compile_gaze_data_sustained

September 16, 2019

1 Imports

[300]:
```
import pandas as pd; import seaborn as sb; import statsmodels.formula.api as␣
 ↪smf; import pingouin;
from itertools import combinations; import os; import seaborn as sb
from analysis_helpers import *; import scipy; from scipy import stats; import␣
 ↪os; import math; import scipy

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

2 Compile gaze data from files

Organize gaze data from each participant into one dataframe for the whole experiment.

[381]:
```
# Variable attention experiment
data_str = '../variable_attention_experiment/data'

exp_gaze = []

# load each participant's data as a dataframe, and append to a list
for idx,subject in enumerate(os.listdir(data_str)):
    exp_gaze.append(eye_initial(data_str + '/' + subject +'/eye_data/'))
    exp_gaze[idx]['Subject'] = subject.split('_',1)[0]
    print(idx)
```

```
0
1
2
3
4
5
6
7
```

8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

[382]:
```
exp_gaze = pd.concat(exp_gaze)
exp_gaze['Subject'].unique()
exp_gaze.to_csv('exp2_eye_initial.csv')
```

[383]:
```
# Sustained attention experiment
data_str = '../sustained_attention_experiment/data'

exp_gaze = []

# load each participant's data as a dataframe, and append to a list
for idx,subject in enumerate(os.listdir(data_str)):
    exp_gaze.append(eye_initial(data_str + '/' + subject +'/eye_data/'))
    exp_gaze[idx]['Subject'] = subject.split('_',1)[0]
    print(idx)
```

0
1
2
3
4
5
6
7
8
9

[384]:
```
exp_gaze = pd.concat(exp_gaze)
exp_gaze['Subject'].unique()
exp_gaze.to_csv('exp1_eye_initial.csv')
```

[475]:
```
exp2_sub_gaze = pd.read_csv('exp2_eye_initial.csv')
exp1_sub_gaze = pd.read_csv('exp1_eye_initial.csv')
```

3 Data Explore

[476]:
```
for exp_sub_gaze in [exp1_sub_gaze, exp2_sub_gaze]:
    print('Total number of gazepoints collected: '+str(exp_sub_gaze.shape[0]))
```

```
Total number of gazepoints collected: 2539798
Total number of gazepoints collected: 2600789
```

[477]:
```
for exp_sub_gaze in [exp1_sub_gaze, exp2_sub_gaze]:
    print('Average gazepoints per subject, across whole experiment:␣
 ↪'+str(exp_sub_gaze.groupby(['Subject']).count().mean().iloc[0]))
```

```
Average gazepoints per subject, across whole experiment: 84659.93333333333
Average gazepoints per subject, across whole experiment: 86692.96666666666
```

4 Explore States

We see that the data that has non-zero values is collected when the tracker is recording in State 7. As such, we eliminate the zero-valued gazepoints form other states

[478]:
```
for exp_sub_gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    for state in exp_sub_gaze['state'].unique():

        print('State: '+str(state))

        for coords in ['xRaw_righteye','xRaw_lefteye', 'yRaw_righteye',␣
↪'yRaw_lefteye']:

            unique = exp_sub_gaze[exp_sub_gaze['state']==state][coords].unique()
            to_print = 'unique '+coords+' '+str(len(unique))

            #if len(unique)<5:
            to_print = to_print +' (values: '+str(unique)+')'

            print(to_print)

        print()
    print('------------------------------')
    print()
```

```
State: 7
unique xRaw_righteye 1339374 (values: [904.8224 890.2061 640.9722 … 865.1318
863.5794 706.0865])
unique xRaw_lefteye 1297408 (values: [860.5577 839.7444 661.8791 … 846.0601
842.958  668.2572])
unique yRaw_righteye 1329194 (values: [237.2562 198.5501 337.994  … 449.376
442.1798 424.2562])
unique yRaw_lefteye 1304335 (values: [302.4497 276.5439 520.9327 … 321.5517
306.4873 331.6215])

State: 4
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])

State: 8
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])
```

```
State: 16
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])

----------------------------

State: 7
unique xRaw_righteye 1285721 (values: [ 1022.426   1019.7242  1012.9861 …
1565.7662 -1003.9531   117.7235])
unique xRaw_lefteye 1289411 (values: [1013.5128  999.8469  995.6805 …
542.4052  690.6105 2385.2949])
unique yRaw_righteye 1322630 (values: [ 343.5346  354.8827  442.9772 …
1191.2832 1441.9098  972.6319])
unique yRaw_lefteye 1339039 (values: [ 426.7873  448.6418  529.766  …
1175.5151  443.9463 1474.5623])

State: 8
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])

State: 4
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])

State: 16
unique xRaw_righteye 1 (values: [0.])
unique xRaw_lefteye 1 (values: [0.])
unique yRaw_righteye 1 (values: [0.])
unique yRaw_lefteye 1 (values: [0.])

----------------------------
```

5 Select data only from state 7

[481]:
```
exp1_sub_gaze = exp1_sub_gaze[exp1_sub_gaze['state']==7]
exp2_sub_gaze = exp2_sub_gaze[exp2_sub_gaze['state']==7]
```

6 Convert to centimeters

[482]:
```
# convert raw values to centimeters
exp1_sub_gaze = exp1_sub_gaze.apply(lambda x: x*(33.6/1152) if x.name in␣
↪['xRaw_righteye','xRaw_lefteye'] else x)
exp1_sub_gaze = exp1_sub_gaze.apply(lambda x: x*(59.8/2048) if x.name in␣
↪['yRaw_righteye','yRaw_lefteye'] else x)

# convert raw values to centimeters
exp2_sub_gaze = exp2_sub_gaze.apply(lambda x: x*(33.6/1152) if x.name in␣
↪['xRaw_righteye','xRaw_lefteye'] else x)
exp2_sub_gaze = exp2_sub_gaze.apply(lambda x: x*(59.8/2048) if x.name in␣
↪['yRaw_righteye','yRaw_lefteye'] else x)
```

7 Explore Values

7.0.1 zero vals

We saw, in `Explore States`, that when the tracker is not properly tracking, it records values of zeroes for both eyes. We want to eliminate this data. However, we don't want to eliminate actual data where participants may have stared directly at the corner of the screen.

First, let's explore the proportion of gazepoints indicating gaze of both eyes directly at point (0,0), relative to the proportion of gazepoints indicating gaze directly at each of the other three corners of the screen.

[483]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    for pair in [(0,0), (59.8,0), (59.8,33.6), (0,33.6)]:

        # proportion of gazepoints at 0,0
        prop = gaze[((gaze['xRaw_lefteye']==pair[0]) &␣
↪(gaze['xRaw_righteye']==pair[0]))
                    & ((gaze['yRaw_lefteye']==pair[1]) &␣
↪(gaze['yRaw_righteye']==pair[1]))].shape[0] / gaze.shape[0]

        print('Proportion of gazepoints with zeros: '+str(prop))
    print()
```

```
Proportion of gazepoints with zeros: 0.008514257392640646
Proportion of gazepoints with zeros: 0.0
Proportion of gazepoints with zeros: 0.0
Proportion of gazepoints with zeros: 0.0

Proportion of gazepoints with zeros: 0.008145257494457941
Proportion of gazepoints with zeros: 0.0
```

```
Proportion of gazepoints with zeros: 0.0
Proportion of gazepoints with zeros: 0.0
```

We find NO datapoints indicating direct gaze of both eyes directly at any corner of the screen other than at (0,0).

