def add\_ts\_to\_srdf\_session(session\_metadata, rat\_srdf, parent\_directories):  
  
 phot\_proc\_folder = navigation\_utilities.find\_photometry\_processing\_folder(session\_metadata, parent\_directories)  
 pickled\_metadata\_fname = navigation\_utilities.get\_pickled\_metadata\_fname(session\_metadata, parent\_directories)  
 pickled\_analog\_processeddata\_fname = navigation\_utilities.get\_pickled\_analog\_processeddata\_fname(session\_metadata,  
 parent\_directories)  
 pickled\_ts\_fname = navigation\_utilities.get\_pickled\_ts\_fname(session\_metadata, parent\_directories)  
  
 # may update this to read in Kendall's processed files in the future. Her analyses would be in .mat files in phot\_proc\_folder,  
 # at least as of 6/12/2025  
 phot\_metadata = io\_utils.read\_pickle(pickled\_metadata\_fname)  
 processed\_analog = io\_utils.read\_pickle(pickled\_analog\_processeddata\_fname)  
  
 Fs = phot\_metadata['Fs']  
 n\_samples = np.shape(processed\_analog['dff'])[0]  
 session\_duration = n\_samples / Fs  
  
 ts\_dict = io\_utils.read\_pickle(pickled\_ts\_fname)  
 eventlist = list(ts\_dict.keys())  
 event\_ts = [ts\_dict[event][:, 0] for event in eventlist]  
  
 # collect all actuator3 events  
 act3\_idx = eventlist.index('Actuator3')  
 act3\_ts = event\_ts[act3\_idx]  
  
 # collect all vidtrigger events  
 vidtrig\_idx = eventlist.index('vid\_trigger')  
 vidtrigger\_ts = event\_ts[vidtrig\_idx]  
 sr\_ts = {'act3': act3\_ts, 'vidtrigger': vidtrigger\_ts}  
  
 trials\_df = add\_trial\_ts\_to\_srdf(rat\_srdf, session\_metadata, sr\_ts, session\_duration)  
  
 return trials\_df  
  
  
def add\_trial\_ts\_to\_srdf(rat\_srdf, session\_metadata, sr\_ts, session\_duration):  
  
 vidtrigger\_ts = sr\_ts['vidtrigger']  
 act3\_ts = sr\_ts['act3']  
  
 # find rows  
 session\_rows = (rat\_srdf['session\_date'] == session\_metadata['date']) & (rat\_srdf['date\_session\_num'] == session\_metadata['session\_num'])  
 session\_row\_idx = np.where(session\_rows)[0]  
  
 # if there are more session rows than vidtrigger events, that's because the task kept running after the photometry recording stopped  
 # if n\_vidtriggers is the number of vidtrigger events recorded in the photometry stream, assign the first n\_vidtrigger rows to have a vidtrigger timestamp  
  
 # also, there are some where there are more vidtrigger events than there are videos. This sometimes happens when there is a "line pop"  
 # I think I fixed this, but if it continues to happen, can look at video names for times.  
 if len(session\_row\_idx) < len(vidtrigger\_ts):  
 pass # to catch instances when there are more vidtrigger events than there should be  
  
 for i\_trial\_in\_session, row\_idx in enumerate(session\_row\_idx):  
 if i\_trial\_in\_session < len(vidtrigger\_ts):  
 # find the last actuator 3 timestamp before the current reach  
 trial\_act3\_ts = act3\_ts[act3\_ts < vidtrigger\_ts[i\_trial\_in\_session]]  
 if len(trial\_act3\_ts) > 0:  
 trial\_act3\_ts = trial\_act3\_ts[-1]  
 trial\_act3\_idx = np.where(act3\_ts == trial\_act3\_ts)[0][0]  
 else:  
 trial\_act3\_ts = 0  
 rat\_srdf.loc[row\_idx, 'act3\_ts'] = trial\_act3\_ts  
 rat\_srdf.loc[row\_idx, 'vidtrigger\_ts'] = vidtrigger\_ts[i\_trial\_in\_session]  
 rat\_srdf.loc[row\_idx, 'session\_type'] = session\_metadata['task']  
 rat\_srdf.loc[row\_idx, 'session\_duration'] = session\_duration  
 else:  
 rat\_srdf.loc[row\_idx, 'session\_type'] = session\_metadata['task']  
 rat\_srdf.loc[row\_idx, 'session\_duration'] = session\_duration  
 break  
  
