90s_analysis

May 5, 2025

1 SLEEP APENA DATA ANALYSIS NOTEBOOK

Author: Qianxi Gong/Group SLEEP APENA
Created: May 2025
Institution: USC - BME 555

This notebook analyzes 90-second sensor recordings (IMU, SpO , heart rate) to detect patterns related to posture and sleep apnea risk. It includes data preprocessing, visualization, and comparison tools.

1.1 How to Use This Notebook

1. Add your new `.csv` file to the `data/` folder.
2. Make sure the filename reflects the activity (e.g., `standing.csv`).
3. It will automatically be imported as a NumPy array named `standing`.
4. Use analysis functions like:
   - `plot_posture_axes_pretty("standing")`
   - `plot_accel_mag_compare("laying", "standing", "sitting")`

1.2 Imports and Environment Setup

[1]:
```python
from scipy.ndimage import uniform_filter1d, median_filter
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os
```

1.3 Load and Prepare Data

Place your CSV files in the `data/` folder. Filenames should describe the activity (e.g. `tossing.csv`, `rapidbreathing.csv`, `laying.csv`).

[2]:
```python
# Get all .csv files in the dataset folder
data_dir = '90s_data'
csv_files = [f for f in os.listdir(data_dir) if f.endswith('.csv')]

print("Found CSV files:", csv_files)
# Automatically read and assign names based on filename (without extension)
for file in csv_files:
```

```
    name = os.path.splitext(file)[0]  # removes '.csv'
    globals()[name] = pd.read_csv(os.path.join(data_dir, file)).to_numpy()
```

```
Found CSV files: ['tossing.csv', 'naturalistic laying.csv', 'sitting.csv',
'mild.csv', 'rapidbreathing.csv', 'holding breath laying.csv', 'standing.csv',
'severe.csv', 'medium.csv', 'restless laying.csv']
```

1.4 Analysis Functions

[3]:
```
def plot_physio(name, csv_files=csv_files, data_dir=data_dir):
    """
    Check if 'name.csv' is in the list of csv_files, and if so, plot:
    1. Heart rate & SpO
    2. Movement signal

    Args:
        name (str): Base name of the file (e.g., 't1')
        csv_files (list): List of filenames like ['t1.csv', 't2.csv']
        data_dir (str): Folder containing the CSVs (default: 'data')
    """
    target_file = f"{name}.csv"

    if target_file not in csv_files:
        print(f"  File '{target_file}' not found in provided list.")
        return

    # Load file
    filepath = os.path.join(data_dir, target_file)
    df = pd.read_csv(filepath)
    df = df.iloc[77:].reset_index(drop=True)
    time = np.linspace(0, 90, len(df))

    # Plot Heart Rate & SpO
    bpm = df['BPM'].rolling(window=5, center=True, min_periods=1).mean()
    spo2 = df['SpO2'].rolling(window=5, center=True, min_periods=1).mean()

    plt.figure(figsize=(12, 5))
    plt.plot(time, bpm, label='Heart Rate (BPM)', color='crimson', linewidth=2)
    plt.plot(time, spo2, label='SpO  (%)', color='navy', linewidth=2,␣
 ↪linestyle='--')
    plt.title(f'Heart Rate and SpO  - {name}')
    plt.xlabel('Time (s)')
    plt.ylabel('Value')
    plt.grid(True, linestyle='--', alpha=0.4)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

```
    # Plot Movement
    movement = df.iloc[:, -1].rolling(window=5, center=True, min_periods=1).
 ↪mean()

    plt.figure(figsize=(12, 4))
    plt.plot(time, movement, color='green', linewidth=2, label='Movement')
    plt.title(f'Movement Signal - {name}')
    plt.xlabel('Time (s)')
    plt.ylabel('Movement')
    plt.grid(True, linestyle='--', alpha=0.3)
    plt.ylim(0, movement.max() * 1.1)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

```
[4]: def plot_posture_axes_pretty(name, duration=90, center=True, smooth=True,␣
 ↪smooth_size=5, show_angle=False):
    """
    Plot accelerometer X/Y/Z for posture classification from named global NumPy␣
 ↪arrays.

    Args:
        name (str): Name of the CSV file (without '.csv') previously loaded␣
 ↪into a NumPy array via globals()
        duration (int): Total assumed time (default 90s)
        center (bool): Center each axis to the mean of the first 20 samples
        smooth (bool): Apply smoothing to reduce noise
        smooth_size (int): Size of smoothing window
        show_angle (bool): Overlay scaled pitch and roll angles
    """
    if name not in globals():
        print(f" Data array for '{name}' not found.")
        return

    data = globals()[name]
    imu = data[:, 1:7]  # IMU is in columns 1-6 (Accel+Gyro)

    ax, ay, az = imu[:, 0], imu[:, 1], imu[:, 2]
    time = np.linspace(0, duration, len(ax))

    # Center
    if center:
        ax -= np.mean(ax[:20])
        ay -= np.mean(ay[:20])
        az -= np.mean(az[:20])
```

```
    # Smooth
    if smooth:
        ax = uniform_filter1d(ax, size=smooth_size)
        ay = uniform_filter1d(ay, size=smooth_size)
        az = uniform_filter1d(az, size=smooth_size)

    # Plot
    plt.figure(figsize=(12, 5))
    plt.plot(time, ax, color='#e74c3c', label='Accel X (Left/Right)',␣
↪linewidth=1.5)
    plt.plot(time, ay, color='#27ae60', label='Accel Y (Front/Back)',␣
↪linewidth=1.5)
    plt.plot(time, az, color='#2980b9', label='Accel Z (Up/Down)', linewidth=1.
↪5)

    # Optional angle overlay
    if show_angle:
        pitch = np.arctan2(ay, np.sqrt(ax**2 + az**2)) * 180 / np.pi
        roll = np.arctan2(ax, np.sqrt(ay**2 + az**2)) * 180 / np.pi
        plt.plot(time, pitch / 90, '--', color='gray', label='Pitch (scaled)')
        plt.plot(time, roll / 90, ':', color='gray', label='Roll (scaled)')

    plt.title(f'Posture Axes - {name}', fontsize=16, weight='bold')
    plt.xlabel('Time (s)', fontsize=13)
    plt.ylabel('Acceleration (g)', fontsize=13)
    plt.ylim(-1.2, 1.2)
    plt.grid(True, linestyle='--', alpha=0.3)
    plt.legend(fontsize=11, loc='upper right')
    plt.tight_layout()
    plt.show()
```

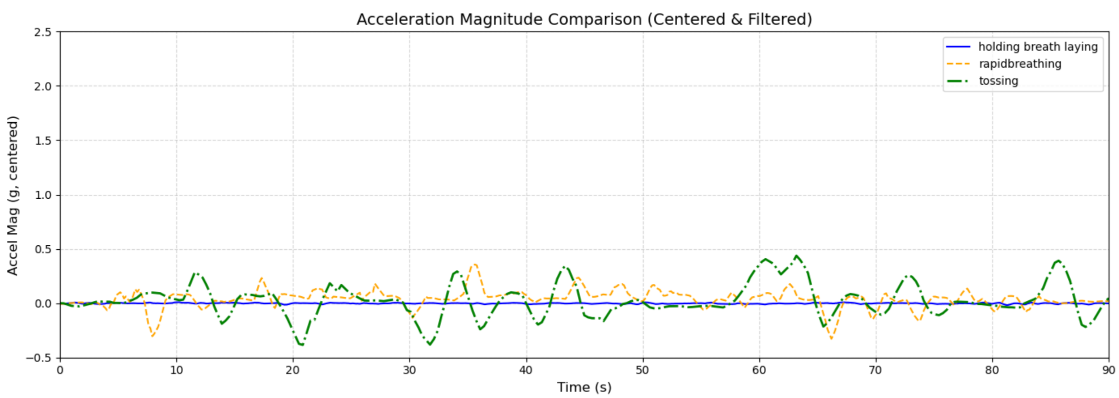
[5]:
```
def plot_accel_mag_compare(
    *names,
    duration=90,
    filter_type='median',
    filter_strength=5,
    clip_range=(0, 8),
    center=True,
    smooth_size=7
):
    """
    Plot centered, clipped, and optionally smoothed acceleration magnitude for␣
↪comparison.

    Args:
        *names (str): Names of datasets (same as .csv filename without .csv)
        duration (int): Assumed total time in seconds
```

```python
        filter_type (str): 'median' or 'none'
        filter_strength (int): Median filter window size
        clip_range (tuple): Min/max clip range for suppressing outliers
        center (bool): Zero-center each signal using first 20 samples
        smooth_size (int): Smoothing window size
    """
    plt.figure(figsize=(14, 5))

    styles = ['-', '--', '-.', ':']
    colors = ['blue', 'orange', 'green', 'red', 'purple', 'brown']

    for i, name in enumerate(names):
        if name not in globals():
            print(f" Data '{name}' not found.")
            continue

        data = globals()[name]
        imu = data[:, 1:7]  # Assumes IMU is in columns 1-6
        accel_mag = np.linalg.norm(imu[:, 0:3], axis=1)

        # Clip outliers
        if clip_range:
            accel_mag = np.clip(accel_mag, clip_range[0], clip_range[1])

        # Apply median filter
        if filter_type == 'median':
            accel_mag = median_filter(accel_mag, size=filter_strength)

        # additional smoothing

        accel_mag = uniform_filter1d(accel_mag, size=smooth_size)

        # Center the signal
        if center:
            accel_mag -= np.mean(accel_mag[:20])

        # Time axis
        time = np.linspace(0, duration, len(accel_mag))

        # Plot
        style = styles[i % len(styles)]
        color = colors[i % len(colors)]
        plt.plot(
            time, accel_mag,
            linestyle=style, color=color,
            linewidth=2 if name == "tossing" else 1.5,
            label=name
```

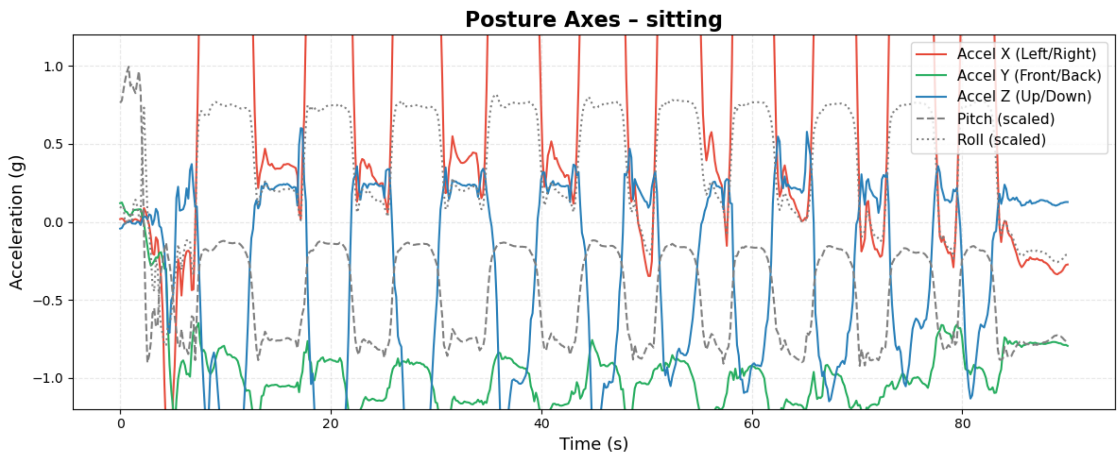
```
        )

    # Styling
    plt.title('Acceleration Magnitude Comparison (Centered & Filtered)',␣
↪fontsize=14)
    plt.xlabel('Time (s)', fontsize=12)
    plt.ylabel('Accel Mag (g, centered)', fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.legend()
    plt.xlim(0, duration)
    plt.ylim(-0.5, 2.5)
    plt.tight_layout()
    plt.show()
```

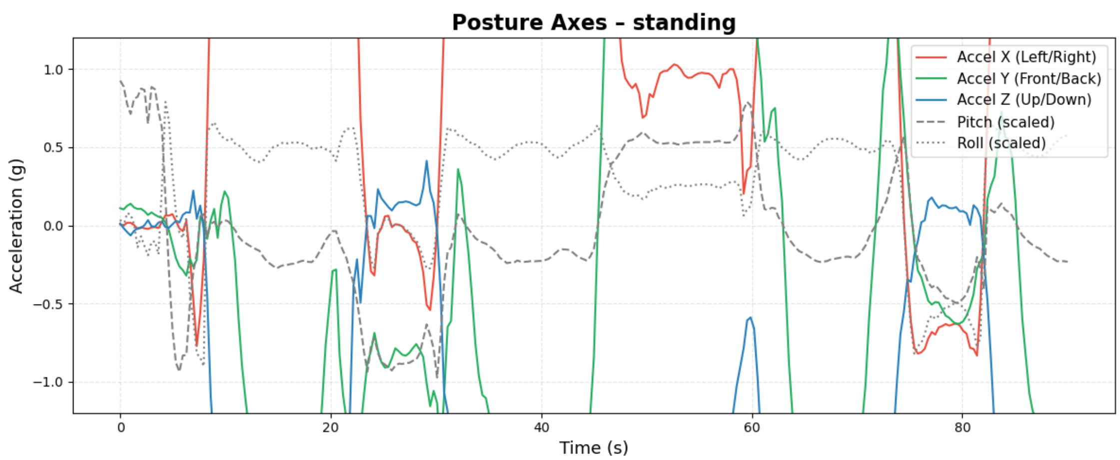
1.5 Example Visualizations

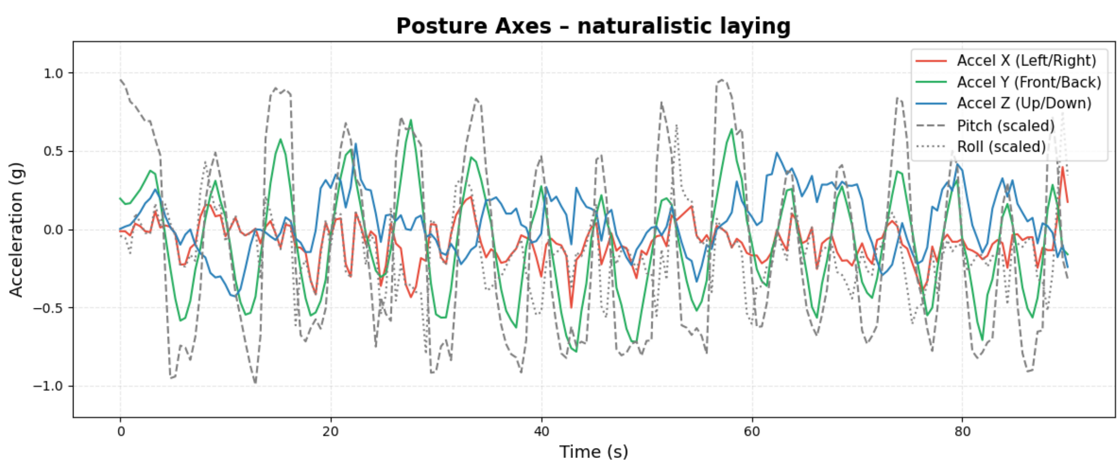
[6]:
```
plot_accel_mag_compare("holding breath laying", "rapidbreathing", "tossing")
```



[7]:
```
plot_posture_axes_pretty("sitting", show_angle=True)
plot_posture_axes_pretty("standing", show_angle=True)
plot_posture_axes_pretty("naturalistic laying", show_angle=True)
```

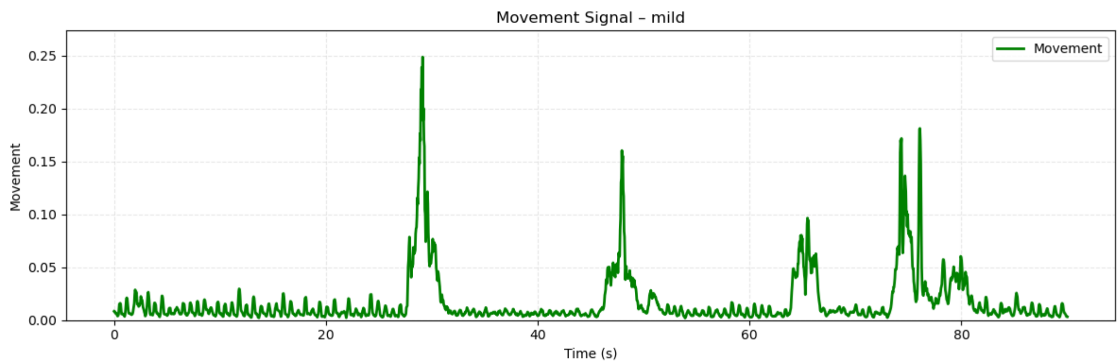

Posture Axes – sitting
Accel X (Left/Right)
Accel Y (Front/Back)
Accel Z (Up/Down)
Pitch (scaled)
Roll (scaled)
Acceleration (g)
Time (s)
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80

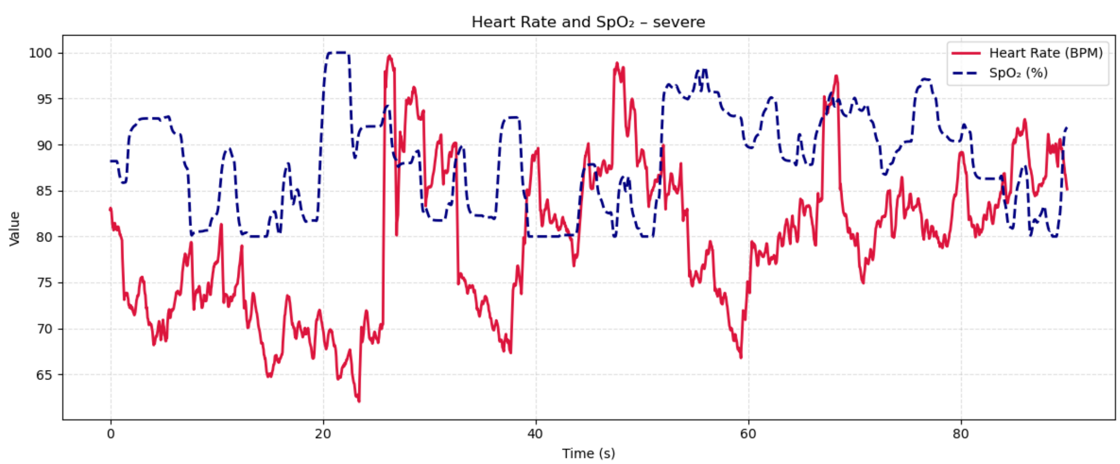

Posture Axes – standing
Accel X (Left/Right)
Accel Y (Front/Back)
Accel Z (Up/Down)
Pitch (scaled)
Roll (scaled)
Acceleration (g)
Time (s)
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80

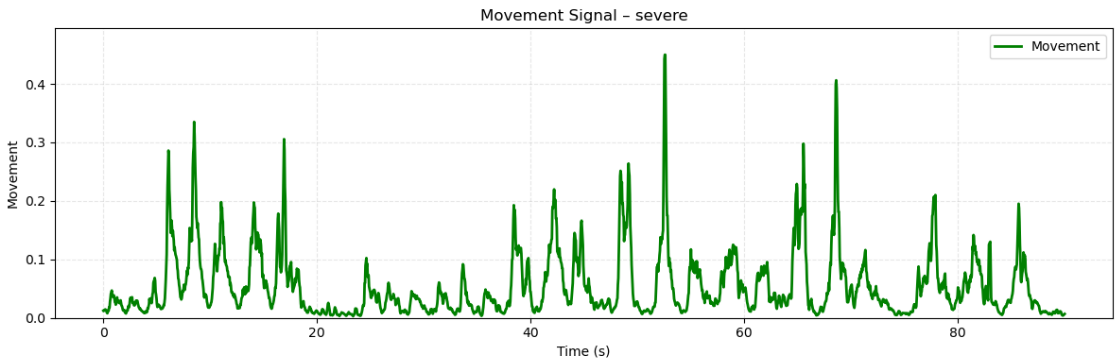

Posture Axes – naturalistic laying
Accel X (Left/Right)
Accel Y (Front/Back)
Accel Z (Up/Down)
Pitch (scaled)
Roll (scaled)
Acceleration (g)
Time (s)
1.0
0.5
0.0
−0.5
−1.0
0
20
40
60
80

[12]:
```
plot_physio('t1')
plot_physio('mild')
plot_physio('severe')
```

```
File 't1.csv' not found in provided list.
```







1.6 Metrics to Watch For

- Accel magnitude variation during movement
- Z-axis dominance when laying flat
- SpO dips under 90%
- Heart rate spikes during movement