As such, we find it reaonable to exclude the cases where the data indicated both eyes are staring directly at the bottom left corner of the screen (0,0), which comprise 0.0085 proportion of the data.

[484]:
```
# select non-zero points
exp1_sub_gaze = exp1_sub_gaze[(~((exp1_sub_gaze['xRaw_righteye']==0) &␣
↪(exp1_sub_gaze['xRaw_lefteye']==0)))
         & (~((exp1_sub_gaze['yRaw_righteye']==0) &␣
↪(exp1_sub_gaze['yRaw_lefteye']==0)))]

exp2_sub_gaze = exp2_sub_gaze[(~((exp2_sub_gaze['xRaw_righteye']==0) &␣
↪(exp2_sub_gaze['xRaw_lefteye']==0)))
         & (~((exp2_sub_gaze['yRaw_righteye']==0) &␣
↪(exp2_sub_gaze['yRaw_lefteye']==0)))]
```

[485]:
```
exp2_sub_gaze.loc[exp2_sub_gaze['xRaw_righteye'] ==␣
↪exp2_sub_gaze['xRaw_righteye'].max()]
```

[485]:
```
       Unnamed: 0                                        avg    fix  \
20541       20541  {'x': 55553968.0, 'y': 1459324.625}  False

                                                 lefteye  \
20541  {'avg': {'x': 165952064.0, 'y': 4357050.5}, 'p…

                                          raw  \
20541  {'x': 393425760.0, 'y': -142217664.0}

                                              righteye  state       time  \
20541  {'avg': {'x': 128923960.0, 'y': -301705696.0},…      7  896893505

          timestamp  xRaw_righteye  yRaw_righteye   xRaw_lefteye  \
20541  1.551380e+09  109674.896528 -257232.579499  559695.313333

       yRaw_lefteye   av_x_coord   av_y_coord  Subject
20541  14724.630558  393425756.0 -142217657.0        8
```

[486]:
```
exp1_sub_gaze.loc[exp1_sub_gaze['xRaw_righteye'] ==␣
↪exp1_sub_gaze['xRaw_righteye'].max()]
```

[486]:
```
         Unnamed: 0                                        avg    fix  \
1211843       13168  {'x': 206530.625, 'y': -286836.625}  False
```

```
                                                      lefteye  \
1211843  {'avg': {'x': 0.0, 'y': 0.0}, 'pcenter': {'x':…

                                          raw  \
1211843  {'x': 943126.75, 'y': -864619.375}

                                                    righteye  state       time  \
1211843  {'avg': {'x': 350589.8125, 'y': -365397.7812},…         7  531859263

            timestamp  xRaw_righteye  yRaw_righteye  xRaw_lefteye  \
1211843  1.539204e+09     802.312687    -737.169616           0.0

         yRaw_lefteye  av_x_coord   av_y_coord  Subject
1211843           0.0  471563.375 -432309.6875       17
```

