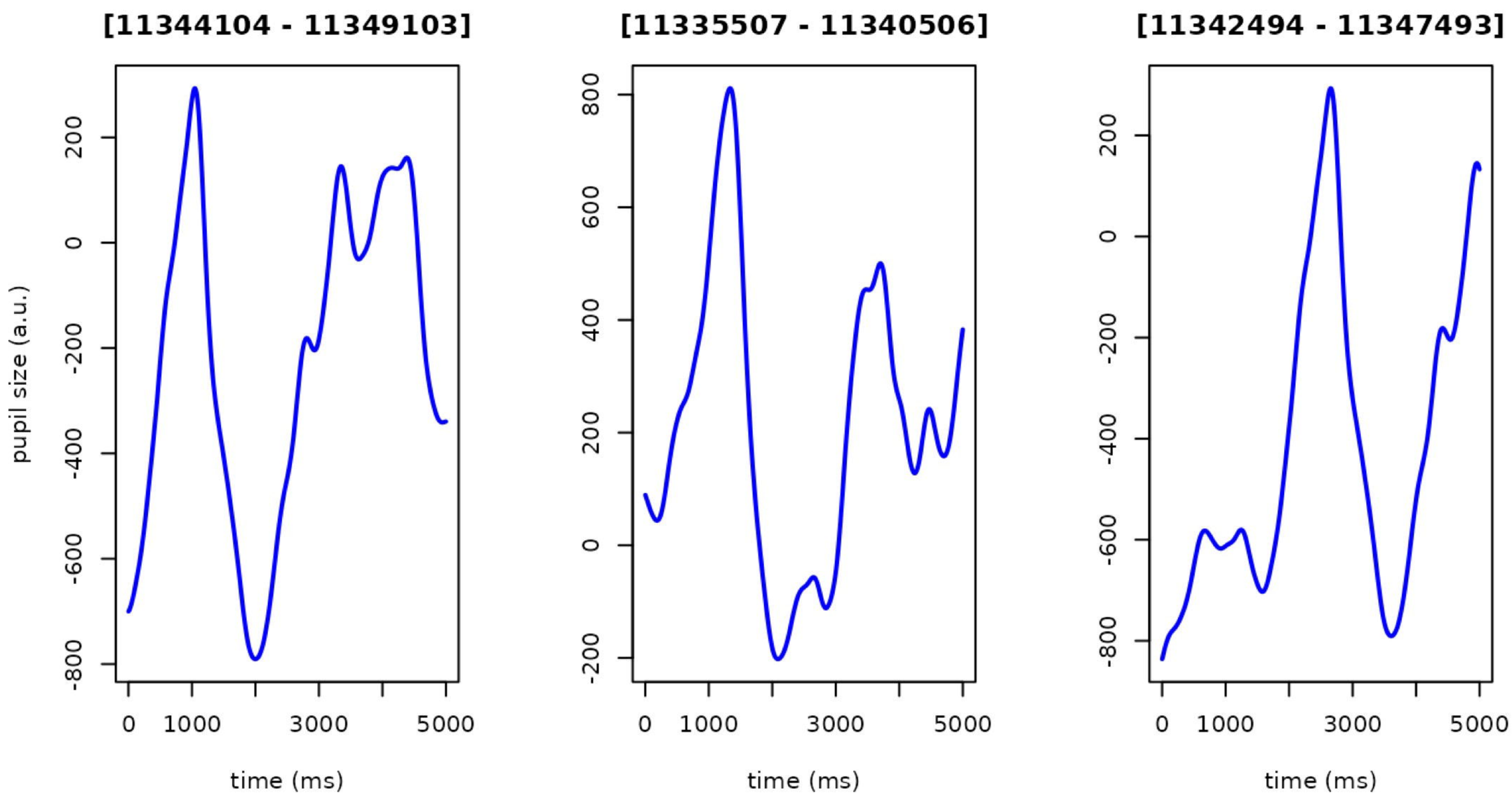
Before



After




Pass band (dB)
0
-2
-4
0
5
10
15
Stop band (dB)
0
-40
-80
0
5
10
15
Phase (degrees)
-200
-500
0
5
10
15
Frequency


detrend:
raw > deblink > detransient > interpolate > lpfilt
pupil timeseries
linear trend
pupil size (a.u.)
7000
6500
6000
5500
11335000
11340000
11345000
11350000
11355000
raw tracker time (ms)

deblink > detransient > interpolate > lpfilt > detrend