Documentation for **{lasertrapr}**

Brent Scott

2025-07-11

Contents

Chapter 1

Getting Started

{lasertrapr} is an R-package/Shiny application built using the `{golem}` framework for automating the analysis of laser trap data. Please note that the app is currently still under development. Users should proceed with caution.

NOTE: an R-package is denoted by the {} braces

1.1 Install R & RStudio

Currently, **{lasertrapr}** can only be launched from an active R-session. Before you begin you will need to download and install R and RStudio. Both of these are free. RStudio is an IDE (integrated development environment) and is not 100% necessary, but is recommended. Follow the instructions on the respective websites to complete installation.

Additionally, you will need to install the R build toolchain. This allows a user to build an R-package from source on their own computer. The link provides directions on how to do this on Windows, Mac, and Linux.

At the moment there is a lot of overheard needed to get the application up and running. Hopefully, in the future a more stable version can be released onto CRAN (Comprehensive R Archive Network) or preferably as a standalone program.

1.2 Download {lasertrapr}

Open R/RStudio on your computer. You can download the latest stable version of the app from my *drat* repository on Github with the following code. Copy/paste the following into the R-Console or into an R-Script to run:

```r
install.packages("drat") # installs drat package
drat::addRepo("brentscott93") # adds my drat repo
install.packages("lasertrapr") # installs the lasertrapr package
```

For users that want the latest developmental version to fork and contribute you can install from GitHub with:

```r
install.packages("devtools") # i.e. "developer tools"
library(devtools) # loads the package
install_github("brentscott93/lasertrapr") # downloads lasertraor from github
```

Both {devtools} and {lasertrapr} will need to install dependencies. Update and install all the packages that they want when prompted. Alternatively you can install {devtools} from within the RStudio IDE by navigating to the "Packages" tab in the lower right box and clicking "Install".

1.3 Launch the App

Once your have successfully installed and built the `{lasertrapr}` package, you are ready for launch:

```r
library(lasertrapr) # load the package and its dependencies
run_app() # this will launch the app GUI in your default web browser
```

Once the initial setup and installation is completed, the above two lines is the only code that will need to be run each time you want to use the app. You can update to the current developers version anytime by re-running `install.packages("lasertrapr")` or `devtools::install_gitub("brentscott93/lasertrapr")`.

Chapter 2

Upload Data

2.1 The ~/lasertrapr/ folder

The **{lasertrapr}** app offers more than just the analysis of single laser trap data traces, but the application also serves as an *opinionated data management* tool. The app forces users to adopt a specific directory tree structure. All data can be organized into specific **project**, **conditions**, **date**, and **observation** folders. Additionally, this structure can be leveraged in the post-analysis stage to easily handle statistical analyses and auto-generation of plots for quick summary features provided by the app. Upon initial launch of the app, a *lasertrapr* folder will be created under `~/lasertrapr`. The exact location of the folder will vary depending on OS, but you can get the exact location by running `path.expand("~/lasertrapr")` in the R-console. All of your data and analyses will be created and saved within this folder.

The app will take any uploaded data and copy it into an **observation** folder. In the laser trap, we will assume an "observation"is all the data collected at one mogul at a given time. This data, or observation, would have been collected on a specific day (date) under specific solution conditions (ionic strength, pH, mutations, etc.) and would belong to one bigger project. This logic provides the basis for the data management provided by the app. **The project, conditions, and date folders must exist in order for data to be uploaded.** The app will automatically create the observation folders when data is uploaded.

2.2 Create Folders

Located in the top right of the app is the **Folder Manager**. Click to open the menu (there is a known bug that when opening menu for first time some of the menu is cut off, you can click outside menu to close, and just re-opening will fix this menu display). Click the dropdown and either select your specific project folder or select "Create New…" to create a new folder. Avoid using spaces in the folder names. The prefix "project_" will be appended to the beginning. Continue to create folders in the same way for the conditions and date.

