In [1]:
```
import numpy as np
import pandas as pd
from sklearn.feature_selection import mutual_info_regression
import os
import matplotlib.pyplot as plt
from datetime import datetime
from statistics import mean, median
%matplotlib inline
```

In [2]:
```
read_dir_adhd = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_test_compl
mi_dir_adhd   = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_test_compl
mi_dir_adhd_overlap   = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_te
epoch_adhd_dir = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_test_comp


read_dir_control = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_test_co
mi_dir_control   = r'C:\Users\yl646\Documents\ADHD_research\DATA\OUTPUT\step_6_test_co
mi_dir_control_overlap   = r'C:\Users\yl646\Documents\ADHD_research\DATA\OUTPUT\step_6
epoch_control_dir = r'C:\Users\yl646\Documents\ADHD_Research\DATA\OUTPUT\step_6_test_c
```

Create Mutual Information Table with 20 Channels for Graph Construction

In [3]:
```
# create ADHD dataset with correct dimension
list_of_ADHD = []
total_epoch = 0
n_epoch_each_adhd = []
n_epoch_each_control = []

# Each file has column labels where first column is the time stamp.
# Data was epoched at 4 second window, so time stampt is repeated values 0~512
for i in os.listdir(read_dir_adhd):
    df = pd.read_csv(read_dir_adhd+"\\"+i)
    arr = df.to_numpy()
    num_epoch = arr.shape[0] / 512 # 512 data points in 1 epoch (4 sec x 128 hz)

    n_epoch_each_adhd.append(num_epoch)
    total_epoch += num_epoch

    list_of_epoch = []
    for i in range(int(num_epoch)):
        single_epoch = arr[ i*512 : (i+1)*512 , 1: ].transpose() # slice for each epoc
        list_of_epoch.append(single_epoch)
    list_of_ADHD.append(list_of_epoch)

all_epoch = []
for patient in list_of_ADHD:
    for epoch in patient:
        all_epoch.append(epoch)
ADHD_dataset = np.stack(all_epoch)

print('Total Epoch: ',total_epoch)
print('ADHD dataset dimension: ',ADHD_dataset.shape, '(epoch, channel, time)')
np.save(epoch_adhd_dir, n_epoch_each_adhd) # save number of epochs per patient. this f
```

```
Total Epoch:  2231.0
ADHD dataset dimension:  (2231, 20, 512) (epoch, channel, time)
```

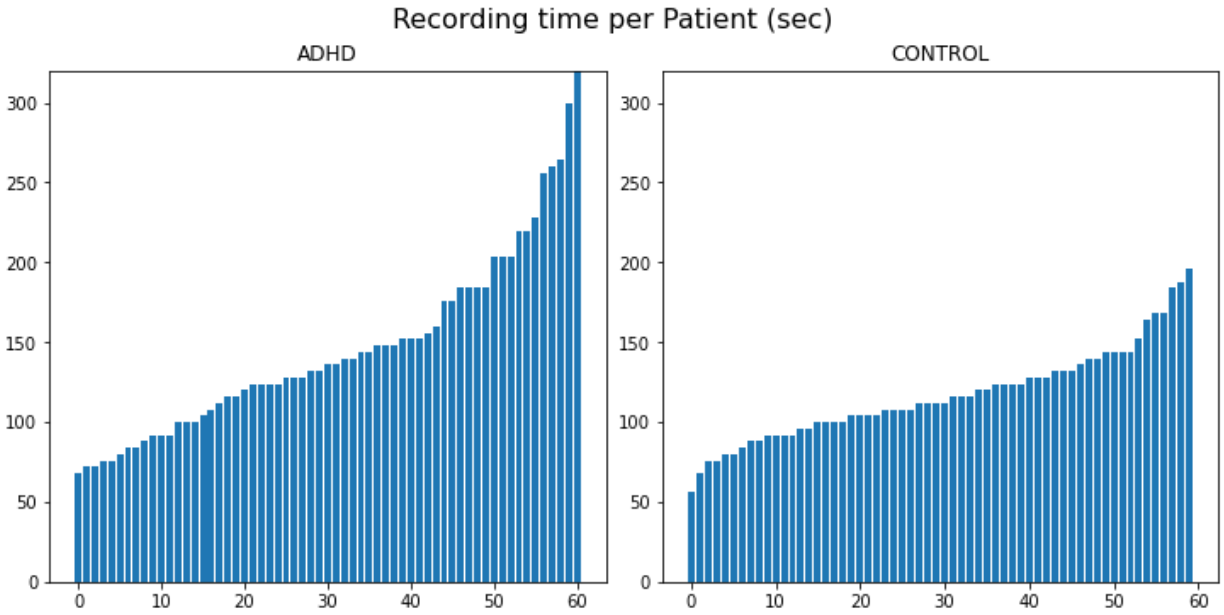
In [5]:
```
# create CONTROL dataset with correct dimension
# Same process is repeated for control group
list_of_CONTROL = []
total_epoch = 0
n_epoch_each_control = []

for i in os.listdir(read_dir_control):
    df = pd.read_csv(read_dir_control+"\\"+i)
    arr = df.to_numpy()
    num_epoch = arr.shape[0] / 512

    n_epoch_each_control.append(num_epoch)
    total_epoch += num_epoch

    list_of_epoch = []
    total_epoch += num_epoch
    for i in range(int(num_epoch)):
        single_epoch = arr[ i*512 : (i+1)*512 , 1: ].transpose()
        list_of_epoch.append(single_epoch)
    list_of_CONTROL.append(list_of_epoch)

all_epoch = []
for patient in list_of_CONTROL:
    for epoch in patient:
        all_epoch.append(epoch)

CONTROL_dataset = np.stack(all_epoch)

print('Total Epoch: ',total_epoch)
print('CONTROL dataset dimension: ',CONTROL_dataset.shape, '(epoch, channel, time)')
np.save(epoch_control_dir, n_epoch_each_control)
```

```
Total Epoch:  3514.0
CONTROL dataset dimension:  (1757, 20, 512) (epoch, channel, time)
```

