

Accelerometer Project

May 10, 2024

1 The data

1.1 Capture24 zip file

The capture-24 zip file comprises 153 CSV files:

- The "Annotation Label Dictionary."
- The "Metadata": This includes a participant's description, indicating their gender (M or F) and age range, categorized as "18-29", "30-37", "38-52" or "53"
- 151 "PXXX" files (XXX = 001 to 151) representing the collected data for the 151 participants.

Participants' Data

For each participant, the data was collected over a period of approximately 24 hours, during 2014-2016 in the Oxfordshire area ¹. Data was collected on any day of the week (week day or weekend).

Variables

Each dataframe contains 5 variables:

- **time**: The period during which the data was collected in the format `yyyy-MM-dd HH:mm:ss.SSSSSS`.
- **x**, **y**, and **z**: The tri-axial acceleration values.
- **annotation**: Types of activities with their associated metabolic equivalent of task (MET) values.

See Annex 1 for data samples.

1.2 UKBB data

The acceleration data provides Physical Activity measurements recorded for 103,660 participants. The data is accessible through **Beluga**. The path for each data fields are provided in this document.

Participants' Data

There are two types of data provided:

- **Acceleration data - cwa format** (90001)
- **Acceleration intensity time-series data** (90004)

¹Chan Chang, S., et al. Capture-24: Activity Tracker Dataset for Human Activity Recognition. University of Oxford, 2021.

90001 - Raw data

Biobank notes: *"These files contain the raw acceleration data, per participant, from a single use (over a 7-day period) of the physical activity monitor. Format is native cwa as returned by the device. Files are typically 250MB each in the raw state and usually compress to around 100MB."*

The UKBB raw data initially comes in a CWA file extension. Once converted to CSV, there are, we can observe that there are 4 variables: `time`, `x`, `y`, and `z`.

There is data for 103,660 participants and 115,432 item counts (see: [Biobank Documentation](#)).

⇒ See Annex 2 for samples of UKBB data.

Below is the directory to access the raw (CWA) data:

Path: `/lustre03/project/6008063/neurohub/UKB/Bulk/90001`

90004 - Time series data

Biobank notes: *"These time-series files describe a participant's accelerometer measured physical activity intensity every five seconds. Each measurement represents the average vector magnitude across all samples recorded in each five second epoch/period. This file may be useful for those interested in analysing accelerometer measured physical activity by time."*

The first column contains the acceleration value where the units of measurement are milli-gravity. The second column indicates whether this is an actual measurement (value = <blank>) or imputed (value = 1). The time associated with each reading can be inferred from the very first row, which describes the start and end times of the data. Each successive reading is incremented by five seconds."

There is data for 103,660 participants and 103,660 item counts (see: [Biobank Documentation](#)).

Below is the directory to access the time series data:

Path: `/lustre03/project/6008063/neurohub/UKB/Bulk/90004`

2 Accelerometer Package

The `accelerometer` package was developed by a team at Oxford University and is designed for use with Python. It serves as a comprehensive toolkit for processing accelerometer data. The package facilitates the extraction of graphs, activity levels, and various metrics from the collected data. It also offers functionality to train custom models tailored to make specific predictions based on the dataset (train a bespoke model).

Documentation for the `accelerometer` package can be found on this [webpage](#) or their [GitHub page](#).

2.1 Processing the data

2.1.1 `accProcess`

The `accProcess` allows us to process .CSV and .CWA (raw) files. We can then create time-series files which allows us to create accelerometer graphs (shown in the "graphs" section of this document).