In lieu of using spaces, users are encouraged to use "-" and "under_score" in your conditions naming conventions. I prefer to use a combination of both. I use a "-" as a space *within* a given variable name and "under_score" as a space separator *between* distinct variables in the conditions. For instance, if I collected data with a wild-type myosin (WT) at pH 7.0 and 30mM Pi my conditions would be "WT_pH-7.0_30mM-Pi". The app will add a "conditions" column to all the uploaded trapping data to ID each observation from one another (along with columns for the project, date, and observation info). The benefit of having a standard naming convention and knowing what the roles of the special space seperators are will allow for more flexibility and robust analysis later on. For example, eventually the conditions column can be separated into *many* variable columns easily in the post-analysis by knowing that variables are separated by "under_score". It doesn't matter what you use, just be consistent.

2.3 Load Data

2.3.1 Simple Upload

A simple upload means the user has a complete file of trap data. One file = one complete record.

Any regular delimited file type can be loaded into the app (thanks to `fread()` from {data.table}). Currently, the app only supports single trap detector data which means only one signal columns can be used for analysis. **The first column in the data file will automatically be chosen as the trap data**. The app will make a copy of the data and re-format it for use within itself. Uploading a dataset to the app will create an "obs-##" folder inside the currently selected data folder. The data will be saved within that "obs-##" as "trap-data.csv". This last detail is purely informative. One of the benefit of using the app is not worrying about file management. You technically will never need to look inside the files created by the app, but it is still nice to know how it works and where to find things.

Multiple files can be uploaded at once too. Each file will be treated as a new observation. If three files are uploaded, then the folders "obs-01","obs-02", and "obs-03" will be created within the selected date folder and "trap-data.csv" files will be initialized from the uploaded data within the respective observation folders.

If you are uploading simulated data or data that has already been processed you can check the "Ready for Analysis?" box which will let you enter a trap stiffness (pN/nm) value and assumes you don't need to convert data from mV to nm (the app will use a value of 1 for the mV to nm conversion during subsequent analysis). This allows users to skip the “Clean & Process” section of the app since the data will be ready for analysis once the data is initialized.

2.3.2 Advanced Upload

If you are uploading multi-channel data or have a custom file header with all the system’s calibration information, you can also upload these files types. This would be the case for data coming from the Greenberg’s trap, where the trap data in X and Y from the two detectors is preceeded by a 68 line header with various calibrations and information about the current state of the system.

To upload these types of dataset, select “Number of Channels” for how many detectors your system has.

If there is a header in your data, check “Cal in header?” This will display a series of value boxes to fill out. The default values in these boxes are values that work with the trap data from the Greenberg lab’s trap. If you are using Greenberg lab data, you can just hit upload since everything is pre-filled.

Fill in the boxes with the LINE NUMBERS that correspond to your specific header file. Looking at the example, the “Hz” box has a value of 15. This means the sampling frequency information is on line 15 of the header.

Header Size: The number of lines in the header. The app will save the header in a separate file called “header.csv”.

Hz: sampling frequency. Enter the line number that this is found in the header.

nm/V and pN/nm: calibration numbers. Enter the line number that this is found in the header.

Feedback motor bead: value should be 1 or 2 denoting which trap was the motor bead when performing the isometric force clamp experiment.

Trap 1/2 Col: The column numbers where the actual data are after the header. Default is 1, 3 which means that the 1st and 3rd column have the data in the X dimensions recorded by the QPD.

2.3.3 Lumicks

You can upload Lumicks H5 data files into the app as well. Just select the Lumicks tab and browse for the H5 files. Currently, this will only extract the high frequency Trap1X and Trap2X data. This is sampled at ~78 kHz, so you will most likely want to downsample this. Input the factor number you want to downsample by (e.g. 2 would mean to downsample by a factor of 2, cutting the number of datapoints in half).

You can also include the position of the stage by checking the box.

2.3.4 Split Observations

This is a special use case for the Debold Lab. The trapping computer saves a separate .txt file for every 5-seconds of data collected. All .txt files from a given conditions/date combination can be uploaded at once and the app will read the time-stamps and concatenate corresponding observation files together based upon the user selected time threshold.

2.4 tl;dr

Just show me a video…

Your browser does not support the video tag.

Chapter 3

Clean and Process

3.1 Clean

Sometimes your filament velcros, snaps, you stickdown, or you accidentally forgot to stop recording. There are many reasons that you may need to clean an optical trapping data set.

One of the main benefits of `{lasertrapr}` is the ability to easily clean your data. Even if you do not like/need the analyzers or other features in the app, you can easily just use it to clean, process, and export your data to another application.

3.1.1 Cut data

The most common use case for needing to cut data from a trap data trace is when during collection an actin-filament snaps or the myosin sticks down. In these cases, there is still good/usable data present in the trace, but the presence of the large signal disruptions caused by the snapping filament or stick-down could throw off the analyzers. The easiest fix is to cut these portion of the data out. I generally do **not** recommend deleting data except for these cases in which case I refer to this as "trimming" the data.

Trimming (deleting) an observation to make it analyzer ready is easy with `{lasertrapr}`. Use the **Folder Manager** to select an observation, load the observation, select the data to delete, and hit the Cut button. *NOTE: This permanently deletes the range of data selected from the trap trace and is irreversible (unless you re-upload your data).*

Your browser does not support the video tag.

3.1.2 Move data

In some cases, you do not want to delete data, but to split one record into 2 different observations. This is called "moving" data in `{lasertrapr}`. Sometimes, during collection stage drift occurs so the trace starts with a stable/horizontal time-series, but then over time the data starts to trend with time upwards in the y-dimensions turning the signal into a diagonal line. One way to deal with this is to split the single **obs** into two seperate ones so the two-halves can be processed separately with the diagonal potion getting detrended later.

Moving data is the same procedure as cutting data, except for the final button pushed! Load an **obs**, select the data to move, and click move. A new observation folder will be made with the selected data and the selected data will be deleted out of the current **obs**. *NOTE: This cannot be undone without manual intervention (you would have to load the trap-data.csv files into R and `rbind` them back together or re-upload the data and start again).* Your browser does not support the video tag.

3.2 Process

Another benefit that I have enjoyed while analyzing my own data with `{lasertrapr}` is the ability to easily visualize how processing will transform my data before deciding to save/analyze it. Currently, you can convert data from mV-to-nm with a pre-determined user conversion value, center the baseline mean to zero using either the "baseline range" or "remove mv" techniques, or you can detrend your data with a peice-wise linear detrend-er.

3.2.1 Convert to nm

Short and sweet. Enter your pre-determined mV-to-nm conversion in the **Step Cal** box and hit **Graph** to preview.

Your browser does not support the video tag.

3.2.2 Remove baseline

When collecting laser trap data the detector is measuring the relative intensity of light across its four-quadrants. The data is saved in units of volts (V or mV) and is usually not centered around 0 V (unless your detector is on a translatable mounting bracket that allows the user to 0 out the detector manually). So, when the data is converted to nanometers the y-axis range becomes some arbitrarily large or nonsense negative value. Technically there is nothing wrong with this since we are interested in making *relative* measurements of displacements from baseline, but it makes more sense and is easier to read when the y-axis is centered around 0nm. This can be accomplished by calculating the average position of the baseline signal and subtracting that value from every point in the y-dimension. Baseline removal is currently implemented in 3 ways: baseline range, remove

MV, and de-trending the data. Currently, these all work for 1-channel data sets. Only the baseline range and de-trending are implements for 2-channel data sets.

3.2.2.1 Range

The baseline range is simplest and the most "legacy" (i.e. this was easiest for me to implement when I was doing this all manually before `{lasertrapr}`). You can manually select a quiescient perioed of data that represents the baseline signal and they mean position of this period of data will be calculated. By selecting **remove base** from the **Graph Options** and hitting **Graph** to update the app will provide a preview. *NOTE: this will not be saved until you explicitly hit save.*

Your browser does not support the video tag.

3.2.2.2 Remove MV

Sometimes it can be tricky to find a nice quiescent period of baseline signal to calcualte the range. This is expecially the case with fast motors and mini-ensemble experiments. Instead it can be helpful to use the **Remove MV** option. This will perform a Mean-Variance transformation of the entire data trace and show the plot in an interactive window. You can then select the area that represents the baseline population, the mean is calculated, and by selecting **Remove MV** in **Graph Options** and hitting **Graph** to update, the app will provide a preview. *NOTE: this will not be saved until you explicitly hit save.*

Your browser does not support the video tag.

3.2.3 Detrend data

Stage drift can occur in longer records, or put another way the displacement on the y-axis will start to trend with time on the x-axis. There should be no relationship between time and displacement (slope should be 0). If this occurs the data record will look like it is tilted diagonally. This can be compensated by *de*-trending the data. A piecewise linear regression is fit to every 5 seconds of data and the resulting slope is removed from the data. Select **Detrend** in **Graph Options** and click **Graph** to preview the results. This also centers the baseline around 0. *NOTE: this will not be saved until you explicitly hit save.*

Your browser does not support the video tag.

3.2.4 To Include, or not to Include...

I do not like deleting data, but I also do not like wasting my time. Unfortunately, not all that glitters is gold, or not all trap data that is collected is usable. If I **know** that data does not look like exceptional signal-to-noise, there are no events, or will probably not analyze well I want to exclude those events from analysis so they do not take time getting analyzed etc.

By default, `{lasertrapr}` excludes all data from analysis so you need to **Include** the data for the app to analyze it.

If you like the data check the **Include** button when saving data.

Your browser does not support the video tag.

3.2.5 Save!

NOTE: The app will not save anything unless you save the changes!

Chapter 4

Analyzers

Currently, there are several analyzers available within the app. The HM-Model/Changepoint analyzer was designed for single molecule trapping data and the mini-ensemble analyzer is self-explanatory. Both analyzers have a similar UI with the ability to impose user control over some of the analysis parameters.

The covariance analyzer is for single molecule experiments collected on systems with 2 QPDs.

There is also an analyzer for the isometric force clamp, a feedback experiment.

For all analyzers, you have the option to run the analyzer through all of the observations in a given date folder, or you can select just a single observation. If you select to analyze a single observation you must also select the observation you want to analyze in the Folder Navigator (top right button).

4.1 HM-Model/Changepoint

Intended for single molecule data, this analyzer uses a combination approach to identify single molecule binding events. A Hidden Markov Model is implemented with the {depmixS4} R-package on a running window transformation of the data to estimate locations of binding events. Then a changepoint analysis is applied to a small subset of the original trapping data around the HM-Model estimated transition periods with the {changepoint} R-package to precisely choose the most probable data point (at the original sampling frequency) where the binding event occurred.

Within the HM-Model/Changepoint analyzer, users have control (to some extent) over most aspects of the analyzer including the running window transformations, some of the HM-Model, and the type of changepoint.

Clicking the “Options” button will open up a menu that will allow you to set the analysis option.

4.1.1 Options: HM-Model

The Hidden Markov Model analyzes a running window transformation of the original data trace. Here you can select the window width (in data points) and how you would like the window to progress. Users are referenced to Smith, Steffon, Simmons, and Sleep 2001 for further details on how to optimize the windows. Per their recommendations the default of the progression of the windows is by 1/2 overlap. **Note: not all window slide options have been tested. Possible bugs may exist and app crash potential**. You can just restart the app and try another option.

The “Channels” options lets you choose if you want the Hidden Markov Model to use both the running mean AND running variance transformations or just the running variance. A personal anecdotal recommendation is to use both the running mean and variance.

“EM Random Start” is `FALSE` (unchecked) by default. If `TRUE` (checked) the analyzer will use random number generation to start the EM-Algorithm.

4.1.2 Options: Changepoint

There are two sections the changepoint options. Since changepoint analysis is applied separately to the beginning and ends of the events so you can control the behavior of both. The default is to use the changepoint method “Mean/Var” which has the changepoint alogorithm use the mean and variance position to identify the most probable change. Whereas if “Variance” is selected a slider will appear allowing the user to select a window width for the running variance transformation. The changepoint will then look for a change in the mean signal position of the variance transformation to identify the most probable change.

4.1.3 Options: Displacements

Users can select one of two methods for peak displacements to be calculated. The “average” method calculates the mean signal position of the entire ID’d event minus the first and last 5ms. Alternatively, users can opt to use the “peak” method which returns a maximum value from a 5ms running mean of the ID’d event.

4.1.3.1 Options: Hz

Users need to specify the sampling frequency (in Hz) for proper conversion between data points and seconds. Defaults to 5000 Hz.

4.2 Mini-Ensemble

This mini-ensemble analyzer uses a simple thresholding method to ID events. Users can control the threshold parameters for the displacement and minimum time on as well as the running window width.

4.3 Isometric Force Clamp

The Isometric Force Clamp analyzer only works with 2-channel datasets. It works by applying a 2-state hidden markov model to the motor bead position to identify the binding events. The average force of the event and the corresponding lifetime are saved in the output files. After analysing all your data, you can use the “Summarize” tab to fit the Bell-bond equation to this resulting data.

4.3.1 Options: HM-Model

You can select the window width (in data points) and window slide, along with “EM Random Start” (see above).

4.4 tl;dr

Just show me a video…

Your browser does not support the video tag.

Chapter 5

Summarize

The app will automatically detect which type of analyzer was used to analyze each project and provide a project summary. Note, only one analyzer should be used for a specific project. If you have single molecule and mini ensemble data you are working on for a paper, these need to be divided into separate projects. The summarize feature will then run a specific summarize analysis dependent on the type of analysis performed on the data.

5.1 Single- and mini-ensemble standard mode

The summarize feature is designed to be more of a quick summary feature that provides a quick look at the summary statistics of your project with minimal user-implemented choices. I found it to be convenient when you just want a quick check on how a project is progressing wanting to see averages, standard errors, sample size, etc to update myself and the lab. You can change colors or re-reorder the factors for display purposes, but other than that it will create the same default statistical figures powered by the {ggstatsplot} package. This provides a quick statistical tasks to check on significance. Currently, conditions are grouped indivudally and performed in a "one-way" fashion. Click the link to visit the {ggstatsplot} website for insight into the meanings of all the statistical symbols. These quick summaries can be exported to standalone .html dashboards to share. In the future I would like to add support to export to .ppt files as well.

Summarizing data will read, filter, combine, and save all "measured-events.csv" into your "summary" folder within your project with the date, project name, and "all-measured-events.csv" as the identifying file name. The summarized data will also be saved in a similar fashion but as "summary-data.csv".

Note: The **split conditions** feature can be used to separate your **conditions** name into multiple unique variable ID's which can be useful for later use when creating plots. Split conditions only works if you follow the condition naming convention described in these docs. No spaces - EVER! Underscores "_" separate distinct varaibles and dashes "-" can be used as spaces within a variable. When selecting split conditions, *n* number of textboxes will appear for the number of variables present in your conditions name, which is solved by identifying the number of underscores present plus 1. You can then enter the variables names which will become column names in the data.

5.2 Isometric Force Clamp

The isometric force clamp summary analysis will plot the Force vs lifetime of the binding events and use maximum likelihood estimation to fit the Bell-bond equation as detailed in MEMLET. To perform this summary analysis, you need to select a project that has been analyzed with the Isometric Force Clamp analyzer and then the buttons will automatically populate in the "Control" box. You can select the "factor order", the minimum force to include in the plots/fits, the minimum time duration of events to include, and the colors of the resulting graphs.

Click summarize and the plots will appear eventually. This could take several seconds because it also calculates 95% confidence intervals around the parameter estimates using boostrapping techniques. These data are saved in the lasertrapr/summary folder. The parameter estimations and the bootstrap results can be uploaded in the MATLAB based computational tool to test for significant differences. Follow the instructions for "Script_hypothesis_testing.m". In the future, translating this script into R to bake into the app would be great to have.

5.3 tl;dr

Just show me a video…

Your browser does not support the video tag.

Chapter 6

Ensemble Average

Ensemble averaging in **{lasertrapr}** is divided into three distinct steps 1) preparing the ensembles, 2) averaging and fitting the ensembles, and 3) plotting. Separating these distinct tasks provides the user with more control over the prococess without having to re-run computationally long tasks repeatedly.

6.1 Prep Ensembles

An individual single molecule (myosin) event should be composed of two substeps (d1 & d2). Events can be aligned at the changepoint identified start or end of the events, temporally synchronized to the same duration, and then averaged together to create one "average" event. The user can control how the event synchronization occurs.

All the original trapping data traces are saved in the "trap-data.csv" files and their ID'd events are saved in "measured-events.csv". The "measured-events.csv" file not only includes displacements, event durations, etc., but also the changepoint ID'd datapoint for the start and end of each event. This information can be used for the ensemble averaging.

Ultimately, the "Prep Ensembles" button will create an "ensemble-data.csv" file in each **obs-##** folder that will have all of the individual forwards and backwards extended events that was constructed with the selected parameters described below in one file in long format.

This step usaully takes the most time computationally as it involved reading in every single "trap-data.csv" file, extracting the events, and writing the synchronized events back to the folder.

NOTE: Only observations that were **included**, analyzed as **success**, and whose **review** marked as TRUE will be included in the ensemble averages. Additionally, if the observation passes that first check then any events that were user excluded will be filtered out. Each time you re-prep your ensembles, all existing "ensemble-data.csv" files are **ERASED** and the data is re-read and prepped again. This allows the user to go back and exclude a trace or event from analysis and re-prep without leaving old straggling files behind.

6.1.1 Extending Forward

The goal with the forward ensemble average is to align the beginning of the events and "stretch" the ends of the short events to be equal to the longest event in time. The app lets the user choose how much (or little) time to use to extend the event. A 3ms "Avg of ms to extend forward" (I should probably come up with better names for these parameters) value means the position of the last 3ms of an events will be averaged and that resulting average value will be used (repeated) to extend the event out. The window size you select would probably be dependent on [ATP]. The higher the [ATP] the lower the window should be to decrease the likelihood of averaging pre-hitch (d1) displacement position. Whereas, at high [ATP] myosin will spend a longer time in a post-hitch (d2) final displacement position waiting for ATP to induce dissociation so you can be less aggressive in the size of the window in attempts to truly capture the average of that final d2 displacement.

6.1.2 Extending Backwards

Similar idea of the forwards, but for the backward ensemble average the goal is to capture the average position signal of the d1 pre-hitch position signal to extend the events. However, this is a little tricky. The changepoint algorithm used identifies the transition period going from un-bound to bound as a part of the event. This is the very brief period in time where the bead is moving through solution as it is displaced from the center of the trap. As a result, even though these datapoint are considered part of the event, they are not a true representation of the post-powerstroke/pre-hitch d1 position. The app gives the option to "skip" into the event before performing an average in effort to try to avoid averaging in these transition points. The "Number of ms to skip before s1 avg" (again not a catchy name...) lets the user decide how many ms to skip into each event before averaging. Similarly, the "Avg of ms to extend s1 backwards" allows the user to select how many ms to include in the average after skipping ahead.

6.2 Average & Fit Ensembles

Once the ensembled are prepped they can then be averaged and optionally fit with an exponential curve. The options allow the user to select the maximum amount of data to plot and whether to fit a single, double, or no exponential to the data. This part is relateively strighforward, just click "Avg Ensembles".

Averaging the ensmbles will automatically save the fit parameters under “lasertrapr/project/summary”.

6.3 Plot Ensembles

After the ensembles are averaged and optionally fit, you can navigate to the “Plot Options” tab to activate the graph. Prior to activating this tab only the Forward/Backwards Fit Parameter tables will be view-able.

Within the app, there is a limited selection of plot customizations that allow the user to tweak the appearance of the plot. You can provide custom labels to the facets, change facetting directions, shift the backwards ensembls undernearth the forwards to save on space, change theme size, along with some other minor features. Plots can be saved with “Save Plot” and figures are saved under the selected project folder inside “lasertrapr/project/summary/figures”.

NOTE: In order to get the custom labels to work you must select a factor order.

6.4 tl;dr

Just show me a video...

Your browser does not support the video tag.

Chapter 7

Figures

Currently experimental. Unsure whether this will be fully implemented ever. Scientists are generally quite picky about the appearances of their figures, so it is usually easiest to read in data in R/any other plotting software to have the most control over the appearance. This would be more of a quick plotting feature that provides minimal customization.

7.1 tl;dr

Just show me a video…

Your browser does not support the video tag.