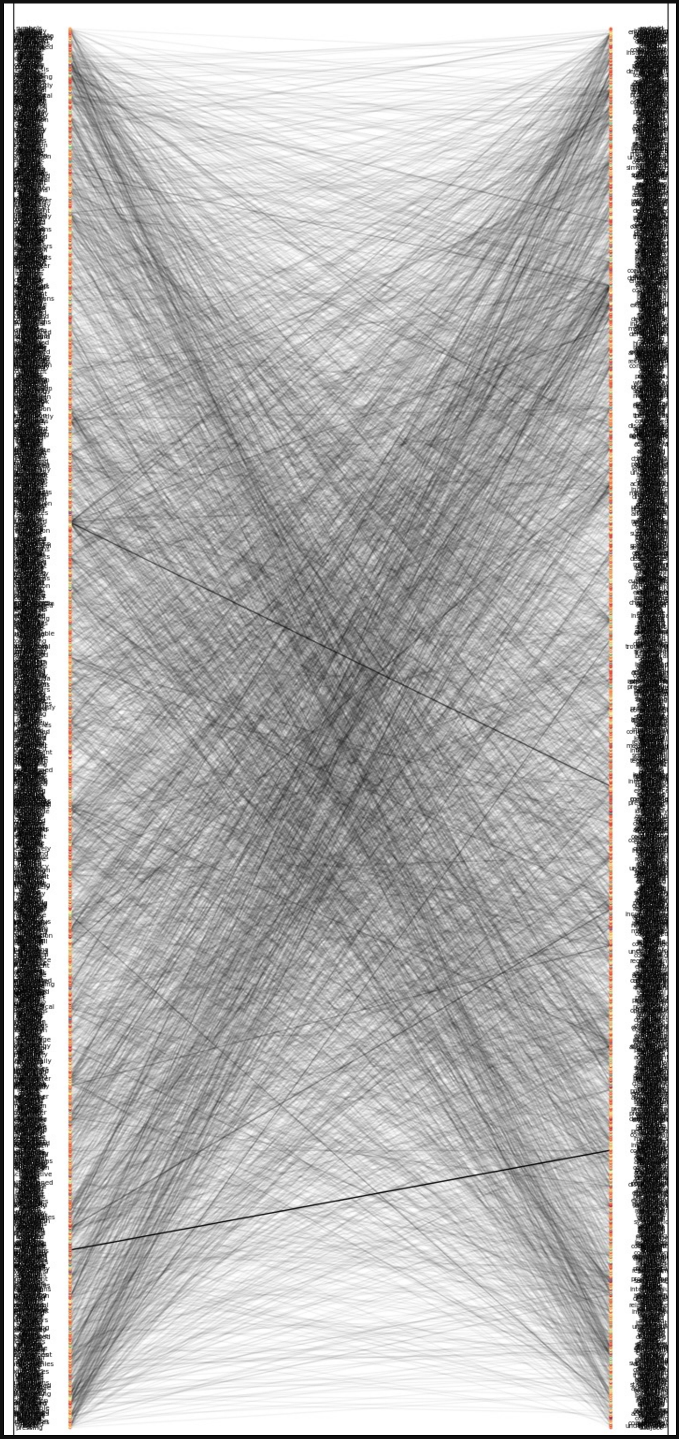
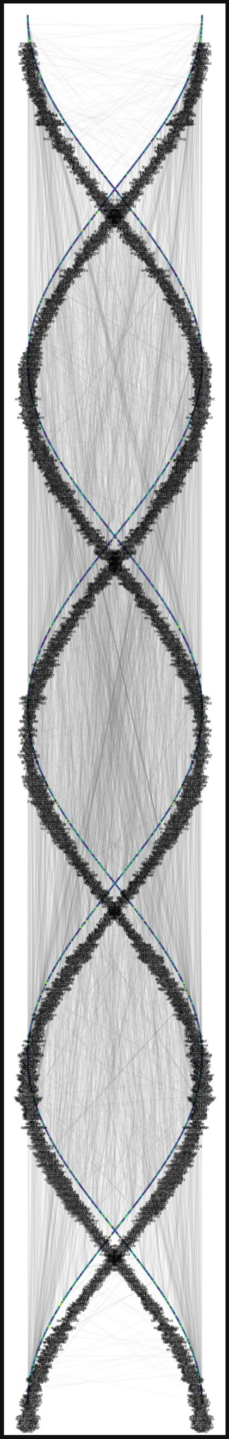
The Experimental Analysis of Human Behavior Special Interest Group Presents

THE BULLETIN

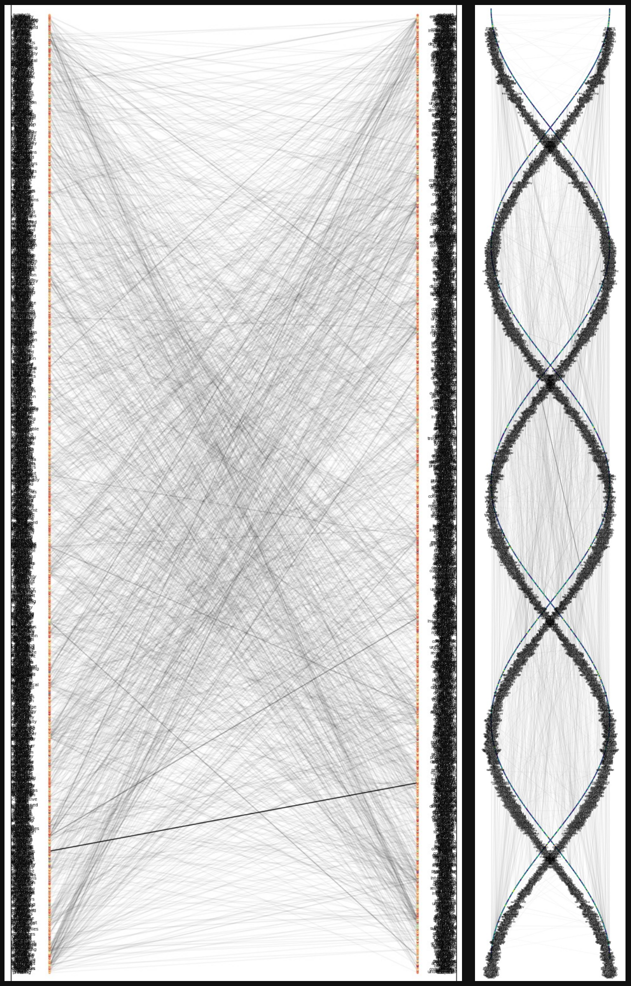




LEFT. Bipartite plot of the sequence of words in bigrams for all articles contained in this volume (left column = first word in the two-word sequence; right column = second word in the two-word sequence). Line transparency corresponds with the count of times that bigram occurred in the corpus (darker = occurred more).
RIGHT. The exact same graph but coiled in ($\cos(\theta)$, 1-$\cos(\theta)$) pattern.

Volume 34 Spring 2024 Number 1

TECHNICAL REPORTS



GAMIFIED HUMAN OPERANT RESEARCH: A BRIEF INTRODUCTION TO MINECRAFT EDUCATION

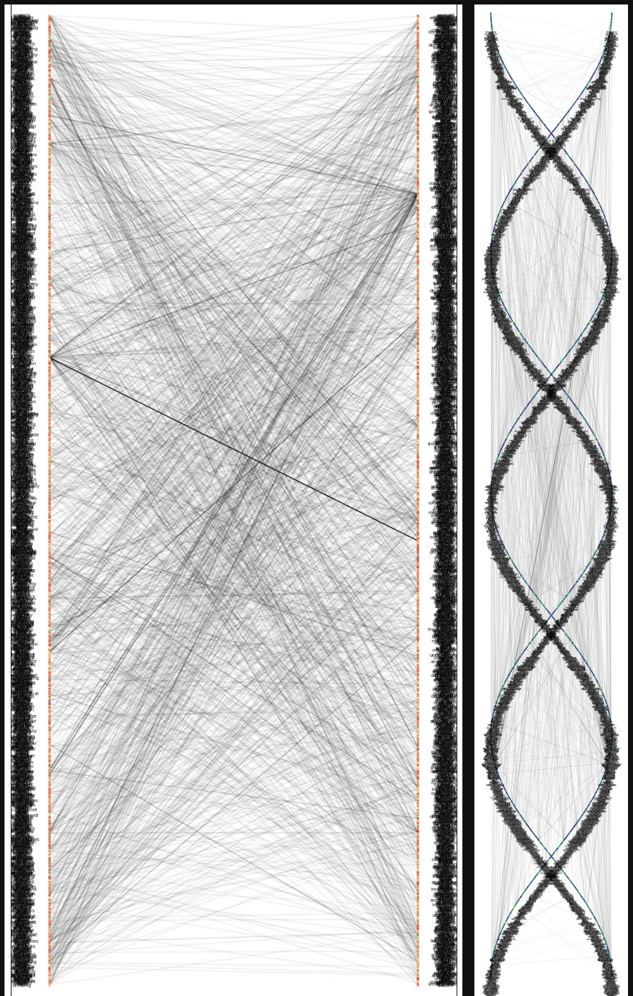
Casey Irwin Harvey, Lucille Gates, Paige Rountree, Tom Cariveau

pp. 1-8

EXTRACTING PUBLISHED GRAPHICAL DATA: A GUIDE FOR RESEARCHERS WANTING TO SYNTHESIZE SINGLE-CASE DATA

John R. Wooderson, Lewis A. Bizo, Kirsty Young

pp. 9-17

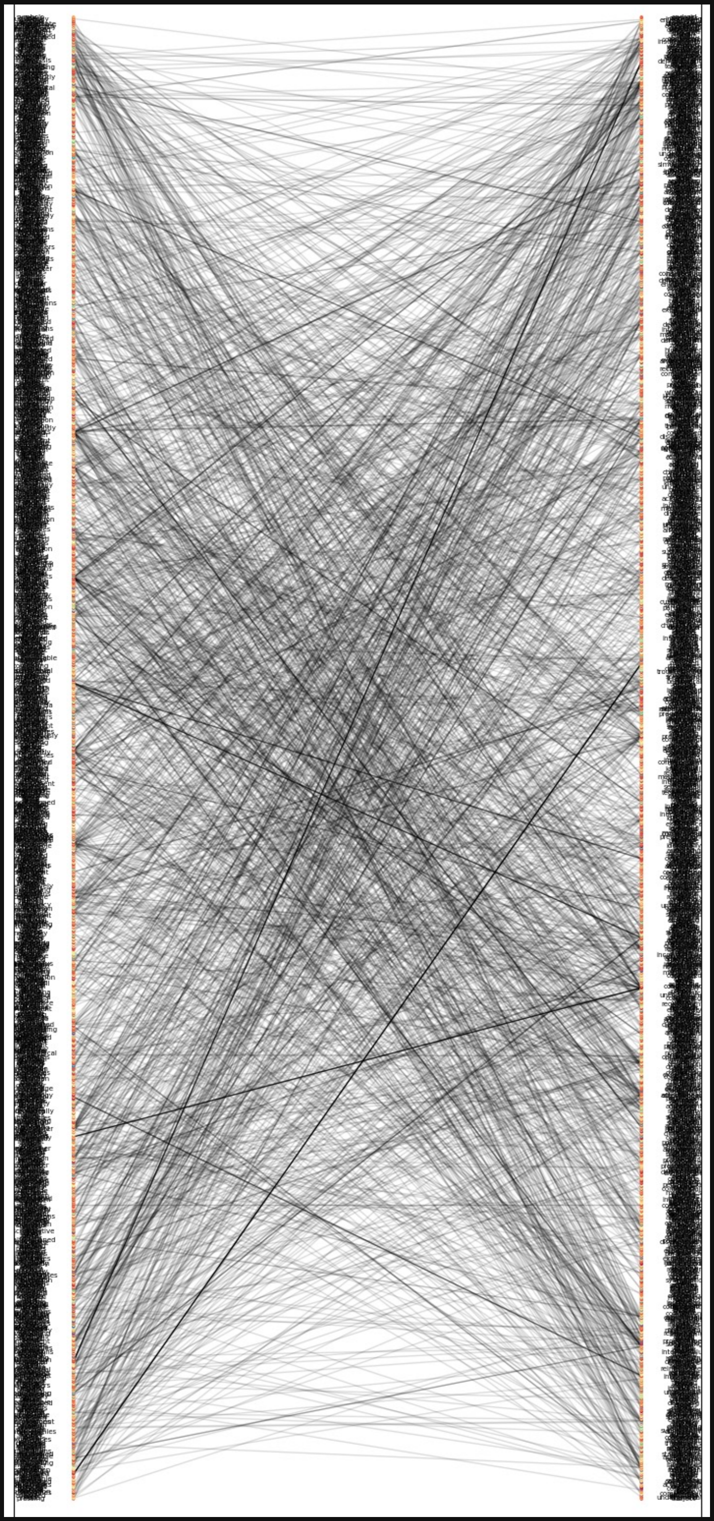


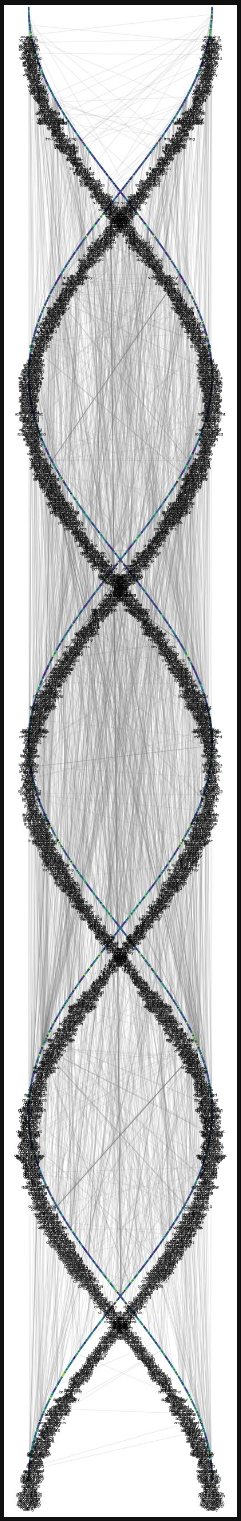
RESEARCH REPORT

ESTABLISHING A SURROGATE CONDITIONED MOTIVATING OPERATION EFFECT WITHOUT (UNCONDITIONED) MOTIVATING OPERATIONS: A PILOT INVESTIGATION

Cassidy Lehrke, Benjamin N. Witts

pp. 18-24





THE EXPERIMENTAL ANALYSIS OF HUMAN BEHAVIOR BULLETIN

About the Bulletin:

- The EAHB Bulletin (ISSN: 1938-7237) is published electronically on an ongoing basis by the Experimental Analysis of Human Behavior Special Interest Group (EAHG SIG), a group organized under the auspices of the Association for Behavior Analysis, International (ABAI). The EAHB SIG is dedicated to promoting and supporting basic behavior analytic research with human participants. Articles in the Bulletin represent the views of the authors. They are not intended to represent the approved policies of the EAHB SIG or ABAI, or the opinions of the membership of the EAHB SIG or ABAI. For more information, write to the Editor.

- The Bulletin intends to focus on the demonstration of experimental control through thoughtful experimental designs, not noteworthy or systematic findings. Therefore, we welcome manuscripts with null or unsystematic findings that may serve as fodder for future research and examples of methodological "lessons learned". In addition to experimental research on human behavior, we welcome conceptual papers, reviews, and technological papers, especially those that make inspire and improve the quality of future research in the experimental analysis of human behavior.

Types of Manuscripts Accepted

- Brief Reports (≤ 2,000 words): This section is for manuscripts that make an original, and valid contribution to the experimental analysis of human behavior. Examples include pilot studies, empirical reports, literature reviews, or theoretical treatises that address issues of interest the experimental analysis of human behavior. These submissions undergo a rigorous peer-review process and, once published, are considered final publications. As such, these reports have the status as work published in print journals.

- Research Reports: This section is for manuscripts related to the techniques and findings of studying human behavior. An experimental analysis refers to studies that systematically manipulate the environment surrounding and consequences of human behavior and measure changes in behavior using arrangements that demonstrate experimental control. These analyses can involve a large range of human populations, research designs, experimental apparatuses, and behavioral subject matter. These topics include, but are by no means limited to, discounting, stimulus control, stimulus equivalence, human-operant research, etc.

- Laboratory Descriptions (≤ 2,000 words): This section is intended for brief descriptions of laboratory procedures. Examples include novel methods or preparations, innovative data analytic methods, and descriptions of research strategies. These submissions are subject to editorial review only. These reports are not considered final publications.

- Technical Information: This section is intended for brief descriptions of technical information unique to the study of the behavior of human participants. Examples include, but are not limited to, descriptions of particularly effective programming code, innovative methods for stimulus presentation, response detection, or contingency management with human participants.

TECHNICAL INFORMATION

GAMIFIED HUMAN OPERANT RESEARCH: A BRIEF INTRODUCTION TO MINECRAFT EDUCATION

Casey Irwin Helvey[1, 2], Lucille Gates[3], Paige Rountree[3], Tom Cariveau[3]

[1]RUTGERS ROBERT WOOD JOHNSON MEDICAL SCHOOL,
[2]CHILDREN'S SPECIALIZED HOSPITAL—RUTGERS UNIVERSITY CENTER FOR AUTISM RESEARCH, EDUCATION, AND SERVICES (CSH–RUCARES),
[3]UNIVERSITY OF NORTH CAROLINA WILMINGTON

The notion of boredom as it relates to subject interest in experimental tasks has appeared in discussions about human operant research for decades (e.g., Baron & Perone, 1998; Case et al., 1990; Galizio & Buskist, 1988; Saini & Roane, 2018). Programming dynamic, game-like experimental paradigms may help minimize issues with attrition or boredom associated with otherwise arbitrary human operant tasks (Baron & Perone, 1998; Galizio & Buskist, 1998; Pilgrim 1998; Savastano & Fantino, 1994; Schmitt, 1998). Modern devices and software are particularly suitable for executing gamified research programs due to their accessibility and malleability. Indeed, several games currently exist that could be used by researchers to present complex experimental tasks using in-game building or editing features in lieu of coding.

Establishing a research program that utilizes computer-based games can be a resource-intensive endeavor, demanding significant time and financial investment due to the reliance on individuals skilled in coding or programming. However, employing commercially available games that offer customizable features with minimal or no coding requirements (e.g., Minecraft, Unreal Editor for Fortnite, Roblox) represents a practical solution to these challenges. Many of these games come equipped with integrated tools that allow human operant researchers to create well-controlled experimental paradigms and, in some cases, automate data collection. Although researchers in behavior analysis have used commercially available games to answer questions relevant to the experimental analysis of human behavior (e.g., Schenk & Reed, 2020), certain game mechanics, particularly those that cannot be manipulated by the experimenter, can introduce experimental confounds that may impact the findings of the study (McMahan et al., 2011; Schenk & Reed, 2020). For example, in Mario Kart, randomly generated items (e.g., bananas, red shells, mushrooms, coins) may have greater effects on responding than any possible contingencnies arranged by the experimenter. Alternatively, commercially available games that are highly customizable, such as sandbox games, offer a unique advantage: researchers can design highly controlled but dynamic experimental paradigms for the study of human operant behavior.

The term *sandbox game* describes a class of games that include open-world elements, meaning that the player's movement and interactions within the game are largely unrestricted such that tasks that are player, rather than game or plot, initiated (Rouse, 2022). These games may be contrasted with other games that include progression-based and linear storylines much like the plot of a movie. The customization of sandbox games may be particularly relevant to human operant researchers as the many open-world elements allow for diverse antecedent and consequent conditions to be arranged in video game environments. Although participant's experience with these or similar video games might represent a possible experimental confound, given the versatility of sandbox games, human operant researchers can arrange discrete response-consequence relations that are unique to the experimental paradigm. Relatedly, participants' history with these or related video games might benefit the human operant researcher as participants might be

Author Note: Address correspondence to Casey Irwin Helvey, CSH-RUCARES, 888 Easton Ave., Somerset, NJ 08873. Email: cih18@rutgers.edu

https://doi.org/10.17605/OSF.IO/VT58F

familiar with the in-game controls. Such a history might absolve the researcher from providing extensive training or instruction before the experimental conditions can be introduced.

To highlight the potential power of sandbox games as a tool for developing experimental procedures, the current paper will briefly introduce Minecraft Education (hereafter *Minecraft*), which we have used in our lab to conduct a small program of human operant research. Minecraft is played in a three-dimensional world in which players can use different tools (e.g., pickaxe) and items to interact with distinct environmental components (e.g., block types). In 2016, Microsoft, in collaboration with Mojang Studios, released an education edition of the game which was designed to be used as an instructional tool in classroom settings. This version of Minecraft allows users to engage in single- and multi-player game modes across more than 500 user- and developer-built lessons in science, coding, history, and various other subjects (Minecraft, n.d.). Minecraft can be used across various platforms or devices, including Mac, Windows, Chromebook, iPad, iPhone, and Android phones and tablets. This wide accessibility is noteworthy as it allows a diverse range of participants and researchers to engage with the software. Just one year after being released, Minecraft had over 2 million licensed users in over 115 countries worldwide. At the time of this paper, licenses are free to anyone with an active Microsoft Office 365 Education (.edu) account or can be purchased for $12.99 annually per user for individuals who are not part of an eligible educational institution. By virtue of its expansive and adaptable three-dimensional world and widespread accessibility, Minecraft is an innovative program for developing experimental procedures to study human operant behavior.

USING MINECRAFT

The following sections will describe several features of Minecraft that we have used to develop or manage human operant paradigms. Additional links that include videos or other descriptions of this content are accessible as a Supplemental Material.

User Controls

From the participant's perspective, navigating the Minecraft world requires mastery of only a few controls. These controls differ across devices (e.g., computer or tablet). Because most human operant laboratories continue to use standard desktop computers, computer controls will be described but readers are referred to the Supplemental Material for controls on other platforms. Once a participant joins a Minecraft world using a unique code, they can move their character using the WASD keys on the keyboard. Participants can also alter their view by moving the mouse, left-click to break (i.e., mine) blocks, and right-click to place blocks or interact with items in the world (e.g., press buttons). These controls are assigned by default but can be modified or removed by the experimenter.

Hosting and Player Permissions

A key feature of Minecraft is the ability to host a world that is remotely accessible to other players from any device. The hosting player (i.e., the experimenter) can invite other players to join by sharing a unique code or link. Although participants may join remotely from any location, our lab required participants to join the experimental world from a room in the lab equipped with a desk, chair, and desktop computer. This in-person requirement was used to reduce possible experimental confounds such as access to (a) different peripherals (e.g., computer mouse and keyboard vs. trackpad), (b) internet or graphic capabilities of the device, or (c) distraction-free settings. Moreover, the experimenter was able to monitor the participant during the experimental tasks both in the room and in the game using a one-way mirror and recording software, respectively. Allowing remote participation might enhance the recruitment of diverse and understudied populations in human operant research, but researchers should consider additional safeguards, such as remote monitoring to screen for potential impostor participants (e.g., bots) and to account for extraneous influences on participant responding. In-game observations also allow for periodic fidelity checks to ensure that the player permissions are set properly.

A standard set of player permissions can be programmed by assigning the participant to one of three in-game roles: member, visitor, or operator. *Members* can place or break blocks and other items in the world, whereas *visitors* can only explore without interacting with blocks or items. *Operators* can select default permissions for each player and execute slash commands (described below). In our lab, participants' roles are set as *members* and other game rules are set (described below) to ensure that the participant cannot alter the environment. Additional settings can be programmed to best suit the experimenters' needs including various game rules and world settings (e.g., visibility, weather conditions). Although these tools were intended to help educators customize instructional lessons and manage students' experiences, they can also be utilized by human operant researchers to maintain controlled experimental conditions.

Two game rules that are important to highlight are (a) world builder and (b) immutable world. Both game rules can be toggled between *true* and *false* by an operator. Only when world builder is set to *true* can operators and members place or break blocks in the Minecraft world. When world builder is set to *false*, players will not be able to place or break blocks; however, the player can interact with other elements of the world, such as pressing buttons or levers, shooting arrows, and picking up items. Setting world builder to *false* is recommended during experimental tasks so that the participant cannot alter other features of the environment. The second game rule is *immutable world*, which can be used to allow members and operators without the world builder ability to place or break blocks. Specifically, if immutable world is set to *false*, then players can change the world regardless of their world builder status. However, if immutable world is set to *true*, then players can only change the world if their world builder status is also set to *true*.

Executing Commands

Human operant researchers might also utilize various in-game commands (known as *slash commands*), which allow the experimenter to present in-game effects (e.g., teleporting or delivering an item to a participant). Several default commands can be used to perform various operations in the world. To enable commands, the experimenter must *activate cheats* in the settings menu. If a command is

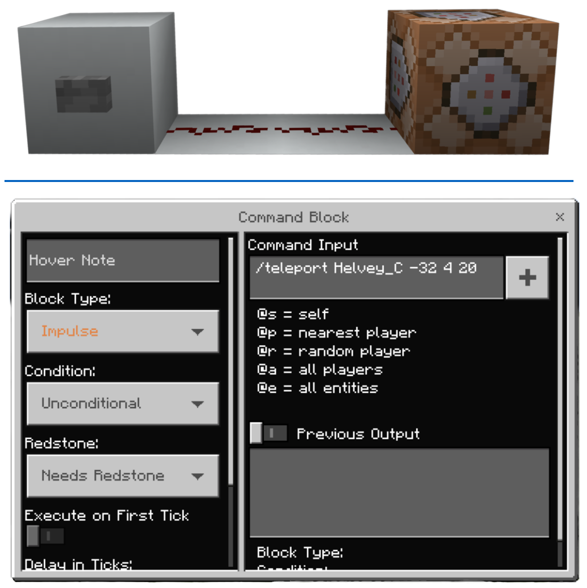

Command Block ×
Hover Note
Block Type: Impulse
Condition: Unconditional
Redstone: Needs Redstone
Execute on First Tick
Delay in Ticks:
Command Input
/teleport Helvey_C -32 4 20
@s = self
@p = nearest player
@r = random player
@a = all players
@e = all entities
Previous Output
Block Type:

Figure 1. Example Logic Gate (top) and Command Block User Interface (bottom).

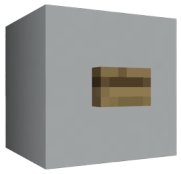
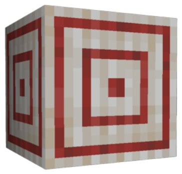
used infrequently, the experimenter can execute any command using the chat window. If a command may be executed repeatedly or triggered by some environmental event (e.g., participant's responding), then the experimenter can instead program the command into a *command block*, which can be placed anywhere in the world. Command blocks are not regularly available in the inventory and must be obtained using a command in the chat window. Once placed, each command block has an interface where the designated command can be input (Figure 1, bottom panel).

Human operant researchers should also be aware of the *clone* command. Using this command, researchers are able to duplicate previously built structures or paradigms. The experimenter must define the area to be cloned as well as the final placement of the cloned structure. In our research, cloning has been used to duplicate chambers which serve as individual trials. Without this function, reproducing identical chambers would represent a nearly insurmountable task.

Circuitry

Minecraft includes powered blocks, known as *redstone*, that can be used as digital logic gates. Logic gates are electronic circuits that process binary information: 0 (i.e., OFF) or 1

Table 1. *Examples of In-Game Manipulandum*

Name	Item	Effect
Lever		Pressing the lever closes/opens the circuit.
Button		Pressing the wooden button closes the circuit for 2 s.
Target		Shooting the target with a bow and arrow closes the circuit for 2 s.
Pressure plate		Standing on the pressure plate closes the circuit.
Trip wire		Walking through the trip wire closes the circuit for 2 s.

(i.e., ON). To build a simple input/output logic gate in Minecraft, the experimenter can connect a lever, or other inputs, to a line of redstone. Right-clicking the lever will power the redstone (i.e., close the circuit) and right-clicking again will remove power (i.e., open the circuit) to the redstone.

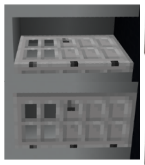
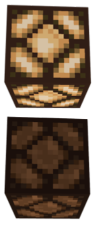
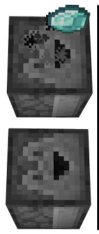
Several distinct manipulanda can be used as the input for a simple logic gate (see Table 1), which can then be connected to various in-game mechanics (see Table 2) as outputs. An example logic gate using a button and command block to teleport a participant (Helvey_C) is shown in Figure 1. Pressing the button would power the logic gate and execute the teleport command through the command block. Redstone is not required to power a command block as many of the manipulanda can be placed directly on the block by crouching (Shift) and right clicking on the block. However, experimenters may choose to use redstone to connect the manipulandum and command block so that it (a) is not visible to the participant, (b) can be connected to other mechanics, or (c) cannot be accessed by the participant.