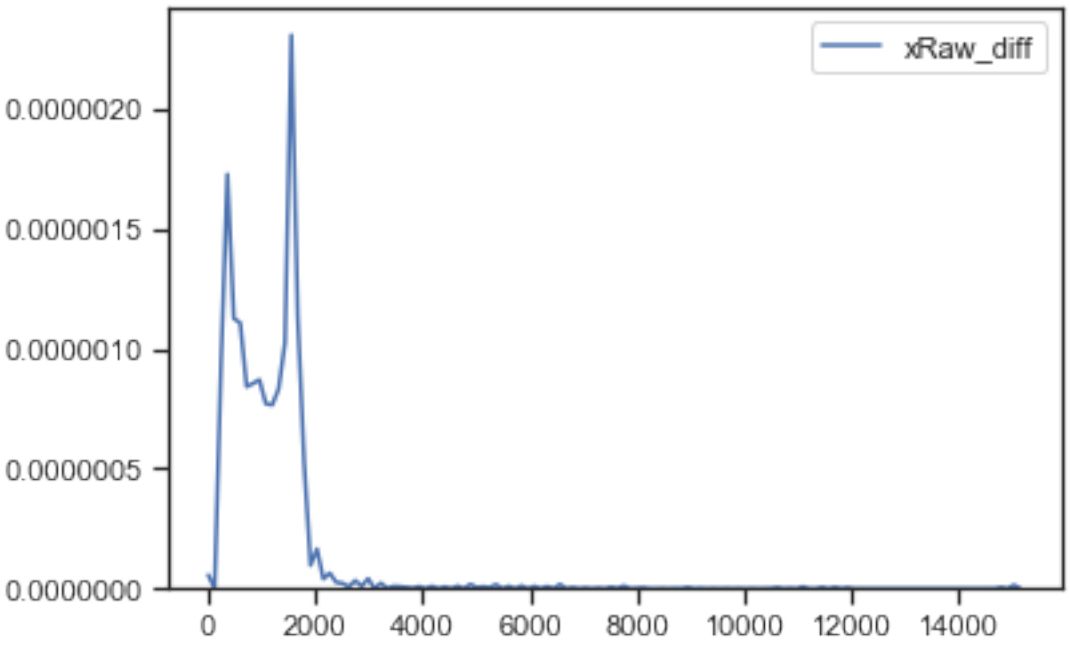
7.0.2 values that differ between eyes

Some gazepoints also have highly disparate values for the left and right eyes

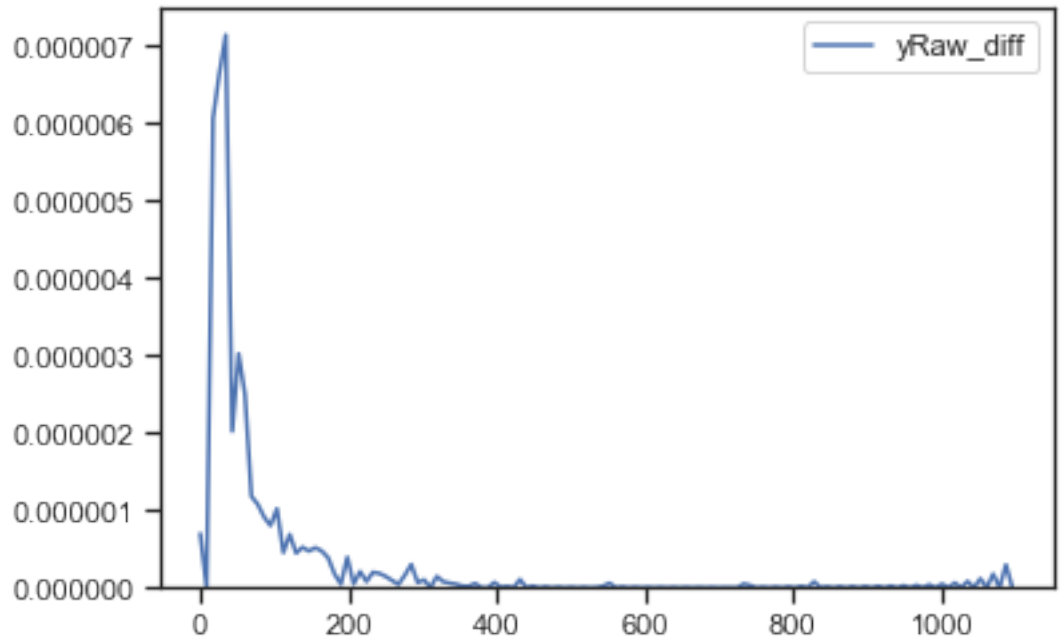
[487]:
```
exp1_sub_gaze['xRaw_diff'] = abs(exp1_sub_gaze['xRaw_righteye'] -␣
↪exp1_sub_gaze['xRaw_lefteye'])
exp1_sub_gaze['yRaw_diff'] = abs(exp1_sub_gaze['yRaw_righteye'] -␣
↪exp1_sub_gaze['yRaw_lefteye'])

exp2_sub_gaze['xRaw_diff'] = abs(exp2_sub_gaze['xRaw_righteye'] -␣
↪exp2_sub_gaze['xRaw_lefteye'])
exp2_sub_gaze['yRaw_diff'] = abs(exp2_sub_gaze['yRaw_righteye'] -␣
↪exp2_sub_gaze['yRaw_lefteye'])
```

[488]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    for diff in ['xRaw_diff','yRaw_diff']:

        print('Average diff: '+str(gaze[diff].mean()))
        print('Median diff: '+str(gaze[diff].median()))
        print('Mode diff: '+str(gaze[diff].mode()))

        sb.kdeplot(gaze[diff])
        plt.show()
    print()
    print()
```

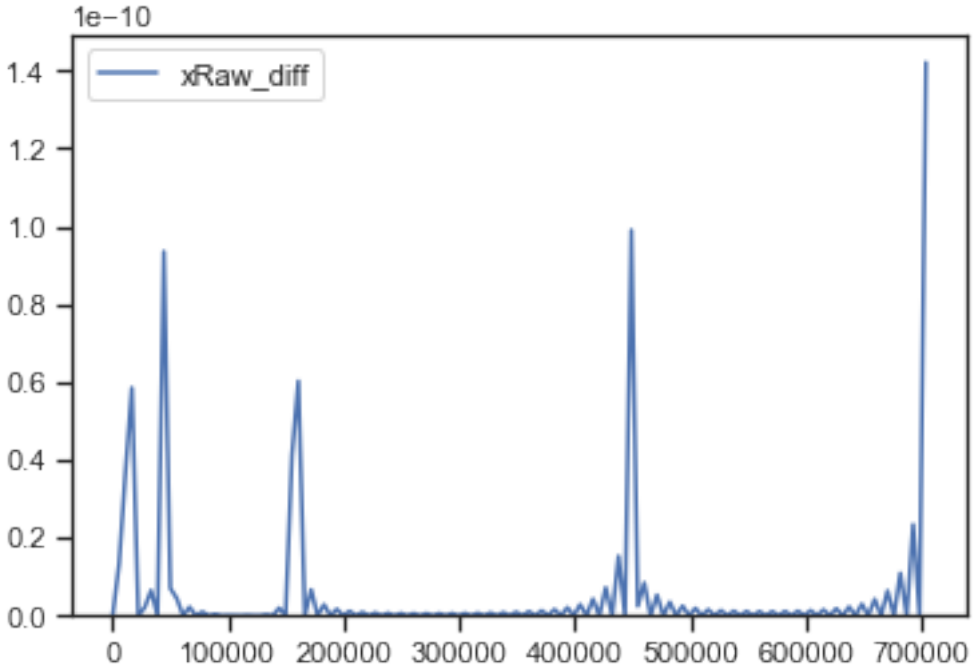
```
Average diff: 2.4156748111777477
Median diff: 0.05157658333333326
Mode diff: 0    0.022433
dtype: float64
```



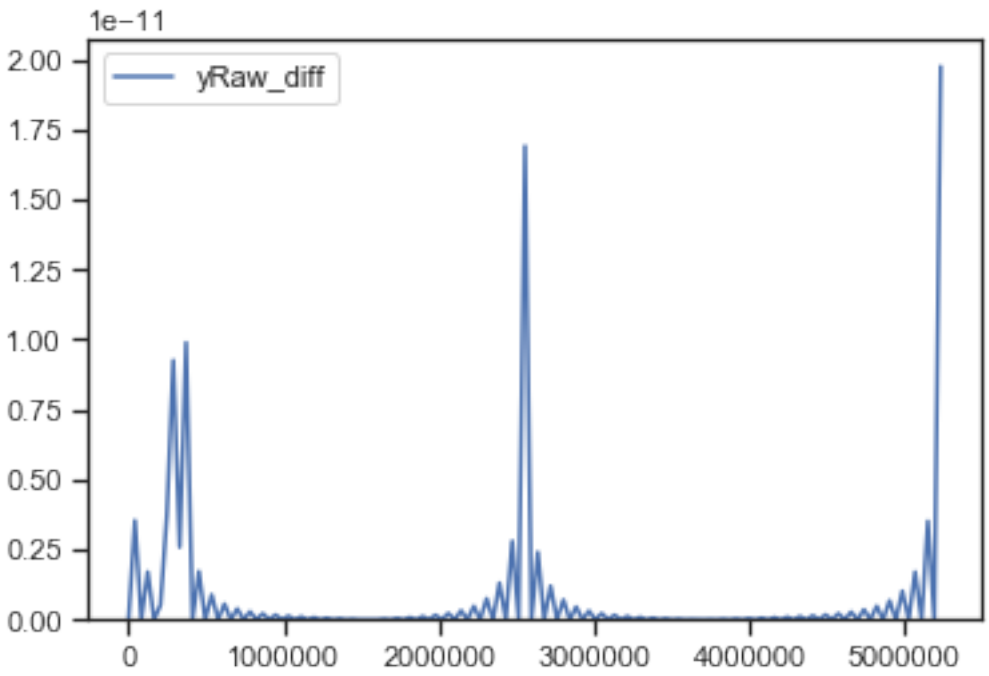

```
Average diff: 0.1350789187546988
Median diff: 0.033043829696655314
Mode diff: 0    0.007413
1    0.008291
2    0.022965
dtype: float64
```



```
Average diff: 0.7904109323088886
Median diff: 0.04599364583333332
Mode diff: 0    0.122412
dtype: float64
```




```
Average diff: 4.214185016394192
Median diff: 0.02770215541458121
Mode diff: 0    0.022444
dtype: float64
```



[402]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    val  = 5
    prop = gaze[(gaze['xRaw_diff']<=val)&(gaze['yRaw_diff']<=val)].shape[0] /␣
↪gaze.shape[0]

    print(str(prop)+' proportion of datapoints indicate a difference in gaze␣
↪location between the left and right eyes of '+str(val)+'cm or less.')
    print()
```

```
0.801889979170475 proportion of datapoints indicate a difference in gaze
location between the left and right eyes of 5cm or less.

0.8444420097727721 proportion of datapoints indicate a difference in gaze
location between the left and right eyes of 5cm or less.
```

Inter eye gaze distance greater than 5 cm..

