In [273]:

```
import numpy as np
import heartpy as hp
import sensormotion as sm
import pandas as pd

from scipy.signal import butter,filtfilt,find_peaks
from scipy import stats
from itertools import product
import glob

import matplotlib.pyplot as plt
import plotly_express as px
```

1 Read files / Adjust Timestamps

In [241]:

```
bvp_file_list = glob.glob("Performance3/*_bvp.csv")
eda_file_list = glob.glob("Performance3/*_eda.csv")
acc_file_list = glob.glob("Performance3/*_acc.csv")
```

In [242]:

```
df_bvp = dict()
for file in bvp_file_list:
    df_bvp[file[13:16]] = pd.read_csv(file).query('signalOK == 1')

df_eda =dict()
for file in eda_file_list:
    df_eda[file[13:16]] = pd.read_csv(file)

df_acc =dict()
for file in acc_file_list:
    df_acc[file[13:16]] = pd.read_csv(file)
```

In [243]:

```
sublist = ['301','302','303', '304','305','306', '309', '312','313', '314', '317','319', '321', '322', '325', '326','327','328','329','330',
 '331', '334','335','340','341','342']
```

In [244]:

```
def set_timepass(data, start):
    data['timepass'] = data['localTime'] - start
    data['timepassSec'] = round(data['timepass']/1000, ndigits= 0)
    data['timepassMin'] = round(data['timepassSec']/60, ndigits= 0)
    data = data.query('20< timepassMin< 95').reset_index()    # performance time duration
    return data
```

In [245]:

```
concated_list = []
for key in df_bvp.keys():
    concated_list.append(df_bvp[key])
df_all = pd.concat(concated_list,axis=0)
earliestTime = df_all['localTime'].min()

for key in sublist:
    df_bvp[key] = set_timepass(df_bvp[key], earliestTime)
    df_eda[key] = set_timepass(df_eda[key], earliestTime)
    df_acc[key] = set_timepass(df_acc[key],earliestTime)
```

2 Clean/Filter

In [246]:

```
# calculate the root sum of squared three dimensional acc values
rss_std = dict()

for key in sublist:
    b, a = sm.signal.build_filter(frequency=2, sample_rate=50, filter_type='low', filter_order=4)

    df_acc[key]['accX_filtered'] = sm.signal.filter_signal(b, a, signal=df_acc[key]['accX'])
    df_acc[key]['accY_filtered'] = sm.signal.filter_signal(b, a, signal=df_acc[key]['accY'])
    df_acc[key]['accZ_filtered'] = sm.signal.filter_signal(b, a, signal=df_acc[key]['accZ'])

    df_acc[key]['rss'] = np.sqrt(df_acc[key]['accX_filtered']**2+df_acc[key]['accY_filtered']**2+df_acc[key]['accZ_filtered']**2)
    rss_std[key] = df_acc[key]['rss'].std()

    df_acc[key]['rss_peaks_max'] = df_acc[key].iloc[ find_peaks(df_acc[key]['rss'], distance =25 ,prominence=rss_std[key]*1.5)[0]]['rss']
    df_acc[key]['rss_peaks_min'] = df_acc[key].iloc[ find_peaks(df_acc[key]['rss']*-1, distance =25 ,prominence=rss_std[key]*1.5)[0]]['rss']
```

In [247]:

```
#bvp low pass filter
freq = 50
fc = 3.5 # Cut-off frequency of the filter
w = fc / (freq / 2) # Normalize the frequency
b, a = butter(2, w, 'low')

for key in sublist:

    df_bvp[key] = pd.merge(df_bvp[key],df_acc[key],on=['remoteTime','timepass','timepassSec','timepassMin'],how = 'inner')
    df_bvp[key]['peaks_combined'] = df_bvp[key]['rss_peaks_max'].astype(str) + df_bvp[key]['rss_peaks_min'].astype(str)
    droplist = df_bvp[key][df_bvp[key]['peaks_combined'] !='nannan']['timepassSec'].unique().tolist()
    df_bvp[key].drop(df_bvp[key][df_bvp[key]['timepassSec'].isin(droplist)].index,inplace = True)

    df_bvp[key]['bvp_filtered'] = filtfilt(b, a, df_bvp[key]['bvp'])
    df_bvp[key] = df_bvp[key].loc[:,['remoteTime','timepass','timepassSec','timepassMin','bvp','bvp_filtered']]
```

3 Explore/Plot

In [248]:

```
def plot_peak_signal( start, end, data, time, signal1, signal2,marker1):

    plt.figure(figsize=(20,5))
    plt.scatter(data[start:end][time], data[start:end][signal2], c='orange', s =10)
    plt.scatter(data[start:end][time], data[start:end][signal1], c='blue', s =10)
    plt.plot(data[start:end][time], data[start:end][signal2], c='grey')

    plt.plot(data[start:end][time], data[start:end][marker1], 'o', c='green')

    plt.show()
```

3.1 Detect RR-intervals

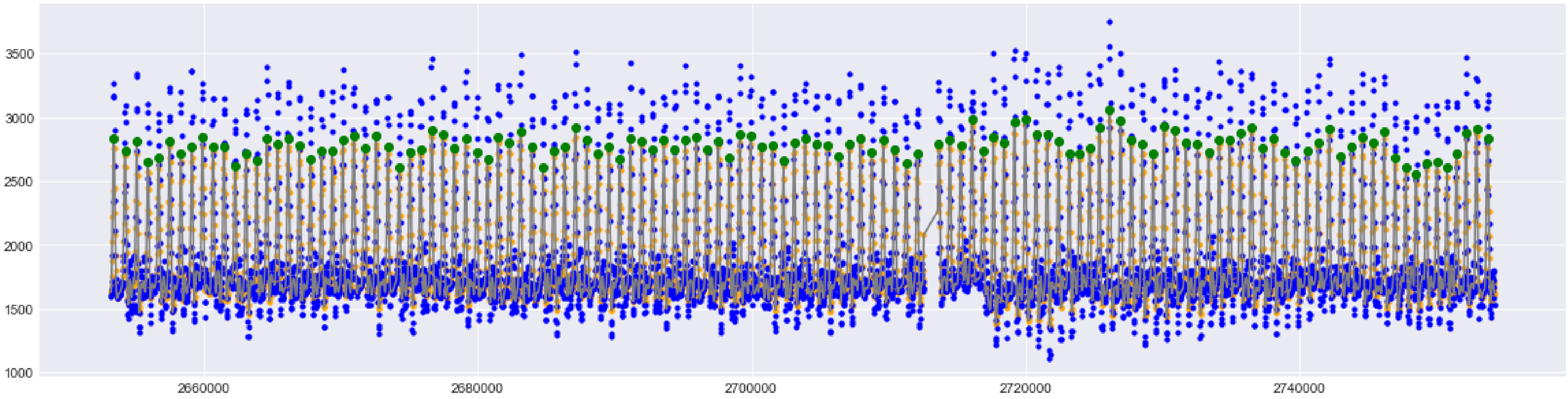
In [249]:

```
peaks = dict()  # bvp peaks
rr = dict()  # inter-beat-interval (RR interval)
bpm = dict() # beats per minute


for key in sublist:

    w,m = hp.process(df_bvp[key]['bvp_filtered'].to_numpy(),50, clean_rr=True)
    peaks[key] = w['peaklist']

    rr[key] = w['RR_list']

    bpm[key] = m['bpm']

    p = np.array(peaks[key])
    df_bvp[key]['peaks'] = df_bvp[key].iloc[p]['bvp_filtered']
```

...

In [250]:

```
plot_peak_signal(20000,25000, df_bvp['328'],'remoteTime', 'bvp','bvp_filtered','peaks' )
```



3.2 Sliding Window

In [251]:

```
WINDOW = 240
INTERVAL = 240 * 0.5

df_sliced = dict()
for key in sublist:
    i = 1440
    df_sliced[key] = []
    while True:
        mask = (df_bvp[key]['timepassSec']>= i )& (df_bvp[key]['timepassSec'] < i + WINDOW)
        df_sliced[key].append(df_bvp[key][['remoteTime','timepass','timepassSec','timepassMin', 'bvp','bvp_filtered']][mask])
        i = i + INTERVAL
        if i >5600:
            break
```

In [252]:

```
rr_intervals_window = dict()
for key in sublist:
    value = df_sliced[key]
    rr_intervals_window[key] =[]
    for i in range (len(value)):
        w,m = hp.process(value[i]['bvp_filtered'].to_numpy(), 50, clean_rr= True)
        rr_intervals_window[key].append(w['RR_list'])
```

...

3.3 Generate RR list

In [253]:

```
from hrvanalysis import remove_outliers, remove_ectopic_beats,  interpolate_nan_values
from boltons.iterutils import remap
```

In [254]:

```
#remove outliers
rr_intervals_without_outliers = dict()
for key in sublist:
    value = rr_intervals_window[key]
    rr_intervals_without_outliers[key] = []
    for i in range(len(value)):
        rr_intervals_without_outliers[key].append(remove_outliers(rr_intervals = value[i], low_rri =200, high_rri =1500))

    drop_falsey = lambda path, key, value: bool(value)
    rr_intervals_without_outliers[key] = remap(rr_intervals_without_outliers[key], visit=drop_falsey)
```

...

In [255]:

```
# remove ectopic beats
nn_list_window = dict()
for key in sublist:
    value = rr_intervals_without_outliers [key]
    nn_list_window[key] =[]

    for i in range(len(value)):
        nn = remove_ectopic_beats(rr_intervals = value[i], method = "malik")
        nn_array = np.asarray(nn)

        result = nn_array[np.logical_not(np.isnan(nn_array))]
        nn_list_window[key].append(result.tolist())
```

...

3.4 Generate HRV features

In [258]:

```
from hrvanalysis import get_time_domain_features, get_frequency_domain_features
```

In [259]:

```
timedomain_features = dict()
frequencydomain_features = dict()
for key in sublist:
    print(key)
    value = nn_list_window[key]
    timedomain_features[key] = []
    frequencydomain_features[key] = []
    for i in range(len(value)):
        if len(value[i])>2:
            timedomain_features[key].append(get_time_domain_features(value[i]))
            frequencydomain_features[key].append(get_frequency_domain_features(value[i]))
```

...

3.5 Plot HRV features

In [260]:

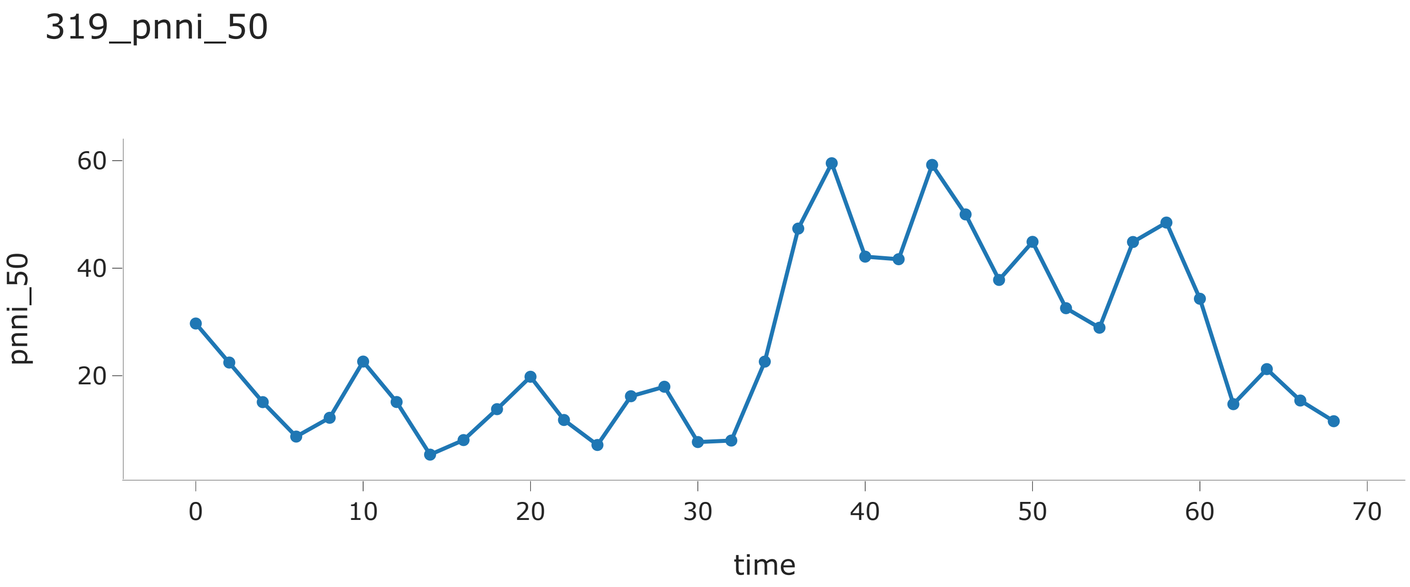
```
def data_to_plot (subject,domain, feature):
    data = []
    featurevalue = domain[subject]
    for i in range(len(domain[subject])):
        data.append(df_sliced[subject][i].iloc[0].to_frame().T)
        data[i][feature] = featurevalue[i][feature]

    dataframe= pd.concat(data)
    dataframe['time'] = dataframe['timepassMin'] -24
    dataframe['time']= dataframe['time'].apply(lambda x:  x if x % 2 == 0 else (x-1))

    return dataframe
```

In [329]:

```
def plot_hrv_feature(subject,domain,feature):
    df = data_to_plot(subject,domain,feature)
    fig = px.line(df, x="time", y=feature, title=subject+'_'+feature,width=800, height=350,template='simple_white')
    fig.update_traces(mode='markers+lines')
    fig.show()
```

In [330]:

```
plot_hrv_feature('319',timedomain_features,'pnni_50')
# change [ subject_number, hrv_domain_type ('timedomain_features'/'frequencydomain_features', feature name ]to plot
```

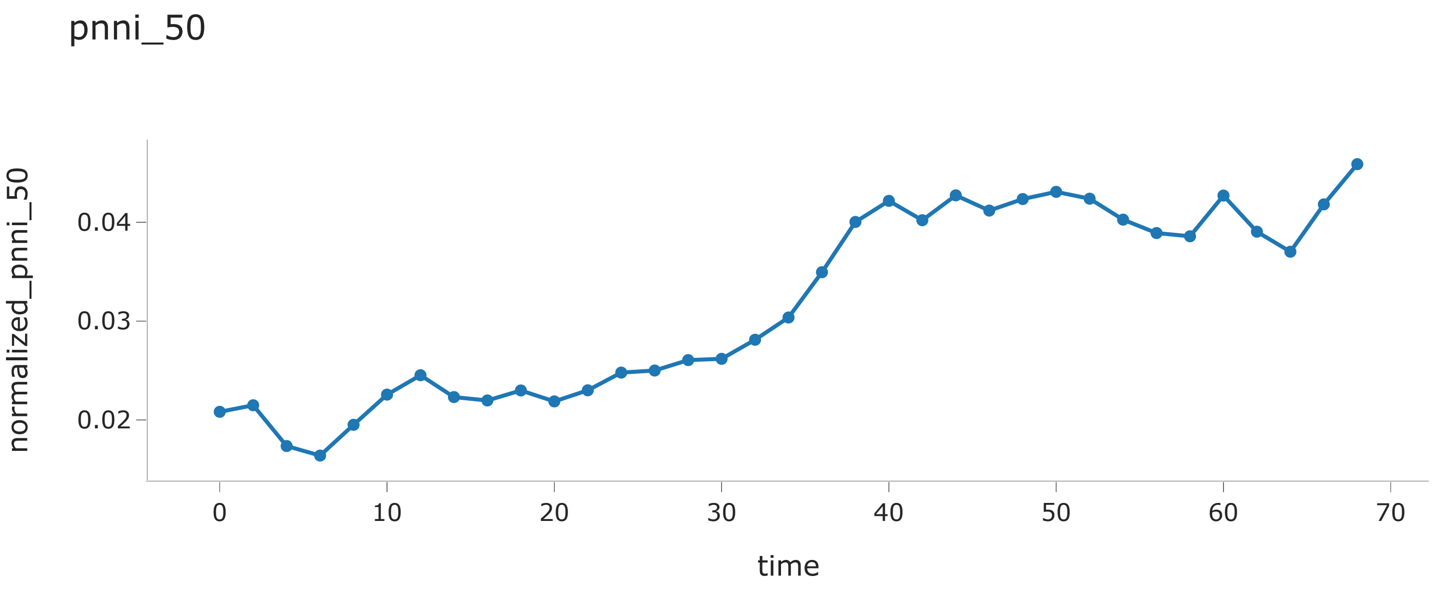


In [327]:

```
def plot_hrv_all(domain,feature):
    concated_list = []
    for subject in sublist:
        dataframe = data_to_plot(subject,domain,feature)
        dataframe['normalized_' +feature] = dataframe[feature]/(60000/bpm[subject])
        # normalized each subject's hrv feature value
        concated_list.append(dataframe)

    dataframe_all = pd.concat(concated_list, axis=0)
    df = dataframe_all.groupby('time')['normalized_'+feature].mean().reset_index()
    fig = px.line(df, x="time", y='normalized_'+feature, title= feature,width=800, height=350, template='simple_white')
    fig.update_traces(mode='markers+lines')
    fig.show()
```

In [328]:

```
plot_hrv_all(timedomain_features, 'pnni_50')
# change [ hrv_domain_type ('timedomain_features'/'frequencydomain_features', feature name ]to plot all subjects' average value
```



In [ ]: