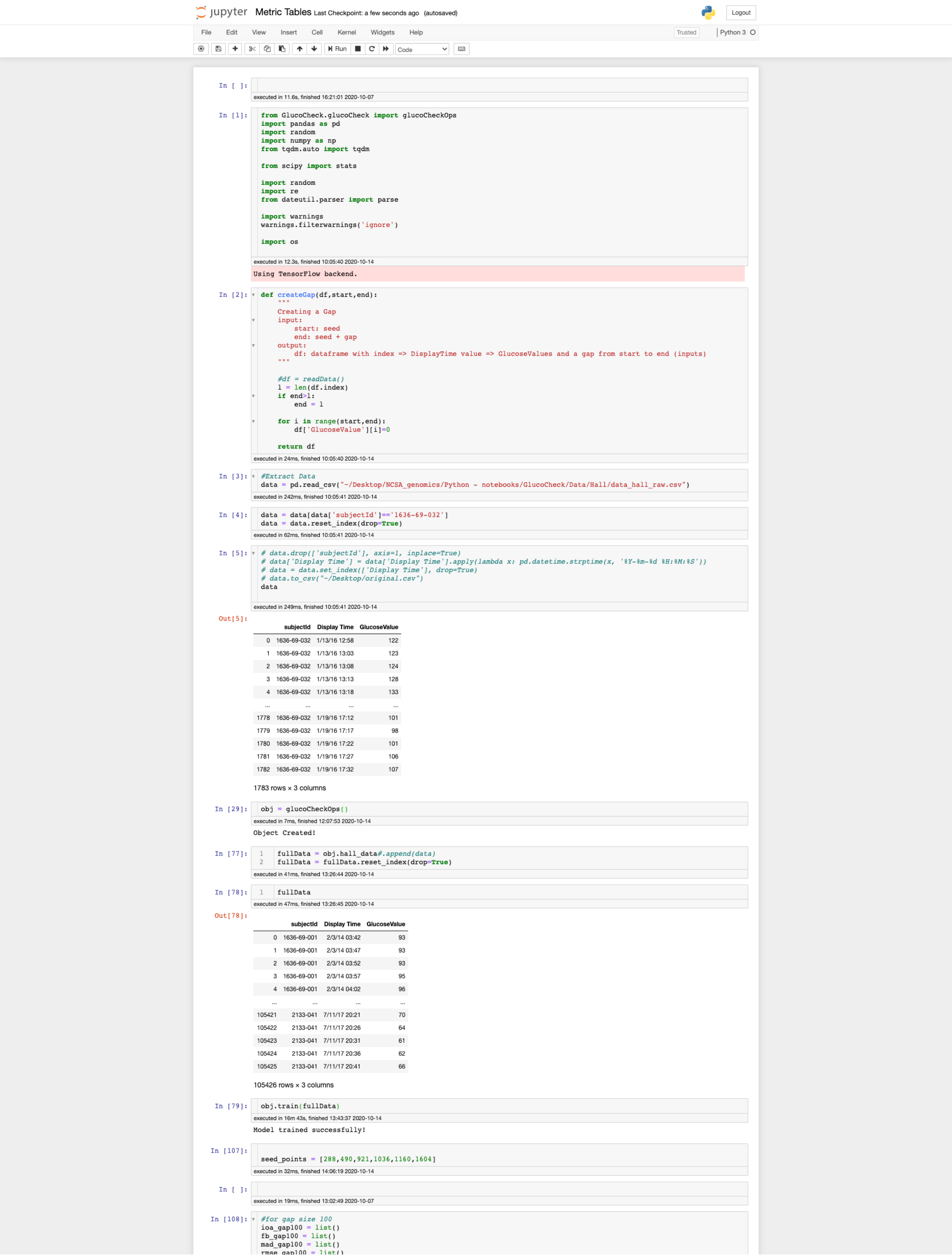


In [ ]:

executed in 11.6s, finished 16:21:01 2020-10-07

In [1]:

```python
from GlucoCheck.glucoCheck import glucoCheckOps
import pandas as pd
import random
import numpy as np
from tqdm.auto import tqdm

from scipy import stats

import random
import re
from dateutil.parser import parse

import warnings
warnings.filterwarnings('ignore')

import os
```

executed in 12.3s, finished 10:05:40 2020-10-14

```
Using TensorFlow backend.
```