[403]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    gaze_dist = gaze[(gaze['xRaw_diff']>5)&(gaze['yRaw_diff']>5)]
```

```
    print(gaze_dist['xRaw_lefteye'].mean())
    print(gaze_dist['xRaw_righteye'].mean())
    print(gaze_dist['yRaw_righteye'].mean())
    print(gaze_dist['yRaw_lefteye'].mean())
    print()
    print()
```

```
330.7853302236626
12.221910881672978
13.069102944886618
9.634451131836517


185.83050269610322
-84.02140022454319
886.9640578278298
1421.6933575486746
```

[404]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    gaze_dist = gaze[(gaze['xRaw_diff']<=5)&(gaze['yRaw_diff']<=5)]
    print(gaze_dist['xRaw_lefteye'].mean())
    print(gaze_dist['xRaw_righteye'].mean())
    print(gaze_dist['yRaw_righteye'].mean())
    print(gaze_dist['yRaw_lefteye'].mean())
    print()
    print()
```

```
27.98012010747185
28.29960986034673
16.89912582930029
16.972018880874337


29.63385453298211
29.315436309397896
16.871156622437443
16.824744924257292
```

Select gazepoints where eyes are not reported as gazing far apart from each other

[405]:
```
exp1_sub_gaze = exp1_sub_gaze[(exp1_sub_gaze['xRaw_diff']<=5)&(exp1_sub_gaze['yRaw_diff']<=5)]
```

```
exp2_sub_gaze =␣
 ↪exp2_sub_gaze[(exp2_sub_gaze['xRaw_diff']<=5)&(exp2_sub_gaze['yRaw_diff']<=5)]
```

8 Take average gaze coords

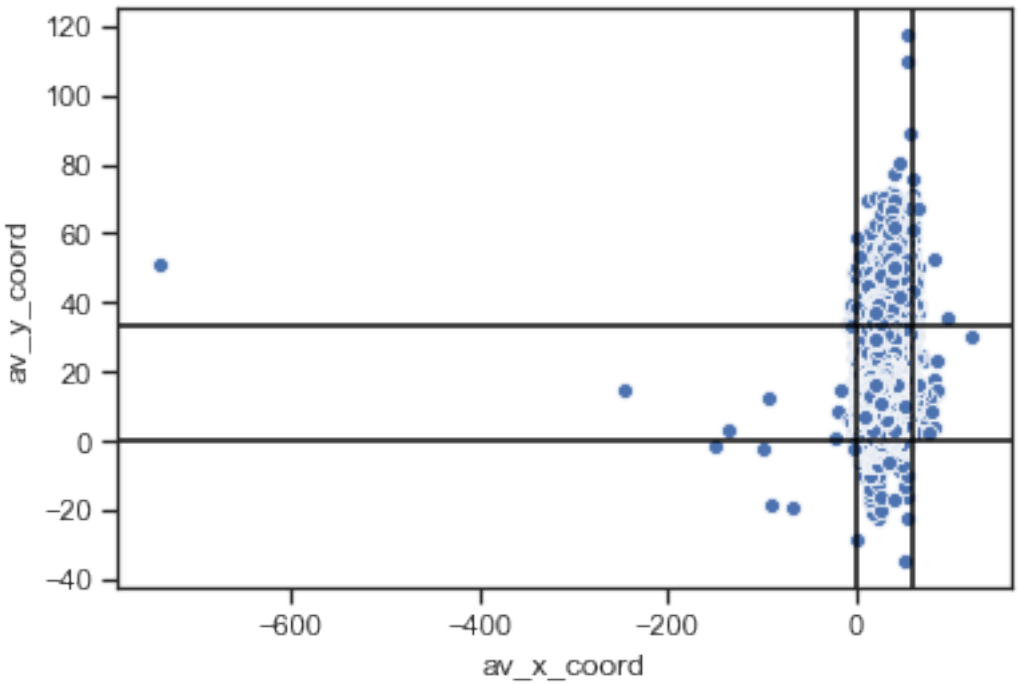
[406]:
```
# take the mean
exp1_sub_gaze['av_x_coord'] = exp1_sub_gaze[['xRaw_righteye', 'xRaw_lefteye']].
 ↪mean(axis=1)
exp1_sub_gaze['av_y_coord'] = exp1_sub_gaze[['yRaw_righteye', 'yRaw_lefteye']].
 ↪mean(axis=1)

# take the mean
exp2_sub_gaze['av_x_coord'] = exp2_sub_gaze[['xRaw_righteye', 'xRaw_lefteye']].
 ↪mean(axis=1)
exp2_sub_gaze['av_y_coord'] = exp2_sub_gaze[['yRaw_righteye', 'yRaw_lefteye']].
 ↪mean(axis=1)
```

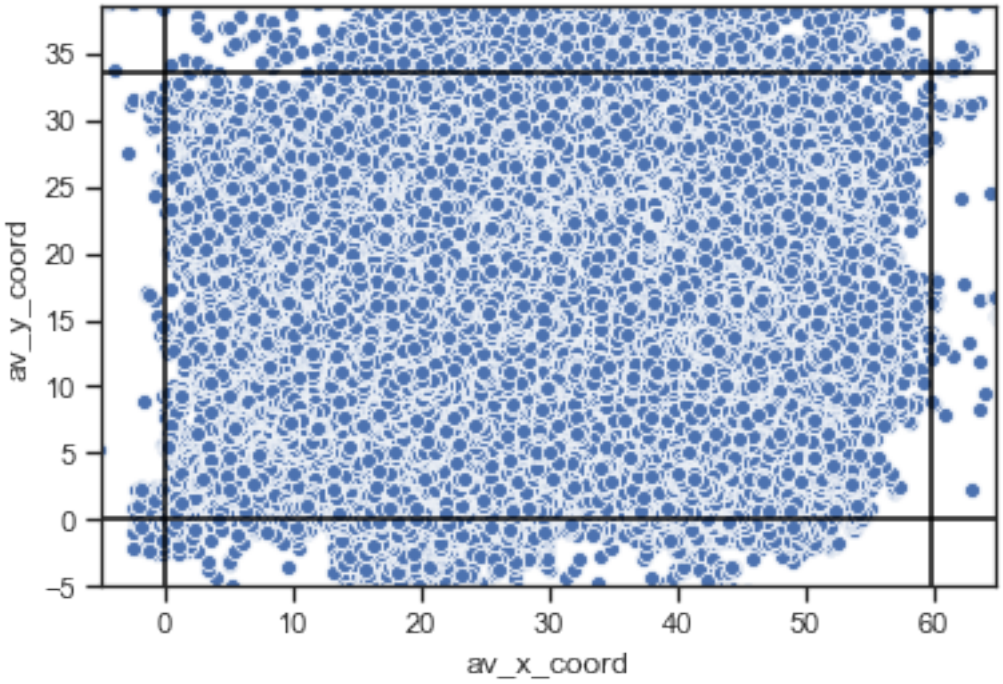
8.0.1 Plot of gaze data from full experiment