Specialty Blocks and Crafting Features

Minecraft includes a variety of blocks that experimenters can use to create and customize their experimental paradigm. Several distinct types of blocks may be used to build (e.g., grass, dirt, stone, glass), decorate (e.g., flowers, banners, paintings), transport (e.g., rails, minecarts, boats), and provide instructions to participants (e.g., signs).

In addition to these basic blocks, Minecraft also features several specialty blocks that can be placed to constrain participants' activity. First, *border blocks* may be used to create an invisible barrier, which can be placed inconspicuously in

Table 2. *Examples of In-Game Effects.*

Consequence	Effect	Slash command	In-game effect
Door	A door can be opened given a redstone input.		
Redstone lamp	A lamp can be illuminated given a redstone input.		
Dropper	An item can be dispensed from a dropper given a redstone input.		
Title message	A message can be presented across the participants' screen using a slash command.	/title [participant] title [-1/+1] Point	−1 Point +1 Point
Nausea	The participant's view is distorted.	/effect [participant] nausea	
Speed	The participant's walking speed can be increased or decreased.	/effect [participant] speed or slowness	
Give/remove	An item can be delivered or removed from the participant's inventory.	/give or clear [participant] [item name] #	
Teleport	The participant can be teleported to a different coordinate.	/teleport [participant] [X Y Z]	

the paradigm (e.g., underground). Participants are unable to pass over or under these blocks. Border blocks can be placed to keep participants in certain areas and prevent them from accessing key components of the experimental paradigm (e.g., command blocks). The second type of specialty block that we have found useful for research is the *allow block*. An allow block permits users to place and break blocks in areas above them, even when other player permissions would not otherwise allow building. These blocks can be placed in areas that the researchers plan to allow participants to build, such as part of a free-operant experimental task or in an area of the world that participants are given contingent access to during reinforcement. The third type of specialty block is the *deny block*, which prevents users from placing or breaking blocks in areas above them, even when other player permissions would otherwise permit building.

Program Elements Relevant to Human Operant Researchers

We began using Minecraft in our research in 2019 as it represented a particularly malleable program that could be used to generate complex experimental tasks for researchers without coding experience. Although our initial paradigms would best be described as replicas of a standard operant chamber, researchers can include more sophisticated arrangements to capitalize on the various game elements that can be programmed in Minecraft. As one example, we recently prepared a dungeon-like map (see Supplemental Materials for video example of one such paradigm) in which participants progressed through dark chambers as they completed experimental tasks (e.g., delayed matching-to-sample, concurrent operants). Doing so allowed for the repeated measurement of the target response, while avoiding the tedious stationary button-pressing of our initial paradigms.

When creating a gamified research paradigm in Minecraft, the human operant researcher may be heartened by the substantial variety of blocks at their disposal. These blocks can be used to arrange various contexts, discriminative conditions, or bolster the general aesthetics of their paradigm. A description of the types of blocks is beyond the scope of this paper and may best be realized by exploring the game. Several manipulanda can also be arranged as can various effects used as consequent events. Some examples of the manipulanda and consequence events are described below, although they represent only a portion of possible mechanics that may be used by human operant researchers.

The manipulanda available in Minecraft can be used to trigger consequent events directly or according to whatever experimental parameter is of interest to the researcher (e.g., schedule, delay). Some of the manipulanda are shown in Table 1, which include a lever, button, target, pressure plate, and trip wire. These represent the majority of the manipulanda used by our lab and each serves a distinct function therein. As an example, the lever can be rapidly pressed using the right-mouse button, which results in the logic gate being powered on or off. In contrast, right clicking a button will power the gate for a brief period (e.g., 2 s) before automatically turning off. The duration of the interval differs across the types of buttons (e.g., wood or stone) and another response cannot be emitted until the interval has elapsed. These manipulanda can then be connected to various outputs.

Table 2 shows a small sample of outputs that may be of interest to the human operant researcher. These outputs are not specific to any given input mechanism. Indeed, the same manipulanda could open a door, produce access to a new area of a chamber, deliver or remove points and items, or spawn enemy players. The human operant researcher can use these outputs to setup whatever conditions are germane to their research program.

In addition to the mechanical outputs that may be arranged in Minecraft, experimenters can also arrange in-game status effects. Like the outputs described above, these participant-directed effects can be added or removed following some environmental event or at any point by the experimenter. One example is the nausea effect, which alters the participant's field-of-view by creating a wobble effect. Experimenters can also alter the participants' speed across a range of values (i.e., 1=slowest, 255=fastest). These effects can also be applied for a set duration, can be terminated, or can be reinstated based on an environmental event. Overall, the abundance of manipulanda and effects that can be programmed in Minecraft offers incredible flexibility to human operant researchers, which can be leveraged to study a myriad of behavioral phenomena.

Another program element that can enhance the utility of Minecraft in human operant research is automated data collection. Experimenters can automatically record the participant's responding on the manipulanda by adding a *scoreboard* using a slash command (see Supplemental Materials). Once a scoreboard is added, the experimenter can arrange for any response by the participant to be logged in the scoreboard. This scoreboard can be made visible to the participants or only to the experimenter. The placement of the scoreboard can also be set by the experimenter. Specifically, the scoreboard can be made to appear on the right side of the participant's screen or accessible in the player menu, which is accessed by pressing the Escape key. The scoreboard can also be reset to allow for trial- or session-specific data collection.

ADVANTAGES AND POTENTIAL USES

Minecraft is a software program with many advantages for human operant researchers interested in gamifying their research paradigms. Some of the most notable advantages include: (a) the ability to host and join an experiment from a wide range of devices, (b) user-friendly interfaces and controls, and (c) the considerable number of aesthetic and functional resources that are available in the game. These advantages may serve the human operant researcher by allowing for: (a) greater accessibility by understudied populations, (b) reduced instructional or training requirements, and (c) nearly endless customization of experimental paradigms.

Given the abundance of aesthetic and functional designs available in Minecraft, we anticipate that this program would not be limited to any particular research question. Nevertheless, we have found it to be a particularly helpful paradigm for exploring our research interests in social behavior, observing behavior, and choice. As one example, we have conducted several experiments on dyadic competition, which sought to extend the limited research on competitive and cooperative contingencies (e.g., Schmitt, 1976; 1998). Previous research in this area has reported instances of participant attrition (Schmitt, 1976), which were likely related to features of the study (e.g., schedule of reinforcement), but also the monotony of the task and associated operanda (e.g., Lindsley knob, audit counters, toggles, and switches). Given that most video games include a cooperative or competitive contingency, we felt that similar gamified arrangements, developed in Minecraft, would be a meaningful approach to extend the previous work of David Schmitt and Don Hake (e.g., Hake & Vukelich, 1972; Schmitt, 1984; 1986). Further, some of the challenges faced by these researchers were related to the delivery of conditioned reinforcers (Schmitt & Marwell, 1972), which might also be diminished in video game paradigms. Indeed, a large amount of behavior and money is allocated to the accumulation of in-game resources with no tangible benefit to the player (Marder et al., 2019).

We also recently began to explore ways to arrange in-game conditioned reinforcers, which has proven to be a fruitful area of exploration particularly as it relates to the financial sustainability of our lab. For example, experimenters may program the delivery of in-game items (e.g., gems) as tokens, which many be exchanged for other in-game resources (e.g., apparel, weaponry) instead of traditional monetary compensation that is delivered contingent on performance or general participation. We hope that similar gamified arrangements might be utilized by other research labs and, in doing so, might provoke the human operant researcher to consider alternatives to trinkets, candies, or raffle tickets in their own research program. While our research using Minecraft has predominantly employed discrete-trial or performance-based arrangements, the nature of Minecraft as a sandbox game with minimal character or environment restrictions makes it an ideal program for researchers interested in studying free operant behavior.

We hope that we have sufficiently described the potential utility and malleability of Minecraft in human operant research and also goaded our fellow researchers into considering how their own research questions may be pursued in a gamified format. Doing so might help sustain otherwise costly research programs, but also increase participation, enjoyment, and acceptability of our procedures.

REFERENCES

Baron, A., Perone, M. (1998). Experimental design and analysis in the laboratory study of human operant behavior. In K. A. Lattal & M. Perone (Eds.), *Handbook of research methods in human operant behavior* (pp. 45-91). Springer. https://doi.org/10.1007/978-1-4899-1947-2_3

Case, D. A., Ploog, B. O., & Fantino, E. (1990). Observing behavior in a computer game. *Journal of the Experimental Analysis of Behavior, 54*(3), 185–199. https://doi.org/10.1901/jeab.1990.54-185

Galizio, M., & Buskist, W. (1988). Laboratory lore and research practices in the experimental analysis of human behavior: Selecting reinforcers and arranging contingencies. *The Behavior Analyst*, *11*(1), 65-69. https://doi.org/10.1007/BF03392457

Hake, D. F., & Vukelich, R. (1972). A classification and review of cooperation procedures. *Journal of the Experimental Analysis of Behavior*, *18*(2), 333-343. https://doi.org/10.1901/jeab.1972.18-333

Marder, B., Gattig, D., Collins, E., Pitt, L., Kietzmann, J., & Erz, A. (2019). The avatar's new clothes: Understanding why players purchase non-functional items in free-to-play games. *Computers in Human Behavior, 91*, 72-83. https://doi.org/10.1016/j.chb.2018.09.006

McMahan, R. P., Ragan, E. D., Leal, A., Beaton, R. J., & Bowman, D. A. (2011). Considerations for the use of commercial video games in controlled experiments. *Entertainment Computing*, *2*(1), 3-9. https://doi.org/10.1016/j.entcom.2011.03.002

Minecraft Education. (n.d.) Retrieved March 27, 2023, from https://education.minecraft.net/en-us

Pilgrim, C. (1998). The human subject. In K. A. Lattal & M. Perone (Eds.), *Handbook of research methods in human operant behavior* (pp. 15-44). Springer. https://doi.org/10.1007/978-1-4899-1947-2_2

Rouse, M. (2022, June 3). *Sandbox*. Techopedia. https://www.techopedia.com/definition/3952/sandbox-gaming

Saini, V., & Roane, H. S. (2018). Technological advances in the experimental analysis of human behavior. *Behavior Analysis: Research and Practice*, *18*(3), 288-304. https://doi.org/10.1037/bar0000124

Savastano, H. I., & Fantino, E. (1994). Human choice in concurrent ratio-interval schedules of reinforcement. *Journal of the Experimental Analysis of Behavior*, *61*(3), 453-463. https://doi.org/10.1901/jeab.1994.61-453

Schenk, M. J., & Reed, D. D. (2020). Experimental evaluation of matching via a commercially available basketball video game. *Journal of Applied Behavior Analysis*, *53*(1), 209-221. https://doi.org/10.1002/jaba.551

Schmitt, D. R. (1976). Some conditions affecting the choice to cooperate or compete. *Journal of the Experimental Analysis of Behavior*, *25*(2), 165-178. https://doi.org/10.1901/jeab.1976.25-165

Schmitt, D. (1984). Interpersonal relations: Cooperation and competition. *Journal of the Experimental Analysis of Behavior*, *42*(3), 377-383. https://doi.org/10.1901/jeab.1984.42-377

Schmitt, D. (1986). Competition: Some behavioral issues. *The Behavior Analyst*, *9*(1), 27-34. https://doi.org/10.1007/BF03391927

Schmitt, D.R. (1998). Social behavior. In K. A. Lattal & M. Perone (Eds.), *Handbook of research methods in human operant behavior* (pp. 471-505). Springer. https://doi.org/10.1007/978-1-4899-1947-2_15

Schmitt, D. R., & Marwell, G. (1972). Experimental use of points or money as reinforcers: Does what is risked make a difference? *Psychological Reports*, *31*(2), 425-426. https://doi.org/10.2466/pr0.1972.31.2.42

TECHNICAL INFORMATION

EXTRACTING PUBLISHED GRAPHICAL DATA: A GUIDE FOR RESEARCHERS WANTING TO SYNTHESIZE SINGLE-CASE DATA

John R. Wooderson[1, 2], Lewis A. Bizo[3], Kirsty Young[1]

[1]UNIVERSITY OF TECHNOLOGY SYDNEY,
[2]THE KAMELEON GROUP,
[3]CHARLES STUART UNIVERSITY

Graphical data displays and visual analysis are cornerstones of behavior-analytic research. However, graphical data present challenges when conducting analyses across published studies. Specifically, single-case experimental design graphs frequently employ different axis scales across studies and sometimes within studies; and researchers often don't publish the raw data. Consequently, researchers wanting to compare or reanalyze data across published literature often need to extract data from plotted line graphs. In cases where raw data is inaccessible, data extraction software programs provide a valid and reliable method for digitizing graphed data sets from published single-case experimental design studies. Our purpose in writing this article is to demonstrate how to extract graphical data from published articles using the software program *DigitizeIt*™. We present a series of task analyses and an example for readers to follow to import single-case graphs into the program, plot data points, and export the numerical data to a spreadsheet program.

Keywords: single-case experimental designs, data extraction, task analysis, visual analysis

Single-case experimental designs (SCED) are characterized by ongoing, repeated measurement and replication across conditions or participants—which are critical concerns in studying the behavior of individual organisms (Kazdin, 2021). One of the many benefits of SCEDs is that they allow for precise examination of behavioral variability at the level of the individual. In comparison, group-level designs focus on the effect of an intervention at the group level, which may fail to detect potentially important sources of variability among individuals (Johnston & Pennypacker, 2010). Further, well-designed SCEDs allow researchers to control for multiple threats to internal validity and draw causal inferences. Consequently, SCEDs provide researchers with viable alternatives to large group studies (Lobo et al., 2017). Due to these methodological advantages, researchers and practitioners in applied behavior analysis and related fields primarily use SCEDs and visual analysis methods to evaluate intervention outcomes (Horner & Swoboda, 2014; Wolfe et al., 2019). Specifically, they use visual inspection of data plotted on line graphs to make decisions about whether to adjust treatment during intervention and determine the strength of causal relations between behavior and environmental variables (Kratochwill et al., 2013; Ledford et al., 2018).

