**Software documentation**

**Folder: pupilanalysismodule-master**

Run\_analysis

run\_batched\_analysis

get\_eye\_activations -> *face\_model\_output*

get\_centers\_from\_4D\_predictions\_gaussian -> *coordinates of the eyes*

reshape\_centers *(just rescaling coordinates)*

\_smooth\_facetracks *(Savitzky–Golay filter to smooth face\_coord)*

get\_pupil\_segment -> *pupil activation (frame, height, width)*

add\_filtered\_traces

\_filter\_trace *(interpolate blinks)*

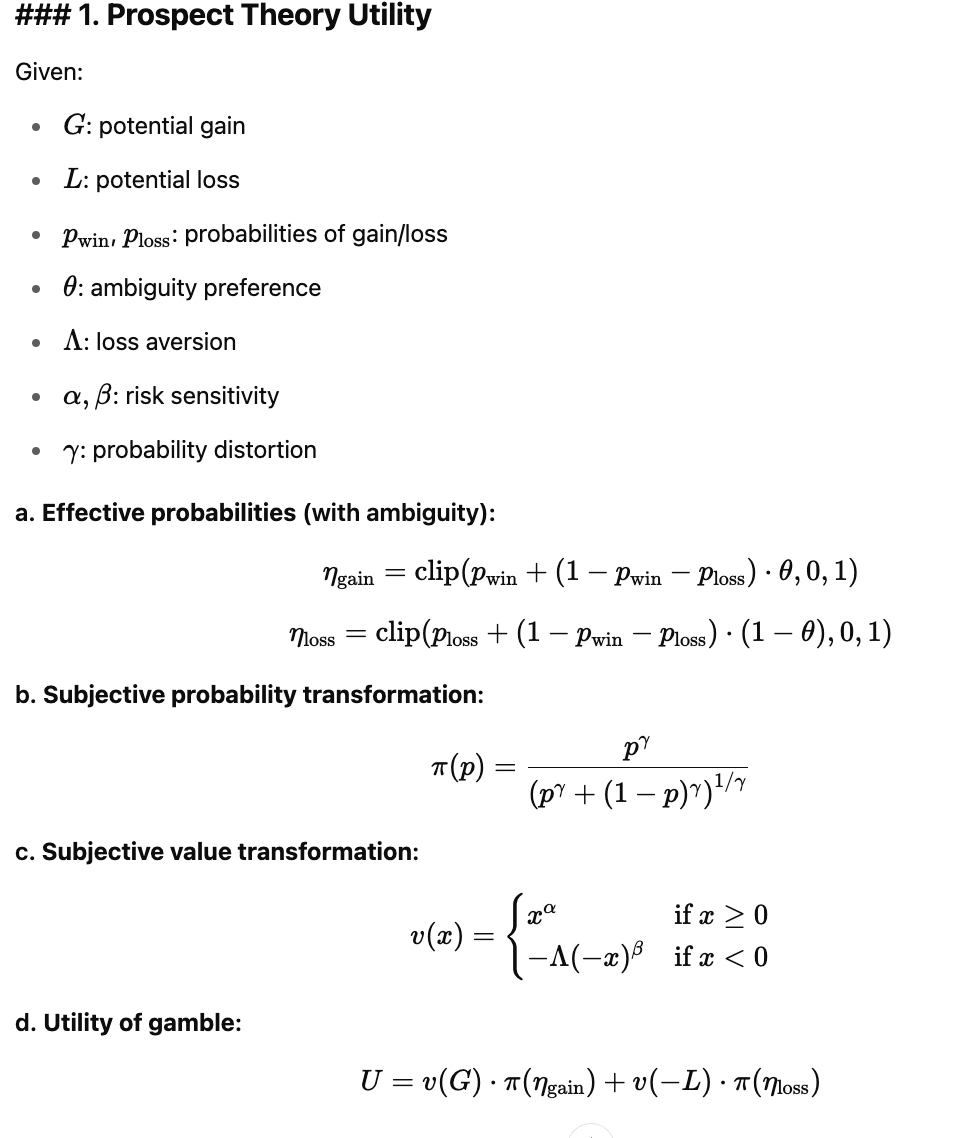
render\_result\_video

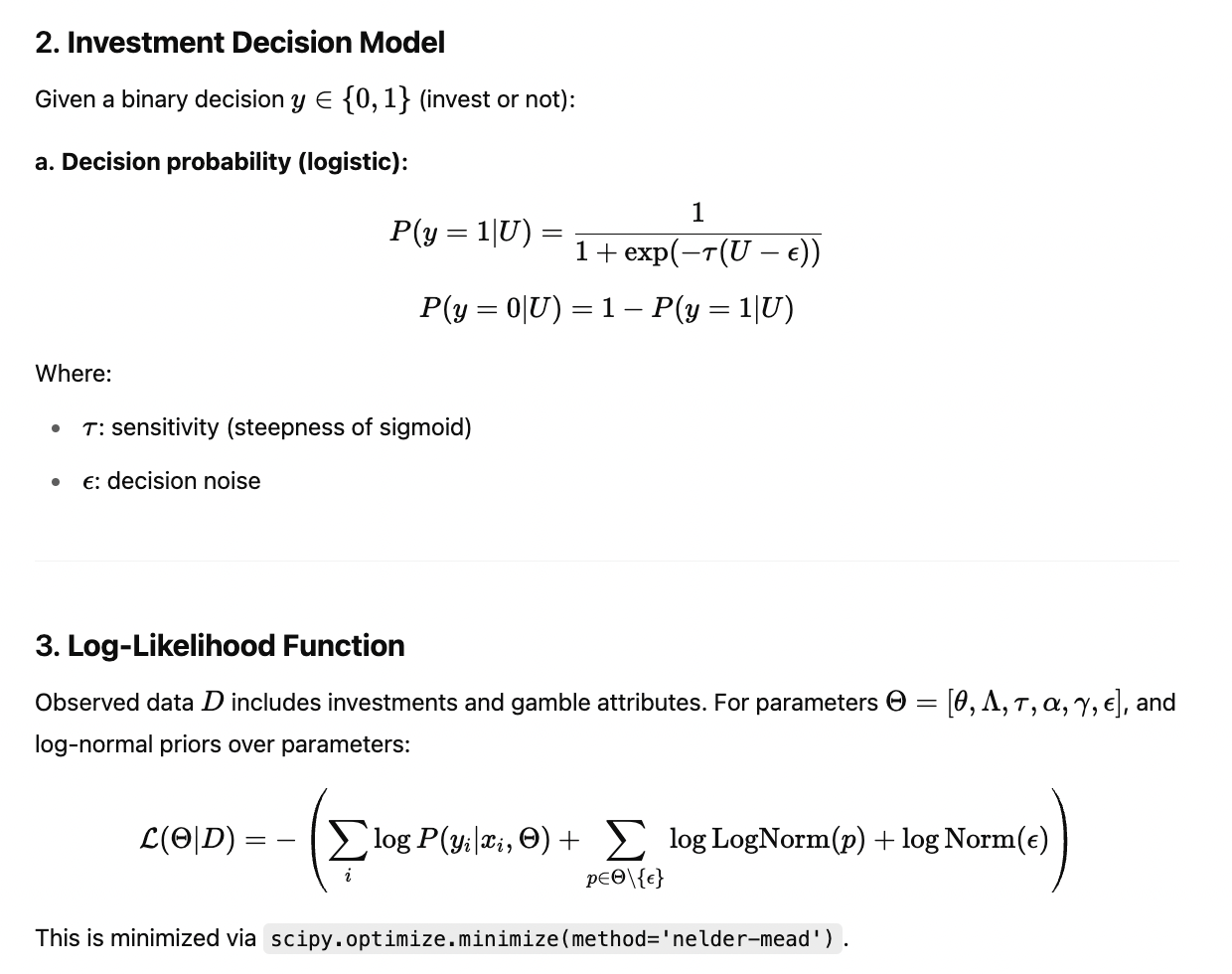
Folder:/Users/administrator/MGME/GDrive/6. Product Development/**6.5 Pupiltracking\_Prototype/code\_20231007**