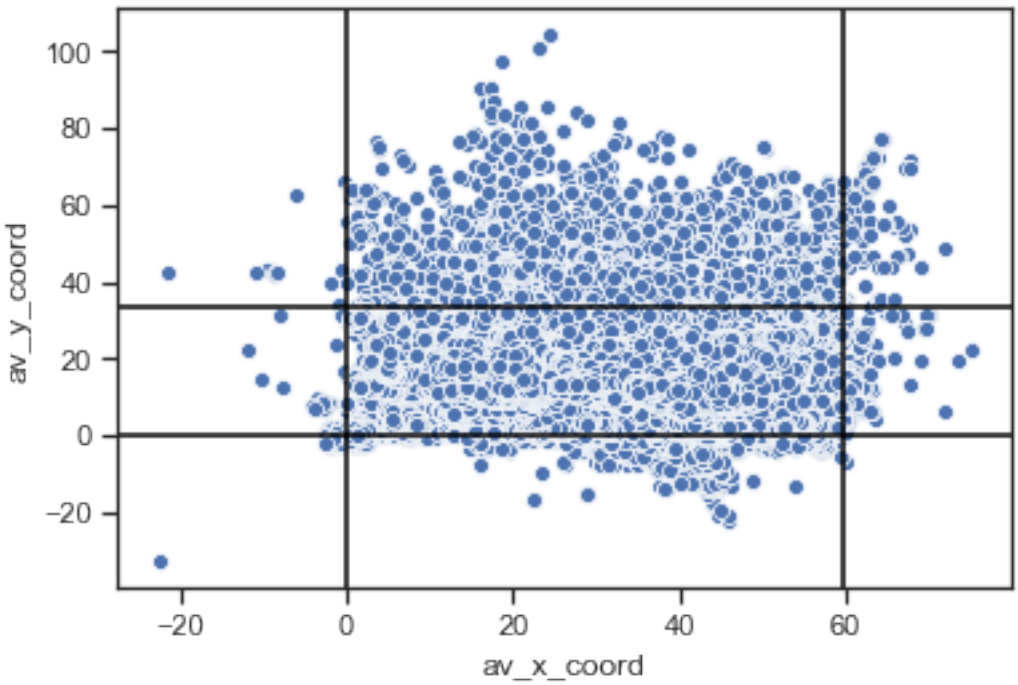
This is all of the gaze data collected (including while participants read instructions, etc.).

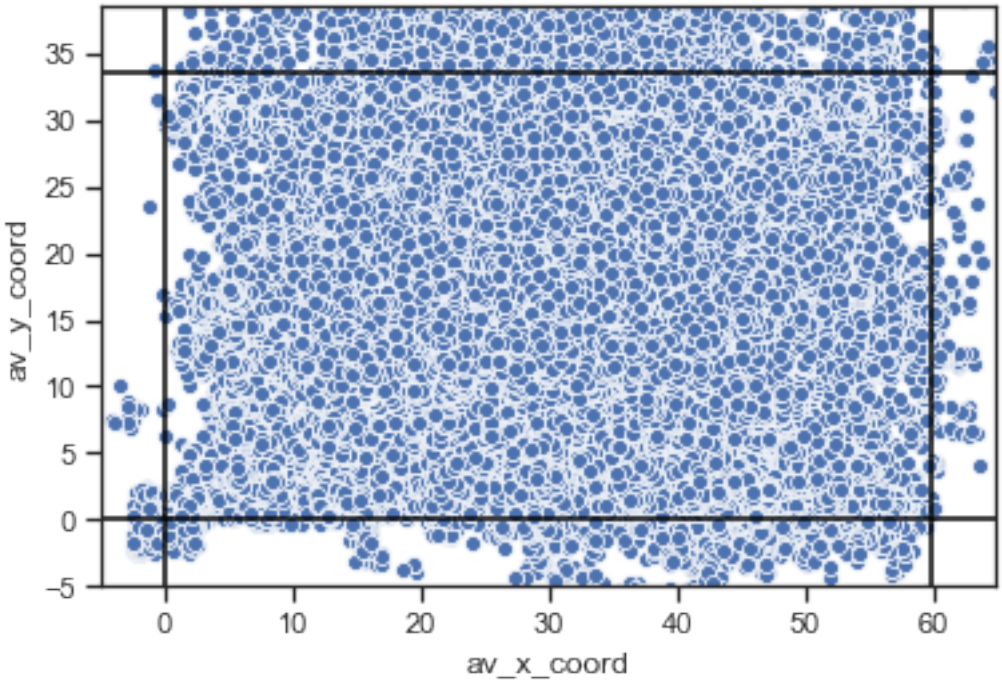
The first plot shows the full dataset. The second plot is scaled to show datapoints falling closer to the actual screen.

[407]:
```
for gaze in [exp1_sub_gaze, exp2_sub_gaze]:

    ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = gaze)
    ax.axhline(0, color='black'); ax.axhline(33.6, color='black')
    ax.axvline(0, color='black'); ax.axvline(59.8, color='black')
    plt.show()

    ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = gaze)
    ax.axhline(0, color='black'); ax.axhline(33.6, color='black')
    ax.axvline(0, color='black'); ax.axvline(59.8, color='black')
    ax.set_ylim(-5, 38.6)
    ax.set_xlim(-5, 64.8)
    plt.show()
```


