Published in final edited form as:

J Neurosci Methods. 2017 July 15; 286: 114–124. doi:10.1016/j.jneumeth.2017.06.002.

Dexterity: A MATLAB-based analysis software suite for processing and visualizing data from tasks that measure arm or forelimb function

Samuel D. Butensky^a, Andrew P. Sloan, PhD^{b,c}, Eric Meyers, PhD^{b,c}, and Jason B. Carmel, MD, PhD^{a,d,e,*}

^aBurke Medical Research Institute, White Plains, NY, USA, 10605

^bTexas Biomedical Center, The University of Texas at Dallas, Richardson, TX, USA, 75080

^cErik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, Richardson, TX, USA, 75080

^dBrain and Mind Research Institute, Weill Cornell Medical College, New York, NY, USA, 10065

^eDepartments of Neurology and Pediatrics, Weill Cornell Medical College, New York, NY, USA

Abstract

Background—Hand function is critical for independence, and neurological injury often impairs dexterity. To measure hand function in people or forelimb function in animals, sensors are employed to quantify manipulation. These sensors make assessment easier and more quantitative and allow automation of these tasks. While automated tasks improve objectivity and throughput, they also produce large amounts of data that can be burdensome to analyze. We created software called Dexterity that simplifies data analysis of automated reaching tasks.

New Method—Dexterity is MATLAB software that enables quick analysis of data from forelimb tasks. Through a graphical user interface, files are loaded and data are identified and analyzed. These data can be annotated or graphed directly. Analysis is saved, and the graph and corresponding data can be exported. For additional analysis, Dexterity provides access to custom scripts created by other users.

Results—To determine the utility of Dexterity, we performed a study to evaluate the effects of task difficulty on the degree of impairment after injury. Dexterity analyzed two months of data and allowed new users to annotate the experiment, visualize results, and save and export data easily.

Comparison with Existing Method(s)—Previous analysis of tasks was performed with custom data analysis, requiring expertise with analysis software. Dexterity made the tools required

*Corresponding Author: Jason B. Carmel: jason.carmel@med.cornell.edu; P: (914) 368-3135; 785 Mamaroneck Avenue, White Plains, NY, 10605.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

to analyze, visualize and annotate data easy to use by investigators without data science experience.

Conclusions—Dexterity increases accessibility to automated tasks that measure dexterity by making analysis of large data intuitive, robust, and efficient.

Keywords

Behavior; Analysis; Automated; Manipulation; Reach; Software

1. INTRODUCTION

To understand how the nervous system controls the hand and how control is lost with injury, dexterity is studied in both animals and humans. Several of the most common behavioral tasks used to study rodent forelimb function, including reaching and manipulation tasks, are slow, subjective, and labor-intensive (Allred et al., 2008; Ballermann, Metz, McKenna, Klassen, & Whishaw, 2001; Carmel, Berrol, Brus-Ramer, & Martin, 2010; Whishaw, Pellis, Gorny, & Pellis, 1991). And for humans, tests that assess hand and forearm dexterity, including the Fugl-Meyer test or Box and Block test, can be variable in both the administration of the test and the data gathered (Carroll, 1965; Fugl-Meyer, Jaasko, Leyman, Olsson, & Steglind, 1975; Platz et al., 2005).

Recently, several groups have begun to automate forelimb tasks in rodents (Gadiagellan, 2016; Seth A. Hays et al., 2013; S. A. Hays et al., 2013; Meyers et al., 2016; Sindhurakar et al., 2016; Wong, Ramanathan, Gulati, Won, & Ganguly, 2015). Meanwhile, others have focused on automating arm and hand tests in humans (Bosecker, Dipietro, Volpe, & Krebs, 2010; Wikholm & Bohannon, 1991). This increase in prevalence of automated tasks is largely due to decreased costs of sensors and increased accessibility to the tools that control these sensors. Automated tasks offer several advantages over manual tasks: objectivity, increased throughput, and decreased analysis time. In some cases, they provide a more sensitive loss of dexterity than conventional tasks (Sindhurakar et al., 2016). In addition, they allow neuroscientists to utilize some of the advantages of large data that weren't previously possible with non-automated tasks, such as analyzing kinetic and kinematic information in addition to success rate; quantifying learning or skill acquisition; and analyzing the quality of training (Gomez-Marin, Paton, Kampff, Costa, & Mainen, 2014).

However, with these advantages comes an increase in complexity of the analysis required. Analyzing large data sets requires the ability to manage the files, process the raw data, and visualize it in a format that is easy to understand. In addition, there are challenges storing processed data in a file format that minimizes its data footprint on the computer's hard drive. Generally, this would require the skills of a specialist with computational and/or engineering experience. These special requirements limit the widespread adoption of automated forelimb or arm tasks, and thus far, few have provided robust, intuitive solutions to analyze data from automated reaching tasks. We sought to create a software program that would make this complicated data analysis accessible to neuroscientists without data science expertise.

The program we built is called Dexterity. It's MATLAB-based and built around six core tasks for data analysis (Figure 1): Data can be selected and loaded easily (Fig. 1A), and the software will identify which task the raw data belongs to and analyze it appropriately (1B). Then, analyzed data can be annotated by group and experimental timeline (1C). Once loading, processing, and annotation are complete, the software visualizes the processed data (1D). If further analysis of a task is needed, the software provides custom analyses (1E). At the conclusion of an analysis, Dexterity can save the analysis and provide the option to export the analyzed data in spreadsheet format and the graph (1F). Overall, the software was designed to be intuitive, accessible, and expandable, so that other tasks that measure dexterity can be analyzed using the software in the future.