This reliance on visual analysis presents some challenges when comparing data across studies. Specifically, SCED graphs frequently employ different axis scales across studies and, sometimes, within studies. Also, SCED graphs may contain more than one independent variable (e.g., alternating treatments designs), which may confound visual comparisons with other studies. As a result, several supplemental statistical measures exist for estimating effect sizes for SCED data (Maggin et al., 2017). Still, visual analysis remains the most appropriate methodology for examining variability in SCED data stability, level, and trend (Ledford et al., 2018). Researchers and practitioners wanting to use visual analysis to evaluate the impact of interventions by comparing data patterns across studies must first extract the data from the published literature.

Author Note: Address correspondence to John R. Wooderson (ORCID ID:0000-0001-9255-7874). Email: john.r.wooderson@student.uts.edu.au

Additional Author Information: Lewis A. Bizo: ORCID: 0000-0001-6030-6149, lbizo@csu.edu.au; Kirsty Young: ORCID: 0000-0002-6455-995X, Kirsty.Young@uts.edu.au

https://doi.org/10.17605/OSF.IO/RFHJX

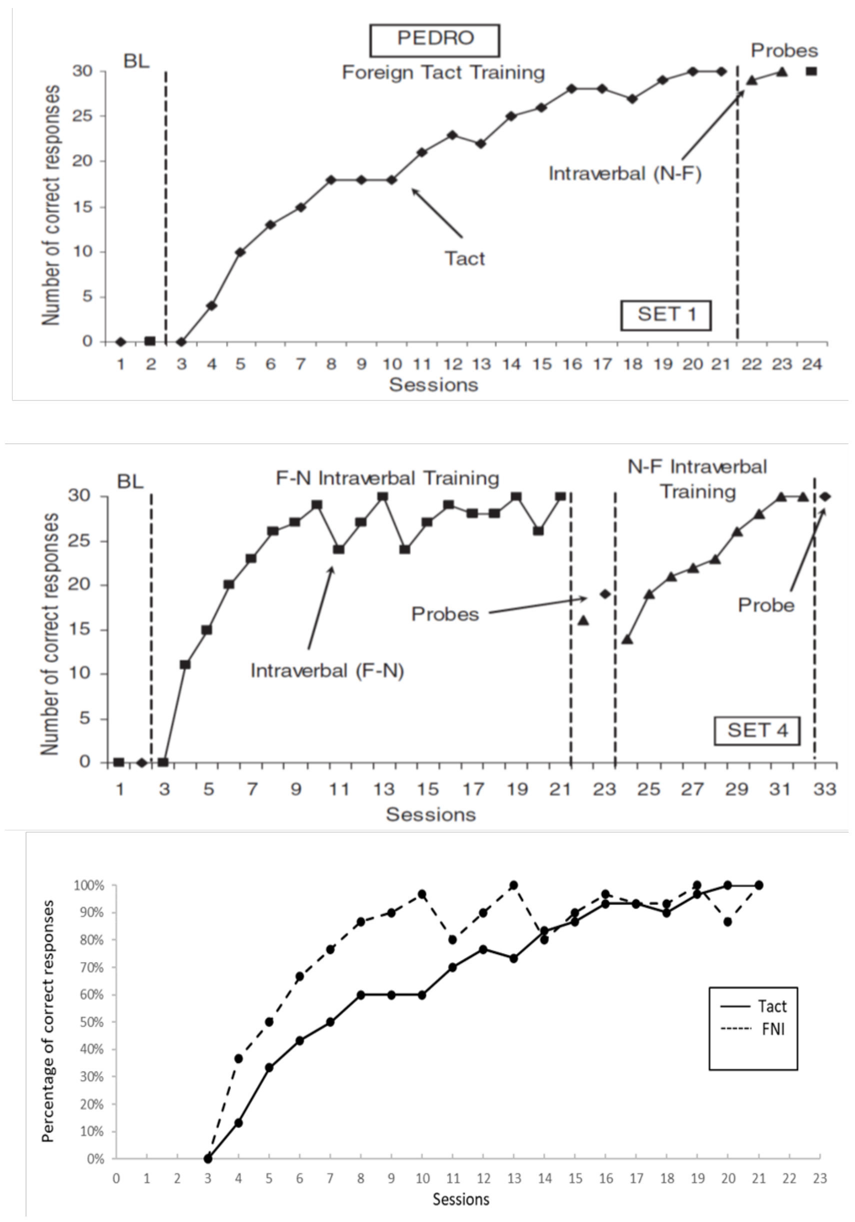


Figure 1. Example graph panels (top 2 panels) with differing x-axis scales (reprinted, with permission, from Dounavi, 2014 © John Wiley and Sons) and synthesized graph (bottom panel) containing both the Foreign Tact and Foreign-Native Intraverbal training data extracted from Dounavi (2014; p. 168).

For example, Wooderson et al. (2022) recently conducted a meta-analysis comparing foreign language vocabulary training procedures across seven studies and 23 learners. One of the study's aims was to compare the effectiveness of several verbal operant training procedures. In addition to statistical measures, the researchers employed descriptive visual analysis by extracting data from the training phases of each study and graphing the data on standardized panels. Figure 1 shows graphs of two training phases included in the systematic review and a synthesized graph showing both phases in one panel. Even though the top two graphs were from the same study (Dounavi, 2014; p. 168), their *x*-axes used different scales and were difficult to compare using visual inspection. We found it easier to compare the data after plotting them on one graph (bottom panel, figure 1).

Although it would have been preferable to regraph the studies' raw data, Wooderson et al.

(2022) could access the data directly from the authors of four of the seven papers, and could not obtain any data for three studies. Unfortunately, requesting data from researchers often does not work, as it may not be possible to contact them, they do not respond to requests, or advise that they no longer have access to the datasets (Van Der Zee & Reich, 2018). Although the advent of the Open Science movement and the capabilities afforded by digital technologies create the potential for greater access to empirical data, uptake by researchers is limited (Robson et al., 2021). Moreover, there is abundant basic and applied behavioral data not stored on digital repositories but published in the extant SCED literature.

In addition to the example above, there are other reasons why practitioners may want to extract and reanalyze published research data. First, engaging with and conducting research allows practitioners to approach problems in ways in which they have yet to be directly trained. Sidman (2011) considered the development of practitioners into scientist-practitioners a critical imperative within the field of behavior analysis. He encouraged practitioners to engage in basic and translational research because it improves their practice and offers a "whole new slant on behavior analysis" (Sidman, 2011, p. 976). Meta-analyses can be useful tools for practitioners to pose their own research questions and assess the strength of evidence for a given practice, particularly in the case of emerging or promising practices that have yet to be evaluated in the published literature. Second, essential criteria for behavioral interventions include the requirement that they are effective (Baer et al., 1968). Practitioners evaluating whether a given treatment intervention produced enough of a behavior change to be considered effective might compare their results with those in the published literature. Again, visual analysis is useful because it allows the practitioner to examine the data across interventions according to level, trend, and variability.

Difficulty accessing raw data is an issue for researchers and practitioners who want to reanalyze SCED data from the extant literature. Behavior-analytic literature employs graphical data presentations almost exclusively, and raw data or effect sizes are rarely presented within publications. In cases where raw data is inaccessible, data extraction software programs provide a valid and reliable method for digitizing graphed data sets from published SCED studies (Aydin & Yassikaya, 2022; Drevon et al., 2017; Flower et al., 2016; Rakap et al., 2016). These data are available for reanalysis or meta-analysis if one knows how to use tools like *DigitizeIt*™.

METHOD

This technical article demonstrates how to extract data from published SCED graphs using the data extraction software program. We recognize that readers may experience difficulties when using these procedures, so we included a simple example for readers to follow. However, the supplemental material also includes a demonstration video (https://youtu.be/3XVkYUEWxkY) and troubleshooting guide for more complex situations (e.g., plots with multiple *y*-axes). The basic procedure, however, is similar for most extraction programs and includes five key steps (Moeyaert et al., 2016; Rakap et al., 2016):

1. Import the graph into the program
2. Define XY axes
3. Plot data points
4. Create datasets
5. Export data

The task analyses below were performed using a registered copy of *DigitizeIt*™ (Version 2.5.3; Bormann, 2020). We selected *DigitizeIt*™ for this demonstration because it is the only program we could identify that includes the capability to automatically plot data points based on their shape (i.e., symbols). Thus, *DigitizeIt*™ potentially significantly reduces the time required to extract data from SCED graphs. We tested other software programs (Biosoft, 2004; Geomatix, 2021; Rohatgi, 2020; Tummers, 2015) that include automated extraction tools, but they are limited to tracing lines or curves rather than matching symbols.

DigitizeIt™ was downloaded from http://www.digitizeit.xyz/ and installed on a desktop computer running Windows 10. At the time of writing, the unregistered version was available to download and use for evaluation purposes for up to 21 days. The reader should install *Adobe Acrobat*™ on their computer if following the steps that describe importing graphs. We used the free version, *Adobe*

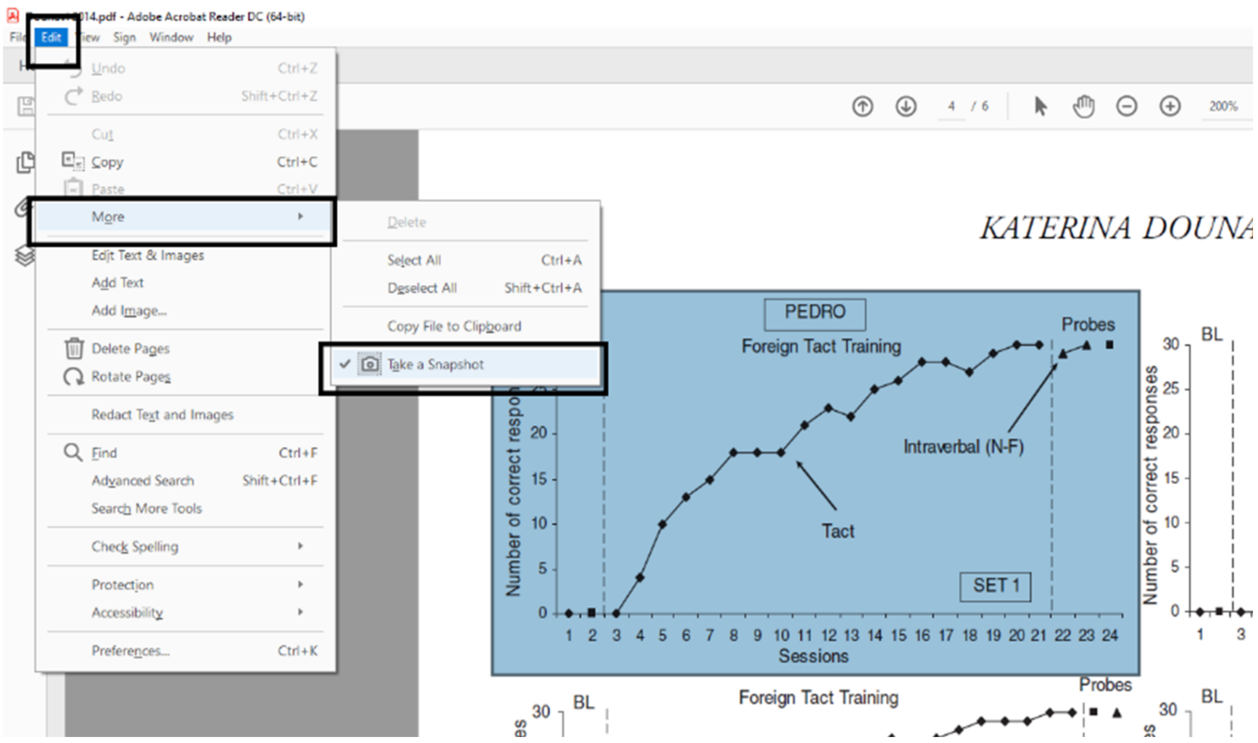


Figure 2. Copying graph images directly from pdf files using the SNAPSHOT tool in Adobe.

Acrobat™ Reader DC, downloaded from https://get.adobe.com/reader/. We also used the online version of *Google Sheets*™ (https://docs.google.com/spreadsheets/) for the final set of steps regarding exporting data. *Google Sheets*™ is free to use but requires a Google account, which is free to create. The reader may also download or access all three programs following Google searches using the keywords 'DigitizeIt', 'Adobe Acrobat Reader', and 'Google Sheets'.

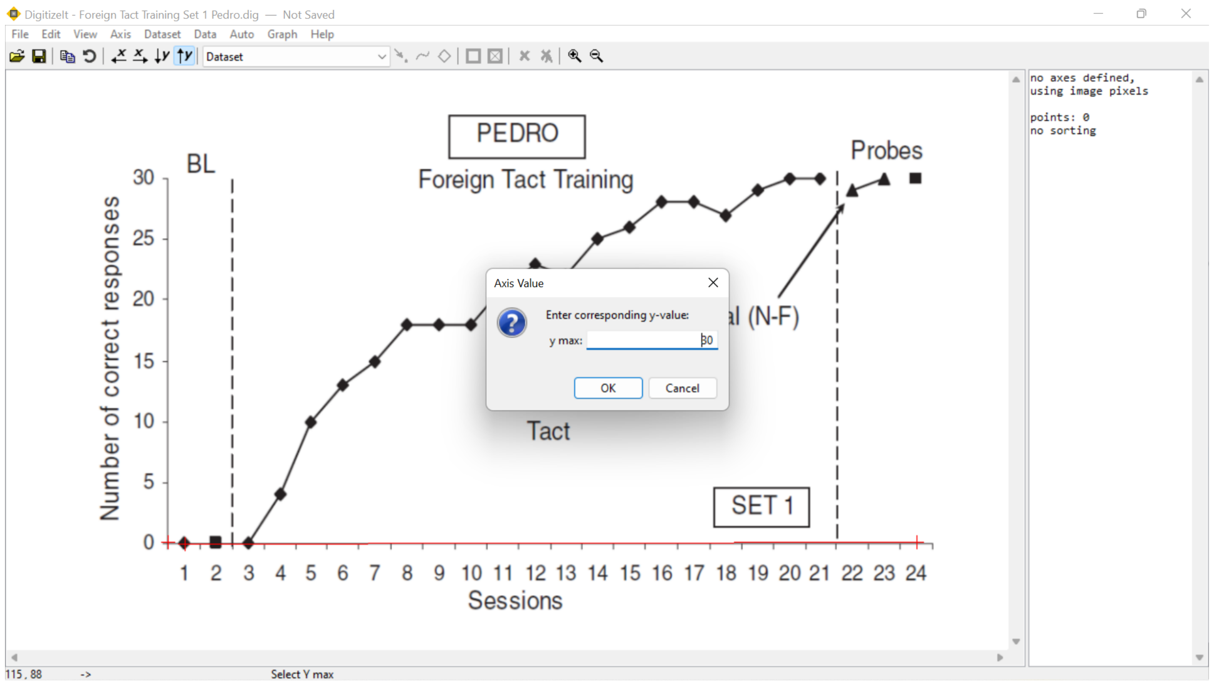
Importing the graph into the program

First, the reader should copy the graph image and import it into *DigitizeIt*™. Adobe Acrobat's *snapshot tool* provides a convenient method for copying graph images directly from .pdf files.

1. Open the research article in *Adobe Acrobat*™ and navigate to the page that displays the graph panel from which you intend to extract the data. Our example uses the top panel in Figure 1 – 'Foreign Tact Training' (Dounavi, 2014; p. 168).
2. Next, click on the EDIT menu, select MORE, then TAKE A SNAPSHOT. Position the cursor at the top-left corner of the graph's image, then press and hold the left mouse button while dragging the cursor to highlight a bounding box around the graph (Figure 2). Ensure that the selection window includes all necessary information, including the x- and y-axes and the graph's key if it has one. After releasing the left mouse button, a dialogue box should open and state that *the selected area has been copied*. At this point, you should click OK, then open and maximize the *DigitizeIt*™ program.
3. From within *DigitizeIt*™, select EDIT, then PASTE GRAPH to import the graph into the workspace.

Defining the XY axes

The following task analysis describes the steps required to calibrate and align the coordinate system with the imported graph's axes. The program requires four coordinates to define the XY axes – x min, x max, y min, and y max. The reader must complete these steps accurately before moving on to plotting data points.


DigitizeIt - Foreign Tact Training Set 1 Pedro.dig — Not Saved
File Edit View Axis Dataset Data Auto Graph Help
Dataset
PEDRO
Foreign Tact Training
BL
Probes
Number of correct responses
Sessions
SET 1
Tact
l (N-F)
Axis Value
Enter corresponding y-value:
y max:
30
OK
Cancel
115 , 88
Select Y max

```
no axes defined,
using image pixels

points: 0
no sorting
```

Figure 3. Defining the XY axes in *DigitizeIt*™.

1. Select AXIS, then X MIN, and you should see the cursor change to a crosshair.
2. Position the crosshair at the lowest labelled point on the graph's x-axis, click the left mouse button, and enter the corresponding x-axis value into the *Axis value* dialogue box that appears. If following our example, click on the center of the first data point and enter 1 as the *x min* value. After you click on OK, a red crosshair should then appear, marking the *x min* position.
3. Now select AXIS, then X MAX, click on the highest labelled point on the x-axis and enter its value into the *Axis value* dialogue box. In our example, you should click on the midpoint between the final two tick marks along the x-axis and enter 24 as the *x max* value. A horizontal red line will appear along the x-axis connecting the first and second crosshairs after clicking on OK.
4. Repeat the previous two steps by selecting AXIS, followed by Y MIN and then Y MAX to enter the *y min* and *y max* values accordingly. Following along with our example, position and set the *y min* value at 0 (i.e., the origin) and the *y max* value at 30 (i.e., the top of the y-axis; Figure 3) along the y-axis. Once this is done, a vertical red line connecting the *y min* and *y max* points should be visible.

Plotting data points and creating datasets

A powerful feature of *DigitizeIt*™ is the ability to automatically match and digitize symbols within graphs. This feature sets *DigitizeIt*™ apart from other data extraction programs that lack this capability. Automatic digitization (i.e., data plotting) can reduce the effort and time needed to extract data from SCED graphs; however, *DigitizeIt*™ sometimes fails to identify and match all symbols. In such cases, *DigitizeIt*™ also includes the capability to plot data points manually. The task analysis below describes both procedures, including steps for adjusting settings and defining search regions to improve automatic digitization.

Automatic data plotting

1. If your graph includes more than one phase or condition, you may choose to restrict *automatic digitization* to a specified region of the graph by defining a *search region*. To do this, click on AUTO and select SEARCH

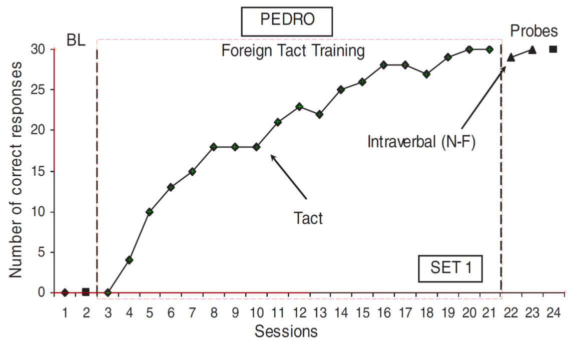


Figure 4. Screenshot from *DigitizeIt*™ showing the search region for the foreign tact training dataset and automatically digitized data points (Reprinted, with permission, from Dounavi, 2014 © John Wiley and Sons).

REGION. Then click and drag a rectangle with the mouse around the data points you intend to include in the search region. Try to exclude any unwanted data points or text. Following our example below, create a search region around the middle phase containing sessions 3 – 21 (i.e., 'Foreign Tact Training phase'). If you wish to include the whole graph, skip this step and proceed to step 2.

2. To automatically digitize the data points, click on AUTO, then select FIND SYMBOLS, and click on one of the symbols that you want to digitize within the graph panel. In the case of our example, click on one of the Tact symbols (i.e., filled black diamond) from within the search region. Now *DigitizeIt*™ will try to find all similar symbols and put them into a new *dataset*. If successful, you will see a green crosshair positioned at each symbol's center (Figure 4). You may also see a popup "New User Tip" asking if you would like to "Change symbol finder parameters"; dismiss these tips as they pop up.

If *DigitizeIt*™ fails to match all the symbols correctly, try adjusting the similarity settings in the *automatic digitizing* dialogue box. Before doing this, clear the existing data points by selecting DATASET, then DELETE to prevent the program from digitizing the same data points twice. Then, open the AUTO menu and click on OPTIONS to access the *automatic digitizing* settings. You can adjust the similarity

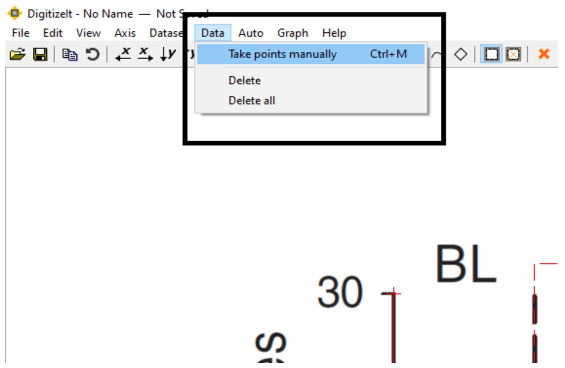


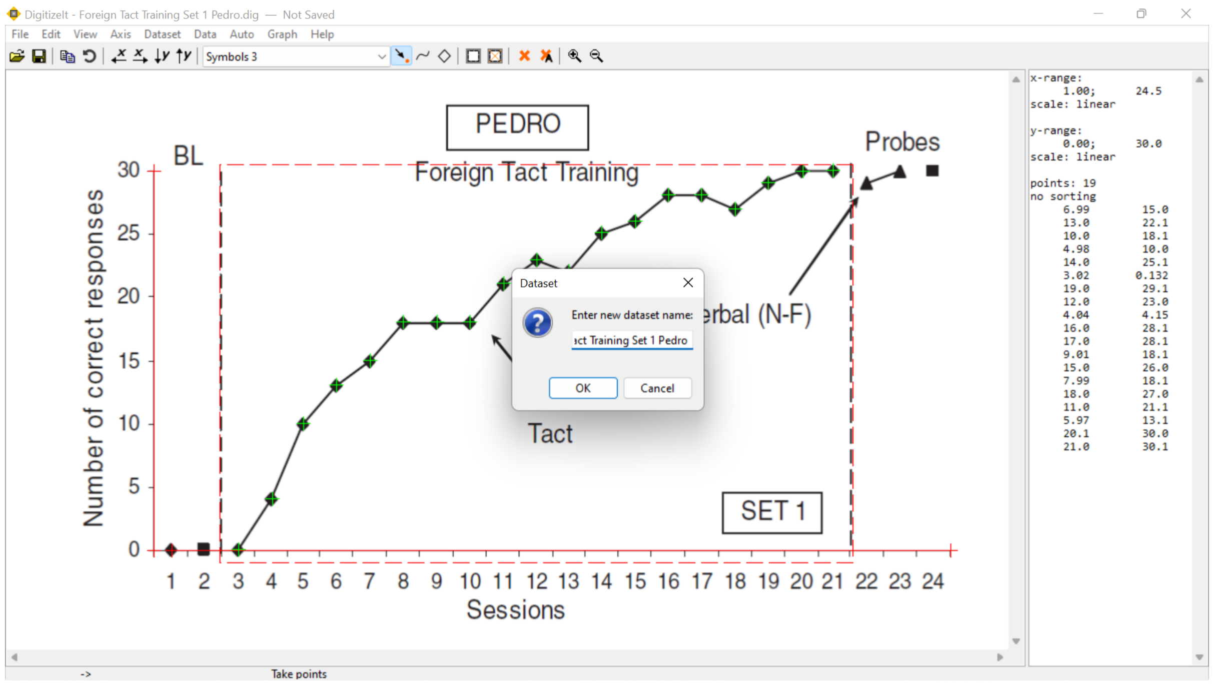
Figure 5. Selecting the TAKE POINTS MANUALLY and DELETE points tools in *DigitizeIt*™.

settings by moving the SYMBOL MATCHING IN % slider control up or down. This setting's value determines how well the selected symbol must match the one you want to digitize to be considered the same. If *DigitizeIt*™ does not find all the data points you want to digitize, try a lower value. If it finds too many (i.e., other symbols or text), try a higher value. After adjusting the slider, try clicking on one of the symbols again. You may need to make several adjustments; remember to clear the existing data points before each attempt.

In some cases, *DigitizeIt*™ appears to have difficulty detecting symbols if the graph's image is too uniform. The software may perform better with 'noisy' images containing slight variations between the symbols. If you have attempted all the above adjustments and *DigitizeIt*™ does not find any matched symbols, try importing a screenshot of the graph with a lower image resolution. One way to achieve this is by zooming out in *Adobe Acrobat*™ before taking a snapshot and importing it into *DigitizeIt*™.

Manual data plotting

1. You can add data points manually if *automatic digitization* fails to find all of them. To do this, click on DATA, and then TAKE POINTS MANUALLY (Figure 5).
2. Then, click on the center of each data point that you want to add to the dataset.
3. To remove unwanted data points, click on DATA and select DELETE. Then, click on each data point you want to delete from the dataset.



```
x-range:
     1.00;      24.5
scale: linear

y-range:
     0.00;      30.0
scale: linear

points: 19
no sorting
     6.99        15.0
     13.0        22.1
     10.0        18.1
     4.98        10.0
     14.0        25.1
     3.02       0.132
     19.0        29.1
     12.0        23.0
     4.04        4.15
     16.0        28.1
     17.0        28.1
     9.01        18.1
     15.0        26.0
     7.99        18.1
     18.0        27.0
     11.0        21.1
     5.97        13.1
     20.1        30.0
     21.0        30.1
```

Figure 6. Naming a dataset using the RENAME tool in *DigitizeIt*™.

Creating datasets

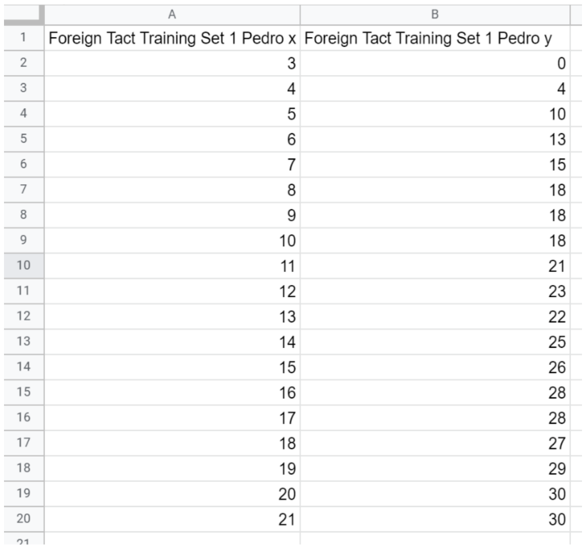
1. After digitizing, *rename* the *dataset*, so it is easy to identify later when exporting the data. To *rename* a *dataset*, click on DATASET, then RENAME, type the name into the *dataset* window that pops up, and click on OK. In our example, we named the first *dataset* "Foreign Tact Training Set 1 Pedro" (Figure 6).
2. Repeat the above steps, digitizing and creating datasets as necessary for each dataset you intend to digitize.
3. To switch between *datasets* in *DigitizeIt*™, click on the dropdown menu on the command ribbon (located to the right of the SET Y MAX button (i.e., the upward arrow and a "y"). At this point, any unwanted datasets should be deleted. For our example, switch to the empty *dataset* named Dataset, then click on DATASET, then DELETE.

Exporting data

The final set of steps below describes how to export digitized data out of the program. The registered version of *DigitizeIt*™ supports export to text via the clipboard or .csv (comma-separated values) file, which can be used with most spreadsheet programs, including *Microsoft Excel*™ or *Google Sheets*™. The following steps demonstrate to the reader how to export the data to a .csv file. Note that the unregistered version of *DigitizeIt*™ does not support data exportation.

Exporting to .csv

1. By default, the digitized data are unsorted. Before exporting, arrange the data, smallest to largest, based on the data points' x-axis values. To do this, select DATASET, then SORT, and ASCENDING.
2. Then select FILE, then EXPORT ALL AS CSV. In the SAVE AS window that opens, enter a file name, choose a location to save the file, and click on SAVE.
3. Next, open *Google Sheets*™ and click on + to create a new blank spreadsheet. Then, open the .csv file you created in *DigitizeIt*™ by clicking on FILE, selecting OPEN, and UPLOAD. In the UPLOAD window, click on SELECT A FILE FROM YOUR DEVICE, and locate the .csv file.