120
100
80
60
40
20
0
−20
−40
av_y_coord
−600
−400
−200
0
av_x_coord


35
30
25
20
15
10
5
0
−5
av_y_coord
0
10
20
30
40
50
60
av_x_coord


100
80
60
40
20
0
−20
av_y_coord
−20
0
20
40
60
av_x_coord


35
30
25
20
15
10
5
0
−5
av_y_coord
0
10
20
30
40
50
60
av_x_coord

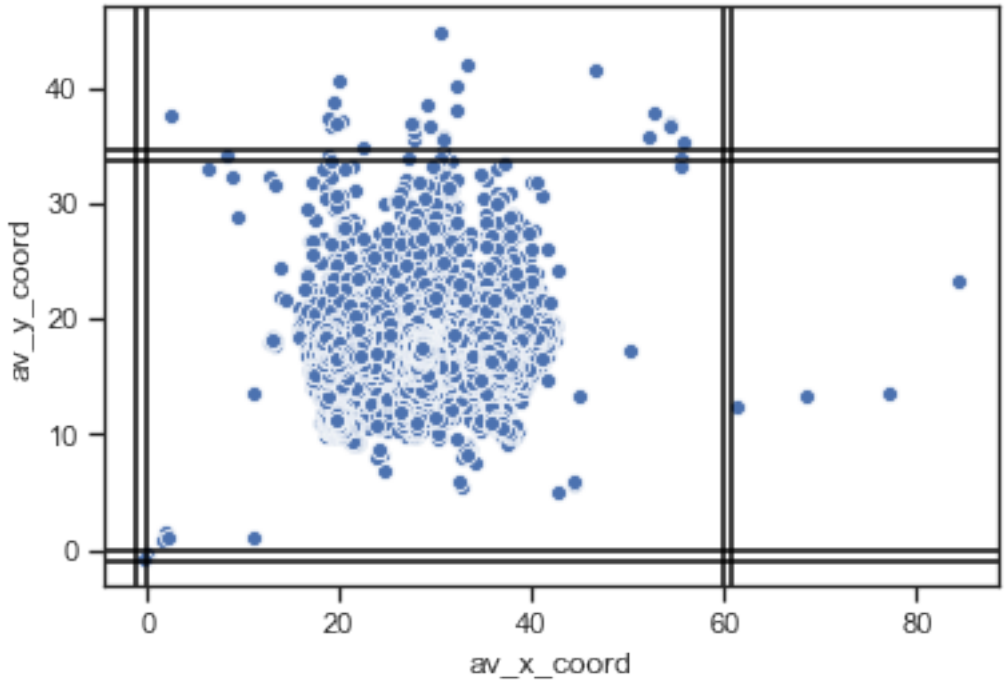
[410]:
```
# select only the gazepoints from times when presentation images were on screen
exp1_pres = []
for sub_file in os.listdir('../sustained_attention_experiment/data'):
    exp1_pres.append(pres_gaze_image('../sustained_attention_experiment/data' +␣
 ↪'/' + sub_file +'/',
                                     exp1_sub_gaze[exp1_sub_gaze['Subject']==␣
 ↪int( sub_file.split('_',1)[0] )]))
pres_gaze_1 = pd.concat(exp1_pres)


# select only the gazepoints from times when presentation images were on screen
exp2_pres = []
for sub_file in os.listdir('../variable_attention_experiment/data'):
    exp2_pres.append(pres_gaze_image('../variable_attention_experiment/data' +␣
 ↪'/' + sub_file +'/',
                                                 ␣
 ↪exp2_sub_gaze[exp2_sub_gaze['Subject']== int( sub_file.split('_',1)[0] )]))
pres_gaze_2 = pd.concat(exp2_pres)
```

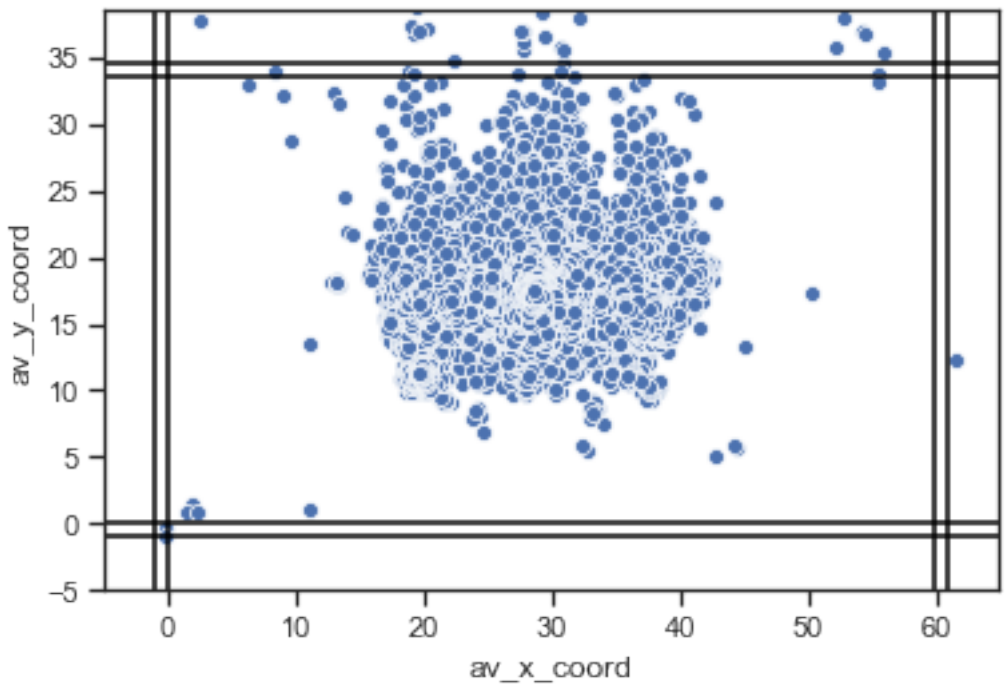
8.0.2 Plot of gaze data exclusively from when images on screen

Lines represent 1 cm width around the edge of our screen

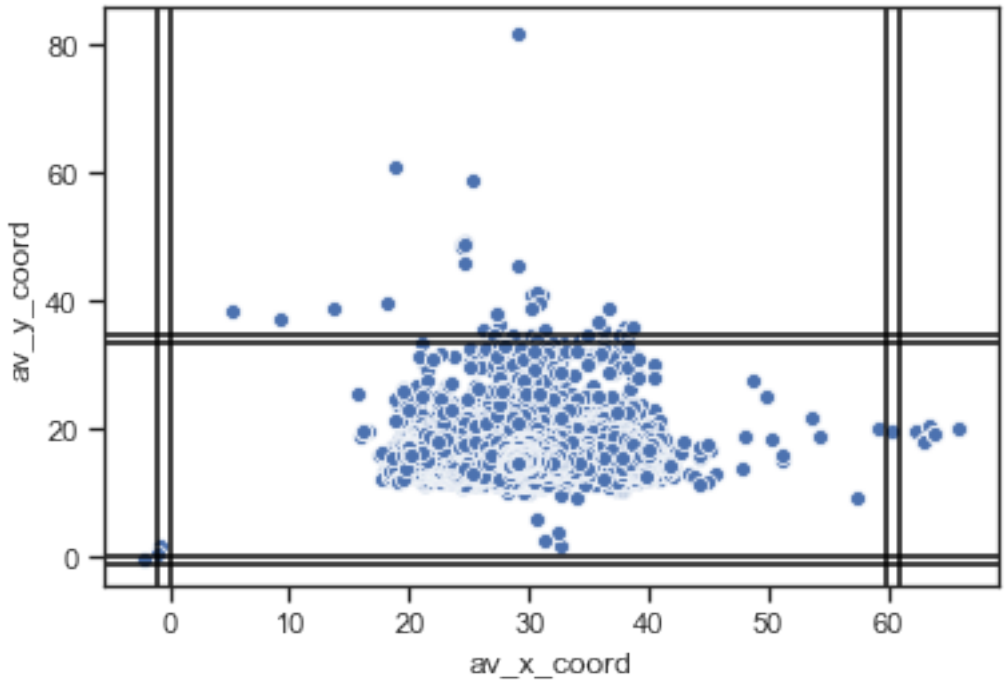
[491]:
```
for full in [pres_gaze_1, pres_gaze_2]:

    ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = full)
    ax.axhline(0, color='black'); ax.axhline(-1, color='black'); ax.axhline(33.
 ↪6, color='black'); ax.axhline(34.6, color='black')
    ax.axvline(0, color='black'); ax.axvline(-1, color='black'); ax.axvline(59.
 ↪8, color='black'); ax.axvline(60.8, color='black')
    plt.show()

    ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = full)
    ax.axhline(0, color='black'); ax.axhline(-1, color='black'); ax.axhline(33.
 ↪6, color='black'); ax.axhline(34.6, color='black')
    ax.axvline(0, color='black'); ax.axvline(-1, color='black'); ax.axvline(59.
 ↪8, color='black'); ax.axvline(60.8, color='black')
    ax.set_ylim(-5, 38.6)
    ax.set_xlim(-5, 64.8)
    plt.show()

    # sum the number of gazepoints falling within half a centimeter outside the␣
 ↪screen periphery
    x_bound = full[((full['av_x_coord'].between(-1,0)) | (full['av_x_coord'].
 ↪between(59.8,60.8))) & ((full['av_y_coord'].between(-1,34.6)))]
```