Dexterity incorporates tools we have used for analysis of an automated forelimb reaching task and has been made general for analysis of other forelimb or arm specific tasks (Sindhurakar et al., 2016). To add additional tasks to Dexterity's capabilities, the software needs to be modified at the file reading level, and our intention is to collaborate with other researchers to add more tests of dexterity to the software's capabilities.

2. METHODS

OVERVIEW

Dexterity is a MATLAB program that operates through a Graphical User Interface (GUI). We chose MATLAB as the software environment because it is familiar to most neuroscientists. In comparison to other commercially available and open source analysis programs, such as R, MATLAB has an extensive toolbox for graphing, analysis, and GUI development. MATLAB is largely designed by engineers for data analysis and visualization - the main focus of Dexterity - while R is built largely by statisticians for statistics. With regards to MATLAB's software infrastructure, its operations are vectorized, which reduces the number of commands needed for data processing and increases the processing speed.

When developing the software, we wanted an intuitive flow of information for the GUI (Figure 1). First, the user selects their data (Fig. 1A). Then, the program steps through each file and computes task-specific results (1B). If the program finds incomplete files, it prompts the user to keep or discard these files for analysis. Once task-specific processing is complete, the user annotates the experiment as necessary (1C) and graphs the results of the experiment or analysis (1D). The user can then add a custom analysis using a custom written MATLAB script of their own or another lab's (1E). Lastly, the user can export an image of the graph and the data associated with the graph, as well as access the Master File (1F).

Below, Dexterity is described in the context of Vulintus' MotoTrak system, specifically, the knob supination task (Meyers et al., 2016; Sindhurakar et al., 2016). Rats are placed into a reaching box, and the target of the reach is a knob that must be grasped and turned in supination to a success criterion (Figure 2F, "Supination"). Subjects are trained on the knob task twice daily for five days a week until they reach a success criterion, at which point a baseline is collected and the injury is performed. This is followed by administration of any treatment after injury.

Subjects are tested once a week after injury.

However, as mentioned above, data management and visualization is applicable to multiple automated forelimb or arm and hand tasks. The development of Dexterity was an iterative process, with each iteration being tested by both naive and experienced users. After each iteration, feedback was gathered and incorporated into the next version to refine the user experience.

2.1 Select Data

2.1.1 Choosing an Analysis Program—There are three ways of approaching an analysis session: Standard, Custom, and Previous. Standard analysis guides a user through loading and analyzing their data. This is the main function of Dexterity (Figure 2). Custom Analysis provides the user with a structure to find custom written analyses aimed at delving deeper into a subject's data, as well as a place to load custom analyses that they have written for their data analysis. Previous allows a user to access previously saved annotated and unannotated analyses.

2.1.2 Loading Data—After the user has clicked “Standard,” (Fig. 2A) the GUI prompts the user to select the file directory (2B). The GUI extracts the subject name from the file headers and asks the user to select which subjects to include (2C). After the user selects the subject or subjects to include in the analysis, the program steps through and loads in data from each file. If the program detects a file with less than five trials, it will ask the user if s/he wants to “Keep” or “Discard” the file (2D). During training, sessions are occasionally created accidentally and closed immediately, resulting in no data but a file created; these are usually discarded. However, after injury, a subject may be very impaired and not engage with the task. In these cases, the success rate and turn angle is recorded as a “Not-a-Number” (NaN), or the average of the success rate and turn angle of the trials present in the file (Duricki et al. 2016).

After keeping or discarding files, the user then annotates the experiment or returns to annotate later (2E). Annotating an experiment organizes the data by experimental parameters, such as timeline and groups. This is explained later in Section 2.3 and Figure 3. Dexterity can identify the task by the file format or file header (2F), and read the data using the appropriate MATLAB function (2G).

2.2 Standard Data Analysis

2.2.1 Processing Task Specific Data—From each trial, Dexterity calculates variables that are relevant to the task identified in the file reading step. Some variables like success rate and latency are relevant for multiple tasks, but other variables like turn angle are specific to an individual task. As an example, Figure 2H shows the analysis of variables for the supination task. The peak angle is calculated as the global maximum during the defined trial window. Latency is calculated only during a successful trial and is defined as the time from the initiation of the trial window to achieving the criterion turn angle. The velocity is calculated using the slope of the waveform, and the maximum velocity achieved during the waveform is recorded as the peak velocity. Acceleration is calculated as the derivative of the

velocity. The program then calculates the session average for peak angle, latency, peak velocity, and acceleration; both mean and median peak angle are calculated.

2.2.2 Smoothing data—For any task that requires a derivative during the analysis, a smoothing algorithm is used to account for noise in the waveform that may be amplified when calculating derivatives. While MATLAB offers many smoothing functions, we used the Savitsky-Golay filter to increase our signal-to-noise ratio while minimizing the distortion of the raw waveform (Savitzky & Golay, 1964).

2.2.3. Creating a Master File—Once the data has been loaded and analyzed, a Master File is created in .MAT format (2I). This Master File contains each subject's processed data and any task specific information. All graphs are created from the Master File.