In [2]:

```python
def createGap(df,start,end):
    """
    Creating a Gap
    input:
        start: seed
        end: seed + gap
    output:
        df: dataframe with index => DisplayTime value => GlucoseValues and a gap from start to end (inputs)
    """

    #df = readData()
    l = len(df.index)
    if end>l:
        end = l

    for i in range(start,end):
        df['GlucoseValue'][i]=0

    return df
```

executed in 24ms, finished 10:05:40 2020-10-14

In [3]:

```python
#Extract Data
data = pd.read_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Data/Hall/data_hall_raw.csv")
```

executed in 242ms, finished 10:05:41 2020-10-14

In [4]:

```python
data = data[data['subjectId']=='1636-69-032']
data = data.reset_index(drop=True)
```

executed in 62ms, finished 10:05:41 2020-10-14

In [5]:

```python
# data.drop(['subjectId'], axis=1, inplace=True)
# data['Display Time'] = data['Display Time'].apply(lambda x: pd.datetime.strptime(x, '%Y-%m-%d %H:%M:%S'))
# data = data.set_index(['Display Time'], drop=True)
# data.to_csv("~/Desktop/original.csv")
data
```

executed in 249ms, finished 10:05:41 2020-10-14

Out[5]:

	subjectId	Display Time	GlucoseValue
0	1636-69-032	1/13/16 12:58	122
1	1636-69-032	1/13/16 13:03	123
2	1636-69-032	1/13/16 13:08	124
3	1636-69-032	1/13/16 13:13	128
4	1636-69-032	1/13/16 13:18	133
...	...	...	...
1778	1636-69-032	1/19/16 17:12	101
1779	1636-69-032	1/19/16 17:17	98
1780	1636-69-032	1/19/16 17:22	101
1781	1636-69-032	1/19/16 17:27	106
1782	1636-69-032	1/19/16 17:32	107

1783 rows × 3 columns

In [29]:

```python
obj = glucoCheckOps()
```

executed in 7ms, finished 12:07:53 2020-10-14

```
Object Created!
```

In [77]:

```python
fullData = obj.hall_data#.append(data)
fullData = fullData.reset_index(drop=True)
```

executed in 41ms, finished 13:26:44 2020-10-14

In [78]:

```python
fullData
```

executed in 47ms, finished 13:26:45 2020-10-14

Out[78]:

	subjectId	Display Time	GlucoseValue
0	1636-69-001	2/3/14 03:42	93
1	1636-69-001	2/3/14 03:47	93
2	1636-69-001	2/3/14 03:52	93
3	1636-69-001	2/3/14 03:57	95
4	1636-69-001	2/3/14 04:02	96
...	...	...	...
105421	2133-041	7/11/17 20:21	70
105422	2133-041	7/11/17 20:26	64
105423	2133-041	7/11/17 20:31	61
105424	2133-041	7/11/17 20:36	62
105425	2133-041	7/11/17 20:41	66

105426 rows × 3 columns

In [79]:

```python
obj.train(fullData)
```

executed in 16m 43s, finished 13:43:37 2020-10-14

```
Model trained successfully!
```

In [107]:

```python

seed_points = [288,490,921,1036,1160,1604]
```

executed in 32ms, finished 14:06:19 2020-10-14

In [ ]:

executed in 19ms, finished 13:02:49 2020-10-07

In [108]:

```python
#for gap size 100
ioa_gap100 = list()
fb_gap100 = list()
mad_gap100 = list()
rmse_gap100 = list()
```

```
rmse_gap100 = list()
mape_gap100 = list()

for seed in tqdm(seed_points):
    start = seed
    end = seed+99
    data_with_missing = data.copy()
    data_with_missing = createGap(data_with_missing,start,end)

    imputed_data = obj.impute(data_with_missing,1)

    mad = obj.mad(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][star
    ioa = obj.index_agreement(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['Glucose
    fb = obj.fracBias(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][
    rmse = obj.rmse(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st
    mape = obj.mape(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st

    mad_gap100.append(mad)
    ioa_gap100.append(ioa)
    fb_gap100.append(fb)
    rmse_gap100.append(rmse)
    mape_gap100.append(mape)
```