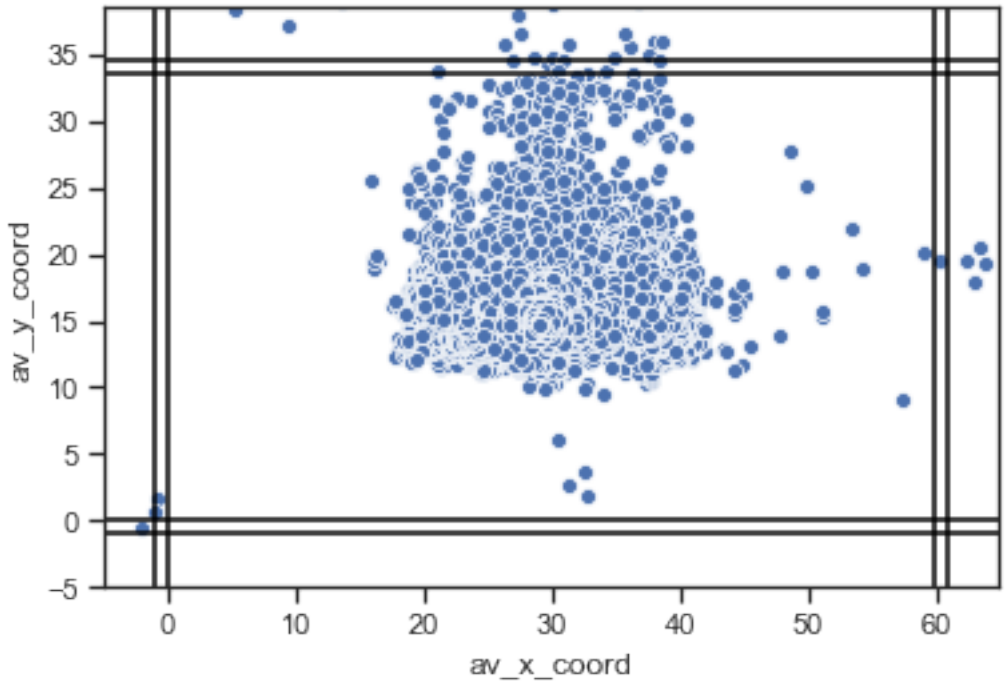
```
   y_bound = full[((full['av_y_coord'].between(-1,0)) | (full['av_y_coord'].
↪between(33.6,34.6))) & ((full['av_x_coord'].between(0,59.8)))]

   edges = pd.concat([x_bound, y_bound])
   print('Total number of gazepoints falling in the 1cm screen perimeter:␣
↪'+str(edges.shape[0]))
   print('Proportion of gazepoints falling in the 1cm screen perimeter:␣
↪'+str(edges.shape[0]/full.shape[0]))
```






```
Total number of gazepoints falling in the 1cm screen perimeter: 9
Proportion of gazepoints falling in the 1cm screen perimeter:
5.379236148466918e-05
```






```
Total number of gazepoints falling in the 1cm screen perimeter: 12
Proportion of gazepoints falling in the 1cm screen perimeter:
6.611460967587313e-05
```

[413]:
```
for full in [pres_gaze_1, pres_gaze_2]:

    # obtain the number of unique runs for each participant for each trial type␣
↪(Presentation and Memory)
    # unique_trials = full.groupby(['Subject','Run'])['Trial'].nunique()
    unique_runs = full.groupby(['Subject'])['Run'].nunique()

    print()
    print('Below, we can see the number of unique runs loaded for each subject,␣
↪for each trial type.')
    print()
    print("The set of all numbers of run from all participants contains "
          + str(unique_runs.nunique()) + ' unique value: '+str(unique_runs.
↪unique()))
    print()
    print(str(unique_runs))
```

```
Below, we can see the number of unique runs loaded for each subject, for each
```

```
trial type.

The set of all numbers of run from all participants contains 2 unique value: [7
8]

Subject
0     7
2     8
6     8
7     8
8     7
9     8
10    8
11    8
12    8
13    8
14    8
15    8
16    8
17    8
18    8
19    8
20    8
21    8
22    8
23    8
24    8
25    8
26    8
27    8
30    8
31    7
32    8
33    8
34    8
36    8
Name: Run, dtype: int64

Below, we can see the number of unique runs loaded for each subject, for each
trial type.

The set of all numbers of run from all participants contains 3 unique value: [7
8 6]

Subject
0      7
1      7
2      8
```

```
3      8
4      8
5      6
7      8
8      8
10     7
12     7
13     7
14     8
15     8
16     7
17     8
18     8
19     8
20     8
21     7
22     8
23     8
24     7
25     7
26     8
27     8
28     8
29     8
33     7
34     8
327    8
Name: Run, dtype: int64
```

9 See manual spot check of subjects with less than 8 runs, verifying there isn't gaze data from the missing runs

[492]:
```
# for full in [pres_gaze_1, pres_gaze_2]:

#     # obtain the number of unique runs for each participant for each trial␣
 ↪type (Presentation and Memory)
#     # unique_trials = full.groupby(['Subject','Run'])['Trial'].nunique()
#     unique_runs = full.groupby(['Subject','Run']).count()
#     print(unique_runs)

# #     print()
# #     print('Below, we can see the number of unique runs loaded for each␣
 ↪subject, for each trial type.')
# #     print()
# #     print("The set of all numbers of run from all participants contains "
```

```
# #           + str(unique_runs.nunique()) + ' unique value: '+str(unique_runs.
 ↪unique()))
# #     print()
# #     print(str(unique_runs))
```