2.3 Annotate Experiment

Often, studies are designed to compare groups of subjects and their response to injury and/or treatment. If, when the user is loading data, they choose to annotate their experiment, they must provide information about that experiment (Fig. 3A): experiment name, number and labels for groups, event data such as injury, length of the experiment, and labels for time points such as "Week 1" or "Day 1" (3B). After this information is entered, the user is asked where event data occurs (3C). Then the user places subjects in groups and states how many sessions should be included in each time point for each subject (3C). The program will organize the processed data into groups and then further into bins according to the time points provided by the user. Time points within a single bin will be averaged to provide a single data point for each group.

2.4 Graph Results

2.4.1 Single or Multiple Subject Visualization—If the user decides not to annotate the master file, the data for a subject or multiple subjects can still be accessed. The data are presented in a tabbed GUI, where each tab represents a subject and its data (Fig. 4A1). Within each tab is the subject's name, the time of the last behavioral session and the last behavioral stage on which they were run, their total number of sessions, and the devices they've been run on.

If a user wants to graph all subjects' data, the user can select "Graph All Subjects" and a separate GUI appears (4A2). This contains a dropdown menu of the variables calculated during data processing and displays a line-graph of whichever variable is selected to graph; the default is "Success Rate." The user chooses whether to graph by session, day, or week. The user can also hover over each data point and see the date or dates, subject name, number of successful trials, and the total number of trials contained in that data point. The user is able to concatenate dates (e.g. to align by the time of injury) and group subjects; grouping subjects will replace individual subjects' colors with the group color but does not create a single line per group. If the user wants to group subjects by cohort and display a single line per cohort, the user can choose to annotate their experiment from the "Graph All Subjects" GUI, discussed below in Section 2.4.2.

In addition to graphing all subjects, the user can graph a single subject (4A3). Should the user want a more in-depth analysis of that individual data point, the user can click the data point, and a separate GUI appears which allows the user to step through each trial and export the waveform as an image (4A4). This has usefulness during training when certain behaviors need to be shaped out or accentuated.

2.4.2 Graphing Groups of Subjects—Annotated experiments enable comparison of groups of subjects or conditions. Once annotation is performed (steps in Figure 3), a separate GUI appears that provides an overview of the experiment, including subjects, groups, event data, and time point labels (Figure 4B1). Results of the experiment can be plotted by clicking the “Plot” button which opens a separate window with various graphing capabilities (4B2). The user can switch between variables and graphing styles, either line or bar graph. In addition, they can toggle between color or grayscale color schemes. Event data is plotted as a vertical line where the event occurred, and an arrow points to this line with the name of the event next to the arrow. Standard error bars are created for the user and plotted as well. A legend contains information about the graph.

2.5 Statistical Analysis

In an annotated analysis, the user can perform statistical analysis on their data. After clicking “STATS” (Fig. 4B2), Dexterity will prompt the user to select the data that they would like to analyze. Dexterity then performs a 2-Way ANOVA that compares baseline values to values after an intervention. If the comparison is significant, Dexterity will prompt the user to select a post-hoc test from a list: Tukey-Kramer, Bonferroni, Dunn-Sidak, LSD, and Scheffe. After picking a post-hoc test, Dexterity will display the results for the 2-Way ANOVA and the post-hoc comparisons. All data sets that are complete and don’t contain missing values (i.e. NaN) can be analyzed in Dexterity. However, if the dataset is unbalanced and contains missing values, Dexterity is unable to analyze the data using a standard 2-Way ANOVA. For unbalanced data sets, Dexterity will alert the user that the data set is unbalanced. The user can then export their data and use a statistical package of their choosing to analyze their data. For our purposes, we use SPSS (Version 22). In the results, we had two unbalanced data sets: Isometric Pull Data, and Turn Angle for the 75 degree group. For these two data sets, we used SPSS to run a repeated measures ANOVA with Bonferroni post-hoc test to evaluate statistical significance.

2.6 Custom Analysis

Our lab has written several MATLAB scripts to look deeper into the kinetics and kinematics of forelimb reaching, accessible via the “Custom” analysis function of Dexterity (Figure 5). For example, when subjects display minimal impairment after injury, or appear to have recovered after significant impairment, we can use several scripts to look closer at turn profiles and measures of smoothness to see if there are subtle differences in how they are doing the task. These MATLAB scripts are stored in a master repository hosted by BitBucket. The repository is organized such that each lab has a folder containing their custom scripts. When a new lab uses the software, they can create a new lab folder in the master repository and upload any custom scripts they may have written for their analysis. These scripts get pushed to the master repository so that each lab using Dexterity has access

to this analysis. A user also has the option to load in another lab's software from the master repository. This organization is meant to foster collaboration among labs using the same automated tests.

2.7 Save and Export

The user has the option to save their analysis so that they can resume analysis later. All master files are saved as .MAT and can be opened in MATLAB to view raw and processed data. All data can be exported in CSV format and read by the user's preferred text-file editor or spreadsheet application. Which data are exported depends on the data displayed graphically. For example, in unannotated analyses, if the user is looking at multiple subjects in a "Weekly" view, when the user clicks "Export," weekly data for multiple subjects will be exported. However, if the user is viewing a single subject by session, then only that subject's data, organized by session, will be exported. With regards to annotated analyses, data is exported in CSV format, with each line representing one data point per subject per time point. All graphs can also be exported in JPEG, PNG, or PDF format.

2.8 Adding Tasks

Currently, Dexterity analyzes data from three automated tasks (Supination, Isometric Pull, and Lever) and one non-automated task (Vermicelli Manipulation). The supination task is described above. The isometric pull task uses the same Vulintus, Inc. behavior box as the supination task. In brief, the isometric pull task requires animals to pull a lever shaped manipulandum to a certain force threshold to receive a reward (Hays et al., 2013). The lever task (Hays et al., 2013) has not been shown here to avoid redundancy. Finally, the Vermicelli Manipulation task records the number of manipulations an animal makes while eating a 7cm piece of pasta (Allred et al., 2008).