	A	B
1	Foreign Tact Training Set 1 Pedro x	Foreign Tact Training Set 1 Pedro y
2	3	0
3	4	4
4	5	10
5	6	13
6	7	15
7	8	18
8	9	18
9	10	18
10	11	21
11	12	23
12	13	22
13	14	25
14	15	26
15	16	28
16	17	28
17	18	27
18	19	29
19	20	30
20	21	30

Figure 7. Sample showing extracted data points (x and y values) formatted as rounded whole numbers in *Google Sheets™*.

4. After opening the file, you will see that the data appear in scientific notation format. For ease of use, change the format in *Google Sheets™* from scientific notation to rounded whole numbers (Figure 7). First, click and drag the mouse button to select all the numerical data. Then, click on FORMAT, then NUMBER, and select NUMBER once again. With the numerical data still highlighted, click twice on the DECREASE DECIMAL PLACES button located on the command ribbon to round the data to whole numbers.

CONCLUSION

This paper demonstrated how to extract graphical data from published SCED articles using the *DigitizeIt™* software program (Version 2.5.3; Bormann, 2020). We acknowledge that several other data extraction programs are available to the reader to perform these tasks; however, we believe *DigitizeIt™* to be the most efficient due to its automatic digitizing tools. After extracting the data, the reader may conduct statistical analyses or graph and reanalyze it using their preferred graphing software. Readers are encouraged to apply the above procedures to examine their own practice or research questions through reanalysis or metanalysis of empirical data from the research literature.

REFERENCES

Aydin, O., & Yassikaya, M. Y.. (2022). Validity and reliability analysis of the Plotdigitizer software program for data extraction from single-case graphs. *Perspectives on Behavior Science*, *45*(1), 239–257. https://doi.org/10.1007/s40614-021-00284-0

Baer, D. M., Wolf, M. M., & Risley, T. R. (1968). Some current dimensions of applied behavior analysis. *Journal Of Applied Behavior Analysis*, *1*(1), 91–97. https://doi.org/10.1901/jaba.1968.1-91

Biosoft. (2004). Ungraph (Version 5.0.1). Retrieved from: http://www.biosoft.com/w/ungraph.htm

Bormann, I. (2020). DigitizeIt (Version 2.5.3): Bormisoft. Retrieved from: http://www.digitizeit.xyz/

Dounavi, K. (2014). Tact training versus bidirectional intraverbal training in teaching a foreign language. *Journal of Applied Behavior Analysis*, *47*(1), 165-170. https://doi.org/10.1002/jaba.86

Drevon, D., Fursa, S. R., & Malcolm, A. L. (2017). Intercoder reliability and validity of WebPlotDigitizer in extracting graphed data. *Behavior Modification*, *41*(2), 323-339. https://doi.org/10.1177/0145445516673998

Flower, A., Mckenna, J. W., & Upreti, G. (2016). Validity and reliability of GraphClick and DataThief III for data extraction. *Behavior Modification*, *40*(3), 396-413. https://doi.org/10.1177/0145445515616105

Geomatix. (2021). XYit (Version 3.1.10). Retrieved from: https://www.geomatix.net/xyit/

Horner, R. H., & Swoboda, C. M. (2014). Visual analysis of single-case intervention research: Conceptual and methodological issues. In T. R. Kratochwill & J. R. Levin (Eds.), *Single-case intervention research: Methodological and statistical advances*. (pp. 91-125). American Psychological Association.

Johnston, J. M., & Pennypacker, H. S. (2010). *Strategies and tactics of behavioral research.* Taylor & Francis.

Kazdin, A. E. (2021). Single-case experimental designs: Characteristics, changes, and challenges. *Journal of the Experimental Analysis of Behavior*, *115*(1), 56-85. https://doi.org/10.1002/jeab.638

Kratochwill, T. R., Hitchcock, J. H., Horner, R. H., Levin, J. R., Odom, S. L., Rindskopf, D. M., & Shadish, W. R. (2013). Single-case intervention research design standards. *Remedial and Special Education*, *34*(1), 26-38. https://doi.org/10.1177/0741932512452794

Ledford, J. R., Lane, J. D., & Severini, K. E. (2018). Systematic use of visual analysis for assessing outcomes in single case design studies. *Brain Impairment, 19*(1), 4-17. https://doi.org/10.1017/brimp.2017.16

Lobo, M. A., Moeyaert, M., Baraldi Cunha, A., & Babik, I. (2017). Single-case design, analysis, and quality assessment for intervention research. *Journal of Neurologic Physical Therapy*, *41*(3), 187-197. https://doi.org/10.1097/npt.0000000000000187

Maggin, D. M., Lane, K. L., & Pustejovsky, J. E. (2017). Introduction to the special issue on single-case systematic reviews and meta-analyses. *Remedial and Special Education*, *38*(6), 323-330. https://doi.org/10.1177/0741932517717043

Moeyaert, M., Maggin, D., & Verkuilen, J. (2016). Reliability, validity, and usability of data extraction programs for single-case research designs. *Behavior Modification*, *40*(6), 874-900. https://doi.org/10.1177/0145445516645763

Rakap, S., Rakap, S., Evran, D., & Cig, O. (2016). Comparative evaluation of the reliability and validity of three data extraction programs: UnGraph, GraphClick, and DigitizeIt. *Computers in Human Behavior*, *55*, 159-166. https://doi.org/10.1016/j.chb.2015.09.008

Robson, S. G., Baum, M. A., Beaudry, J. L., Beitner, J., Brohmer, H., Chin, J., . . . Thomas, A. (2021). Promoting Open Science: A holistic approach to changing behavior. Retrieved from: https://doi.org/10.31234/osf.io/zn7vt

Rohatgi, A. (2020). Webplotdigitizer (Version 4.4). Retrieved from: https://automeris.io/WebPlotDigitizer

Sidman, M. (2011). Can an understanding of basic research facilitate the effectiveness of practitioners? Reflections and personal perspectives. *Journal of Applied Behavior Analysis*, *44*, 973-991. https://doi.org/10.1901/jaba.2011.44-973

Tummers, B. (2015). DataThief III (Version 1.7). Retrieved from: https://datathief.org/

Van Der Zee, T., & Reich, J. (2018). Open Education Science. *AERA Open*, *4*(3), 233285841878746. https://doi.org/10.1177/2332858418787466

Wolfe, K., Barton, E. E., & Meadan, H. (2019). Systematic protocols for the visual analysis of single-case research data. *Behavior Analysis in Practice*, *12*(2), 491-502. https://doi.org/10.1007/s40617-019-00336-7

Wooderson, J. R., Bizo, L. A., & Young, K. (2022). A systematic review of emergent learning outcomes produced by foreign language tact training. *The Analysis of Verbal Behavior*, *38*(2), 157–178. https://doi.org/10.1007/s40616-022-00170-z

ESTABLISHING A SURROGATE CONDITIONED MOTIVATING OPERATION EFFECT WITHOUT (UNCONDITIONED) MOTIVATING OPERATIONS: A PILOT INVESTIGATION

Cassidy Lehrke[1], Benjamin N. Witts[1]

[1] ST. CLOUD STATE UNIVERSITY

The surrogate conditioned motivating operation (CMO-S) is not extensively studied and therefore lacks a wide empirical base. We sought to test whether CMO-S effects could be produced when no unconditioned motivation operation (UMO) was explicitly programmed. Four undergraduate students played a dot-clicking game on a computer. Game-related stimuli (background color or sound) changed throughout each session, which coincided with changes to earned points for dot clicking (a distractor variable). During training sessions, some stimulus changes were reliably correlated with particular edible deliveries and consumption. Pre-training, mid-training, and post-training probe sessions tested for general (any edible) and specific (particular edibles coordinated with particular stimuli) CMO-S effects when stimuli were presented without programmed UMOs. Two of the four participants provided evidence of CMO-S effects, while the other two did not. Limitations around interfering motivating operations and future directions (e.g., preparedness) are discussed.

Keywords: motivation; behavior analysis; simulation

Successful demonstrations and replications that establish the surrogate conditioned motivating operation (CMO-S) are minimal in both the applied and basic literatures (e.g., Adelinis et al., 1997; Calvin et al., 1953; Lanovaz et al., 2014; McDiffett, 2019; McGill, 1999; Ormandy, 2018). A common definition of a CMO-S effect requires an unconditioned motivating operation (UMO) be paired with a neutral stimulus (NS), resulting in a relation where the once NS will influence behavior similarly or identically to the UMO's effect (Ormandy, 2018). The prototypical example of this concept is eating lunch at noon because noon (i.e., the NS) and eating (i.e., the UMO of hunger) are historically paired. In an applied example, Lanovaz et al. (2014) paired colored poster boards (NS) with items known to evoke stereotypy (UMO). After pairing, the presence of the posterboards alone increased stereotypy, in comparison to baseline. Like its umbrella concept, the motivating operation (MO), CMO-S effects are measured in two ways. One being value-altering effects that are determined by rate of acquisition and the other behavior-altering effects determined by a relative increase in behavior historically related to the CMO-S (see Malott, 2007).

Failure to produce a CMO-S effect might be less dependent on the NS or competing stimuli, but on whether the MO occurred variably or at all in their presence. Without testing for behavior and value-altering effects, one cannot ensure pairing actually occurred and thus it is impossible to rule out MO presence as a confound. However, it is currently unclear how researchers could effectively test for MO relations during pairing sessions without disrupting the procedure or what behaviors satisfy the MO. MOs are transient (Ormandy, 2018) and can be satisfied by myriad responses. It might be beneficial to consider response classes over individual responses unless the study permits finer-grained analyses. For example, when cold, putting on a sweater, turning up the thermostat, or closing a window can satisfy the MO. Metabolic processes also accomplish this, but might be undetectable.

In designing a study to establish the CMO-S effect, the first step is to identify a NS (i.e., produces no UMO effect) and a UMO (or perhaps just an MO) to pair. Some UMOs might work better than others, though what UMO-NS pairings make for more efficient conditions are not yet documented. For example, for some species, food deprivation or satiation might

Author Note: Address correspondence to Benjamin N. Witts (ORCID ID:0000-0002-5554-3147). Email: bnwitts@stcloudstate.edu

This paper is based on a thesis by the first author in partial fulfillment of a MS in ABA.

https://doi.org/10.17605/OSF.IO/XCMHE

take long periods to establish (Ormandy, 2018; see McDiffett, 2019). Similarly, each human has different metabolic processes and thus two individuals can consume the same amount of food, yet one will be full, and one will not. Second, testing requires a NS presentation in the absence of the UMO. A CMO-S is said to develop when tests show increased UMO-related behaviors above pre-experimental levels. Due to limited successful demonstrations, it is unclear how the pairing procedure between the UMO and NS should be arranged (e.g., simultaneously [Ormandy, 2018], sequentially [Lanovaz et al., 2014]). Other considerations, such as time between and number of pairings, also lack empirical basis. Much of the confusion of the CMO-S concept might be owed to the lack of clarity over essential conditions.

With such ambiguity, we might question the necessary role of UMOs in the development of a CMO-S effect; does any response related to any MO lead to the same outcome? To test the assumption that the UMO might not need to be present, we conducted the following study in which probes provided free operant access to a UMO-related stimulus (i.e., food) that had been paired with a stimulus event (i.e., sound or color) with no programmed UMO for food consumption. If multiple stimuli are individually paired with certain edibles and other stimuli are paired with no edibles to serve as control stimuli, then a strong argument for a CMO-S effect can be made if the results of the analyses show an increased probability of specific edible consumption when its paired stimulus is presented. If general food consumption is higher in the presence, but not the absence, of these stimuli after pairing, a moderate case for a CMO-S effect can be made.

METHOD

Participants

Four undergraduates with no identified sensory impairments from a mid-sized Midwestern university participated. Course credit was offered for participating and all students consented to data sharing. Participants will be referred to as P1, P2, P3, and P4.

Setting and Materials

Sessions were held in a 9′ x 19.5′ office. Participants were seated in front of a computer monitor at a desk facing a blank wall. The researcher sat at a desk behind the participants. Each session was recorded by a hidden camera located on top of a cabinet situated between the participant and researcher; video footage was reviewed for IOA and procedural integrity methods.

Various necessary items (e.g., computer, mouse, edibles, plates) were included. Six PsychoPy3 (Peirce et al., 2019) computer programmed games, referred to as G1-S (i.e., Game 1, Sound), G2-S, G3-S, G1-C (i.e., Game 1, Color), G2-C, G3-C were programmed to randomly present either three supplemental sounds (i.e., S1, S2, S3) or three alternative colors (i.e., C1, C2, C3). The computer used was set to the same volume for all trials.

PsychoPy3 Game Details

A white circle, 1/20th the height of the monitor's size, moved around the screen when clicked. Points were earned for each click on the circle and appeared at the top of the screen. Clicks were worth one point during intervals with the default color/sound, and worth 3, 4, or 5 points during C1/S1, C2/S2, or C3/S3 intervals, respectively. Circle clicks and point values served only to provide participants the opportunity to invent a reason for the color or sound changes or edible delivery, which the researcher alluded to in session instructions.

For sound games (i.e., G1-S, G2-S, and G3-S), the screen remained gray throughout play. A repetitious instrumental jazz-like soundtrack (i.e., default sound) played throughout. Fifteen s sound clips were used; S1 was bongo drums, S2 was 'cosmic bubbles', and S3 consisted of 'industrial sounds' (e.g., machines working).

Color games had no programmed audio; rather, the screen remained gray (i.e., default color) until an alternative color replaced the default color for 15 s. Alternative colors were assigned as follows: C1 was green, C2 was blue, and C3 was orange.

A random number generator determined when each stimulus change occurred, with two caveats: (1) Changes could not occur during the first and last 15 s of the game and (2) at least 15 s transpired between each presentation. Stimuli and default changes were timed identically across sound and color games for G1, G2, and G3.

Design

This study used a multiple probe design. Conditioning sessions occurred between probes.

Procedure

Participant Screening. Participants started by completing allergy and food restriction screening, as well as three preference assessments of edible items. Three different classes of edibles (i.e., chocolate, candy, salty) were presented as a list of twenty edible options. Participants divided edibles into two categories: those they would eat and those they would not, for each of the three classes. The "would eat" pile was then subdivided in three: most preferred (two edibles max), least preferred (two edibles max), and the remaining into a "moderately preferred" pile ranked from most to least. The three middle-most ranked edibles were chosen for each participant as they were deemed the most likely to be neutral. Edibles were randomly assigned as E1, E2, and E3 for each participant.

Participant Assignments. Participants were randomly assigned to either color or sound games; P1 and P4 had color games, P2 and P3 had sound. Each participant had pairings of E1 to C/S1 and E2 to C/S2. E3 and C/S3 were never presented together or with other stimuli as they served as controls. The order of the three games was assigned to each session using a random number generator, and that order was shared across all participants.

General Procedure. Sessions 1, 5, and 10 were probes and sessions 2-4 and 6-9 were conditioning. Participants had no restrictions on their food or water consumption prior to sessions and each session lasted about 20 minutes. The primary researcher presented instructions, collected data, and, on conditioning sessions, delivered edibles.

Probes. During probes, participants had access to water and three plates of four edibles each, ordered E1, E2, and E3. The researcher watched from the live video footage of the participant and recorded when and what edibles were chosen. Choice was defined as any part of the participant's hand contacting the edible, followed by the edible's removal from the plate. Before beginning, the researcher presented the following instructions:

> "You will have 15 minutes to play a game. Help yourself to the snacks provided. I will let you know when the time is up. If you need more water or want to withdraw, please let me know, but otherwise refrain from asking any questions. Please keep your mask up at all times and only lower it when eating or drinking. Do not touch anything else in the room, other than the snacks, water, and your mouse."

Conditioning. During conditioning sessions, participants had free access to water and their computer mouse and the researcher presented edibles according to their programmed time. Twelve edibles were delivered (i.e., 4 of each 3 types) to ensure the number of edibles were constant across training and probe trials. During the instructions, participants were told that edibles were delivered when they met a predetermined goal (i.e., a deception) and to consume edibles as soon as they were delivered. The instructions were read as follows:

> "You will have fifteen minutes to play a game. While you work, you will be presented a food reward when you've met our predetermined goal. You will not be informed of what this goal is. When food is presented, pause your game, immediately eat the item, then resume working. I will let you know when the time is up. Do not touch anything else in the room, other than the snacks, water, and mouse."

If the game malfunctioned (e.g., the dot they must click on to gain points disappeared), the researcher recorded the time, and instructed the participant to take an intermission away from the game. The researcher then loaded the next game to play for the remainder of the fifteen min session. This scenario occurred once for P4 only.

Dependent Variables and Measurement. Data were only collected during probe sessions, given that participants had no opportunity to independently select edibles (i.e., the DV) during conditioning trials. and consisted of recording the time each edible was chosen. Data were analyzed on two levels: stimulus class (i.e., any edible) and individual stimulus (i.e., particular edibles), and both in terms of overlapping with a stimulus change.

Procedural Integrity and Interobserver Agreement. A random number generator was used to determine which sessions a second observer would take interobserver agreement (IOA; 50% of probe sessions) and procedural integrity (35% of all sessions) data across all participants. Both IOA and procedural integrity for probe sessions were completed via video recordings after all participants had given consent for their videos to be reviewed.

Procedural integrity as scored by the secondary researcher was 169/170 or 99.41%. Due to video recording limitations, some items could not be verified (e.g., door being closed, personal devices being turned off). For IOA, if both researchers listed a time within 3 seconds of the other, or if both researchers listed an item as not selected during the session, an agreement was scored. IOA was 100%.

Post-Study Assessments. Participants conducted a sensory discrimination test, matched to their assignment (i.e., either sounds or colors). Two sounds/colors were presented sequentially, and the participant indicated if the two sounds were the same. Each sound/color used in the study was presented with itself and with each other sound/color at least once. This test was completed at the end of the study to decrease reactivity and priming effects. Both color-assignment participants (i.e., P1 and P4) scored 100%, whereas the two sound-assignment participants (i.e., P2 and P3) scored 10/12 and 7/12, respectively. These latter results could have altered the effectiveness of conditioning sessions.

Debrief and Exit Survey. Following the discrimination test, the researcher debriefed with each of the participants. Participants had been told during sessions they would receive edibles when they met a specific point goal; the researcher clarified there was no goal and edibles were presented according to predetermined times. Participants were told of the hidden camera and were given an opportunity to either delete their footage or give consent to this footage being used for research purposes. All four participants consented.

Finally, participants completed an exit survey to provide more information on their experience. While subjective, some responses suggested limitations to the study. For example, all participants reported choosing certain edibles due to preference, and P1 and P3 both claimed they knew stimulus changes and edibles were paired together. What is most notable is P3 claimed they typically do not eat at the time most of their sessions were ran, they were sick of the snack options, and were often more thirsty than hungry; this response suggests there were multiple competing MOs. Similarly, P2 claimed they were full and did not want to eat candy for two of their three probe sessions. This suggests another AO for snack consumption that could have altered responding.

RESULTS

Figure 1 depicts the analyses for P1 and P4; P2 and P3 are described only in text for clarification purposes (see supplemental files for copies of these graphs). Matches refer to a participant selecting an edible during the stimulus change event it was paired with during training (e.g., an E1 select during a C/S1 event). Overlaps refer to selections during non-paired stimulus change events (e.g., an E1 select during a C/S2 or C/S3 event). Bar graphs represent the timing and duration of stimulus changes. Circles, squares, and diamonds represent timing of edible selection anchored to the *x*-axis and order of edible selection anchored to the *y*-axis. Circles represent edible selection outside of events, squares represent edible selections that overlap any stimulus change event, and diamonds represent edible selections that match the stimulus change event it was paired with during training. Blue, green, and orange bars represent C1, C2, and C3 (control) presentations, respectively. Blue, green, and red shapes represent E1, E2, and E3 (control) selections, respectively.

P1 consumed 12 edibles in the first probe; of those twelve, one was overlapped and two were matched. After training, on the second probe, they consumed 12 edibles (4 overlapped; 0 matched). On the third probe, they consumed 12 edibles (3 overlapped; 1 matched). P4 consumed 10 edibles in the first probe (1 overlapped; 0 matched). They consumed 9 edibles in the second probe (0 overlapped; 0 matched) and 10 edibles in the third (2 overlapped; 2 matched [1 of each of the stimulus-edible pairings from training]). P2 and P3 experienced sound changes during training and probes. P2 consumed 2 edibles in the first probe, 4 in the second, and 2 in the third; no overlaps or matches occurred. P3 consumed 4 edibles in the first probe (1 overlapped; 0 matched), 6 edibles in the second probe (1

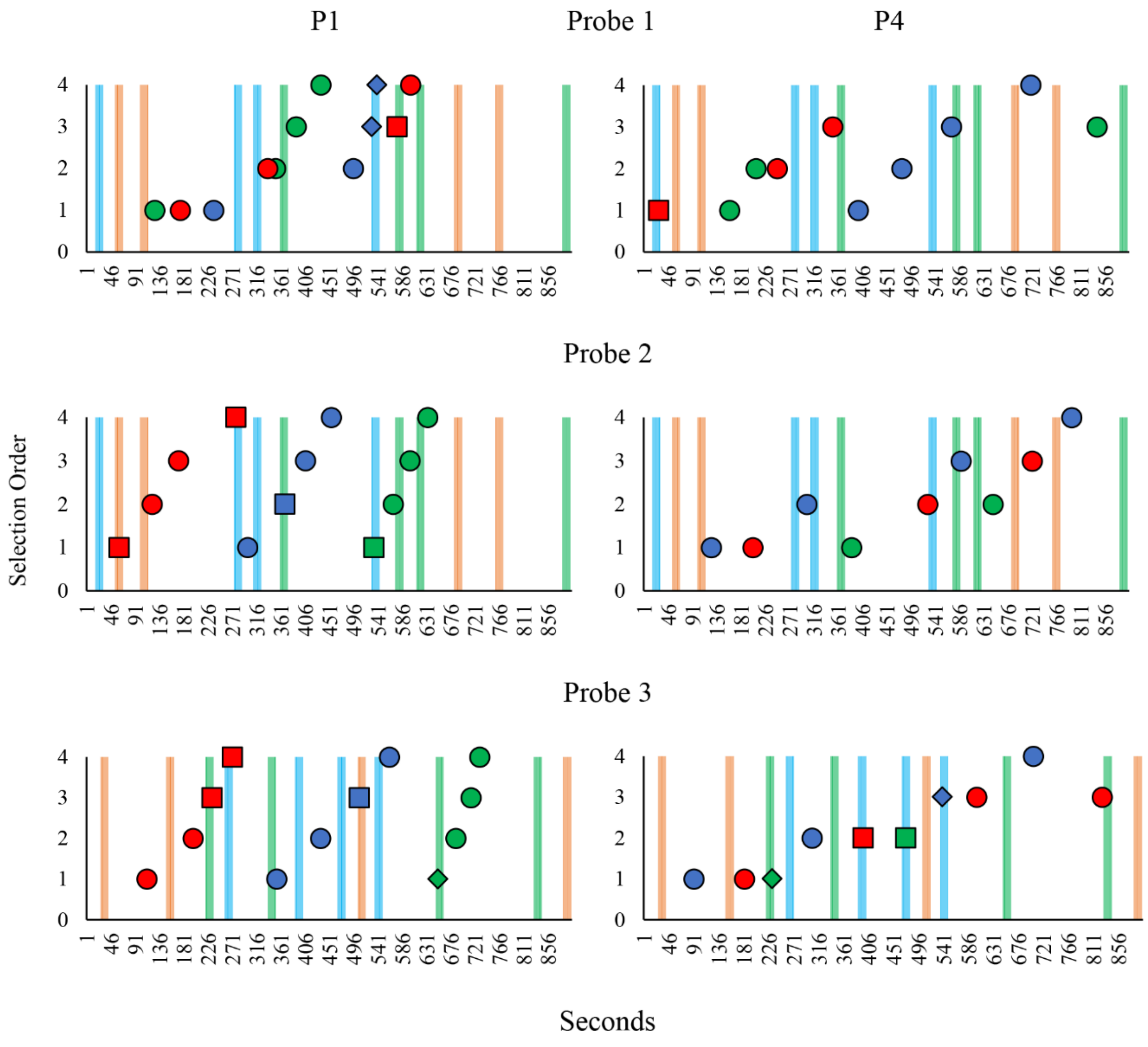


Figure 1. Stimulus change timing, edible selection timing, and selection order on probe sessions for P1 and P4.

overlapped; 1 matched) and 4 edibles in the third probe (0 overlapped).

DISCUSSION

P1 and P4's data suggest moderate evidence of a CMO-S. For these two participants, edible consumption in the presence of any alternative color during probes increased over the course of the study. While an increase was observed, neither participant reached consistent or high levels of overlapped or matched selections (e.g., 6-12) by the final probe, suggesting moderate, rather than strong, evidence of an observable effect. While both participants mentioned in their exit survey that they chose edibles in the order of their preferences, the timing of those choices is most important. As more training trials occurred, they made more selections during stimulus change events; suggesting edibles were more valuable at those times and a general CMO-S effect may have occurred. The development of at least a general CMO-S effect is further supported by noting both P1 and P4 scored 100% on their sensory discrimination tests, suggesting that pairing opportunities were salient and thus more likely to produce an effect during probe trials. Contrastingly, P2 and P3 both failed their sensory discrimination tests and did not produce an effect during probe trials, suggesting salient pairings are needed to produce an effect. Further, consider that P1 and P4 had color-changing sessions with salty snacks and P2 and P3 had chocolate snacks with sound-changing sessions. Perhaps particular combinations of stimuli and MOs might more readily be conditioned; a phenomenon known as preparedness (see Seligman, 1970). The idea of preparedness in CMO-S development has not yet been explored in the literature, but this area seems like a logical next step in the study of this MO subtype.

A few limitations are worth exploring. The exit survey results suggest the preference assessment was not successful in identifying equally neutral, or neither highly nor non-preferred edibles. For example, during probe sessions, P1 typically ate all of the E3, then E1, then E2, suggesting the presence of an interfering MO from the edibles themselves. Here, consuming E3 edibles might have blocked consumption of E1 edibles during C1 intervals. Additionally, competing MOs might have influenced participant responding. Consider that exit surveys suggested some participants were more thirsty than hungry or did not want the snacks during the session. Future research would do well to explore more effective methods of establishing neutral stimuli for pairing purposes, either by using better assessments (e.g., progressive ratio assessments) or by using arbitrary stimuli, perhaps tokens.

Points were worth more during supplemental or alternative stimulus conditions and edible consumption during these times might have interfered with the participant's ability to earn points. If this was indeed an interfering MO, it would be interesting, as point accumulation was meaningless; participants were told points were used to determine when edibles would be delivered during training (a deception), and edibles were provided in a free operant format during probes.

P2 and P3 both failed their sensory discrimination test. It is unclear why failures occurred, as the sounds are arguably distinct (see supplemental files for a sample of each sound). Speculatively, participants were not motivated to respond correctly during this task, instructions did not acknowledge they could ask for sounds to be repeated, and the interstimulus interval between sounds each could have contributed to the failed discrimination test.

Due to scheduling conflicts, P2 and P3 played the same game twice in the same day, each game separated by just a few minutes. Sessions occurring in rapid succession could increase the likelihood of the participant identifying the experimental manipulation (coordinating stimulus conditions with edibles). Second, habituation or satiation effects could interfere with the CMO-S procedure. Additionally, spacing out sessions could capitalize on MO effects; behavior altering effects are more likely to occur when an EO is in place, and the hungrier (or less habituated) organism will be more readily conditioned.

The results of this study suggest the CMO-S is worth pursuing; however, researchers may need to adapt different methods when designing their studies to create successful demonstrations and improved data analysis methods (e.g., conditional probability analyses to account for chance responding). For example, this study demonstrated an effect could occur despite no active creation of an MO, which differs from previous research on this concept. Researchers must control for AO effects, which were likely present in this study. Researchers should also consider preparedness to find the most effective combinations to create a specific or general CMO-S effect.