In [8]:
```
n_adhd = list(range(len(n_epoch_each_adhd)))
n_control = list(range(len(n_epoch_each_control)))

t_adhd=np.multiply(n_epoch_each_adhd,4)
t_control = np.multiply(n_epoch_each_control,4)

print(f'Mean Recording Time ADHD: {mean(t_adhd):.2f}')
print(f'Mean Recording Time CONTROL: {mean(t_control):.2f}')
print(f'Median Recording Time ADHD: {median(t_adhd):.2f}')
print(f'Median Recording Time CONTROL: {median(t_control):.2f}')
print(f'Min Recording Time ADHD: {min(t_adhd):.2f}')
print(f'Min Recording Time CONTROL: {min(t_control):.2f}')
print(f'Max Recording Time ADHD: {max(t_adhd):.2f}')
print(f'Max Recording Time CONTROL: {max(t_control):.2f}')
```

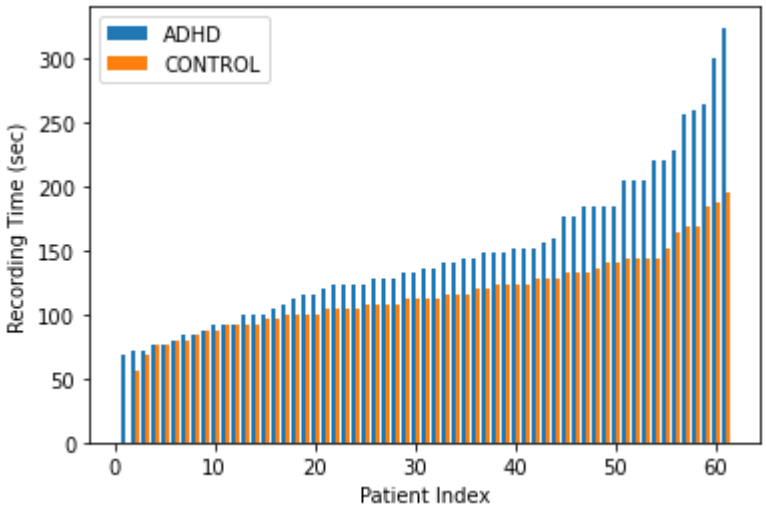
```
Mean Recording Time ADHD: 146.30
Mean Recording Time CONTROL: 117.13
Median Recording Time ADHD: 136.00
Median Recording Time CONTROL: 112.00
Min Recording Time ADHD: 68.00
Min Recording Time CONTROL: 56.00
Max Recording Time ADHD: 324.00
Max Recording Time CONTROL: 196.00
```

In [9]:
```
fig, axs = plt.subplots(1, 2, constrained_layout=True, figsize=(10,5))
fig.suptitle('Recording time per Patient (sec)', fontsize=16)

axs[0].bar(n_adhd, sorted(list(t_adhd)))
axs[0].set_title('ADHD')
axs[0].set_ylim([0,320])

axs[1].bar(n_control, sorted(list(t_control)))
axs[1].set_title('CONTROL')
axs[1].set_ylim([0,320])

fig.savefig('recording_time_per_patient.png')
```