Each task has a file reading function in MATLAB that reads raw data and analyzes it for task-specific parameters (Figure 2H). For the Vulintus, Inc. tasks, raw data from the infrared sensor is stored in a custom binary format with headers, whereas, for the Vermicelli Manipulation Task, raw data is in an excel spreadsheet that the experimenter has manually filled out. For tasks record data using Excel Spreadsheets or text files with separators, a template is created (<http://bit.ly/2pjnKUT>). To add a new task, the MATLAB file reading function needs to be modified to read in data from the new task. The file reading code has three main functions. It identifies the task, stores the raw data files, and organizes data by animal. It stores these three parameters in a structured variable called "data"; structured variables are explained further on BitBucket. Each file reading function is annotated so that other labs will be able to see how their analyzed data needs to be configured and adopt it to their needs. Once this file reading function is complete, the rest of Dexterity's workflow (Figure 1C–1F) can continue. Support for creating new file reading functions or modifying existing ones as well as how to add tasks can be found on Dexterity's wiki on BitBucket (<http://bit.ly/2oU6liw>).

3. RESULTS

To demonstrate the utility of Dexterity, we performed a study to evaluate the effects of task difficulty on the degree of impairment after injury. We hypothesized that a more difficult training regimen and criterion is required to see a deficit after a cut to the corticospinal tract, since rats rely much less on the corticospinal tract than do humans (Friedli et al., 2015). Eight rats were trained on the automated forelimb supination task: four using a turn angle criterion of 60 degrees and weight of 7.5g, and four using a turn angle criterion of 75 degrees and 6g. Subjects were considered trained to baseline performance when they could turn to their respective turn criterion at a success rate of 75% or above. After a subject reached baseline, four consecutive sessions were collected. Then, the subject received a pyramidotomy, which is a unilateral cut injury to the corticospinal tract, as we have done previously (Brus-Ramer, Carmel, Chakrabarty, & Martin, 2007; Brus-Ramer, Carmel, & Martin, 2009; Carmel et al., 2010; Carmel, Kimura, & Martin, 2014; Sindhurakar et al., 2016; Sindhurakar et al., 2017). Testing was performed at 3, 7, 14, 21, 28, and 35 days after injury. We also show data from two additional tasks, as described below. All protocols were approved by the Institutional Animal Use and Care Committee of Weill Cornell Medicine.

3.1 Select Data

Using a Standard analysis, the data was loaded by selecting the directory, then the eight subjects included in this study (Fig. 2A–C). Selecting the subjects caused eight weeks of data comprising 131 files to be loaded and processed. Two files contained no or incomplete data (Fig. 2D). All five were kept because they were post-injury testing sessions in which a subject engaged with the task for less than five trials. For sessions that didn't include any data, a non-zero identifier was automatically placed for mean turn angle and success rate. This non-zero identifier is called a NaN, or "Not-a-Number," and is commonly used to represent missing data (Duricki et al. 2016). After keeping or discarding files, the Annotate Now option was chosen from the next prompt (Fig. 2E).

3.2 Standard Data Analysis

Dexterity identified the behavioral task as the knob supination task based on the file format (Fig. 2F). The program read in the raw data (Fig. 2G) and analyzed each waveform for peak angle, latency, velocity, and success rate (Fig. 2H). These variables were averaged per session and compiled into a table of data per subject. Each subject's table of data was stored in a Master File (Fig. 2I).

3.3 Annotate Experiment

The Annotate Experiment function was used to identify two groups, 60 degree training and 75 degree training, an injury event, and six total weeks within the Master File (Fig. 3). After processing the data, subjects and their corresponding processed data were placed into groups, and Dexterity prompted the user to establish the number of sessions within each time period; data was averaged for each period. Dexterity then asked the user about time point labels and where any event data occurred. In this case, the labels were "Pre, 3, 10, 17, 24, 31," and the event occurred between "Pre" and "3."

3.4 Graph Results and Perform Statistics

After this information was provided, a separate window appeared that provided a general overview for the experiment (Fig. 4B1). The success rate and turn angle were then plotted (Figure 6A and 6B). For the 60 degree there were deficits in success rate post-injury that remained 31 days post-injury (Figure 6A). The success rate before injury was $86 \pm 1.0\%$, and dropped to $44 \pm 7.4\%$ 3 days post-injury, a significant decrease ($p = 0.04$). With regards to turn angle (Figure 6B), the 60 degree group decreased from 69 ± 0.96 degrees pre-injury to 52 ± 4.8 degrees 3 days post-injury, which was not significant ($p = 0.09$).

For the 75 degree training group, there were deficits in both success rate and turn angle post-injury that remained out to 31 days post-injury. From pre-injury to 3 days post-injury, the success rate dropped from $83 \pm 1.5\%$ to $4.6 \pm 3.5\%$, respectively, a significant decrease ($p < 0.01$). The decrease in turn angle from 85 ± 1.5 degrees to 26 ± 6.8 degrees 3 days post-injury was not significant ($p = 0.79$).