 return rat\_srdf

CODE BELOW BUILDS THE TABLE/DATAFRAME WITHOUT EVENT TIMESTAMPS THAT CAN BE FILLED IN USING THE CODE ABOVE

def create\_rat\_srdf(ratID, parent\_directories, expt\_name):  
 *'''  
 0 – No pellet, mechanical failure  
 1 - First trial success (obtained pellet on initial limb advance). If more than one pellet on pedestal, successfully grabbing any pellet counts as success for scores 1 and 2  
 2 - Success (obtain pellet, but not on first attempt)  
 3 - Forelimb advance - pellet dropped in box  
 4 - Forelimb advance - pellet knocked off shelf  
 5 - Obtained pellet with tongue  
 6 - Walked away without forelimb advance, no forelimb advance  
 7 - Reached, pellet remains on shelf  
 8 - Used only contralateral paw  
 9 – Laser/video fired at the wrong time  
 10 – Used preferred paw after obtaining or moving pellet with tongue  
 11 – Obtained pellet with preferred paw after using non-preferred paw  
 '''* scores\_xlsx = navigation\_utilities.find\_scores\_xlsx(parent\_directories, expt\_name)  
 scores\_df = skilled\_reaching\_io.read\_xlsx\_scores(scores\_xlsx, ratID)  
  
 rat\_srdf = create\_empty\_skilledreaching\_dataframe()  
 # get all column headers in the scores\_df dataframe that include the ratID. Column headers should be "vid\_number"  
 # for the first column, then ratID\_YYYYMMDD\_xx where xx is 01, 02, etc. for each column with actual scores  
 session\_cols = [col for col in scores\_df.columns if ratID in col]  
 if scores\_df.empty:  
 # if this rat doesn't have any scores yet, just return an empty dataframe  
 return rat\_srdf  
  
 vid\_numbers = scores\_df['vid\_number']  
  
 num\_rows = scores\_df.shape[0]  
 trial\_num = 0  
 for i\_session, session in enumerate(session\_cols):  
 session\_scores = scores\_df[session]  
 session\_nameparts = session.split('\_')  
 session\_date = datetime.strptime(session\_nameparts[1], '%Y%m%d')  
 session\_type = '' # *todo: figure out whether this is a chrimson session* try:  
 date\_session\_num = int(session\_nameparts[2])  
 except:  
 pass  
 trial\_num\_in\_session = 0  
 vid\_number\_in\_session = 0  
 for i\_row in range(num\_rows):  
 if np.isnan(session\_scores[i\_row]):  
 # no video was recorded for this video number  
 continue  
 vid\_number\_in\_session += 1 # important to make sure we keep track of which timestamp goes with which video  
 if session\_scores[i\_row] in (6, 9):  
 # there was a video recorded but no reach; add a row so we can keep track, but don't count it as a reach  
 new\_row\_dict = {'trial\_num': np.nan,  
 'overall\_session\_num': i\_session,  
 'session\_date': session\_date,  
 'session\_type': session\_type,  
 'date\_session\_num': date\_session\_num,  
 'trial\_num\_in\_session': np.nan,  
 'vid\_number\_in\_session': vid\_number\_in\_session,  
 'vid\_number\_in\_name': vid\_numbers[i\_row],  
 'act3\_ts': np.nan,  
 'vidtrigger\_ts': np.nan,  
 'outcome': session\_scores[i\_row],  
 'act3\_interval': -2,  
 'vidtrigger\_interval': -2, # this is the valid signal interval in which this vidtrigger event occurred (-1 if it was outside a valid recording interval, -2 if not a valid video)  
 'session\_duration': 0.  
 }  
 else:  
 trial\_num += 1  
 trial\_num\_in\_session += 1  
 new\_row\_dict = {'trial\_num': trial\_num,  
 'overall\_session\_num': i\_session,  
 'session\_date': session\_date,  
 'session\_type': session\_type,  
 'date\_session\_num': date\_session\_num,  
 'trial\_num\_in\_session': trial\_num\_in\_session,  
 'vid\_number\_in\_session': vid\_number\_in\_session,  
 'vid\_number\_in\_name': vid\_numbers[i\_row],  
 'act3\_ts': np.nan,  
 'vidtrigger\_ts': np.nan,  
 'outcome': session\_scores[i\_row],  
 'act3\_interval': -3,  
 'vidtrigger\_interval': -3, # this is the valid signal interval in which this vidtrigger event occurred (-1 if it was outside a valid recording interval); placeholder because it hasn't been assigned yet  
 'session\_duration': 0.  
 }  
  
 new\_row = pd.DataFrame(data=new\_row\_dict, index=[0])  
 rat\_srdf = pd.concat([rat\_srdf, new\_row], ignore\_index=True)  
  
 return rat\_srdf