It also allows us to generate an epoch file with 70 features. The features generated are listed below:

<code>enmoTrunc</code>	<code>sd</code>	<code>sdroll</code>	<code>fft5</code>	<code>f2</code>
<code>enmoAbs</code>		<code>sdpitch</code>	<code>fft6</code>	<code>p2</code>
<code>xMean</code>	<code>coefvariation</code>	<code>sdyaw</code>	<code>fft7</code>	<code>f625</code>
<code>yMean</code>	<code>median</code>	<code>rollg</code>	<code>fft8</code>	<code>p625</code>
<code>zMean</code>	<code>min</code>	<code>pitchg</code>	<code>fft9</code>	<code>totalPower</code>
<code>xRange</code>	<code>max</code>	<code>yawg</code>	<code>fft10</code>	<code>temp</code>
<code>yRange</code>	<code>25thp</code>	<code>fmax</code>	<code>MAD</code>	<code>samples</code>
<code>zRange</code>	<code>75thp</code>	<code>pmax</code>	<code>MPD</code>	<code>dataErrors</code>
<code>xStd</code>	<code>autocorr</code>	<code>fmaxband</code>	<code>skew</code>	
<code>yStd</code>	<code>corrxy</code>	<code>pmaxband</code>	<code>kurt</code>	
<code>zStd</code>	<code>corrxz</code>	<code>entropy</code>	<code>avgArmAngel</code>	<code>clipsBeforeCalibr</code>
<code>xyCov</code>	<code>corryz</code>	<code>fft1</code>		
<code>xzCov</code>	<code>avgroll</code>	<code>fft2</code>	<code>avgArmAngelAbsDiff</code>	<code>clipsAfterCalibr</code>
<code>yzCov</code>	<code>avgpitch</code>	<code>fft3</code>	<code>f1</code>	<code>rawSamples</code>
<code>mean</code>	<code>avgyaw</code>	<code>fft4</code>	<code>p1</code>	<code>participant</code>

Considering the width of the epoch dataframe, I've opted not to include a screenshot of the table in this document. Instead, I'll upload it to Sharepoint and provide a link for reference as necessary.

2.1.2 Notes on processing the data

Below are a few notes/observations/comments regarding the "data processing" step using the `accelerometer` package:

Removing the annotation variable (Capture24): For this analysis and to ensure compatibility with the `accelerometer` library², the `annotation` column was excluded from each dataframe. Notably, this column was specific to the Capture24 data and absent in the UKBB raw data. The `accelerometer` library necessitates a minimum of four columns (`time`, `x`, `y`, and `z`) with an optional

²Note: The `accelerometer` package **does not work** if there are more than 4 columns; hence, we get an error code if we try to process the data with the presence of the `"annotation"` column

column for `temperature`. Hence, the annotation column was omitted to adhere to the library's requirements.

Activity classification: The `accelerometer` library provides the option to specify the activity classification model by modifying the `activityModel` command. The default activity model is `"walmsley"`. For the analysis, two graphs were plotted for each participant (see Annex 3):

- The first graph utilized the generic Walmsley model (without specifying the `"activityModel"`). There are 4 classifications (sedentary, light, moderate-vigorous and sleep).
- For the second graph, the activity classification was specified using the Willetts 2018 classification, which was also referenced in the work of Chan Chang et al. (2021). There are 6 classifications (sit-stand, mixed, bicycling, sleep, vehicle and walking).

Notes on arguments: ³

- `rawOutput` : Has to be specified as `"True"` to convert raw CWA file to CSV file.
- `deleteIntermediateFiles`: Has to be specified `"False"` in order to generate epoch files.

³Will continue to add notes

2.2 Training a bespoke model

After processing the data and generating the epoch file, we are ready to train the bespoke activity classification model using the `trainClassification` command from the `accelerometer` package. You can find more information on the command and its arguments [here](#).

2.2.1 Notes on training the bespoke classification model

A few additions and modifications have to be added to the epoch file in order to be able to use the `trainClassification` command:

Create an "annotation" column: It's necessary to add an "annotation" column to use the "trainClassification" command (for the "annotationCol" argument). In the time series table generated from the "accProcess" command, activities are represented by dummy/binary columns (e.g., light, moderate-vigorous, sedentary, sleep, etc.). To streamline the dataset for classification purposes, we need to consolidate these activities into a single column. This new "annotation" column will contain the name of the activity corresponding to each observation.

Add "participant" column: The dataframes lack a "participant" column (indicating the participant ID). Therefore, it's necessary to manually append a "participant" column for participant identification. The "participant" column is also an obligatory argument for using the "trainClassification" command (refer to the script below).

