UNIVERSITAT DE BARCELONA

Final Degree Project

Biomedical Engineering Degree

Disentangling human decisions under strong time pressure in an expectation-based 2AFC auditory task

Barcelona, 8th June 2022

Author: Alexandre Garcia-Duran Castilla

Director/s: Manuel Molano Mazón

Jaime de la Rocha

Tutor: Roser Sala Llonch

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my project director Manuel Molano for his constant support, guidance and willing to transmit knowledge and motivation, and for his kindness and time.

I would also like to acknowledge and give special thanks to Jaime de la Rocha for giving me the opportunity to work on this project and for the valuable suggestions and advice given.

A special thanks to all the researchers in the laboratory, who also played a key role in giving advice and providing a friendly working environment.

Moreover, I would like to give my appreciation to my tutor Roser Sala, who guided and advised me on this last part of the journey.

Finally, I would like to extend my heartfelt thanks to my family and friends for their unconditional support all the time.

Only these countless attempts, corrected by natural selection, could, like in any other organ, make the central nervous system a system adapted to its particular function. For the brain: give the sensitive world a suitable representation for the functionalities of each species [...] and subjectively simulate the experience to foresee the results and prepare the action.

Jacques Monod – *Chance and necessity: Essay on the Natural Philosophy of Modern Biology* [1]

ABSTRACT

Decision-making in humans is difficult to study because it is a complex process influenced by many factors. How are these factors combined and their effect is studied in rats. Recent studies have thoroughly characterized the behavior of rats performing a Two-Alternative Forced Choice (2AFC) auditory task, in which the probability to repeat the previous stimulus category is varied in a blockwise fashion. These studies showed that rats exhibit a transition bias: a tendency to alternate/repeat the previous response using an estimate of the probability given the recent trial history. However after error trials, the transition bias was null. Even though it is suboptimal, this so-called reset strategy has been shown to be highly robust and present in many task variants. On the other hand, the reaction times of rats performing the 2AFC task has been shown to be governed by two independent processes: one that depends on the accumulation of the stimulus evidence and a second, stimulus-independent process that only depends on the time elapsed since the beginning of the trial. Here we have investigated the behavior of human subjects performing an auditory 2AFC task presenting the same type of correlations experienced by the rats. We found that their strategies were more heterogeneous, with some subjects displaying a clear reset strategy while others developed a more optimal strategy. Furthermore, the reaction times of the human subjects showed evidence of being influenced by the two processes mentioned above, suggesting that the existence of the two different mechanisms described in rats may be a general feature present across species.

Keywords

Auditory 2AFC Task, Reset/Reverse Strategy, Transition Bias, Reaction Time, Expectations

Table of contents

List of figures

List of tables

1. Introduction

The human brain is an intricate organ full of unanswered questions. From the anatomy, its components can be studied, but what makes this organ unique and knotty is its complex network formed by its main components: neurons. A single neuron inhibits or activates the surrounding ones, which can be modeled at the unity scale; but the problem comes when facing the 86 billion neurons [2] conforming the brain. And not only with neurons, but with the connections between them, called synapses, because one unit can be connected to many and vice-versa. Studies show that each neuron may be connected to up to 10,000 other neurons, passing signals to each other via as many as 1,000 trillion synapses [3]. Knowing this, brain's complexity is evident. Therefore, researchers have found ways to simplify this huge complexity by finding patterns and relationships in many different aspects, such as imaging, behavior, language, etc. making this an interdisciplinary science involving many different fields.

In this project, we will focus on the study of human behavior. All mammals have complex behaviors, but these are generally stereotyped in nature and lack the flexibility of human behavior [4]. Since the split from the chimpanzee lineage, the human brain has increased three-fold in size and has acquired abilities for vocal learning, language, and intense cooperation [5]. To date, researchers have made substantial progress toward defining uniquely human aspects of cognition [6], but there are still numerous unknowns. To answer them, simplifying the concept of behavior and studying its basis has been a powerful tool in the neuroscience field. We will focus on decision-making, a fundamental component of behavior. After the past 50 years of work in judgment and decision making, we now know that human choice is often not as rational as one might expect. In several contexts, human decisions tend to systematically deviate from what rational choice models would predict [7], for example, the context and situation influence them [8] even in scenarios where it is not informative.

Amongst all the different kind of choices, the ones studied here are the ones driven by sensory information. Due to their complexity, the mechanism used by the brain to integrate multisensory information during decision making is still unclear [9]. Hence, over the past two decades, understanding how perceptual decisions are made has become a central topic in neurosciences [10]. Thus, traditional techniques for studying decision-making are now complemented by novel approaches, which include cell specific targeting, the use of naturalistic stimuli, new analyses for population recordings and automated methods for movement tracking [11]. Some of these techniques try to simplify the task for the subject to use the lowest number of senses possible, so the small interactions and the basis of perceptual decision making can be understood.

Taking a general view, all the species have evolved to optimize their decisions and compete for survival. In our everyday life, we are constantly receiving sensory inputs because of years and years of evolution, which made possible the development of what we now call the senses. These senses provide information of our near surroundings and can be used in many different situations: to see a dangerous animal and thus being able to hide, to smell herbs and discern between venomous or medicinal plants, etc. Therefore, perceptual decision-making is the process by which animals detect, discriminate, and categorize information from the senses [10]. Facing perceptual uncertainty, the brain combines information from different senses to make optimal perceptual decisions and to guide behavior [9]. However, not only sensory information is gathered to decide, studies have shown that perceptual decisions can also be influenced by expectations built from previous experiences [12]–[15]. These previous experiences can present

patterns understandable for the animal and an expectation can be formed, and joining this information with the sensorial, a more optimal choice can be made. This is related to the concept of cause, since our brain creates a probabilistic model of the surroundings, using information from past experiences, building relations cause-effect, which defined by David Hume [16] are:

A cause is an object precedent and contiguous to another, and so united with it, that the idea, of the one determines the mind to form the idea of the other, and the impression of the one to form a more lively idea of the other.

And by Pierre-Simon Laplace [17]:

Present events are connected with preceding ones with a tie based upon the evident principle that a thing cannot occur without a cause which produces it. […] We ought to regard the present state of the universe as the effect of its anterior state and as cause of the one which is to follow.

In other words, when two objects are constantly conjoined in our experience, observing the one naturally leads us to form an idea of the other. For example, all classical physics laws can be understood by animals in a perceptive way: due to repetition of the events, we know that the sun will rise next morning the same way that rose today, that apples fall from trees to the ground, etc. Animals save a memory of these events received by the senses, and by means of repeating, create expectations of what will happen next time because a seemingly non-alterable repetition pattern can be detected. Ludwig Wittgenstein [18] said that, perceptually speaking:

That the sun will rise tomorrow is a hypothesis; and that means: we do not know whether it will rise.

Meaning that we build expectations from experience, but we do not know for sure whether the sun will rise or not, we just assume it because of the huge evidence of the repeating experience. Our brain creates a predictive model, of what will happen in the next exact same situation. Therefore, patterns can be detected in past experiences and our behavior, and choices, are affected by them.

Another example is a tennis match [12], [13], where players can alternate (left, right, left, right) to make the opponent move the maximum possible, and afterwards repeat when the opponent is thinking that there is an alternating scenario. Players have a few milliseconds to receive sensory information, make an estimation and a motor plan, and execute it. Therefore, expectation can suppose a great advantage because subjects can respond faster. There, players try to understand the patterns performed by the opponent while they are playing, they accumulate evidence of these sequences and while the rival is doing the same, try to surprise the opponent with an abrupt change.

To sum up, this project will study how human beings retain memory of recent perceptual experiences and are able to build expectations in sequential situations. Also, there will be a comparison between subjects of different species whose data comes from Brain Circuits and Behavior Lab (IDIBAPS). This project includes all the analyses performed to data and a fitted model.

These situations are frequent in the daily life and understanding how people behave in them can be used to help in the early diagnose of mental disorders, such as Alzheimer or Parkinson diseases; or to study how a certain medicine can affect the reaction time in front of these stimulus.

1.1 Objectives

Here the objectives of the project are defined, separated by the general, the analysis-specific and the secondary. The general objectives are:

- Study if humans are affected by prior expectations in a sequential task.
- Analyze the modulation of reaction times in humans.

In order to reach these objectives, the following tasks were defined:

- Create an acquisition system of psychometric data.
- Device and software implementation.
- Quantify the strategies adopted by humans in sequential tasks.
- Model the choices of humans in perceptual decision-making tasks.
- Understand sequential tasks.

Finally, secondary objectives have been defined:

- Create a predictive model.
- Apply the analyses in a clinical environment.

1.2 Temporal and spatial limitations

Regarding the analysis, since the work is based in computer programming, there are no spatial limitations because work and meetings can be done at home. However for the experiment, which is supervised by me, is run in the August Pí i Sunyer Biomedical Research Institute (IDIBAPS for the initials in Catalan). The other parts of the project do not require a specific location.

Concerning time limitations, bibliographic research and programming courses were done five months before the project is defined to finish. Afterwards, data was extracted from twenty from subjects, and this process lasted one month, extracting data from five subjects each week. During data extraction, some analyses were done, and hypotheses proposed. Then, the programming work was developed in the four remaining months. To summarize, the project lasted up to eight months.

1.3 Aim of the project

This work is directed towards whoever follows my line of research work, because one of its purposes is learning and acquiring knowledge regarding the human brain and its complexity, full of conundrums for the moment. Many possible applications could be developed regarding different sectors. Some examples could be:

- In hospitals, it can be centered in the study of neurodegenerative diseases' affections regarding human behavior and understanding it can be key to early detection and therefore prevent or control the disease.
- In the pharmaceutical industry, it could be used for testing as a drug validation system.
- In sports industry, its use could be centered in training elite athletes to reduce its reaction time to perceptual stimulus.
- In research, since it is a non-invasive study, using it in humans would reduce the use of laboratory animals.

2. Background

To have a better understanding of the topic, general knowledge and previous studies, I will analyze them in more detail in this section.

2.1 Context

Neuroscience now stands at the boundary of an exciting new era, with computational formalisms playing a significant role [19], because to learn how cognition is implemented in the brain, we must build computational models that can perform cognitive tasks and evaluate such models with brain and behavioral experiments [20]. Nowadays computational cognitive neuroscience is an emerging discipline that employs mathematical analysis and computational models to understand the neural basis of cognitive functions [21]. Its main goal is to reveal the computational mechanisms by which neural circuits encode, store, and analyze perceptual signals; combine them with other behaviorally relevant information; and use them to resolve conflicts between competing motor plans [22]. One advantage of computational models is that they provide access to latent variables, which are the ones inferred indirectly from visible ones, which cannot be directly observed from behavior [23]. In our project, we focus on sequential effects in perceptual decision-making tasks.

2.1.1 Perceptual decision-making and Two-alternative forced choice (2AFC) task

Perceptual decision making is the process by which sensory information is used in the brain to guide behavior in the external world. This involves gathering information through the senses, evaluating, and integrating it according to the current goals and internal state of the subject, and using it to produce motor responses [22]. This process of extraction, integration, and interpretation of sensory information enables individuals to perform perceptual decisions about the most likely correct response of the task [24]. These perceptual decisions can be of varying complexity as determined by different contexts [25]. Since the context of the experiments is well-known, analysis can be done with all boundaries settled. In addition, the process of perceiving and identifying sensory signals is often a split-second instance of perception. However, this rapid action is the outcome of multiple stages of decision-making based on various levels of neural and cognitive processing [24].

Since cognition is intricate, researchers try first to understand simplistic models and interactions. In other words, build the basis of understanding a certain cognitive process, which in our case is perceptual decision-making. Therefore, most of the literature has focused on tasks involving categorization of a single stimulus into one of two response options [26]. The standard two-alternative forced choice (2AFC) method is a ubiquitous paradigm. During one trial of a typical 2AFC experiment, a subject is asked to decide about the perceived difference between two stimuli regarding a particular stimulus parameter of interest [27]. Therefore, it is ideal for measuring detection or discrimination thresholds. Most of them are based on simple choice tasks, where subjects need to choose between two possibilities (2AFC tasks). However, a small number of studies focus on 3AFC or NAFC tasks as well, which means the possibility to choose across three or more alternatives [14], [28].

These tasks can provide several distinct types of stimuli, depending on the sense which will receive the stimulus:

- Visual [29], [30]

- Auditory [12], [15], [30]
- Somatosensory [31], [32]

Tasks can also be classified depending on the fundamentals of the task, on what does the subject has to choose between. For example, we can find measurements of contrast sensitivity, where subjects are asked to report the position of an object, sound intensity, etc. which has different contrast from trial to trial [12], [29], [33]. Another example is the discrimination of different objects or sounds, where subjects are asked to recognize a specified target [34]. Also, these tasks can present trial-to-trial correlations, such as patterns [12], [14]. For example, repetition or alternation sequences.

In our case, we have focused on Intensity Level Discrimination Auditory 2AFC. In the task design, we can tune the following variables:

- Fixation Time: the time that the subject has to wait from the start of the trial until the stimulus is presented.
- Response Time: the time in which the subject has to answer. It starts counting as soon as the Fixation Time ends.
- Stimulus strength/Evidence/Coherence: standardized stimulus balance between left/right sides. The higher the value, the more unbalanced the sound is, and thus the easier to discern between sides.

And the common outcomes returned by the task are:

- Reaction Time (RT): is the time that passes since the Fixation Time ends and the subject makes a movement to answer. This can give information on the time that the subject takes to process the stimulus, or whether it has planned an action before the stimulus starts and therefore it can answer faster. This kind of responses are known as Proactive Responses [15].
- Accuracy: how many correct response trials over the total number of trials, considering only valid trials.
- Motor Time (MT): is the time that the subject takes move and finish the answer. Its beginning is the Reaction Time and ends when the participant finishes the trial.

2.1.2 Sequential effects

Many of the events that we observe, as well as the behaviors we produce, are sequential in nature. Consequently, to adapt and survive, all higher organisms must learn to operate within a temporally bounded environment where sequential events occur [35], making these sequential effects important for learning and prior beliefs [36]. All these higher organisms rely on expectations, derived from experience, to make quick decisions. We can define sequential effects as the influence that a recent experience has on the performance of the current and subsequent decisions.

Sequential effects have been observed in a wide range of experimental tasks conducted both in non-human animals [12], [15] and in humans [37]. Moreover, studies have shown that they increase when individuals perform a task repeatedly or perform a series of tasks [34], and that subjects, indeed, present faster responses with a higher accuracy if they detect a pattern [38]. However, when following an error, subjects manifest a more cautious behavior, which is

demonstrated by an increase of the Reaction Time (time until the subject responds) and a decrease in accuracy [34].

Moreover, subjects can show a so-called win-stay-lose-switch bias [12]. It means that they are more likely to pick the same choice if it previously led to a reward and, vice versa, to opt-away from the unrewarded one. This competitive behavior is common and has been identified even in people playing the game of "rock-scissors-paper" [39]. The previous biases have been seen when stimulus is harder to discern or perceive. In fact, it has been shown in rats and humans that the impact of the prior on accuracy is largest for low stimulus strength and it vanishes as the stimulus strength increases [12], [37]. In other words, when stimuli were hard to categorize, expectations benefitted subjects to provide a correct response. Nonetheless, they kept following the patterns even if the bias led to a lower performance and there is evidence that doing it did not lead to any advantage [40]. In fact, agents applied this approach even in situations where there was no correlation between the events [41]. To prove that, subjects were set in tasks with totally randomized trials, and they would still exhibit a bias toward one particular choice depending on the recent history of trials, even when subjects were explicitly informed that the stimuli were completely random [34].

This is thought to indicate that sequential effects are part of an intrinsic behavior that cannot be easily modulated, being the result of the brain's attempt to adapt to a dynamic and constantly changing environment [42]. They look like a consequence of a learning process in which subjects try to increase future rewards, minimizing the error possibilities [43], [44]. In fact, it has been shown that this mechanism is dysfunctional, over or under-relying on expectations, in subjects with schizophrenia [45], autism and dyslexia [46]. Moreover, perceptual decisions of individuals with dyslexia and individuals with autisms are less influenced by previous trials than neurotypical participants [46].

2.2 Previous work

This project emerges from four previous studies conducted in recent years at Brain Circuits and Behavior Lab, at IDIBAPS. Two of them studied rats, and another one studied recurrent neural network. The last one, based in humans, is the precursor of this project. I summarize these studies below.

2.2.1 Response outcomes gate the impact of expectations on perceptual decisions

This study was developed by Ainhoa Hermoso-Mendizábal et al. and published in 2020 [12]. They wanted to study how the recent history of stimuli, responses and outcomes influence perceptual choices in rats. They created a framework that could explain how rats form their expectations, investigating if and how the combination of these and sensory evidence can be dynamically modulated.

Pursuing this objective, they developed and trained rats in a two alternative forced choice (2-AFC, Fig. 1, left) task, in which animals are exposed to a stimulus, and must choose between two options. In this case, the stimulus was auditive and was applied after a fixation break of 300ms. The rats received it from both left and right, but with variable amplitudes between them, meaning in some trials the left side was heard stronger and vice-versa. Once they chose a side, they had to lick the corresponding port (Left or Right). Therefore, their options were left or right, and they received a reward (water) when getting a correct answer. After an erroneous response, rats were punished with light and 5 seconds of waiting. Hence, feedback was present.

Also, the trials presented repetition and alternation patterns. These were created using a two-sided Markov chain in which the probability of repeating the same correct side P_{rep} was 0.7 in repeating context and 0.2 in alternating context (Fig.1, right). This way, they could study whether rats understood the statistics behind the task and developed an optimal behavior considering trial history.

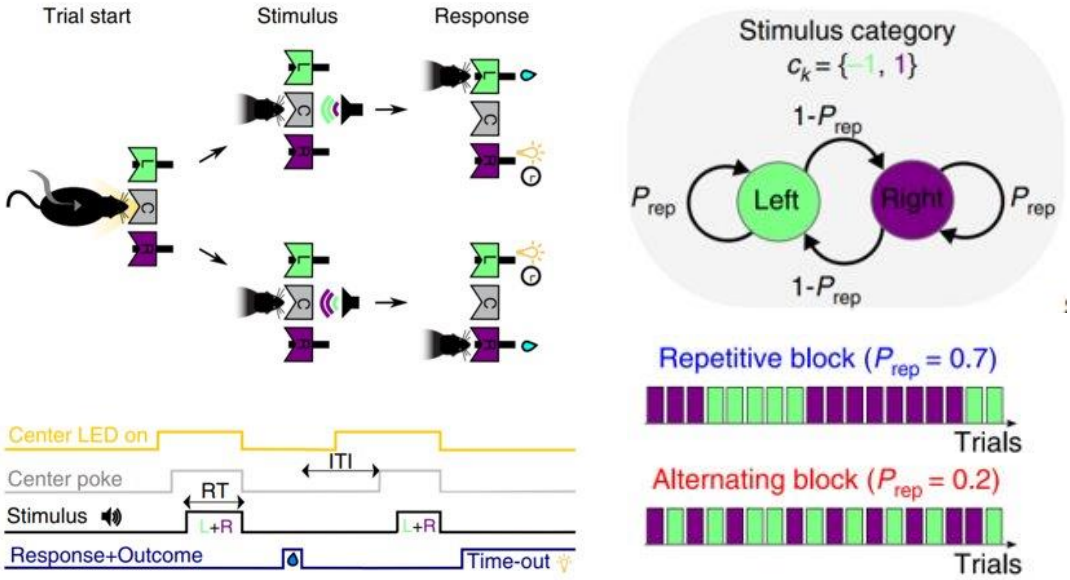


Figure 1. Auditory discrimination task and stimulus sequence statistics

After studying psychometric parameters, they saw that after correct responses, rats showed a transition bias: they tended to repeat answers in repetitive blocks and alternate in alternating blocks, meaning that they combined the stimulus with previous trials to compute the decision. The psychometric curves, which are the subject's repeating probability over stimulus' repeating evidence, show a horizontal shift for both repeating and alternating blocks, this shift is called transition bias (Fig. 2, left). However, after error answers, rats showed a reset strategy, meaning that they did not use the previous trials, but relied only on the stimulus. In this case, the shift in the psychometric curves would be zero, null (Fig.2, center).

To quantify the use of previous trials, the authors developed a Generalized Linear Model (GLM) using different regressors, distinguishing after error and after correct trials:

- Lateral: the subjects consider the last choices made and they create an attraction to the most repeated response (e.g., if the last four response have been R+ R+ R+ L+, the subject will be more likely to repeat R, where "+" means correct answer)

- Transition: the subjects detect a pattern in the trials such as repetition or alternation of the correct answer. (e.g., if the last four response have been L+ R+ R+ R+ R+ it means there have been Alt, Rep, Rep, Rep; subjects detect a repetition pattern). This transition regressors are used to quantify the reset strategy: it is expected that the weights after correct trials are positive due to transition bias and after error trials are zero, since rats ignored the transition bias after error.

Consistent with the previous analysis, the GLM showed that after error responses, rats relied only on the stimulus, ignoring expectations, but not erasing them, because after a correct response they used expectations from before the error. Consequently, after error weights for the transition

regressor (T++) are zero (Fig. 2, right), meaning that they did not contribute to the decision; whereas for after correct the weights are positive and decayed to zero some trials back, meaning that anterior trials to the previous 6-10 did not affect the decision.

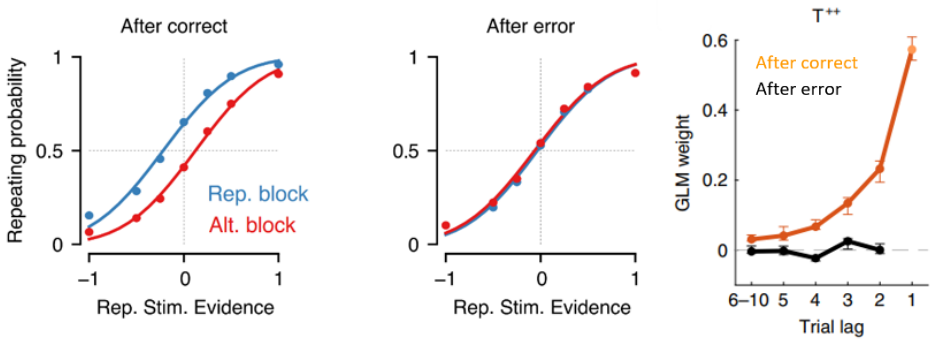


Figure 2. Left, center: psychometric curves after correct and after error trials, respectively. Transition bias and reset strategy can be visualized in the presence or non-presence of the horizontal shift, respectively. Right: GLM transition weights, differentiating after correct and after error trials.

2.2.2 Pre-training RNNs on ecologically relevant tasks explains sub-optimal behavioral reset

Molano-Mazon et al. [14] dealt thoroughly with the sub-optimal reset strategy adopted in a 2AFC task with serial correlations, behavior originally identified by Hermoso-Mendizabal et al. [12].

As previously said, animals' decision-making processes are determined by the perceptual stimulus but also by individual experience and structural priors. However, after error trials, animals ignored the information acquired by the choice history and followed a strategy only based on the current stimulus. An optimal agent, instead, would have used the error to learn and have reversed the transition bias (reverse strategy). Fig. 3 shows an example: in the bottom part the optimal agent is clearly in an alternation block (L-R-L) and after doing an error (next R), this subject must think that Left should be the correct, and since it was an alternation block, now the correct answer must be Right. The moment where the subject considers and learns from the error is called Conterfactual inference. Finally, the responses have been L-R-L-R-R (alternation-repetition), so that is why there is a reverse strategy.

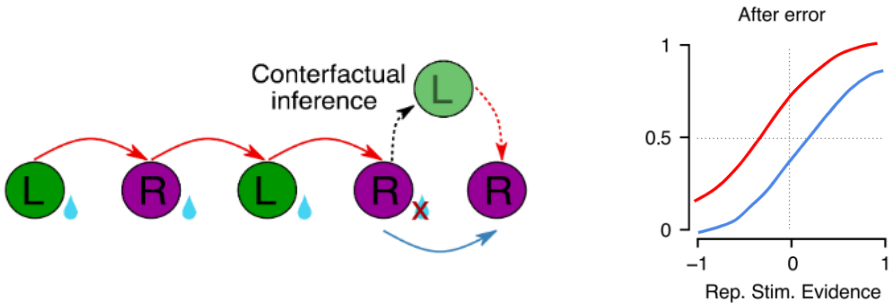


Figure 3. Left: example showing the counterfactual inference intuition. Right: theoretical result of the reverse strategy in the psychometric curves.

The first aim of this study was to quantify the extent to which rats followed the reset or the reverse strategies. To do so, they trained rats in different variants of a 2AFC task and modeled their

choices using a Generalized Linear Model (GLM). One of the main results was that rats showed the reset strategy in numerous different tasks.

Afterwards, they trained Recurrent Neural Networks (RNN), which constitute a useful tool to study the neural circuit mechanisms implementing the computations required to solve sequential tasks [47], [48]; in the same conditions as rats. RNNs were able to fully exploit the symmetry of the 2AFC task, adopting the reverse strategy and therefore outperforming rats, and their GLM transition weight was negative after error trials (Fig. 4, left).

They hypothesized that the difference may reside in the nature of the task, in RNNs the task is well defined and constrained, nothing else than the task is affecting them. However, the brain of rats evolved in very complex environment, and thus is designed to deal with different statistical structures. RNNs were pre-trained in a more ecological environment where they had more alternatives to choose, therefore mimicking real world scenarios. These RNNs pre-trained on ecological tasks showed the same reset strategy as rats (Fig.4, right).

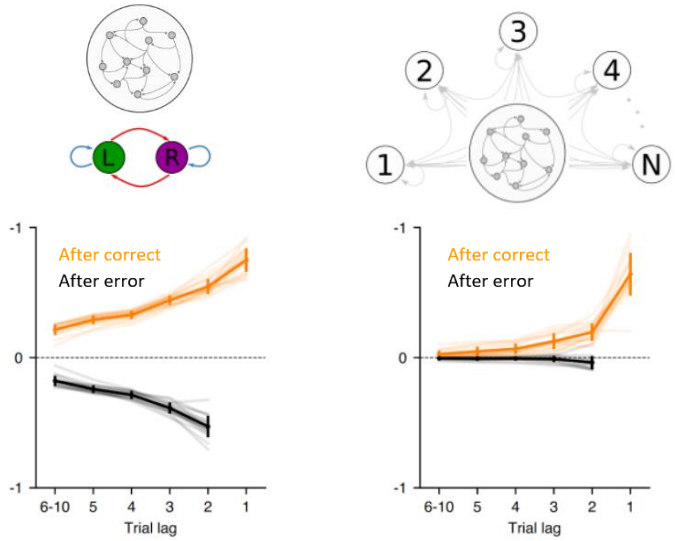


Figure 4. GLM transition weights for a 2AFC limited (Left) and a more naturalistic (Right) environments.

They concluded that the reset strategy originates from evolution as adaptation to a dynamic environment. The reset strategy, indeed, is the result of a structural prior which adapted to an environment with multiple alternatives.

2.2.3 Proactive and reactive accumulation-to-bound processes compete during perceptual decisions

This is a study with a different focus conducted by Lluís Hernández-Navarro et al. [15] and published in 2021.The main objective was to provide an alternative model to the standard ones regarding the way perceptual decisions are taken for rats. These standard models englobe Drift Diffusion Models (DDM), and postulate that a response is triggered in reaction to stimulus presentation when the accumulated stimulus evidence reaches a decision threshold. This architecture excludes the possibility that responses are generated proactively at a time independent of stimulus. Instead, the authors suggest that evidence accumulation and motor initiation, commonly viewed as serial processes, might occur in parallel as decoupled processes. For this study, rats were trained in the same 2-AFC task.

In Fig. 5 we can see both processes acting in parallel, Evidence Accumulation (EA) leads to a Reactive response: subjects listen to the stimulus and initiate a motor response only after gathering enough evidence about the right choice. Whereas Action Initiation (AI) leads to a Proactive response: initiated regardless of the integration of the stimulus. On a certain trial, participants are more likely to perform a proactive response if they can anticipate the stimulus, for example driven by choice bias, or if they are under a strong time pressure.

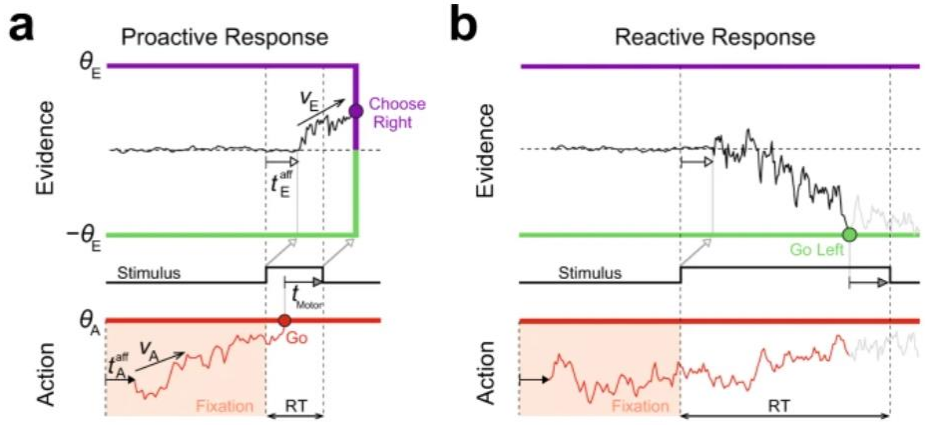


Figure 5. In (a) AI reaches the Go bound first, triggering a proactive response. In (b) EA process reaches a decision bound first, setting both RT and choice; the AI process plays no role.

The authors defined proactive responses, which are so fast the subject could have not processed the stimulus, as express responses. These rapid answers occur before the modulation onset, where timing is independent of the evidence accumulation, but choice depends on stimulus. Since the fixation period is constant, animals can predict the time when the stimulus starts, and they can prepare the motor action; in this way they respond rapidly after stimulus onset.

Henceforth, they developed the Parallel Sensory Integration and Action Model (PSIAM). This model contains a standard Evidence Accumulation process that integrates stimulus evidence over time and that is bounded by left and right decision bounds, i.e., a standard DDM; and an independent Action Initiation process (AI) which reflects the preparation of a response. The first process reaching bound initiates the response.

2.2.4 Investigating sequential effects in humans performing a 2AFC psychophysical task

This Bachelor Thesis by Debora Lombardo [49], developed in 2022, is the precursor of my project. The author designed a psychophysical task adapted to humans, based in the work with rats of Hermoso-Mendizábal et al.[12], so results can be compared, even if they come from distinct species. In the task, participants are presented on each trial with an auditory stimulus, and they are required to necessarily choose between two responses, which are left or right, depending on the strength of the stimulus, since the white noise sound presented to the subjects has different distributions on the left or on the right side. Subjects receive feedback at each trial, so this is something to consider regarding history trials. By being a simplified environment with only two options, it is possible to correctly measure all the needed values such as hit rate, misses, time of reaction etc. and evaluate the influence that distinct factors may have. Moreover, the interface design has been created in this way to simulate the trajectories in rats' studies. The task presents trial-to-trial correlations, as in the ones done by Molano-Mazón et al. [14]. These were created using a two-sided Markov chain in which the probability of repeating the same correct side P_{rep} was 0.8 in repeating context and 0.2 in alternating context (Fig. 7, right).

The trial workflow of the task:

1. The subject presses and maintains pressed the central gray button for 500ms (Fixation Time, Fig.6).

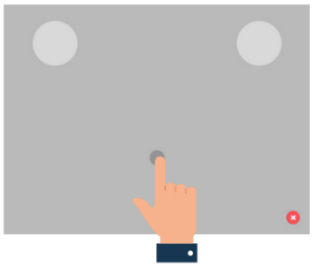


Figure 6. The figure shows the main interface of the task. The two light grey circles represent left or right choice; the dark grey one is the circle participants have to move towards the light grey ones in order to indicate which side they heard the sound mainly coming from.

2. The auditory stimulus begins and lasts up to 500 or 300ms (Response time, Version 1 and Version 2 of the task, respectively). In this time, the participant must answer. Possible cases regarding response timing:
    a. If the response was before ending the fixation time (fixation break) or after ending the response time (missed trial), the screen will turn yellow, and the trial will count as invalid.
    b. If the response is in the correct time range, during the Response Time, the screen will turn green or red, for correct or incorrect responses respectively.

In the following figure (Fig. 7) we can see an example of a correct response trial workflow for a response time of 500ms, and two examples of different blocks.

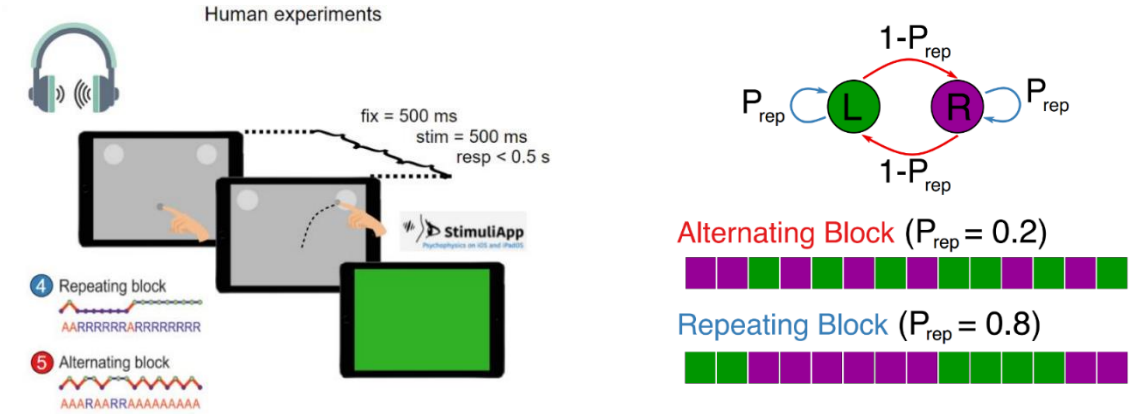


Figure 7. Left: task workflow and design for Version 1 (Response time: 500ms). Right: Markov chain with the two types of blocks (top) and a chronicle possible presentation of the correct choices (bottom).

3. Market analysis

In this part, the market involving the context of the project will be analyzed.

3.1 Sectors involved

Psychophysical experiments have been conducted for the study of perception and cognition in neuroscience. The study of how perceptual processes affect subject's behavior is a matter of interest for neuroscientists and psychologists. Moreover, several other categories of researchers are hired to contribute to their realization, such as back-end and front-end programmers, neuropsychologists, etc.. The findings of these studies can be further applied to, for example, in marketing, where some studies showed that the use of neuroscience tools to study consumer behavior and the decision-making process in marketing has improved our understanding of cognitive, neuronal, and emotional mechanisms related to marketing-relevant behavior [50], [51]. Therefore, these experiments contribute to generate new fields of study, such as neuromarketing, following the previous example. At the same time that a company can benefit from the studies with the adaptation of its marketing strategies according to the consumers, the scientific community can take profit and learn from the results [51].

In addition, it constitutes a relevant basis for the investigation of neurological diseases: indeed, inquiring which behaviors are linked to a specific pathology allows to investigate which areas of the brain are not functioning correctly in those cases. These studies can help to develop early detection or control processes for a certain pathology. Therefore, these studies can be addressed to clinical and biomedical institutions and pharmaceutical or engineering companies that specifically focus on neuroscience and the study of neurological diseases.

To sum up, this project is aimed to research, to get a better understanding of the human brain in perceptual decision-making situations. Therefore, it is directed to whoever wanting to continue my line of work, helping in the contribution of this field, with obvious advantages.

3.2 Research distribution

Regarding research by countries, United States leads the world rankings of most documents published of the Neuroscience field, followed by United Kingdom and Germany, whereas Spain is in 12th position [52]. However, United States has a low international collaboration percentage, while the others have a higher, led by United Kingdom and followed by Spain in 2021 [53]. Regarding Barcelona, the Barcelona Computational, Cognitive and Systems Neuroscience Community (BARCCSYN) [54] englobes research in neuroscience in Barcelona, with 21 public and private research groups.

From private research, one of the main contributors to the topic is Google [55], who joined DeepMind [56] in 2014, a project based in the understanding of the human intelligence using computation, creating detection and prediction algorithms, altogether with Deep Learning cognitive models. Neuroscience is one of the main research focuses of this project [57], investigating human decision-making from a computer interface perspective, for example in a virtual reality environment [58].

3.3 Applications

In the pharmaceutic industry it has been shown that some drugs can have side effects on humans, for example, Diazepam and opioids can increase the Reaction Time [59], [60]. Hence, psychophysical tasks should be used to determine the impact of these medicines in decision-making situations. These kinds of tasks have been also used in the alcohol industry, to see

quantitatively how it affects to the decision-making process and to the individuals' reaction time [61]. In hospitals or clinics, there are many applications, for example aiding in the diagnosis of several disorders, such as Attention Deficit Hyperactivity Disorder (ADHD) [62] or Alzheimer's Disease [63]. In sports, there are several tests to evaluate Reaction Times. For example, S.I. Instruments [64] have the Multi-Operational Apparatus for Reaction Time (MOART) system, with which simple reaction time tasks can be employed such as Go/No Go tasks for the study of higher centers of the brain, and more complex discriminate reaction time tasks to study cognitive processing [65]. Also, there are some used for the improvement of these capacities, such as Batak [66], which provides a piece of equipment specifically designed to improve reaction, hand eye co-ordination and stamina by enabling sportsmen and women to train under simulated 'sports like' conditions. On the other side, it can be used to measure athletes' doping attitudes [67], as proved by the Substance Abuse Treatment, Prevention, and Policy Journal [68], which encompasses research concerning substance abuse, with a focus on policy issues.

3.4 Market's historical evolution and future perspectives

Interest in the measurement of human reaction time apparently began in 1865 [69]. There were sporadic investigations of the relations between age and RT until about 1950, when interest in this topic increased because of an assumption that an individual's RT might be informative about the status of his or her neurological system [70]. 1985 was a landmark year for research on motion perception [71], and led to a huge increase in research on this topic until now. However, it is a field which is barely 50 years old, considering publication density, and therefore there is still much potential behind. In Fig. 8 the evolution of articles published, with the topic *Psychophysics*, (in logarithmic scale) can be seen. The data used is from PubMed [72].

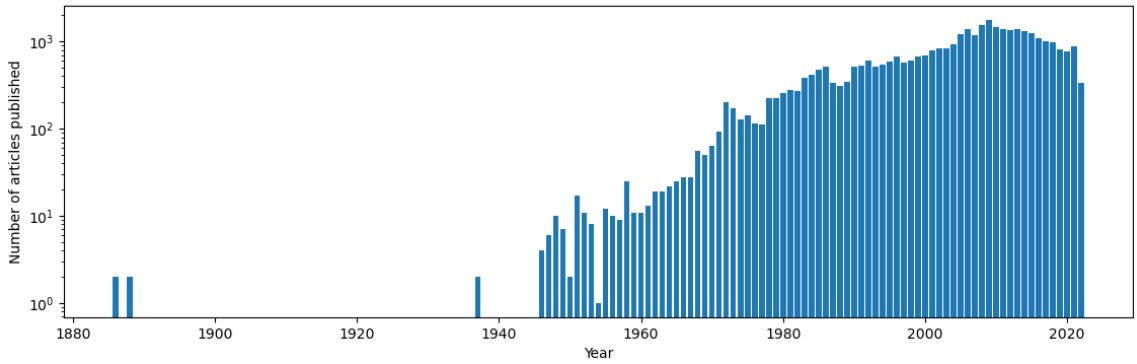


Figure 8. Number of articles published evolution over the years. Data from PubMed [70].

All the applications commented before are new, and are part of a fresh field to study, with big potential and a vast number of opportunities. A future expectation is that more and accurate applications and software for mobile devices will be released, becoming the first method used in psychophysical experiments. Other technologies can be used as well, such as a facial recognition system controlled by the main camera, that, for example, can track eye movements: this would be useful for a wide range of experiments, paving the way to further analysis.

4. Conception Engineering

This section describes the approaches and methods that were implemented in this project to achieve its objectives. The choice of these proposed solutions among the possible solutions is detailed below.

4.1 Brief summary of the methods and solution determination

The first part deals with the previous learning for understanding neuroscience topics involving this project, so it can start and be developed dynamically. Afterwards, the experiments will be performed, and data will be acquired and extracted, followed by its processing and analysis. In Fig. 9 we can see a general conception schematic with the main topics to be conceived as an interrogation mark. These are:

- The experiment: englobes how is the task performed and which components of the design should be selected to optimize the task.
- Data processing: comprehends the programming language used and its compiler.
- Missing data: comprises how missing values are treated.
- Methodological choices: encompasses the analysis to be performed and the variables to be studied.

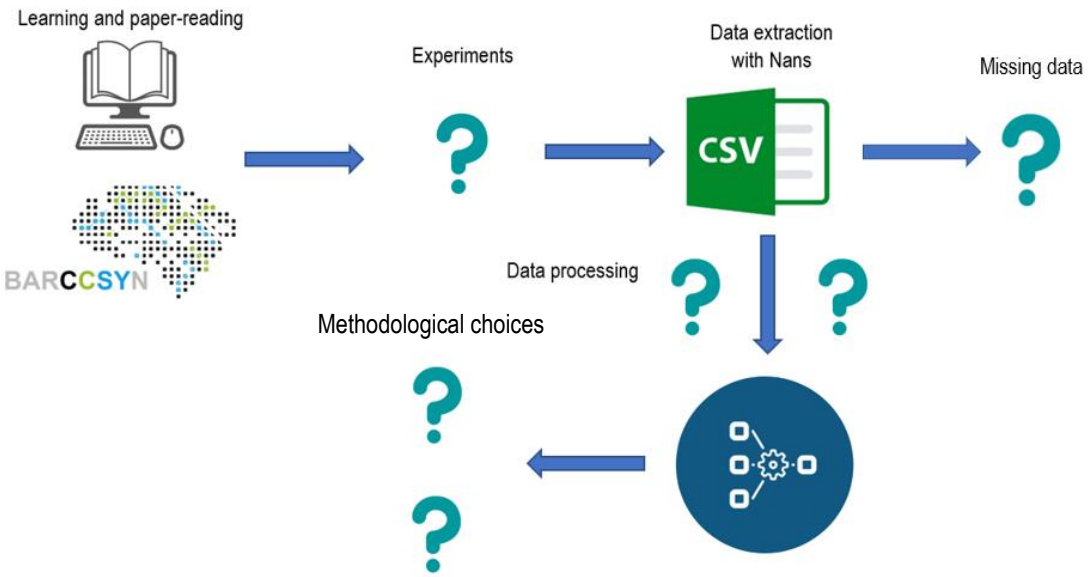


Figure 9. General conception schematic with main topics to be discussed marked with an interrogation mark (?).

4.1.1 Methodological choices

As the task makes possible the extraction of a vast amount of data, topic selection must be performed to optimize computational time and focus the project on a certain area. The main focuses are:

- Reaction Time (RT): from the RT we can study the Proactive responses of human subjects, analyzing the proportions of them for each participant. Afterwards, PSIAM model [15] can be fitted to quantify these responses, and proof that these are existent also in humans. Duration time: 3 months.

- Transition biases: analyses of psychometric curves [12], [49] and computing a General Linear Model (GLM) [12], [14], to quantify the previous trials importance and therefore see which strategy uses each subject. Duration time: 3 months.
- Trajectories: subject fingers' trajectories can be extracted, and from them the Change of Mind (CoM) can be studied. The CoM is basically when the subject starts going to one side and changes drastically to the other, changing the response abruptly. This could give information of the confidence of each subject, together with the individual influence of the stimulus evidence. Duration time: 4 months.

Studying the time for each focus, we will aim to analyze transition biases, since this will follow Debora's work [49]. As second priority, we will also study Reaction Times, since the time to do it is less than for trajectories. However, due to time limitations we may not be able to finish the project, leaving an opened line of work.

4.1.2 Coding language

Since there are different languages available, and all are powerful enough for the project, here we can find a detailed explanation of each one:

- Matlab: an abbreviation of "MATrix LABoratory", developed by Mathworks [73], combines a desktop environment perfected for the iterative analysis and design processes with a programming language that expresses matrices and arrays mathematics directly. The license is private, but since I am a Universitat de Barcelona student, it can be used for free. If that was not the case, it would cost up to 2000€. The toolboxes used, which are available with the license, would be Statistics and Machine Learning Toolbox and Signal Processing Toolbox. The releases used would be R2021b and R2022a, corresponding to the versions and release dates 9.11, September 22nd of 2021 and 9.12.0, March 9th of 2022 respectively. I have experience with Matlab, mainly in signal and image analysis. There is a great community and a lot of documentation available, and data visualization is user-friendly, with many options available.
- Python: Python is a high-level, interpreted, general-purpose programming language. It is free and so are the libraries. Its design philosophy emphasizes code readability with the use of significant indentation [74], [75]. The toolboxes used would include *matlplotlib*, *numpy*, *pandas*, *sklearn*, *scipy*, *seaborn*, among others. It could be interpreted using Jupyter Notebook or Spyder. The version used would be 3.8, with release date Sept. 24th, 2020. I have a lot of experience with it, in the areas of data processing, analysis and visualization, signal processing and Deep Learning. Many different file types can be read easily, for example Excel, CSV or PDF. There is also a great community and plenty of available documentation; and using *matplotlib* and *seaborn* packages, data visualization is very clear and neat.
- R: is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS [76]. It would be compiled using RStudio. The version used would be 4.1.2, released the November 1st of 2021. The libraries used would be *ggplot2*, *dplyr*, *MASS*, among others. I also have experience with it, but I am not as used as to the others.

Table 1. Summary of the subjective analysis regarding programming language.

	Matlab	Python	R
License price	0-2000€	0€	0€
Personal experience	+++	++++	++
Community and documentation	++++	++++	+++
Data visualization	++	+++	++++

Since I have more experience in coding with Python, its open source, which is better if someone will follow my line of work; there is plenty of documentation available, works perfectly in mathematics and statistics environments; and data visualization is clean, Python will be the programming language used.

4.1.3 Compiler

Since Python was chosen, a computational environment must be chosen now between the following:

- Spyder is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language [77]. Not much experience with it, but it is easier to use and does not have to run the code in blocks. Also, has tools such as debugging that let the user an easier interaction with the code.
- Jupyter Notebook: Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating notebook documents [78]. The code is run in blocks, which is great when working with data frames, because you can analyze each set visually. I have more experience with it.

Considering the user-friendly tools from each one, Spyder was chosen to be the computational environment used for this project.

4.1.4 Considerations in recruiting new subjects

Because new subjects were needed to do the task and extract more data, the version of the task had to be thought. Debora [49] extracted data from the following tasks:

- Response time of 500ms, with easier stimulus (Version 1).
- Response time of 300ms, with harder stimulus (Version 2).

The second one was too easy for the subjects, which are all between 19 and 25 years. Therefore, the first one made them think quickly and be more focused on the task. A new version of the task was designed but was not applied because by implementing the same tasks as Debora, data could be joined and therefore compared.

In addition, since one of the objectives is to implement this system in a clinical environment, we had the opportunity to work with Dr. Lorena Rami, from the research group 'Alzheimer Disease and other Cognitive Disorders' of Hospital Clinic, who wanted to use the task to study for differences in subjects with possible cognitive disorder and control ones. Therefore, since their subjects' age range is from 50 to 70 years old, another version of the task had to be designed so they could adapt easily. Hence, the task implemented had a reaction time of 500ms, and same coherences.

4.1.5 Missing/Not a Number (NaN) data

Invalid trials, when the subject responded before the Fixation Time ended or after the Response Time, are saved as NaN in the variables: “choices” (whether the subject answers left or right, as 0/1 respectively) and “hit” (correct/incorrect answers saved as 0/1). The possible solutions were to delete NaN values directly or maintain and consider them in the analysis. Since we are studying trial history and its implications and influence in decision-making, we cannot manipulate the timeline by deleting NaNs, but must consider them in the analysis.

4.2 Final solution

The project will be developed with Python, using Spyder as the computational environment. The main focuses will be Transition Biases and Reaction Time, with the respective order of priority. The task implemented to the new subjects to analyze for the project, will have a Response Time of 300ms and harder stimulus (Version 2). For the subjects for the Hospital Clinic, the difference will be in the Response Time, which will be of 500ms. Nan values will not be eliminated and will be considered for the analyses.

General conception schematic (Fig. 10):

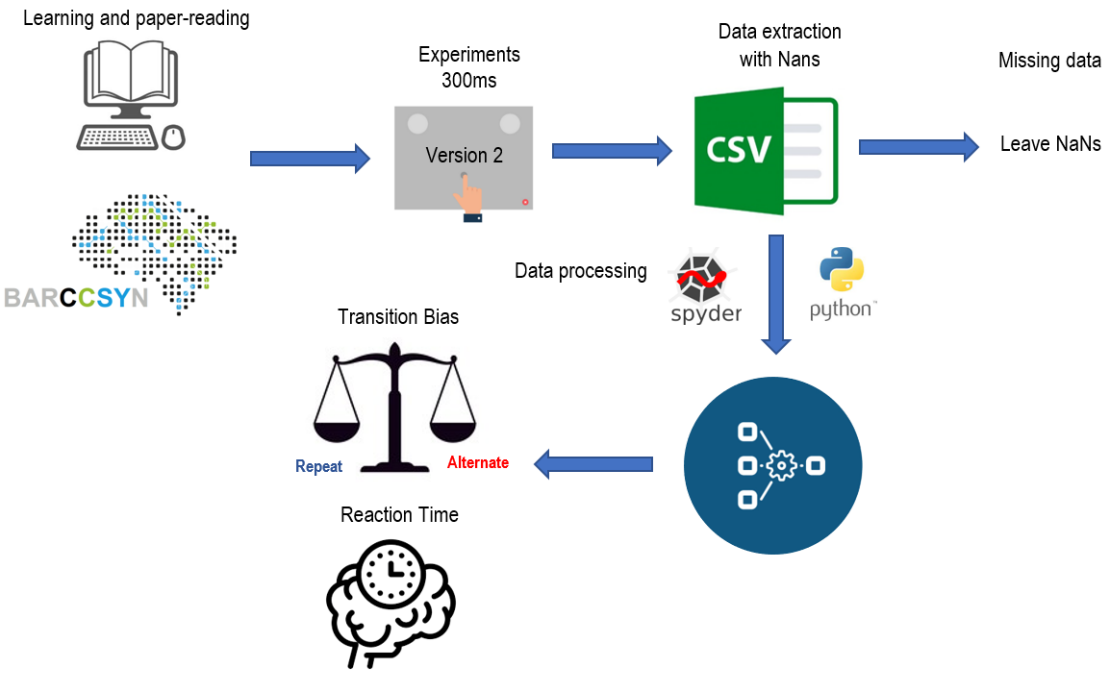


Figure 10. General conception schematic.

5. Detailed Engineering

5.1 Task

The task is the same as used in a previous study from the group [49], based on the 2AFC done to rats [12], [14], [15]. It is explained on 2.2 Previous work, in 4. 2.2.4 Investigating sequential effects in humans performing a 2AFC psychophysical task. The task is designed to be administrated to healthy subjects using a tablet.

5.1.1 Workflow

The subject is presented with a tablet whose screen has three main buttons (see Fig. 6): two light gray in the right and left top corners and a dark gray one in the bottom center. Moreover, he/she/they will use headphones provided by the laboratory, which will emit the auditory stimulus. The subject presses and maintains pressed the central dark gray button for 500ms (Fixation Time). The auditory stimulus begins and lasts up to 300 or 500ms (Response time). In this time, the participant must answer. Possible cases regarding response timing:

a. If the response was before ending the fixation time or after ending the response time, the screen will turn yellow, and the trial will count as invalid.
b. If the response is in the correct time range, during the Response Time, the screen will turn green or red, for correct or incorrect responses respectively.

Therefore, feedback was shown to the subjects through the screen color at each trial: green, yellow, and red for correct, invalid, and incorrect, respectively. This is a relevant feature of the task because subjects know their performance at each trial.

5.1.2 Structure

Trials:

A total of 2000 trials per subject, separated in 20 sections of 100. Between sections the accuracy is displayed, and subjects with less than 40% will be removed from the study.

Fixation time:

500ms

Response time:

500ms (Version 1) and 300ms (Version 2).

Reward:

Subjects receive 0.01€ for correct response. This was done to motivate participants to engage in the experiment and not give random responses.

Stimulus strength:

Stimulus strength is referred as coherence or evidence. Left stimuli have a coherence of $0 \leq x < 0.5$. For the right ones, instead, it is $0.5 < x \leq 1$. The values set in one of the versions of the task are:

- 0.2, 0.4, 0.45, 0.499: left stimuli, Version 1
- 0.8, 0.6, 0,55, 0.501: right stimuli, Version 1

- 0.3, 0.4, 0.45, 0.499: left stimuli, Version 2
- 0.7, 0.6, 0,55, 0.501: right stimuli, Version 2

Which were transformed by taking the absolute values and rounding the subtraction between these absolutes and 0.5. Therefore, we finally have:

- 0, 0.05, 0.1 and 0.3 for Version 1
- 0, 0.05, 0.1 and 0.2 for Version 2

This way the strength is standardized and is a way of knowing how informative the stimulus is, from 0 (no deviated) to 0.2 (maximum deviation, clear sound to one of the sides).

Stimulus sequence:

A two-state Markov chain parametrized by the conditioned probabilities PREP =P(L|L) and P(R|R) generated a sequence of stimulus category ck = {-1,1}, which determined the side of the reward (left/right). The stimulus strength s_k was randomly drawn from the set of values from before. The stimulus evidence was defined in each trial as the combination $e_k = c_k * s_k$, thus generating seven different options (0, ±0.05, ±0.1, ±0.2 or 0.3) (see Fig. 11).

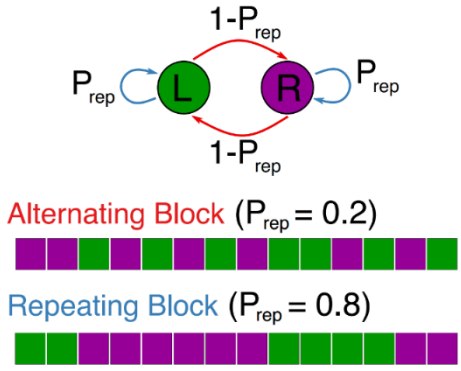


Figure 11. Two-state Markov chain (top) with examples of sequences (bottom).

5.2 Dataset

We have a total of 10 subjects for Version 1 (500ms) and 20 subjects for Version 2 (300ms). Sexes are balanced at 50%. The age ranges from 20 to 26, with mean of 22.30±1.16 years old. None of them presented auditory problems nor psychological or psychiatric disorders. A summary table can be seen in Tab. 2.

Table 2. General characteristics of the dataset.

	Version 1 (N=8)	Version 2 (N=20)	Differences
Age (mean ± SD, years)	22.25±1.20	22.30±1.14	No
Sex (Male/Female)	4/4	10/10	No
Right-handed (Yes/No)	7/1	19/1	No
Auditory problems (Yes/No)	0/8	0/20	No

5.3 Metrics

The data that the app returns in csv format is made of several features:

- Hit: Boolean variable, expresses whether the subject is correct (1) or not (0).
- Choice: Boolean variable, expresses whether the subject chose right (1) or left (0).
- Correct: Boolean variable, expresses which was the correct response, right (1) or left (0).
- Stimulus strength: which values go from 0 to 1, expressing how lateralized is the stimulus. The closest to 0.5, the more balanced and indistinguishable it is.
- Time of the answer: time that the subject lasted to answer, in seconds. It considers Fixation Time also, so afterwards it must be subtracted to get the Reaction Time.
- Motor Time: time that passes from the Reaction Time to the end of the trial, in other words, for how long has the subject moved its finger.
- Space coordinates: trajectories in time of the subjects' fingers, in coordinates X, Y, according to the screen dimensions. These values are given in millimeters, and the sample rate is 59 Hz.

5.4 Analyses

5.4.1 Transition and lateral biases

Psychometric curve:

Measuring thresholds is probably the most common psychophysical procedure in use today [79]. The psychometric function relates an observer's performance to an independent variable, usually some physical quantity of a stimulus in a psychophysical task [80], [81]. This curve is shaped by important parameters, such as biases and sensitivity to the stimulus. However, it is not possible to solve them analytically, so a fitting must be done with the results in order to extract the parameters. The proportion of rightward responses vs. stimulus evidence e is given by:

$$P_{rightwards}(e) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{\beta e + b}{\sqrt{2}}\right)\right)$$

Where $\mathrm{erf}(x)$ is the error function, also known as Gauss error function, defined as:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-x^2}dx\,, \mathrm{erf}(x) \in (-1, 1)$$

β is a sensitivity that quantifies the stimulus discrimination ability, e is the stimulus strength and b indicates the subject's tendency to repeat (b>0) or alternate (b<0). In order to analyze this probability, trials are grouped by the stimulus strength values. Subsequently, for each stimulus strength, the probability to choose the right side over the left one is calculated, which is the percentage of times that the subject has chosen right according to a certain stimulus value. Therefore, the curve can be fitted, and the parameters extracted.

Also, the proportion of repeated responses as a function of the repeating stimulus evidence is fitted and is expressed by the following function:

$$P_{repeat}(\hat{e}) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{\beta'\hat{e} + B}{\sqrt{2}}\right)\right)$$

β' is a sensitivity that quantifies the stimulus discrimination ability, and:

$$\hat{e}_t = r_{t-1}\, e_t$$

Where t is a trial and r is a response, with:

$$r_{t-1} = \{-1, 1\}$$

Representing if the response in the previous trial was left or right, respectively. For example, a rightward stimulus with evidence e_t = 0.4 after a left response implies a repeating stimulus evidence equal to $\hat{e}_t$ = - 0.4. The process is the same as before, the difference is the probability calculated, which is based on repeating responses for each stimulus strength.

Generalized Linear Model (GLM):

We built a GLM where different features, such as the current stimulus and previous history events, were linearly summed to give rise to the probability that the subject's response r_t in trial t was toward the right port [12], [14], [82]–[84].

The following regressors were the chosen for this model, trying to have the optimal number to avoid combination of effects in a single regressor. For example, the lateral regressors could be hidden in the transition ones. For each regressor two weights were computed: after correct and after error.

- Evidence: current stimulus effect.
- Lateral:
    - L+: 1 if right and correct, -1 if left and correct.
    - L-: same as L+ but with incorrect responses.
- Transition:
    - T+-: a correct response followed by an error, right/left are {1, -1} respectively.
    - T-+: an error followed by a correct, right/left are {1, -1} respectively.
    - T--: two consecutive errors, right/left are {1, -1} respectively.
    - T++: two consecutive correct answers, right/left are {1, -1} respectively.
- Intercept: fixed side bias.

In Fig. 12 we can see a general schematic of the GLM intuition, with the sum of effects from the Lateral and Transition biases, and the evidence. In this example, the subject starts with three correct Rights, which correspond to a repetitive block; however since this blocks are designed with probabilities, it changes to Left, but because it is a repetitive Block, the transition evidence is influencing the subject to choose Left again, depending of course on the stimulus evidence; but the subject's expectations are built biased to the left.

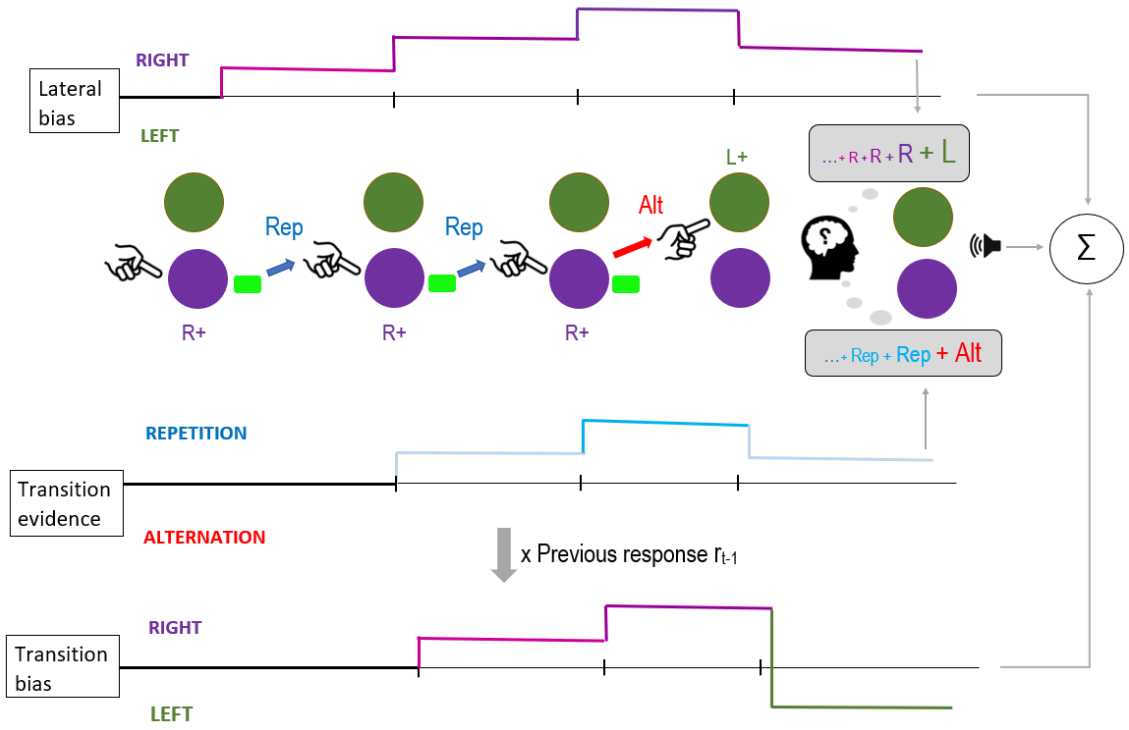


Figure 12. Generalized Linear Model (GLM) intuition. Lateral and Transition biases in top and bottom of the figure, respectively.

The probability $p(r_t=+1|y_t)$ that the response r_t in the t-th trial was Rightwards was modeled as a linear combination of the current stimulus and trial history passed through a logistic function:

$$p(r_t = 1|y_t) = \frac{1}{1 + e^{-y_t}}$$

Where the argument of the function was:

$$y_t = \beta_{stim} S_t + \sum_{k=1}^{6} (\beta_{r,k}^{+} r_{t-k}^{+} + \beta_{r,k}^{-} r_{t-k}^{-}) + \left(\sum_{o,q} \sum_{k=1}^{6} \beta_{T,k}^{o,q} T_{t-k}^{o,q} \right) r_{t-1} + \beta_0$$

The current stimulus was given by S_t defined as the intensity difference between the two tone sounds in trial t. The trial history contributions included the impact of the previous ten trials (t-1, t-2, t-3...; grouping the impact of trials t−6 to trial t−10 in one term):

$$For\ k = 6 - 10, \quad r_{t-6}^{o} = \sum_{k=6}^{10} (\beta_{r,k}^{+} r_{t-k}^{+} + \beta_{r,k}^{-} r_{t-k}^{-})$$

The terms $r_{t-k}+$ represented the previous rewarded responses being −1 (correct left), +1 (correct right), or 0 (error response). Similarly, $r_{t-k}-$ represented previous unrewarded responses being −1 (incorrect left), +1 (incorrect right), or 0 (correct response). Previous transitions were given by:

$$T_k^{o,q} = r_{k-1}^{o} r_k^{q}$$

Meaning that they were $T_k^{o,q} = +1$ for repetitions and $T_k^{o,q} = -1$ for alternations. The superindices {o,q} refer to the type of transition which depended on the outcomes of the last two

trials *t*-1 and *t*, respectively: correct-correct {+, +}, correct-error {+, −}, error-correct {−, +}, and error-error {−, −}. Finally, the coefficient β_0 represented a fixed side bias.

For rats [12] and RNN [14] the weights were the following (for the T^{++} regressor):

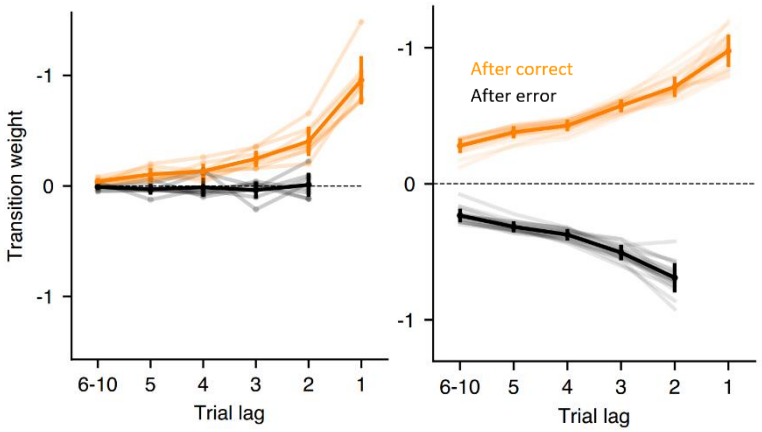


Figure 13. GLM transition weights from rats [13] and RNNs [15], showing an exponential decay.

We can see an exponential behavior in After-correct (orange) transition weights for rats (Fig. 13, left) and for both After-correct (orange) and After-error (black) weights in RNNs (Fig. 13, right). Therefore, each regressor was discretely convoluted with an exponential kernel, as in previous studies [84], to reduce the computational cost: instead of computing a value for each history trial (1, 2, 3, 4, 5, 6-10), it only had to compute one.

$$r_t^o = (f * g)[t] = \sum_{k=1}^{6} r_{t-k}^o \cdot e^{-\frac{k}{\tau}}$$

Where τ is the decay constant of the exponential. First we fitted the model using a fixed $\tau = 1$. And then we fitted different values of it: 0.5, 1, 2, 3, 4, 5 using Cross-Validation with five folds.. This indicated how many trials influenced each subject. Therefore, after convoluting with a specific τ, we get:

$$y_t = \beta_{stim} S_t + \beta_r^+ r_t^+ + \beta_r^- r_t^- + \left(\sum_{o,q} \beta_T^{o,q} T_t^{o,q} \right) r_{t-1} + \beta_0$$

Which reduced computational time and increased algorithm's robustness. Therefore, we have only one value for each regressor. Since values from invalid trials are stored as not a number (Nan), the convolution had to consider them, and predefined Python functions could not be used, therefore, it had to be done manually with a Boolean mask that considered Nans:

$$mask(r_{t-k}^o) = \begin{cases} 0, & r_{t-k}^o = Nan \\ 1, & r_{t-k}^o \neq Nan \end{cases}$$

$$r_t^o = \sum_{k=1}^{6} r_{t-k}^o \cdot e^{-\frac{k}{\tau}} \cdot mask(r_{t-k}^o)$$

By using this, the computational time raised, but it ensured that the GLM did not receive any Nan value. This was done because Nan could not be replaced by any other value, since the trial history is being accumulated and therefore an artifact would be added if replaced.

The number of trials for our subjects is limited compared to works with rats, which undergo an average of 50000 trials each individual [12]; or with RNN, whose number of trials is in the order of millions [14]. Therefore, statistical significance of the weights, which shows whether they contribute or not in the GLM (different from zero or not, respectively), had to be computed using the Python package *statsmodel.api* [85].

Reset/reverse strategies:

As explained before (see 2.2 Previous work for details), rats presented the reset strategy: ignored previous transitions' information after error trials; whereas RNNs fully exploited the task adopting the reverse strategy: learning from errors. Here the GLM transition weights after error will be studied to see if they are zero (reset strategy) or negative (reverse strategy).

Reset Index:

The Reset Index (RI) quantifies the extent to which an agent follows the reset strategy. It is computed as:

$$RI = 1 - \frac{Tr_{after-error}}{Tr_{after-correct}}$$

Where $Tr_{after-error}$ and $Tr_{after-correct}$ are computed as the absolute value of the sum of transition weights $w^{T}_{k,+,+}$ with k = 2, …, 6-10. Therefore, for values near to one, subjects are more likely to present reset; whereas for values far from one, they are expected to present the reverse.

5.4.2 Reaction time

Proactivity and express performance:

Proactive (express) answers are the ones done by the subject before being able to receive, process and execute a motor action. The simple auditory reaction time is one of the fastest reaction times and in normal population is rarely less than 100ms [86], [87]. However, when people have a bias, they can make a motor plan even before the stimulus starts and these decisions' timing is independent of the evidence accumulation, but choice depends on stimulus [15]. In these situations, the Reaction Time (RT) drops before this threshold of 100ms. Therefore, the RT distribution was analyzed to see whether subjects responded before a threshold of 100ms. Afterwards, the trials that were under this threshold were selected and the accuracy was computed (Express performance) conditioned by the stimulus strength, to see if the stimulus still influenced in the decision. The RT histogram and its density function (proportion) were computed to see if there was a proportion of reaction times before the threshold.

Tachometric curves:

These curves show how the accuracy changes depending on the reaction time, conditioning on stimulus strengths: comparing each accuracy for each different stimulus strength. Since data was discrete (sample rate ~59Hz, reaction time precision of 17ms), a binning of 18ms was done.

Reaction time and motor time dependence with stimulus strength:

This dependence was analyzed to see whether the subjects perform the task systematically, that happens when RT does not depend on stimulus strength, because participants respond periodically. On the other side, it is intuitive to think that for most informative responses, there would be less doubt and therefore lower Motor Time.

Cumulative distribution functions and modulation onset:

The cumulative distribution function (CDF) of the reaction times' histogram was computed. It was obtained by performing the cumulative sum of the probability density function (from the RT histogram). We computed the CDF in two scenarios: trials presenting more informative stimulus (strengths of 0.05, 0.1 and 0.2) and the ones which does not give much information (strength of 0). By this means we could see if the reaction times were independent or not from stimulus strengths. Then we assessed the modulation onset, which is the point where the two curves start to be independent one from another. For each time t, we computed a one-tailed Kolmogorov–Smirnoff test [15], [88] comparing the RT cumulative distributions for trials with strongest and weakest stimulus evidence (stimulus strength $s = 0.2$ versus stimulus strength $s = 0$), and excluding all reaction times larger than t. For each subject, we defined the modulation onset as the minimal value of t at which this comparison became significant ($p < 0.05$). On the other hand, since data is discrete (reaction time precision of 17ms), the CDF was expected to be stair-like. Consequently, the CDF was mean filtered, which consists in replacing each data value with the mean value of its neighbors, including itself. It was done with a window size of 17 because it coincided with the Reaction Time precision and thus returned a smoothed CDF useful for visualization.

6. Results and discussion

I will now present and discuss the results of the analysis. All analyses were performed to Version 2 (300ms) subjects.

6.1 General statistics: Humans learn to do the task

First I will describe the general stats, such as accuracy and valid trials. In Fig. 14 we can see an example subject, which increases the proportion of valid trials, meaning that the participant learns to accommodate to the timing of the task. In general, the first 200 trials are the ones where participants are still learning (for this reason this period will not be taken into account in the analysis). This participant has a mean performance of 85%, and a percentage of valid trials of 97.2%. However, there are sudden drops in accuracy most likely due to breaks taken by the subject during the experiment. This is a feature I have observed in most subjects (data not shown).

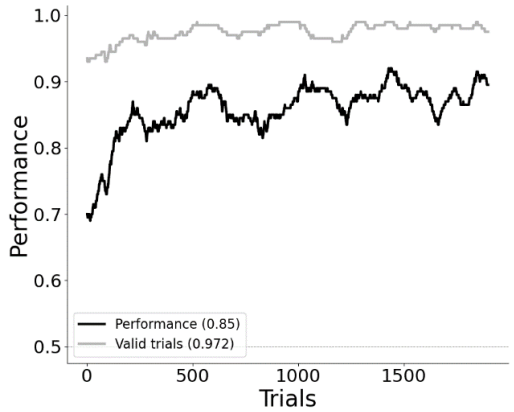


Figure 14. Performance over trial index. It can be seen a positive progression, stronger the first 200-300 trials.

In Fig. 15 we can see the mean accuracy of all subjects and its distribution, with a median of 85.6%. The minimum mean accuracy is 69.5% and the maximum 90.1%. The group's mean accuracy is 83.6±5.8 %.

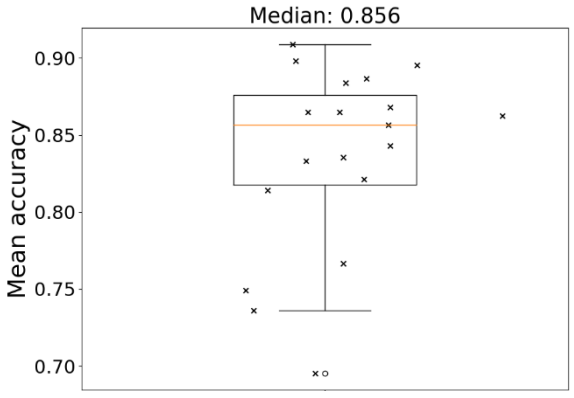


Figure 15. Mean accuracy for all subjects. Median: 85.6%

From here we can conclude that humans learn to do the task.

6.2 Transition bias

6.2.1 Subjects do not show lateral bias

We first studied lateral biases, which is the tendency to respond leftwards or rightwards, and can be seen in the psychometric curves on the right/left space, which compute the probability of choosing right over the evidence of right. Both biases will be computed in the GLM, which will provide a more precise quantification than extracting them by fitting a complex curve with only seven points.

First we visualized the psychometric curves in the right/left space for an example subject (Fig. 16). We have a subject (left, Subject 5) that presents a bit of lateral bias to the left, and another (right, Subject 19) for which the lateral bias seems to be null. Both learnt the task correctly.

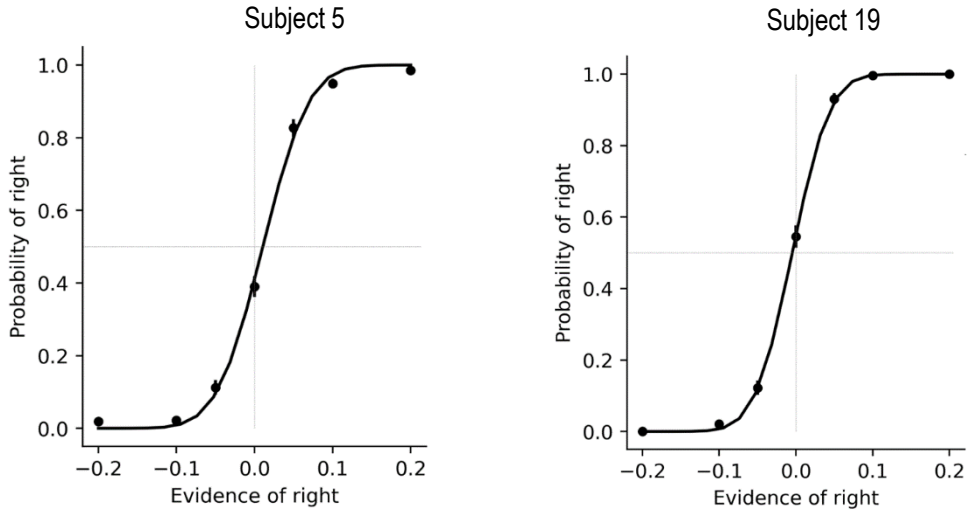


Figure 16. Psychometric curves on the lateral space: we can see the data points with error bars and the fitted curve. Left: showing bias to the left. Right: not showing bias.

Lateral bias was seen in a low number of subjects and was not significant.

6.2.2 All subjects present transition bias

I then studied the transition bias, which is the tendency to repeat/alternate given a certain experience. This bias can be visualized in the psychometric curves on the repeating/alternating space, which compute the probability of repeating the previous choice over the repeating evidence: the original evidence with a negative sign if the repeating side is the left one (Fig. 17). We will see the same subjects as before (top, Subject 5; and below, Subject 19). Transition bias is present in both subjects after correct trials (left), whereas after error curves they show a mix of reverse (see Fig. 3), alternating (red) curve is over the repeating (blue) one; and reset, both curves cross at zero evidence and 0.5 probability.

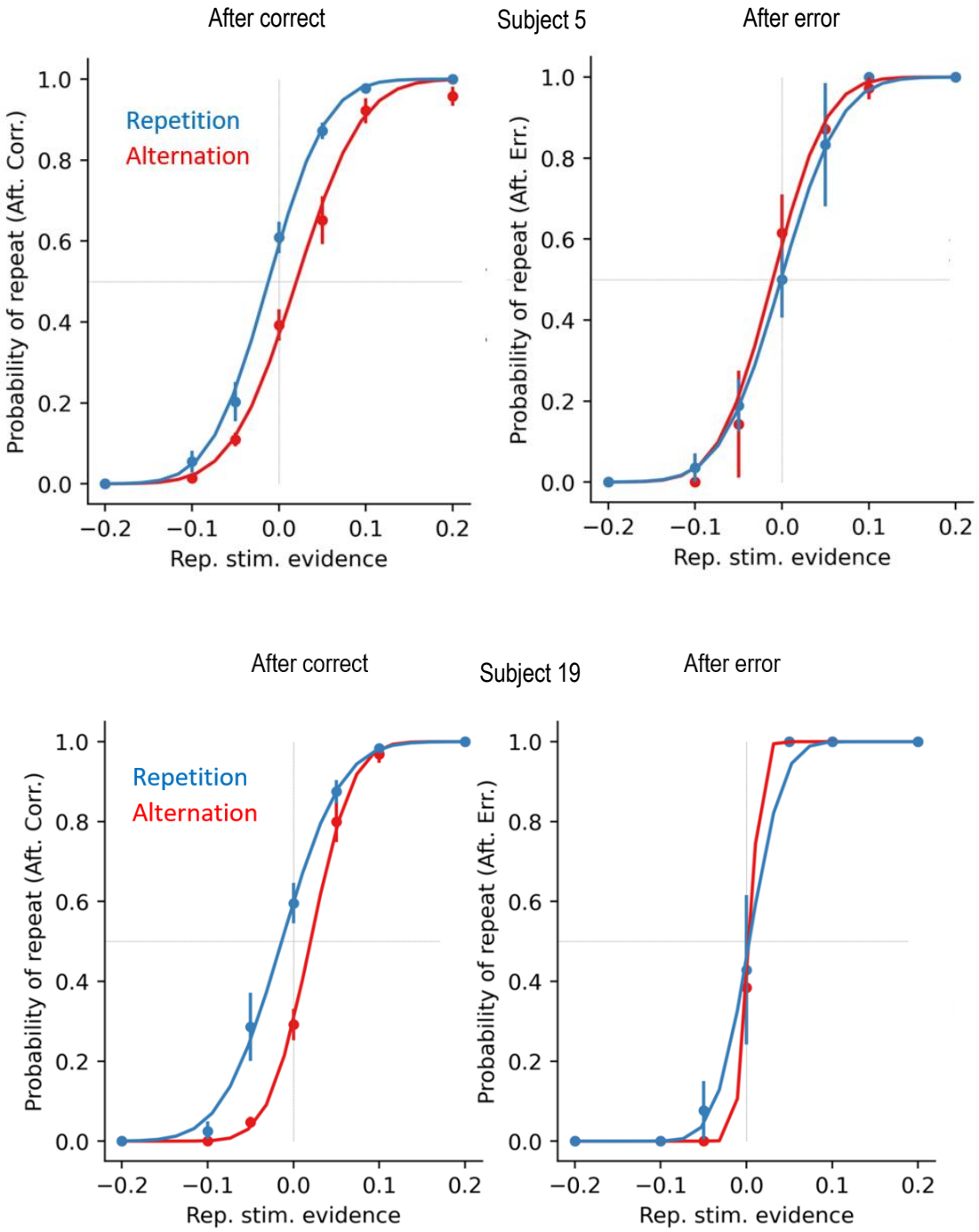


Figure 17. Psychometric curves in the transition space: we can see data points with error bars and psychometric curve fitting. Top and bottom, left both subjects show transition bias; Right: subject 5 seems to adopt a stronger reverse strategy, whether subject 19 adopts a weaker version. Subject 19 has steeper curves, meaning that the task was well understood.

