Meeting #8

10/28/21

Omar Luna

Deliverables

- Parse livedata.json for left and right pupil diameter
  - Index is time
  - columns are
    - Left pd
    - Right pd
- Sort rows by time
- Use participant.json file to add datetime stamp to rows

Methodology and Learnings

- json_to_dataframe(p_filename, d_filename)
- get_ptimestamp(filename)
- data_parsing(dict_data)
- get_data(filename, list_data)
- String to dictionary for parsing
  - Dictionary to dataframe
    - Faster method
  - Dictionary to string to dataframe
    - Dataframe.loc inefficient
- Datetime

Learn

- Dictionaries = fast
- Datetime format and functions

Results

```
def data_parsing(s):
  if "gidx" in s:
    if "pd" in s:
      data = [s["ts"], s["pd"], s["eye"]]
      return data
import json as js
import pandas as pd
import numpy as np

df_pd = pd.DataFrame(columns= ['ts', 'pd' ,'eye'])
with open('livedata.json') as f_livedata:
    for i, line in enumerate(f_livedata):
        #print(line)
        s = js.loads(line)
        l_data = data_parsing(s)
        if l_data!=None:
          df_pd.loc[len(df_pd.index)] = l_data
        else:
          continue
print(df_pd)
df_pd = df_pd.pivot(index=['ts'], columns= ['eye'])
print("pd DataFrame\n",df_pd)
```

```
              pd
eye         left right
ts
2225159387  0.00  4.50
2225169378  4.05  4.48
2225179369  0.00  4.51
2225189361  4.06  4.48
2225199356  0.00  4.52
...          ...   ...
2508082548  3.52  4.31
2508092537  3.48  4.05
2508102534  3.53  4.01
2508112525  3.53  3.94
2508122511  3.55  3.95

[28305 rows x 2 columns]
```

✓ 7m 31s

```
# Open file with data values and store them in lis
def get_data(filename, list_data):
  with open(filename) as f_livedata:
      for i, line in enumerate(f_livedata):
        p_data = data_parsing(js.loads(line))
        if p_data!=None:
          list_data.append(p_data)
        else:
          continue
```

```
                                pd
eye                           left right
2020-06-04 23:13:42.000000  0.00  4.50
2020-06-04 23:13:42.009991  4.05  4.48
2020-06-04 23:13:42.019982  0.00  4.51
2020-06-04 23:13:42.029973  4.06  4.48
2020-06-04 23:13:42.039964  0.00  4.52
...                          ...   ...
2020-06-04 23:18:24.745300  3.52  4.31
2020-06-04 23:18:24.755291  3.48  4.05
2020-06-04 23:18:24.765282  3.53  4.01
2020-06-04 23:18:24.775273  3.53  3.94
2020-06-04 23:18:24.785264  3.55  3.95

[28305 rows x 2 columns]
```

✓ 1s complete

```
def json_to_dataframe(participant_filename, data_filename):
```

```
# function returns initial timestamp from participant.json file
def get_ptimestamp(filename):
  with open(filename) as f:
    for i, line in enumerate(f):

      # find index of substring containing pa_created
      ss_index = line.find('pa_created')
      if ss_index > -1:

        # parse the timestamp and convert to datetime
        ts = line[ss_index+14: len(line)-2]
        dt_timestamp = dt.strptime(ts, "%Y-%m-%dT%H:%M:%S+%f" )
        return dt_timestamp

def data_parsing(dict_data):

  # searches for pd key in dictionary
  if "pd" in dict_data:

    # removes keys not used and returns dictionary
    del dict_data["s"], dict_data["gidx"],
    return dict_data



# Open file with data values and store them in list
def get_data(filename, list_data):
  with open(filename) as f_livedata:
      for i, line in enumerate(f_livedata):
        p_data = data_parsing(js.loads(line))
        if p_data!=None:
          list_data.append(p_data)
        else:
          continue
```

```
# List that contains pd data
list_data = []
# List that contains final datetime values
l_final_ts = []

# stores pd data in list_data from file
get_data(data_filename, list_data)
# returns initial timestamp from file
p_ts = get_ptimestamp(participant_filename)
# Creates initial dataframe
df_pd = pd.DataFrame.from_dict(list_data)
df_pd = df_pd.pivot(index=['ts'], columns= ['eye'])

# Converts initial datetime value to int
timestamp = int(round(p_ts.timestamp()))
# stores index values
index_vals = df_pd.index.values
# calculates the difference in seconds between values (1e6 = microseconds)
step = (index_vals[1] - index_vals[0])/1e6

# Converts int values to datetime values
for i in range(0, index_vals.size, 1):
  l_final_ts.append(dt.fromtimestamp(timestamp + (i*step)))

#final dataframe
df_final = pd.DataFrame(df_pd.values, columns= df_pd.columns,  index=l_final_ts)
return df_final
```

```
df = json_to_dataframe("participant.json", "livedata.json")
print(df)
```

```
                                pd
eye                           left right
2020-06-04 23:13:42.000000  0.00  4.50
2020-06-04 23:13:42.009991  4.05  4.48
2020-06-04 23:13:42.019982  0.00  4.51
2020-06-04 23:13:42.029973  4.06  4.48
2020-06-04 23:13:42.039964  0.00  4.52
...                          ...   ...
2020-06-04 23:18:24.745300  3.52  4.31
2020-06-04 23:18:24.755291  3.48  4.05
2020-06-04 23:18:24.765282  3.53  4.01
2020-06-04 23:18:24.775273  3.53  3.94
2020-06-04 23:18:24.785264  3.55  3.95

[28305 rows x 2 columns]
```