executed in 2.15s, finished 14:06:22 2020-10-14

```
100%  6/6 [00:04<00:00, 1.36it/s]
```

```
In [109]:
#for gap size 50
ioa_gap50 = list()
fb_gap50 = list()
mad_gap50 = list()
rmse_gap50 = list()
mape_gap50 = list()

for seed in tqdm(seed_points):
    start = seed
    end = seed+49
    data_with_missing = data.copy()
    data_with_missing = createGap(data_with_missing,start,end)

    imputed_data = obj.impute(data_with_missing,1)

    mad = obj.mad(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][star
    ioa = obj.index_agreement(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['Glucose
    fb = obj.fracBias(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][
    rmse = obj.rmse(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st
    mape = obj.mape(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st

    mad_gap50.append(mad)
    ioa_gap50.append(ioa)
    fb_gap50.append(fb)
    rmse_gap50.append(rmse)
    mape_gap50.append(mape)
```

executed in 1.25s, finished 14:06:23 2020-10-14

```
100%  6/6 [00:02<00:00, 2.66it/s]
```

```
In [110]:
#for gap size 30
ioa_gap30 = list()
fb_gap30 = list()
mad_gap30 = list()
rmse_gap30 = list()
mape_gap30 = list()

for seed in tqdm(seed_points):
    start = seed
    end = start+29
    data_with_missing = data.copy()
    data_with_missing = createGap(data_with_missing,start,end)

    imputed_data = obj.impute(data_with_missing,1)

    mad = obj.mad(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][star
    ioa = obj.index_agreement(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['Glucose
    fb = obj.fracBias(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][
    rmse = obj.rmse(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st
    mape = obj.mape(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st

    mad_gap30.append(mad)
    ioa_gap30.append(ioa)
    fb_gap30.append(fb)
    rmse_gap30.append(rmse)
    mape_gap30.append(mape)
```

executed in 909ms, finished 14:06:24 2020-10-14

```
100%  6/6 [00:01<00:00, 5.89it/s]
```

```
In [111]:
#for gap size 15
ioa_gap15 = list()
fb_gap15 = list()
mad_gap15 = list()
rmse_gap15 = list()
mape_gap15 = list()

for seed in tqdm(seed_points):
    start = seed
    end = start+14
    data_with_missing = data.copy()
    data_with_missing = createGap(data_with_missing,start,end)

    imputed_data = obj.impute(data_with_missing,1)

    mad = obj.mad(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][star
    ioa = obj.index_agreement(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['Glucose
    fb = obj.fracBias(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][
    rmse = obj.rmse(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st
    mape = obj.mape(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st

    mad_gap15.append(mad)
    ioa_gap15.append(ioa)
    fb_gap15.append(fb)
    rmse_gap15.append(rmse)
    mape_gap15.append(mape)
```

executed in 597ms, finished 14:06:25 2020-10-14

```
100%  6/6 [05:39<00:00, 56.52s/it]
```

```
In [112]:
#for gap size 5
ioa_gap5 = list()
fb_gap5 = list()
mad_gap5 = list()
rmse_gap5 = list()
mape_gap5 = list()

for seed in tqdm(seed_points):
    start = seed
    end = start+4
    data_with_missing = data.copy()
    data_with_missing = createGap(data_with_missing,start,end)

    imputed_data = obj.impute(data_with_missing,1)
```

```
        mad = obj.mad(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][star
        ioa = obj.index_agreement(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['Glucose
        fb = obj.fracBias(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][
        rmse = obj.rmse(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st
        mape = obj.mape(np.asarray(imputed_data['GlucoseValue'][start:end-1].tolist()),np.asarray(data['GlucoseValue'][st

        mad_gap5.append(mad)
        ioa_gap5.append(ioa)
        fb_gap5.append(fb)
        rmse_gap5.append(rmse)
        mape_gap5.append(mape)
```

executed in 402ms, finished 14:06:25 2020-10-14

```
100% 6/6 [00:04<00:00, 1.28it/s]
```

In [113]:

```
IOA = pd.DataFrame({'Gap:5':ioa_gap5, 'Gap:15':ioa_gap15, 'Gap:30':ioa_gap30, 'Gap:50':ioa_gap50,'Gap:100':ioa_gap100
IOA
```

executed in 78ms, finished 14:06:25 2020-10-14

Out[113]:

	Gap:5	Gap:15	Gap:30	Gap:50	Gap:100
0	0.524590	0.732829	0.794466	0.688385	0.633212
1	0.282136	0.599247	0.496989	0.514146	0.471362
2	0.418301	0.846204	0.910656	0.851831	0.619313
3	0.444444	0.730517	0.683936	0.419430	0.440915
4	0.830189	0.929628	0.783506	0.754882	0.461313
5	0.512315	0.639980	0.680301	0.605894	0.318035

In [114]:

```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
```

executed in 6ms, finished 14:06:25 2020-10-14

In [ ]:

executed in 11ms, finished 14:05:48 2020-10-14

In [ ]:

executed in 8ms, finished 14:05:45 2020-10-14

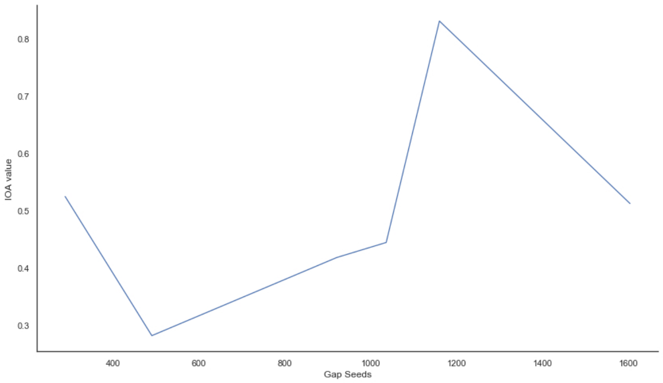
In [119]:

```

plt.figure(figsize=(14,8))
sns.set(style="white")
fig = sns.lineplot(x = seed_points, y = IOA['Gap:5'], data = IOA, palette="tab10", linewidth=1.25)
sns.despine()

fig.set_xlabel('Gap Seeds')
fig.set_ylabel('IOA value')
```

executed in 848ms, finished 14:10:52 2020-10-14

Out[119]:

```
Text(0, 0.5, 'IOA value')
```



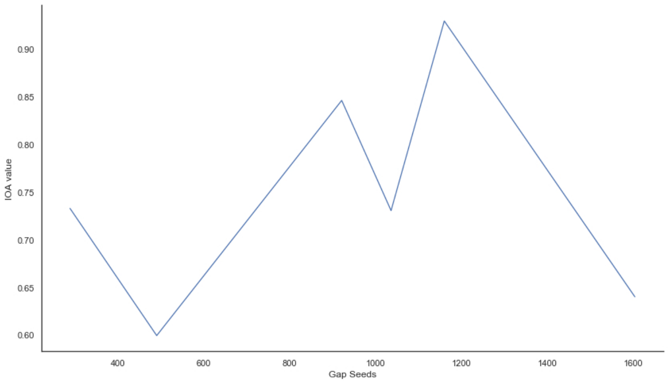
In [120]:

```

plt.figure(figsize=(14,8))
sns.set(style="white")
fig = sns.lineplot(x = seed_points, y = IOA['Gap:15'], data = IOA, palette="tab10", linewidth=1.25)
sns.despine()

fig.set_xlabel('Gap Seeds')
fig.set_ylabel('IOA value')
```

executed in 1.02s, finished 14:11:45 2020-10-14

Out[120]:

```
Text(0, 0.5, 'IOA value')
```



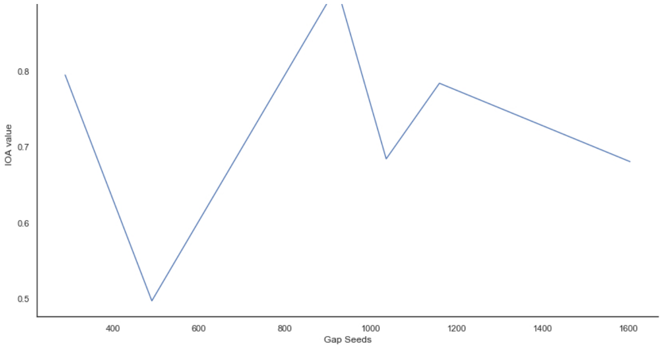
In [121]:

```

plt.figure(figsize=(14,8))
sns.set(style="white")
fig = sns.lineplot(x = seed_points, y = IOA['Gap:30'], data = IOA, palette="tab10", linewidth=1.25)
sns.despine()

fig.set_xlabel('Gap Seeds')
fig.set_ylabel('IOA value')
```