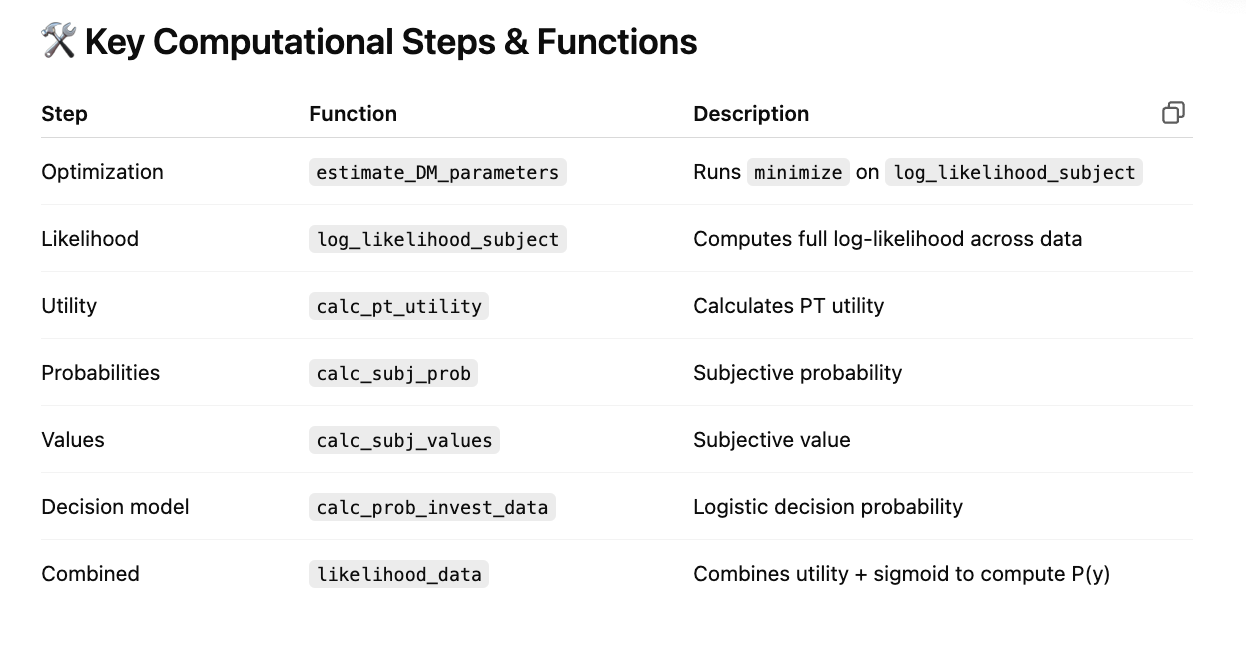
Testing code to generate the video of the tracked pupil superimposed on the video analyzed.

<https://docs.google.com/presentation/d/11epEdplT8RCtoDThMENqCWKAupCk6cgihHrg_8fSU90/edit#slide=id.g359a713fbb1_0_4>

**Personality Estimator**

****

****

****

**PersonalityScoreEstimator**

**analysis = PersonalityScoreEstimator(cfg, stroop\_input, dm\_self\_input, dm\_other\_input, self\_assessment\_results)**

**→ calculate\_personality\_dimensions(self):**

Take as input the outcome of the assessment as **dm\_single, dm\_self\_input**

→ estimate\_DM\_parameters(**dm\_single**)

Fits the model and estimated the model parameters: theta, lamba, alpha, gamma

→ calculate\_gaze\_scores(**dm\_self\_input**)

Extract eye gaze information and computes the anchoring info (fixations and their distribution)

→ calculate\_responsablity\_aversion(dm\_single, dm\_group)

It computes the difference between self and group task decision by category and return the responsibility aversion

→ def normalized\_persionality\_dimensions

Rescale these variables from in a predefined range [-100 100]

Reorders real and perceived values

DATA STRUCTURE DETAILS

**dm\_single.columns**

Index(['probability\_loss\_percent', 'probability\_win\_percent', 'potential\_gain',

'potential\_loss', 'invest', 'task\_onset\_at', 'user\_answered\_at'],

dtype='object')

**dm\_self\_input[0][0].keys() or dm\_self\_input[1][0].keys()**

dict\_keys(['frame', 'pupil\_trace\_raw', 'pupil\_trace\_filtered', 'eye\_tracks', 'eye\_tracks\_score', 'blink\_score', 'pupil\_center\_of\_mass', 'gaze\_position', 'time\_stamp\_ms'])

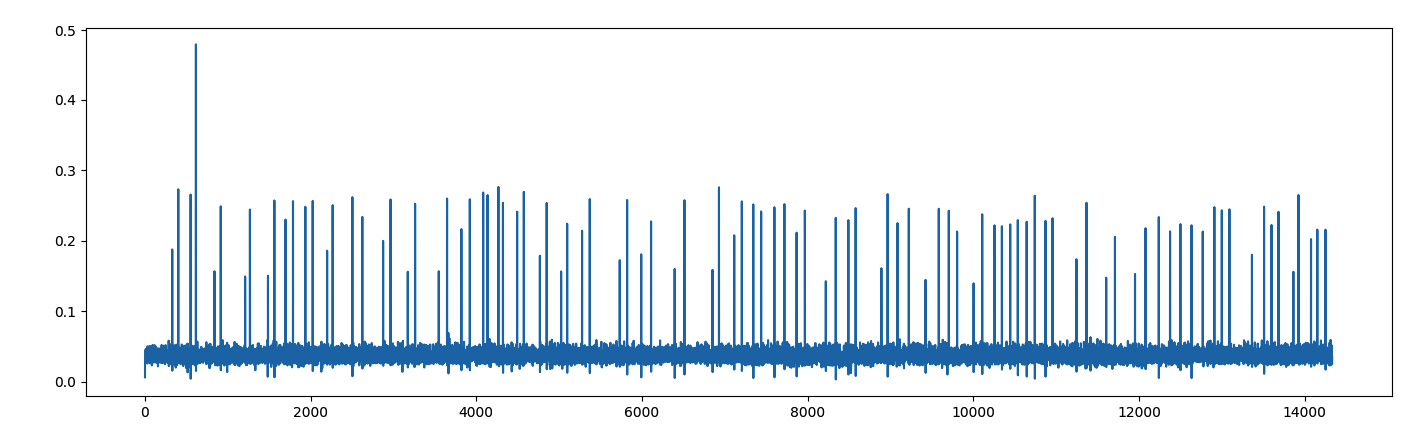
**dm\_self\_input[0][1].keys() or dm\_self\_input[1][1].keys()**

dict\_keys(['probability\_loss\_percent', 'probability\_win\_percent', 'potential\_gain', 'potential\_loss', 'invest', 'task\_onset\_at', 'user\_answered\_at'])

**dm\_self\_input[0][0]['time\_stamp\_ms']**

ranges between 0.43 and 495, size 14335 not consecutive frames, median around 0.033

**plt.plot(np.diff(dm\_self\_input[0][0]['time\_stamp\_ms']))**

****

**Dm\_self\_input[0][1]['task\_onset\_at']**

50 elements ranging between 0 and 500 s, median around 10 s

**Extrapolation of behavioral variables for the report:**

**scores['raw'].keys()** *# estimation\_list + perceived\_list*

dict\_keys(['theta', 'Lambda', 'alpha', 'gamma', 'information', 'neg\_consq', 'pos\_consq', 'likely\_fail', 'affecting\_others', 'being\_stressed'])

**!!! before normalizing, parameters value are assigned to their meaning:**theta ← score\_uncertainty

Lambda ← score\_neg\_outcome

alpha ← score\_pos\_outcome

gamma ← score\_risk

resp\_aversion ← score\_responsibility

stress\_resilience ← score\_stress

information ← perceived\_uncertainty

neg\_consq ← perceived\_neg\_outcome

pos\_consq ← perceived\_pos\_outcome

likely\_fail ← perceived\_risk

affecting\_others ← perceived\_responsibility

being\_stressed ← perceived\_stress