Comparing the two training groups, there was an effect size in both groups, but there was a larger effect size in the 75 degree group as compared to the 60 degree group. With regards to success rate, the 75 degree group showed an impairment of 94% from pre- to 3 days post-injury, while the 60 degree group showed an impairment of 48% pre- to 3 days post-injury. By 31 days post-injury, each group was still different than their respective pre-injury data, but the 75 degree group was impaired by 85% at 31 days post-injury while the 60 degree group was impaired 13%. As for turn angle, there was a 25% impairment in the 60 degree group from pre- to 3 days post-injury and a 70% impairment in the 75 degree group. By 31 days post-injury, the 60 degree group was turning within 1% of their pre-injury average turn angle, while the 75 degree group was still 57% impaired.

Overall, training to 75 degrees showed greater differences between pre- and 3 days post-injury when compared to the 60 degree training group.

3.5 Custom Analysis

To delve deeper into how subjects in each cohort were performing the task, we utilized two custom written functions as part of the “Custom” analysis portion of Dexterity. The first program graphs turn angle against time for each time point (Figure 6C). The program compiles the raw waveforms for a single time point and averages them. This produces an average waveform and is repeated for each time period. In Figure 6C, one representative subject was selected from each cohort. For the subject in the 60 degree group (Figure 6C1), there was a deficit post-injury, as evidenced by the lower peak, but this subject recovered gradually each week until the “Wk5” waveform was similar to the pre-injury waveform. However, for the subject in the 75 degree group (Figure 6C2), there was a clear deficit post-injury with partial recovery over 5 weeks. Therefore, while in both groups there was deficit, for the subject in the 60 degree group there was a full recovery, whereas for the subject in the 75 degree group there was a partial recovery.

We also used another program as part of the “Custom” portion of Dexterity to plot the peak turn angle against latency for all time periods for a subject in each group (Figure 6D). This allows for easy visualization of the relationship between both latency and turn angle. For the

subject in the 60 degree group (Figure 6D1), there was an increase in latency and decrease in turn angle after injury. By week five, the subject has regained the ability to turn to pre-injury angles, however, the latency was longer as compared to the pre-injury point. As for the subject in the 75 degree cohort (Figure 6D2), the latency by week five was still longer than pre-injury. Thus, using the peak angles program, we observed that while the 60 degree subject recovered the ability to turn 60 degrees, the time to the peak was longer. For the 75 degree subject, the performance on the task was worse out to 5 weeks, the end of the study.

3.6 Save and Export

All experimental data was saved in .MAT file and filed under the experimental name. Raw data were exported in .XLS format and imported into SPSS for statistical analysis. All graphs in Figure 6 were exported as .PDFs; the images appear as they did on export. The images were then imported into Adobe Illustrator to label and combine images for figures.

3.7 Additional tasks

Two additional tasks were analyzed using Dexterity: an automated task, the Isometric Pull task, and a non-automated task, the Vermicelli Manipulation task (Figure 7). Data for the isometric pull task was reused from a study that conducted a within-animal comparison of skilled forelimb assessments in rats using a different analysis tool (Sloan et al. 2015). The file reading function used for Vulintus, Inc. supination task was modified to read in raw data from the pull task. Raw data for the pull task was in the same custom, binary format as the supination task, but the task-specific parameters that needed to be calculated were different, notably, peak force. Once the file reading function was modified, raw isometric pull force from 10 animals was read and processed using “VulintusRead.m” (Figure 7A1). The data was then annotated to include the injury time point and group the 10 animals into an injury group. A unilateral motor cortex ischemic lesion was performed after animals reached baseline, and animals were given one week for recovery (Figure 7A2). In the beginning of week two, animals were tested once weekly until week six. After lesion, animals show a significant decrease in peak force ($p < 0.01$) from 149 ± 5.4 grams pre-injury to 84 ± 7.1 grams 7 days post-injury. The peak force at 6 weeks post-injury was 103 ± 11 grams; this was not significantly different from baseline after post hoc correction ($p = 0.12$).

Data for the Vermicelli Manipulation was previously unpublished. Scoring for the Vermicelli Manipulation task was input into a scoring template. A file reading function was created to read in raw data from a spreadsheet, and this function was modified to analyze task-specific parameters for the pasta manipulation task (Figure 7B1). The data was annotated to include the injury time point and group the four animals into an injury group. A unilateral internal capsule lesion targeting the right paw was performed after animals reached baseline, and animals were tested starting day one post-injury for three days (Figure 7B2). The number of manipulations for each paw were similar at baseline (data not shown), although they vary considerably from animal to animal. To reduce the variability, the number of baseline manipulations were normalized to 1. This is a function within Dexterity that the user can choose during the visualization. Normalization was performed by dividing the average number of manipulations at each time point by the average number of manipulations before injury. This normalized number of manipulations for each subject was averaged for the

group. After lesion, animals show a significant decrease in right paw manipulations ($p < 0.01$) to 0.20 ± 0.08 while animals show a significant increase in left paw manipulations to 1.9 ± 0.7 ($p = 0.03$).

4. DISCUSSION

Dexterity allows us to load and process data from automated and non-automated reaching tasks, annotate this data, graph results, observe trends, and export processed data. Most importantly, it significantly shortens the time required to analyze and visualize our data, and the learning curve for the naive user is short and gradual. While we have not submitted this software to usability scales, we believe that the iterative development process and time spent refining the flow of information makes Dexterity intuitive for both the naive and experienced user (Brooke, 1996). Evaluated in the context of the knob supination task, Dexterity allowed the user to load in over two months of data and fully analyze and export graphs in approximately 10 minutes. From loading data to a final, processed figure with statistical analysis included, Figure 6A and 6B required one hour to complete. This is a significant reduction in time spent analyzing data and processing the figure images.

While Dexterity offers a robust solution for analyzing data from automated and non-automated tasks, there are some limitations. Four tasks can currently be processed by Dexterity without any modification. However, additional automated and non-automated tasks can be added by modifying the file reading function. As long as non-automated tasks utilize a scoring template in spreadsheet or CSV format, they can be read by modifying the file reading function for the Vermicelli Manipulation Task. For example, tasks like the staircase test which utilize a template scoring sheet are well suited for Dexterity (Montoya, Campbell-Hope, Pemberton, & Dunnett, 1991). In addition, there are limitations inherent in using MATLAB as the environment for Dexterity. Dexterity's file reading and saving functions are built for Windows, but with slight modifications, they can run on Mac and Linux. Or, Dexterity can be run on Mac/Linux computers without any changes if the machine is running a Windows emulator. And if deployment on these machines becomes crucial, Dexterity can be converted to a friendlier coding environment for ubiquitous deployment, such as C or C++.

Dexterity is hosted on a public repository (see Methods) so that it can be freely accessed. This allows labs to share issues, solutions, and custom analyses. When used in this way, data from other tasks that utilize different sensors can be analyzed with Dexterity. To do this, Dexterity needs to be modified at the file reading mechanism. If subjects are tested on multiple tasks, this data can be loaded in one batch and Dexterity will recognize the different tasks based on the file format or headers. This makes comparing a subject's performance on multiple tasks, such as the supination, isometric pull, and lever task, efficient.

Dexterity is designed to adapt. It can be used to analyze longitudinal data, both from sensors and observer recorded data. For example, kinematic data that can be represented by analog variables, such as distance from midline or variance of motion, can be analyzed and visualize those variables. New features can be added using the Custom function, and popular analyses will be made available in the Standard program. As open access software, Dexterity

is designed to be adopted by any interested laboratories; new implementations can then be shared across the scientific community.

Acknowledgments

This research has been funded by NIH grants R03NS091737 and R01NS092875 and a New York State Spinal Cord Injury Research Board grant to J.B.C. Special thanks to Thelma Bethea, Joshua Santos, Anil Sindhurakar, and Shivakeshavan Ratnadurai Giridharan for their contributions in the early stages of Dexterity's development and ongoing beta-testing.

References

- Allred RP, Adkins DL, Woodlee MT, Husbands LC, Maldonado MA, Kane JR, Schallert T, Jones TA. The vermicelli handling test: a simple quantitative measure of dexterous forepaw function in rats. *J Neurosci Methods*. 2008; 170(2):229–244. DOI: 10.1016/j.jneumeth.2008.01.015 [PubMed: 18325597]
- Ballermann M, Metz GA, McKenna JE, Klassen F, Whishaw IQ. The pasta matrix reaching task: a simple test for measuring skilled reaching distance, direction, and dexterity in rats. *J Neurosci Methods*. 2001; 106(1):39–45. [PubMed: 11248339]
- Bosecker C, Dipietro L, Volpe B, Krebs HI. Kinematic robot-based evaluation scales and clinical counterparts to measure upper limb motor performance in patients with chronic stroke. *Neurorehabil Neural Repair*. 2010; 24(1):62–69. DOI: 10.1177/1545968309343214 [PubMed: 19684304]
- Brooke, J. SUS: a “quick and dirty” usability scale. In: Jordan, PW, Thomas, B, Weerdmeester, BA., McClelland, AL., editors. *Usability Evaluation in Industry*. 1996.
- Brus-Ramer M, Carmel JB, Chakrabarty S, Martin JH. Electrical stimulation of spared corticospinal axons augments connections with ipsilateral spinal motor circuits after injury. *J Neurosci*. 2007; 27(50):13793–13801. DOI: 10.1523/JNEUROSCI.3489-07.2007 [PubMed: 18077691]
- Brus-Ramer M, Carmel JB, Martin JH. Motor cortex bilateral motor representation depends on subcortical and interhemispheric interactions. *J Neurosci*. 2009; 29(19):6196–6206. DOI: 10.1523/JNEUROSCI.5852-08.2009 [PubMed: 19439597]
- Carmel JB, Berrol LJ, Brus-Ramer M, Martin JH. Chronic electrical stimulation of the intact corticospinal system after unilateral injury restores skilled locomotor control and promotes spinal axon outgrowth. *J Neurosci*. 2010; 30(32):10918–10926. DOI: 10.1523/JNEUROSCI.1435-10.2010 [PubMed: 20702720]
- Carmel JB, Kimura H, Martin JH. Electrical stimulation of motor cortex in the uninjured hemisphere after chronic unilateral injury promotes recovery of skilled locomotion through ipsilateral control. *J Neurosci*. 2014; 34(2):462–466. DOI: 10.1523/JNEUROSCI.3315-13.2014 [PubMed: 24403146]
- Carroll D. A Quantitative Test of Upper Extremity Function. *J Chronic Dis*. 1965; 18:479–491. [PubMed: 14293031]
- Duricki DA, Soleman S, Moon LDF. Analysis of longitudinal data from animals with missing values using SPSS. *Nat Protoc*. 2016 Jun; 11(6):1112–29. [PubMed: 27196723]
- Friedli L, Rosenzweig ES, Barraud Q, Schubert M, Dominici N, Awai L, Nielson JL, Musienko P, Nout-Lomas Y, Zhong H, Zdunowski S, Roy RR, Strand SC, van den Brand R, Havton LA, Beattie MS, Bresnahan JC, Bezaud E, Bloch J, Edgerton VR, Ferguson AR, Curt A, Tuszynski MH, Courtine G. Pronounced species divergence in corticospinal tract reorganization and functional recovery after lateralized spinal cord injury favors primates. *Science Translational Medicine*. 2015; 7(302):302ra134.
- Fugl-Meyer AR, Jaasko L, Leyman I, Olsson S, Steglind S. The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance. *Scand J Rehabil Med*. 1975; 7(1):13–31. [PubMed: 1135616]
- Gadiagellan, D., Nanayakkara, T., Moon, LDF. The in-Cage Automation of the Single Pellet Reaching Task. A Test for Dexterity in Rats. 2016. Retrieved from <http://www.abstractsonline.com/pp8/index.html#!4071/presentation/31759>

- Gomez-Marín A, Paton JJ, Kampff AR, Costa RM, Mainen ZF. Big behavioral data: psychology, ethology and the foundations of neuroscience. *Nat Neurosci*. 2014; 17(11):1455–1462. DOI: 10.1038/nn.3812 [PubMed: 25349912]
- Hays SA, Khodaparast N, Sloan AM, Fayyaz T, Hulse DR, Ruiz AD, Pantoja M, Kilgard MP, Rennaker RL II. The bradykinesia assessment task: An automated method to measure forelimb speed in rodents. *Journal of Neuroscience Methods*. 2013; 214(1):52–61. <http://dx.doi.org/10.1016/j.jneumeth.2012.12.022>. [PubMed: 23353133]
- Hays SA, Khodaparast N, Sloan AM, Hulse DR, Pantoja M, Ruiz AD, Kilgard MP, Rennaker RL 2nd. The isometric pull task: a novel automated method for quantifying forelimb force generation in rats. *J Neurosci Methods*. 2013; 212(2):329–337. DOI: 10.1016/j.jneumeth.2012.11.007 [PubMed: 23183016]
- Meyers E, Sindhurakar A, Choi R, Solorzano R, Martinez T, Sloan A, Carmel J, Kilgard MP, Rennaker RL II, Hays S. The supination assessment task: An automated method for quantifying forelimb rotational function in rats. *J Neurosci Methods*. 2016; 266:11–20. DOI: 10.1016/j.jneumeth.2016.03.007 [PubMed: 26976724]
- Montoya CP, Campbell-Hope LJ, Pemberton KD, Dunnett SB. The “staircase test”: a measure of independent forelimb reaching and grasping abilities in rats. *Journal of Neuroscience Methods*. 1991; 36(2–3):219–228. [http://dx.doi.org/10.1016/0165-0270\(91\)90048-5](http://dx.doi.org/10.1016/0165-0270(91)90048-5). [PubMed: 2062117]
- Platz T, Pinkowski C, van Wijck F, Kim IH, di Bella P, Johnson G. Reliability and validity of arm function assessment with standardized guidelines for the Fugl-Meyer Test, Action Research Arm Test and Box and Block Test: a multicentre study. *Clin Rehabil*. 2005; 19(4):404–411. [PubMed: 15929509]
- Savitzky A, Golay MJE. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*. 1964; 36(8):1627–1639. DOI: 10.1021/ac60214a047
- Sindhurakar A, Butensky SD, Meyers E, Santos J, Bethea T, Khalili A, Sloan AP, Rennaker RL III, Carmel JB. An Automated Test of Rat Forelimb Supination Quantifies Motor Function Loss and Recovery After Corticospinal Injury. *Neurorehabil Neural Repair*. 2016; doi: 10.1177/1545968316662528
- Sindhurakar A, Mishra AM, Gupta D, Iaci JF, Parry TJ, Carmel JB. Clinically Relevant Levels of 4-Aminopyridine Strengthen Physiological Responses in Intact Motor Circuits in Rats, Especially After Pyramidal Tract Injury. *Neurorehabil Neural Repair*. 2017; 1545968316688800. doi: 10.1177/1545968316688800
- Sloan AM, Fink MK, Rodriguez AJ, Lovitz AM, Khodaparast N, Rennaker RL, et al. A Within-Animal Comparison of Skilled Forelimb Assessments in Rats. *PLoS ONE*. 2015 Oct 27;10(10):e0141254. [PubMed: 26506434]
- Whishaw IQ, Pellis SM, Gorny BP, Pellis VC. The impairments in reaching and the movements of compensation in rats with motor cortex lesions: an endpoint, videorecording, and movement notation analysis. *Behav Brain Res*. 1991; 42(1):77–91. [PubMed: 2029348]
- Wikholm JB, Bohannon RW. Hand-held Dynamometer Measurements: Tester Strength Makes a Difference. *J Orthop Sports Phys Ther*. 1991; 13(4):191–198. DOI: 10.2519/jospt.1991.13.4.191 [PubMed: 18796845]
- Wong CC, Ramanathan DS, Gulati T, Won SJ, Ganguly K. An automated behavioral box to assess forelimb function in rats. *Journal of Neuroscience Methods*. 2015; 246:30–37. <http://dx.doi.org/10.1016/j.jneumeth.2015.03.008>. [PubMed: 25769277]

Highlights

- We built MATLAB-based software, Dexterity, for analysis of forelimb reaching tasks
- Dexterity handles data from optical encoders and force sensors
- It can be modified to incorporate additional tasks
- Dexterity provides intuitive and quick data selection, annotation, and visualization
- We used Dexterity to analyze behavior data from three rodent forelimb tasks

Figure 1. Overview of Dexterity

Bolded text represents user actions while green boxes with white text reflect how Dexterity handles and processes data. (A) The user selects files containing raw data from a supported task and then chooses the subjects to be analyzed. (B) Dexterity analyzes the raw data, computing task specific parameters and compiling the processed data into a Master File (.MAT). (C) The user can then annotate data from the Master File and (D) graph the results. (F) They can also add custom analysis if necessary and produce custom visualizations, then (F) save and export the graph and/or data.

Figure 2. Data management and processing

(A) A user can select between three options: “Standard” “Custom,” and “Previous.” If the user selects “Standard”, the user will (B) load in their data, and Dexterity will prompt the user to provide the directory where files are located and (C) select subjects. (D) It will then step through each file and prompt the user to keep or discard files with incomplete or no data. (E) If the user chooses, they can annotate their experiment now or later. (F) Dexterity identifies which task it is analyzing, (G) utilizes the appropriate file reading MATLAB function, (H) analyzes that task, and then (I) creates a Master File with all of the processed data organized by subject.

Figure 3. Annotate Experiment

(A) A user can annotate a new experiment and provide information about the experiment: experiment name, group names, event data, and total weeks. (B) Once an experiment is created, the user will group subjects and (C) organize the data for each subject, describing how many sessions occurred, per time point, labeling time points and elaborating on where event data occurred.

A. Graph Animal(s)

A1. Subject Overview

A2. Graph All Subjects

A3. Graph Single Subject

A4. View Individual Trials

B. Graph Group(s)

B1. View Experiment Overview

B2. Graph Experiment

Figure 4. Graph Results

(A) Graph Subject(s). (A1) Dexterity displays an overview of task data in a tabbed GUI where each tab contains a summary of each subject's data. (A2) The user also has the ability to graph all subjects on a single graph and concatenate dates or group subjects. (A3) Dexterity can isolate a single subject from that group of subjects. A user can hover over each data point and receive quick information about that data point. (A4) A user can also click on each data point and view the trials contained in that data point as well as the raw signal from the sensor, and export these trials as image files. (B) Graph Group(s). (B1) Dexterity displays an overview of the annotated experiment, including subjects, groups, event data, and time point labels. (B2) The user can then graph the results from the experiment. If the user wants to perform statistical analysis, they can click "STATS" and Dexterity will perform a 2-Way-ANOVA on the selected variable and prompt the user to select a post-hoc analysis: Tukey-Kramer, Bonferroni, Dunn-Sidak, LSD, and Scheffe.

A. User Interface

B. Infrastructure

Figure 5. Custom Analysis

(A) If the user chooses “Custom” from the analysis programs, another GUI will appear that contains custom MATLAB analyses organized by labs. If it’s the first time a user is using a program, Dexterity will prompt the user to load an existing lab’s analysis or add a new lab. Once the user has decided on a custom analysis program, they can single-click on the program and the custom analysis will run. (B) Custom analysis depends on the ability of Dexterity to communicate with a master repository of functions organized by lab hosted on BitBucket. If a new lab is created, the program will ask for the lab name and where the functions are. It will then check the main repository for that program and if it doesn’t exist, Dexterity will push a new folder labeled by the lab name containing the custom analyses to the main repository. If an existing lab is loaded, Dexterity will prompt the user to choose a lab from the main repository and copy the lab and corresponding files to the local user’s computer.

Figure 6. Case study results

We compared two criteria for pre-injury performance level on task performance before and after corticospinal tract injury. (A) and (B) show the “Standard” analysis program of Dexterity, while (C) and (D) show the “Custom” analysis program. (A) Both the 60 degree ($n = 4$) and 75 degree ($n = 4$) training groups showed a decrease in success rate post-injury and each week after injury, as compared to pre-injury success rate, for 31 days post-injury. (B) The 75 degree training group showed a decrease in turn angle post-injury and each week after injury, as compared to pre-injury turn angle, for 31 days post-injury. The 60 degree training group also a decrease in turn angle post-injury, though less than the 75 degree training group. Turn profiles (C) and Peak Angles (D) for a single subject from each training group. (C1) This subject showed a decrease in turn profile after injury, but by 31 days post-injury has a nearly identical waveform as before injury. (C2) This subject also showed a decrease in turn profile after injury but only has a partial recovery that plateaus by 31 days post-injury. (D1) This subject showed a decrease in peak angle and increase in latency after injury. While it recovered its peak angle, the latency remained longer than before injury. (D2) This subject is highly impaired after injury as shown by a decrease in peak angle. Latency decreased post-injury and recovered to a pre-injury level by week five but the peak angle remained impaired.

Figure 7. Additional Tasks

(A) The Isometric Pull task ($n = 10$). (A1) Raw data from the Ardymotor File was read and processed using the MATLAB function “VulintusRead.m.” (A2) This data was annotated and the peak force was visualized. After injury, there is a decrease in peak force starting in week two post-injury and lasting till week six post-injury. (B) The Vermicelli Manipulation task ($n = 4$). Four animals were trained and tested on the Vermicelli manipulation task and (B1) the raw data was read and processed using the MATLAB function “VermicelliRead.m.” (A2) This data was annotated and the normalized manipulations for the right and left paw were visualized. After injury, there is a decrease in normalized manipulations for the targeted paw (right paw) and an increase for the non-targeted paw (left paw) that lasts till day three after injury.