executed in 815ms, finished 14:11:51 2020-10-14

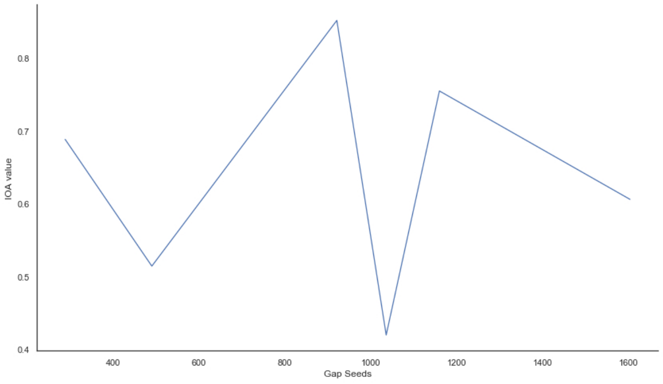
Out[121]:

```
Text(0, 0.5, 'IOA value')
```



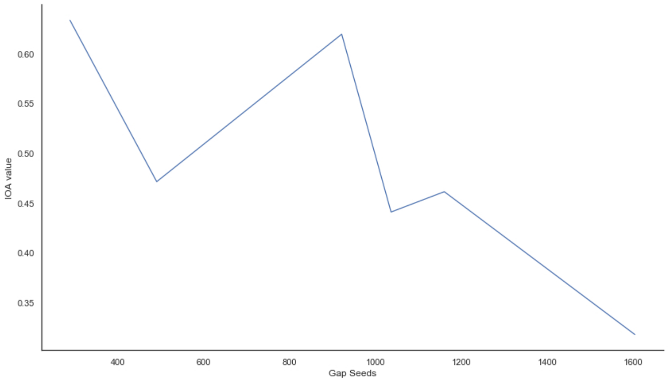
In [122]:
```
plt.figure(figsize=(14,8))
sns.set(style="white")
fig = sns.lineplot(x = seed_points, y = IOA['Gap:50'], data = IOA, palette="tab10", linewidth=1.25)
sns.despine()

fig.set_xlabel('Gap Seeds')
fig.set_ylabel('IOA value')
```
executed in 1.15s, finished 14:11:56 2020-10-14

Out[122]:
```
Text(0, 0.5, 'IOA value')
```



In [123]:
```
plt.figure(figsize=(14,8))
sns.set(style="white")
fig = sns.lineplot(x = seed_points, y = IOA['Gap:100'], data = IOA, palette="tab10", linewidth=1.25)
sns.despine()

fig.set_xlabel('Gap Seeds')
fig.set_ylabel('IOA value')
```
executed in 843ms, finished 14:12:04 2020-10-14

Out[123]:
```
Text(0, 0.5, 'IOA value')
```



In [ ]:

In [ ]:

In [73]:
```
MAD = pd.DataFrame({'Gap:5':mad_gap5, 'Gap:15':mad_gap15, 'Gap:30':mad_gap30, 'Gap:50':mad_gap50, 'Gap:100':mad_gap10
MAD
```
executed in 47ms, finished 13:25:47 2020-10-14

Out[73]:

