Overview of Filters For Colloid Motion

Jordan Dehmel, 2023, jdehmel@outlook.com

Itemized Filtering Process

1) Read in raw input data
2) Drop all non-relevant columns
3) Filter out any tracks with **sub-Brownian displacements**
4) Filter out any tracks with **mean qualities below the 40th percentile**
5) Filter out any tracks which have **mean straight line speed which is 1.5 * (Inner Quartile Range) below the pre-filter frequency-wide mean for this value**
6) Filter out any tracks which have **track mean quality which is 1.5 * (Inner Quartile Range) below the pre-filter frequency-wide mean for this value**
7) Compute the post-filter means and standard deviations
8) Save output data to file

In total, all data must go through exactly **4 filters**, two of which compare it the control data. The other two filters compare it against itself.

Input

We take the following inputs from `trackMate` via `ImageJ`.

- LABEL
- TRACK_INDEX
- TRACK_ID
- NUMBER_SPOTS
- NUMBER_GAPS
- NUMBER_SPLITS
- NUMBER_MERGES
- NUMBER_COMPLEX
- LONGEST_GAP
- TRACK_DURATION
- TRACK_START
- TRACK_STOP
- TRACK_DISPLACEMENT
- TRACK_X_LOCATION
- TRACK_Y_LOCATION
- TRACK_Z_LOCATION
- TRACK_MEAN_SPEED
- TRACK_MAX_SPEED
- TRACK_MIN_SPEED
- TRACK_MEDIAN_SPEED
- TRACK_STD_SPEED

- TRACK_MEAN_QUALITY
- TOTAL_DISTANCE_TRAVELED
- MAX_DISTANCE_TRAVELED
- CONFINEMENT_RATIO
- MEAN_STRAIGHT_LINE_SPEED
- LINEARITY_OF_FORWARD_PROGRESSION
- MEAN_DIRECTIONAL_CHANGE_RATE

Of these, we deem the following relevant.

- TRACK_DISPLACEMENT
- TRACK_MEAN_SPEED
- TRACK_MEDIAN_SPEED
- TRACK_MEAN_QUALITY
- TOTAL_DISTANCE_TRAVELED
- MEAN_STRAIGHT_LINE_SPEED
- LINEARITY_OF_FORWARD_PROGRESSION

We will analyze and filter by these attributes.

Output

For a given particle size and voltage, we will record the following items as an average by applied frequency.

- TRACK_DISPLACEMENT
- TRACK_MEAN_SPEED
- TRACK_MEDIAN_SPEED
- TRACK_MEAN_QUALITY
- TOTAL_DISTANCE_TRAVELED
- MEAN_STRAIGHT_LINE_SPEED (pixels / frame)
- LINEARITY_OF_FORWARD_PROGRESSION
- TRACK_DISPLACEMENT_STD
- TRACK_MEAN_SPEED_STD
- TRACK_MEDIAN_SPEED_STD
- TRACK_MEAN_QUALITY_STD
- TOTAL_DISTANCE_TRAVELED_STD
- MEAN_STRAIGHT_LINE_SPEED_STD
- LINEARITY_OF_FORWARD_PROGRESSION_STD

We will also record the following data which was not contained in the input.

- INITIAL_TRACK_COUNT
- FILTERED_TRACK_COUNT
- STRAIGHT_LINE_SPEED_UM_PER_S

Quality Percentile Filtering

In an effort to remove data which is low-quality (for instance, tracks which were observed which did not exist) we can accept only data above a certain quality percentile. This is most important in control samples, which seem to have a large quantity of low-quality tracks. **We will be filtering out any tracks which are below 50th percentile in quality.** We could also use a raw threshold for quality, but this tends to filter out all tracks in some datasets.

Sub-Brownian Filtering

We also have the ability to filter any track with sub-Brownian (control) values in the following.

- Straight line speed
- Displacement
- Linearity
- Quality

We will be filtering any data which has sub-Brownian displacement.

Note: In previous attempts, filtering any sub-Brownian straight line speed values led to the erasure of all higher frequency data, so we will not be applying that. Filtering by linearity tends to not do much in our data.

Internal Filtering (Under 2 Standard Deviations or Under 1.5 Inner Quartile Range)

This technique can be applied to any track attribute (displacement, mean instantaneous speed, mean quality, total distance traveled, mean straight line speed, linearity of forward progression). **We will be applying the** $mean - (1.5 * IQR)$ **method to mean straight line speed and mean quality.**

With a given attribute, we can filter anything below 2 standard deviations of the mean, or we can filter anything below 1.5 inner quartile range (IQR) of the mean. **We will be using the 1.5 inner quartile range method.**

Source Code

All source code is available at github.com/jorbDehmel/physicsScripts. Not all included source code was written by me.