Infant Behavior and Development 54 (2019) xxx–xxx

Contents lists available at ScienceDirect

Infant Behavior and Development

journal homepage: www.elsevier.com/locate/inbede

Full length article

PyHab: Open-source real time infant gaze coding and stimulus presentation software

Jonathan F. Kominsky

Psychology Department, Harvard University, United States

ARTICLE INFO

Keywords:
Infant
Habituation
Preferential looking
Software
Gaze coding
Looking time

ABSTRACT

Infant looking-time paradigms often use specialized software for real time manual coding of infant gaze. Here, I introduce PyHab, the first open-source looking-time coding and stimulus presentation solution designed specifically with open science in mind. PyHab is built on the libraries of PsychoPy (Peirce, 2007). PyHab has its own graphical interface for building studies and requires no programming experience to use. When creating a study, PyHab saves a folder that contains all of the code required to run the study and all of the stimuli, making each experiment a self-contained, easily shared package. This feature guarantees the ability to replicate a PyHab experiment exactly as it was run. PyHab also has several features designed to support rigorous methodology, such as experimenter blinding and automatic stimulus control. In addition, because it is open-source, PyHab can be modified and improved not only by its developer but by anyone who knows Python.

1. Introduction

As discussed throughout this special issue, there are many challenges to replication in infant work. One set of challenges, which I will focus on here, has to do with sharing stimuli and practices between labs for either replications or cross-lab collaborations. Methods sections, regardless of the diligence of the authors, will always miss certain tiny details. These details can range from elements of the stimuli that cannot be captured in text or still images (e.g., the sound made by a mechanism in a live display) to details of live looking-time coding practices (e.g., when looking-time coding began relative to an attention-getter and the onset of stimuli).

In an ideal world we could simply share a whole experiment as a complete package that any lab could pick up and run. Some recent technological developments have moved us closer to this. With tools like OSF Asante et al. (2018) and Databrary (2012), it is now easier to share stimuli and other materials, which can be critical not only to precise replications but to interpreting existing results (e.g., Phillips et al., 2015). However, the tools currently used to conduct infant looking-time studies impose some limitations on what can and cannot be easily shared. Some limitations are insurmountable; a live-acted stimulus will never transfer perfectly from one lab to another. Others can be solved with technological advances. Here I present one such advance: PyHab, a free and open-source system for looking time coding and (optional) computer-controlled stimulus presentation, specifically designed to support rigorous and replicable infant research.

E-mail address: jkominsky@g.harvard.edu.

https://doi.org/10.1016/j.infbeh.2018.11.006
Received 2 March 2018; Received in revised form 23 June 2018; Accepted 19 November 2018

1.1. Existing infant looking-time tools

Infant looking-time research takes many forms, but most studies fall into one of three categories: Habituation/dishabituation (e.g., Colombo & Mitchell, 2009; Rankin et al., 2009), violation of expectation (e.g., Munakata, 2000), or preferential looking (e.g., Golinkoff, Ma, Song, & Hirsh-Pasek, 2013). There are many different ways to run all three types of studies, but there is one critical methodological feature of habituation studies in particular: Looking times must be coded "live" during the study run. Live coding is also often done with the other two types of studies as well, but is not always strictly necessary. With habituation studies, live coding is necessary in order to determine a habituation criterion (e.g., based on the sum of the infants' looking time over the first few trials) and when that criterion has been met, so that the experiment can proceed to the test trial.

In order to code infant gaze behavior in real time in a precise and efficient manner, experimenters typically rely on one of a handful of software solutions, most commonly XHab (Pinto, 1996), jHab (Casstevens, 2007), and Habit (Oakes, Sperka, & Cantrell, 2015). XHab and jHab can only be used to code looking times, they cannot directly control stimulus presentation. In other words, while XHab and jHab can tell an experimenter when to proceed to the next trial or when a habituation criterion has been met, the experimenter(s) must actually change the stimuli on their own. Habit can be used to control stimulus presentation directly, or be used without stimuli. All three programs share the important capability of being able to export (and import) all of the parameters for a given experiment. As our field moves to adopt the philosophy and practices of open science, this is increasingly important. By sharing these settings, it is possible to replicate the exact parameters of a particular experimental procedure.