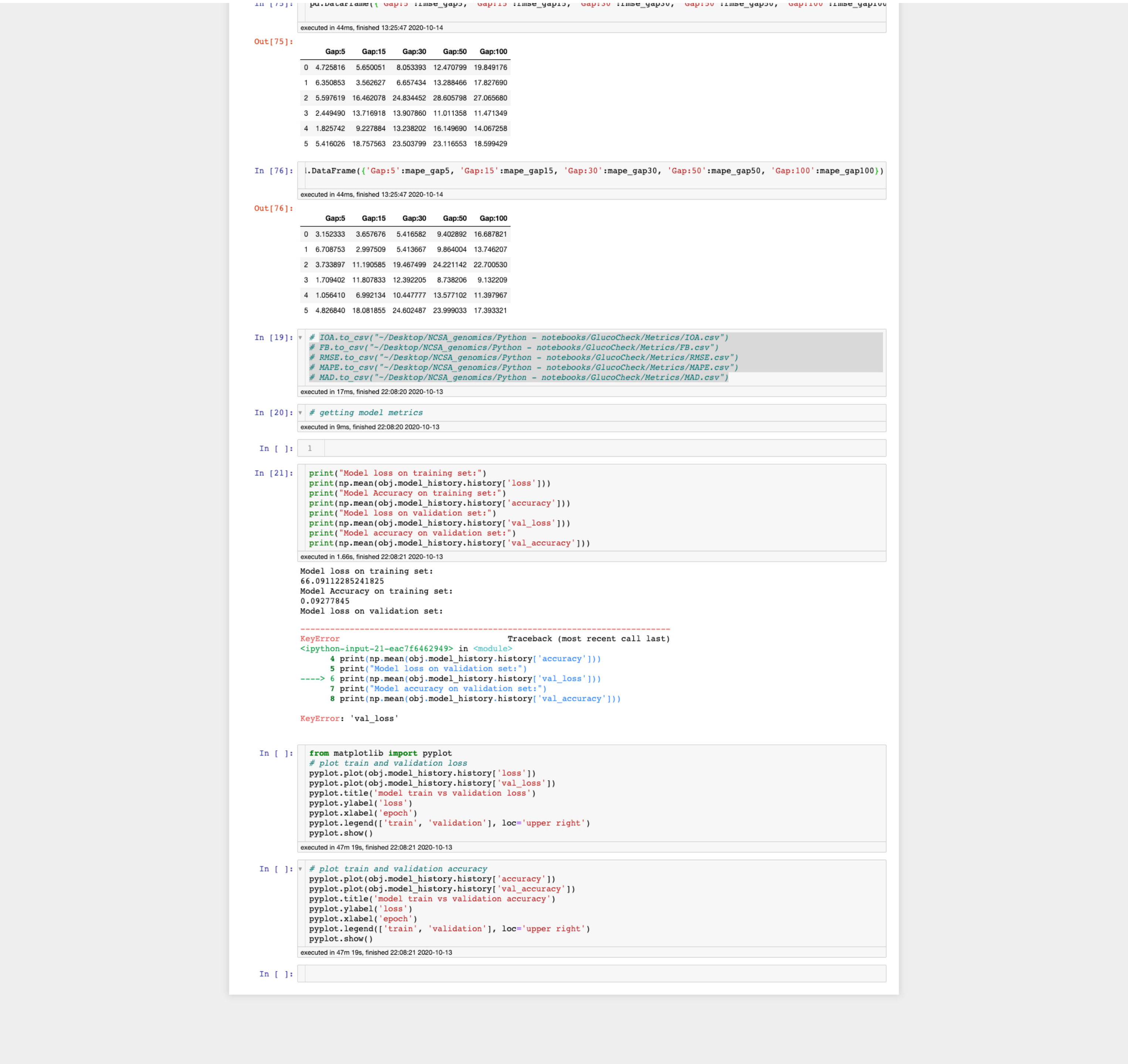
	Gap:5	Gap:15	Gap:30	Gap:50	Gap:100
0	4.333333	4.692308	6.428571	10.520833	17.091837
1	6.333333	2.846154	5.178571	10.041667	14.704082
2	5.333333	13.307692	21.250000	25.750000	24.551020
3	2.000000	11.538462	12.142857	8.833333	9.163265
4	1.333333	8.076923	11.535714	14.645833	12.540816
5	5.333333	17.230769	22.428571	21.875000	16.387755

In [74]:
```
FB = pd.DataFrame({'Gap:5':fb_gap5, 'Gap:15':fb_gap15, 'Gap:30':fb_gap30, 'Gap:50':fb_gap50, 'Gap:100':fb_gap100})
FB
```
executed in 42ms, finished 13:25:47 2020-10-14

Out[74]:

	Gap:5	Gap:15	Gap:30	Gap:50	Gap:100
0	0.031083	0.035674	0.051890	0.088006	0.149169
1	0.069425	0.030546	0.052337	0.091316	0.124905
2	0.038118	0.103091	0.170869	0.209447	0.198190
3	0.017316	0.108942	0.114510	0.082013	0.085342
4	0.010479	0.066834	0.097473	0.125138	0.106183
5	0.047094	0.162981	0.216415	0.210963	0.155640