In [7]:
```
x=list(range(1, 62))
x1=[i-0.2 for i in x]
x2=[i+0.2 for i in x]
t_control=list(t_control)
if len(t_control)!=61:
    t_control.append(0)
print(len(x1))
print(len(x2))
print(len(list(t_adhd)))
print(len(t_control))
plt.bar(x1, sorted(list(t_adhd)), label='ADHD',width=0.4)
plt.bar(x2, sorted(t_control), label='CONTROL',width=0.4)
plt.xlabel('Patient Index')
plt.ylabel('Recording Time (sec)')
plt.legend()
plt.savefig('recording_time.png')
plt.show()
```

```
61
61
61
61
```



Create MI Table for GNN Construction

In [6]:

```
# create mutual information table
(epochs, channels, frames) = ADHD_dataset.shape
mi_table = np.zeros([epochs, channels, channels])
for j in range(epochs):
    if j%10==0:
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        print("ADHD", j, current_time)
    example = ADHD_dataset[j,:,:]
    for k in range(channels):
        x = np.delete(example,k,axis=0)
        y = example[k,:]
        mi = mutual_info_regression(x.transpose(),y) #  This is where MI is calculated
        mi = np.insert(mi,k,0)  # assign 0 for position (k,k) (self-loop value 0)

        mi_table[j,k,:] = mi

# mi_table dimension: (patients, epochs, channel, channel)
np.save(mi_dir_adhd, mi_table)


# create mutual information table
(epochs, channels, frames) = CONTROL_dataset.shape
mi_table = np.zeros([epochs, channels, channels])
for j in range(epochs):
    if j%10==0:
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        print("CONTROL", j, current_time)
    example = CONTROL_dataset[j,:,:]
    for k in range(channels):
        x = np.delete(example,k,axis=0)
        y = example[k,:]
        mi = mutual_info_regression(x.transpose(),y)
        mi = np.insert(mi,k,0)
```

```
        mi_table[j,k,:] = mi

# mi_table dimension: (patients, epochs, channel, channel)
np.save(mi_dir_control, mi_table)
```

```
CONTROL 1360 19:05:45
CONTROL 1370 19:05:57
CONTROL 1380 19:06:08
CONTROL 1390 19:06:20
CONTROL 1400 19:06:32
CONTROL 1410 19:06:44
CONTROL 1420 19:06:56
CONTROL 1430 19:07:08
CONTROL 1440 19:07:20
CONTROL 1450 19:07:31
CONTROL 1460 19:07:43
CONTROL 1470 19:07:55
CONTROL 1480 19:08:06
CONTROL 1490 19:08:18
CONTROL 1500 19:08:29
CONTROL 1510 19:08:41
CONTROL 1520 19:08:53
CONTROL 1530 19:09:05
CONTROL 1540 19:09:16
CONTROL 1550 19:09:28
CONTROL 1560 19:09:40
CONTROL 1570 19:09:51
CONTROL 1580 19:10:03
CONTROL 1590 19:10:15
CONTROL 1600 19:10:27
CONTROL 1610 19:10:39
CONTROL 1620 19:10:50
CONTROL 1630 19:11:02
CONTROL 1640 19:11:14
CONTROL 1650 19:11:26
CONTROL 1660 19:11:37
CONTROL 1670 19:11:49
CONTROL 1680 19:12:00
CONTROL 1690 19:12:12
CONTROL 1700 19:12:23
CONTROL 1710 19:12:35
CONTROL 1720 19:12:46
CONTROL 1730 19:12:58
CONTROL 1740 19:13:10
CONTROL 1750 19:13:21
```

