Reproduction of a comparison between operant and classical conditioning of identical stimuli in tethered *Drosophila*

Björn Brembs, bjoern@brembs.net
Universität Regensburg, Institut für Zoologie - Neurogenetik. Regensburg, Germany

Introduction

My very first experiment as a graduate student followed from the work of my Diploma (Masters) thesis and compared two very similar experimental conditions in tethered fruit flies (*Drosophila*). In one experiment, the animals could explore four visual patterns, arranged around them. The flies were tethered and a torque meter recorded their turning attempts around their vertical body axis (yaw torque, roughly corresponding to left or right turns). Using the torque input an electric motor rotated the four patterns such that the fly could choose flight directions with respect to the angular position of the patterns (i.e., a *Drosophila* flight simulator, DFS). The four patterns were always of two types. Two identical patterns were arranged opposite of one another, such that if the fly chose to fly towards one of them, the other was behind it and the two other patterns of the different type were to the fly's left and right side. A punishing heat beam was controlled such that one pattern type was associated with heat, conditioning the fly to avoid the heat-associated pattern and prefer flight directions towards the other pattern type. The experiment lasted for 30 minutes divided into 15 two-minute periods. The experimental setup during each period was identical, with the exception of the heat. During test periods, heat was permanently switched off, while during training periods, the heat was associated with one of the two patterns. In such an operant learning experiment, the flies quickly learn to avoid the heat and are starting to prefer the unpunished pattern already after four minutes of training (Fig. 1, left panel).
The other experiment is nearly identical to the first, but during training periods, the fly has no control over its environment. Instead, it perceives the pattern/heat sequences of a previous fly, where these sequences were recorded. Such 'replay' experiments are considered a case of classical (or Pavlovian) conditioning and flies learn comparatively more slowly in such an experiment (Fig. 1, right panel). The difference between these two experiments is a case of "learning by doing": the flies in the first, operant, experiment, can actively explore their environment and learn from their actions. As a consequence, they learn faster and more reliably than their "yoked" siblings who had to passively endure the pattern/heat combinations (Fig. 1, shaded bars).

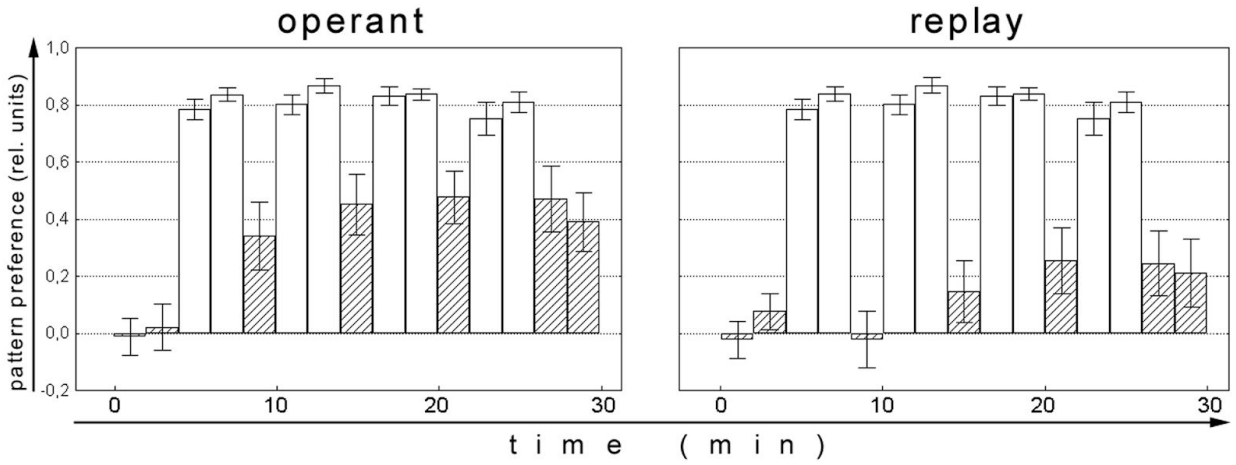

Fig. 1: *Pattern preferences (performance indices, PIs) develop over time in two different learning experiments*. Bars depict the performance index of a two-minute period. Shaded bars indicate periods without heat, where the spontaneous pattern preference was recorded (test periods). Open bars indicate periods where heat was applied (training periods). *Left panel*: Flies were in full control of the pattern position and heat at all times (operant conditioning). The high PI values of the training periods (open bars) indicate very efficient heat avoidance. Test PIs (shaded bars) show rapid learning as the heat-associated patterns are avoided and the 'non-heated' patterns preferred. *Right panel*: The flies only controlled pattern position during test periods (shaded bars) The training PIs (open bars) are identical to the ones in the left panel, as pattern position was recorded in the flies from the operant experiment and played back to these flies (classical conditioning). The behavior of the flies during training was recorded but not evaluated. Pattern preference for the unpunished patterns develops more slowly and to a lower overall level, compared to operant conditioning.

Methods and Results

The experimental methods are described in the original article. The pattern preferences of the flies were computed from the time traces of the pattern positions. A custom-developed (by Reinhard Wolf, co-author of the original publication) Turbo Pascal for MS-DOS program (code in the experiemnt_code folder) controlled the experiment and saved the data. The data were collected at 20 Hz and stored in a compressed format so as to fit on a floppy disk at the end of an experimental day. Because of hardware limitations at the time, each individual fly experiment was separated into files with three 2-min periods. Meta-data and data traces were stored in separate files. Thus, an experiment such as those shown in Fig. 1 was stored in a total of 10 files (five data files and five meta-data files). The pattern preference of individual flies was calculated as the performance index, $PI = (t_a - t_b)/(t_a + t_b)$. During training periods, t_b indicates the number of data points (i.e., time) the fly was exposed to the heat and t_a the number of data points without heat. During test periods, t_a and t_b refer to the times when the fly chose a flight direction towards the formerly unpunished or punished patterns, respectively. Computed like this, PIs vary between -1 (the fy chose to fly towards the punished patterns for the entire duration of the two minute period) and 1 (the fy chose to fly towards the unpunished patterns for the entire duration of the two minute period), with 0 indicating equal distribution of time between the two pattern types.

A sampling depth of 12 bit meant that there were 4096 potential values for pattern position that were sorted into the punished and unpunished quadrants to calculate the performance index. For each fly, the performance index of all 15 periods was calculated. The mean PIs for each experimental group over all periods was plotted with the standard errors to generate Fig. 1.

Initially, Turbo Pascal for MS-DOS was used to evaluate the data and plot the graphs. However, with the advent of MS-Windows, in the mid-late 1990s, a graduate student in our laboratory (Roman Ernst) developed C++ code using the Microsoft Foundation Classes to perform these evaluations in an interactive application (Fig. 2). This application was also able to export the performance indices in ASCII text format for statistical analysis.

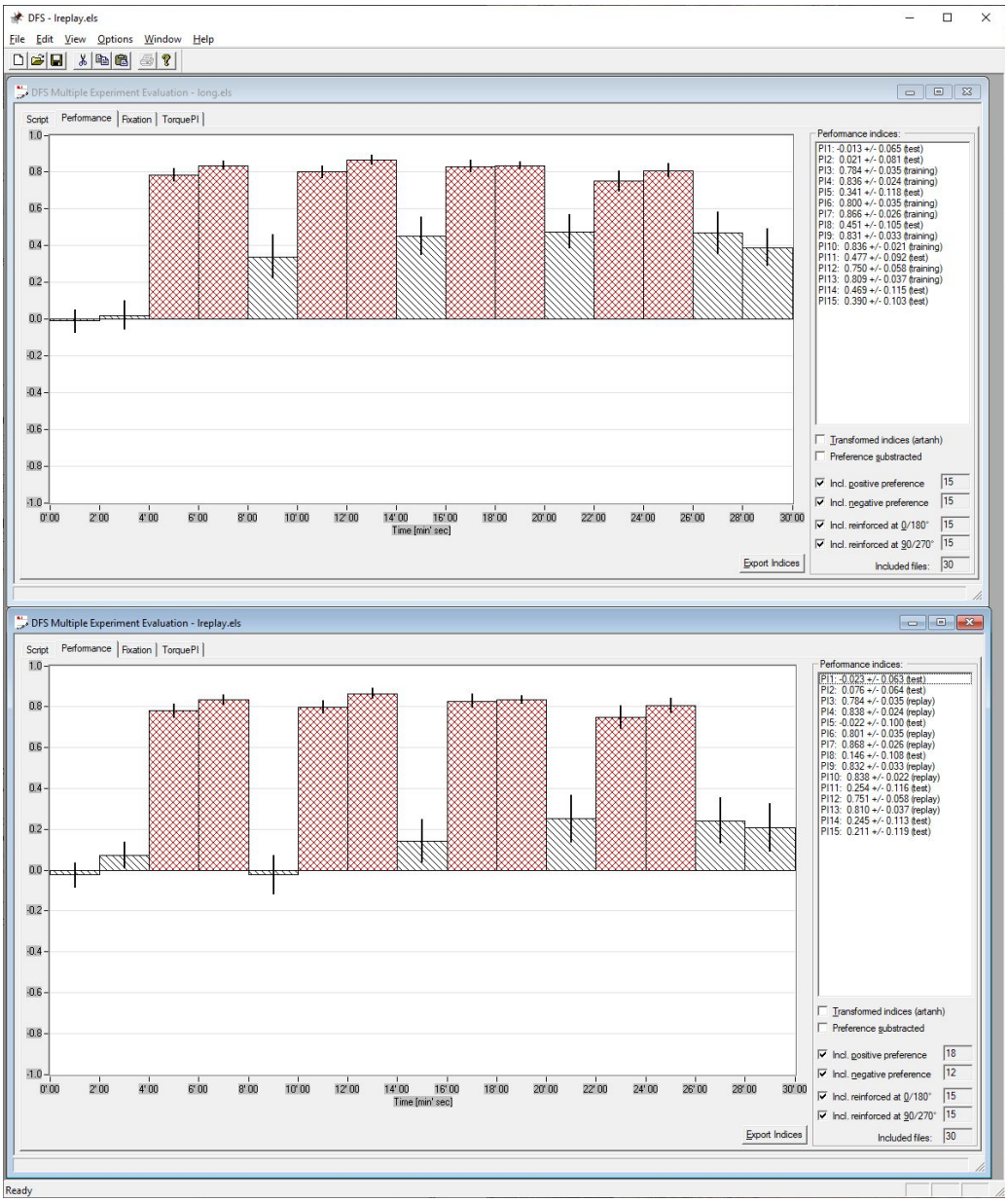

Fig. 2: *Screenshot of the interactive C++ application for DFS data evaluation and visualization*. Depicted are the same two experimental groups as in Fig. 1.

As Turbo Pascal for DOS does not run on Windows 10 any more (and its usage and development dropped in favor for the windows application already in the 1990s), the C++ application executable (in the evaluation_code folder) was used to read the original raw data (collected in the time period January 28 - February 21, 1997), compute the PIs and export them to text format (long.txt and lreplay.txt in the data folder). Until the advent of R, these text files were imported into the proprietary “Statistica” application from Statsoft. For recreating the original graph, an R script was developed (in the evaluation_code folder). To aid the plotting of the data, the headers of the ASCII text files were edited to indicate training and test periods (long_edited.txt and lreplay_edited.txt in the data folder). The R-script reads the text files and plots the data (Fig. 3).

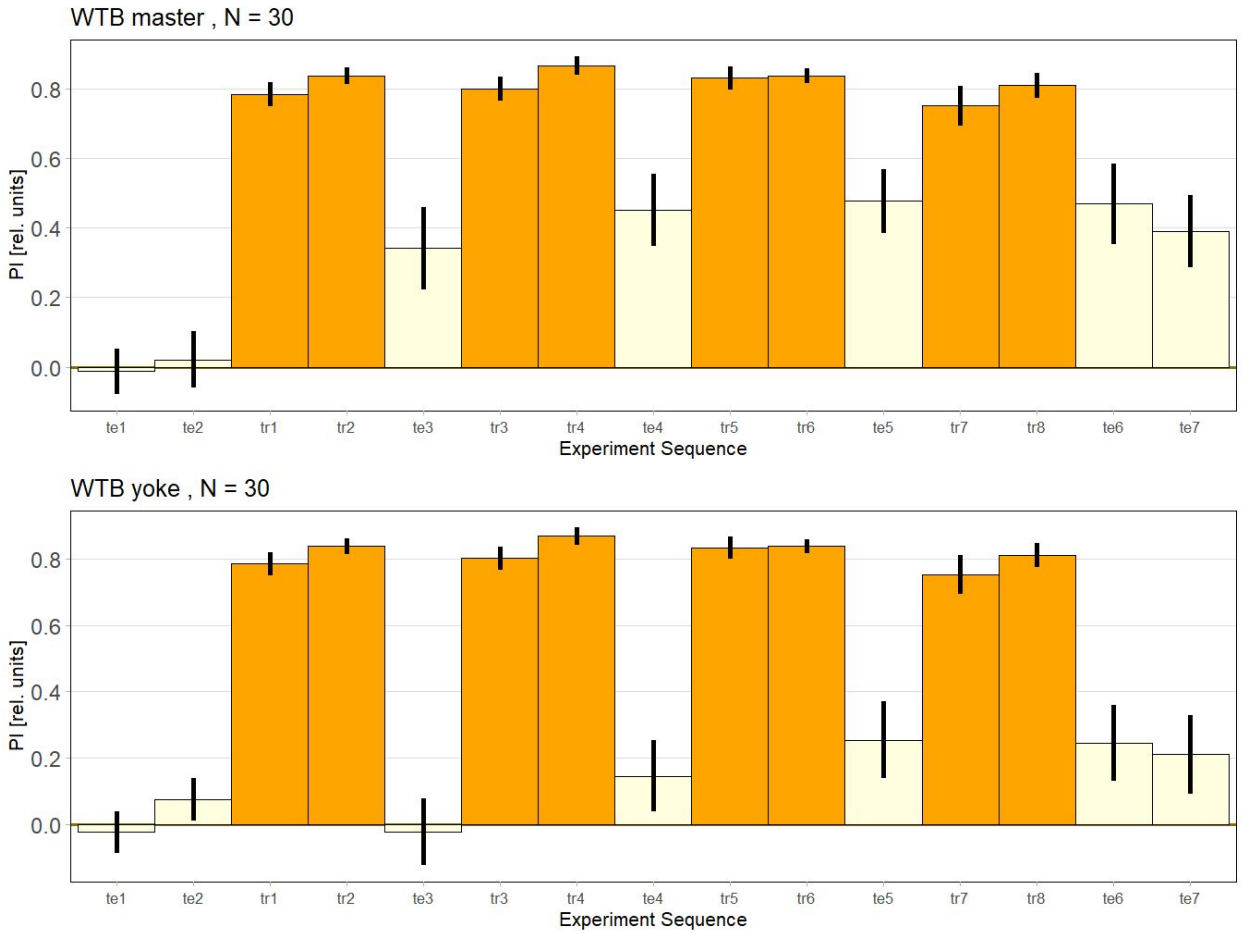

Fig. 3: *Plot from R-script recreating the original figure (Fig. 1)*.

Discussion

While the Turbo-Pascal for MS-DOS application to control the DFS was still in use until 2019, Turbo Pascal code ceased to be used or developed for data analysis already in the 1990s. Instead, a C++ application was used to visualize basic derived data such as performance indices and to export these data into text files for statistical analysis. The C++ executable is still running on current Windows 10 machines and was used to read all the original data files and export performance indices in text format.The exported text files were read by custom-written R-Scripts for recreating the original figure. Thus, for as long as the C++ executable runs on Windows, performance indices of the original raw data can be continued to be exported into text files. However, for evaluations other than performance indices (and some other evaluations provided by the C++ application), there is currently no solution. Performance index data exist in text format for all DFS data, such that at least these derived data can be reproduced for any of the published or unpublished figures.