Now the group's psychometric curves are shown in Fig. 18. We can see that all subjects presented transition bias, whereas the strategy adopted cannot be studied in here, since it happens the same as before: reset and reverse are combined and therefore the group's curve is a mix of both..

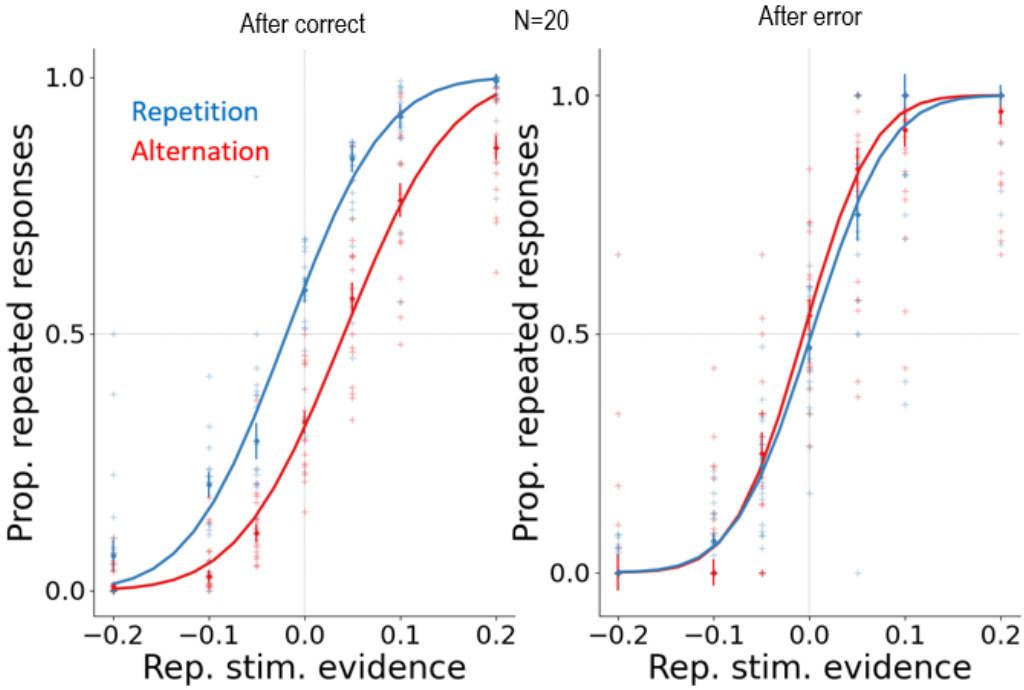


Figure 18. Group psychometric curves in the transition space. Left: all subjects showed transition bias. Right: there is a mix of both strategies (reset and reverse), but the presence of a reverse is noted.

In order to see more clearly the transition biases, the values were extracted from the psychometric curves' fitting (see 5. Detailed Engineering) and visualized through a boxplot of for each situation (Fig. 19). These scenarios are the following:

- Repetition block after correct trial (Rep. ac)
- Alternation block after correct trial (Alt. ac)
- Repetition block after error trial (Rep. ae)
- Alternation block after error trial (Alt. ae)

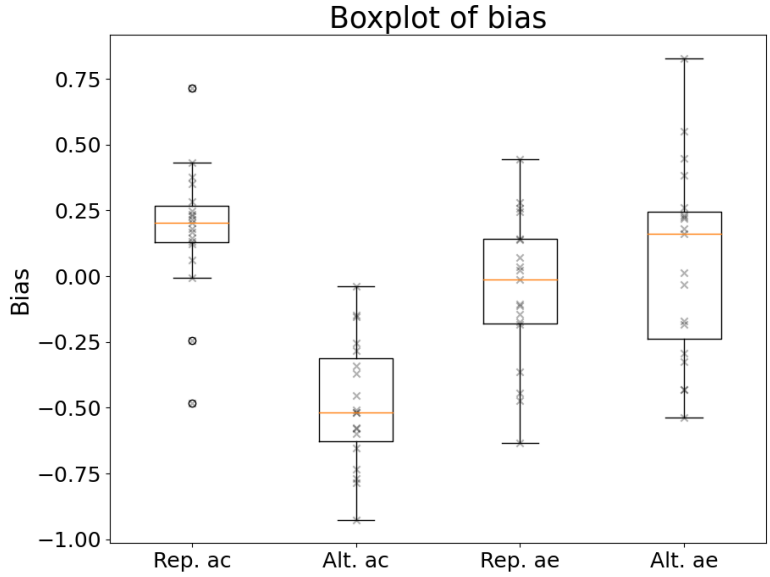


Figure 19. Boxplot of bias. Clear behaviors after correct, showing transition bias. However, after error the strategies are unclear.

And it can be clearly seen that after correct trials the transition bias is present: positive for repetition and negative for alternation. However, it is not so clear after error trials. Therefore, for visualization, a meta-subject was created by joining all subjects that most-likely presented the reverse strategy. In Fig. 20 we can see that after correct responses all present transition bias, and after error responses, the reverse is even more clear.

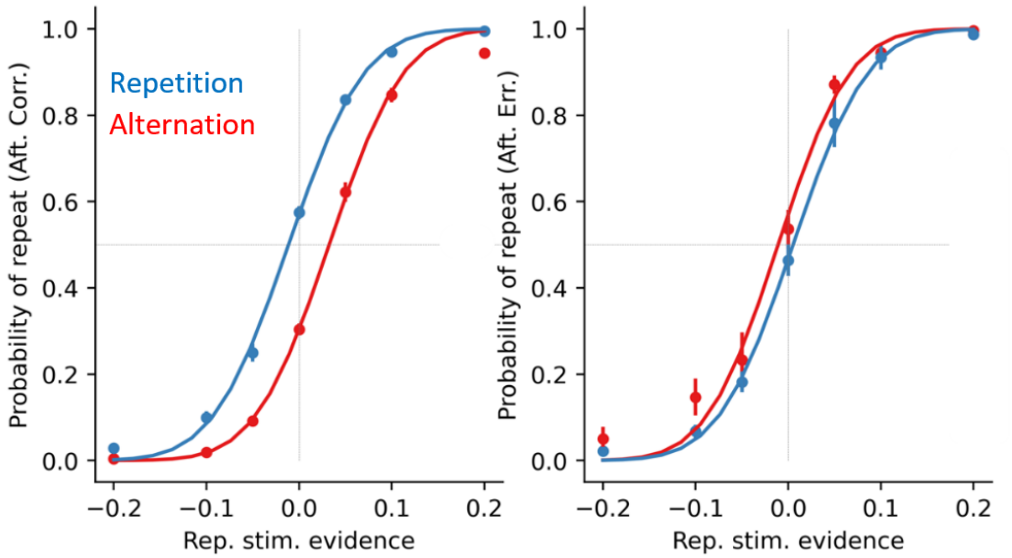


Figure 20. Psychometric curves from the subjects that presented the reverse strategy grouped as one single subject. Left: clear transition bias, stronger on the alternation blocks. Right: clear reverse strategy.

However, this is only for visualization, since these curves can be combining different factors, it is not accurate extract the transition bias or the strategy from them.

6.2.3 The reverse strategy is the predominant

To quantify these biases, I used a GLM (See 5. Detailed Engineering for details), this GLM separates the influence of different factors (regressors) and extracts only the transition bias wanted1. Once performed, we could extract the transition weight (T^{++}) and plot it for visualization, first with a fixed characteristic decay constant for the exponential decay (τ=1, Fig. 21 left). The constant was fixed to 1 because it approximated what was seen in rats and RNNs. Afterwards, we computed the trial lag weights for an individual-specific constant (Fig. 21, right), selected using cross-validation.

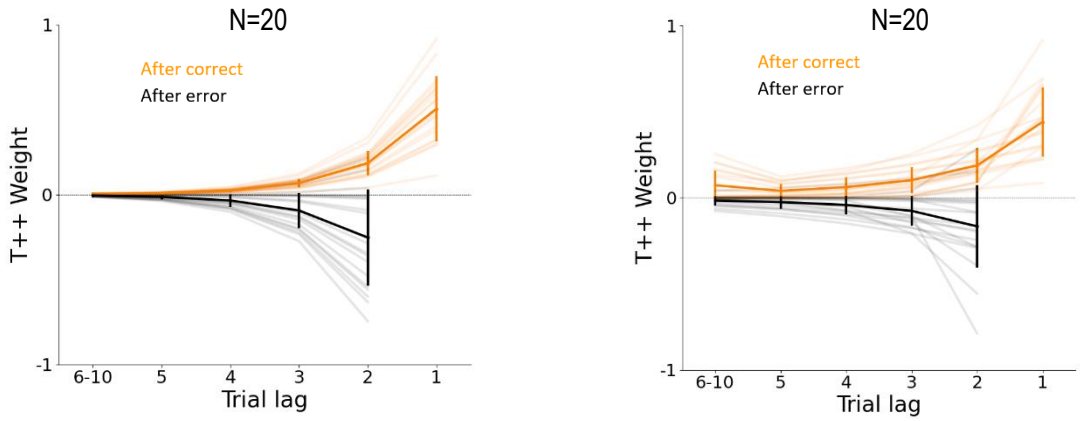


Figure 21. Transition (T++) weights from the GLM. Left: fixed exponential decay constant to one. Right: fitted a constant for each subject. We see more reverse strategy than reset.

Even though for a specific constant the result is more precise, for visualization purposes we will use the fixed constant so comparisons with other studies are easier. Some subjects do the reset strategy, as rats [12] (Fig. 22, left): after error values are zero, meaning that these participants ignore previous experiences in these situations. However, the most predominant strategy is the reverse, which is more optimal. This strategy is the one adopted by RNNs [14] (Fig. 22, center), which fully exploited the task statistics and can be seen in the values after error with a negative contribution: it means the subject change from repeating to alternating, and vice-versa, in after error trials scenarios. However, the reverse in human subjects was not as strong as in RNNs, where even previous 6-10 trials affected a lot. In most subjects, the 5th previous trial influence was null.

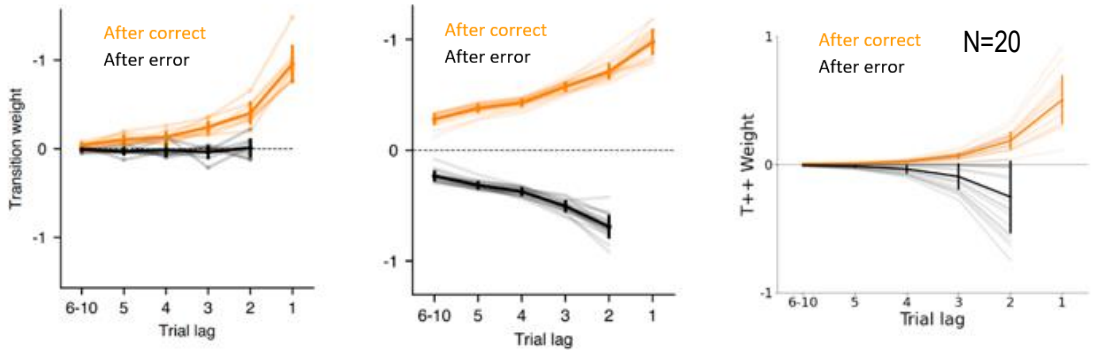


Figure 22. Left: GLM transition weights for rats [13]. Center: GLM transition weights for RNNs [15]. Right: GLM transition weights for humans.

Then, I computed the Reset Index (RI), whose values can be seen in Fig. 23. There we can see the four subjects which are above 0.8, whereas all the others are below. With mean 0.47±0.40 and median 0.56. Separating the values taking 0.8 as threshold, the upper mean is 0.96±0.02 and the lower is 0.34±0.34. Regarding the decay characteristic constant τ, it was uniformly distributed across subjects, as it can be seen in Tab. 3.

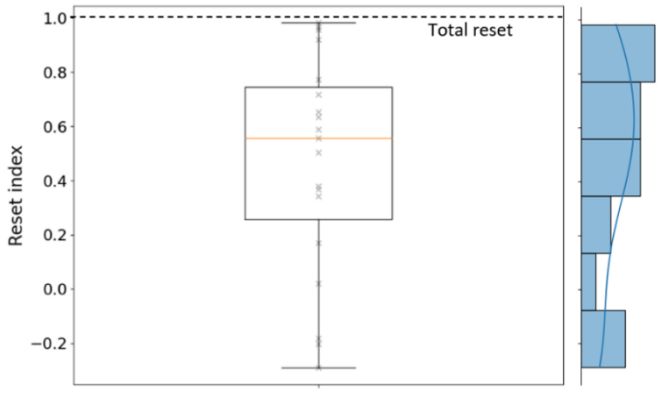


Figure 23. Reset index and its distribution.

τ	Number of subjects
0.5	4
1	5
2	4
3	6

Table 3. Characteristic exponential constant distribution.

Since these 2000 trials are separated into after correct and after error situations and the mean performance is 85.6%, the mean number of after error trials is 288. Therefore, the statistical significance of the weights (whether the weights are different from zero or not) had to be computed for each subject. We can see here a subject (Fig. 24 left, Subject 17) that presented reset: T++ is close to zero after error trials; and one that presented reverse (Fig. 24 right, Subject 12): T++ is negative after error trials. However, as said before, since there is a small number of after error trials, there was no statistical significance. There is no common strategy regarding lateral (L) weights.

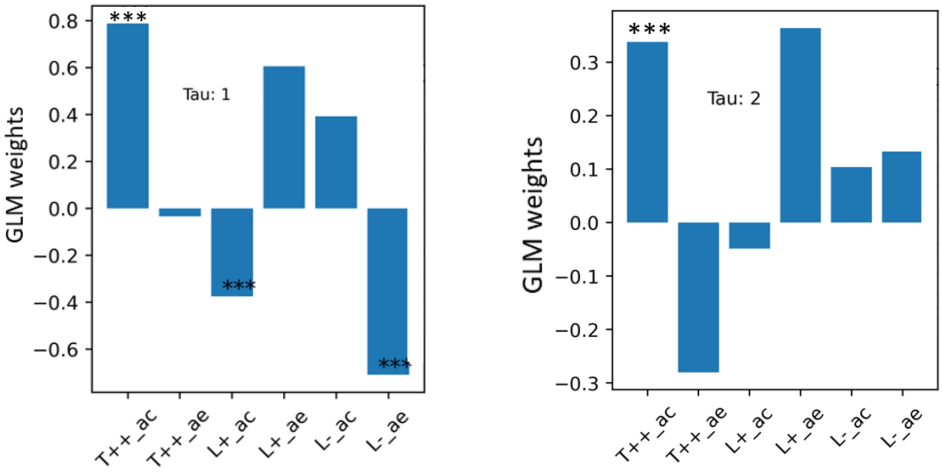


Figure 24. GLM weights and its significance for two subjects. Left: reset. Right: reverse.

Two meta-subjects joining all subjects presenting reverse and reset, respectively, were computed and the results were the following (Fig. 25, top): there was statistical significance for the weights after error for the reverse meta-subject (top-left). However, no statistical significance for the weights after error was found in the reset meta-subject (top-right),indicating that the reset strategy is also present in humans. However, we speculate that if the task was longer (e.g., 5000 trials), all subjects would present the reverse strategy. This is supported with the GLM performed on the Version 1 subjects, which had more time to think, and therefore they understood earlier the task (Fig. 25, bottom).

On the other hand, we see a clear significance on lateral biases for L+, which are negative after correct and positive after error, meaning that subjects were negatively influenced by right choices after correct answers. Moreover, statistically significant weights are seen in L+ after error, where participants are strongly influenced by previous correct choices after error. We have not seen a common strategy regarding L- weights.

To summarize, humans present different strategies for 2000 trials, ranging from full reset to a more reverse behavior, which was more predominant. However, it is thought that for more trials (e.g., 5000) they would all present the reverse strategy because for tasks with less time pressure subjects presented even more this strategy, indicating that if they did longer tasks, they all would adopt this more optimal strategy.

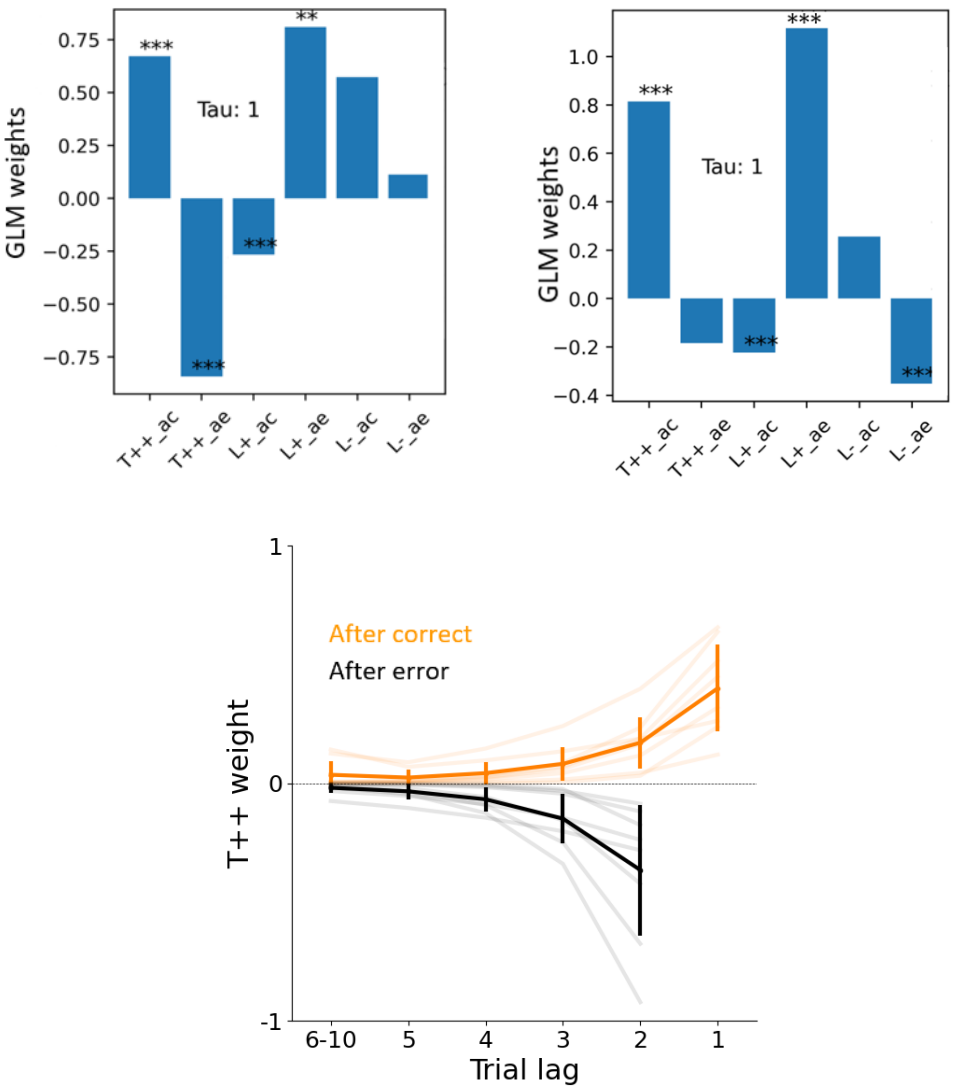


Figure 25. Top: meta-subjects, reverse and reset corresponding to left and right top parts of the figure, respectively. Bottom: GLM weights for the Version 1 subjects.

6.3 Reaction time

6.3.1 Express performance depend on stimulus strength

I also studied Reaction Times (RT). First, the distribution of them was visualized to see if there were express responses (Fig. 26, left), assumed to be the responses before 100ms after the stimulus onset (blue line). There is a great proportion of them before this threshold. Therefore, we filtered the responses with a threshold of 100ms to see the performance of the subjects in these express responses over the stimulus strength (Fig. 26, center).Importantly, subjects accuracy depended on the stimulus strength, even though subjects did not have enough time to integrate and process the stimulus and prepare a motor plan. This means that subjects prepared this motor plan even before the stimulus started, during the Fixation Time. Accuracies for express responses did not start at 50% as expected, they were increased by the transition bias, consequently we could say that subjects are influenced by the previous expectations in express responses. These results were also seen in rats [15] (Fig. 26, right) and taking the same conclusions as this study, we could say that the reaction time of human subjects is governed by two independent processes: one that corresponds to the stimulus accumulation (called Reactive) and another that is stimulus' timing independent (called Proactive).

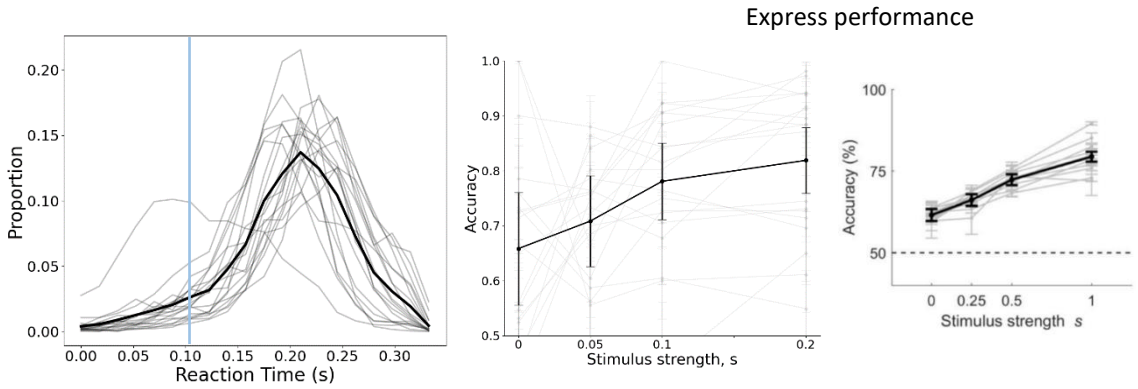


Figure 26. Left: distribution of the reaction times and 100ms threshold (blue). Center: express responses performance in humans. Right: express responses performance in rats [16].

On the other hand, tachometric curves were visualized (Fig. 27, left), conditioning in stimulus strength. As expected, for the strongest stimulus strength (0.2) the accuracy was the highest and descended according to this evidence. Performance did not start at 50% (effect of transition bias) and curves are sorted from highest to lowest stimulus strength even at the start, evidencing the same as in the previous figure. Also, for stimulus strength equal to zero (black), it can be seen that the accuracy decreased with the reaction times, meaning that when the stimulus was hard to differentiate, the longer the time to answer the lowest the accuracy. This was also seen in rats [15] (Fig. 27, right).

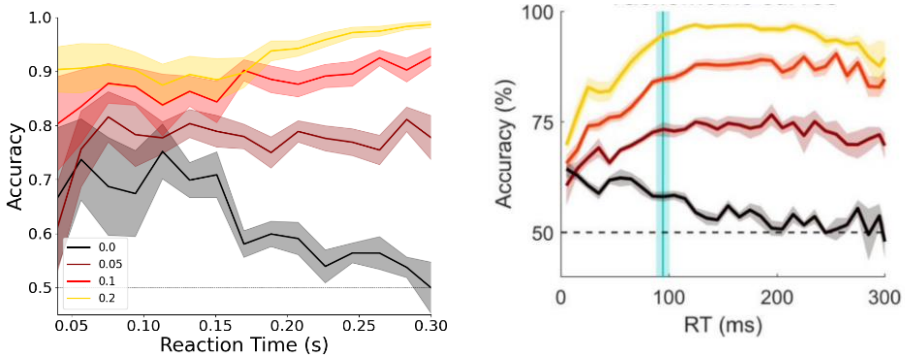


Figure 27. Tachometric curves. Left: humans. Right: rats [16].

6.3.2 Reaction time in proactive responses is not modulated by stimulus strength

Finally, I analyzed the dependence of the Reaction and Motor Time (RT and MT, respectively) with the stimulus strength (see Fig. 28). Intuitively, the MT negatively depended on the stimulus strength, because the higher the auditive difference, the easier and faster to respond. However for the RT, since the task has a limited sampling rate, significant changes in RT could not be perceived and therefore it seems that it is independent from stimulus strength. But we will see with the RT Cumulative Distribution Function (C.D.F.) that it is not exactly like that.

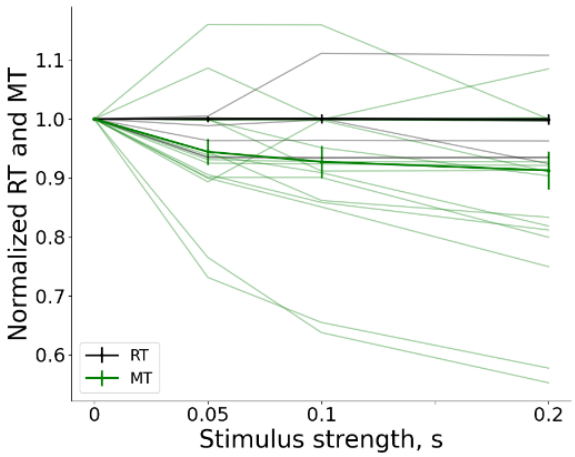


Figure 28. RT (black) and MT (green) over stimulus strength.

After analyzing the CDFs of the Reaction Times (Fig. 29), we can see that before the modulation onset (mean: 78ms, SD: 21ms. Fig.29, top-right and bottom: blue line) the reaction times did not depend on the stimulus strength (Fig.29, bottom), but after this point it did. Concluding that these proactive responses were independent from the stimulus, as in rats.

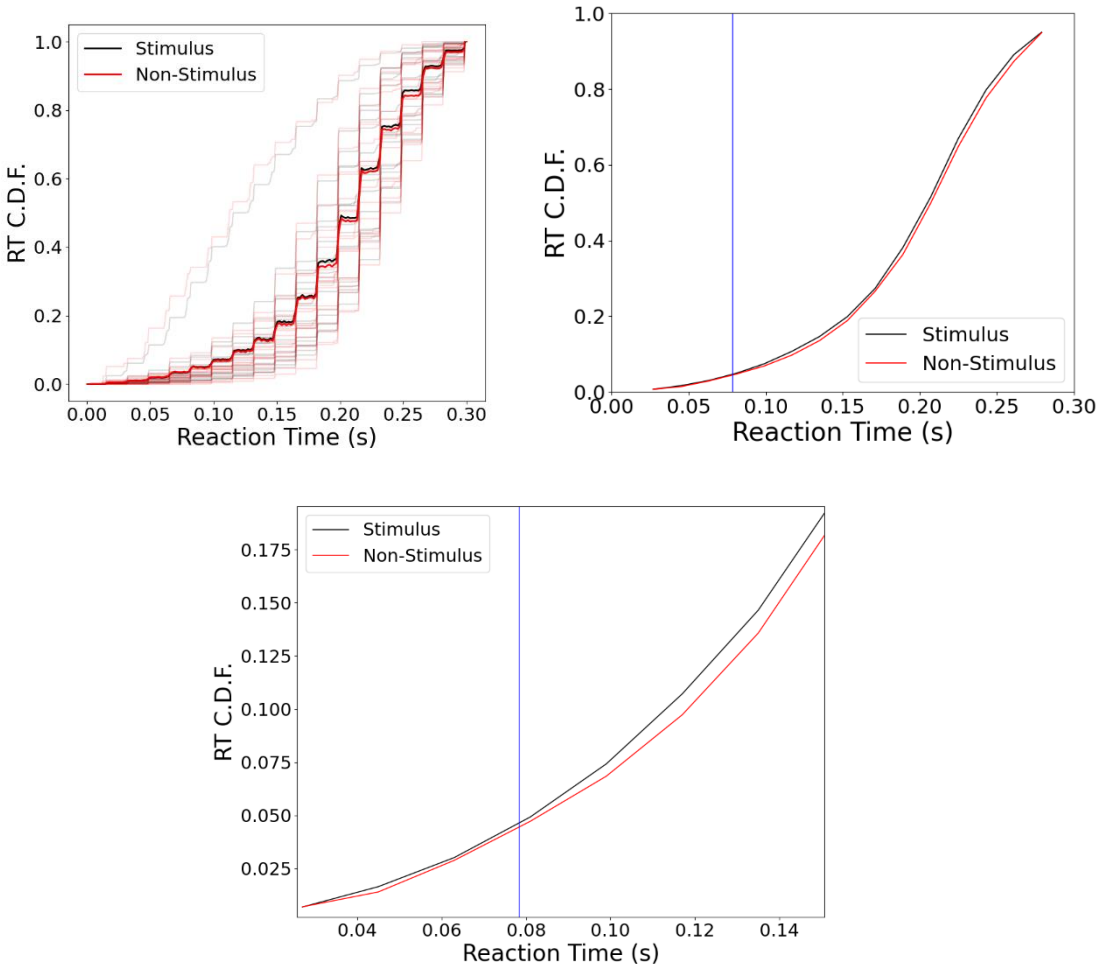


Figure 29. Top-Left: raw RT C.D.F. with the means for all subjects in wider lines. Top-right: mean filtered RT C.D.F. with the modulation onset (78ms) in blue. Bottom: top-right plot zoomed at the first 160ms.

To sum up, the reaction time of human subjects is governed by two independent processes: one that corresponds to the stimulus accumulation and another that is stimulus independent.

6.4 Scientific output

These results were presented in the annual meeting of the Barcelona Computational, Cognitive and Systems Neuroscience Community (BARCCSYN: https://www.crm.cat/barccsyn-2022/) [89] with a poster (See Annex Fig. 2 and 3).

7. Execution Chronogram

In this chapter it will be analyzed the temporal organization of the project and the completed tasks. The study started on the 1st of September, and it lasted until the 30th of June, having a duration of 350 hours.

7.1 Work Breakdown Structure (WBS)

In this section the project will be itemized, providing the Work Breakdown Structure. The WBS can be seen in Fig. 29.

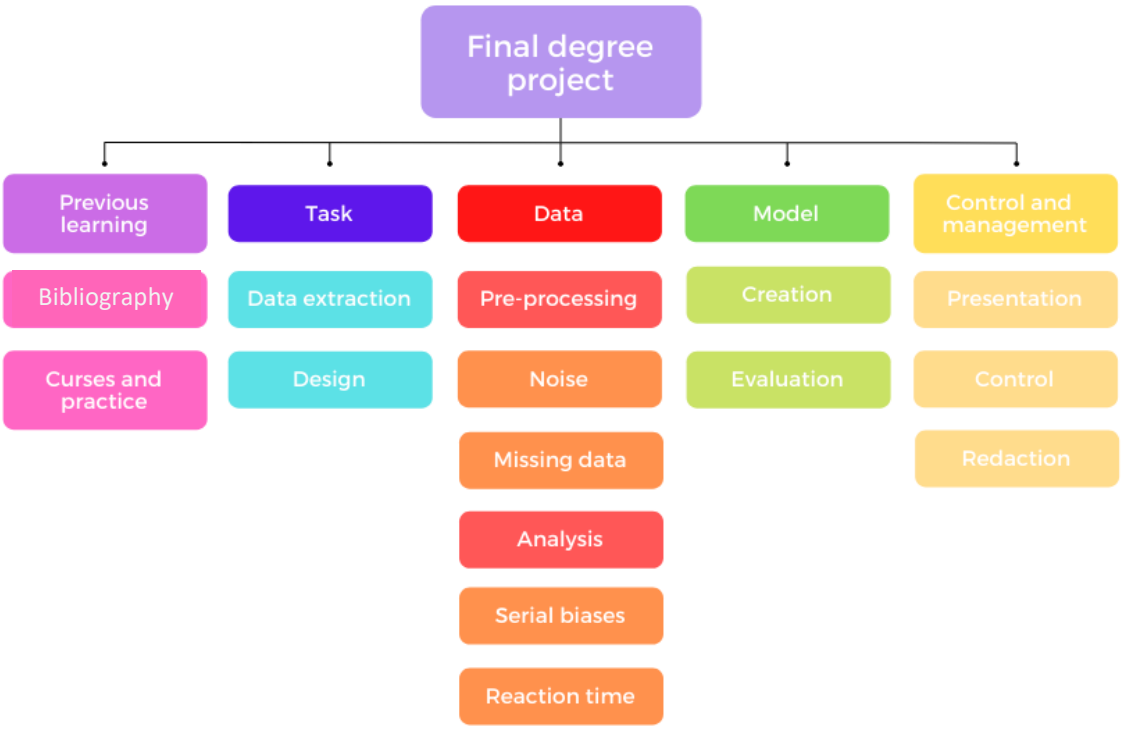


Figure 30. Work Breakdown Structure schematic.

7.2 WBS dictionary

Here we will describe each component of the WBS visualized before.

- Previous learning: since the fields of computational and theoretical neuroscience are very specific, some previous learning is highly recommended so the project can start dynamically and develops with a correct pace. This learning consists in two different strategies in parallel:
    - Bibliography: The Brain Circuits and Behavior laboratory has published many articles, but this project is based mainly in three of them [12], [14], [15], [49]. So, by reading and understanding these, the main concepts of the project can be understood, and the analysis can be applied.
    - Courses and practice: a 4-month duration course was done, named Deep Learning Specialization, and taught by DeepLearning.IA and available at Coursera. In this course, the focus is on acquiring the tools necessary for build and train different architectures of neural networks. Furthermore, data pre-

processing techniques are also taught, useful in any other fields of data analysis. The language used for instruction is Python, and the specialized library is Tensorflow.

The practice consists in the use of the concepts learnt for individual projects, so the concepts of the course are consolidated.

- Task: the data extraction process. 18 subjects did the task in the previous work done by Debora Lombardo [49], but the analysis performed needed more data. Thus, more participants were needed. The subjects came to the laboratory to perform the task. It consists of a two-alternative forced choice (2-AFC) task, in which participants received an auditory stimulus in both ears and had to discern on which side the stimulus was greater.
    - Data extraction: The response was indicated in a tactile tablet, and a lot of data was saved: reaction times, motor times, choice, finger trajectories, and others.
    - Design: since this field of study is very sensitive to different conditions, the optimal task had to be chosen so the data has the minimum noise possible, and all subjects are in a standardized environment. Since our analysis is also part of a project in Hospital Clinic for Alzheimer early detection, the design had to be changed because subjects were older and were less used to tactile technologies.
- Data: pre-processing and posterior analysis of data so substantial information can be extracted.
    - Pre-processing: here the data received will be pre-processed to get rid of the noise and missing values.
        - Noise: here noisy data was dealt with, performing smoothing because of the discrete nature of it. The sampling period was 17ms, so non-binned histograms were discrete and cumulative distribution functions were staircase functions consequently. Also, since the first 100-200 trials of each subject are noisy, because the participant is still learning the task, they were not considered.
        - Missing values: invalid trials are saved as missing values (not a number, *nan*), and it can give problems in the analysis. Since the proportion of valid trials for most of the subjects is bigger than 90%, they can be avoided. However, for transition analysis, these cannot be done because they are part of the trial history, in other words, the subject can be late in responding (invalid trial), but the stimulus can be clear, and an expectation of the posterior trial can be built.
    - Analysis: the data will be analyzed using different techniques. Two main focuses were analyzed. For more details, see 4.1.1 Methodological choices and 5. Detailed engineering.
        - Transition/lateral biases: these biases explain how subjects rely on previous trials, considering blocks (repetition/alternation) or sides (left/right). These are analyzed in two different scenarios: after correct and after error trials. Using this, we can know which strategy uses each subject (reset/reverse).
        - Reaction time: the time that a participant takes to answer a single trial. With this we can know if the subject relies more in the stimulus or in the previous trials history and see if the statistics (patterns) behind the task are understood.

- Model:
    - Model creation: programming. (GLM)
    - Model evaluation: comparison with previous studies and computing statistical significance of the weights.
- Control and management:
    - Prepare a presentation.
    - Redaction of the memory.
    - Control: recurrent meetings with the Director, Dr. Molano. At least 3 meetings per week.

7.3 Precedence analysis

In this section a temporal sequence of the project's main tasks is built, analyzing the precedencies of each one (Tab. 4).

Table 4. Precedence analysis table.

Activity	Precedent	Duration (weeks)
Learning (L)	-	17
Task (T)	L	4
Data pre-processing (DP)	T	1
Data analysis (DA)	DP	8
Model creation (MC)	DA	5
Model evaluation (ME)	MC	4
Redaction (R)	-	35
Presentation preparation (PP)	R, ME	1

7.4 Program Evaluation and Review Technique (PERT)

Here the tasks are ordered according to their timings and graphed (Fig. 30). The critical path is shown in orange. This path shows the minimum amount of time that the project must last (40 weeks).

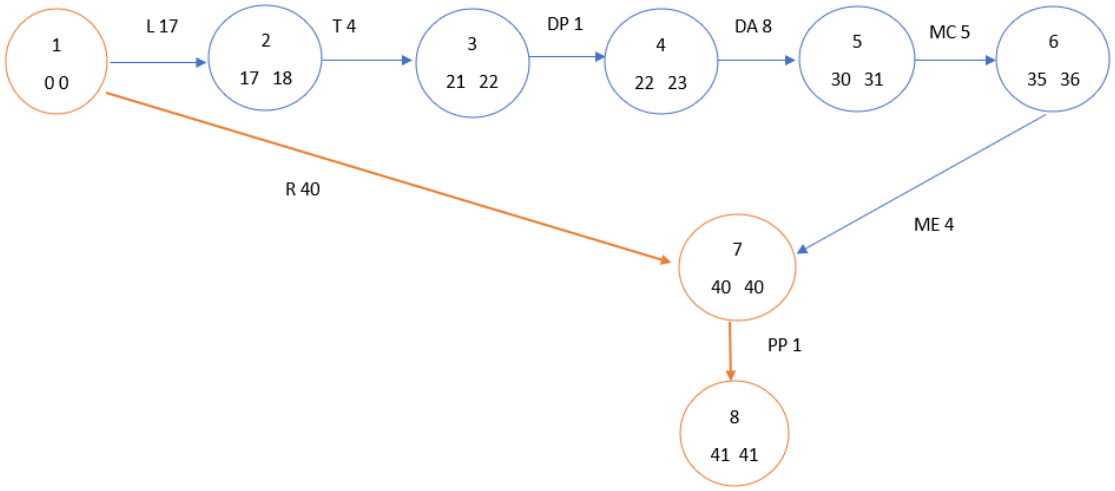


Figure 31. PERT schematic with the critical path in orange.

7.5 Gantt chart

In this part that illustrates the project's schedule (Tab. 5), with the tasks on the vertical axis, and time intervals on the horizontal axis. In the Gantt chart (Fig. 31), the width of the horizontal bars in the graph shows the duration of each activity.

Activity	Start	End
Control and management	**01-09-21**	**29-06-22**
Redaction	01-09-21	22-06-22
Control	01-09-21	22-06-22
Presentation	22-06-22	29-06-22
Previous learning	**31-08-21**	**06-01-22**
Courses and practice	31-08-21	24-12-21
Literature	07-11-21	06-01-22
Task	**06-01-22**	**06-02-22**
Design	06-01-22	13-01-22
Data extraction	14-01-22	06-02-22
Data	**06-02-22**	**10-04-22**
Pre-processing	06-02-22	13-02-22
Noise	06-02-22	10-02-22
Missing data	10-02-22	13-02-22
Analysis	14-02-22	10-04-22
Serial biases	15-02-22	22-03-22
Reaction time	28-02-22	10-04-22
Model	**11-04-22**	**12-06-22**
Creation	11-04-22	15-05-22
Evaluation	15-05-22	12-06-22

Table 5. Project schedule and chronogram.

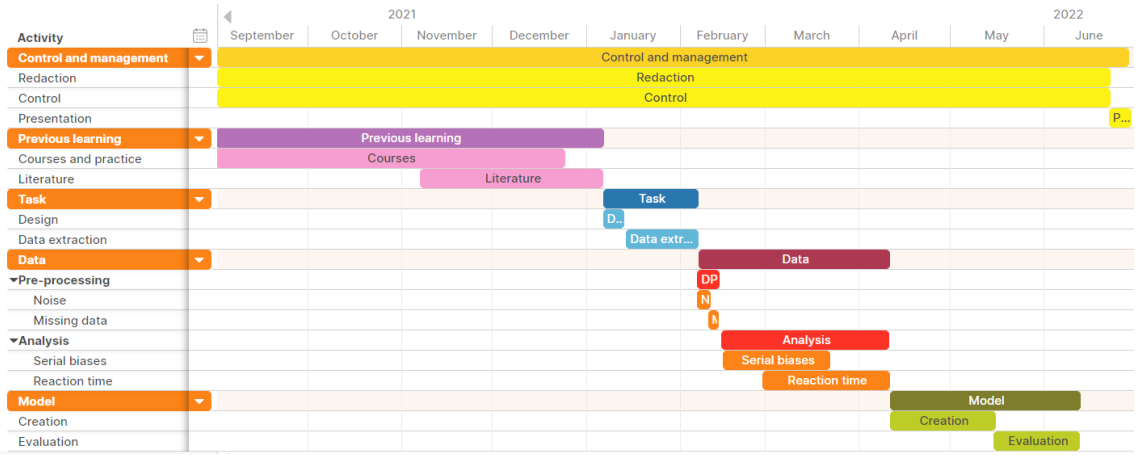


Figure 32. Gantt chart.

8. Technical feasibility

8.1 Technical Aspects

This project has been carried out at the Center Esther Koplowitz (CEK), where IDIBAPS is based. It has been fundamental the use of an experimental room located at the laboratory, where subjects could perform the task. The room is situated on the basement of CEK, it has no windows, and it is provided with a desk and a chair facing the wall, so that participants would not be distracted by the surrounding environment.

8.2 SWOT

Here are presented the internal aspects - such as strengths (S) and weaknesses (W)- and the external ones - such as opportunities (O) and threats (T) (Tab. 6).

	Internal origin	**External origin**
-	WEAKNESSES	THREATS
	- Lack of prior knowledge of computational neuroscience. - Low number of trials per subject. - Limited experience with Spider.	- Small number of resources destined to scientific investigation. - Limitations due to pandemic of SARS-Cov-2. - Field in early stages of development. - Low economic benefit.
+	STRENGTHS	OPORTUNITIES
	- Periodic meetings, recurrent control. - Experience with Python language. - Availability of IDIBAPS equipment and facilities. - Possibility of working remotely. - No environmental impact. - User-friendly, free, and handy application.	- Field in development. - Minimal initial inversion. - Hospital Clinic of Barcelona and IDIBAPS leading biomedical investigation in Spain [90]. - Useful in different market sectors.

Table 6. S.W.O.T. analysis table.

9. Economic feasibility

In this section the economic aspects of the project will be discussed, and the costs shown.

9.1 Economic planification

In Tab. 7 the economic planification over the Gantt diagram can be seen.

Activity	Cost	Acc. Cost	Start	End
Control and management	**7500 €**	**7500€**	**01-09-21**	**29-06-22**
Redaction	4500€	4500 €	01-09-21	22-06-22
Control	3000€	7500 €	01-09-21	22-06-22
Presentation	0 €	7500€	22-06-22	29-06-22
Previous learning	**40€**	**7540€**	**31-08-21**	**06-01-22**
Courses and practice	40€	7540€	31-08-21	24-12-21
Literature	0 €	7540 €	07-11-21	06-01-22
Task	**2394€**	**9934€**	**06-01-22**	**06-02-22**
Design	0 €	7540€	06-01-22	13-01-22
Data extraction	2394€	9934€	14-01-22	06-02-22
Data	**0 €**	**9934€**	**06-02-22**	**10-04-22**
Pre-processing	0 €	9934€	06-02-22	13-02-22
Noise	0 €	9934€	06-02-22	10-02-22
Missing data	0 €	9934€	10-02-22	13-02-22
Analysis	0 €	9934€	14-02-22	10-04-22
Serial biases	0 €	9934 €	15-02-22	22-03-22
Reaction time	0 €	9934€	28-02-22	10-04-22
Model	**0 €**	**9934€**	**11-04-22**	**12-06-22**
Creation	0 €	9934 €	11-04-22	15-05-22
Evaluation	0 €	9934€	15-05-22	12-06-22

Table 7. Economic planification over the chronogram.

9.2 Costs and importance

- Licenses: total cost of 0€
  - Spyder: to work with Python, since this was the chosen option, its compiler becomes an indispensable requisite.
  - GitHub: it is an essential tool to update the code in a common repository, so supervisors and other people can benefit from the work and see the progress.
  - Asana: optional tool that is designed to organize projects, where people can see the progress, the tasks undone, etc.
  - Slack: it is an optional, but very recommended tool to improve communication with co-workers.
  - Inkscape: optional tool used to modify figures and images.
- Staff: total cost of 7500 €
  - Junior engineer: indispensable requisite. Project developer. Salary: 15 €/h. Hours of work: 300h. Total cost: 4500€.
  - Project supervisor: indispensable requisite to guide the engineer in the project development. Salary: 25 €/h. Hours of work: 120h. Total cost: 3000 €.
- Hardware:
  - Computer: indispensable item to read and analyze data. For a fast execution of the code, it is recommended 8GBof RAM memory, no storage limit since files' size is in the order of hundreds of kB. Cost: 1000 €.
  - Tablet: indispensable to execute the task. No limitations regarding storage capacities. Cost: 400 €.

- Headphones: indispensable requisite to do the task. These must have good noise isolating properties for the correct development of the task. Cost: 600 €.

- Training:
  - Courses: optional requisite to have a dynamical start with the project. Cost: 40€.
- Others:
  - Subject payment: subjects received 20€ as basis and 0.01 € for correct response. Considering the hours spent and the rewards obtained, the total cost went up to 394 €.

Therefore, we have a maximum cost of 9934 € and a minimum of 9894 €.

Costs summary:

Items		Importance	Cost
Licenses	Spyder	Indispensable requisite	0 €
	GitHub	Indispensable requisite	0 €
	Asana	Optional requisite	0 €
	Slack	Optional requisite	0 €
	Inkscape	Optional requisite	0 €
Staff	Junior engineer	Indispensable requisite	4500 €
	Project supervisor	Indispensable requisite	3000€
Hardware	Computer	Indispensable requisite	1000€
	Tablet	Indispensable requisite	400€
	Headphones	Indispensable requisite	600€
Training	Courses	Optional requisite	0 – 40€
Others	Subject payment	Indispensable requisite	394 €
Total, maximum			9934 €
Total, minimum			9894 €

Table 8. Costs table, and the importance of each item.

9.3 Funding

The funding of this project comes from both institutions that take part in this project: the University of Barcelona and the IDIBAPS, the institution where this project has been carried out. The Brain Circuits and Behavior Laboratory, from IDIBAPS, has received the fundings by the European Research Council (ERC). Therefore, financial issues derived by the project, such as the payment of the subjects or the purchase of needed equipment have been expensed by the IDIBAPS. The student's work insurance has been covered by the University of Barcelona thanks to the existing academic cooperation agreement between both institutions and the placement program.

This research was supported by the Spanish Ministry of Economy and Competitiveness together with the European Regional Development Fund (IJCI-2016-29358 to D.D.; RTI2018-099750-B-I00 to J.R.), and the European Research Council (ERC-2015-CoG - 683209 Priors to J.R.).

10. Regulation and legal aspects

Since the study has involved human subjects, it has been fundamental to define the regulations and the legal aspects. They have all been communicated to the participants before taking part in the study. An information consent was presented, and the subjects had to sign it to prove their agreement. In addition, they all could refuse to participate and withdraw from the study at any time without any type of penalty. The information consent template can be found in the Annex (13.4 Legal Documents).

The treatment, communication and transfer of personal data of all participants comply with the provisions of Organic Law 15/1999, of December 13, on the protection of personal data and the Royal Decree that develops it (RD 1720/2007). Although the results obtained from the research carried out are published in scientific fields, their identity will never be disclosed.

Personal data become part of the Investigations and Clinical Trials File, which is responsible for the Consortium of the Institut de Investigacions Biomèdiques August Pi i Sunyer and are processed solely and exclusively within the framework of the subjects' participation in this study.

Moreover, data is not saved in hard copies, and is pseudonymized, so during its processing, analysis or publication, only the people involved in the study can identify who have participated in it. The pseudonymized data may be transmitted to third parties and to other countries but in no case will they contain information that can identify the subjects, such as name and surname, initials, address, etc. If this sharing of the anonymous data occurs, it will be for the same purposes of the study described or for use in scientific publications. The promoter undertakes to establish the necessary measures to guarantee the same level of confidentiality as in Spain. Participants have the right of access, rectification, cancellation and opposition to said data by contacting the person in charge of the study.

In addition, different software has been used for the development of this study. However, they all have open-source licenses, which means that they are allowed to be freely used, modified, and shared. In particular, it has been used the programming language Python, which is protected by the General Public License (GNU) [91]. The GNU is a set of commonly used free software licenses that give users the right to run, study, distribute, and change the software.

11. Conclusions

After analyzing twenty human subjects, I have seen that all learnt correctly to do the task, and the learning took between 100 and 300 trials. Moreover, all participants presented transition bias after correct responses. Whereas after error trials, the reverse, which is more optimal, was the most predominant strategy. I hypothesize that all humans can adopt the reverse strategy, but it would have to be proved by making tasks longer and with a larger dataset. On the other hand, humans' reaction times have been seen to be modulated by two independent processes: one that corresponds to the stimulus accumulation and another that is stimulus independent.

In our day-to-day life, previous decisions can affect our current ones, in many different situations. This is because we build expectations from the past experiences, therefore creating a probabilistic model of our surroundings. This is an easier task when these experiences present patterns such as repetitions or alternations, therefore under the influence of serial correlations. The reason why sequential effects dominate our behavior is still a matter of study. However, they showed to be an intrinsic characteristic not only for humans but also for other animals such as rats, manifesting similar behaviors between them. For this reason, we can hypothesize that these behaviors derive from a long evolutionary process in which animals tended to adapt to the surrounding circumstances and their statistics [14].

The project has shown to be economically and technically viable. Moreover, the method used to carry out the experiment has shown to be effective to detect sequential effects, making it the perfect candidate to conduct further studies and for its application in the market.

Regarding the task and software, it could be useful for many other researchers in the neuroscience field. Since it is a non-invasive technique in humans and understandable for subjects, its implementation is easy; and increasing the number of human experiments, the use of other animals would be directly reduced, therefore minimizing the exposure of those to laboratory conditions and providing a good impact to their wellbeing.

To sum up, humans showed to be under the influence of these sequential effects, understood as an efficient solution to the challenges we all face, reflecting the attempt to predict future events in a dynamic environment; and harnessing them in the most optimal way after error trials.

11.1 Further applications

These behaviors could be affected under certain effects, such as drugs or cognitive disorders. Also, they can be trained to give a better performance and reduced reaction time, desirable for some professional sectors. More specifically, the task and analysis could be used in:

- Drug validation: task performed to subjects and see if there are side effects regarding reaction times, decision-making, and sequential effects.
- Help in diagnosis and/or treatment of cognitive disorders
- For professional profiles whose reaction times are important, such as elite athletes, the task could be used to train them in these effects, reducing the response time, trying to reduce their reaction time. It could be used even as a doping test.

However, to implement the task, it should be standardized by taking a much larger dataset, for example by expanding the software to be used worldwide, such as Adrian Owen, who created a web-based platform for the assessment of cognitive function [92], therefore collecting a vast amount of data, which might be noisier than the dataset used in the current study but since there

are many samples, more relevant and significant results could be extracted, capturing general behaviors in a big population. By this means we would be able to standardize the procedure and provide a service to the sectors mentioned above. The same process could be used to detect similarities and differences between different categories of participants, by considering factors such as age, sex, pathologies or different approaches and strategies according to distinct task settings.

Regarding task design, neurofeedback [93], [94] could be implemented so the task can adapt to each subject and thus it is fully exploit: characteristic response time and stimulus strength values for each participant.

11.2 Future work

Regarding research, the future analysis to be performed are:

- Fitting the model developed by Hernández-Navarro et al. [15] regarding proactive and reactive responses. This model is called Parallel Sensory Integration and Action Model (PSIAM).
- Study of humans' finger trajectories in the task (Fig. 32): this study would give information about changes of mind (Fig. 33), which are the trials where participants start answering to one side but mid-trial change abruptly the answer side. This could be because of the transition bias mixed with a strong stimulus evidence: for example, a subject is in a repetition pattern on the right and then there is a strong stimulus on the left, but the subject started going to the right due to the bias and mid-trial changed the trajectory because of a great stimulus strength.

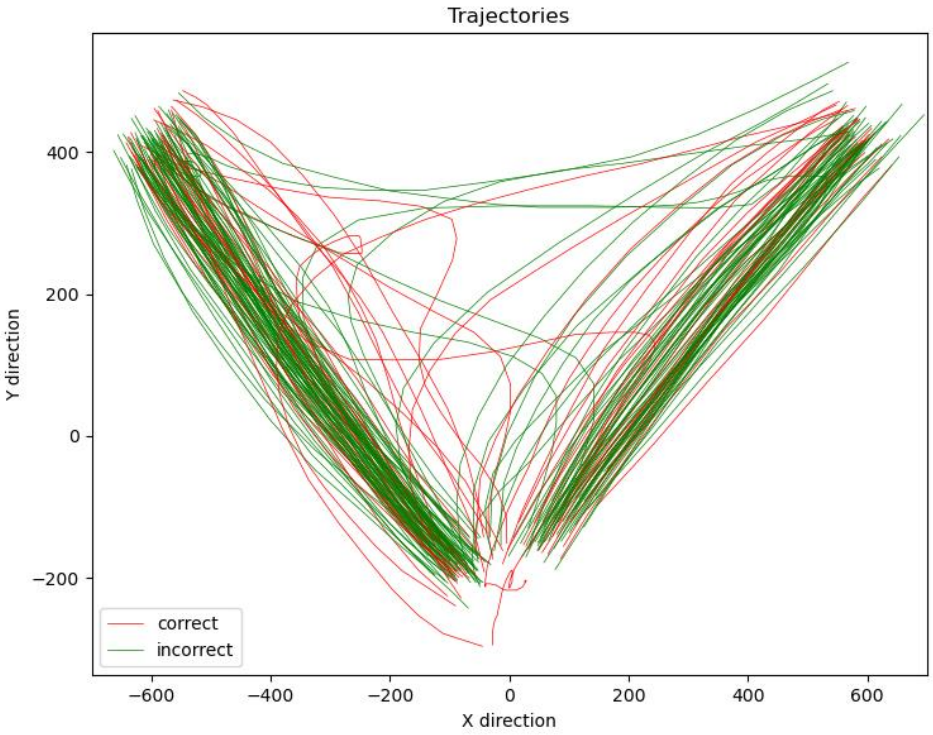


Figure 33. Finger trajectories from 100 trials from subject 10.

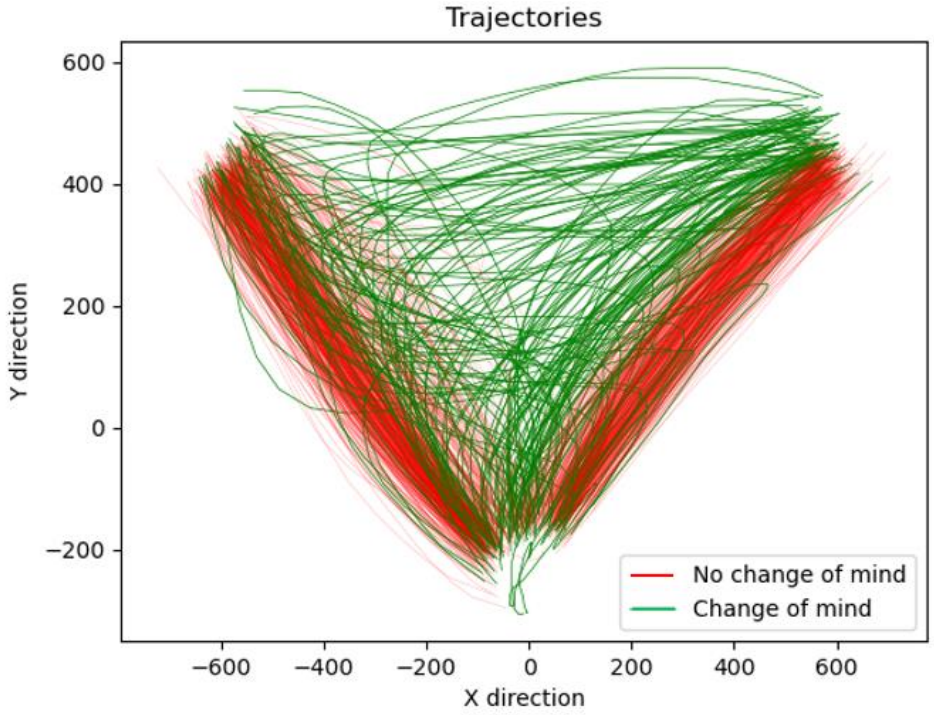


Figure 34. Trajectories with changes of mind (green) of 800 trajectories from subject 13.

- Track and analysis of eye movement: this could give insight of whether the subjects look at the response sides before answering or the finger movement is prior to the eyes.
- Electroencephalogram (EEG) recording and analysis: this could give information about subjects' brain activity when doing the task, which will be a broad new field of analysis.

12. References

[1] J. Monod, *Le hasard et la nécessité: Essai sur la philosophie naturelle de la biologie moderne*. 1981.

[2] S. Herculano-Houzel, "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost," *Proc Natl Acad Sci U S A*, vol. 109, no. SUPPL.1, pp. 10661–10668, Jun. 2012, doi: 10.1073/PNAS.1201895109/ASSET/41169010-7103-422B-8F77-AF073BA4D5D7/ASSETS/GRAPHIC/PNAS.1201895109FIG06.JPEG.

[3] J. Zhang, "Basic Neural Units of the Brain: Neurons, Synapses and Action Potential," May 2019, doi: 10.48550/arxiv.1906.01703.

[4] D. Neill, "Cortical evolution and human behaviour," *Brain Res Bull*, vol. 74, no. 4, pp. 191–205, Sep. 2007, doi: 10.1016/J.BRAINRESBULL.2007.06.008.

[5] W. Enard, "The Molecular Basis of Human Brain Evolution," *Curr Biol*, vol. 26, no. 20, pp. R1109–R1117, Oct. 2016, doi: 10.1016/J.CUB.2016.09.030.

[6] E. L. Maclean, "Unraveling the evolution of uniquely human cognition," *Proc Natl Acad Sci U S A*, vol. 113, no. 23, pp. 6348–6354, Jun. 2016, doi: 10.1073/PNAS.1521270113.

[7] L. R. Santos and A. G. Rosati, "The Evolutionary Roots of Human Decision Making," *Annu Rev Psychol*, vol. 66, p. 321, 2015, doi: 10.1146/ANNUREV-PSYCH-010814-015310.

[8] S. Danziger, J. Levav, and L. Avnaim-Pesso, "Extraneous factors in judicial decisions," *Proc Natl Acad Sci U S A*, vol. 108, no. 17, pp. 6889–6892, Apr. 2011, doi: 10.1073/PNAS.1018033108/-/DCSUPPLEMENTAL.

[9] M. R. Mercier and C. Cappe, "The interplay between multisensory integration and perceptual decision making," *Neuroimage*, vol. 222, Nov. 2020, doi: 10.1016/J.NEUROIMAGE.2020.116970.

[10] T. D. Hanks and C. Summerfield, "Perceptual Decision Making in Rodents, Monkeys, and Humans," *Neuron*, vol. 93, no. 1, pp. 15–31, Jan. 2017, doi: 10.1016/J.NEURON.2016.12.003.

[11] F. Najafi and A. K. Churchland, "Perceptual Decision-making: a field in the midst of a transformation," *Neuron*, vol. 100, no. 2, p. 453, Oct. 2018, doi: 10.1016/J.NEURON.2018.10.017.

[12] A. Hermoso-Mendizabal, A. Hyafil, P. E. Rueda-Orozco, S. Jaramillo, D. Robbe, and J. de la Rocha, "Response outcomes gate the impact of expectations on perceptual decisions," *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–13, Feb. 2020, doi: 10.1038/s41467-020-14824-w.

[13] G. Vernon, D. Farrow, and M. Reid, "Returning serve in tennis: a qualitative examination of the interaction of anticipatory information sources used by professional tennis players," *Front. Psychol.*, vol. 9, no. JUN, p. 895, Jun. 2018, doi: 10.3389/fpsyg.2018.00895.

[14] M. Molano-Mazon, D. Duque, G. R. Yang, J. de La Rocha, and M. Molano-Mazón, "Pre-training RNNs on ecologically relevant tasks explains sub-optimal behavioral reset," *bioRxiv*, p. 2021.05.15.444287, May 2021, doi: 10.1101/2021.05.15.444287.

[15] L. Hernández-Navarro, A. Hermoso-Mendizabal, D. Duque, J. de la Rocha, and A. Hyafil, "Proactive and reactive accumulation-to-bound processes compete during perceptual

decisions," *Nature Communications 2021 12:1*, vol. 12, no. 1, pp. 1–15, Dec. 2021, doi: 10.1038/s41467-021-27302-8.

[16] D. Hume, *A Treatise of Human Nature: Being an Attempt to Introduce the Experimental Method of Reasoning into Moral Subjects*. 1740.

[17] P.-S. de Laplace, *Essai philosophique sur les probabilités*. 1814.

[18] L. Wittgenstein, *Tractatus Logico-Philosophicus*. 1921.

[19] L. Jan, "Neuroscience: Past and Future," *Neuron*, vol. 98, no. 1, pp. 10–11, Apr. 2018, doi: 10.1016/J.NEURON.2018.03.029.

[20] N. Kriegeskorte and P. K. Douglas, "Cognitive computational neuroscience," *Nat Neurosci*, vol. 21, no. 9, p. 1148, Sep. 2018, doi: 10.1038/S41593-018-0210-5.

[21] S. Becker, "Preface to the special issue: Computational cognitive neuroscience," *Brain Research*, vol. 1202, pp. 1–2, Apr. 2008, doi: 10.1016/J.BRAINRES.2007.06.030.

[22] C. K. Hauser and E. Salinas, "Perceptual Decision Making," *Encyclopedia of Computational Neuroscience*, pp. 1–21, 2014, doi: 10.1007/978-1-4614-7320-6_317-1.

[23] W. van den Bos, R. Bruckner, M. R. Nassar, R. Mata, and B. Eppinger, "Computational neuroscience across the lifespan: Promises and pitfalls," *Developmental Cognitive Neuroscience*, vol. 33, p. 42, Oct. 2018, doi: 10.1016/J.DCN.2017.09.008.

[24] M. Dricu and S. Frühholz, "A neurocognitive model of perceptual decision-making on emotional signals," *Human Brain Mapping*, vol. 41, no. 6, p. 1532, Apr. 2020, doi: 10.1002/HBM.24893.

[25] M. Dricu, L. Ceravolo, D. Grandjean, and S. Frühholz, "Biased and unbiased perceptual decision-making on vocal emotions," *Scientific Reports*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/S41598-017-16594-W.

[26] R. Ratcliff, C. Voskuilen, and A. Teodorescu, "Modeling 2-Alternative Forced-Choice Tasks: Accounting for both Magnitude and Difference Effects," *Cogn Psychol*, vol. 103, p. 1, Jun. 2018, doi: 10.1016/J.COGPSYCH.2018.02.002.

[27] M. Jogan and A. A. Stocker, "A new two-alternative forced choice method for the unbiased characterization of perceptual bias and discriminability," *Journal of Vision*, vol. 14, no. 3, pp. 20–20, Mar. 2014, doi: 10.1167/14.3.20.

[28] J. C. Lee, T. Beesley, and E. J. Livesey, "Sequential effects and sequence learning in a three-choice serial reaction time task," *Acta Psychol (Amst)*, vol. 170, pp. 168–176, Oct. 2016, doi: 10.1016/J.ACTPSY.2016.08.004.

[29] D. Bindman and C. Chubb, "Brightness assimilation in bullseye displays," *Vision Res*, vol. 44, no. 3, pp. 309–319, 2004, doi: 10.1016/S0042-6989(03)00430-9.

[30] Á. Lukács, K. S. Lukics, and D. Dobó, "Online Statistical Learning in Developmental Language Disorder," *Frontiers in Human Neuroscience*, vol. 15, Sep. 2021, doi: 10.3389/FNHUM.2021.715818.

[31] D. A. Ganea, A. Bexter, M. Günther, P. M. Gardères, B. M. Kampa, and F. Haiss, "Pupillary Dilations of Mice Performing a Vibrotactile Discrimination Task Reflect Task Engagement and Response Confidence," *Front Behav Neurosci*, vol. 14, Sep. 2020, doi: 10.3389/FNBEH.2020.00159.

[32] E. C. Ketel, R. A. de Wijk, C. de Graaf, and M. Stieger, “Effect of cross-cultural differences on thickness, firmness and sweetness sensitivity,” *Food Res Int*, vol. 152, Feb. 2022, doi: 10.1016/J.FOODRES.2020.109890.

[33] A. Abrahamyan, L. L. Silva, S. C. Dakin, M. Carandini, and J. L. Gardner, “Adaptable history biases in human perceptual decisions,” *Proc Natl Acad Sci U S A*, vol. 113, no. 25, p. E3548, Jun. 2016, doi: 10.1073/PNAS.1518786113.

[34] R. Y. Cho *et al.*, “Mechanisms underlying dependencies of performance on stimulus history in a two-alternative forced-choice task,” *Cogn Affect Behav Neurosci*, vol. 2, no. 4, pp. 283–299, 2002, doi: 10.3758/CABN.2.4.283.

[35] C. M. Conway and M. H. Christiansen, “Sequential learning in non-human primates,” *Trends in Cognitive Sciences*, vol. 5, no. 12, pp. 539–546, Dec. 2001, doi: 10.1016/S1364-6613(00)01800-3.

[36] F. H. Petzschner, S. Glasauer, and K. E. Stephan, “A Bayesian perspective on magnitude estimation,” *Trends in Cognitive Sciences*, vol. 19, no. 5, pp. 285–293, May 2015, doi: 10.1016/J.TICS.2015.03.002.

[37] F. O’Brien, “Sequential contrast effects with human subjects,” *Journal of the Experimental Analysis of Behavior*, vol. 11, no. 5, p. 537, Sep. 1968, doi: 10.1901/JEAB.1968.11-537.

[38] “Sequential effects reflect parallel learning of multiple environmental regularities.” https://papers.nips.cc/paper/2009/hash/522a9ae9a99880d39e5daec35375e999-Abstract.html (accessed May 17, 2022).

[39] Z. Wang, B. Xu, and H. J. Zhou, “Social cycling and conditional responses in the Rock-Paper-Scissors game,” *Scientific Reports 2014 4:1*, vol. 4, no. 1, pp. 1–7, Jul. 2014, doi: 10.1038/srep05830.

[40] A. Abrahamyan, L. L. Silva, S. C. Dakin, M. Carandini, and J. L. Gardner, “Adaptable history biases in human perceptual decisions,” *Proc Natl Acad Sci U S A*, vol. 113, no. 25, p. E3548, Jun. 2016, doi: 10.1073/PNAS.1518786113.

[41] W. S. Verplanck, J. W. Cotton, and G. H. Collier, “Previous training as a determinant of response dependency at the threshold,” *Journal of Experimental Psychology*, vol. 46, no. 1, pp. 10–14, Jul. 1953, doi: 10.1037/H0050386.

[42] M. H. Wilder, M. Jones, and M. C. Mozer, “Sequential effects reflect parallel learning of multiple environmental regularities.”

[43] M. Wilder, “Common Principles Underlying Models of Sequential Effects.”

[44] D. Gökaydin, D. J. Navarro, A. Ma-Wyatt, and A. Perfors, “The Structure of Sequential Effects,” *Journal of Experimental Psychology: General*, vol. 145, no. 1, pp. 110–123, Jan. 2016, doi: 10.1037/XGE0000106.

[45] H. Stein *et al.*, “Reduced serial dependence suggests deficits in synaptic potentiation in anti-NMDAR encephalitis and schizophrenia,” *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–11, Aug. 2020, doi: 10.1038/s41467-020-18033-3.

[46] I. Lieder, V. Adam, O. Frenkel, S. Jaffe-Dax, M. Sahani, and M. Ahissar, “Perceptual bias reveals slow-updating in autism and fast-forgetting in dyslexia,” *Nature Neuroscience 2019 22:2*, vol. 22, no. 2, pp. 256–264, Jan. 2019, doi: 10.1038/s41593-018-0308-9.

[47] D. Sussillo, “Neural circuits as computational dynamical systems,” *Curr Opin Neurobiol*, vol. 25, pp. 156–163, Apr. 2014, doi: 10.1016/J.CONB.2014.01.008.

[48] G. R. Yang and X. J. Wang, “Artificial Neural Networks for Neuroscientists: A Primer,” *Neuron*, vol. 107, no. 6, pp. 1048–1070, Sep. 2020, doi: 10.1016/J.NEURON.2020.09.005.

[49] Debora Lombardo, “ Investigating sequential effects in humans performing a 2AFC psychophysical task,” Universitat de Barcelona, 2022. Accessed: Feb. 15, 2022. [Online]. Available: Investigating sequential effects in humans performing a 2AFC psychophysical task

[50] A. Javor, M. Koller, N. Lee, L. Chamberlain, and G. Ransmayr, “Neuromarketing and consumer neuroscience: contributions to neurology,” *BMC Neurology*, vol. 13, p. 13, Feb. 2013, doi: 10.1186/1471-2377-13-13.

[51] L. Alvino, L. Pavone, A. Abhishta, and H. Robben, “Picking Your Brains: Where and How Neuroscience Tools Can Enhance Marketing Research,” *Frontiers in Neuroscience*, vol. 14, p. 577666, Dec. 2020, doi: 10.3389/FNINS.2020.577666.

[52] “SJR - International Science Ranking.” https://www.scimagojr.com/countryrank.php?category=2802&order=it&ord=desc (accessed May 23, 2022).

[53] “SJR Compare Countries.” https://www.scimagojr.com/comparecountries.php?ids[]=us&ids[]=cn&ids[]=gb&ids[]=de&ids[]=ca&ids[]=it&area=2800 (accessed May 23, 2022).

[54] “BARCCSYN – Barcelona Computational, Cognitive and Systems Neuroscience Community.” https://barccsyn.org/ (accessed May 30, 2022).

[55] “Google.” https://www.google.es/ (accessed May 20, 2022).

[56] “DeepMind.” https://www.deepmind.com/ (accessed May 20, 2022).

[57] “Research.” https://www.deepmind.com/research?tag=Neuroscience (accessed May 20, 2022).

[58] P. M. Pilarski, A. Butcher, M. Johanson, M. M. Botvinick, A. Bolt, and A. S. R. Parker, “Learned human-agent decision-making, communication and joint action in a virtual reality environment,” May 2019, doi: 10.48550/arxiv.1905.02691.

[59] S. Sarkar *et al.*, “Effects of Diazepam on Reaction Times to Stop and Go,” *Frontiers in Human Neuroscience*, vol. 14, p. 403, Oct. 2020, doi: 10.3389/FNHUM.2020.567177/BIBTEX.

[60] S. Baykara and K. Alban, “Visual and Auditory Reaction Times of Patients with Opioid Use Disorder,” *Psychiatry Investigation*, vol. 16, no. 8, p. 602, Aug. 2019, doi: 10.30773/PI.2019.05.16.

[61] O. H. Hernández, M. Vogel-Sprott, and V. I. Ke-Aznar, “Alcohol impairs the cognitive component of reaction time to an omitted stimulus: a replication and an extension,” *J Stud Alcohol Drugs*, vol. 68, no. 2, pp. 276–281, 2007, doi: 10.15288/JSAD.2007.68.276.

[62] C. Bolfer *et al.*, “Reaction time assessment in children with ADHD,” *Arq Neuropsiquiatr*, vol. 68, no. 2, pp. 282–286, 2010, doi: 10.1590/S0004-282X2010000200025.

[63] B. U. Christ, M. I. Combrinck, and K. G. F. Thomas, “Both reaction time and accuracy measures of intraindividual variability predict cognitive performance in Alzheimer’s

disease," *Frontiers in Human Neuroscience*, vol. 12, p. 124, Apr. 2018, doi: 10.3389/FNHUM.2018.00124/BIBTEX.

[64] "Force and Torque Testing Instruments | S. I. Instruments." https://www.si-instruments.com.au/ (accessed May 30, 2022).

[65] "Reaction Time Panel | SI Instruments." https://www.si-instruments.com.au/supplier/lafayette/reaction-time-panel.html (accessed May 30, 2022).

[66] "BATAKPro." https://www.batak.com/batakpro.htm (accessed May 30, 2022).

[67] R. Brand, W. Wolff, and D. Thieme, "Using response-time latencies to measure athletes' doping attitudes: The brief implicit attitude test identifies substance abuse in bodybuilders," *Substance Abuse: Treatment, Prevention, and Policy*, vol. 9, no. 1, pp. 1–10, Oct. 2014, doi: 10.1186/1747-597X-9-36/FIGURES/3.

[68] "Substance Abuse Treatment, Prevention, and Policy | Home page." https://substanceabusepolicy.biomedcentral.com/ (accessed May 30, 2022).

[69] "MUSEUM OF THE HISTORY OF REACTION TIME RESEARCH." http://tomperera.com/psychology_museum/mrt.htm (accessed Jun. 01, 2022).

[70] T. A. Salthouse, "Reaction Time," *Encyclopedia of Gerontology*, pp. 407–410, 2007, doi: 10.1016/B0-12-370870-2/00158-X.

[71] D. Burr and P. Thompson, "Motion psychophysics: 1985-2010," *Vision Research*, vol. 51, no. 13, pp. 1431–1456, Jul. 2011, doi: 10.1016/J.VISRES.2011.02.008.

[72] "psychophysics - Search Results - PubMed." https://pubmed.ncbi.nlm.nih.gov/?term=psychophysics (accessed Jun. 01, 2022).

[73] "MathWorks - MATLAB & Simulink Creators." https://es.mathworks.com/ (accessed May 15, 2022).

[74] "A Python Book: Beginning Python, Advanced Python, and Python Exercises." https://web.archive.org/web/20120623165941/http://cutter.rexx.com/~dkuhlman/python_book_01.html (accessed May 15, 2022).

[75] "About Python™ | Python.org." https://www.python.org/about/ (accessed May 15, 2022).

[76] "R: The R Project for Statistical Computing." https://www.r-project.org/ (accessed May 15, 2022).

[77] "Home — Spyder IDE." https://www.spyder-ide.org/ (accessed May 16, 2022).

[78] "Project Jupyter | Home." https://jupyter.org/ (accessed May 16, 2022).

[79] Y. Yi and D. M. Merfeld, "Decision Making: Neural Mechanisms and Innovative Methodology: A quantitative confidence signal detection model: 1. Fitting psychometric functions," *Journal of Neurophysiology*, vol. 115, no. 4, p. 1932, Apr. 2016, doi: 10.1152/JN.00318.2015.

[80] F. A. Wichmann and N. J. Hill, "The psychometric function: I. Fitting, sampling, and goodness of fit," *Perception & Psychophysics 2001 63:8*, vol. 63, no. 8, pp. 1293–1313, 2001, doi: 10.3758/BF03194544.

[81] H. Dai and C. Micheyl, "Psychometric functions for pure-tone frequency discrimination," *J Acoust Soc Am*, vol. 130, no. 1, pp. 263–272, Jul. 2011, doi: 10.1121/1.3598448.

[82] A. Abrahamyan, L. L. Silva, S. C. Dakin, M. Carandini, and J. L. Gardner, “Adaptable history biases in human perceptual decisions,” *Proc Natl Acad Sci U S A*, vol. 113, no. 25, pp. E3548–E3557, Jun. 2016, doi: 10.1073/PNAS.1518786113.

[83] L. Busse *et al.*, “The Detection of Visual Contrast in the Behaving Mouse,” *The Journal of Neuroscience*, vol. 31, no. 31, p. 11351, Aug. 2011, doi: 10.1523/JNEUROSCI.6689-10.2011.

[84] I. Fründ, F. A. Wichmann, and J. H. Macke, “Quantifying the effect of intertrial dependence on perceptual decisions,” *J Vis*, vol. 14, no. 7, pp. 1–16, 2014, doi: 10.1167/14.7.9.

[85] “Introduction — statsmodels.” https://www.statsmodels.org/stable/index.html (accessed Jun. 06, 2022).

[86] M. T. G. Pain and A. Hibbs, “Sprint starts and the minimum auditory reaction time,” *J Sports Sci*, vol. 25, no. 1, pp. 79–86, Jan. 2007, doi: 10.1080/02640410600718004.

[87] J. Valls-Solé, A. Solé, F. Valldeoriola, E. Muñoz, L. E. Gonzalez, and E. S. Tolosa, “Reaction time and acoustic startle in normal human subjects,” *Neurosci Lett*, vol. 195, no. 2, pp. 97–100, Aug. 1995, doi: 10.1016/0304-3940(94)11790-P.

[88] F. Antoneli, F. M. Passos, L. R. Lopes, and M. R. S. Briones, “A Kolmogorov-Smirnov test for the molecular clock based on Bayesian ensembles of phylogenies,” *PLoS One*, vol. 13, no. 1, Jan. 2018, doi: 10.1371/JOURNAL.PONE.0190826.

[89] “BARCCSYN 2022 - Centre de Recerca Matemàtica.” https://www.crm.cat/barccsyn-2022/ (accessed Jun. 05, 2022).

[90] “El IDIBAPS y sus entidades consorciadas son líderes en producción científica según el estudio SCImago Institutions Rankings | Hospital Clínic Barcelona.” https://www.clinicbarcelona.org/noticias/el-idibaps-y-sus-entidades-consorciadas-son-lideres-en-produccion-cientifica-segun-el-estudio-scimago-institutions-rankings (accessed May 23, 2022).

[91] “The GNU Operating System and the Free Software Movement.” https://www.gnu.org/home.en.html (accessed May 20, 2022).

[92] “Online Cognitive Assessment Platform | Cambridge Brain Sciences.” https://www.cambridgebrainsciences.com/ (accessed Jun. 01, 2022).

[93] C. Loriette, C. Ziane, and S. ben Hamed, “Neurofeedback for cognitive enhancement and intervention and brain plasticity,” *Rev Neurol (Paris)*, vol. 177, no. 9, pp. 1133–1144, Nov. 2021, doi: 10.1016/J.NEUROL.2021.08.004.

[94] R. Markiewicz, “The use of EEG Biofeedback/Neurofeedback in psychiatric rehabilitation,” *Psychiatr Pol*, vol. 51, no. 6, pp. 1095–1106, 2017, doi: 10.12740/PP/68919.

13. Annex

List of figures

13.1 BARCCSYN 2022

13.1.1 Certificate of attendance

CRM
CENTRE DE RECERCA MATEMÀTICA

CERTIFICATE OF ATTENDANCE

We certify that

Alexandre Garcia-Duran

has participated in the **BARCCSYN 2022**, organised by the Centre de Recerca Matemàtica from May 26 to 27, 2022.

Professor Lluís Alsedà
Director of the Centre de Recerca Matemàtica

May 27, 2022
Bellaterra

Annex Figure 1. Certificate of attendance at the BARCCSYN 2022 annual meeting.

13.1.2 Abstract

Human idiosyncratic biases in an expectation-based 2AFC auditory task under strong time pressure

Alex Garcia-Duran[1], Debora Lombardo[1], Jaime de la Rocha[1], Manuel Molano-Mazón[1,2]

[1] IDIBAPS, Rosselló 149, Barcelona, 08036, Spain
[2] Laboratoire de Neurosciences Cognitives, INSERM U960, École Normale

Recent studies have thoroughly characterized the behavior of rats performing a Two-Alternative Forced Choice (2AFC) auditory task, in which the probability to repeat the previous stimulus category is varied in a blockwise fashion. These studies showed that rats exhibit a transition bias: a tendency to alternate/repeat the previous response using an estimate of the probability given the recent trial history. However after error trials, the transition bias was null (Hermoso-Mendizabal et al. 2020). Even though it is suboptimal, this so-called reset strategy has been shown to be highly robust and present in many task variants (Molano-Mazón et al. 2021). On the other hand, the reaction times of rats performing the 2AFC task has been shown to be governed by two independent processes: one that depends on the accumulation of the stimulus evidence and a second, stimulus-independent process that only depends on the time elapsed since the beginning of the trial (Hernández-Navarro et al. 2021). Here we have investigated the behavior of human subjects performing an auditory 2AFC task presenting the same type of correlations experienced by the rats. We found that their strategies were more heterogeneous, with some subjects displaying a clear reset strategy while others developed a more optimal strategy. Furthermore, the reaction times of the human subjects showed evidence of being influenced by the two processes mentioned above, suggesting that the existence of two the different mechanisms described in rats may be a general feature present across species.

References

Hermoso-Mendizabal, A., Hyafil, A., Rueda-Orozco, P. E., Jaramillo, S., Robbe, D., & De la Rocha, J. (2020). Response outcomes gate the impact of expectations on perceptual decisions. Nature communications, 11(1), 1-13.

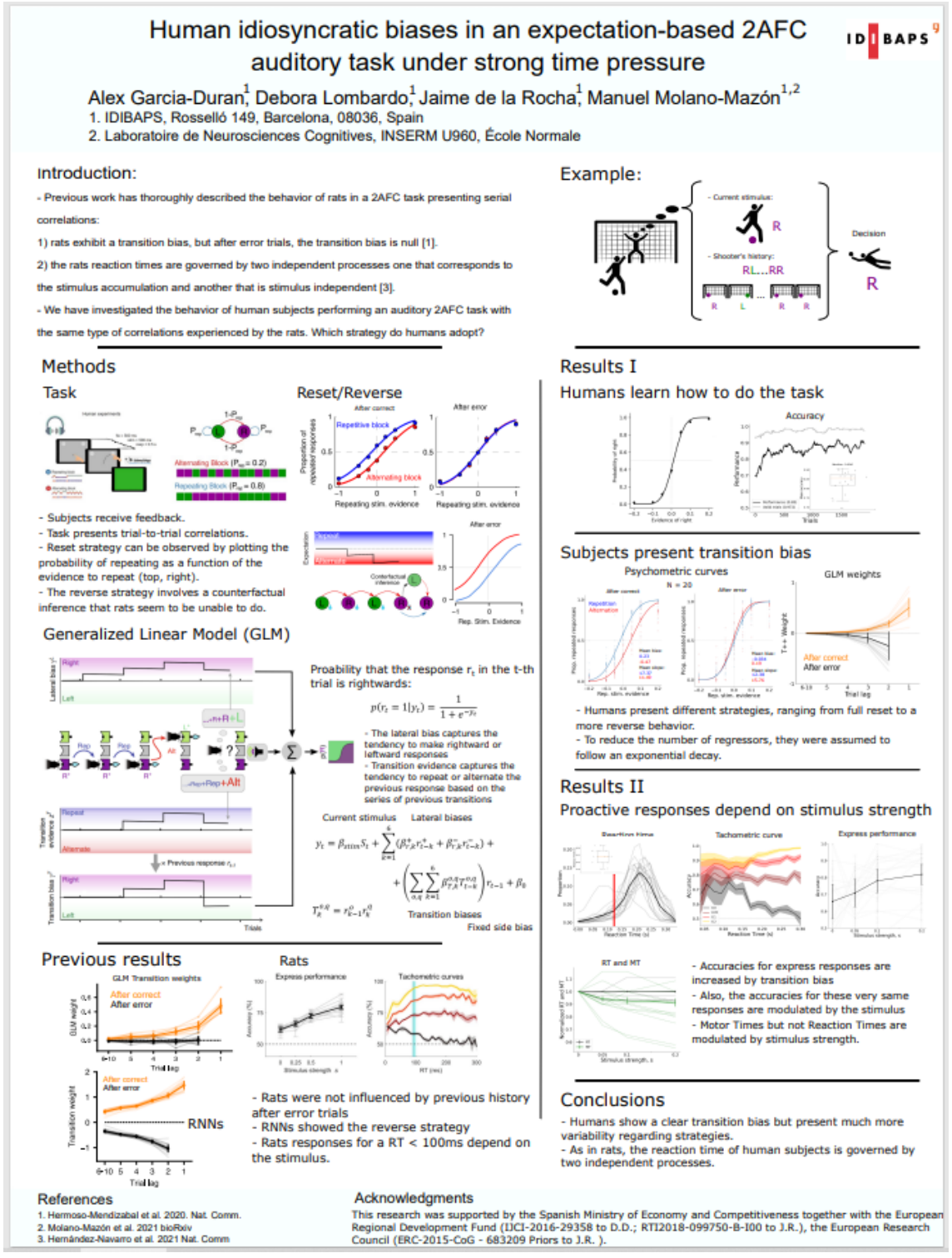
Molano-Mazon, M., Shao Y., Duque, D., Yang, G. R., Ostojic S. & de la Rocha, J. (2021). Ecologically pre-trained RNNs explain suboptimal animal decisions. *bioRxiv*.

Hernández-Navarro, L., Hermoso-Mendizabal, A., Duque, D., de la Rocha, J., & Hyafil, A. (2021). Proactive and reactive accumulation-to-bound processes compete during perceptual decisions. Nature communications, 12(1), 1-15.

Acknowledgements

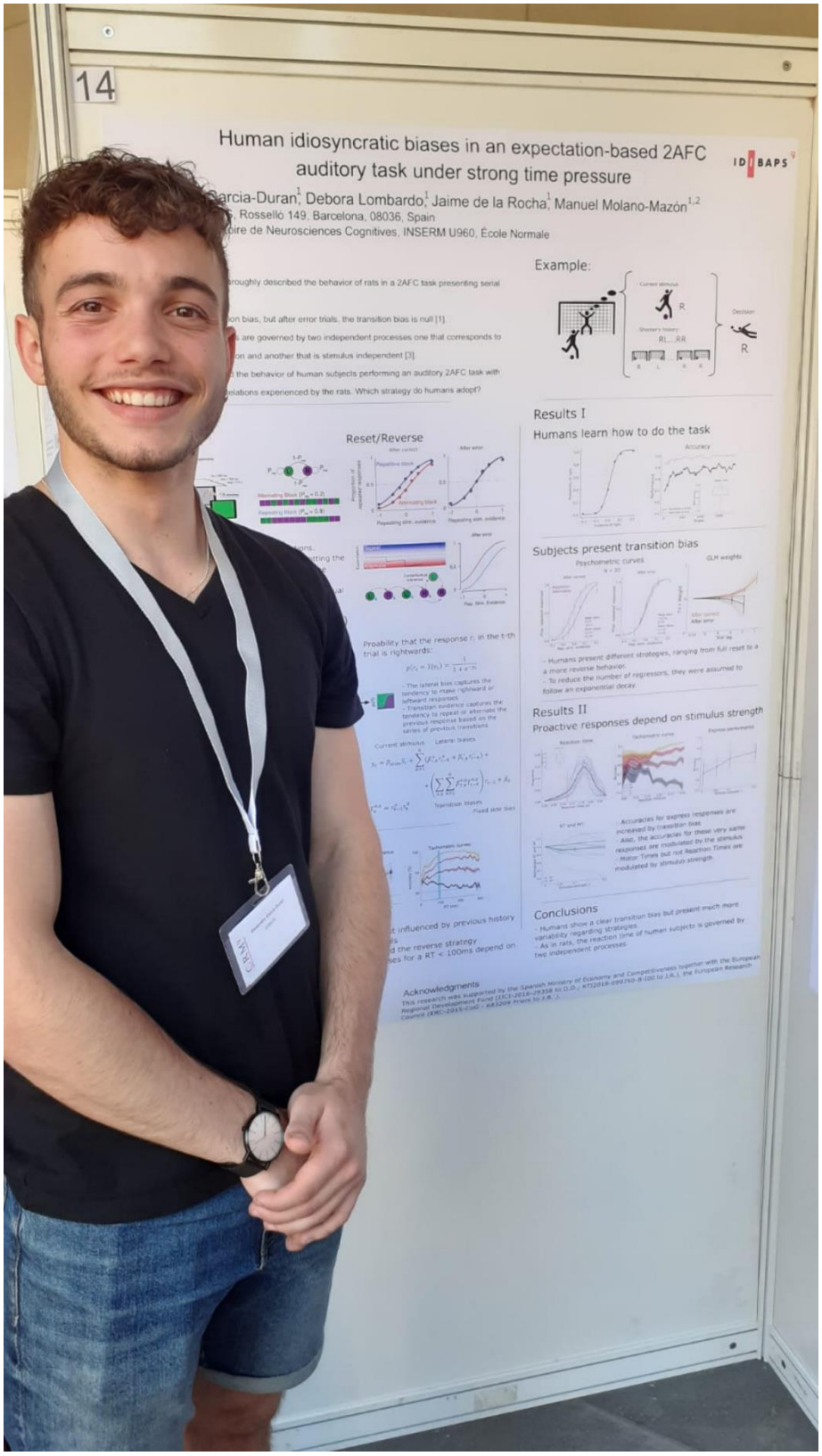
This research was supported by the Spanish Ministry of Economy and Competitiveness together with the European Regional Development Fund (IJCI-2016-29358 to D.D.; RTI2018-099750-B-I00 to J.R.), the European Research Council (ERC-2015-CoG - 683209 Priors to J.R.).

13.1.3 Poster



Annex Figure 2. Poster presented in the BARCCSYN 2022 annual meeting.

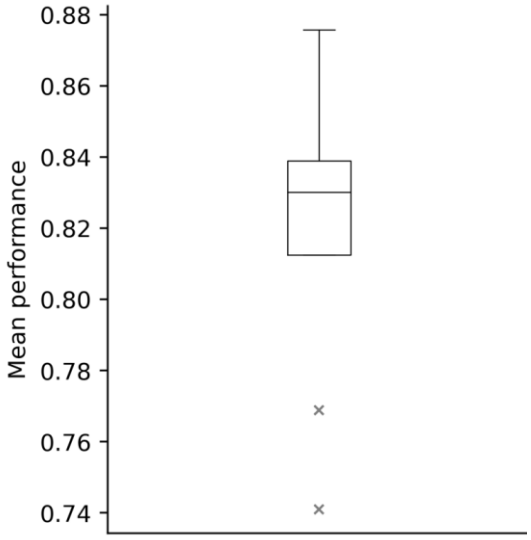
13.1.4 Image



Annex Figure 3. Poster presentation in BARCCSYN 2022 annual meeting.

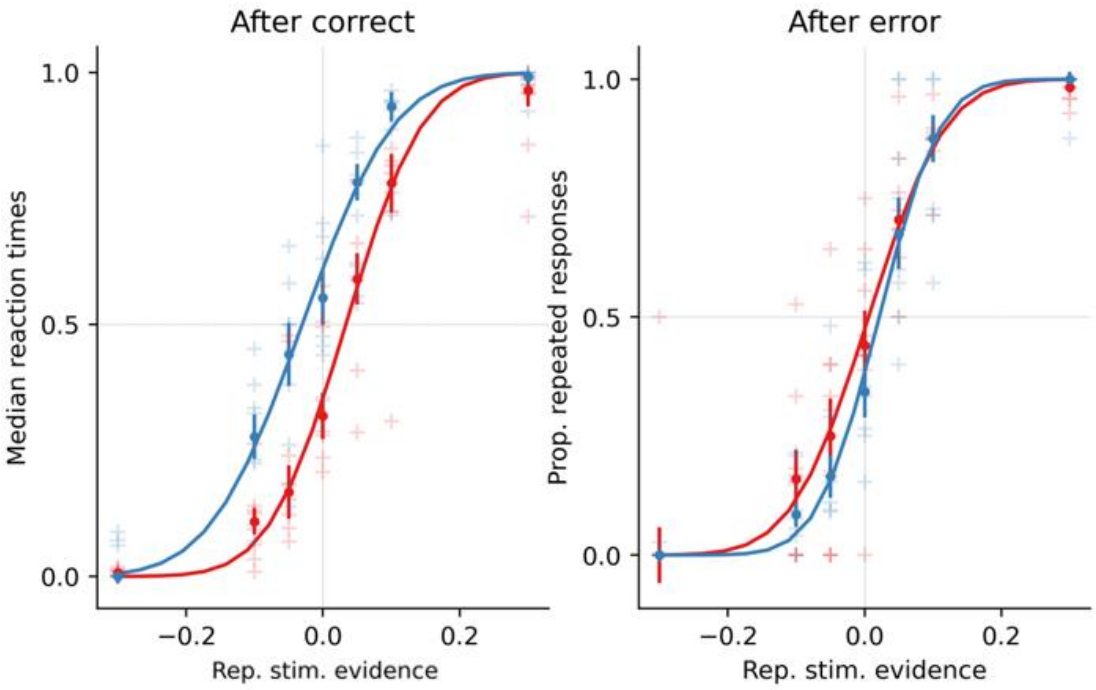
13.2 Figures from Version 1 (500ms)

Mean performance of all the group:



Annex Figure 4. Mean performance of Version 1 subjects

Group psychometric curves in the rep/alt space:



Annex Figure 5. Psychometric curves of Version 1 subjects.

13.3 Code

The code is available at the following GitHub repository:
https://github.com/manuelmolano/psycho_priors

13.3.1 Main: analyses.py

```
import pandas as pd
import helper_functions as hf
from scipy.optimize import curve_fit
import numpy as np
import matplotlib.pyplot as plt
import glob
import os
from scipy.signal import find_peaks
import seaborn as sns

# GLOBAL VARIABLES
FIX_TIME = 0.5
NUM_BINS_RT = 6
NUM_BINS_MT = 7
MT_MIN = 0.05
MT_MAX = 1
START_ANALYSIS = hf.START_ANALYSIS  # trials to ignore
TAU_LIST = hf.tau_list


def box_plot(data, ax, x, lw=.5, fliersize=4, color='k', widths=0.15):
    bp = ax.boxplot(data, positions=[x], widths=widths)
    for p in ['whiskers', 'caps', 'boxes', 'medians']:
        for bpp in bp[p]:
            bpp.set(color=color, linewidth=lw)
    bp['fliers'][0].set(markeredgecolor=color, markerfacecolor=color, alpha=0.5,
                        marker='x', markersize=fliersize)
    ax.set_xticks([])


def get_data(subj, main_folder):
    # subject folder
    # folder = main_folder+subj+'/'   # Manuel
    folder = main_folder+'\\'+subj+'\\'  # Alex
    # find all data files
    files = glob.glob(folder+'*trials.csv')
    # take files names
    file_list = [os.path.basename(x) for x in files
                 if x.endswith('trials.csv')]
    # sort files
    sfx = [x[x.find('2021'):x.find('2021')+15] for x in file_list]
```

```
    sorted_list = [x for _, x in sorted(zip(sfx, file_list))]
    print(sorted_list)
    # create data
    data = {'correct': np.empty((0,)), 'answer_response': np.empty((0,)),
            'soundPlay_object1_leftRightBalance': np.empty((0,)),
            'respondedInTime': np.empty((0,)), 'block': np.empty((0,)),
            'soundPlay_responseTime': np.empty((0,)),
            'soundPlay_duration': np.empty((0,)),
            'answer_responseTime': np.empty((0,))}
    # go over all files
    for f in sorted_list:
        # read file
        # df1 = pd.read_csv(main_folder+subj+'/'+f, sep=',')  # Manuel
        df1 = pd.read_csv(main_folder+'\\'+subj+'\\'+f, sep=',')  # Alex
        for k in data.keys():
            values = df1[k].values
            if k == 'soundPlay_object1_leftRightBalance':
                values = values-.5
                values[np.abs(values) < 0.01] = 0
            data[k] = np.concatenate((data[k], values))
    return data


def plot_learning(perf, valid, ax, w_conv=200):
    m_p = np.mean(perf)
    learning_curve = np.convolve(perf, np.ones((w_conv,))/w_conv, mode='valid')
    ax.plot(learning_curve,
            'k', label='Performance ('+str(np.round(m_p, 3))+')')
    m_v = np.mean(valid)
    valid_curve = np.convolve(valid, np.ones((w_conv,))/w_conv, mode='valid')
    ax.plot(valid_curve,
            color=(.7, .7, .7),
            label='Valid trials ('+str(np.round(m_v, 3))+')')
    tune_panel(ax=ax, xlabel='Trials', ylabel='Performance')
    ax.axhline(y=0.5, linestyle='--', lw=0.2, color=(.5, .5, .5))
    ax.legend()
    return learning_curve, valid_curve


def plot_ws_lsw_psycho_curve(choice_12, ev, prev_perf, lbs, ws_lsw_panel):
    colors = [(.5, .5, .5), (0, 0, 0)]
    mean_rep_prop = []  # XXX: this is actually the bias and the std is not used
    for ip, p in enumerate([0, 1]):
        # WS/LS
        popt, pcov, ev_mask, repeat_mask =\
            hf.bias_psychometric(choice=choice_12.copy(), ev=-ev.copy(),
                                 mask=(prev_perf == p), maxfev=100000)
```

```
        d_ws_ls = hf.plot_psycho_curve(ev=ev_mask, choice=repeat_mask, popt=popt,
                                       ax=ws_lsw_panel, color_scatter=colors[ip],
                                       color=colors[ip],
                                       label=lbs[ip], plot_errbars=True)
        # STORE BIAS
        mean_rep_prop.append(popt[1])
        if p == 1:
            x_ws_ls_ac = d_ws_ls['x_fit']
            y_ws_ls_ac = d_ws_ls['y_fit']
        else:
            x_ws_ls_ae = d_ws_ls['x_fit']
            y_ws_ls_ae = d_ws_ls['y_fit']
    tune_panel(ax=ws_lsw_panel, xlabel='Rep. stim. evidence',
               ylabel='Probability of repeat')
    ws_lsw_panel.axhline(y=0.5, linestyle='--', lw=0.2, color=(.5, .5, .5))
    ws_lsw_panel.legend()  # bbox_to_anchor=(1, 1.5))
    d_ws_ls = {'x_ws_ls_ac': x_ws_ls_ac, 'y_ws_ls_ac': y_ws_ls_ac,
               'x_ws_ls_ae': x_ws_ls_ae, 'y_ws_ls_ae': y_ws_ls_ae}
    return mean_rep_prop, d_ws_ls


def plot_rep_alt_psycho_curve(choice_12, ev, prev_perf, blocks,
                              rep_alt_panel, lbs):
    colors = [hf.rojo, hf.azul]
    all_means = []
    all_xs = []
    biases = []
    for i_b, blk in enumerate([1, 2]):  # blk = 1 --> alt / blk = 2 --> rep
        for p in [0, 1]:
            plt.sca(rep_alt_panel[p])
            alpha = 1 if p == 0 else 1
            lnstyl = '-' if p == 0 else '-'
            plt_opts = {'color': colors[i_b],
                        'alpha': alpha, 'linestyle': lnstyl}
            # rep/alt
            popt, pcov, ev_mask, repeat_mask =\
                hf.bias_psychometric(choice=choice_12.copy(), ev=-ev.copy(),
                                     mask=hf.and_(prev_perf == p,
                                                  blocks == blk),
                                     maxfev=100000)
            # this is to avoid rounding differences
            ev_mask = np.round(ev_mask, 2)
            d =\
                hf.plot_psycho_curve(ev=ev_mask, choice=repeat_mask,
                                     popt=popt, ax=rep_alt_panel[p],
                                     color_scatter=colors[i_b],
                                     label=lbs[p], plot_errbars=True,
```

```
                            **plt_opts)
            means = d['means']
            xs = d['xs']
            if blk == 1:
                if p == 1:
                    x_alt_ac = d['x_fit']
                    y_alt_ac = d['y_fit']
                else:
                    x_alt_ae = d['x_fit']
                    y_alt_ae = d['y_fit']
            elif blk == 2:
                if p == 1:
                    x_rep_ac = d['x_fit']
                    y_rep_ac = d['y_fit']
                else:
                    x_rep_ae = d['x_fit']
                    y_rep_ae = d['y_fit']
            all_means.append(means)
            all_xs.append(xs)
            biases.append(popt[1])
    d_curves = {'x_alt_ac': x_alt_ac, 'y_alt_ac': y_alt_ac,
                'x_rep_ac': x_rep_ac, 'y_rep_ac': y_rep_ac,
                'x_alt_ae': x_alt_ae, 'y_alt_ae': y_alt_ae,
                'x_rep_ae': x_rep_ae, 'y_rep_ae': y_rep_ae}
    return all_means, all_xs, biases, d_curves  # TODO: return bias (popt[1])


def tune_panel(ax, xlabel, ylabel, font=10):
    ax.set_xlabel(xlabel, fontsize=font)
    ax.set_ylabel(ylabel, fontsize=font)
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)


def get_hist(values, bins):
    hist, x_hist = np.histogram(values, bins=bins)
    hist = hist/np.sum(hist)
    return hist, x_hist


def extract_vars_from_dict(data, steps=None):
    steps = get_steps(steps, num_tr=len(data['correct']))
    ev = data['soundPlay_object1_leftRightBalance'][steps[0]:steps[1]]
    # TODO: change ev to decibels
    choice = data['answer_response'][steps[0]:steps[1]]
    perf = data['correct'][steps[0]:steps[1]]
    valid = data['respondedInTime'][steps[0]:steps[1]] == 1
```

```
    reaction_time = data['soundPlay_responseTime'][steps[0]:steps[1]]-FIX_TIME
    answ_rt = data['answer_responseTime'][steps[0]:steps[1]]
    sound_dur = data['soundPlay_duration'][steps[0]:steps[1]]-FIX_TIME
    blocks = data['block'][steps[0]:steps[1]]
    return ev, choice, perf, valid, reaction_time, blocks, answ_rt, sound_dur


def get_fig(figsize):
    f, ax = plt.subplots(nrows=2, ncols=5, figsize=figsize)
    return f, ax[0][0], ax[0][1], ax[0][2], [ax[1][1], ax[1][0]], ax[0][3],\
        ax[1][2], ax[1][3], ax[0][4], ax[1][4]


def get_steps(steps, num_tr):
    if steps is None:
        steps = [0, num_tr]
    else:
        if steps < 0:
            steps = [num_tr+steps, num_tr]
        else:
            steps = [0, steps]
    return steps


def process_subject(main_folder, subj, subjects, steps=None, kernels=True, conv_w=2,
                    min_num_samples=50, figsize=(18, 8)):
    """
    Process subject data.

    Parameters
    ----------
    main_folder : TYPE
        DESCRIPTION.
    subj : TYPE
        DESCRIPTION.
    steps : TYPE, optional
        DESCRIPTION. The default is None.
    conv_w : TYPE, optional
        DESCRIPTION. The default is 2.
    figsize : TYPE, optional
        DESCRIPTION. The default is (5, 5).
    margin : TYPE, optional
        DESCRIPTION. The default is 0.1.
    top_row : TYPE, optional
        DESCRIPTION. The default is 0.55.

    Returns
```

```
-------
TYPE
   DESCRIPTION.
mean_rep_prop : TYPE
   DESCRIPTION.

"""
n_bins = 75  # número de bins para histograma
# pre-process data
data = get_data(subj=subj, main_folder=main_folder)
# get relevan variables
ev, choice, perf, valid, _, _, _, _ =\
   extract_vars_from_dict(data, steps=steps)
ev = ev[valid]
choice = choice[valid]
# perf = perf[valid]
print('Percentage of invalid trials:')
print(np.sum(valid == 0)/len(valid))
print('Performance:')
print(np.sum(perf == 1)/len(perf))
f, learning_panel, psycho_panel, ws_lsw_panel, rep_alt_panel, rt_panel,\
   weights_panel, rt_mt_vs_ev, rt_vs_ev,\
   mt_vs_perf = get_fig(figsize=figsize)
# LEARNING
learning_curve, valid_curve = plot_learning(perf=perf,
                                    valid=valid, ax=learning_panel)
learning_curve = learning_curve/learning_curve[0]
valid_curve = valid_curve/valid_curve[0]
learning_panel.set_title(subj)

# TODO: remove first part of training from here
# PSYCHO-CURVE
popt, pcov = curve_fit(hf.probit_lapse_rates,
                  ev[START_ANALYSIS:], choice[START_ANALYSIS:],
                  maxfev=10000)
# bias = popt[1]

# d_individual was to save the data points to save a csv file
# d_individual =
hf.plot_psycho_curve(ev=ev[START_ANALYSIS:],
                 choice=choice[START_ANALYSIS:],
                 popt=popt, ax=psycho_panel,
                 color_scatter='k',
                 color='k', plot_errbars=True)
tune_panel(ax=psycho_panel, xlabel='Evidence of right',
         ylabel='Probability of right')
psycho_panel.axhline(y=0.5, linestyle='--', lw=0.2, color=(.5, .5, .5))
```

```
ev, choice, perf, valid, reaction_time, blocks, answ_rt, sound_dur =\
    extract_vars_from_dict(data, steps=steps)
# plot reaction times
# _, ax = plt.subplots(ncols=3)
# num_bins = 20
# # plot delay between RT and stim duration (sampling rate seems to be ~18ms)
# ax[0].hist(sound_dur-reaction_time, num_bins)
# tune_panel(ax=ax[0], xlabel='Sound duration - resp. time',
#            ylabel='Counts')
# ax[1].hist(reaction_time, num_bins)
# tune_panel(ax=ax[1], xlabel='Resp. time', ylabel='Counts')
# ax[2].hist(answ_rt, num_bins)
# tune_panel(ax=ax[2], xlabel='Answer time', ylabel='Counts')
prev_perf = np.concatenate((np.array([0]), perf[:-1]))
choice_11 = 2*(choice-0.5)
perf_11 = 2*(perf-0.5)
gt = choice_11*perf_11
# CHECK REP. PROBS.
gt_blk1 = gt[blocks == 1]  # alternating block
reps = hf.get_repetitions(gt_blk1)
print('Rep. Prob. in Alt. block: ', np.round(np.mean(reps), 3))
gt_blk2 = gt[blocks == 2]  # repeating block
reps = hf.get_repetitions(gt_blk2)
print('Rep. Prob. in Rep. block: ', np.round(np.mean(reps), 3))
choice_12 = choice + 1
choice_12[~valid] = 0
lbs = ['after error', 'after correct']
# WIN-STAY/LOSE-SWITCH, REACTION TIMES AND ZERO-COHERENCE ANALYSIS
# TODO: remove first part of training
mean_rep_prop, d_ws_ls = plot_ws_lsw_psycho_curve(
    choice_12=choice_12[START_ANALYSIS:], ev=ev[START_ANALYSIS:],
    prev_perf=prev_perf[START_ANALYSIS:], lbs=lbs,
    ws_lsw_panel=ws_lsw_panel)
valid = valid[START_ANALYSIS:]
# RT
# TODO: remove -1s
# ignore first START_ANALYSIS trials
reaction_time = reaction_time[START_ANALYSIS:]
sound_dur = sound_dur[START_ANALYSIS:]
median_rt = np.median(reaction_time)
motor_time = answ_rt[START_ANALYSIS:]+(sound_dur-reaction_time)
m_str = str(np.round(median_rt, 3))
counts, bins =\
    np.histogram(reaction_time, n_bins)
rt_panel.plot(bins[1::], counts/sum(counts), label='Median: '+m_str)
rt_panel.axvline(x=0, linestyle='--', color='k', lw=0.5)
rt_panel.axvline(x=RESP_W, linestyle='--', color='k', lw=0.5)
```

```
rt_panel.set_xlim([0, 0.5])
rt_panel.legend()
tune_panel(ax=rt_panel, xlabel='RT (s)', ylabel='Proportion')
rt_panel.legend(loc='lower right')  # bbox_to_anchor=(1, 1.5))
# tune_panel(ax=rt_panel, xlabel='Reaction time (s)', ylabel='Frequency')
# rt_panel.legend()

# REP/ALT
# plt.figure()
# plt.plot(choice)
# plt.plot(blocks)
# TODO: get bias
all_means, all_xs, biases, d_curves =\
    plot_rep_alt_psycho_curve(choice_12=choice_12[START_ANALYSIS:],
                              ev=ev[START_ANALYSIS:],
                              prev_perf=prev_perf[START_ANALYSIS:],
                              blocks=blocks[START_ANALYSIS:],
                              rep_alt_panel=rep_alt_panel, lbs=lbs)
# d_curves['x_normal'] = d_individual['x_fit']
# d_curves['y_normal'] = d_individual['y_fit']
# d_curves.update(d_ws_ls)
# df = pd.DataFrame(d_curves)
# df.to_csv(main_folder + '\\' + subj + '\\psychometric_curves_' +
#           str(subj)+'.csv')
for ax, tag in zip(rep_alt_panel, ['(Aft. Err.)', '(Aft. Corr.)']):
    tune_panel(ax=ax, xlabel='Rep. stim. evidence',
               ylabel='Probability of repeat '+tag)
    ax.axhline(y=0.5, linestyle='--', lw=0.2, color=(.5, .5, .5))

if steps is not None:
    steps = get_steps(steps=steps, num_tr=len(data['correct']))
    steps = '_'.join([str(stp) for stp in steps])
else:
    steps = 'whole_exp'
# PLOT ACCURACY VS RT
perf = perf[START_ANALYSIS:]
ev_abs = np.round(np.abs(ev), 3)[START_ANALYSIS:]
ev_vals = np.unique(ev_abs)
rt_bins = np.linspace(0, RESP_W, NUM_BINS_RT)
mean_perf_rt = []
for ev_fix in ev_vals[1:]:
    mean_perf_rt_temp = []
    for i_rt in range(len(rt_bins)-1):
        indx = np.logical_and.reduce((reaction_time > rt_bins[i_rt],
                                      reaction_time < rt_bins[i_rt+1],
                                      ev_abs == ev_fix))
        print(np.sum(indx))
```

```
        if np.sum(indx) == 0:
            print('XXXXXXXXXXXXXXXXXXXXXXXXXXXX')
            print('NON VALID')
            print('XXXXXXXXXXXXXXXXXXXXXXXXXXXX')
        mean_perf_rt_temp.append(np.mean(perf[indx]))
    mean_perf_rt_temp = np.array(mean_perf_rt_temp)
    mean_perf_rt_temp = mean_perf_rt_temp/mean_perf_rt_temp[0]
    mean_perf_rt.append(mean_perf_rt_temp)
mean_perf_rt = np.array(mean_perf_rt)
# sem_perf_rt = np.nanstd(mean_perf_rt, axis=0)/np.sqrt(3)
mean_perf_rt = np.nanmean(mean_perf_rt, axis=0)
# PLOT performance vs Reaction times
# rt_vs_perf.errorbar(rt_bins[:-1]+(rt_bins[1]-rt_bins[0])/2,
#                     mean_perf_rt, sem_perf_rt, marker='.')
# tune_panel(ax=rt_vs_perf, xlabel='RT (s)', ylabel='Accuracy')

# REACTIVE/PROACTIVE on EV vs RT
# proactive = reaction_time[reaction_time < 0.15]
# reactive = reaction_time[reaction_time >= 0.15]
# perf_reac = perf[reaction_time >= 0.15]
# perf_proac = perf[reaction_time < 0.15]
rt_threshold = 0.12
perf_reac_list = []
perf_proac_list = []
sems_reac = []
sems_proac = []
for e in ev_vals:
    indx = ev_abs == e
    perf_reac_list.append(np.mean(perf[indx*(reaction_time >= rt_threshold)]))
    perf_proac_list.append(np.mean(perf[indx*(reaction_time < rt_threshold)]))
    sems_reac.append(np.std(perf[indx*(reaction_time >= rt_threshold)]) /
                     np.sqrt(np.sum(indx*(reaction_time >= rt_threshold))))
    sems_proac.append(np.std(perf[indx*(reaction_time < rt_threshold)]) /
                      np.sqrt(np.sum(indx*(reaction_time < rt_threshold))))
rt_vs_ev.errorbar(ev_vals, perf_reac_list, sems_reac, marker='.', label='Reac')
rt_vs_ev.errorbar(ev_vals, perf_proac_list, sems_proac, marker='.',
                  label='Proac')
rt_vs_ev.legend()
tune_panel(ax=rt_vs_ev, xlabel='Stimulus strength',
           ylabel='Accuracy')
# PLOT RT VS COHERENCE
median_rt_coh = []
sems_rt_coh = []
for e in ev_vals:
    indx = ev_abs == e
    median_rt_coh.append(np.median(reaction_time[indx]))
    sems_rt_coh.append(np.std(reaction_time[indx])/np.sqrt(np.sum(indx)))
```

```
# PLOT MOTOR AND REACTION TIMES VS COHERENCE
median_mt_coh = []
sems_mt_coh = []
for e in ev_vals:
   indx = ev_abs == e
   median_mt_coh.append(np.median(motor_time[indx]))
   sems_mt_coh.append(np.std(motor_time[indx])/np.sqrt(np.sum(indx)))
rt_mt_vs_ev.errorbar(ev_vals, median_mt_coh,
                sems_mt_coh, marker='.', label='MT')
tune_panel(ax=rt_mt_vs_ev, xlabel='Stimulus strength',
         ylabel='Reaction and Motor times (s)')
rt_mt_vs_ev.errorbar(ev_vals, median_rt_coh,
                sems_rt_coh, marker='.', label='RT')
rt_mt_vs_ev.legend()
# BARPLOT WEIGHTS WITH SIGNIFICANCES
# if kernels:
#    d_kern = pd.read_csv(main_folder + '\\kernels.csv')
#    tau_dict_ini = pd.read_csv(main_folder + '\\tau_dict.csv').to_dict()
#    indx = [i for i, nam in enumerate(subjects) if nam == subj]
#    cols_ac = [val for i, val in enumerate(d_kern.columns)
#               if d_kern.columns[i].endswith('c') or
#               d_kern.columns[i].endswith('pt')
#               or d_kern.columns[i].endswith('nce')]
#    cols_ae = [val for i, val in enumerate(d_kern.columns)
#               if d_kern.columns[i].endswith('e') or
#               d_kern.columns[i].endswith('pt')
#               or d_kern.columns[i].endswith('nce')]
#    ws_ac = [d_kern[cols_ac].iloc[indx].values]
#    ws_ae = [d_kern[cols_ae].iloc[indx].values]
# else:
scores_ac = []
scores_ae = []
try:
   # tau_dict_ini = pd.read_csv(main_folder + '\\tau_dict.csv').to_dict() # Alex
   tau_dict_ini = pd.read_csv(main_folder + '/tau_dict.csv').to_dict()
   if subj in tau_dict_ini.keys():
      tau_final = tau_dict_ini[subj][0]
      ws_ac, ws_ae, df, score_ac, score_ae, p_z_ac, p_z_ae =\
         standard_glm(ev, choice_12, perf, tau_final)
   else:
      for tau in TAU_LIST:
         _, _, _, score_ac, score_ae, p_z_ac, p_z_ae =\
            standard_glm(ev, choice_12, perf, tau)
         scores_ac.append(score_ac)
         scores_ae.append(score_ae)
      combined = np.mean((np.array(scores_ac) + np.array(scores_ae))/2,
                     axis=1)
```

```
        tau_final = TAU_LIST[np.where(combined == max(combined))[0][0]]
        # tau_final = TAU_LIST[0]
        ws_ac, ws_ae, df, score_ac, score_ae, p_z_ac, p_z_ae =\
            standard_glm(ev, choice_12, perf, tau_final)
    except Exception:
        for tau in TAU_LIST:
            _, _, _, score_ac, score_ae, p_z_ac, p_z_ae =\
                standard_glm(ev, choice_12, perf, tau)
            scores_ac.append(score_ac)
            scores_ae.append(score_ae)
            combined = np.mean((np.array(scores_ac) + np.array(scores_ae))/2,
                               axis=1)
            tau_final = TAU_LIST[np.where(combined == max(combined))[0][0]]
            # tau_final = TAU_LIST[0]
            ws_ac, ws_ae, df, score_ac, score_ae, p_z_ac, p_z_ae =\
                standard_glm(ev, choice_12, perf, tau_final)
    index_ws = [i for i, x in enumerate(hf.model_cols) if x == 'T++' or
                x == 'L+' or x == 'L-']
    bars_names = ['T++_ac', 'T++_ae', 'L+_ac', 'L+_ae', 'L-_ac', 'L-_ae']
    heights = []
    w_ac = ws_ac[0][0]
    w_ae = ws_ae[0][0]
    p_list = []
    for i in [0, 1, 2]:
        if i == 0:
            heights.append(w_ac[index_ws[2]])
            heights.append(w_ae[index_ws[2]])
            p_list.append(p_z_ac['T++'])
            p_list.append(p_z_ae['T++'])
        elif i == 1:
            heights.append(w_ac[index_ws[0]])
            heights.append(w_ae[index_ws[0]])
            p_list.append(p_z_ac['L+'])
            p_list.append(p_z_ae['L+'])
        else:
            heights.append(w_ac[index_ws[1]])
            heights.append(w_ae[index_ws[1]])
            p_list.append(p_z_ac['L-'])
            p_list.append(p_z_ae['L-'])
    weights_panel.bar(bars_names, heights)
    weights_panel.tick_params(axis='x', labelrotation=45)
    for i, pv in enumerate(p_list):
        text = ''
        p = .05
        while pv < p and p > 0.05/1e3:
            text += '*'
            p /= 10.
```

```
    weights_panel.text(x=i-0.2, y=heights[i], s=text,
                       size='large')
weights_panel.text(x=1.5, y=heights[1]+0.5, s='Tau: '+str(tau_final))
weights_panel.set_ylabel('GLM weights')
# PLOT ACCURACY VS MOTOR
mt_bins = np.linspace(MT_MIN, MT_MAX, NUM_BINS_MT)
print('Number of MT below MIN', np.sum(motor_time < MT_MIN))
print('Number of MT above MAX', np.sum(motor_time > MT_MAX))
print('Minimum MT', np.min(motor_time))
print('Max MT', np.max(motor_time))
mean_mt_perf = []
sem_mt_perf = []
for i_mt in range(len(mt_bins)-1):
    indx = np.logical_and(motor_time > mt_bins[i_mt],
                          motor_time < mt_bins[i_mt+1])
    print(np.sum(indx))
    if np.sum(indx) > min_num_samples:
        mean_mt_perf.append(np.mean(perf[indx]))
        sem_mt_perf.append(np.std(perf[indx])/np.sqrt(np.sum(indx)))
    else:
        mean_mt_perf.append(np.nan)
        sem_mt_perf.append(np.nan)

mt_vs_perf.errorbar(mt_bins[:-1]+(mt_bins[1]-mt_bins[0])/2,
                    mean_mt_perf, sem_mt_perf, marker='.')
tune_panel(ax=mt_vs_perf, xlabel='MT (s)', ylabel='Accuracy')
# TODO: PLOT CUMMULATIVE RT VS ???

f.savefig(main_folder+'\\'+subj+'\\psycho_curve_'+steps+'.png', dpi=400,
          bbox_inches='tight')  # Alex
# f.savefig(main_folder+subj+'/psycho_curve_'+steps+'.svg', dpi=400,
#           bbox_inches='tight')  # Manuel
plt.close(f)
# return exponential decays with tau
index_kernel = [i for i, x in enumerate(hf.model_cols) if x == 'T++']
decay_ac = w_ac[index_kernel[0]]*np.exp(-np.arange(10)/tau_final)
decay_ac[5] = sum(decay_ac[5:10])
decay_ac = decay_ac[0:6]
decay_ac = decay_ac[::-1]
decay_ae = w_ae[index_kernel[0]]*np.exp(-np.arange(10)/tau_final)
decay_ae[5] = sum(decay_ae[5:10])
decay_ae = decay_ae[0:5]
decay_ae = decay_ae[::-1]
# TODO: return dictionary
# lst = [np.mean(perf[valid]), mean_rep_prop, median_rt, all_means,
#        all_xs, mean_perf_rt, median_mt_coh, median_rt_coh, mean_mt_perf,
#        ev_vals, motor_time, learning_curve, valid_curve, biases]
```

```
    # stg = ["np.mean(perf[valid]), mean_rep_prop, median_rt, all_means,"
    #        "all_xs, mean_perf_rt, median_mt_coh, median_rt_coh, mean_mt_perf,"
    #        "ev_vals, motor_time, learning_curve, valid_curve, biases"]
    # d = hf.list_to_dict(lst, stg)
    return np.mean(perf[valid]), mean_rep_prop, median_rt, all_means,\
        all_xs, mean_perf_rt, median_mt_coh, median_rt_coh, mean_mt_perf, \
        ev_vals, motor_time, learning_curve, valid_curve, biases, perf, \
        choice_12[START_ANALYSIS:], ev_abs, perf_reac_list, \
        perf_proac_list, ws_ac, ws_ae, df, score_ac, score_ae, tau_final,\
        decay_ac, decay_ae, reaction_time, sems_proac
    # TODO: add bias


def psycho_analysis(main_folder, subjects, steps=[None], name=''):
    """
    Go over all subjects and plot summary figs.

    Parameters
    ----------
    main_folder : str
        folder where the data for all subjects is.
    subjects : list
        list with the name of the folders corr. to each subject.
    steps : list, optional
        timesteps to process subject for different periods during
        training ([None])

    Returns
    -------
    None.

    """
    # TODO: adjust size
    f2, ax2 = plt.subplots(ncols=4, nrows=2, figsize=(8, 5),
                           sharey=True)
    f2.suptitle('weights' + name + 'ms')
    m_rep_probs = []
    m_perf = []
    m_rt_VS_perf = []
    m_stimStr_VS_rt = []
    m_stimStr_VS_mt = []
    m_reaction_times = []
    m_mt_VS_perf = []
    median_rep_alt = []
    cohs_rep_alt = []
    all_rt_mat = []
    all_mt_mat = []
```

```
all_learn_curv = []
all_valid_curve = []
bias_total = []
d_final = {}
y_cols = ['y_Alt_ae', 'y_Rep_ae', 'y_Alt_ac', 'y_Rep_ac']
x_cols = ['x_Alt_ae', 'x_Rep_ae', 'x_Alt_ac', 'x_Rep_ac']
f_ri, ax_ri = plt.subplots(1)
ax_ri.set_title(name + ' ms')
ax_ri.set_ylabel('Reset index')
# resets = {}
# kernels_over_time_ac = {}
# kernels_over_time_ae = {}
# times_final = {}
d_kernels = {}
reset_total = []
tau_dict = {}
decays_ac = []
decays_ae = []
acc_rte = {}
rte_bins = {}
proac_arr = np.zeros((len(subjects), 4))
err_proac_arr = np.zeros((len(subjects), 4))
time_delay = []
# reac_times = []
reverse_st = {}
reset_st = {}
ae_list = []
cdf_tog_all = []
bins_all = []
st_mod = []
for i_s, subj in enumerate(subjects):
    print('-----------')
    print(subj)
    # lbls = ['', subj]
    # go over steps to get psychometric curves for different periods
    for i_stp, stp in enumerate(steps):
        print('---')
        m_p, m_r_prop, m_rt, m_repalt, c_repalt,\
            m_perf_rt, m_mt_coh, m_rt_coh, m_perf_mt, ev_vals, all_mt,\
            l_c, v_c, biases, perf, choice_12, \
            ev, _, proac, ws_ac, ws_ae, df, score_ac, score_ae, tau_final,\
            decay_ac, decay_ae, all_rt, sems_proac =\
            process_subject(main_folder, subj, subjects, steps=stp, kernels=True)
        m_perf.append(m_p)
        m_rep_probs.append(m_r_prop)
        m_rt_VS_perf.append(m_perf_rt)
        m_stimStr_VS_rt.append(m_rt_coh)
```

```
m_stimStr_VS_mt.append(m_mt_coh)
m_mt_VS_perf.append(m_perf_mt)
m_reaction_times.append(m_rt)
all_mt_mat.append(all_mt)
median_rep_alt.append(np.array([np.array(a) for a in m_repalt]))
cohs_rep_alt.append(np.array([np.array(a) for a in c_repalt]))
acc_vs_rte, r_bins = hf.tachometric_curves(perf, all_rt, ev_vals,
                                np.abs(ev))
acc_rte[subj] = acc_vs_rte
rte_bins[subj] = r_bins
for ind, j in enumerate(m_repalt):
    d_final[subj + '_' + y_cols[ind]] = m_repalt[ind]
    d_final[subj + '_' + x_cols[ind]] = c_repalt[ind]
all_rt_mat.append(all_rt)
all_learn_curv.append(l_c)
all_valid_curve.append(v_c)
bias_total.append(biases)
hf.call_plot_kernels(ws_ac, ws_ae, ax2)
# hf.check_dictionary(data=df, start=250, num_tr=50)
reset = hf.compute_reset_index(ws_ac, ws_ae, tau_final)
prev_perf = np.concatenate((np.array([0]), perf[:-1]))
after_error = perf[(prev_perf == 0)*(ev == 0)]
ae_list.append(np.mean(after_error))
# after_correct = perf[prev_perf == 1]
if reset > 0.66:
    reset_st[subj+'_ae'] = np.mean(after_error)
    reset_st[subj+'_acc'] = np.mean(perf)
    # reset_st[subj+'_ac'] = np.mean(after_correct)
if reset <= 0.66:
    reverse_st[subj+'_ae'] = np.mean(after_error)
    reverse_st[subj+'_acc'] = np.mean(perf)
hf.plot_reset_index(reset,
                    ax_ri, step=None, per=None, xs=0.5)
reset_total.append(reset)
d_kernels = hf.kernel_dicts(ws_ac, ws_ae, d_kernels)
decays_ac.append(decay_ac)
decays_ae.append(decay_ae)
tau_dict[subj] = tau_final
cdf_tog, reacbins, tdel, _, _ =\
    hf.time_delay(all_rt, ev_vals, np.abs(ev))
cdf_tog_all.append(cdf_tog)
bins_all.append(reacbins)
# hf.plot_time_delay_cdf(cdf_tog, cdf_shift_tog, rt_bins)
proac_arr[i_s, :] = proac
err_proac_arr[i_s, :] = sems_proac
time_delay.append([td for td in tdel])
modulation = hf.cdf_comparison(ev, all_rt, ev_vals)
```

```
        st_mod.append(modulation)
        # reac_times.append([round(rt, 4) for rt in rt_bins])
d_kernels_df = pd.DataFrame(d_kernels, index=[a for a in range(20)])
d_kernels_df.to_csv(main_folder + '\\kernels' + '.csv')
tau_df = pd.DataFrame(tau_dict, index=[0])
tau_df.to_csv(main_folder + '\\tau_dict' + '.csv')
hf.boxplot_ae_acc(reset_st, reverse_st)
hf.plot_all_tachometric(ev_vals, acc_rte, rte_bins)
hf.boxplot_reset_ind(reset_total, ax_ri)
hf.plot_all_tachometric(ev_vals, acc_rte, rte_bins)
hf.plot_ac_ae(d_kernels['T++_ae'], d_kernels['T++_ac'])
hf.plot_expo_kernel(decays_ac, decays_ae)
# start analysis
median_rep_alt = np.array(median_rep_alt)
cohs_rep_alt = np.array(cohs_rep_alt)
m_rt_VS_perf = np.array(m_rt_VS_perf)
m_stimStr_VS_rt = np.array(m_stimStr_VS_rt)
m_stimStr_VS_mt = np.array(m_stimStr_VS_mt)
m_mt_VS_perf = np.array(m_mt_VS_perf)
# TODO: use array to get different lists
bias_total = np.array(bias_total)
bias_alt_error = bias_total[:, 0]
bias_alt_correct = bias_total[:, 1]
bias_rep_error = bias_total[:, 2]
bias_rep_correct = bias_total[:, 3]
bias_total = [bias_rep_correct, bias_alt_correct,
              bias_rep_error, bias_alt_error]
# psycho plots: create figure
f, ax = plt.subplots(nrows=2, ncols=4, figsize=(15, 9))
f.suptitle(name)
ax = ax.flatten()
# learning
min_l = np.min([len(x) for x in all_learn_curv])
all_learn_curv = [x[:min_l] for x in all_learn_curv]
all_learn_curv = np.array(all_learn_curv)
# valid
min_l = np.min([len(x) for x in all_valid_curve])
all_valid_curve = [x[:min_l] for x in all_valid_curve]
all_valid_curve = np.array(all_valid_curve)
# m_stimStr_VS_rt
first_value_Str_rt = m_stimStr_VS_rt[:, 0]
m_stimStr_VS_rt = m_stimStr_VS_rt/first_value_Str_rt[:, None]
# m_stimStr_VS_mt
first_value_Str_mt = m_stimStr_VS_mt[:, 0]
m_stimStr_VS_mt = m_stimStr_VS_mt/first_value_Str_mt[:, None]
# m_rt_VS_perf
first_value_m_rt_VS_perf = m_rt_VS_perf[:, 0]
```

```
m_rt_VS_perf = m_rt_VS_perf/first_value_m_rt_VS_perf[:, None]

# MEAN PSYCHO-CURVES FOR REP/ALT, AFTER CORRECT/ERROR
counter = -1
colors = [hf.rojo, hf.azul]
ttls = ['After error', 'After correct']
bias_final = []
slope_final = []
for i_b, blk in enumerate([1, 2]):
    for ip, p in enumerate([0, 1]):
        counter += 1
        if i_b == 0:
            ax[2-ip].axvline(x=0., linestyle='--', lw=0.2,
                             color=(.5, .5, .5))
            ax[2-ip].axhline(y=0.5, linestyle='--', lw=0.2,
                             color=(.5, .5, .5))
            ax[2-ip].set_title(ttls[ip])
            ax[2-ip].set_yticks([0, 0.5, 1])
            tune_panel(ax=ax[2-ip], xlabel='Rep. stim. evidence',
                       ylabel='Prop. repeated responses')
        ax[2-ip].plot(cohs_rep_alt[:, counter, :].flatten(),
                      median_rep_alt[:, counter, :].flatten(),
                      color=colors[i_b], alpha=0.2, linestyle='',
                      marker='+')
        medians = np.median(median_rep_alt, axis=0)[counter]
        sems = np.std(median_rep_alt, axis=0)[counter] /\
            np.sqrt(median_rep_alt.shape[0])
        ax[2-ip].errorbar(cohs_rep_alt[0][0], medians, sems,
                          color=colors[i_b], marker='.', linestyle='')
        ev_gen = cohs_rep_alt[0, counter, :].flatten()
        popt, pcov = curve_fit(hf.probit_lapse_rates,
                               ev_gen,
                               np.median(median_rep_alt, axis=0)[counter],
                               maxfev=10000)
        bias_final.append(popt[1])
        slope_final.append(popt[0])
        x_fit = np.linspace(-np.max(ev), np.max(ev), 20)
        y_fit = hf.probit_lapse_rates(x_fit, popt[0], popt[1], popt[2], popt[3])
        ax[2-ip].plot(x_fit, y_fit, color=colors[i_b])
box_plot(data=m_perf, ax=ax[0], x=0, lw=.5, fliersize=4, color='k',
         widths=0.15)
regs = ['L+_ac', 'L+_ae', 'L-_ac', 'L-_ae', 'T++_ac', 'T++_ae']
kernels = [d_kernels[key] for key in d_kernels.keys() if key in regs]
ax[3].boxplot(kernels)
ax[3].set_xticklabels(regs)
ax[3].tick_params(axis='x', labelrotation=45)
tune_panel(ax=ax[0], ylabel='Mean performance', xlabel='')
```

```
tune_panel(ax=ax[1], ylabel='Median reaction times', xlabel='')
# TODO: plot individual traces (alpha=0.2)
rt_bins = np.linspace(0, RESP_W, NUM_BINS_RT)
# ev_abs = np.round(np.abs(ev), 3)
# ev_vals = np.unique(ev_abs)
# mt_bins = np.linspace(MT_MIN, MT_MAX, NUM_BINS_MT)
ax[5].plot(rt_bins[:-1]+(rt_bins[1]-rt_bins[0])/2,
         m_rt_VS_perf.T, alpha=0.2, color='k')
ax[5].errorbar(rt_bins[:-1]+(rt_bins[1]-rt_bins[0])/2,
             np.nanmedian(m_rt_VS_perf, axis=0),
             np.nanstd(m_rt_VS_perf, axis=0) /
             np.sqrt(m_rt_VS_perf.shape[0]),
             marker='.', color='k')
ax[6].plot(ev_vals, m_stimStr_VS_rt.T, alpha=0.2, color='k')
ax[6].errorbar(ev_vals, np.median(m_stimStr_VS_rt, axis=0),
             np.std(m_stimStr_VS_rt, axis=0) /
             np.sqrt(m_rt_VS_perf.shape[0]),
             marker='.', color='k')
ax[6].plot(ev_vals, m_stimStr_VS_mt.T, alpha=0.2, color='g')
ax[6].errorbar(ev_vals, np.median(m_stimStr_VS_mt, axis=0),
             np.std(m_stimStr_VS_mt, axis=0) /
             np.sqrt(m_rt_VS_perf.shape[0]),
             marker='.', color='g')
ax[6].legend(['', '', 'RT', 'MT'])
rt_num_bins = 20
atm_h_list = []
# vals inferred by looking at hists
for atm in all_rt_mat:
    atm_hist, atm_bins = np.histogram(atm, bins=rt_num_bins, range=[0, 0.35])
    atm_hist = atm_hist/np.sum(atm_hist)
    ax[4].plot(atm_bins[:-1]+(atm_bins[1]-atm_bins[1]) /
             2, atm_hist, color='k', alpha=0.25)
    atm_h_list.append(atm_hist)
mean_rt = np.mean(atm_h_list, axis=0)
ax[4].plot(atm_bins[:-1]+(atm_bins[1]-atm_bins[1]) /
          2, mean_rt, color='k', alpha=1, linewidth=2)
# Boxplot bias:
# ax[7].boxplot(bias_total)
# ax[7].scatter(np.repeat([1, 2, 3, 4], len(subjects)), bias_total,
#               alpha=0.5, marker='x',
#               c=hf.gris)
# ax[7].plot([np.repeat(1, len(subjects)), np.repeat(2, len(subjects)),
#             np.repeat(3, len(subjects)), np.repeat(4, len(subjects))],
#            bias_total, c=hf.gris, linewidth=0.5)
# ax[7].set_xticks([1, 2, 3, 4])
# ax[7].set_xticklabels(["rep cor", "alt cor", "rep err",
#                        "alt err"])
```

```
    # ax[7].set_ylim([-1.5, 1])
    # Proac
    for i, pr in enumerate(proac_arr):
        ax[7].errorbar(ev_vals, pr, err_proac_arr[i], marker='.',
                       label='Proac', alpha=0.1, color='gray', capsize=8,
                       fmt='o-')
        ax[7].scatter(ev_vals, pr, linewidth=1, color='gray', alpha=0.08)

    mean_proac = np.nanmean(proac_arr, axis=0)
    mean_err = np.nanmean(err_proac_arr, axis=0)
    ax[7].errorbar(ev_vals, mean_proac, mean_err, marker='.',
                   label='Proac', alpha=1, color='k', capsize=8, fmt='o-',
                   linewidth=2.5)
    ax[7].scatter(ev_vals, mean_proac, linewidth=2, color='k')
    tune_panel(ax=ax[7], xlabel='Stimulus strength',
               ylabel='Accuracy')
    # TODO: tune panels
    tune_panel(ax=ax[4], xlabel='RT', ylabel='Proportion')
    tune_panel(ax=ax[5], ylabel='Accuracy', xlabel='Reaction time')
    tune_panel(ax=ax[6], ylabel='RT and MT', xlabel='Stimulus strength')
    # f.savefig(main_folder+'rep_props_'+name +
    #           '.svg', dpi=400, bbox_inches='tight')
    f.savefig(main_folder+'\\rep_props_'+name +
              '.png', dpi=400, bbox_inches='tight')
    # f, ax = plt.subplots(nrows=1, ncols=1, figsize=(4, 2))
    # ax.plot(all_learn_curv.T, color='k', alpha=0.2)
    # ax.plot(np.median(all_learn_curv, axis=0), color='k')
    # tune_panel(ax=ax, ylabel='Accuracy', xlabel='Trials')
    # f, ax = plt.subplots(nrows=1, ncols=1, figsize=(4, 2))
    # ax.plot(all_valid_curve.T, color='k', alpha=0.2)
    # ax.plot(np.median(all_valid_curve, axis=0), color='k')
    # tune_panel(ax=ax, ylabel='Proportion of valid trials', xlabel='Trials')


def process_trajectories_rep_alt(data_tr, data_traj):
    """
    With this function, data from trials and trajectories is extracted.
    We want to study the slope of the lines so we take the diff on each
    trajectory and then we measure the average slope for each trajectory.
    Since the objective points are at ratio 2:1 to the center, each objective
    point is at ratio 1:1 to the starting point so the slope must be
    approximately 1. With this function we are computing the slope, the
    trajectories and separating them by correct/incorrect, to see if there
    is some bias in these.
    ***
    Inputs:
        data_tr : data from trials
```

```
        data traj: data from trajectories
        condition: wheter the user wants to see repetition (rep),
        alternation (alt), or total (all) data
    Outputs:
        dict_all: has information about trajectories, slopes, left/right
        choices, correct/incorrect choices
    """
    choice = np.array(data_tr['answer_response'])
    correct = np.array(data_tr['correct'])
    x_traj = [x for x in data_traj['answer_positionsX']
              if x not in [np.nan]]
    y_traj = [x for x in data_traj['answer_positionsY']
              if x not in [np.nan]]
    times = [x for x in data_traj['answer_times']
             if x not in [np.nan]]
    difference_x = []
    difference_y = []
    slope = []
    slope_mean = []
    choice_f = []
    correct_f = []
    slope_correct = []
    slope_incorrect = []
    xtraj_correct = []
    ytraj_correct = []
    xtraj_incorrect = []
    ytraj_incorrect = []
    right_correct = []
    right_incorrect = []
    left_correct = []
    left_incorrect = []
    right = []
    left = []
    ind_cor = []
    ind_incor = []
    # first we process the data from the repetition pattern
    for inde in range(len(x_traj)):
        x_traj[inde] = str(x_traj[inde]).split(';')
        y_traj[inde] = str(y_traj[inde]).split(';')
        times[inde] = str(times[inde]).split(';')
        for i in range(len(x_traj[inde])):
            x_traj[inde][i] = float(x_traj[inde][i])
            times[inde][i] = float(times[inde][i])
        difference_x.append(np.diff(x_traj[inde]))  # compute the diff in x
        for j in range(len(y_traj[inde])):
            y_traj[inde][j] = float(y_traj[inde][j])
        difference_y.append(np.diff(y_traj[inde]))  # compute the diff in y
```

```
    slope_dummy = []
    for r in range(len(np.diff(y_traj[inde]))):
        # compute the slope by steps of a trajectory
        slope_dummy.append(np.diff(y_traj[inde])[r] /
                           np.diff(x_traj[inde])[r])
    slope.append(slope_dummy)
    slope_mean.append(np.mean(slope_dummy))  # compute the mean of slope
    choice_f.append(int(choice[inde]))  # save choice
    correct_f.append(correct[inde])  # save correct answer
    if correct[inde] == 1:
        # if subject is correct
        slope_correct.append(np.mean(slope_dummy))  # save slope
        xtraj_correct.append(x_traj[inde])  # save trajectory in x
        ytraj_correct.append(y_traj[inde])  # save trajectory in y
        ind_cor.append(inde)
        if int(choice[inde]) == 1:  # if choice is right
            right_correct.append(np.mean(slope_dummy))  # append mean
            right.append(inde)  # also append the index
        else:
            left_correct.append(np.mean(slope_dummy))  # same with left
            left.append(inde)
    else:  # if choice is incorrect
        slope_incorrect.append(np.mean(slope_dummy))  # same as before
        xtraj_incorrect.append(x_traj[inde])
        ytraj_incorrect.append(y_traj[inde])
        ind_incor.append(inde)
        if int(choice[inde]) == 1:
            right_incorrect.append(np.mean(slope_dummy))
            right.append(inde)
        else:
            left_incorrect.append(np.mean(slope_dummy))
            left.append(inde)
slope_mean = np.nan_to_num(slope_mean, copy=True, posinf=10)
slope_incorrect = np.nan_to_num(slope_incorrect, copy=True, posinf=10)
slope_correct = np.nan_to_num(slope_correct, copy=True, posinf=10)
right_incorrect = np.nan_to_num(right_incorrect, copy=True, posinf=10)
right_correct = np.nan_to_num(right_correct, copy=True, posinf=10)
left_incorrect = np.nan_to_num(left_incorrect, copy=True, posinf=10)
left_correct = np.nan_to_num(left_correct, copy=True, posinf=10)
# replace infinite by 10 and save dictionary
dict_all = {'slope': slope, 'mean': slope_mean, 'choice': choice,
            'correct': correct, 'x_correct': xtraj_correct,
            'y_correct': ytraj_correct,
            'x_incorrect': xtraj_incorrect,
            'y_incorrect': ytraj_incorrect,
            'slope_correct': slope_correct,
            'slope_incorrect': slope_incorrect,
```

```
            'right_correct': right_correct, 'left_correct': left_correct,
            'left_incorrect': left_incorrect,
            'right_incorrect': right_incorrect, 'right': right,
            'left': left, 'x_traj': x_traj, 'y_traj': y_traj,
            'ind_cor': ind_cor, 'ind_incor': ind_incor, 'times': times}
    return dict_all


def change_of_mind(dict_n, threshold):
    first_thought = np.zeros(len(dict_n['x_traj']))
    final_thought = np.zeros(len(dict_n['x_traj']))
    change_o_m = np.zeros(len(dict_n['x_traj']))
    corr = np.zeros(len(dict_n['x_traj']))
    x_points = [np.array(x)-x[0] for x in dict_n['x_traj']]
    correct = dict_n['correct']
    indx_com = []
    indx_no_com = []
    for ind, x_list in enumerate(dict_n['x_traj']):
        # dummy_first = []
        # for x in x_list:
        #     if abs(x) < threshold:
        #         dummy_first.append(x)
        # if len(dummy_first) == 0:
        #     max_num = x_points[ind][1]
        # else:
        #     max_num = max(dummy_first, key=abs)
        if abs(x_points[ind][1]) < threshold:
            max_num = x_list[1]
        else:
            max_num = x_points[ind][1]
        if max_num > 0:
            first_thought[ind] = 1
        else:
            first_thought[ind] = -1
        final_number = x_list[-1]
        if final_number > 0:
            final_thought[ind] = 1
        else:
            final_thought[ind] = -1
        if first_thought[ind] == final_thought[ind]:
            change_o_m[ind] = int(0)
            indx_no_com.append(ind)
        else:
            change_o_m[ind] = int(1)
            indx_com.append(ind)
        if change_o_m[ind] == 1 and correct[ind] == 0:
            corr[ind] = 1
```

```
        else:
            corr[ind] = 0
    if len(correct[indx_no_com]) != 0:
        prop_incorrect_no_com = sum(correct[indx_no_com] == 0) / \
                                len(correct[indx_no_com])
    # incorrect w/o change of mind
    prop_incorrect_com = sum(correct[indx_com] == 0) / len(correct[indx_com])
    # incorrect with change of mind
    proportion = sum(corr)/sum(correct == 0)  # general proportion
    # changes of mind over incorrect answers
    proportion_com = len(indx_com) / (len(indx_no_com) + len(indx_com))
    prop_errors = sum(correct == 0)/len(correct)
    return change_o_m, indx_no_com, indx_com, proportion,\
        prop_incorrect_no_com, prop_incorrect_com, proportion_com, prop_errors


def plot_slope_traj(dict_n):
    """
    This function receives a dictionary from the process_trajectories_rep_alt
    function and plots distributuons of slopes and trajectories, with different
    colors for correct/incorrect.
    ***
    Input:
        dict_n : dictionary from previous function
    Output:
        histogram of mean slope (for correct and incorrect)
        boxplot of mean slope
        plots trajectories
        boxplot of mean slopes, differences between correct/incorrect
        boxplot of mean slopes, differences between right/left +
        correct/incorrect
    """
    # histogram
    plt.figure('Histogram')
    plt.hist(abs(dict_n['mean']), 100)
    plt.ylabel('Counts')
    plt.xlabel('Mean of slope')
    plt.title("Histogram of the slope's mean absolute value")
    plt.xlim([0, 5])
    # boxplot
    plt.figure('Boxplot')
    plt.boxplot(abs(dict_n['mean'][dict_n['mean'] < 7]))
    plt.title("Boxplot of the slope's mean absolute value")
    # separated correct/incorrect histograms
    plt.figure('Histograms')
    plt.subplot(1, 2, 1)
    plt.hist(np.nan_to_num(dict_n['slope_correct'], copy=True, posinf=15), 90,
```

```
        label='correct', color='g')
plt.ylabel('Counts')
plt.xlabel('Mean slope in correct choices')
plt.legend()
plt.subplot(1, 2, 2)
plt.hist(np.nan_to_num(dict_n['slope_incorrect'], copy=True, posinf=15),
        90, label='incorrect', color='r')
plt.ylabel('Counts')
plt.xlabel('Mean slope in incorrect choices')
plt.legend()
# Trajectories
plt.figure('trajectories')
# N = min(len(dict_n['x_correct']), len(dict_n['x_incorrect']))
for trajectory in range(19, 200):
    if trajectory in dict_n['ind_cor']:
        plt.plot(dict_n['x_correct'][trajectory],
                 dict_n['y_correct'][trajectory], color='g',
                 linewidth=0.5)
    else:
        plt.plot(dict_n['x_incorrect'][trajectory],
                 dict_n['y_incorrect'][trajectory], color='r',
                 linewidth=0.5)
    plt.xlim([-700, 700])
plt.legend(labels=['correct', 'incorrect'])
plt.title('Trajectories')
plt.ylabel('Y direction')
plt.xlabel('X direction')
# Boxplot of mean slopes if correct or incorrect
abs_correct = [abs(x) for x in dict_n['slope_correct'] if abs(x) < 4]
abs_incorrect = [abs(x) for x in dict_n['slope_incorrect'] if abs(x) < 4]
plt.figure('boxplots_1')
plt.boxplot([abs_correct, abs_incorrect])
plt.xticks([1, 2], ['Correct', 'Incorrect'])
# Right/Left boxplot
right_correct = dict_n['right_correct']
right_incorrect = dict_n['right_incorrect']
right = np.concatenate((right_correct, right_incorrect))
plt.figure('boxplots_2')
plt.subplot(1, 3, 1)
plt.boxplot([right_correct, right_incorrect])
plt.ylim([-1, 1.5])
plt.xticks([1, 2], ['right_correct', 'right_incorrect'])
left_correct = dict_n['left_correct']
left_incorrect = dict_n['left_incorrect']
plt.subplot(1, 3, 2)
plt.boxplot([left_correct, left_incorrect])
plt.xticks([1, 2], ['left_correct', 'left_incorrect'])
```

```
    plt.ylim([-2, 2])
    left = np.concatenate((left_correct, left_incorrect))
    plt.subplot(1, 3, 3)
    plt.boxplot([right, left])
    plt.xticks([1, 2], ['right', 'left'])
    plt.ylim([-3, 3])
    plt.figure('Change of mind')
    # N1 = len(dict_n['x_traj'])
    change_o_m, indx_no_com, indx_com, proportion,\
       prop_incorrect_no_com, prop_incorrect_com, proportion_com\
       = change_of_mind(dict_n, 200)
    for traj in range(19, 1019):  # 181 is a bit strange
       if traj in indx_com:
          plt.plot(dict_n['x_traj'][traj],
                dict_n['y_traj'][traj], color='g', linewidth=0.5,
                label='Change of mind')
       if traj in indx_no_com:
          plt.plot(dict_n['x_traj'][traj],
                dict_n['y_traj'][traj], color='r', linewidth=0.1,
                label='No change of mind')
    plt.legend(labels=['No change of mind', 'Change of mind'])
    plt.title('Trajectories')
    plt.ylabel('Y direction')
    plt.xlabel('X direction')
    return


def traj_analysis(main_folder, subjects, steps=[None], name=''):
    prop_inc_com = []
    prop_inc_no_com = []
    prop_com = []  # len(indx_com) / (len(indx_no_com) + len(indx_com))
    prop_correlation = []  # sum(corr)/sum(correct == 0), corr is 1 when C.O.M.
    prop_err_general = []
    prop_acc_general = []
    for i_s, subj in enumerate(subjects):
       print('-----------')
       print(subj)
       for i_stp, stp in enumerate(steps):
          data_tr, data_traj = get_data_traj(subj, main_folder)
          dict_n = process_trajectories_rep_alt(data_tr, data_traj)
          # plot_slope_traj(dict_n)
          change_o_m, indx_no_com, indx_com, proportion,\
             prop_incorrect_no_com, prop_incorrect_com, proportion_com,\
             prop_errors = change_of_mind(dict_n, 200)
          prop_inc_com.append(prop_incorrect_com)
          prop_inc_no_com.append(prop_incorrect_no_com)
          prop_com.append(proportion_com)
```

```
            prop_correlation.append(proportion)
            prop_err_general.append(prop_errors)
            prop_acc_general.append(1-prop_errors)
    # boxplots of proportion of incorrects when COM and when not COM
    fig, ax = plt.subplots(nrows=1, ncols=3)
    ax = ax.flatten()
    ax[0].boxplot([prop_inc_no_com, prop_inc_com, prop_com])
    ax[0].scatter(np.repeat(1, len(prop_inc_no_com)), prop_inc_no_com,
                alpha=0.5, marker='x', c=hf.gris)
    ax[0].scatter(np.repeat(2, len(prop_inc_com)), prop_inc_com,
                alpha=0.5, marker='x', c=hf.gris)
    ax[0].scatter(np.repeat(3, len(prop_com)), prop_com,
                alpha=0.5, marker='x', c=hf.gris)
    ax[0].set_xticks([1, 2, 3])
    ax[0].set_xticklabels(['Inc_COM',
                           'Inc_NO_COM', 'Changes of mind'])
    ax[0].set_ylabel('Proportion')
    ax[0].set_xlabel('Proportion of changes of mind')
    ax[0].set_title(name + ' ms')
    # scatter plot of accuracy vs proportion of change of mind
    ax[1].scatter(prop_com, prop_acc_general)
    ax[1].set_ylabel('General accuracy')
    ax[1].set_xlabel('Proportion of changes of mind')
    ax[1].set_title(name + ' ms')
    # boxplots of proportion of incorrects general
    ax[2].scatter(prop_com, prop_inc_com, label='Incorrect_COM')
    ax[2].scatter(prop_com, prop_inc_no_com, label='Incorrect_NO_COM')
    ax[2].set_ylabel('Error proportion')
    ax[2].set_xlabel('Proportion')
    ax[2].legend()
    ax[2].set_title(name + ' ms')
    fig2, ax2 = plt.subplots(1)
    ax2.plot(dict_n['times'][[indx_com][0][90]][1:-1],
            dict_n['x_traj'][[indx_com][0][90]][1:-1], 'o-', marker='x')
    ax2.set_xlabel('Time from movement (ms)')
    ax2.set_xticks([0, 0.1, 0.2, 0.3, 0.4, 0.5])
    ax2.set_xticklabels([0, 100, 200, 300, 400, 500])
    ax2.set_ylabel('X position')
    ax2.set_title('Change of mind')
    return


def standard_glm(ev, choice_12, perf, tau):
    # data = np.load(folder+'/bhvr_data_all.npz', allow_pickle=1)
    # assert len(data) > 0
    # TODO: get functions from helper_functions_reset_paper.py and
    # copy them in helper_functions.py
```

```
    data = {'signed_evidence': ev, 'choice': choice_12,
            'performance': perf}
    df = hf.get_GLM_regressors(data, tau, chck_corr=False)
    fit_ac, fit_ae, score_ac, score_ae, p_z_ac, p_z_ae = hf.glm(df)
    ws_ac = fit_ac.coef_[None, :, :]
    ws_ae = fit_ae.coef_[None, :, :]
    return ws_ac, ws_ae, df, score_ac, score_ae, p_z_ac, p_z_ae


def glm_krnls(main_folder='/home/molano/priors/rats/', tag='.mat', x=0,
              axs_glm_krnls=None, ax_inset=None, color=None, name='', contr_th=.05,
              tags_mat=[['T++', 'T-+', 'T+-', 'T--']], sv_folder=None,
              ax_wsls=None, plt_ind_trcs=True, plt_ind_indx=True, ax_2d_plot=None):
    xtcks = ['T++'+x for x in ['2', '3', '4', '5', '6-10']]
    sv_folder = sv_folder or main_folder+'figures'
    if not os.path.exists(sv_folder):
        os.makedirs(sv_folder)
    files = glob.glob(main_folder+'*'+tag+'*')
    # figures
    if axs_glm_krnls is None:
        f_glm, ax_tr = plt.subplots(nrows=2, ncols=2, sharey=True,
                                    figsize=(8, 6))
        # f_glm, ax = plt.subplots(nrows=1, ncols=4, sharey=True,
        #                          figsize=(16, 3))
        ax_inset = plt.axes((0.22, 0.45, 0.06, 0.425))
        axs_glm_krnls = [ax_tr.flatten()]
        sv_figs = True
    else:
        sv_figs = False
    # plotting options
    plot_opts = {'alpha': 0.2, 'fntsz': 12}
    if color is not None:
        plot_opts['color_ac'] = color
        plot_opts['color_ae'] = color
        plot_opts['lstyle_ac'] = '-'
        plot_opts['lstyle_ae'] = '--'
        color_inset = color
    else:
        color_inset = 'k'

    all_ws_glm_ac = []
    all_ws_glm_ae = []
    all_resets = []
    for f in files:
        if f.find('PreProcess') == -1:
            # print('-----------------')
            # print(f)
```

```
        if f.find('.mat') != -1:
            folder = f[:-4]+'/'
        else:
            folder = f+'/'
        # TODO: get function from process_rats_data.py
        hf.exp_results_process(f)
        if not os.path.exists(folder+'/weights_150222.npz'):
            ws_ac, ws_ae = standard_glm(folder=folder)
            np.savez(folder+'/weights_150222.npz',
                     **{'ws_ac': ws_ac, 'ws_ae': ws_ae})
        weights = np.load(folder+'/weights_150222.npz')
        ws_ac = weights['ws_ac']
        ws_ae = weights['ws_ae']
        # TODO: get function from plotting_functions_reset_paper.py and
        # copy it in helper_function.py
        reset, krnl_ac, krnl_ae =\
            hf.compute_reset_index(ws_ac, ws_ae, xtcks=xtcks,
                                   full_reset_index=False)
        contr_ac = np.abs(np.mean(krnl_ac))
        contr_ae = np.abs(np.mean(krnl_ae))
        if contr_ac+contr_ae > contr_th:
            all_ws_glm_ac.append(ws_ac)
            all_ws_glm_ae.append(ws_ae)
            all_resets.append(reset)
            if plt_ind_trcs:
                hf.plot_kernels(ws_ac, ws_ae, ax=axs_glm_krnls,
                                regrss=tags_mat, **plot_opts)
            if plt_ind_indx:
                ax_inset.scatter(x+np.random.randn()*0.02, reset,
                                 edgecolor='none', color=color_inset,
                                 alpha=0.2)
            if ax_2d_plot is not None:
                ax_2d_plot.plot(contr_ae, contr_ac, color=color_inset,
                                marker='+')

mean_ws_ac = np.mean(np.array(all_ws_glm_ac), axis=0)
mean_ws_ae = np.mean(np.array(all_ws_glm_ae), axis=0)
std_ws_ac = np.std(np.array(all_ws_glm_ac), axis=0)
std_ws_ae = np.std(np.array(all_ws_glm_ae), axis=0)

plot_opts['alpha'] = 1
plot_opts['label'] = name
plot_opts['lw'] = 1.5
hf.plot_kernels(mean_ws_ac, mean_ws_ae, std_ac=std_ws_ac, std_ae=std_ws_ae,
            ax=axs_glm_krnls, regrss=tags_mat, **plot_opts)
box_plot(data=all_resets, ax=ax_inset, x=x)
print('Mean Reset Index:')
```

```
print(np.mean(all_resets))
print('STD Reset Index:')
print(np.std(all_resets))
print('N = ', len(all_resets))

if ax_wsls is not None:
    f, ax = plt.subplots()
    # print(np.array(all_ws_glm_ac).shape)
    ac_mat = []
    ae_mat = []
    for awgac, awgae in zip(all_ws_glm_ac, all_ws_glm_ae):
        k_Lp, _ = hf.get_krnl(name='L+', cols=hf.afterc_cols, weights=awgac,
                              n_stps_ws=1)
        k_Lm, _ = hf.get_krnl(name='L-', cols=hf.aftere_cols, weights=awgae,
                              n_stps_ws=1)
        ax.plot(k_Lp, color=hf.naranja)
        ax.plot(k_Lm, color='k')
        ac_mat.append(k_Lp[0])
        ae_mat.append(k_Lm[0])
    ac_mean = np.median(ac_mat)
    ac_mat = np.array(ac_mat)/ac_mean
    ae_mat = np.array(ae_mat)/ac_mean
    ax_wsls.scatter(0.01*np.random.rand(len(ac_mat)), ac_mat, color=hf.naranja)
    ax_wsls.scatter(0.01*np.random.rand(len(ac_mat)), ae_mat, color='k')
    box_plot(data=ac_mat, ax=ax_wsls, x=0, lw=.5, fliersize=4,
             color=hf.naranja)
    box_plot(data=ae_mat, ax=ax_wsls, x=0, lw=.5, fliersize=4, color='k')
    # TODO: change folder
    SV_FOLDER =\
        '/home/molano/Dropbox/project_Barna/reset_paper/figures/' +\
        'figs_from_python/'
    hf.sv_fig(f=f, name='wsls_krnls', sv_folder=SV_FOLDER)

    # ax_wsls.scatter(all_ws_glm_ac)
if sv_figs:
    for ax in ax_tr:
        ax.spines['right'].set_visible(False)
        ax.spines['top'].set_visible(False)
    ax_inset.spines['right'].set_visible(False)
    ax_inset.spines['top'].set_visible(False)
    ax_inset.set_xticks([])
    ax_inset.set_yticks([0, 1])
    ax_inset.set_xlim([x-0.5, x+0.5])
    ax_inset.set_ylabel('Reset Index')
    f_glm.savefig(sv_folder+'/GLM_kernels_transition.png', dpi=400,
                  bbox_inches='tight')
    reset_data = {'rest_indexes': all_resets}
```

```
        np.savez(sv_folder+'/all_resets.npz', **reset_data)


def get_data_traj(subj, main_folder):
    # subject folder
    # folder = main_folder+subj+'/'   # Manuel
    folder = main_folder+'\\'+subj+'\\'  # Alex
    # find all data files
    files_trials = glob.glob(folder+'*trials.csv')
    files_traj = glob.glob(folder+'*trials-trajectories.csv')
    # take files names
    file_list_trials = [os.path.basename(x) for x in files_trials
                        if x.endswith('trials.csv')]
    file_list_traj = [os.path.basename(x) for x in files_traj
                      if x.endswith('trials-trajectories.csv')]
    # sort files
    sfx_tls = [x[x.find('2021'):x.find('2021')+15] for x in file_list_trials]
    sfx_trj = [x[x.find('2021'):x.find('2021')+15] for x in file_list_traj]

    sorted_list_tls = [x for _, x in sorted(zip(sfx_tls, file_list_trials))]
    sorted_list_trj = [x for _, x in sorted(zip(sfx_trj, file_list_traj))]
    # create data
    data_tls = {'correct': np.empty((0,)), 'answer_response': np.empty((0,)),
                'soundPlay_object1_leftRightBalance': np.empty((0,)),
                'respondedInTime': np.empty((0,)), 'block': np.empty((0,)),
                'soundPlay_responseTime': np.empty((0,)),
                'soundPlay_duration': np.empty((0,)),
                'answer_responseTime': np.empty((0,))}
    # go over all files
    for f in sorted_list_tls:
        # read file
        # df1 = pd.read_csv(main_folder+subj+'/'+f, sep=',')  # Manuel
        df1 = pd.read_csv(main_folder+'\\'+subj+'\\'+f, sep=',')  # Alex
        for k in data_tls.keys():
            values = df1[k].values
            if k == 'soundPlay_object1_leftRightBalance':
                values = values-.5
                values[np.abs(values) < 0.01] = 0
            data_tls[k] = np.concatenate((data_tls[k], values))
    data_trj = {'answer_positionsX': np.empty((0,)),
                'answer_positionsY': np.empty((0,)),
                'answer_times': np.empty((0,))}
    for f in sorted_list_trj:
        # read file
        # df1 = pd.read_csv(main_folder+subj+'/'+f, sep=',')  # Manuel
        df1 = pd.read_csv(main_folder+'\\'+subj+'\\'+f, sep=',')  # Alex
        for k in data_trj.keys():
```

```
            values = df1[k].values
            if k == 'soundPlay_object1_leftRightBalance':
                values = values-.5
                values[np.abs(values) < 0.01] = 0
            data_trj[k] = np.concatenate((data_trj[k], values))
    return data_tls, data_trj
```

13.3.2 Helper functions: helper_functions.py

```
import numpy as np
from numpy import logical_and as and_
# from copy import deepcopy as deepc
import seaborn as sns
from numpy import concatenate as conc
import matplotlib.pyplot as plt
from scipy.special import erf
from scipy.optimize import curve_fit
from scipy.stats import mstats
# import scipy.io as sio
# import seaborn as sns
from sklearn.linear_model import LogisticRegression
import pandas as pd
import os
# import re
# import glob
import sys
from scipy.io import loadmat
from matplotlib.ticker import MultipleLocator
from scipy.stats import shapiro, ttest_ind, wilcoxon, ttest_1samp
from sklearn.model_selection import cross_val_score
import statsmodels.api as sm
# from scipy.stats import norminvgauss as norminv
# from scipy.stats import norm
from scipy.interpolate import interp1d
# from scipy.signal import savgol_filter, medfilt
# from scipy.optimize import curve_fit
from scipy.signal import find_peaks
from scipy.stats import ks_2samp, kstest, chisquare  # Kolmogorov-Smirnov test
sys.path.append(os.path.expanduser("~/neurogym"))
rojo = np.array((228, 26, 28))/255
azul = np.array((55, 126, 184))/255
verde = np.array((77, 175, 74))/255
morado = np.array((152, 78, 163))/255
naranja = np.array((255, 127, 0))/255
marron = np.array((166, 86, 40))/255
amarillo = np.array((155, 155, 51))/255
rosa = np.array((252, 187, 161))/255
```

```
cyan = np.array((0, 1, 1))
gris = np.array((.5, .5, 0.5))
azul_2 = np.array([56, 108, 176])/255
rojo_2 = np.array([165, 15, 21])/255
XTICKS = np.array(['1', '2', '3', '4', '5', '6-10'])

COLORES = conc((azul.reshape((1, 3)), rojo.reshape((1, 3)),
                verde.reshape((1, 3)), morado.reshape((1, 3)),
                naranja.reshape((1, 3)), marron.reshape((1, 3)),
                amarillo.reshape((1, 3)), rosa.reshape((1, 3))),
               axis=0)
COLORES = np.concatenate((COLORES, 0.5*COLORES))
model_cols = ['evidence',
              'L+', 'L-', 'T+-', 'T-+', 'T--', 'T++', 'intercept']
afterc_cols = [x for x in model_cols if x not in ['L+2', 'L-1', 'L-2']]
aftere_cols = [x for x in model_cols if x not in ['L+1', 'L+2', 'L-2']]
# tau_list = [0.5, 1, 2, 3]
tau_list = [1]
START_ANALYSIS = 200
# --- PLOTTING FUNCTIONS


def check_dictionary(data, start=200, num_tr=200):
    """
    Plots responses and hit, together with the regressors and the fitted
    regressors.
    """
    nrows = 8
    f, ax = plt.subplots(nrows=nrows, ncols=1, figsize=(8, 8), sharex=True)
    ax = ax.flatten()
    ax[0].plot(data['R_response'][start:start+num_tr], '-+',
               label='Choice')
    ax[0].plot(data['hit'][start:start+num_tr], '--+',
               label='hit')
    ax[0].legend()
    ax[1].plot(data['T++1'][start:start+num_tr], label='T++1')
    ax[1].set_ylabel('T++1')
    ax[1].plot(data['T++'][start:start+num_tr], label='T++')
    ax[1].legend()
    ax[2].plot(data['rep_response_11'][start:start+num_tr])
    ax[2].set_ylabel('rep_response_11')
    ax[2].legend()
    ax[3].plot(data['T--1'][start:start+num_tr], label='T--1')
    ax[3].set_ylabel('T--1')
    ax[3].plot(data['T--'][start:start+num_tr], label='T--')
    ax[3].legend()
    ax[4].plot(data['T+-1'][start:start+num_tr], label='T+-1')
```

```
    ax[4].set_ylabel('T+-1')
    ax[4].plot(data['T+-'][start:start+num_tr], label='T+-')
    ax[4].legend()
    ax[5].plot(data['T-+1'][start:start+num_tr], label='T-+1')
    ax[5].set_ylabel('T-+1')
    ax[5].plot(data['T-+'][start:start+num_tr], label='T-+')
    ax[5].legend()
    ax[6].plot(data['L+1'][start:start+num_tr], label='L+1')
    ax[6].set_ylabel('L+1')
    ax[6].plot(data['L+'][start:start+num_tr], label='L+')
    ax[6].legend()
    ax[7].plot(data['L-1'][start:start+num_tr], label='L-1')
    ax[7].set_ylabel('L-1')
    ax[7].plot(data['L-'][start:start+num_tr], label='L-')
    ax[7].legend()
    for i in range(nrows):
        ax[i].grid(axis='x', which='major')
        ax[i].xaxis.set_major_locator(MultipleLocator(5))


def compute_reset_index(weights_ac, weights_ae, full_reset_index=False,
                        xtcks=None):
    """
    Computes reset index as: r_index = 1-abs(w_T++_ac)/abs(w_T++_ae)
    """
    xtcks = 'T++'
    indx = np.array(np.where(np.array(afterc_cols) == xtcks))
    indx = np.array([x for x in indx if len(x) > 0])
    w_ac = abs(weights_ac[0, 0, indx][0])
    w_ae = abs(weights_ae[0, 0, indx][0])
    reset = 1-(w_ae+1e-6)/(w_ac+1e-6)
    return reset[0]


def significant_difference(w_ae):
    """
    First it computes whether the distribution is normal and then performs a
    ttest (if normal) to see if the weights after error are different from zero.
    """
    normal_ae = shapiro(w_ae)[1] > 0.05
    if normal_ae:
        p_value = ttest_1samp(w_ae, 0)[1]
    else:
        p_value = wilcoxon(w_ae)[1]
    return p_value
```

```
def plot_ac_ae(weights_ac, weights_ae):
    """
    Plots GLM weights of AE vs weights of AC of the desired regressor
    """
    fig_w, ax_w = plt.subplots(1)
    ax_w.scatter(weights_ae, weights_ac)
    ax_w.set_xlabel('GLM Weights AE')
    ax_w.set_ylabel('GLM Weights AC')


def kernel_dicts(weights_ac, weights_ae, d):
    """
    This function is used to update the dictionary where the weights of all
    subjects will be saved, differentiating between after correct and after error.
    """
    weights_ac = weights_ac[0][0]
    weights_ae = weights_ae[0][0]
    cols = model_cols
    for j in range(1, len(weights_ac)-1):
        w_ac = weights_ac[j]
        w_ae = weights_ae[j]
        if len(d.keys()) < 12:
            d[cols[j]+'_ac'] = [w_ac]
            d[cols[j]+'_ae'] = [w_ae]
        else:
            d[cols[j]+'_ac'].append(w_ac)
            d[cols[j]+'_ae'].append(w_ae)
    return d


def boxplot_kernels(d_kernels, name):
    """
    This function plots a boxplot of the GLM weights. Separating between
    after correct and after error.
    """
    fb, axb = plt.subplots(ncols=4, nrows=2, figsize=(8, 5),
                           sharey=True)
    axb = axb.flatten()
    fb.suptitle('weights' + name + 'ms')
    cols = [x for x in model_cols if x != 'intercept' and x != 'evidence']
    for j in range(len(cols)):
        axb[j].boxplot([d_kernels[cols[j]+'_ac'],
                        d_kernels[cols[j]+'_ae']])
        axb[j].scatter(np.repeat(1, len(d_kernels[cols[j]+'_ac'])),
                       d_kernels[cols[j]+'_ac'],
                       alpha=0.5, marker='x', c=gris)
        axb[j].scatter(np.repeat(2, len(d_kernels[cols[j]+'_ae'])),
```

```
                d_kernels[cols[j]+'_ae'],
                alpha=0.5, marker='x', c=gris)
        axb[j].set_xticks([1, 2])
        axb[j].set_xticklabels([cols[j]+'_ac', cols[j]+'_ae'])
        axb[j].set_ylabel('GLM weights')


def boxplot_reset_ind(reset_list, ax):
    """
    This function plots the boxplot of the reset index of all subjects.
    """
    ax.boxplot(reset_list)
    ax.scatter(np.repeat(1, len(reset_list)), reset_list,
            alpha=0.5, marker='x', c=gris)
    ax.set_ylabel('Reset index')
    ax.set_xticks([1])
    ax.set_xticklabels([''])


def boxplot_ae_acc(reset_st, reverse_st):
    revae = [a for i, a in enumerate(reverse_st.values()) if i % 2 == 0]
    resae = [a for i, a in enumerate(reset_st.values()) if i % 2 == 0]
    acc_res = [a for i, a in enumerate(reset_st.values()) if i % 2 == 1]
    acc_rev = [a for i, a in enumerate(reverse_st.values()) if i % 2 == 1]
    fig, ax = plt.subplots(1)
    data = [revae, resae, acc_rev, acc_res]
    ax.boxplot(data)
    ax.set_xticks([1, 2, 3, 4])
    ax.set_xticklabels(['Rev_ae', 'Res_ae', 'Acc_rev', 'Acc_res'])
    ax.tick_params(axis='x', labelrotation=45)
    ax.set_ylabel('Accuracy')
    for i, y in enumerate(data):
        ax.scatter(np.repeat(i+1, len(y)) + np.random.randn(len(y)) / 10, y,
                marker='x', alpha=0.4, color='k')


def sv_fig(f, name, sv_folder):
    """
    Save figure.

    Parameters
    ----------
    f : fig
        figure to save.
    name : str
        name to use to save the figure.
```

```
    Returns
    -------
    None.

    """
    f.savefig(sv_folder+'/'+name+'.svg', dpi=400, bbox_inches='tight')
    f.savefig(sv_folder+'/'+name+'.pdf', dpi=400, bbox_inches='tight')
    f.savefig(sv_folder+'/'+name+'.png', dpi=400, bbox_inches='tight')


def plot_psycho_curve(ev, choice, popt, ax, color_scatter, plot_errbars=False,
                      **plt_opts):
    """
    Plot psycho-curves (fits and props) using directly the fit parameters.

    THIS FUNCTION ASSUMES PUTATIVE EVIDENCE (it will compute response proportions
                                        for all values of ev)

    Parameters
    ----------
    ev : array
        array with **putative** evidence for each trial.
    choice : array
        array with choices made by agent.
    popt : list
        list containing fitted parameters (beta, alpha, piL, piR).
    ax : axis
        where to plot.
    **plt_opts : dict
        plotting options.

    Returns
    -------
    means : list
        response means for each evidence value.
    sems : list
        sem for the responses.
    x : array
        evidences values for which the means/sems are computed.
    y_fit : array
        y values for the fit.
    x_fit : array
        x values for the fit.

    """
    x_fit = np.linspace(np.min(ev), np.max(ev), 20)
    y_fit = probit_lapse_rates(x_fit, popt[0], popt[1], popt[2], popt[3])
```

```
    ax.plot(x_fit, y_fit, markersize=6, **plt_opts)
    means = []
    sems = []
    n_samples = []
    for e in np.unique(ev):
        means.append(np.mean(choice[ev == e]))
        sems.append(np.std(choice[ev == e])/np.sqrt(np.sum(ev == e)))
        n_samples.append(np.sum(ev == e))
    x = np.unique(ev)
    plt_opts['linestyle'] = ''
    if 'label' in plt_opts.keys():
        del plt_opts['label']
    if plot_errbars:
        ax.errorbar(x, means, sems, **plt_opts)
    ax.scatter(x, means, marker='.', alpha=1, s=60, c=color_scatter)
    ax.plot([0, 0], [0, 1], '--', lw=0.2, color=(.5, .5, .5))
    d_list = [means, sems, x, y_fit, x_fit, n_samples]
    d_str = ['means, sems, xs, y_fit, x_fit, n_samples']
    d = list_to_dict(d_list, d_str)
    return d


def xtcks_krnls(xs, ax):
    xtcks = np.arange(1, max(xs)+1)
    ax.set_xticks(xtcks)
    ax.set_xlim([xtcks[0]-0.5, xtcks[-1]+0.5])
    xtcks_lbls = [str(x) for x in xtcks]
    xtcks_lbls[-1] = '6-10'
    ax.set_xticklabels(xtcks_lbls)


def get_opts_krnls(plot_opts, tag):
    opts = {k: x for k, x in plot_opts.items() if k.find('_a') == -1}
    opts['color'] = plot_opts['color'+tag]
    opts['linestyle'] = plot_opts['lstyle'+tag]
    return opts


def plot_reset_index(reset, ax, step=None, per=None, xs=0.5, **plot_opts):
    if step is None:
        if 'marker' not in plot_opts.keys():
            plot_opts['marker'] = '+'
        if 'color' not in plot_opts.keys():
            plot_opts['color'] = 'b'
        xs = [xs]
    else:
        plot_opts['marker'] = ''
```

```
        xs = np.arange(reset.shape[0])*step+per/2

    ax.plot(xs, reset, **plot_opts)


def plot_kernels(weights_ac, weights_ae, std_ac=None, std_ae=None, ac_cols=None,
                 ae_cols=None, ax=None, n_stps_ws=20, ax_inset=None, inset_xs=0.5,
                 regressors=['T++', 'T-+', 'T+-', 'T--'], **kwargs):
    plot_opts = {'lw': 1,  'label': '', 'alpha': 1, 'color_ac': naranja,
                 'fntsz': 7, 'color_ae': (0, 0, 0), 'lstyle_ac': '-',
                 'lstyle_ae': '-', 'marker': '.'}
    plot_opts.update(kwargs)
    fntsz = plot_opts['fntsz']
    del plot_opts['fntsz']
    ac_cols = afterc_cols if ac_cols is None else ac_cols
    ae_cols = aftere_cols if ae_cols is None else ae_cols
    if ax is None:
        n_regr = len(regressors)
        if n_regr > 2:
            ncols = int(np.sqrt(n_regr))
            nrows = int(np.sqrt(n_regr))
            figsize = (8, 5)
        else:
            ncols = n_regr
            nrows = 1
            figsize = (8, 3)
        f, ax = plt.subplots(ncols=ncols, nrows=nrows, figsize=figsize,
                             sharey=True)

        ax = ax.flatten() if n_regr > 1 else [ax]
        ax_inset = plt.axes((0.79, 0.15, 0.1, 0.15))
        for a in ax:
            a.invert_xaxis()
            a.axhline(y=0, linestyle='--', c='k', lw=0.5)
    else:
        f = None
    kernel_ac_list = []
    kernel_ae_list = []
    for j, name in enumerate(regressors):
        ax[j].set_ylabel('Weight (a.u.)', fontsize=fntsz)
        ax[j].set_xlabel('Trials back from decision', fontsize=fntsz)
        # after correct
        kernel_ac, xs_ac = get_krnl(name=name, cols=ac_cols, weights=weights_ac,
                                    n_stps_ws=n_stps_ws)
        kernel_ac_list.append(kernel_ac)
        ax[j].set_title(name)
        if std_ac is not None:
```

```
            s_ac, _ = get_krnl(name=name, cols=ac_cols, weights=std_ac,
                               n_stps_ws=n_stps_ws)
        else:
            s_ac = np.zeros_like(kernel_ac)
        opts = get_opts_krnls(plot_opts=plot_opts, tag='_ac')
        ax[j].errorbar(xs_ac, kernel_ac, s_ac, **opts)

        # after error
        kernel_ae, xs_ae = get_krnl(name=name, cols=ae_cols, weights=weights_ae,
                                    n_stps_ws=n_stps_ws)
        kernel_ae_list.append(kernel_ae)
        if std_ae is not None:
            s_ae, _ = get_krnl(name=name, cols=ae_cols, weights=std_ae,
                               n_stps_ws=n_stps_ws)
        else:
            s_ae = np.zeros_like(kernel_ae)
        opts = get_opts_krnls(plot_opts=plot_opts, tag='_ae')
        ax[j].errorbar(xs_ae, kernel_ae, s_ae, **opts)

        # tune fig
        xtcks_krnls(xs=xs_ac, ax=ax[j])
    # PLOT RESET INDEX
    if ax_inset is not None:
        # compute reset
        ws_ac = np.nanmean(weights_ac[-n_stps_ws:, :, :], axis=0)
        ws_ac = np.expand_dims(ws_ac, 0)
        ws_ae = np.nanmean(weights_ae[-n_stps_ws:, :, :], axis=0)
        ws_ae = np.expand_dims(ws_ae, 0)
        xtcks = ['T++'+x for x in ['2', '3', '4', '5', '6-10']]
        reset, _, _ = compute_reset_index(ws_ac, ws_ae, xtcks=xtcks,
                                          full_reset_index=False)
        opts = {k: x for k, x in plot_opts.items() if k.find('_a') == -1}
        plot_reset_index(reset=reset, ax=ax_inset, xs=float(inset_xs), **opts)

    return f, kernel_ac_list, kernel_ae_list, xs_ac, xs_ae


def plot_expo_kernel(decays_ac, decays_ae, **kwargs):
    plot_opts = {'lw': 3,  'label': '', 'alpha': 0.1, 'color_ac': naranja,
                 'fntsz': 10, 'color_ae': (0, 0, 0), 'lstyle_ac': '-',
                 'lstyle_ae': '-', 'marker': ''}
    # plot_opts.update(kwargs)
    fntsz = plot_opts['fntsz']
    del plot_opts['fntsz']
    f, ax = plt.subplots(1, 1, figsize=(5, 5))
    ax.set_ylabel('T++ weight', fontsize=fntsz)
    ax.set_xlabel('Trial lag', fontsize=fntsz)
```

```
    # After correct
    opts = get_opts_krnls(plot_opts=plot_opts, tag='_ac')
    for decay in decays_ac:
        ax.plot(decay, **opts)
    decays_ac = np.array(decays_ac)
    mean_decay_ac = np.mean(decays_ac, axis=0)
    s_ac = np.std(decays_ac, axis=0)
    ax.errorbar(x=np.arange(6), y=mean_decay_ac, yerr=s_ac, linewidth=3,
                color=naranja, marker='.', alpha=1)
    # After error
    opts = get_opts_krnls(plot_opts=plot_opts, tag='_ae')
    for decay in decays_ae:
        ax.plot(decay, **opts)
    decays_ae = np.array(decays_ae)
    mean_decay_ae = np.mean(decays_ae, axis=0)
    s_ae = np.std(decays_ae, axis=0)
    ax.errorbar(x=np.arange(5), y=mean_decay_ae, yerr=s_ae, linewidth=3,
                color=(0, 0, 0), marker='.', alpha=1)
    ax.set_xticks([0, 1, 2, 3, 4, 5])
    ax.set_xticklabels(['6-10', '5', '4', '3', '2', '1'])
    ax.set_xlim([-0.5, 5.5])
    # ax.set_yticks([-1, 0, 1])
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    # ax.set_ylim([-1, 1])
    ytckslbls = ['-1', '0', '1']
    ax.axhline(y=0, linestyle='--', color='k', lw=0.5)
    ylims = ax.get_ylim()
    ylims = [-np.max(np.abs(ylims)), np.max(np.abs(ylims))]
    ax.set_ylim(ylims)
    # ax.set_ylabel('')
    yticks = [ylims[0], 0, ylims[1]]
    ax.set_yticks(yticks)
    ax.set_yticklabels(ytckslbls)


def call_plot_kernels(ws_ac, ws_ae, ax, std_ac=None, std_ae=None,
                      regrss=[['T++', 'T-+', 'T+-', 'T--'], ['L+', 'L-']],
                      **plot_opts):
    # plot kernels
    kernel_d = {}
    for a, r in zip(ax, regrss):
        f, kernel_ac, kernel_ae, xs_ac, xs_ae = \
            plot_kernels(ws_ac, ws_ae, std_ac=std_ac, std_ae=std_ae,
                         ax=a, n_stps_ws=2, regressors=r, **plot_opts)
        for re in range(len(r)):
            kernel_d[r[re]] = [kernel_ac[re], kernel_ae[re]]
```

```
    return kernel_d  # dictionary with each kernel (ac, ae) associated to a reg


# --- SECONDARY FUNCTIONS
def get_krnl(name, cols, weights, n_stps_ws):
    indx = np.array([np.where(np.array([x.startswith(name)
                                        for x in cols]))[0]])
    indx = np.array([x for x in indx if len(x) > 0])
    xtcks = np.array(cols)[indx][0]
    if len(xtcks) < 2:
        xs = [1]
    else:
        xs = [int(x[len(name):len(name)+1]) for x in xtcks]
    kernel = np.nanmean(weights[-n_stps_ws:, 0, indx], axis=0).flatten()
    return kernel, xs


def list_to_dict(lst, string):
    """
    Transform a list of variables into a dictionary.

    Parameters
    ----------
    lst : list
        list with all variables.
    string : str
        string containing the names, separated by commas.

    Returns
    -------
    d : dict
        dictionary with items in which the keys and the values are specified
        in string and lst values respectively.

    """
    string = string[0]
    string = string.replace(']', '')
    string = string.replace('[', '')
    string = string.replace('\\', '')
    string = string.replace(' ', '')
    string = string.replace('\t', '')
    string = string.replace('\n', '')
    string = string.split(',')
    d = {s: v for s, v in zip(string, lst)}
    return d
```

```
def get_tag(tag, file):
    """
    Return value associated to a given tag in a file.

    Parameters
    ----------
    tag : str
        tag to look for in the basename of file.
    file : str
        file with tag and value separated by _.

    Returns
    -------
    val : str
        value associated to tag.

    """
    # process name
    file_name = os.path.basename(file)
    start_val = file_name.find(tag)
    assert start_val != -1, 'Tag ' + tag + ' not found in ' + file_name
    val = file_name[start_val + len(tag) + 1:]
    val = val[:val.find('_')] if '_' in val else val
    return val


def get_repetitions(mat):
    """
    Return mask indicating the repetitions in mat.

    Makes diff of the input vector, mat, to obtain the repetition vector X,
    i.e. X will be 1 at t if the value of mat at t is equal to that at t-1
    Parameters
    ----------
    mat : array
        array of elements.

    Returns
    -------
    repeats : array
        mask indicating the repetitions in mat.

    """
    mat = mat.flatten()
    values = np.unique(mat)
    # We need to account for size reduction of np.diff()
    rand_ch = np.array(np.random.choice(values, size=(1,)))
```

```
    repeat_choice = conc((rand_ch, mat))
    diff = np.diff(repeat_choice)
    repeats = (diff == 0)*1.
    repeats[np.isnan(diff)] = np.nan
    return repeats


def get_transition_mat(choice, conv_w=5):
    """
    Return array indicating the number of repetitions in the last conv_w trials.

    convolves the repetition vector to get a count of the number of repetitions
    in the last conv_w trials

    Parameters
    ----------
    choice : array
        it is expected to be a repetition vector obtained from get_repetitions fn.
    conv_w : str, optional
        window to consider past trials (5)

    Returns
    -------
    np.array
        array is equal to conv_w/2 when there have been conv_w repetitions
        and -conv_w/2 when there have been 0 repetitions.

    """
    # selectivity to transition probability
    limit = -conv_w+1 if conv_w > 1 else len(choice)
    repeat = get_repetitions(choice)
    transition = np.convolve(repeat, np.ones((conv_w,)),
                             mode='full')[0:limit]
    transition_ev = np.concatenate((np.array([np.nan]), transition[:-1]))
    transition_ev -= conv_w/2
    return transition_ev


def probit(x, beta, alpha):
    """
    Return probit function with parameters alpha and beta.

    Parameters
    ----------
    x : float
        independent variable.
    beta : float
```

```
        sensitiviy.
    alpha : TYPE
        bias term.

    Returns
    -------
    probit : float
        probit value for the given x, beta and alpha.

    """
    probit = 1/2*(1+erf((beta*x+alpha)/np.sqrt(2)))
    return probit


def probit_lapse_rates(x, beta, alpha, piL, piR):
    """
    Return probit with lapse rates.

    Parameters
    ----------
    x : float
        independent variable.
    beta : float
        sensitiviy.
    alpha : TYPE
        bias term.
    piL : float
        lapse rate for left side.
    piR : TYPE
        lapse rate for right side.

    Returns
    -------
    probit : float
        probit value for the given x, beta and alpha and lapse rates.

    """
    piL = 0
    piR = 0
    probit_lr = piR + (1 - piL - piR) * probit(x, beta, alpha)
    return probit_lr


def remove_borders(mask):
    """
    Remove the change steps (borders) of a mask.
```

```
    Refines mask by removing blocks' borders, which are detected by
    a change of 1 or -1 (from True to False or viceversa).

    Parameters
    ----------
    mask : array
        array with 0s and 1s indicating a certain period (that should be of
                                                    several steps).

    Returns
    -------
    mask : array
        same array the 1s in the border of the period made 0.

    """
    mask = 1*mask
    if np.sum(mask) < len(mask):
        inside_blk_indx_on = np.diff(mask) != 1
        inside_blk_indx_on = np.append(False, inside_blk_indx_on)
        inside_blk_indx_off = np.diff(mask) != -1
        inside_blk_indx_off = np.append(inside_blk_indx_off, False)
        mask = and_.reduce((inside_blk_indx_on, inside_blk_indx_off, mask))
    return mask


def template_match(mat, templ, plot=False):
    """
    Find the time points in an array where a given template occurs.

    Parameters
    ----------
    mat : array
        array in which to find the template (the array is expected to have at least
                                    one occurrence of the template).
    templ : array
        template.

    Returns
    -------
    mask : array
        array with 1s at the time points right after the template has occurred.

    """
    mat = mat - np.mean(mat)
    temp_match = np.convolve(mat, np.flip(templ), mode='same')
    times = (np.where(temp_match == np.max(temp_match))[0] +
            np.ceil(len(templ)/2)-1).astype('int')
```

```
    mask = np.zeros_like(mat)
    times = times[times < mask.shape[0]]
    mask[times] = 1
    if plot:
        plt.figure()
        plt.plot(mat[max(0, times[0]-100):times[0]+10000], '-+', markersize=6)
        plt.plot(mask[max(0, times[0]-100):times[0]+10000])
    return mask


def get_average(mat):
    """
    Return average of arrays contained in mat.

    Averages across instances data contained in mat. If instances have different
    lenghts they are equalized by padding with nans.

    Parameters
    ----------
    mat : array/list
        array or list containing containing the arrays to average.

    Returns
    -------
    average_matrix : array
        average of arrays.

    """
    # Remove empty instances from data
    a_mat = [x for x in mat if len(x) > 0]
    max_ = np.max([len(x) for x in a_mat])
    a_mat_ =\
        [conc((x, np.nan*np.ones((((int(max_-len(x)),)+np.array(x).shape[1:])))))
         for x in a_mat]  # add nan to have same shape mats
    a_mat_ = np.array(a_mat_)
    average_matrix = np.nanmean(a_mat_, axis=0)
    return average_matrix


def get_std(mat):
    """
    Return std of arrays contained in mat.

    Averages across instances data contained in mat. If instances have different
    lenghts they are equalized by padding with nans.

    Parameters
```

```
    ----------
    mat : array/list
       array or list containing containing the arrays to average.

    Returns
    -------
    average_matrix : array
       std of arrays.

    """
    # Remove empty instances from data
    a_mat = [x for x in mat if len(x) > 0]
    max_ = np.max([len(x) for x in a_mat])
    a_mat_ =\
       [conc((x, np.nan*np.ones((((int(max_-len(x)),)+np.array(x).shape[1:])))))
        for x in a_mat]  # add nan to have same shape mats
    a_mat_ = np.array(a_mat_)
    average_matrix = np.nanstd(a_mat_, axis=0)
    return average_matrix


def get_median(mat):
    """
    Return median of arrays contained in mat.

    Averages across instances data contained in mat. If instances have different
    lenghts they are equalized by padding with nans.

    Parameters
    ----------
    mat : array/list
       array or list containing containing the arrays to average.

    Returns
    -------
    average_matrix : array
       median of arrays.

    """
    # Remove empty instances from data
    a_mat = [x for x in mat if len(x) > 0]
    max_ = np.max([len(x) for x in a_mat])
    a_mat_ =\
       [conc((x, np.nan*np.ones((((int(max_-len(x)),)+np.array(x).shape[1:])))))
        for x in a_mat]  # add nan to have same shape mats
    a_mat_ = np.array(a_mat_)
    average_matrix = np.nanmedian(a_mat_, axis=0)
```

```
    return average_matrix


# --- PRIMARY FUNCTIONS

def compute_transition_probs_mat(data_mat, choices, block_n_ch,
                                 block_tr_hist, num_blocks=3,
                                 extra_condition=None):
    """
    Compute transition probs mat.

    Parameters
    ----------
    data_mat : array
        array of choices from which the transition probs will be inferred.
    choices : array
        array of choices to consider.
    block_n_ch : array
        array with the number of choices 'active' for each trial.
    block_tr_hist : array
        array indicating the block at each trial.
    num_blocks : int, optional
        total number of blocks (3)
    extra_condition : array, optional
        mask that allows further filtering the trials (None)

    Returns
    -------
    trans_mat : array
        array with the transition probs between choices (rows: current choice,
                                                         columns: next choice)
    counts_mat : array
        aray with the number of each type of transition (same as trans_mat but
                                                         without normalizing).

    """
    if extra_condition is None:
        extra_condition = np.full(data_mat.shape, True, dtype=None)

    # get transition blocks
    blck_tr_hist_id = np.unique(block_tr_hist)
    blck_tr_hist_id = blck_tr_hist_id[:num_blocks]
    n_blcks_trh = blck_tr_hist_id.shape[0]
    # get number of choices blocks
    blck_n_ch_id = np.unique(block_n_ch)
    n_blcks_nch = blck_n_ch_id.shape[0]
```

```
    # get choices
    ch_bins = np.append(choices-0.5, choices[-1]+0.5)
    trans_mat = np.empty((n_blcks_trh, n_blcks_nch, choices.shape[0],
                          choices.shape[0]))
    trans_mat[:] = np.nan
    counts_mat = np.empty((n_blcks_trh, n_blcks_nch, choices.shape[0],
                           choices.shape[0]))
    counts_mat[:] = np.nan
    for ind_nch, bl_nch in enumerate(blck_n_ch_id):
        for ind_trh, bl_trh in enumerate(blck_tr_hist_id):
            for ind_ch, ch in enumerate(choices):
                # avoid blocks borders
                blk_nch_mask = block_n_ch == bl_nch
                blk_nch_mask = remove_borders(blk_nch_mask)
                blk_trh_mask = block_tr_hist == bl_trh
                blk_trh_mask = remove_borders(blk_trh_mask)
                condition = and_.reduce((data_mat == ch, blk_nch_mask,
                                         blk_trh_mask, extra_condition))
                indx = np.where(condition)[0]
                indx = indx[indx < len(data_mat)-1]
                next_ch = data_mat[indx + 1]
                counts = np.histogram(next_ch, ch_bins)[0]
                trans_mat[ind_trh, ind_nch, ind_ch, :] = counts/np.sum(counts)
                counts_mat[ind_trh, ind_nch, ind_ch, :] = counts.astype(int)
    return trans_mat, counts_mat


def get_probs(mat, val, mask, num_chs, prev_mat=None):
    """
    Compute counts of each choice at t+1 conditioned on ch at t being val.

    Parameters
    ----------
    mat : array
        array with choices.
    val : int
        choice to condition on.
    mask : array
        array of booleans to filter trials.
    num_chs : int
        total number of possible choices.
    prev_mat : array, optional
        if not None, the previous choice will be conditioned on prev_mat
        instead of mat (this is used for the case in which mat is the ground truth
                        but the conditioning is made on the actual choices (None)

    Returns
```

```
    -------
    counts : array
        array with the counts of each possible choice at time t+1 after the choice
        indicated by val has been selected at time t.

    """
    if prev_mat is None:
        inds_ch = mat[:-1] == val
    else:
        inds_ch = prev_mat[:-1] == val
    # t + 1 selected where choice at t == ch
    inds_ch = conc((np.array([0]), inds_ch))
    inds_ch = np.where(and_(inds_ch, mask))[0]
    counts, _ = np.histogram(mat[inds_ch], bins=np.arange(num_chs+1)+0.5)
    counts = counts[:num_chs].astype('float')
    counts[counts == 0.] = 1
    return counts


def plot_matrices(mat, ref_mat):
    """
    Plot a matrix and a reference matrix (gt) side by side.

    Parameters
    ----------
    mat : 2D array
        matrix.
    ref_mat : 2D array
        reference matrix.

    Returns
    -------
    None.

    """
    plt.figure()
    plt.subplot(1, 2, 1)
    plt.imshow(mat)
    plt.title('Network')
    plt.subplot(1, 2, 2)
    plt.imshow(ref_mat)


def bias_psychometric(choice, ev, mask=None, maxfev=10000):
    """
    Compute repeating bias by fitting probit function.
```

```
Parameters
----------
choice : array
    array of choices made bythe network.
ev : array
    array with (signed) stimulus evidence.
mask : array, optional
    array of booleans indicating the trials on which the bias
# should be computed (None)

Returns
-------
popt : array
    Optimal values for the parameters so that the sum of the squared
    residuals of probit(xdata) - ydata is minimized
pcov : 2d array
    The estimated covariance of popt. The diagonals provide the variance
    of the parameter estimate. To compute one standard deviation errors
    on the parameters use ``perr = np.sqrt(np.diag(pcov))``.

    How the `sigma` parameter affects the estimated covariance
    depends on `absolute_sigma` argument, as described above.

    If the Jacobian matrix at the solution doesn't have a full rank, then
    'lm' method returns a matrix filled with ``np.inf``, on the other hand
    'trf'  and 'dogbox' methods use Moore-Penrose pseudoinverse to compute
    the covariance matrix.

"""
choice = choice.astype(float)
choice[and_(choice != 1, choice != 2)] = np.nan
repeat = get_repetitions(choice).astype(float)
repeat[np.isnan(choice)] = np.nan
# choice_repeating is just the original right_choice mat
# but shifted one element to the left.
choice_repeating = conc(
    (np.array(np.random.choice([1, 2])).reshape(1, ),
     choice[:-1]))
# the rep. evidence is the original evidence with a negative sign
# if the repeating side is the left one
rep_ev = ev*(-1)**(choice_repeating == 2)
if mask is None:
    mask = ~np.isnan(repeat)
else:
    mask = and_(~np.isnan(repeat), mask)
rep_ev_mask = rep_ev[mask]  # xdata
repeat_mask = repeat[mask]  # ydata
```

```
    try:
        # Use non-linear least squares to fit probit to xdata, ydata
        popt, pcov = curve_fit(probit_lapse_rates, rep_ev_mask,
                               repeat_mask, maxfev=maxfev)
    except RuntimeError as err:
        print(err)
        popt = [np.nan, np.nan, np.nan, np.nan]
        pcov = 0
    return popt, pcov, rep_ev_mask, repeat_mask


def distance(len_my_list, idx1, idx2, sign=1):
    d1 = abs(idx1 - idx2)
    d2 = len_my_list - d1
    sign = sign if ((d1 < d2) and (idx1 < idx2)) or ((d1 > d2) and (idx1 > idx2))\
        else -sign
    return sign*min(d1, d2)


def get_trans_prob_mat(choice, mask, n_ch):
    """
    Compute entropy bias frm a choice array.

    The code can compute the absolute entropy (KLD from the uniform) of the choices
    probabilities conditioned on each choice or the KLD from the distribution of
    ground truth sides.

    Parameters
    ----------
    choice : array
        array containing the choice from which to compute the entropy.
    mask : array
        array of boolean values indicating which elements to use for the

    Returns
    -------
    trans_mat : array
        array containing the transition probabilities

    """
    # Associate invalid trials (network fixates) with incorrect choice.
    invalid = and_(choice == 0, choice > n_ch)
    num_invalids = np.sum(invalid)
    # assign random choices to the invalid trials
    aux = np.random.choice(n_ch, (num_invalids,)) + 1
    choice[invalid] = aux
    # one entropy value calculated for trial t + 1 after choice ch
```

```
    trans_mat = np.empty((n_ch, n_ch))
    trans_mat[:] = np.nan
    aligned_mat = np.empty((n_ch, n_ch))
    aligned_mat[:] = np.nan
    for ch in np.arange(1, n_ch+1):
        probs = get_probs(mat=choice, val=ch, mask=mask, num_chs=n_ch)
        trans_mat[:, ch-1] = probs
        aligned_mat[:, ch-1] = np.roll(probs, -int(ch-n_ch/2))

    trans_mat = trans_mat/np.sum(trans_mat, axis=0)
    aligned_mat = np.sum(aligned_mat, axis=1)
    aligned_mat = aligned_mat/np.sum(aligned_mat)
    return trans_mat, aligned_mat


def get_trans_prob_mats_after_seq(data, n_ch=6, num_samples=None, ch_gt='choice',
                                  seq_prps={'templ_perf': [1, 1, 1, -1, 1],
                                            'templ_trans': np.array([1, 1, 1,
                                                                     np.nan,
                                                                     np.nan]),
                                            'start': 0}):
    """
    Compute entropy bias for passed conditions.

    Computes KLD from a uniform distr. of the choices probabilities conditioned
    on each choice or the KLD from the distribution of ground truth sides.
    It does so for each block type in tr_block, after_error and after_correct
    (Shape = (2, num_blocks))
    Parameters
    ----------
    ch : array
        array with choices.
    ev : array
        array with stimulus evidence.
    perf : array
        array with choices outcomes.
    c_tr : array
        array indicating catch trials (correct choice but reward not given).
    tr_block : TYPE
        array with transitions block.
    gt : array, optional
        array with ground truth sides (None)
    nch_mask : array, optional
        array to further filter trials (None)
    plot_tr_mat : bool, optional
        whether to plot transition matrices (False)
    ev_perc : int, optional
```

```
    percetile of evidence used to filter trials bias calculation (5)
plt_mask : bool, optional (for debugging)
    boolean indicating whether to plot mask with choices, perf.. (False)

Returns
-------
biases : array
    array with bias values.

"""
if num_samples is None:
    num_samples = 0
ch = data[ch_gt][-num_samples:]
sig_ev = data['putative_ev'][-num_samples:]
perf = data['performance'][-num_samples:]
print('Number of samples: ', perf.shape[0])
# filter by number of choices
indx = data['nch'] == n_ch
ch = ch[indx].astype(float)
sig_ev = sig_ev[indx].astype(float)
perf = perf[indx].astype(float)
# make nan all choices larger than nch
indx = np.logical_or(ch > n_ch, ch == 0)
ch[indx] = np.nan
n_max_ch = np.nanmax(ch)
sign = 1 if n_ch != 2 else -1  # this is probably not necessary
trans = [distance(n_max_ch, ch[x], ch[x+1], sign=sign)
         for x in range(ch.shape[0]-1)]
trans = conc((trans, np.array([np.nan])))

# context
templ_perf = seq_prps['templ_perf']
templ_trans = seq_prps['templ_trans']
start = seq_prps['start']

# transition blocks
# select evidence below ev_perc quantile
ev_mask = np.abs(sig_ev) < 0.00001
# get rid of first value in evidence
# ev_mask = conc((ev_mask[1:], np.array([0])))
# After error and after correct bias for each block.
trans_mats = []
al_trans_mats = []
num_samples_mat = []
perf_mat = []
ind_ctx = 0
for ind_ctx in range(0, len(templ_perf)):
```

```
            templ_perf_temp = templ_perf[:1+ind_ctx]
            print(templ_perf_temp)
            perf_mask = template_match(perf, templ_perf_temp)
            perf_mask = conc((np.array([False]), perf_mask[:-1]))

            if ind_ctx >= start:
                templ_trans_temp = templ_trans[:1+ind_ctx-start]
                print(templ_trans_temp)
                t_len = len(templ_trans_temp)
                t_comp = templ_trans_temp[~np.isnan(templ_trans_temp)]
                indx = np.array([i+t_len for i in range(len(trans)-t_len-1)
                                 if (trans[i:i+len(t_comp)] == t_comp).all()])
                trans_mask = np.zeros_like(perf_mask)
                if len(indx) > 0:
                    trans_mask[indx+1] = 1
            else:
                trans_mask = np.ones_like(perf_mask)

            mask = and_.reduce((ev_mask, perf_mask, trans_mask))
            if False:
                plot_masks_cond(ch=ch, perf=perf, mask=mask, p_hist=perf_mask,
                                repeat=sig_ev, trans=trans_mask, num=100,
                                start=np.where(mask == 1)[0][0]-10)
                plt.title(templ_perf_temp+[1001]+templ_trans_temp.tolist())
            num_samples = np.sum(mask)
            print('Number of samples: ', num_samples)
            tr_mat, al_tr_mat = get_trans_prob_mat(mask=mask.copy(), choice=ch,
                                                   n_ch=n_ch)
            trans_mats.append(tr_mat)
            al_trans_mats.append(al_tr_mat)
            num_samples_mat.append(num_samples)
            perf_mat.append(np.mean(perf[mask]))
    return trans_mats, al_trans_mats, num_samples_mat, perf_mat


def plot_masks_cond(ch, perf, mask, c_tr=None, general_mask=None, repeat=None,
                    trans=None, p_hist=None, num=500, start=9200):
    """
    Plot mask with choices, performance and other variables.

    Parameters
    ----------
    ch : array
        array with choices.
    perf : array
        array with choices outcomes.
    c_tr : array
```

```
        array indicating catch trials (correct choice but reward not given).
    mask : array
        array indicating which trials will be used.
    repeat : array, optional
        array with repeatitions (1 if repeating, 0 if alternating) (None)
    trans : TYPE, optional
        array indicating the number of past repetitions (None)
    p_hist : TYPE, optional
        array indicating the number of past correct (None)

    Returns
    -------
    None.

    """
    plt.subplots(figsize=(8, 8))
    plt.plot(ch[start:start+num], '-+',
             label='choice', lw=1)
    plt.plot(perf[start:start+num]-3, '--+', label='perf',
             lw=1)
    plt.plot(mask[start:start+num]-3, '-+', label='mask',
             lw=1)
    if general_mask is not None:
        plt.plot(general_mask[start:start+num] - 4, '-+',
                 label='general mask', lw=1)
    if c_tr is not None:
        plt.plot(c_tr[start:start+num] - 4, '-+',
                 label='catch trial', lw=1)
    if repeat is not None:
        plt.plot(repeat[start:start+num], '-+',
                 label='repeat', lw=1)
        plt.plot(trans[start:start+num], '-+',
                 label='transitions', lw=1)
    if p_hist is not None:
        plt.plot(p_hist[start:start+num], '-+',
                 label='perf_hist', lw=1)
    for ind in range(num):
        plt.plot([ind, ind], [-3, 3], '--',
                 color=(.7, .7, .7))
    plt.legend()


def psych_curves_props(x_values, y_values, bins=None, plot=False, num_values=7,
                       **plt_opts):
    """
    Plot average values of y_values in x_values.
```

```
    Parameters
    ----------
    x_values : array
        x values.
    y_values : array
        y values.
    color : tuple, optional
        color of the trace ((0, 0, 0))

    Returns
    -------
    None.

    """
    # conf = 0.95
    if bins is None:
        bin_edges = mstats.mquantiles(x_values,
                                      (np.arange(num_values)+1)/num_values)
    else:
        bin_edges = bins
    xss_bin = np.searchsorted(bin_edges, x_values)
    xss_unq = np.unique(xss_bin)
    mean_perf =\
        np.array([np.nanmean(y_values[xss_bin == x]) for x in xss_unq])
    std_perf = [np.nanstd(y_values[xss_bin == x]) /
                np.sqrt(np.sum(xss_bin == x)) for x in xss_unq]
    mean_indx = [np.nanmean(x_values[xss_bin == x]) for x in xss_unq]
    hist_1 = np.histogram(x_values[y_values == 1], bins=bin_edges)[0]
    hist_all = np.histogram(x_values, bins=bin_edges)[0]
    # plot
    props = hist_1/hist_all
    if plot:
        # plt_opts['alpha'] = 0.2
        # plt.plot(bin_edges[:-1]+np.diff(bin_edges)/2, props, marker='.',
        #          **plt_opts)
        plt_opts['alpha'] = 1
        plt.errorbar(mean_indx, mean_perf, std_perf, marker='.', **plt_opts)
        # asdasd
    return props


def compute_ev_cum(times, ev):
    ev_cum = np.zeros(times.shape)
    prev_t = 0
    for ind_t, t in enumerate(times):
        ev_cum[ind_t] = np.sum(ev[prev_t:t])
        prev_t = t
```

```
    return ev_cum


def exp_results_process(file, step=5000, per=10000):  # , **load_data_opts):
    # load_data_opts = {'bef_aft': 'before', 'tag': 'p4_', 'disc': 'False'}
    if file.find('.mat') != -1:
        sv_folder = file[:-4]
        if not os.path.exists(sv_folder):
            os.makedirs(sv_folder)
        data = loadmat(file)
        times = np.zeros_like(data['stim'])
        times[data['stim'].shape[0]-1, :] = 1
        times = times.T.reshape((-1, 1))
        times = np.where(times == 1)[0]
        ev = data['stim'].T.reshape((-1,))
        ev = compute_ev_cum(times, ev)
        ev = ev[:, None]
        ev = np.concatenate((np.zeros_like(ev), ev, np.zeros_like(ev)), axis=1)
        gt = data['categ'].reshape((-1,))
        perf = data['resptype'].reshape((-1,))
        perf[perf == -3] = 0  # make invalid trials error trials
        ch = data['resp'].reshape((-1,))
        invalid_fraction = np.sum(np.isnan(ch))/ch.shape[0]
        assert invalid_fraction < 0.01, str(invalid_fraction)
        ch[np.isnan(ch)] = 0  # make invalid trials miss trials
    else:
        sv_folder = file
        df_master = pd.read_pickle(file+'/df_master')
        ch = df_master['R_response'].values + 1
        # get mask
        indx_inv = ~np.isnan(ch)
        ch = ch[indx_inv]
        gt = df_master['rewside'].values[indx_inv]+1
        perf = df_master['hit'].values[indx_inv]
        # stimulus
        ev = df_master['evidence_1stfr'].values[indx_inv] +\
            df_master['evidence_rest'].values[indx_inv]
        ev = ev[:, None]
        ev = np.concatenate((np.zeros_like(ev), ev, np.zeros_like(ev)), axis=1)
    data = {'choice': ch, 'stimulus': ev, 'performance': perf,
            'gt': gt.astype(int)}
    np.savez(sv_folder+'/bhvr_data_all.npz', **data)
    params = {'task_kwargs': {'n_ch': 2}}
    np.savez(sv_folder+'/params.npz', **params)
    print('Number of trials: ', ch.shape[0])
    print('Performance: ', np.mean(perf))
```

```
def nanconv(vec_1, vec_2):
    """
    This function returns a convolution result of two vectors without
    considering nans
    """
    mask = ~np.isnan(vec_1)
    return np.nansum(np.multiply(vec_2[mask], vec_1[mask]))


def get_GLM_regressors(data, tau, chck_corr=False):
    """
    Compute regressors.

    Parameters
    ----------
    data : dict
        dictionary containing behavioral data.
    chck_corr : bool, optional
        whether to check correlations (False)

    Returns
    -------
    df: dataframe
        dataframe containg evidence, lateral and transition regressors.

    """
    ev = data['signed_evidence'][START_ANALYSIS::]  # coherence/evidence with sign
    perf = data['performance'].astype(float)  # performance (0/1)
    ch = data['choice'][START_ANALYSIS::].astype(float)  # choice (1, 2)
    # discard (make nan) non-standard-2afc task periods
    if 'std_2afc' in data.keys():
        std_2afc = data['std_2afc'][START_ANALYSIS::]
    else:
        std_2afc = np.ones_like(ch)
    inv_choice = and_(ch != 1., ch != 2.)
    nan_indx = np.logical_or.reduce((std_2afc == 0, inv_choice))

    ev[nan_indx] = np.nan
    perf[nan_indx] = np.nan
    ch[nan_indx] = np.nan
    ch = -(ch-2)  # choices should belong to {0, 1}
    prev_perf = ~ (conc((np.array([True]), data['performance'][:-1])) == 1)
    prev_perf = prev_perf.astype('int')
    prevprev_perf = (conc((np.array([False]), prev_perf[:-1])) == 1)
    ev /= np.nanmax(ev)
    rep_ch_ = get_repetitions(ch)
```

```
# variables:
# 'origidx': trial index within session
# 'rewside': ground truth
# 'hithistory': performance
# 'R_response': choice (right == 1, left == 0, invalid == nan)
# 'subjid': subject
# 'sessid': session
# 'res_sound': stimulus (left - right) [frame_i, .., frame_i+n]
# 'sound_len': stim duration
# 'frames_listened'
# 'aftererror': not(performance) shifted
# 'rep_response'
df = {'origidx': np.arange(ch.shape[0]),
      'R_response': ch,
      'hit': perf,
      'evidence': ev,
      'aftererror': prev_perf,
      'rep_response': rep_ch_,
      'prevprev_perf': prevprev_perf}
df = pd.DataFrame(df)

# Lateral module
df['L+1'] = np.nan  # np.nan considering invalids as errors
df.loc[(df.R_response == 1) & (df.hit == 1), 'L+1'] = 1
df.loc[(df.R_response == 0) & (df.hit == 1), 'L+1'] = -1
df.loc[df.hit == 0, 'L+1'] = 0
df['L+1'] = df['L+1'].shift(1)
df.loc[df.origidx == 1, 'L+1'] = np.nan
# L-
df['L-1'] = np.nan
df.loc[(df.R_response == 1) & (df.hit == 0), 'L-1'] = 1
df.loc[(df.R_response == 0) & (df.hit == 0), 'L-1'] = -1
df.loc[df.hit == 1, 'L-1'] = 0
df['L-1'] = df['L-1'].shift(1)
df.loc[df.origidx == 1, 'L-1'] = np.nan

# pre transition module
df.loc[df.origidx == 1, 'rep_response'] = np.nan
df['rep_response_11'] = df.rep_response
df.loc[df.rep_response == 0, 'rep_response_11'] = -1
df.rep_response_11.fillna(value=0, inplace=True)
df.loc[df.origidx == 1, 'aftererror'] = np.nan

# transition module
df['T++1'] = np.nan  # np.nan
df.loc[(df.aftererror == 0) & (df.hit == 1), 'T++1'] =\
   df.loc[(df.aftererror == 0) & (df.hit == 1), 'rep_response_11']
```

```
df.loc[(df.aftererror == 1) | (df.hit == 0), 'T++1'] = 0
df['T++1'] = df['T++1'].shift(1)

df['T+-1'] = np.nan  # np.nan
df.loc[(df.aftererror == 0) & (df.hit == 0), 'T+-1'] =\
   df.loc[(df.aftererror == 0) & (df.hit == 0), 'rep_response_11']
df.loc[(df.aftererror == 1) | (df.hit == 1), 'T+-1'] = 0
df['T+-1'] = df['T+-1'].shift(1)

df['T-+1'] = np.nan  # np.nan
df.loc[(df.aftererror == 1) & (df.hit == 1), 'T-+1'] =\
   df.loc[(df.aftererror == 1) & (df.hit == 1), 'rep_response_11']
df.loc[(df.aftererror == 0) | (df.hit == 0), 'T-+1'] = 0
df['T-+1'] = df['T-+1'].shift(1)

df['T--1'] = np.nan  # np.nan
df.loc[(df.aftererror == 1) & (df.hit == 0), 'T--1'] =\
   df.loc[(df.aftererror == 1) & (df.hit == 0), 'rep_response_11']
df.loc[(df.aftererror == 0) | (df.hit == 1), 'T--1'] = 0
df['T--1'] = df['T--1'].shift(1)

# shifts now
# exponential fit for T++
decay_tr = np.exp(-np.arange(10)/tau)  # exp(-x/tau)
regs = [x for x in model_cols if x != 'intercept' and x != 'evidence']
N = len(decay_tr)
for reg in regs:  # all regressors (T and L)
   df[reg] = df[reg+str(1)]
   for j in range(N, len(df[reg+str(1)])):
      df[reg][j-1] = nanconv(df[reg+str(1)][j-N:j], decay_tr[::-1])
      # its j-1 for shifting purposes

# transforming transitions to left/right space
for col in [x for x in df.columns if x.startswith('T')]:
   df[col] = df[col] * (df.R_response.shift(1)*2-1)
   # {0 = Left; 1 = Right, nan=invalid}

df['intercept'] = 1

df.loc[:, model_cols].fillna(value=0, inplace=True)
# check correlation between regressors
if chck_corr:
   for j, (t, cols) in enumerate(zip(['after correct', 'after error'],
                                     [afterc_cols, aftere_cols])):
      fig, ax = plt.subplots(figsize=(16, 16))
      sns.heatmap(df.loc[df.aftererror == j,
                         cols].fillna(value=0).corr(),
```

```
                    vmin=-1, vmax=1, cmap='coolwarm', ax=ax)
            ax.set_title(t)

    return df  # resulting df with lateralized T


def glm(df):
    """
    Compute GLM weights for data in df conditioned on previous outcome.

    Parameters
    ----------
    df : dataframe
        dataframe containing regressors and response.

    Returns
    -------
    Lreg_ac : LogisticRegression model
        logistic model fit to after correct trials.
    Lreg_ae : LogisticRegression model
        logistic model fit to after error trials.

    """
    not_nan_indx = df['R_response'].notna()
    X_df_ac, y_df_ac = df.loc[(df.aftererror == 0) & not_nan_indx,
                          afterc_cols].fillna(value=0),\
        df.loc[(df.aftererror == 0) & not_nan_indx, 'R_response']
    X_df_ae, y_df_ae =\
        df.loc[(df.aftererror == 1) & not_nan_indx,
               aftere_cols].fillna(value=0),\
        df.loc[(df.aftererror == 1) & not_nan_indx, 'R_response']
    if len(np.unique(y_df_ac.values)) == 2 and len(np.unique(y_df_ae.values)) == 2:
        Lreg_ac = LogisticRegression(C=1, fit_intercept=False, penalty='l2',
                                     solver='saga', random_state=123,
                                     max_iter=10000000, n_jobs=-1)
        Lreg_ac.fit(X_df_ac.values, y_df_ac.values)
        vals_ac = np.concatenate([Lreg_ac.intercept_,
                                  Lreg_ac.coef_.flatten()])
        smodel_ac = sm.Logit(y_df_ac,
                             X_df_ac).fit(start_params=vals_ac[1::], max_iter=0)
        summary_ac = smodel_ac.summary().tables[1].as_html()
        summary_ac = pd.read_html(summary_ac, header=0, index_col=0)[0]
        p_z_ac = summary_ac['P>|z|']
        score_ac = cross_val_score(Lreg_ac, X=X_df_ac, y=y_df_ac, cv=5)
        Lreg_ae = LogisticRegression(C=1, fit_intercept=False, penalty='l2',
                                     solver='saga', random_state=123,
                                     max_iter=10000000, n_jobs=-1)
```

```
        Lreg_ae.fit(X_df_ae.values, y_df_ae.values)
        vals_ae = np.concatenate([Lreg_ae.intercept_,
                                  Lreg_ae.coef_.flatten()])
        score_ae = cross_val_score(Lreg_ae, X=X_df_ae, y=y_df_ae, cv=5)
        smodel_ae = sm.Logit(y_df_ae,
                             X_df_ae).fit(start_params=vals_ae[1::], max_iter=0)
        summary_ae = smodel_ae.summary().tables[1].as_html()
        summary_ae = pd.read_html(summary_ae, header=0, index_col=0)[0]
        p_z_ae = summary_ae['P>|z|']
    else:
        Lreg_ac = None
        Lreg_ae = None

    return Lreg_ac, Lreg_ae, score_ac, score_ae, p_z_ac, p_z_ae


def time_delay(reaction_time, ev_vals, ev_abs):
    # react_filt = reaction_time[(reaction_time <= 0.3) * (reaction_time >= 0)]
    # bins of 10 ms (10, 20, 30, 40, ... , 280, 290, 300) ms
    # bins = 20
    reaction_time_filt = reaction_time[(0 <= reaction_time) * (reaction_time <= 0.5)]
    cdf_rt = []
    cdf_rt_sm = []
    for i, e in enumerate(ev_vals):
        indx = ev_abs[(0 <= reaction_time) * (reaction_time <= 0.5)] == e
        if i == 0:  # reference CDF
            h_counts, rt_bins = np.histogram(reaction_time_filt[indx],
                                             bins=len(indx),
                                             range=[0, 0.3])
            cdf_rt.append(np.cumsum(h_counts) / np.sum(h_counts))
            cdf_rt_sm.append(mean_filt(cdf_rt[i], 30)[0])
            _, ind_ref = mean_filt(cdf_rt[i], 30)
            rt_shift = rt_bins[ind_ref]
            cdf_shift = np.zeros((len(ev_vals), len(rt_shift)))
            cdf_shift2 = np.zeros((len(ev_vals), len(rt_shift)))
        else:
            h_counts, rt_bins = np.histogram(reaction_time_filt[indx],
                                             bins=len(indx),
                                             range=[0, 0.3])
            cdf_rt.append(np.cumsum(h_counts) / np.sum(h_counts))
            cdf_rt_sm.append(mean_filt(cdf_rt[i], 30)[0])
    cdf_tog = np.zeros((2, len(ind_ref)))  # (4,...) if smoothing
    cdf_rt = np.array(cdf_rt)
    cdf_tog[1] = np.nanmean(cdf_rt_sm[1:4], axis=0)
    cdf_tog[0] = cdf_rt_sm[0]
    for j in range(len(cdf_tog)):
        for k in range(len(cdf_tog[j])):
```

```
            if np.abs(cdf_tog[j][k] - cdf_tog[0][k]) < 0.012:
                cdf_shift[j][k] = 0
            else:
                cdf_shift2[j][k] = np.interp(cdf_tog[j][k],
                                             cdf_tog[0], rt_shift) - rt_shift[k]
                cdf_shift[j][k] = np.interp(rt_shift[k],
                                            rt_shift, cdf_shift2[j])
    cdf_shift[0] = 0
    cdf_shift_tog = np.zeros((2, len(ind_ref)))
    # cdf_shift_tog[1] = np.nanmean(cdf_shift[1:4], axis=0)
    cdf_shift_tog[1] = cdf_shift[1]
    cdf_rt_mn = np.zeros((2, len(cdf_rt[0])))
    cdf_rt_mn[0] = cdf_rt[0]
    cdf_rt_mn[1] = np.nanmean(cdf_rt[1:4], axis=0)
    return cdf_rt_mn, rt_bins[1::], cdf_shift_tog, rt_shift, reaction_time_filt


def plot_time_delay_cdf(cdf_tog, cdf_shift_tog, rt_bins):
    # plt.figure(60)
    # labels = [0, 0.05, 0.1, 0.2]
    # [plt.plot(rt_bins[1::]*1000, a, label=labels[ind]) for ind, a in
    #   enumerate(cdf_rt_sm)]
    # [plt.plot(rt_bins[1::]*1000, a, label=labels[ind]) for ind, a in
    #   enumerate(cdf_rt)]
    # plt.xlabel('Reaction time (ms)')
    # plt.legend()
    plt.figure(59)
    for e_ind, shift in enumerate(cdf_shift_tog):
        # if sum(cdf_shift_tog[1]) >= 0:
        #     colour = 'blue'
        # else:
        #     colour = 'green'
        plt.plot(1000*rt_bins, shift*1000, label='nonzero',
                 color='k', alpha=0.4)
    plt.xlabel('Reaction time (ms)')
    plt.ylabel('Time delay (ms)')
    plt.xlim(0, 300)
    plt.axhline(0, linestyle='--', color='k')
    # plt.legend()


def tachometric_curves(perf, reaction_time, ev_vals, ev_abs):
    bins = 17  # 12
    # r_times = np.linspace(0, 0.3, bins+1)
    _, r_bins = np.histogram(reaction_time, bins=bins, range=[0, 0.321])
    acc_vs_rte = np.zeros((len(ev_vals), len(r_bins)))
    for i, e in enumerate(ev_vals):
```

```
        acc = []
        indx = ev_abs == e
        reac_times_e = reaction_time[indx]
        perf_vs_e = perf[indx]
        digitized = np.digitize(reac_times_e, r_bins)
        for u in range(bins+1):
            if u not in digitized:
                acc.append(np.nan)
            else:
                acc.append(np.mean(perf_vs_e[digitized == u]))
        acc_vs_rte[i, :] = acc
    return acc_vs_rte, r_bins


def cdf_comparison(ev, all_rt, ev_vals):
    for i in [0, 3]:
        indx = ev == ev_vals[i]
        if i == 0:  # reference CDF
            h_count0, rt_bins0 = np.histogram(all_rt[indx],
                                              bins=len(indx),
                                              range=[0, 0.3])
            h_count0 = np.cumsum(h_count0)/sum(h_count0)
        else:
            h_count3, rt_bins3 = np.histogram(all_rt[indx],
                                              bins=len(indx),
                                              range=[0, 0.3])
            h_count3 = np.cumsum(h_count3)/sum(h_count3)
    pl = []
    ind_min0 = np.min(np.where(h_count0 != 0))
    ind_min3 = np.min(np.where(h_count0 != 0))
    ind_min = min(ind_min0, ind_min3)
    for i in range(ind_min+2, len(rt_bins0)):
        _, p = kstest(h_count0[0:i], h_count3[0:i])
        pl.append(p)
        if p <= 0.05:
            # print(rt_bins0[i]*1000)
            # break
            return rt_bins0[i]*1000


def mean_filt(cdf, w_size):
    cdf_f = np.copy(cdf)
    mnl = []
    ind = []
    for i in range(w_size//2, len(cdf_f) - w_size//2, w_size):
        mn = np.mean(cdf_f[i - w_size//2:i + w_size//2])
        mnl.append(mn)
```

```
        ind.append(i)
    # r_mn = rt_bins[ind]
    # plt.figure(4)
    # plt.plot(rt_bins[1::], cdf)
    # plt.plot(r_mn, mnl)
    return np.array(mnl), np.array(ind)


def plot_one_tachometric(ev_vals, acc_vs_rte, r_bins):
    plt.figure()
    [plt.plot(r_bins*1000, a, label=ev_vals[e_i])
     for e_i, a in enumerate(acc_vs_rte)]
    plt.axhline(y=0.5, linestyle='--', color='k', lw=0.5)
    plt.xlabel('Reaction time (ms)')
    plt.ylabel('Accuracy')
    plt.legend()
    # plt.xlim(100, 300)


def plot_all_tachometric(ev_vals, acc_rte, rte_bins):
    plt.figure()
    acc_total_i = np.zeros((len(acc_rte), len(ev_vals), 18))
    c = ['k', 'darkred', 'red', 'gold']
    # for subject in acc_rte.keys():
    #     [plt.plot(rte_bins[subject][1::], a[1::], alpha=0.2,
    #               color=c[e_i])
    #      for e_i, a in enumerate(acc_rte[subject])]
    for i, a in enumerate(acc_rte.values()):
        acc_total_i[i, :, :] = np.array(a)
    acc_total = np.nanmean(acc_total_i, axis=0)
    r_bin_plot = [x for i, x in enumerate(rte_bins.values()) if i <= 3]
    [plt.plot(r_bin_plot[i], acc_total[i], alpha=1, color=c[i], linewidth=2.2,
              label=ev_vals[i]) for
     i, _ in enumerate(r_bin_plot)]
    err_tach = np.nanstd(acc_total_i, axis=0)/np.sqrt(np.nansum(
        ~np.isnan(acc_total_i), axis=0))
    [plt.fill_between(r_bin_plot[i], acc_total[i]-err_tach[i],
                      acc_total[i]+err_tach[i], color=c[i], alpha=0.3) for
     i, _ in enumerate(r_bin_plot)]
    plt.legend()
    plt.xlim(0, 0.3)
    plt.axhline(y=0.5, linestyle='--', color='k', lw=0.5)
    plt.xlabel('Reaction time (s)')
    plt.ylabel('Accuracy')
    # plt.fill_between(x, y-error, y+error)
```

```
def stair_smoothing(x, y, n):
    poly = np.polyfit(x, y, n)
    poly_y = np.poly1d(poly)(x)
    return poly_y


def sigmoid(x, L, x0, k, b):
    y = L / (1 + np.exp(np.float64(-k*(x-x0)))) + b
    return (y)


def time_del_mean(subjects, reac_times):
    r_times = np.arange(0, 0.35, 0.0001)
    r_array = np.empty((len(subjects), len(r_times)))
    r_array[:] = np.nan
    for i_s, _ in enumerate(subjects):
        for i_rt, rt_s in enumerate(reac_times[i_s]):
            for i_bin, r_bin in enumerate(r_times):
                if rt_s == r_bin:
                    r_array[i_s, i_bin] = time_delay[i_s][i_rt]
    td_mean = np.nanmean(r_array, axis=0)
    return td_mean
```

13.4 Legal documents

13.4.1 Consent

Information sheet for volunteer subjects

for the Informed Consent of their participation in the Study of Psychophysics

1. Study title: Investigating sequential effects in humans performing a 2AFC task presenting trial-to-trial correlations**.**

2. Promotors: Dr. Jaime de la Rocha Vázquez, Principal Investigator at Brain Circuits and Behavior Laboratory. Institut d'Investigacions Biomèdiques August Pi i Sunyer (IDIBAPS, Barcelona) is the coordinator of this study financed by the European Research Council (ERC).

3. Introduction: we would like to invite you to take part in a psychophysical experiment consisting of an auditory task run on an iPad. This research is conducted by Dr. Manuel Molano, investigator at IDIBAPS, Debora Lombardo, Biomedical Engineering student and it is directed by Dr. Jaime de la Rocha, principal investigator at IDIBAPS. Before you decide to participate it is necessary that you understand the reason why this research is carried out and how it affects the participants. Please, take your time to read the information included in this document. Do not hesitate to ask any questions if you do not understand something or if you need more information. If you decide to participate in this study we will give you a copy of this document and of the consent for you to keep it.

4. Study goals: the main goal of this research is testing how humans react when exposed to a perceptual stimulus (auditory stimulus) investigating the extent to which perceptual decisions are influenced by expectations built from recent experiences. This research will help to understand the mechanisms behind perceptual decision making. The data collected in this study will be used solely for scientific and non-profit purposes.

5. Descripción: we expect 10-20 people to take part in this experiment. The duration of this study is 2000 trials; it will take roughly two hour to complete them. The subjects can participate according to their time and date availability: it will be possible to complete the task in one session or more. In every trial the participants have to listen to a sound through headphones provided by the experimenter and respond to a question about them.

6. Purpose of the data collected in the study: the data collected in this study will only have the purpose of advancing research in neuroscience and in no case they will be used for commercial purposes. The results of this study will be disseminated to the scientific community but in no case this entails the revealing of the identity of the people who have participated in this study, since all data will be published in a pseudonymised manner [1]. Since there is no scientific reason to preserve the identity of the participants, all the data will be anonymized at the end of the study or in five years at the maximum, definitively preventing the identification of any of the participating people. In this anonymous way, the data will be saved for later analysis by our group or another collaborating research group, for a period of 10 more years.

7. Risks and discomforts

There is no risk from participating in the study. Participants will be sitting in a chair and they will perform a task run on an iPad wearing headphones. They will only have to listen to a sound and detect it.

8. Protection of confidentiality

The treatment, communication and transfer of personal data of all participants will comply with the provisions of Organic Law 15/1999, of December 13, on the protection of personal data and the Royal Decree that develops it (RD 1720/2007). Although the results obtained from the research carried out are published in scientific fields, their identity will never be disclosed. Personal data will become part of the Investigations and Clinical Trials File, which is responsible for the Consortium of the Institut de Investigacions Biomèdiques August Pi i Sunyer and will be processed solely and exclusively within the framework of your participation in this study. This file is registered with the Catalan Authority for the Protection of Dades (APDCAT; June 6, 2012). All the data of this study will be stored in electronic format and never in hard copies. The data collected will be pseudonymised, so that during its treatment, analysis or publication in the scientific field, only those responsible for this study can identify the people who have participated in this study. The pseudonymised data may be transmitted to third parties and to other countries but in no case will they contain information that can identify you, such as name and surname, initials, address, etc. In the event that this assignment occurs, it will be for the same purposes of the study described or for use in scientific publications. The promoter undertakes to establish the necessary measures to guarantee the same level of confidentiality as in Spain. You have the right of access, rectification, cancellation and opposition to said data by contacting the person in charge of the study indicated in this document in accordance with the LOPD.

9. Voluntary participation and right of participants to withdraw.

Participation in this research study is voluntary. You can refuse to participate and withdraw from the study at any time. There is no type of penalty for doing so. Participation in the study will be rewarded with a remuneration of €10 per hour.

10. Información de contacto.

If you have any question or comment about this study you can contact:

Dr. Jaime de la Rocha Vázquez

Email: jrochav@clinic.cat; Telf. 932275400 (4307).

Dirección postal: IDIBAPS, c. Rosselló 149-153, Barcelona 08036

We recommend you to keep a copy of this document to consult it in the future.

---

[1] Pseudonymization is the procedure by which the data collected is dissociated from the identity of the person who has transferred the data so that any information derived from it cannot be associated with an identified or identifiable person.

[2] Anonymized or irreversibly dissociated data refers to those data that cannot be associated with an identified or identifiable person because the link with all information that identifies the subject has been destroyed, or because said association requires an unreasonable effort, understood as the use of a disproportionate amount of time, expense and work.

Informed consent form

1. Study title: Investigating sequential effects in humans performing a 2AFC task presenting trial-to-trial correlations**.**

2. Promotors: Dr. Jaime de la Rocha Vázquez, Principal Investigator at Brain Circuits and Behavior Laboratory. Institut d'Investigacions Biomèdiques August Pi i Sunyer (IDIBAPS, Barcelona) is the coordinator of this study financed by the European Research Council (ERC).

I_____

(participant's name and surname)

With my signature at the end, I declare that:

● I talked to _____ (name of researcher), who has explained the Information Sheet point by point.

● I have read the Information Sheet.

● I have had the opportunity to ask questions about the study and they have been answered.

● I understand that my participation in this study is voluntary and that I can withdraw from the study at any time, without the obligation to give explanations and without any prejudice.

● I authorize the use and transmission of the data as described in the Information Sheet.

● For all of which, I agree to participate in the study

Date and participant's signature

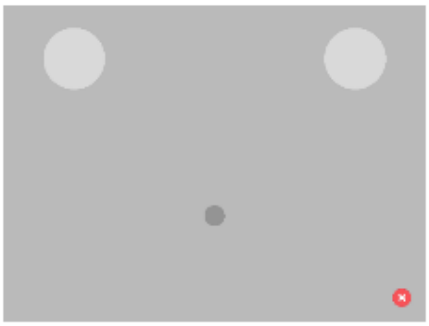
Date and experimenter's signature

13.4.2 Instructions

Instructions

STEPS:

1. Answer to some **personal questions** (name, sex, age, hand, auditory problems).
2. Put on the **headphones** provided by the experimenter.
3. **Task.** The task is divided into trials. In each trial you have to:



- **Place and hold** your finger on the small dark gray circle.
- **Sound** will be played with different intensities from the left and from the right side.
- **Drag the dark gray circle** to the light circle associated with the most intense sound.
- Always finish the trials (even if sound is not played).

You will receive feedback about your answer:
If the response is correct the screen will turn **green**.
If you do not respond in time or respond too early it will turn **yellow**.
If the response is incorrect the screen will turn **red**.

Please always respond with the same hand and use your **index finger**.

Please be **focused**, avoid responding randomly.

You will have a break every 100 trials to rest and recover.