Sample MI table

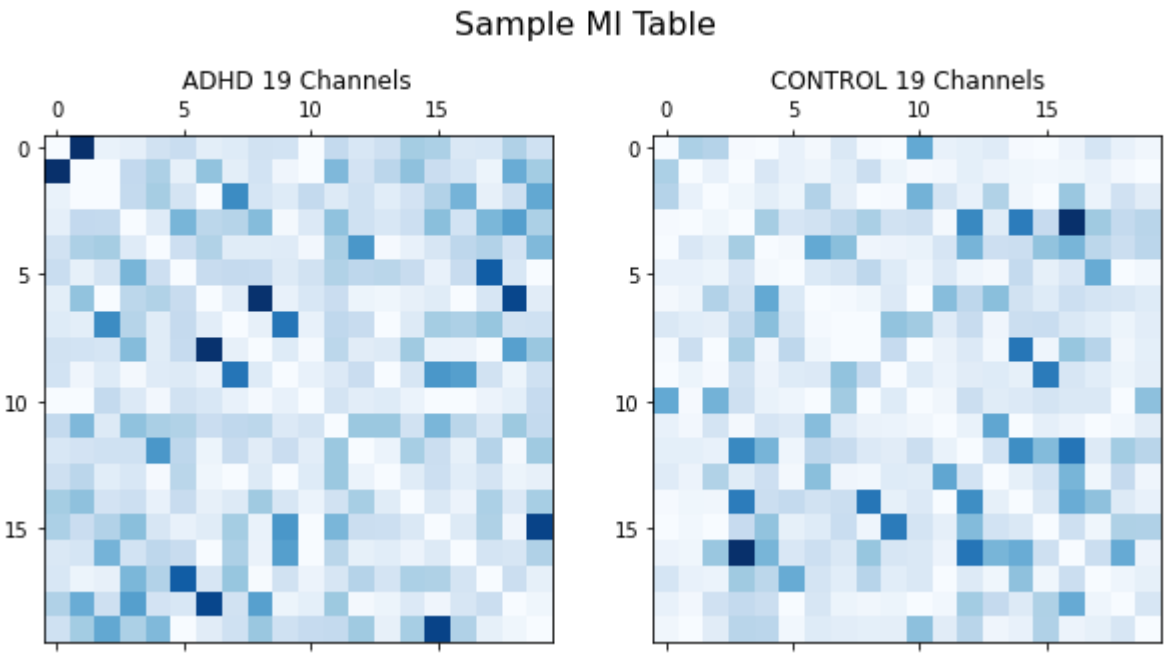
In [7]:

```
a= np.load(mi_dir_adhd)
b = np.load(mi_dir_control)
print(f'ADHD MI table shape: {a.shape}')
print(f'CONTROL MI table shape: {b.shape}')
fig, axs = plt.subplots(1,2, figsize=(10,5.5))
fig.suptitle('Sample MI Table', fontsize=16)
axs[0].matshow(a[1700,:,:], cmap=plt.cm.Blues)
axs[0].set_title('ADHD 19 Channels')
axs[1].matshow(b[1700,:,:], cmap=plt.cm.Blues)
axs[1].set_title('CONTROL 19 Channels')
```

```
ADHD MI table shape: (2231, 20, 20)
CONTROL MI table shape: (1757, 20, 20)
```

Out[7]:

```
Text(0.5, 1.0, 'CONTROL 19 Channels')
```



Create Mutual Information with Overlap for CNN model

Create a overlapped numpy channel array with preprocessed signals

In [8]:
```
#  Dataset contained channel label file. I was able to manually match the numeric valu
overlap_seq = [3,11,1,17,2,12,4,6,14,16,10,8,19,8,18,6,4,17,3,5,7,19,9,15,7,13,5,3,11]

(epochs, channel, time)=ADHD_dataset.shape
overlap_channels = len(overlap_seq)

ADHD_overlap = np.zeros((epochs,overlap_channels,512))

for epoch in range(epochs):  # for each epoch, rearange channels.
    for i in range(overlap_channels):
        index = overlap_seq[i]-1
        ADHD_overlap[epoch, i, :] = ADHD_dataset[epoch,index,:]

print('ADHD overlap dimension: ', ADHD_overlap.shape)
```

```
ADHD overlap dimension:  (2231, 29, 512)
```

In [9]:
```
# Repeat for contol group
overlap_seq = [3,11,1,17,2,12,4,6,14,16,10,8,19,8,18,6,4,17,3,5,7,19,9,15,7,13,5,3,11]

(epochs, channel, time)=CONTROL_dataset.shape
overlap_channels = len(overlap_seq)

CONTROL_overlap = np.zeros((epochs,overlap_channels,512))

for epoch in range(epochs):
    for i in range(overlap_channels):
        index = overlap_seq[i]-1
```

```
        CONTROL_overlap[epoch, i, :] = CONTROL_dataset[epoch,index,:]

print('CONTROL overlap dimension: ', CONTROL_overlap.shape)
```

```
CONTROL overlap dimension:  (1757, 29, 512)
```

Create MI table with overlap. 29x29 adjacency matrix

In [10]:
```
overlap_seq = np.array([3,11,1,17,2,12,4,6,14,16,10,8,19,8,18,6,4,17,3,5,7,19,9,15,7,1

# create mutual information table
(epochs, channels, frames) = ADHD_overlap.shape
mi_table = np.zeros([epochs, channels, channels])
for j in range(epochs):
    if j%10==0:
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        print("ADHD overlap", j, current_time)

    example = ADHD_overlap[j,:,:]

    for k in range(channels):
        pos = np.where(np.array(overlap_seq) == overlap_seq[k])[0] # find indices for
        x = np.delete(example,k,axis=0)
        y = example[k,:]
        mi = mutual_info_regression(x.transpose(),y)  # calculate MI here
        mi = np.insert(mi,k,0) # assign 0 for position (k,k) (self-loop value 0)
        mi_table[j,k,:] = mi
        mi_table[j,k,pos] = 0 # assign 0 for other self loop indices that were found

# mi_table dimension: (patients, epochs, channel, channel)
np.save(mi_dir_adhd_overlap, mi_table)


# create mutual information table
(epochs, channels, frames) = CONTROL_overlap.shape
mi_table = np.zeros([epochs, channels, channels])
for j in range(epochs):
    if j%10==0:
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        print("CONTROL overlap", j, current_time)

    example = CONTROL_overlap[j,:,:]

    for k in range(channels):
        pos = np.where(np.array(overlap_seq) == overlap_seq[k])[0] # find indices for
        x = np.delete(example,k,axis=0)
        y = example[k,:]
        mi = mutual_info_regression(x.transpose(),y)
        mi = np.insert(mi,k,0) # assign 0 for position (k,k)
        mi_table[j,k,:] = mi
        mi_table[j,k,pos] = 0 # assign 0 for other self loop indices that were found

# mi_table dimension: (patients, epochs, channel, channel)
np.save(mi_dir_control_overlap, mi_table)
```

```
CONTROL overlap 1360 21:42:51
CONTROL overlap 1370 21:43:16
CONTROL overlap 1380 21:43:40
CONTROL overlap 1390 21:44:05
CONTROL overlap 1400 21:44:30
CONTROL overlap 1410 21:44:54
CONTROL overlap 1420 21:45:20
CONTROL overlap 1430 21:45:45
CONTROL overlap 1440 21:46:10
CONTROL overlap 1450 21:46:35
CONTROL overlap 1460 21:47:00
CONTROL overlap 1470 21:47:24
CONTROL overlap 1480 21:47:50
CONTROL overlap 1490 21:48:14
CONTROL overlap 1500 21:48:39
CONTROL overlap 1510 21:49:03
CONTROL overlap 1520 21:49:28
CONTROL overlap 1530 21:49:54
CONTROL overlap 1540 21:50:19
CONTROL overlap 1550 21:50:44
CONTROL overlap 1560 21:51:09
CONTROL overlap 1570 21:51:33
CONTROL overlap 1580 21:51:58
CONTROL overlap 1590 21:52:22
CONTROL overlap 1600 21:52:47
CONTROL overlap 1610 21:53:12
CONTROL overlap 1620 21:53:36
CONTROL overlap 1630 21:54:01
CONTROL overlap 1640 21:54:26
CONTROL overlap 1650 21:54:51
CONTROL overlap 1660 21:55:15
CONTROL overlap 1670 21:55:40
CONTROL overlap 1680 21:56:05
CONTROL overlap 1690 21:56:30
CONTROL overlap 1700 21:56:55
CONTROL overlap 1710 21:57:19
CONTROL overlap 1720 21:57:44
CONTROL overlap 1730 21:58:09
CONTROL overlap 1740 21:58:33
CONTROL overlap 1750 21:58:57
```

In [11]:

```
a= np.load(mi_dir_adhd_overlap)
b = np.load(mi_dir_control_overlap)
print(f'ADHD MI table shape: {a.shape}')
print(f'CONTROL MI table shape: {b.shape}')
fig2, axs2 = plt.subplots(1,2, figsize=(10,5.5))
fig2.suptitle('Sample MI Table', fontsize=16)
axs2[0].matshow(a[1700,:,:], cmap=plt.cm.Blues)
axs2[0].set_title('ADHD 29 Channels')
axs2[1].matshow(b[1700,:,:], cmap=plt.cm.Blues)
axs2[1].set_title('CONTROL 29 Channels')
```

```
ADHD MI table shape: (2231, 29, 29)
CONTROL MI table shape: (1757, 29, 29)
```

Out[11]:

```
Text(0.5, 1.0, 'CONTROL 29 Channels')
```

Sample MI Table