Eliminate data with gazepoints falling outside of screen bounds

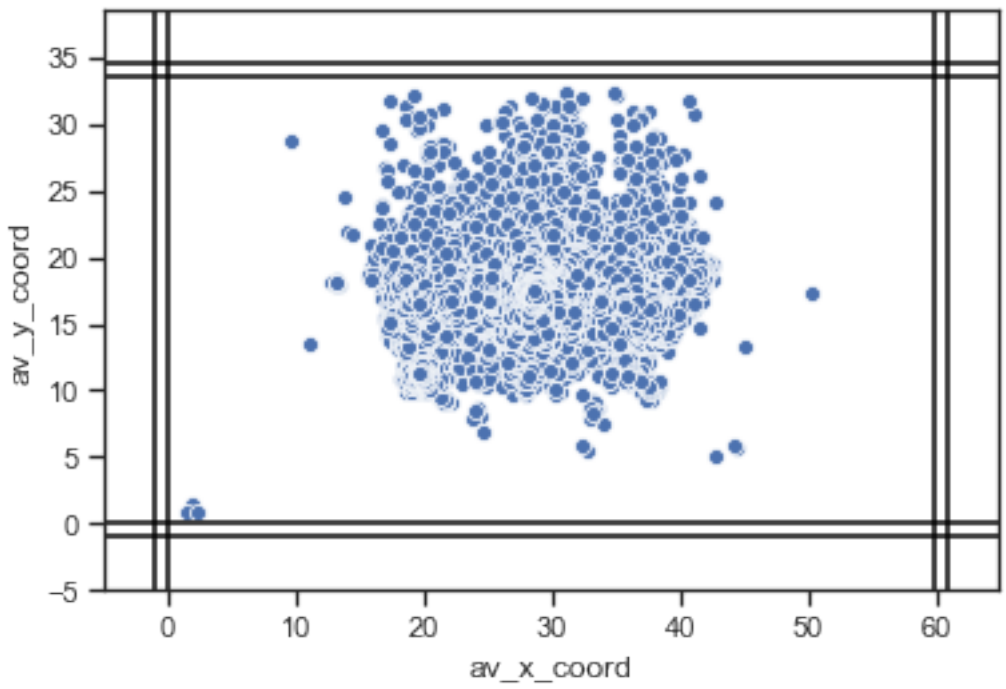
[493]:
```
pres_gaze_1_cropped = pres_gaze_1[(pres_gaze_1['xRaw_righteye']<=59.8) &␣
 ↪(pres_gaze_1['xRaw_lefteye']<=59.8)
                                  & (pres_gaze_1['yRaw_righteye']<=33.6)  &␣
 ↪(pres_gaze_1['yRaw_lefteye']<=33.6)
                                  & (pres_gaze_1['xRaw_lefteye']>=0) &␣
 ↪(pres_gaze_1['xRaw_righteye']>=0)
                                  & (pres_gaze_1['yRaw_lefteye']>=0) &␣
 ↪(pres_gaze_1['yRaw_righteye']>=0)]
```

[494]:
```
pres_gaze_2_cropped = pres_gaze_2[(pres_gaze_2['xRaw_righteye']<=59.8) &␣
 ↪(pres_gaze_2['xRaw_lefteye']<=59.8)
                                  & (pres_gaze_2['yRaw_righteye']<=33.6)  &␣
 ↪(pres_gaze_2['yRaw_lefteye']<=33.6)
                                  & (pres_gaze_2['xRaw_lefteye']>=0) &␣
 ↪(pres_gaze_2['xRaw_righteye']>=0)
                                  & (pres_gaze_2['yRaw_lefteye']>=0) &␣
 ↪(pres_gaze_2['yRaw_righteye']>=0)]
```

Plot again

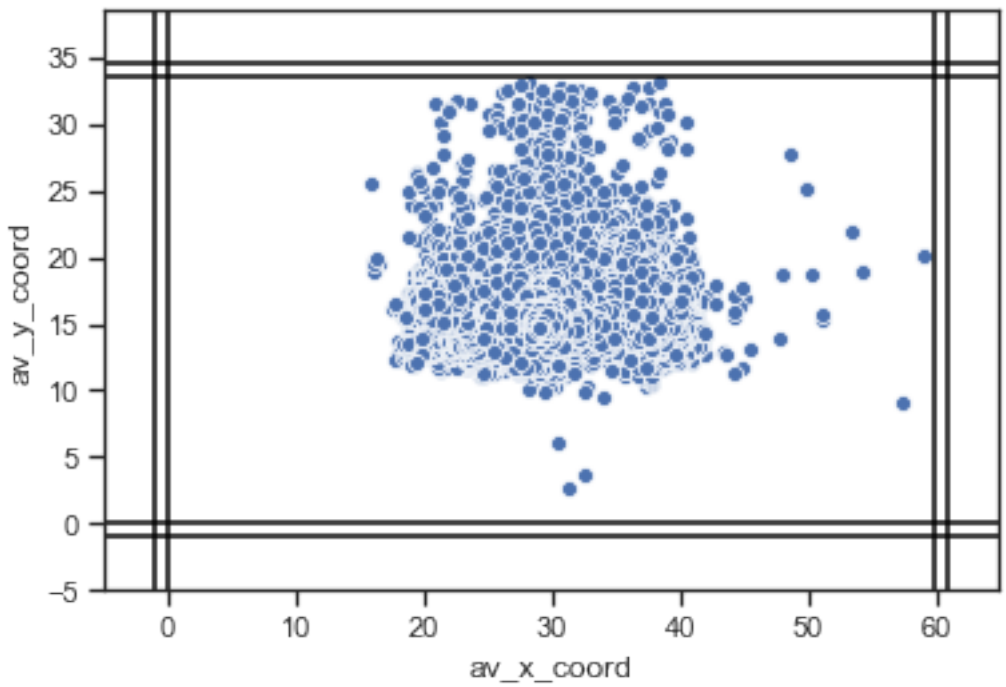
[495]:
```
ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = pres_gaze_1_cropped)

ax.axhline(0, color='black'); ax.axhline(-1, color='black'); ax.axhline(33.6,␣
 ↪color='black'); ax.axhline(34.6, color='black')
ax.axvline(0, color='black'); ax.axvline(-1, color='black'); ax.axvline(59.8,␣
 ↪color='black'); ax.axvline(60.8, color='black')
ax.set_ylim(-5, 38.6)
ax.set_xlim(-5, 64.8)
```

[495]: (-5, 64.8)



[496]:
```
ax = sb.scatterplot(x='av_x_coord', y='av_y_coord', data = pres_gaze_2_cropped)

ax.axhline(0, color='black'); ax.axhline(-1, color='black'); ax.axhline(33.6,␣
↪color='black'); ax.axhline(34.6, color='black')
ax.axvline(0, color='black'); ax.axvline(-1, color='black'); ax.axvline(59.8,␣
↪color='black'); ax.axvline(60.8, color='black')
ax.set_ylim(-5, 38.6)
ax.set_xlim(-5, 64.8)
```

[496]: (-5, 64.8)



[473]:
```
pres_gaze_1_cropped.to_csv('../parsed_data/gaze_data_sustained.csv')
```

[474]:
```
pres_gaze_2_cropped.to_csv('../parsed_data/gaze_data_variable.csv')
```

[ ]:

[ ]:

[ ]: