HOW TO FIND ALGORITHM (SNORE & LEG MOVEMENT)

*ส่วนตัวจะแปลงเป็น dataframe เพื่อง่ายต่อการสังเกตข้อมูล

DATA TRANSFORMATION

STEP 1

แยกข้อมูล info ออกเอาแค่ส่วน event ที่สนใจแล้วนำข้อมูลมารวมกับ signal ของผู้ตรวจ

STEP 2

เพื่อศึกษาลักษณะของข้อมูลจึงต้องคำนวณหา Parameter ต่างๆ

Info

	index	event_name	start	period(s)	start_cal	period_cal	end_cal	amp_start	amp_end	peak	med_amp	pw_sp
0	54	Snore	221571.0	1.0	86551	100.0	86701	-3.0	-9.0	12.0	3.0	1638.0
1	56	Snore	225654.0	1.0	88146	100.0	88296	0.0	-6.0	12.0	3.0	1359.0
2	65	Snore	239314.0	1.0	93482	100.0	93632	-3.0	-3.0	9.0	3.0	1557.0
3	94	Snore	297438.0	1.0	116186	100.0	116336	-3.0	0.0	9.0	3.0	1134.0
4	95	Snore	297438.0	1.0	116186	100.0	116336	-3.0	0.0	9.0	3.0	1134.0
...	...	...	...	...	...	...	...	...	...	...	...	...
694	3088	Snore	6716860.0	1.0	2623773	100.0	2623923	-3.0	0.0	6.0	3.0	1017.0
695	3097	Snore	6735074.0	1.0	2630888	100.0	2631038	-3.0	-3.0	6.0	3.0	1323.0
696	3103	Snore	6748954.0	1.0	2636310	100.0	2636460	-3.0	0.0	6.0	3.0	1404.0
697	3104	Snore	6750253.0	1.0	2636817	100.0	2636967	0.0	0.0	9.0	3.0	1125.0
698	3109	Snore	6765513.0	1.0	2642778	100.0	2642928	0.0	-12.0	6.0	3.0	1242.0

699 rows × 14 columns

Parameter

จากการลอง coding พบว่า parameter ที่เหมาะสม

- Index เริ่ม
- Period of event

จากการคำนวณ

- Peak
- Power spectrum

ภาพตัวอย่าง dataframe ของ snore ที่รวม parameter ต่างๆ

หมายเหตุ

- การคำนวณ index ต้องระวังหาก sampling rate ของ info กับ signal ไม่เท่ากัน
- Leg movement signal ต้องนำ 2 port มาลบกันก่อน และต้องแยกคิดซ้าย-ขวา

FILTER

กรณี leg movement ยังไม่มี filter

Notch Filter กรองความถี่เฉพาะ (default : 50 Hz)

```
[ ]  # highpass before lowpass
     b0 ,a0  = signal.iirnotch(50,20,256)
     b1, a1  = signal.butter(2,10  /256*2,btype='high')
     b2, a2  = signal.butter(2,100 /256*2,btype='low')

     Data_EMG = signal.filtfilt(b0,a0,REMG)
     Data_EMG = signal.filtfilt(b1,a1,Data_EMG)
     REMG_new = signal.filtfilt(b2,a2,Data_EMG)
```

input เป็น list ,output เป็น array

KNOWLAGE

ความรู้ตามทฤษฎีประกอบการหา Algorithm

Snore

- period ≈ 1 s
- บาง event เกิดติดๆกันน้อยกว่า 1 s

Leg Movement

- period ≈ 0.5 - 10 s
- หากจุดเริ่มห่างกันไม่เกิน 5 s จะนับเป็น event เดียวกัน (นำไปใช้ตอนตัดค่าติดกันมากไป)
- ถ้าสองข้างซ้ำกันนับแค่ข้างซ้าย

DATA ANALYSIS

STEP 4

สร้าง for loop ที่มี range เท่ากับจำนวน index ของ signal และใช้ Threshold ที่ได้มาตั้งเงื่อนไขภายใน loop และเก็บผลที่เป็นไปตามเงื่อนไขไว้ในรูปแบบ list

```
start = []
step = 100
for i in range(0,len(Snore_bf),step):
  if power_sp(Snore_bf[i:i+150]) > 1000:
      if (abs(Snore_bf[i+55:i+95]).max() > 11 ).bool():
            start.append(i)
```

peak

ภาพตัวอย่าง code (เป็นส่วนหนึ่งใน function)

- Snore_bf คือ dataframe snore signal ของผู้ตรวจ
- power_sp คือ function หา power spectrum
- step คือ ระยะในการตรวจวนลูป ซึ่งถ้าเพิ่มจะยิ่งใช้เวลารันเร็วขึ้น แต่ความละเอียดจะลดลงและอาจผิดพลาดมากขึ้น (default : 100)
- ช่วงที่ใช้ในการเช็คขึ้นกับระยะของ event นั้นๆ
- ช่วง peak จะเช็ค mid period +- default : 20 และใส่ abs เพื่อเช็คด้านลบด้วย

STEP 3

เช็คค่าที่ได้ด้วยคำสั่ง value.counts() เพื่อหาค่ามาทดลองตั้งค่า Threshold ที่เหมาะสม

```
[ ] df_new['peak'].value_counts() # 28 ค่าที่แตกต่างกันตั้งแต่ 3 ถึง 129 V

15.0    236
12.0    197
18.0    163
21.0     80
24.0     60
27.0     52
9.0      48
30.0     41
33.0     28
36.0     26
42.0     20
39.0     20
45.0     16
48.0     14
54.0     10
57.0      9
51.0      6
63.0      3
60.0      3
66.0      2
84.0      2
69.0      2
87.0      2
6.0       2
72.0      1
81.0      1
129.0     1
3.0       1
Name: peak, dtype: int64
```

*การคำนวณความถูกต้องใช้กับทุกการหา Algorithm

STEP 5

นำผลที่ได้มาตัดค่าที่ติดกันน้อยกว่าที่กำหนดและเช็คจำนวน peak

```
[ ]
    start_new = [start[0]]
    for i in range(1,len(start)-1):
      if start[i] - start[i-1] > 100 :
        n = 0
        for j in Snore_bf['SNORE'][start[i] : start[i]+150]:
          if abs(j) > 11 :
            n += 1
        if n > 4 :
          start_new.append(start[i])
    print(len(start_new),start_new)
```

ตัดค่าที่ห่างกัน < 100

check จำนวน peak

ภาพตัวอย่าง code (เป็นส่วนหนึ่งใน function)

- นำผลที่ได้ก่อนนี้มาเช็คต่อ โดยชั้นแรกจะตัดค่าที่ติดกันมากเกินไป default : period per event
- ส่วนสองจะเช็คจำนวน peak โดยใช้ค่าจากการสร้างเงื่อนไขในตอนแรก ส่วนจำนวนใช้การทดลองสุ่มตามความเหมาะสม แนะนำเริ่มจาก 5 (n = จำนวน peak)

STEP 6

คำนวณความถูกต้อง Part 1

```
[ ]  correct = []
     # for i in df_new['start_cal']:  # real data
     for i in df_new['start']:  # real data
       diff = {}
       for j in start_new: # test data :
         if abs(j-i) < 256:
           diff[i] = abs(j-i)
       if diff == {}:
         diff[i] = -1
       correct.append(diff)
     print(len(correct),correct)
```

ภาพตัวอย่าง code (@github)

- หาค่ามีใกล้เคียงกับผลที่ได้จากการ score ของผู้เชี่ยวชาญโดยวน loop เช็คเงื่อนไขผลต่าง
- จะเก็บในรูป dict {ตำแหน่ง:ผลต่าง} ถ้าผลต่างมากกว่าเกณฑ์จะตั้งค่าเป็น -1
- เก็บค่าไว้เพื่อไปคำนวณ percent ความถูกต้อง

```
[ ] n_correct = 0
    check_t = []
    for i in correct:
      if list(i.values())[0] > -1 :
        n_correct += 1
        check_t.append(i)

    print(round((n_correct/len(df_new))*100,2),n_correct,check_t)
    print(round((n_correct/len(start_new))*100,2))
```

STEP 7

คำนวณความถูกต้อง Part 2

ภาพตัวอย่าง code (@github)

- check จำนวนของตำแหน่งที่เป็นไปตามเงื่อนไขและนำมาคำนวณเทียบกับ
  - จำนวน event จากการ score (เช็คความถูกต้อง)
  - จำนวน event จาก algoritm (เช็คสัดส่วน detect ถูกเทียบกับไม่ถูกต้อง)

OPTIONAL

กรณี leg Movement หากตำแหน่งสองข้างอยู่ใกล้กัจะนับแค่ที่ขาซ้าย

```
for i in st_R:
  for j in st_L:
    if abs(i-j) < 512 :
      st_R = [s for s in st_R if s != i]
```

ภาพตัวอย่าง code (ส่วนหนึ่งใน function)

- check ตำแหน่งของซ้ายและขวาที่ใกล้เคียงกันน้อยกว่า 2 s
- หากใกล้กันจะตัดของข้างขวาออก

ฟังก์ชัน return ตำแหน่ง peak >= h

ความคิดจากการทดลอง ss48

ANOTHER IDEA

ใช้ฟังก์ชัน find_peak()

```
def findbypeak(data_array,h):
    peak_p , _ = signal.find_peaks(data_array,height=h)
    peak_n , _ = signal.find_peaks(-data_array,height=h)
    peakp = pd.DataFrame(np.array(peak_p),columns=['peak'])
    peakn = pd.DataFrame(np.array(peak_n),columns=['peak'])
    peak_all = pd.concat([peakp,peakn],axis=0)
    peak_all = peak_all.sort_values(by='peak',ignore_index=True)
    peak = peak_all['peak'].tolist()
    return peak

Data_sn = filter(file_snore_array)

for i in range(0,len(Data_sn),200):
    if power_sp(Data_sn[i:int(i+150*2.56)]) > pw:
        start.append(i)

start2 = []
p = findbypeak(Data_sn,h)
for i in start:
    n = 0
    for j in p:
        if (i <= j) and (j <= i+256):
            n += 1
    if n >= 5:
        start2.append(i)

start_new = [start2[0]]
for i in range(1,len(start2)-1):
  if start2[i] - start2[i-1] > 200:
    start_new.append(start2[i])
```

เช็คช่วงที่มี Power Spectrum มากกว่าค่าที่กำหนดโดย Default

- Snore : 2000
- Leg movement : 10000

เลือกแค่ช่วงที่มีจำนวน peak >= 5

เลือกแค่ช่วงที่มีจำนวน peak >= 5

หมายเหตุ : parameter --> h in findbypeak() มี default

- Snore : 30
- Leg movement กรณีปกติ : 100

  กรณีรันนาน (สันนิฐานว่าnoiseเยอะ) : ตั้ง (lower,upper) = (h1,h2) = (200,2000)

HOW TO FIND ALGORITHM (DESATURATION)

*ส่วนตัวจะแปลงเป็น dataframe เพื่อง่ายต่อการสังเกตข้อมูล

STEP 1

เนื่องจากในโปรแกรมแสดงค่าทุกๆ 768 index จึงต้องเลือกข้อมูลเพื่อให้สัมพันธ์กัน

```
dsat2 = dsat1[0:len(dsat1):768]
dsat2
```

	start	spo2
0	0	97
768	768	97
1536	1536	97
2304	2304	97
3072	3072	97
...	...	...
8739072	8739072	0
8739840	8739840	0
8740608	8740608	0
8741376	8741376	0
8742144	8742144	0

11384 rows × 2 columns

STEP 2

ศึกษาลักษณะของเหตุการณ์เพื่อการนำไปเขียน Algorithm

- Oxygen Desaturation คือการลดลงของค่าความอิ่มตัวออกซิเจนมากกว่าหรือเท่ากับร้อยละ 3
- เก็บจุดที่ออกซิเจนเริ่มลดลงจนมีค่าตามกำหนดและเก็บจุดสุดท้ายก่อนออกซิเจนจะมีค่าเพิ่มขึ้น
- จากคำแนะนำของ technician เนื่องจากความแกว่งของสัญญาณ หากเหตุการณ์เกิดขึ้นห่างกันน้อยกว่า 2 วินาทีให้นับรวมเป็นเหตุการณ์เดียวกัน

STEP 3

เขียน Algorithm

*ตัวอย่างโค้ดที่ลองเขียนก่อนนำไปใส่ใน def (ฉบับสมบูรณ์ใน github)

```
start_ds_2= {}
end_ds_2 = {}
b = 1
c = 0
t = 1000
l = len(dsat2)
m = 0

# for i in range(c,l-1000):
for i in range(l):
    new_spo2_2 = spo2_2[i]
    if (spo2_2[i] <= 50):
      if spo2_2[i-1] <= 50:
        continue
      else : new_spo2_2 = spo2_2[i-1]
    elif spo2_2[i+1] < new_spo2_2 :
      for b in range(1,t):
        n_spo2_2 = spo2_2[i+b]
        if (n_spo2_2 <= 50):
          n_spo2_2 = m
        else : m = n_spo2_2
        if (spo2_2[i+b+1] > n_spo2_2 ):
          if n_spo2_2 <= 0.97*new_spo2_2:
            if (start_2[i+b] not in end_ds_2)&(start_2[i+b]-start_2[i] > 768)&(start_2[i+b]-start_2[i] <= 20000):
            # if (start_2[i+b]-start_2[i] > 768)&(start_2[i+b]-start_2[i] <= 20000):
              start_ds_2[start_2[i]] = new_spo2_2
              end_ds_2[start_2[i+b]] = n_spo2_2
              c = start_2[i+b+1]
              # print(len(dsat2[start_2[i]:start_2[i+b+1]]),dsat2[start_2[i]:start_2[i+b+1]])
              break
          break


print(len(start_ds_2),start_ds_2)
print(len(end_ds_2),end_ds_2)
```

```
[ ] start_ds_new = []
    end_ds_new = []
    for i in range(len(end_ds_l)-1):
      if (start_ds_l[i+1] - end_ds_l[i] <= 768*2):
        for j in range(1,5):
          if (start_ds_l[i+j+1] - end_ds_l[i+j] > 768*2):
            start_ds_new.append(start_ds_l[i])
            end_ds_new.append(end_ds_l[i+j])
            break
      elif (start_ds_l[i] not in start_ds_new)&(end_ds_l[i] not in end_ds_new):
        start_ds_new.append(start_ds_l[i])
        end_ds_new.append(end_ds_l[i])
```

รวมเหตุการณ์ที่มีระยะห่างกันไม่เกิน 2 วินาที (ปรับเปลี่ยนได้)

for loop 1st จะเช็คสองค่าแรกของเลขนั้นๆ หากเป็นไปตามเงื่อนไขถึงไปเช็คค่าต่อใน loop 2nd

หากค่าน้อยกว่าเท่ากับ 50 ให้ข้ามหรือให้เท่ากับค่าก่อนหน้า (เพราะเป็นค่าที่เกิดจากความผิดพลาดขณะวัดสัญญาณ)

for loop 2nd จะเช็คค่าต่อไปเรื่อยๆ จะหยุดเมื่อค่าต่อไปมากกว่าและค่านั้นมีค่าลดลงจากค่าตั้งต้นมากกว่าเท่ากับร้อยละ 3

จากทฤษฎีระยะของ dsat ส่วนใหญ่มากกว่า 768 จุด (3 s) และจำกัดไม่เกิน 20000 จุด

*หลักการคือใช้ range เช็คตำแหน่ง ใช้ในการไล่ค่าและการเปรียบเทียบค่าก่อนหลัง และตั้งให้จุดๆหนึ่งเท่ากับตัวแปรหนึ่งเพื่อให้สามารถย้อนกลับมาเปรียบเทียบได้

HOW TO FIND ALGORITHM (APNEA)

*ส่วนตัวจะแปลงเป็น dataframe เพื่อง่ายต่อการสังเกตข้อมูล

STEP 1

Data transformation กับข้อมูล info เช่นเดียวกับ snore และ leg movement

	event_name	start	period(256)
70	Obstructive Apnea	567119.0	4326.4
77	Obstructive Apnea	580031.0	4608.0
95	Obstructive Apnea	620693.0	6425.6
99	Obstructive Apnea	629723.0	5785.6
103	Obstructive Apnea	638012.0	6912.0
...	...	...	...
1450	Obstructive Apnea	4171931.0	9139.2
1455	Obstructive Apnea	4183007.0	10086.4
1459	Obstructive Apnea	4196355.0	9241.6
1464	Obstructive Apnea	4208023.0	9472.0
1468	Obstructive Apnea	4219100.0	9472.0

238 rows × 3 columns

STEP 2

ศึกษาลักษณะของเหตุการณ์เพื่อการนำไปเขียน Algorithm

- Apnea หรือภาวะหยุดหายใจ คือการลดลงของความสูงของคลื่นการหายใจเมื่อเทียบกับความสูงพื้นฐานก่อนเกิดเหตุการณ์ มากกว่าหรือเท่ากับร้อยละ 90 ซึ่งมีระยะมากกว่าหรือเท่ากับ 10 วินาที
- นับแค่เหตุการณ์ที่เกิดซ้ำกันแค่ใน PFLOW และ FLOWTH

จุดเริ่ม : peak ด้านลบ

จุดสุดท้าย : peak ด้านบวก

FILTER

input เป็น list ,output เป็น array

```
[ ] b1, a1  = signal.butter(2,0.05  /256*2,btype='high')
    b2, a2  = signal.butter(2,1 /256*2,btype='low')

    Data_pf = signal.filtfilt(b1,a1,pflow)
    Data_pf = signal.filtfilt(b2,a2,Data_pf)
```

```
[ ] b1, a1  = signal.butter(2,0.05  /256*2,btype='high')
    b2, a2  = signal.butter(2,1 /256*2,btype='low')

    Data_fth = signal.filtfilt(b1,a1,flowth)
    Data_fth = signal.filtfilt(b2,a2,Data_fth)
```

ใช้ high pass และ low pass ที่ความถี่หักมุม 0.05 และ 1 ตามลำดับ
*อาจปรับเปลี่ยนได้ตามลักษณะสัญญาณแต่ละคน

KNOWLAGE

คำสั่ง signal.find_peak() จาก scipy

Parameter ที่ใช้
signal.find_peak(data, distance=d, height=h)

- data : ข้อมูลที่ต้องการหา peaks
- distance : เลือกแค่ peaks ที่ห่างระยะ d จุด
- height : เลือกแค่ peaks ที่สูงมากกว่าเท่ากับ h

STEP 3

เขียน Algorithm

*ตัวอย่างโค้ดที่ลองเขียนก่อนนำไปใส่ใน def (ฉบับสมบูรณ์ใน github)

แนวคิด

หา peaks ที่ความสูงมากกว่าเท่ากับค่าหนึ่งที่ห่างกันกันมากกว่า 10 วินาที แล้วตรวจสอบว่า หากเป็นจุดเริ่มให้ไปเช็คค่าที่ใกล้เคียง peak ด้านลบ จุดท้ายไปเช็คค่าที่ใกล้เกียงกับ peak ด้านบวก

```
[ ] x = Data_fth
    h = 40
    peak_p , _ = signal.find_peaks(x,height=h,distance=256)
    peak_n , _ = signal.find_peaks(-x,height=h,distance=256)
    # peak_p , _ = signal.find_peaks(x,height=h)
    # peak_n , _ = signal.find_peaks(-x,height=h)
```

```
[ ] print(peak_p)
    print(peak_n)

    [    382    5310   13214 ... 7851437 7852075 9411290]
    [   3831   11622   12272 ... 7851197 7922874 7926868]
```

```
[ ] peakp = pd.DataFrame(np.array(peak_p),columns=['peak'])
    peakn = pd.DataFrame(np.array(peak_n),columns=['peak'])
```

```
[ ] stt = df_OA['start'].astype(int).tolist()
    ent = df_OA['end'].astype(int).tolist()
```

```
[ ] peak_all = pd.concat([peakp,peakn],axis=0)
    # peak_all.reset_index(drop=True, inplace=True)
    peak_all = peak_all.sort_values(by='peak',ignore_index=True)
    peak_all
```

	peak
0	382
1	3831
2	5310
3	11622
4	12272
...	...
3997	7851437
3998	7852075
3999	7922874
4000	7926868
4001	9411290

4002 rows × 1 columns

1

เริ่มจากหาและรวม peaks ทั้งด้านบวกและด้านลบทั้งหมด

2

สร้าง list ที่ใช้ใน
การตรวจสอบ peaks

```
[825] peak = peak_all['peak'].tolist()
```

```
[827] start = []
      end = []
      for i in range(1,len(peak)-1):
        if (peak[i+1] - peak[i] >= 2560)&(peak[i+1] - peak[i] <= 15000):
          if (peak[i] not in start)&(peak[i+1] not in end):
            start.append(peak[i])
            end.append(peak[i+1])
        elif (peak[i] - peak[i-1] >= 2560)&(peak[i] - peak[i-1] <= 15000):
          if (peak[i-1] not in start)&(peak[i] not in end):
            start.append(peak[i-1])
            end.append(peak[i])
      print(len(start),start)
      print(len(end),end)
```

ระยะเหตุการณ์มากกว่า
เท่ากับ 10 วิ

3

สร้าง for loop if-else
ตามเงื่อนไขที่ต้องการ

STEP 4

รันโค้ดข้างต้นทั้งใน FLOWTH และ PFLOW จากนั้นมาคัดเลือกเหตุการณ์ที่เกิดใกล้กันและนับเป็น 1 เหตุการณ์ โดยใช้จุดเริ่มของ FLOWTH เป็นหลัก

```
[ ]  #len(start_pf)<len(start_fth)
     stf = []
     enf = []
     if len(start_pf) >= len(start_fth):
       second_loop = start_fth
     else: second_loop = start_pf
     for i in range(len(start_fth)):
       m = 5000
       s = 0
       e = 0
       for j in range(len(second_loop)):
         if (abs(start_pf[j]-start_fth[i]) < m)&(abs(end_pf[j]-end_fth[i]) < m):
           m = abs(start_pf[j]-start_fth[i])
           s = start_fth[i]
           e = end_fth[i]
       if s != 0:
         stf.append(s)
         enf.append(e)
     print(len(stf),stf)
     print(len(enf),enf)
```

OPTIONAL

โค้ดเพิ่มเติมเพื่อการไปต่อยอดในอนาคต ทดลองรันเพื่อลดความผิดพลาด เมื่อนำไปทดลองแล้วสามารถลดได้จริงแต่ก็ตัดสัญญาณที่ถูกต้องไปด้วยจึงยังไม่นำไปใส่ในฟังก์ชัน

ปัญหา : สัญญาณระหว่าง peaks ที่ตรวจจับได้ไม่เรียบตามทฤษฎี

```
start_final = []
end_final = []

for i in range(len(start)):
  st = start[i]-d
  en = end[i]+d
  x = abs(airflow_new['fth_f'][st:en])
  y = Data_fth[st:en]
  # print(sum(x))
  if sum(x) <= 200000:
    start_final.append(st)
    end_final.append(en)
```

ผลรวมไม่เกิน 200000

1

แนวคิดแก้ไข : นับผลรวมของ peaks ระหว่างจุดเริ่มและจุดท้ายให้มีค่าไม่เกินค่าค่าหนึ่ง (ค่านั้นสามารถหาได้จาก Data Analysis)

def count_peak ใน github

```
[ ] h1 = 100
    start_final = []
    end_final = []
    for i in range(len(start)):
      st = start[i]
      en = end[i]
      x = Data_fth[st:en]
      peak_p , _ = signal.find_peaks(x,height=(20,h1),distance=256)
      peak_n , _ = signal.find_peaks(-x,height=(20,h1),distance=256)
      if (len(peak_p)<3)&(len(peak_n)<3) :
        start_final.append(st)
        end_final.append(en)
```

หา peaks ขนาด 20 - 100 ระหว่างจุดเริ่มแสิ้นสุด

เลือกแค่ที่มีจำนวน peaks น้อยกว่า 3

2

แนวคิดแก้ไข : นับ peaks ระว่างจุดเริ่มและสิ้นสุดให้มีค่าไม่เกินค่าค่าหนึ่ง

PARAMETER
ที่ปรับค่าได้

*เปลี่ยนแปลงได้ตาม Data Analysis

SNORE

Step ที่ใช้รันปรับได้ ยิ่งเยอะจะยิ่งเร็วแต่ละเอียดน้อยลง (default = 200)
ใน Loop if-else

- เก่า - Power spectrum (Vary ได้ 1000,2000,3500)
  - Peak (Vary 9 - 15 ,default = 11)
- ใหม่ - pw = 2000 , h = 30 (นับแค่ peak สูง >= h) , n >= 5

LEG MOVEMENT

def get_start()

- เก่า - Power spectrum (default = 10000)
  - Peak (Vary 75-100 ,default = 100)
- ใหม่ - pw (default = 10000) , h = 100 : default (นับแค่ peak สูง >= h, ถ้าสัญญาณมี noise เยอะจะกำหนด lower,upper bound (h1,h2) default (200,1000) , n >= 5

DESATURATION

t : ระยะวนใน loop ที่สอง (default = 1000)
th : ค่าขั้นต่ำที่จะนำมาคิดใน Algorithm (default = 50)
d : ต้องการรวมเหตุการณ์ที่ห่างกันน้อยกว่าเท่ากับ d วินาที

APNEA

def filter()

- lowpass : สามารถปรับค่าความถี่หักมุมได้ตามสภาพสัญญาณ (defaulf = 1, Vary 0.5,1)

def findbypeak()

- h : ความสูงของ peaks ขั้นต่ำที่ต้องการหาทั้งหมด (default = 40, Vary 15,20,40)

m : ระยะห่าง index ของ PFLOW,FLOWTH ที่จะนับรวมเป็น 1 event (default = 5000)

LINK COLAB

SNORE

Google Colaboratory
google.com

LEG MOVEMENT

Google Colaboratory
google.com

DESATURATION

Google Colaboratory
google.com

APNEA

Google Colaboratory
google.com

*กดลิ้งได้ใน canva (Canva summary)