Create "full-epoch.csv" dataframe: To use the `trainClassification` command, we need to create a single comprehensive dataframe, which I refer to as the "full-epoch.csv" dataframe. This dataframe combines data from the "time-series.csv", raw CSV, and "epoch.csv" dataframes. This consolidation is crucial because the `trainClassification` command relies on columns from each of these individual dataframes (i.e. "MET" and activities). The final dataframe should have 73 columns (epoch + "participant", "annotation" and "MET" columns)

We encountered a few setbacks using this command which will be discussed below.

2.2.2 "trainClassification" command

Below is the script we used to generate the `trainClassification` command:

```
1 trainClassificationModel(  
2   '/home/yacine/accel/full_epoch_df_3.csv', # directory to the full dataset file  
3   participantCol="participant", # column with participant IDs  
4   labelCol="label", # column with labels  
5   metCol="MET", # column with MET values, if None, MET will not be used  
6   featuresTxt="/home/yacine/accel/bespoke/features.txt", # directory to the list of  
   ↪ features to use  
7   nTrees=1000, # RF parameters: number of trees in the forest  
8   maxDepth=10, # RF parameters: maximum depth of the tree  
9   minSamplesLeaf=1, # RF parameters: minimum number of samples required to be at a leaf  
   ↪ node  
10  cv=None, # whether to report cross-validation results (None, 5, 10, etc.)  
11  testParticipants="22.0,54.0", # comma separated list of participants to use for  
   ↪ testing (as given in the `participantCol` column)  
12  outDir="results/", # directory where to save results  
13  nJobs=10 # number of parallel jobs to run  
14 )
```

When using the `trainClassification` command, a few things should be taken into account:

Features document: It's crucial to generate the "features.txt" document manually as it isn't produced by any of the commands in the `accelerometer` package. An example file is available on their [GitHub page](#). You'll notice that this example file contains 111 features. We had to remove features not generated in the "epoch.csv" file.

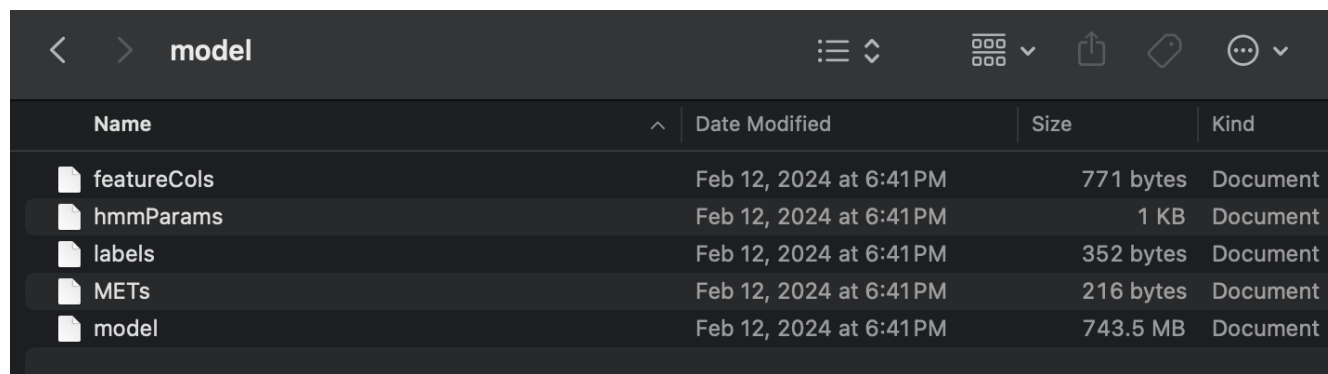
testParticipants: It is important that the participants are called by their "ID". This distinction is important because, for example, the Capture24 data utilizes IDs like "P001" to "P151" IDs, whereas the UKBB data employs numerical IDs.

2.2.3 Problems encountered when training the bespoke model using the `trainClassification` command

Upon executing this command, several error messages were encountered. Screenshots of these errors are provided in Annex 4. The primary error message indicated:

ValueError: Found array with 0 sample(s) (shape=(0, 70)) while a minimum of 1 is required.

Then, the "results/" folder contained a zip folder named "model.tar" with a ".tar" extension. However, this archive contained 5 files without extensions, rendering the results inaccessible. Below is a screenshot of the content of the "model" folder.



Name	Date Modified	Size	Kind
featureCols	Feb 12, 2024 at 6:41PM	771 bytes	Document
hmmParams	Feb 12, 2024 at 6:41PM	1 KB	Document
labels	Feb 12, 2024 at 6:41PM	352 bytes	Document
METs	Feb 12, 2024 at 6:41PM	216 bytes	Document
model	Feb 12, 2024 at 6:41PM	743.5 MB	Document

To date, despite efforts to resolve the issue by contacting the team responsible for the "accelerometer" package, we have been unable to obtain conclusive results.

3 Additional Information

3.1 Links to Biobank documentation

Category 1020: Derived Accelerometry

Description: Derived accelerometry

<https://biobank.ndph.ox.ac.uk/ukb/label.cgi?id=1020>

Data-Field 90001: Acceleration Data - CWA Format

Description: Acceleration data - cwa format

Category: Additional exposures >Physical activity measurement

Accelerometer data

<https://biobank.ndph.ox.ac.uk/ukb/field.cgi?id=90001>

Data-Field 90004: Acceleration Intensity Time-Series

Description: Acceleration intensity time-series

Category: Additional exposures >Physical activity measurement

Accelerometer data

<https://biobank.ndph.ox.ac.uk/ukb/field.cgi?id=90004>

Resource 90216: Accelerometer Processing Source Code

Name: Accelerometer Processing Source Code

Size: 756,756

MD5: d396ea887319862550897adf9299c828

Source codes for accelerometer lib

<https://biobank.ndph.ox.ac.uk/ukb/refer.cgi?id=90216>

Category 1008: Physical Activity Measurement

Description: Physical activity measurement recorded via a wrist-worn accelerometer.

<https://biobank.ndph.ox.ac.uk/ukb/label.cgi?id=1008>

Annex 1 - Capture24 Data Sample

Capture23 csv data

Table 1: Capture24 Raw CSV Data - P001

	time	x	y	z	annotation
0	2016-11-13 02:18:00.000000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
1	2016-11-13 02:18:00.010000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
2	2016-11-13 02:18:00.020000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
3	2016-11-13 02:18:00.030000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
4	2016-11-13 02:18:00.040000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
5	2016-11-13 02:18:00.050000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
6	2016-11-13 02:18:00.060000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
7	2016-11-13 02:18:00.070000	-0.46669	-0.548902	0.658472	7030 sleeping;MET 0.95
8	2016-11-13 02:18:00.080000	-0.46669	-0.533341	0.658472	7030 sleeping;MET 0.95
9	2016-11-13 02:18:00.090000	-0.46669	-0.548902	0.658472	7030 sleeping;MET 0.95

Annex 2 - UKBB Data Sample

UKBB Raw data converted to CSV

Table 2: UKBB Raw CWA data converted to CSV - Participant 1000032

	time	x	y	z
0	2013-11-28 10:00:04.841+0000 [Europe/London]	0.227	-0.844	1.025
1	2013-11-28 10:00:04.851+0000 [Europe/London]	0.120	0.826	0.478
2	2013-11-28 10:00:04.861+0000 [Europe/London]	0.120	0.872	0.494
3	2013-11-28 10:00:04.871+0000 [Europe/London]	0.120	0.872	0.494
4	2013-11-28 10:00:04.881+0000 [Europe/London]	0.120	0.872	0.478
5	2013-11-28 10:00:04.891+0000 [Europe/London]	0.135	0.888	0.510
6	2013-11-28 10:00:04.901+0000 [Europe/London]	0.135	0.888	0.510
7	2013-11-28 10:00:04.911+0000 [Europe/London]	0.135	0.857	0.510
8	2013-11-28 10:00:04.921+0000 [Europe/London]	0.120	0.826	0.510
9	2013-11-28 10:00:04.931+0000 [Europe/London]	0.104	0.842	0.510

4 Annex 3 - Accelerometer Graphs

4.1 Capture-24 Data

4.1.1 P001

The data for participant no.1 (P001) was collected through Sunday November 13th, 2016 to November 14th, 2016. We notice there is missing data from approx. 4:00 a.m. to 5:00 a.m.

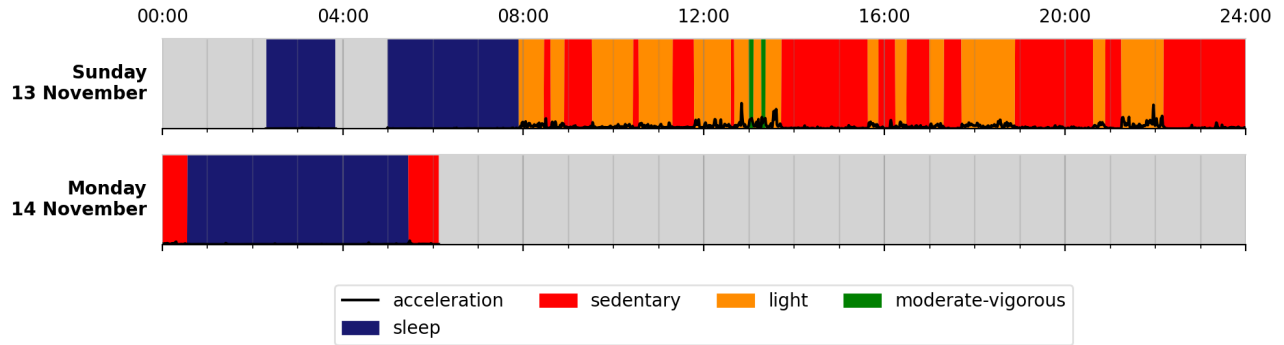


Figure 1: *Graph for participant P001 using the generic Walmsley activity model.*

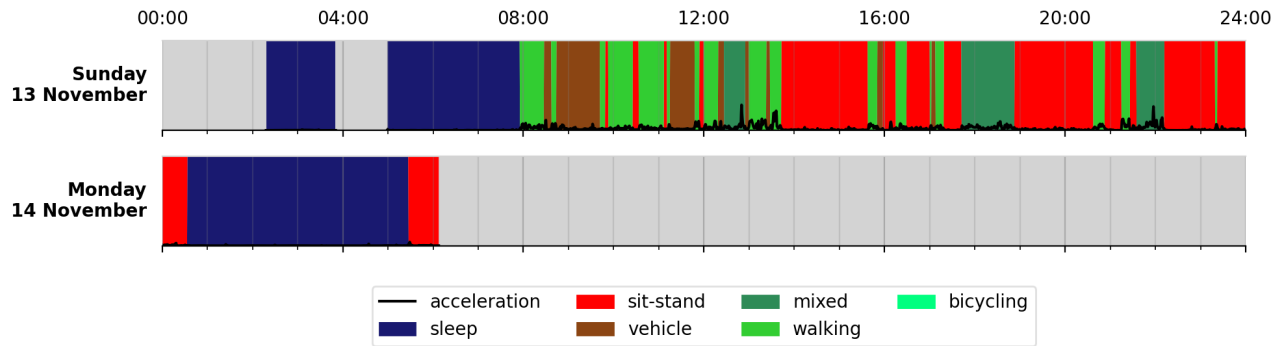


Figure 2: *Graph for participant P001 using the generic Willetts (2018) activity model.*

4.1.2 P002

The data for participant no.2 (P002) was collected through Saturday May 21st, 2016 to Sunday May 22nd, 2016.

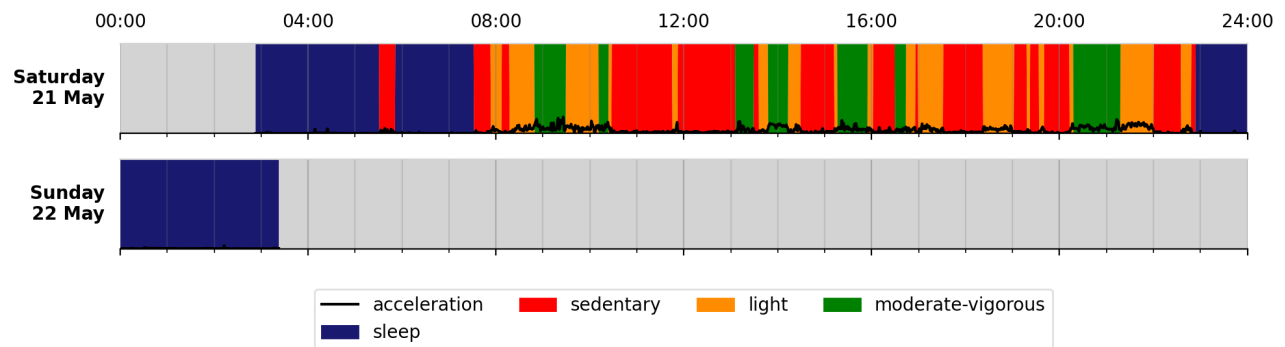


Figure 3: Graph for participant P002 using the generic Walmsley activity model.

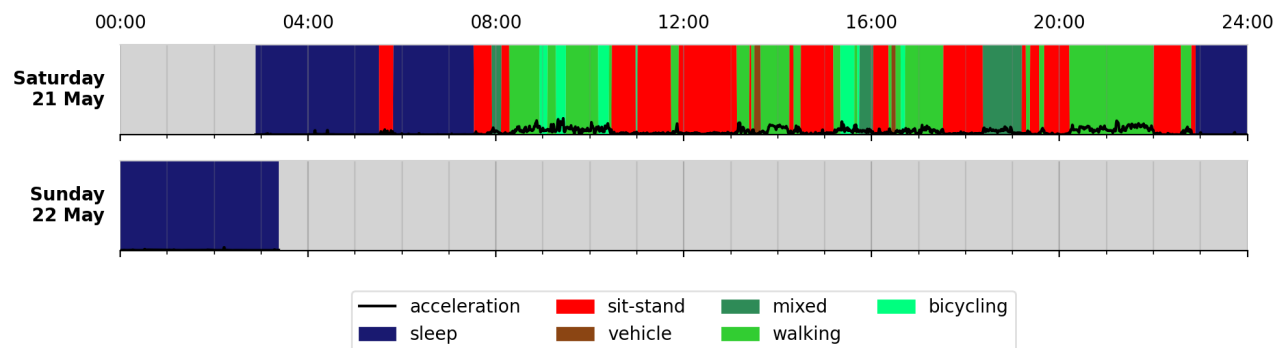


Figure 4: Graph for participant P002 using the generic Willetts (2018) activity model.

4.2 P003

The data for participant no.2 (P002) was collected through Wednesday January 13th, 2016 to Thursday January 14th, 2016.