However, certain limitations inspired me to create my own solution. As noted above, XHab and jHab cannot control stimulus presentation, meaning that studies often require at least two experimenters, one to code looking times and one to control stimulus presentation, and their ability to communicate will determine the precision of the stimulus onset and offsets.

Furthermore, none of these programs are open-source, meaning that their underlying code is not publicly available. This is not inherently a fatal flaw. However, it does limit these programs in certain ways. They cannot be easily modified for designs outside the ones they were originally intended to be used with, or in the case of Habit, the kinds of stimuli it was originally designed to support (movie files, audio files, and images). Because none of these programs are open-source, the creators have the sole responsibility of updating and modifying them for the needs of their users, rather than the users being able to make such modifications themselves. Therefore, I decided to create a solution that would not only suit my needs, but that anyone could modify to suit theirs.

Finally, while each of these existing solutions allowed for the exporting and sharing of settings files, there was no way to ensure that the program's behavior would be the same at some point in the future. If the program itself is updated, there is always the possibility that it might no longer be able to interpret the settings created for an older version, or would interpret them somehow differently (think of the issues you may have experienced moving between different versions of word processing or spreadsheet software). With an eye to the broader goals of open science, I wanted a solution that would allow me to share my experiments in a way that guaranteed future researchers the ability to replicate my experiment *exactly* as it was run. More broadly, I wanted to create a tool that enabled the best possible practices for rigorous, replicable science.

1.2. Introducing PyHab

Here, I introduce a new program called PyHab, which was designed with the specific goal of supporting open science. PyHab is a looking-time coding and stimulus presentation system built on PsychoPy (Peirce, 2007, 2009). PsychoPy is an open-source Python-based stimulus presentation program originally designed for psychophysics research. PyHab is a collection of code designed to run in a PsychoPy environment, analogous to how the psychophysics toolbox (Brainard, 1997) runs in MatLab®. While neither I nor PyHab are directly associated with the creators of PsychoPy, substantial credit is due to Jonathan Peirce and the PsychoPy community for making PyHab possible, and whenever PyHab is used, PsychoPy should be cited as well.

PyHab can be used in place of XHab/jHab and Habit. It can be used to control stimulus presentation directly or simply do looking-time coding alone. It can run on any modern operating system. It is specifically designed to allow experimenters to upload their *entire* experiment in one compact and self-contained package that others can then download an use to replicate the experiment exactly as it was run. Finally, it is both free and open-source, as is PsychoPy. In fact, it is PsychoPy's open-source design that allowed for the development of PyHab at all, and over the course of development I made some contributions to PsychoPy itself. That I was able to improve PsychoPy in the course of this project is a perfect example of one of the critical advantages of open-source software: Future development is not wholly dependent on the creator's ability to devote time to the project. While I intend to continue working on PyHab, anyone can contribute to its development.

In the remainder of this paper, I will briefly introduce PyHab's key features and capabilities, with a specific focus on how it is optimized for rigorous and replicable infant research.

2. Methods

2.1. Installation

Installation instructions can be found at https://pyhab.readthedocs.io/Installation.html as of fall 2018. PyHab is a collection of Python classes and scripts designed to run in PsychoPy (Peirce, 2007). PsychoPy was itself created to be open-source software for psychophysics experiments. In this case, PsychoPy operates as an environment for PyHab to run in. So, PsychoPy must be installed on

your computer (http://www.psychopy.org/installation.html).[1] One by-product of this dependency on PsychoPy is that as PsychoPy receives upgrades (as it does, constantly), PyHab incidentally benefits from them.

After installing PsychoPy, PyHab can be downloaded from http://github.com/jfkominsky/PyHab/releases (as of fall 2018). PyHab can be downloaded as a single. zip archive, which can then be uncompressed into a folder. This folder can be placed anywhere in your file-system, and no special installation script or package is needed in order to use PyHab. The contents of this folder are everything required to create and run experiments in PyHab, as well as a manual with detailed instructions.