In [75]:
```
pd.DataFrame({'Gap:5':rmse_gap5, 'Gap:15':rmse_gap15, 'Gap:30':rmse_gap30, 'Gap:50':rmse_gap50, 'Gap:100':rmse_gap100
```



```
In [75]:  pd.DataFrame({'Gap:5':rmse_gap5, 'Gap:15':rmse_gap15, 'Gap:30':rmse_gap30, 'Gap:50':rmse_gap50, 'Gap:100':rmse_gap100
executed in 44ms, finished 13:25:47 2020-10-14
```

Out[75]:

	Gap:5	Gap:15	Gap:30	Gap:50	Gap:100
0	4.725816	5.650051	8.053393	12.470799	19.849176
1	6.350853	3.562627	6.657434	13.288466	17.827690
2	5.597619	16.462078	24.834452	28.605798	27.065680
3	2.449490	13.716918	13.907860	11.011358	11.471349
4	1.825742	9.227884	13.238202	16.149690	14.067258
5	5.416026	18.757563	23.503799	23.116553	18.599429

```
In [76]:  l.DataFrame({'Gap:5':mape_gap5, 'Gap:15':mape_gap15, 'Gap:30':mape_gap30, 'Gap:50':mape_gap50, 'Gap:100':mape_gap100})
executed in 44ms, finished 13:25:47 2020-10-14
```

Out[76]:

	Gap:5	Gap:15	Gap:30	Gap:50	Gap:100
0	3.152333	3.657676	5.416582	9.402892	16.687821
1	6.708753	2.997509	5.413667	9.864004	13.746207
2	3.733897	11.190585	19.467499	24.221142	22.700530
3	1.709402	11.807833	12.392205	8.738206	9.132209
4	1.056410	6.992134	10.447777	13.577102	11.397967
5	4.826840	18.081855	24.602487	23.999033	17.393321

```
In [19]:  # IOA.to_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Metrics/IOA.csv")
          # FB.to_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Metrics/FB.csv")
          # RMSE.to_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Metrics/RMSE.csv")
          # MAPE.to_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Metrics/MAPE.csv")
          # MAD.to_csv("~/Desktop/NCSA_genomics/Python - notebooks/GlucoCheck/Metrics/MAD.csv")
executed in 17ms, finished 22:08:20 2020-10-13

In [20]:  # getting model metrics
executed in 9ms, finished 22:08:20 2020-10-13

In [ ]:   1

In [21]:  print("Model loss on training set:")
          print(np.mean(obj.model_history.history['loss']))
          print("Model Accuracy on training set:")
          print(np.mean(obj.model_history.history['accuracy']))
          print("Model loss on validation set:")
          print(np.mean(obj.model_history.history['val_loss']))
          print("Model accuracy on validation set:")
          print(np.mean(obj.model_history.history['val_accuracy']))
executed in 1.66s, finished 22:08:21 2020-10-13

          Model loss on training set:
          66.09112285241825
          Model Accuracy on training set:
          0.09277845
          Model loss on validation set:

          ---------------------------------------------------------------------------
          KeyError                                  Traceback (most recent call last)
          <ipython-input-21-eac7f6462949> in <module>
                4 print(np.mean(obj.model_history.history['accuracy']))
                5 print("Model loss on validation set:")
          ----> 6 print(np.mean(obj.model_history.history['val_loss']))
                7 print("Model accuracy on validation set:")
                8 print(np.mean(obj.model_history.history['val_accuracy']))

          KeyError: 'val_loss'

In [ ]:   from matplotlib import pyplot
          # plot train and validation loss
          pyplot.plot(obj.model_history.history['loss'])
          pyplot.plot(obj.model_history.history['val_loss'])
          pyplot.title('model train vs validation loss')
          pyplot.ylabel('loss')
          pyplot.xlabel('epoch')
          pyplot.legend(['train', 'validation'], loc='upper right')
          pyplot.show()
executed in 47m 19s, finished 22:08:21 2020-10-13

In [ ]:   # plot train and validation accuracy
          pyplot.plot(obj.model_history.history['accuracy'])
          pyplot.plot(obj.model_history.history['val_accuracy'])
          pyplot.title('model train vs validation accuracy')
          pyplot.ylabel('loss')
          pyplot.xlabel('epoch')
          pyplot.legend(['train', 'validation'], loc='upper right')
          pyplot.show()
executed in 47m 19s, finished 22:08:21 2020-10-13

In [ ]:
```