REFERENCES

Adelinis, J. D., Piazza, C. C., Fisher, W. W., & Hanley, G. P. (1997). The establishing effects of client location on self-injurious behavior. *Research in Developmental Disabilities*, *18*(5), 383-391. https://doi.org/10.1016/S0891-4222(97)00017-6

Calvin, J. S., Bicknell, E. A., & Sperling, D. S. (1953). Establishment of a conditioned drive based on the hunger drive. Journal of *Comparative and Physiological Psychology*, *46*(3), 173–175. https://doi.org/10.1037/h0060708

Lanovaz, M. J., Rapp, J. T., Long, E. S., Richling, S. M., & Carroll, R. A. (2014). Preliminary effects of conditioned establishing operations on stereotypy. *The Psychological Record*, *64*, 209-216. https://doi.org/10.1007/s40732-014-0027-x

Malott, R. W. (2007). Principles of behavior (6th ed.). Pearson.

McDiffett, C. (2019). Examination of the surrogate conditioned motivating operation to influence eating behavior in pigeons. [Unpublished master's thesis] California State University. http://scholarworks.csustan.edu/bitstream/handle/011235813/1468/mcdiffettcfall2019.pdf?sequence=1

McGill, P. (1999). Establishing operations: Implications for the assessment, treatment, and prevention of problem behavior. *Journal of Applied Behavior Analysis, 32*(3), 393-418. https://doi.org/10.1901/jaba.1999.32-393

Ormandy, S. (2018). An experimental demonstration of the surrogate conditioned motivating operation [Unpublished doctoral dissertation]. The Chicago School of Professional Psychology. https://pqdtopen.proquest.com/doc/2029154166.html?FMT=AI

Peirce, J. W., Gray, J. R., Simpson, S., MacAskill, M. R., Höchenberger, R., Sogo, H., Kastman, E., Lindeløv, J. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods.* https://doi.org/10.3758/s13428-018-01193-7

Seligman, M. E. (1970). On the generality of the laws of learning. *Psychological Review, 77*(5), 406-418. https://doi.org/10.1037/h0029790

Williams, C. (Writer), Whittingham, K. (Director). (2007, February 8). Phyllis' Wedding (Season 3, Episode 16) [TV series episode]. In G. Daniels, R. Gervais, & S. Merchant (Executive Producers), The Office. Reveille Productions; NBC Universal Television Studio.

TECHNICAL INFORMATION

README: THE CODE TO CREATE THE COVER & TOC GRAPHS

David J. Cox[1,2]

[1] ENDICOTT COLLEGE, [2] RETHINKFIRST

Visualizing behavioral data in unique ways may lead to novel methods of analyses and, perhaps, new ways of thinking about environment-behavior relations. In the spirit of transparency and to help others discover interesting things in their own textual data, below is the code to create the plots shown on the cover and table of contents in this volume. Some familiarity with Python is needed (i.e., how to open a script, read in files, and execute the program). A downloadable file of this Python script is also available here: https://osf.io/p8f7j. Otherwise, happy coding and playing.

Keywords: Python, visual analysis, natural language processing

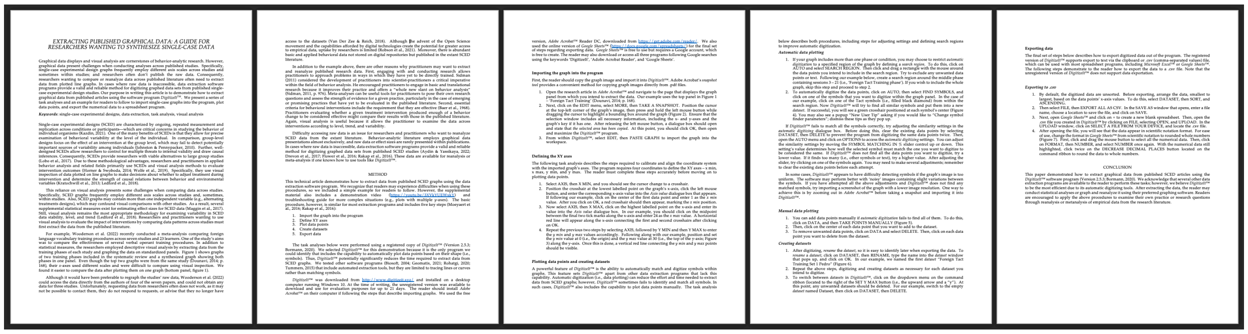


Figure 1. How the text file was saved before running the code. Note the following has been removed: headers and page numbers, figures, references, columnar formatting. It is a simple .docx format.

```
# -*- coding: utf-8 -*-
'''
Automatically generated by Colaboratory.
Original file is located at: https://colab.research.google.com/drive/195B5MPyzA9RkOYr1r6mT_q75tYef7VwZ
'''

# Packages and Modules
# System
from IPython.display import clear_output
from itertools import tee
from collections import Counter

# Data manipulation
import docx
from docx import Document
import networkx as nx
import numpy as np
import pandas as pd

# NLP
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from transformers import DistilBertModel, DistilBertTokenizer
from sklearn.decomposition import PCA

# Data Visualization
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import plotly.graph_objects as go
```

```
"""# Functions We'll Use"""
def extract_text_from_docx(docx_path):
    doc = docx.Document(docx_path)
    return ' '.join([para.text for para in doc.paragraphs])

def tokenize_and_filter(text):
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text.lower())
    return [word for word in words if word.isalpha() and word not in stop_words]

def create_corpus(documents):
    ordered_words = []
    seen_words = set()
    for doc in documents:
        filtered_words = tokenize_and_filter(doc)
        for word in filtered_words:
            if word not in seen_words:
                seen_words.add(word)
                ordered_words.append(word)
    return ordered_words

def get_word_embeddings(words):
    tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
    model = DistilBertModel.from_pretrained('distilbert-base-uncased')
    embeddings = []
    for word in words:
        inputs = tokenizer(word, return_tensors="pt", add_special_tokens=False)
        outputs = model(**inputs)
        word_embedding = outputs.last_hidden_state.mean(dim=1).detach().numpy().flatten()
        embeddings.append(word_embedding)
    return np.array(embeddings)

def create_bipartite_graph(corpus, text, normalized_embeddings):
    # Tokenize current document
    filtered_words = tokenize_and_filter(text)

    # Count pair occurrences
    pair_counts = Counter(zip(filtered_words, filtered_words[1:]))

    # Normalize counts to get alpha values
    max_count = max(pair_counts.values())
    alpha_values = {pair: 0.01 + 0.99 * (count / max_count) for pair, count in pair_counts.items()}

    # Create a bipartite graph
    B = nx.Graph()
    top_nodes = {f"top_{word}" for word in corpus}
    bottom_nodes = {f"bottom_{word}" for word in corpus}
    B.add_nodes_from(top_nodes, bipartite=0)
    B.add_nodes_from(bottom_nodes, bipartite=1)

    # Add edges with alpha values
    for word1, word2 in pair_counts:
        B.add_edge(f"top_{word1}", f"bottom_{word2}", alpha=alpha_values[(word1, word2)])

    # Plotting
    plt.figure(figsize=(10, 20))
    pos = {node: (0, i) for i, node in enumerate(top_nodes)}
    pos.update({node: (1, i) for i, node in enumerate(bottom_nodes)})

    # Create a color map based on embeddings
    assert len(normalized_embeddings) == len(corpus), "Length of embeddings and corpus do not match."
    color_map = {f"top_{word}": normalized_embeddings[i] for i, word in enumerate(corpus)}
    color_map.update({f"bottom_{word}": normalized_embeddings[i] for i, word in enumerate(corpus)})
    node_colors = [color_map[node] for node in B.nodes()]

    # Apply colormap
    colormap = cm.get_cmap('Spectral')
    node_colors = [colormap(color_map[node]) for node in B.nodes()]

    # Draw nodes
    nx.draw_networkx_nodes(B, pos, node_size=2, node_color=node_colors)
    # Draw edges with varying alpha
    for edge in B.edges(data=True):
        nx.draw_networkx_edges(B, pos, edgelist=[edge], alpha=edge[2]['alpha'])

    # Draw labels
    clean_labels = {node: node.split('_')[1] for node in B.nodes()}
    label_pos = {node: (pos[node][0] - 0.075 if node.startswith("top") else pos[node][0] + 0.075, pos[node][1]) for node in B.nodes()}
    nx.draw_networkx_labels(B, label_pos, labels=clean_labels, font_size=5)

    plt.subplots_adjust(left=0.15, right=0.85, top=0.95, bottom=0.05)
    plt.show()

# Normalize to range [0, 1] for coloring
def normalize_embeddings(embeddings):
    embeddings_range = max(embeddings) - min(embeddings)
    if embeddings_range == 0:
        return np.zeros_like(embeddings)
    else:
        return (embeddings - min(embeddings)) / embeddings_range


"""# Read in .docx documents and prep them for plotting"""
# Read in the docx files and convert to data we can use
gamified_operant = extract_text_from_docx('/content/gamified_human_operant_raw_text.docx')
extract_data = extract_text_from_docx('/content/extracting_published.docx')
scmo = extract_text_from_docx('/content/SCMO_plot.docx')

# Join the docs together for a single, combined text
combined_document = " ".join([gamified_operant, extract_data, scmo])

# List of documents to iterate over with each plot type
documents = [gamified_operant, extract_data, scmo, combined_document]



"""# Plot parallel bipartite plots of bigrams"""
corpus = create_corpus([gamified_operant, extract_data, scmo])
embeddings = get_word_embeddings(corpus)

pca = PCA(n_components=1)
reduced_embeddings = pca.fit_transform(embeddings).flatten()
normalized_embeddings = normalize_embeddings(reduced_embeddings)

for doc in documents:
    create_bipartite_graph(corpus, doc, normalized_embeddings)




"""# Function for double-helix and (cos(theta), 1-cos(theta)) rotating bipartite plots of bigrams"""
def create_bipartite_graph(corpus, text, normalized_embeddings, width_factor=0.5, double_helix=True):
    # Tokenize current document and count pair occurrences
    filtered_words = tokenize_and_filter(text)
    pair_counts = Counter(zip(filtered_words, filtered_words[1:]))

    # Normalize counts to get alpha values
    max_count = max(pair_counts.values())
    alpha_values = {pair: 0.01 + 0.99 * (count / max_count) for pair, count in pair_counts.items()}

    # Create a bipartite graph
    B = nx.Graph()
    top_nodes = {f"top_{word}" for word in corpus}
    bottom_nodes = {f"bottom_{word}" for word in corpus}
    B.add_nodes_from(top_nodes, bipartite=0)
    B.add_nodes_from(bottom_nodes, bipartite=1)

    # Add edges with alpha values
    for word1, word2 in pair_counts:
        B.add_edge(f"top_{word1}", f"bottom_{word2}", alpha=alpha_values[(word1, word2)])


# Plotting
    plt.figure(figsize=(5, 40))
    if double_helix==True:
        # Calculate positions for a double helix
        x_offset = width_factor  # Adjusts the horizontal spread
        y_positions = np.linspace(0, 1, len(top_nodes))
        pos = {}
        frequency_multiplier = 4  # Multiplier for frequency of helix turns
        helix_offset = np.pi / len(top_nodes)  # Offset between the two helices

        for i, node in enumerate(sorted(top_nodes)):
            angle = np.pi * y_positions[i] * frequency_multiplier
            pos[node] = np.array([np.cos(angle) * x_offset+0.08, y_positions[i]])

        for i, node in enumerate(sorted(bottom_nodes)):
            angle = np.pi * y_positions[i] * frequency_multiplier + helix_offset
            pos[node] = np.array([np.cos(angle) * x_offset, y_positions[i]])

    else:
        # Calculate positions using (cos(theta), 1-cos(theta)) offset
        x_offset = width_factor  # Use the width_factor to adjust the horizontal spread of the plot
        y_positions = np.linspace(0, 1, len(top_nodes))
        pos = {}
        frequency_multiplier = 4  # Multiplier to increase the frequency the coils

        for i, node in enumerate(sorted(top_nodes)):
            angle = np.pi * y_positions[i] * frequency_multiplier
            pos[node] = np.array([(np.cos(angle) + 1) * x_offset, y_positions[i]])

        for i, node in enumerate(sorted(bottom_nodes)):
            angle = np.pi * y_positions[i] * frequency_multiplier
            pos[node] = np.array([1 - (np.cos(angle) + 1) * x_offset, y_positions[i]])

    # Create a color map based on embeddings
    assert len(normalized_embeddings) == len(corpus), "Length of embeddings and corpus do not match."

    # Ensuring we have color for each node in top_nodes and bottom_nodes:
    color_map = {f"top_{word}": normalized_embeddings[i] for i, word in enumerate(corpus)}
    color_map.update({f"bottom_{word}": normalized_embeddings[i] for i, word in enumerate(corpus)})

    # Apply colormap
    colormap = cm.get_cmap('Spectral')
    node_colors = [colormap(color_map[node]) for node in B.nodes()]

    # The node_color list should now have an entry for every node in the graph:
    node_colors = [color_map[node] for node in top_nodes] + [color_map[node] for node in bottom_nodes]

    # Draw nodes
    nx.draw_networkx_nodes(B, pos, node_size=2, node_color=node_colors)

    # Draw edges with varying alpha
    for edge in B.edges(data=True):
        nx.draw_networkx_edges(B, pos, edgelist=[edge], alpha=edge[2]['alpha'], edge_color='grey')

    # Draw labels
    clean_labels = {node: node.split('_')[1] for node in B.nodes()}
    label_pos = {k: (v[0], v[1] - 0.02) for k, v in pos.items()}  # Adjust label positions
    nx.draw_networkx_labels(B, label_pos, labels=clean_labels, font_size=5)

    plt.axis('off')
    plt.show()


# Plot coiled visual for each article and overall corpus
gamified_operant = extract_text_from_docx('/content/gamified_human_operant_raw_text.docx')
extract_data = extract_text_from_docx('/content/extracting_published.docx')
scmo = extract_text_from_docx('/content/SCMO_plot.docx')
combined_document = " ".join([gamified_operant, extract_data, scmo])

corpus = create_corpus([gamified_operant, extract_data, scmo])
documents = [gamified_operant, extract_data, scmo, combined_document]
```

```
embeddings = get_word_embeddings(corpus)

pca = PCA(n_components=1)
reduced_embeddings = pca.fit_transform(embeddings).flatten()
normalized_embeddings = normalize_embeddings(reduced_embeddings)

for doc in documents:
    create_bipartite_graph(corpus, doc, normalized_embeddings, double_helix=False)
```