To run PyHab, simply open PsychoPy, select the "view" dropdown menu, and select "Coder View". From this new window, go to file, open, find the PyHab folder you downloaded, and open "NewPyHabProject.py". From here, one can now run PyHab and create new experiments.

To start using PyHab, please refer to the "Quickstart Guide", located here (as of fall 2018): https://pyhab.readthedocs.io/en/latest/Quickstart.html.

2.2. Designing studies

PyHab has a "builder" interface that is designed to allow you to easily construct and customize experiments to your needs. The interface is, as of this writing, rudimentary but functional. However, it is also possible to create, manually, a settings file that contains all of the information required to run your experiment. If you have experience with programming in Python, you can also modify the code directly to do whatever you need to do.

2.2.1. Key features

PyHab's basic unit is the 'trial type'. A trial type designates not just a single trial, but a category of trials, such as habituation, test, familiarization, or anything else. Instances ('tokens', if you like) of these trial types are placed into a 'study flow' which determines the trial order. Trial types can be set to be infant-gaze-contingent (i.e., ending when the infant looks away for a certain period of time), or to always play to a set duration regardless of looking behavior. The looking-time and duration criteria for ending a trial is set at the level of the trial type as well.

When presenting stimuli, different trial types can be configured to have different attention-getters play before them, or no attention-getter at all. In addition, you can control whether the program waits for the experimenter to initiate a trial, or if it begins automatically following the end of the previous trial (with or without a set delay).

Trial types can be named however the experimenter wants. There is only one trial type that is in any way special: the 'Hab' trial type. Hab trial types create blocks of habituation trials when inserted into the study flow, which continue until user-defined habituation criteria have been met. Habituation blocks can also consist of multiple trial types, only one of which is 'Hab'. These meta-trials allow you to, for example, present an animation that is not gaze-contingent, and then present a still image of the final frame, and only base your habituation calculations on the gaze behavior for that still image.

PyHab gives you a high degree of control over the habituation criteria as well: The maximum number of habituation trials, the exact way the looking time criterion is computed, and the exact way subsequent trials are compared to that criterion are all modifiable in the process of building your experiment. The criterion can be set based on the looking time over the first N Hab trials (and the user sets a value for N), or continuously updated based on the contiguous window of N Hab trials with the greatest looking time, or even the N Hab trials with the greatest looking time regardless of contiguity. The criteria can then be evaluated against a moving window of the last N trials (e.g., if N = 3, after trial 6 looking at trials 4-5-6, and after trial 7 looking at trials 5-6-7), or fixed intervals of N trials (e.g., 4-5-6 and 7-8-9).

When using PyHab for stimulus presentation, each trial type can have several different stimuli assigned to it. By default, PyHab will cycle through the stimuli assigned to a given trial type each time an instance of that trial type appears in the study flow. For example, imagine that trial type 'A' has two stimuli assigned to it, X and Y, which were added in that order. The first instance of A in the study flow will present stimulus X, the second instance of A will present stimulus Y, the third instance will present X again, and so on. These stimuli can be movie files, images, audio files, or audio files paired with images.

The order of stimuli for each trial type can be manipulated between subjects by creating a condition file (which is also done in the builder). This condition file allows you to create different orders to be shown to different participants. It also allows for between-subjects manipulation of *which* movies are presented, as you can simply omit certain movies from certain conditions. Conditions, once configured, are simply referred to by labels except in the condition menu in the builder, allowing for easy experimenter blinding when running the experiment.

In addition to stimuli, you can create customized attention-getters, and have different attention-getters for each trial type. There is a default attention-getter (a looming yellow rectangle with a rising musical scale), but you can add attention-getters as movie files, or as sound files with a customizable looming shape (similar to the default attention-getter, but with different shapes or colors).

2.2.2. Other modifiable parameters

PyHab studies can be heavily customized using the builder interface alone. This includes what kinds of data are recorded in the output file, the pixel dimensions of the display and stimuli, the time between trials, which attention-getter is used (if any), whether the experiment is single-target or preferential looking, the background color of the stimulus screen, the inter-stimulus interval for

[1] As of the time of this writing, the latest stable build of PsychoPy is 1.90.3.

looping animations, and more. All of this can be changed without the slightest knowledge of programming or Python, making the program accessible to users at every level of technical sophistication.

For experimenters who are proficient at programming in Python, the possibilities become effectively limitless. As one nearly trivial example, in some of my own studies (e.g. Kominsky et al., 2017), rather than presenting movie files as stimuli, I used PsychoPy's built-in stimulus creation to create dynamic animations with simple geometric shapes (Peirce, 2007). Since PsychoPy's original intended usage was psychophysics studies, it has extensive tools for creating highly precise audiovisual stimuli. This was, in fact, one of the reasons for creating PyHab in the first place, as Habit cannot create such precise stimuli. Anything that can be done in PsychoPy can be done in PyHab (at least in principle). For example, instead of using pixels, one could convert all dimensions to degrees of visual angle, or standardize the luminance of stimuli. While such features are not currently implemented in PyHab by default, future development may add these options to the builder interface.

2.2.3. Features supporting best practices in infant research

Many of PyHab's options and features were designed to enable easy adherence to best practices in infant research. For example, one advantage of PyHab's condition system is that, once set, no information about the condition is presented to the experimenter running the study other than an arbitrary label. That means that a PI can design conditions and the experimenter who then runs the study will not know the order of presentation of any given condition.

This is one of several ways in which PyHab is designed to reduce experimenter bias by blinding the experimenter to various features of the stimuli. There are settings that control how much information the experimenter gets. In the most extreme case, the experimenter will only be told whether a trial is currently active or not, with no information about the trial number or type, and no information about whether a "gaze-on" key is currently being held down. This last point allows for simultaneous reliability coding in which neither coder can use the other's coding as cues (provided they cannot see each other physically press the keys).

A further important feature of PyHab is that the looking-time coding functions nearly identically whether or not it is presenting stimuli, excepting one clearly-labeled trial type option that can make some timing stimulus-dependent. You may choose whether or not to present stimuli every time you run an experiment file. So, for example, if one wanted to do a secondary coding of an experimental session that was run with PyHab, all you need to do is turn off stimulus presentation. The timing and trial parameters would remain exactly the same, including timing delays added in to mimic the delay created by an attention-getter.

2.3. Running studies

Once a study has been designed in PyHab, running it is designed to be straightforward, and as easy to use as any other looking-time coding software. The experimenter runs the experiment's launcher script from PsychoPy and selects "run", and whether they would like stimuli to be presented or not (if the stimuli are being controlled by PyHab). During the run, the experimenter presses one key to start each trial (or start the first trial, if the program is set to auto-advance between trials), and another key or keys to indicate whether the infant is looking at the screen (or which side of the screen they are looking at, in preferential looking paradigms).

PyHab also has the ability to abort a trial while it is active and start over from the beginning of the trial, or redo the most recently completed trial. The aborted or redone trial is still recorded but marked in the data as "no good", and not included in habituation calculations. During habituation blocks, PyHab can insert additional habituation trials after the criterion has been met, or jump directly to the test trial before it has been met (replicating similar functionality in XHab and jHab).

PyHab outputs up to four different data files with every run, depending on settings and circumstances. The one file that is always produced is the summary file, which is a csv file that is similar in format and content to what XHab or jHab would produce. This file has one line per trial with information about what was presented on that trial, the total looking time, number of looks to the screen, total time spent looking away, and number of looks away. In cases where there are two coders, the same information for the second coder will also be included. In preferential looking designs, it is not possible to have a second live coder, but information about gazes to each side is recorded instead. The columns included in this data file can be set in the builder. By default, all available data are recorded.

The second data file is a "verbose" data file, that has one line for each look at or look away from the screen, with the onset, offset, and duration of that look. The third and fourth data files are only produced when there is a second coder. The third file is simply the verbose data for the second coder. The fourth file is a set of four reliability statistics that PyHab computes automatically when it detects two coders: Raw percentage agreement, weighted percentage agreement,[2] Cohen's Kappa, and Pearson's *r*. There is also a stand-alone reliability script that will compute these values for any two verbose data files, even if they were created at separate times. Using this script one can compute reliability for preferential looking designs as well.

