plot and analyse bt.py START **286 sloc** main() **VARIABLES** density, particles <-- extracted from s19 stats.txt to add to plot quantity <-- from s19 stats.txt either for density, particle get coords(...) number, or bulk semi-axes Get the coordinates <x,y,z> for each frame, as a || os.getcwd() for CoR mean, range and stdev. Returns the distance and stdev at disruption, and also the perijove. roche distance, error (in roche_distance), closest (perijove) <-- get coords(...) rs range <-- get_coords_range() rs_std, rs_average <-information <-- a row of data containing number of particles, the get_coords_avg() value of quantity, roche distance, errors, perijove, etc get coords() write results(information) **END** Open results.txt and write to it the plot coords(...) row of data. Produce a plot for mean and plot for 4th difference in stdev with matplotlib, and save them in the simulation directory. get coords range() get coords() Returns np array of the range in radius of each frame, through x,y,z data. threshold <-- a limiting value which determines whether or not the comet was get coords avg() disrupted Returns np array of the average in radius of each frame and np array of the stdev of each frame through x,y,z data. get coords() RETURN roche distance, NO exceed_threshold(...)? error, closest get cluster data() RETURN Boolean Opens out file.stuff.bt of each frame, extract mean/range/stdev and return a np get coords() array rbp coords containing those values. YES index disrupted <-get_index_disrupted(...) get cluster data() get coords() RETURN rbp_coords RETURN -1, 0, error, closest get index disrupted() RETURN index disrupted get index disrupted(...) Use np.diff(), which reveals a spike roughly where the comet seems to break apart, however, some simulation seems to have spikes occurring before the "correct" spike, therefore some logic was applied to try and get the most correct place this occurs, does not work 100% of the time, can implement more logic to improve on this. smooth std <-- np.diff(rs std, 4)</pre> If the difference between the biggest trough and peak is more than 10 (usually the biggest absolute value is where disruption occurs) IF smooth std.argmax() - smooth std.argmin() >= 10 index disrupted <-- get biggest_index()</pre> Else, the dip and peak occurred within the same rise and fall (@ disruption distance, o/w need something better), so we want the lower index (earliest spike/dip) ELSE

index disrupted <-- np.min(...argmin(), ...argmax())</pre>