2.4. Sharing studies

Perhaps the greatest unique strength of PyHab is that it is specifically designed to make it easy to share your exact experiment, no matter how idiosyncratic, in a form that other researchers can easily make use of. When you save a project in the builder, it creates a folder that contains all of the PyHab scripts needed to run the study, a copy of the builder script, a csv file with all of the experimental parameters, another csv file with the conditions (if needed), a folder containing all of the stimuli, a folder in which all data will be

[2] The percentage agreement over the whole experiment, with the contribution of each trial weighted according to the length of that trial.

saved, and a launcher script from which one can run the study or open builder to modify it. Even if you make further modifications to the experiment later, everything needed to run it will be in this folder. An example of one such folder is part of the PyHab download. In it is everything required to run a simple demonstration study, or view and modify any of its parameters. That folder alone is sufficient to run that experiment, or modify it in the builder, without having PyHab on your computer already.

Notably, because PyHab saves a copy of all of the presentation and builder code at the time the study was created, even if PyHab itself changes at a later date, the code will be *exactly* as it was when the study was created. This future-proofs your experiment to a degree; the only risk of cross-version incompatibility is from PsychoPy itself. Since previous releases of PsychoPy remain available for download, and PyHab records the version of PsychoPy used to create an experiment in the experiment's launcher script, even then it should be possible to re-create a study exactly as it was run for the forseeable future.

2.5. Example uses

Early versions of PyHab have already been put to use in several labs, and in my own work, but due to its novelty no studies using PyHab have yet been published in peer-reviewed journals. ManyBabies 1 (Bergmann et al., 2017) has a custom-modified early-development version of PyHab as one of the options for the single-screen version of the study. This allowed the standardized design, stimuli, and data formatting to be easily shared across labs.

3. Discussion

There are many challenges in making infant research replicable and robust, and software cannot solve all of them (if only!). Using software designed specifically to support best practices and exact replicability makes these challenges easier to address. PyHab is far from perfect, but offers an opportunity to apply the philosophy of open and rigorous science to the tools we use in our research. PyHab is designed to be as transparent as possible, and is deliberately optimized to make it easy for others to replicate your experiment.

Of course, there are other challenges to replicability which PyHab cannot solve. Live stimuli are difficult to replicate precisely between labs, and physical lab spaces differ in various ways. Different labs may use different practices for coding looking times (e.g., do you assume the infant is looking at the target or not, if it is ambiguous?). PyHab is not meant to be a replacement for a clear and detailed methods section, only to save aspiring replications the trouble of re-creating an entire experiment based on a description alone. PyHab is also extremely useful for multi-lab projects like ManyBabies, as the designers can create a single PyHab folder and simply distribute it to each lab with full confidence that everyone will be able to use it.

In comparison to existing software, as of this writing, PyHab matches or exceeds all the capabilities of XHab or jHab. As looking-time coding software alone, to my knowledge, there is nothing that either of these programs can do that PyHab cannot. Habit is a much closer comparison, and the capabilities of Habit and PyHab are very similar. Habit has the ability to display stimuli on two screens, which PyHab currently does not. There are other minor differences in capabilities, but the most significant difference by far is that PyHab is open-source.

Uniquely among software of this kind, everything PyHab does is transparent. If you want to know how something is being calculated or displayed, all of the relevant code is available to peruse. The value of this was particularly well-illustrated by an episode early in PyHab's development, when creating the reliability calculation. I attempted to determine how reliability was computed in XHab. It turned out, based on careful investigation, that XHab computes reliability over 200 ms "chunks", i.e., if a coder indicated gaze-on for any portion of those 200 ms, the whole chunk is coded as "gaze-on" when comparing it to another coder. This was only discovered because I was able to find XHab's original documentation, which is no longer widely available (in fact, I was lucky that there was a printed copy in the lab). In PyHab, the code for this calculation is easily accessible and interpretable to anyone who knows Python (for the curious, PyHab uses 16.7 ms chunks).

In its current form, PyHab has a number of limitations which will make it inappropriate for some paradigms. As noted above, it cannot present stimuli on multiple screens while having a coder interface on a third screen. While PsychoPy has the ability to interface with eye-trackers and imaging equipment (Peirce, 2009), none of those capabilities have been incorporated into PyHab. In addition, the builder interface is very much a work in progress. However, PyHab is not by any means a final product, and most likely never will be. I plan to continue developing PyHab and improving its capabilities, and anyone who knows Python can contribute to it as well. As long as there is an interested community of users, PyHab will continue to improve. However, by design, it will improve without ever losing the ability to faithfully replicate studies created at earlier points in its development.

Acknowledgements

This work was supported by NIH grant F32HD089595. The author would like to thank Jonathan Peirce for the creation and development of PsychoPy, Susan Carey and the Harvard Lab for Developmental Studies for providing a test-bed for the first PyHab studies, Florin Gheorgiu for assisting with the creation of the reliability calculation, and Molly Dillon, Holly Huey, and Nicole Loncar for being willing beta-testers. The author would also like to thank an anonymous reviewer of this manuscript for offering feedback not only on the manuscript but also PyHab's code-base, greatly improving both.

References

Asante, K., Barbour, E., Barker, L., Benjamin, M., Bowman, S. D., Boughton, A. P., … Harber, B. (2018, January 25). OSF. Retrieved from osf.io/4znzp.

Bergmann, C., Frank, M. C., Gonzalez, N., Bergelson, E., Cristian, A., Ferguson, B., … Shukla, M. (2017, November 29). ManyBabies 1: Infant-Directed speech preference. Retrieved from osf.io/re95x.

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision, 10*(4), 433–436.

Casstevens, R. M. (2007). *jHab: Java habituation software (version 1.0.2)*.

Colombo, J., & Mitchell, D. W. (2009). Infant visua habituation. *Neurobiology of Learning and Memory, 92*(2), 225–234. https://doi.org/10.1016/j.nlm.2008.06.002.

Databrary (2012). *The databrary project: A video data library for developmental science.* Retrieved fromNew York: New York University. http://databrary.org.

Golinkoff, R. M., Ma, W., Song, L., & Hirsh-Pasek, K. (2013). Twenty-five years using the intermodal preferential looking paradigm to study language acquisition: What have we learned. *Perspectives on Psychological Science, 8*(3), 316–339. https://doi.org/10.1177/1745691613484936.

Kominsky, J. F., Strickland, B., Wertz, A. E., Elsner, C., Wynn, K., Keil, F. C., & Carey, S. (2017). Early-developing causal perception is sensitive to physical constraints on collision events. *Poster presentation, cognitive development society biennial meeting*.

Munakata, Y. (2000). Challenges to the violation-of-expectation paradigm: Throwing the conceptual baby out with the perceptual processing bathwater. *Infancy, 1*(4), 471–477.

Oakes, L. M., Sperka, D. J., & Cantrell, L. (2015). *Habit 2. Center for mind and brain.* Davis: University of California.

Peirce, J. W. (2007). PsychoPy—Psychophysics software in python. *Journal of Neuroscience Methods, 162*(1–2), 8–13. https://doi.org/10.1016/j.jneumeth.2006.11.017.

Peirce, J. W. (2009). Generating stimuli for neuroscience using PsychoPy. *Frontiers in Neuroinformatics, 2*, 10. https://doi.org/10.3389/neuro.11.010.2008.

Phillips, J., Ong, D. C., Surtees, A. D., Xin, Y., Williams, S., Saxe, R., & Frank, M. C. (2015). A second look at automatic theory of mind: Reconsidering Kovács, Téglás, and Endress (2010). *Psychological Science, 26*(9), 1353–1367. https://doi.org/10.1177/0956797614558717.

Pinto, J. P. (1996). *XHAB: Experimental control software for MS-DOS (version 6.5).* Palo Alto, CA: Author.

Rankin, C. H., Abrams, T., Barry, R. J., Bhatnagar, S., Clayton, D. F., Colombo, J., … Thompson, R. F. (2009). Habituation revisited: An updated and revised description of the behavioral characteristics of habituation. *Neurobiology of Learning and Memory, 92*(2), 135–138. https://doi.org/10.1016/j.nlm.2008.09.012.