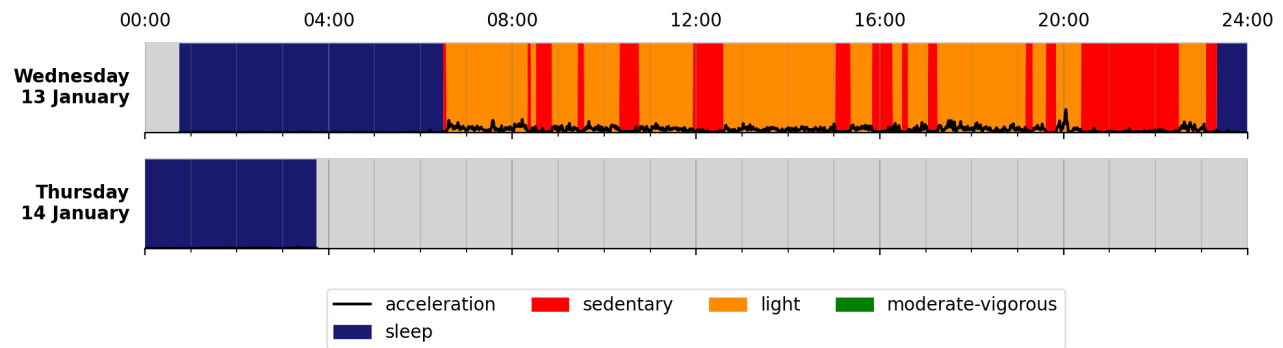


Figure 5: Graph for participant P003 using the generic Walmsley activity model.

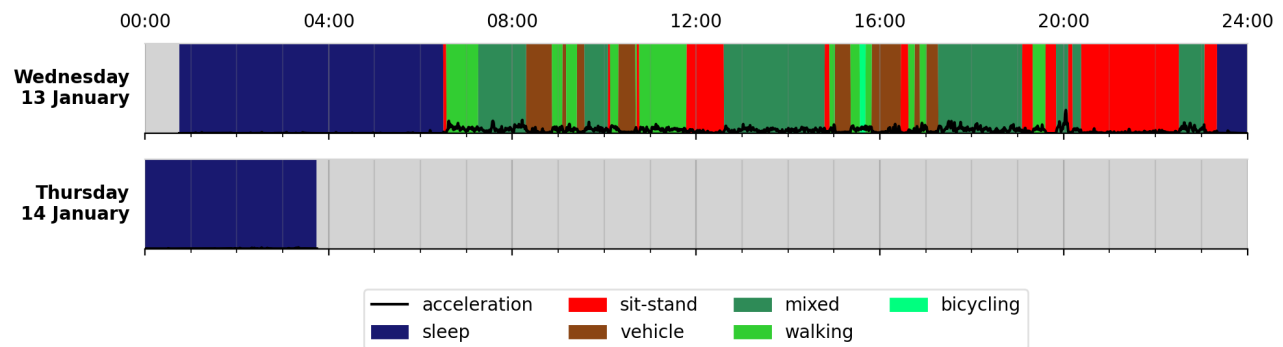


Figure 6: Graph for participant P003 using the generic Willetts (2018) activity model.

4.3 P004

The data for participant no.2 (P002) was collected through Tuesday November 22nd, 2016 to Wednesday November 23rd, 2016.

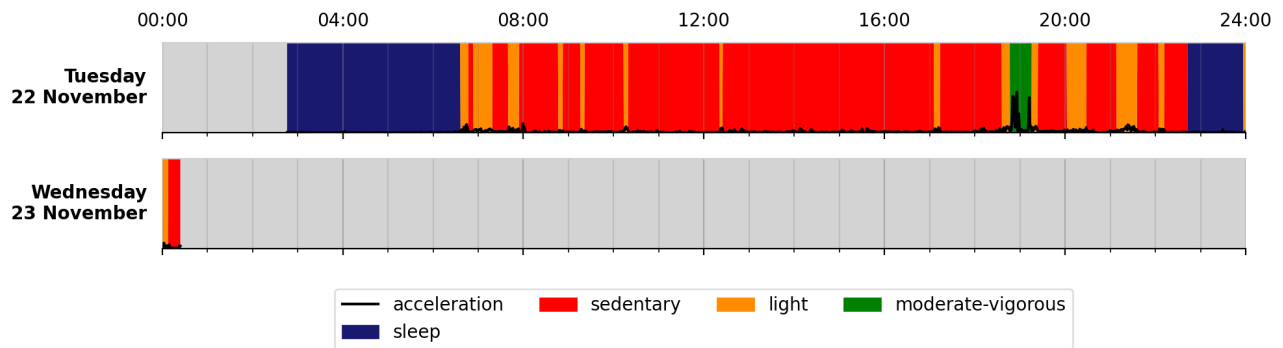


Figure 7: Graph for participant P004 using the generic Walmsley activity model.

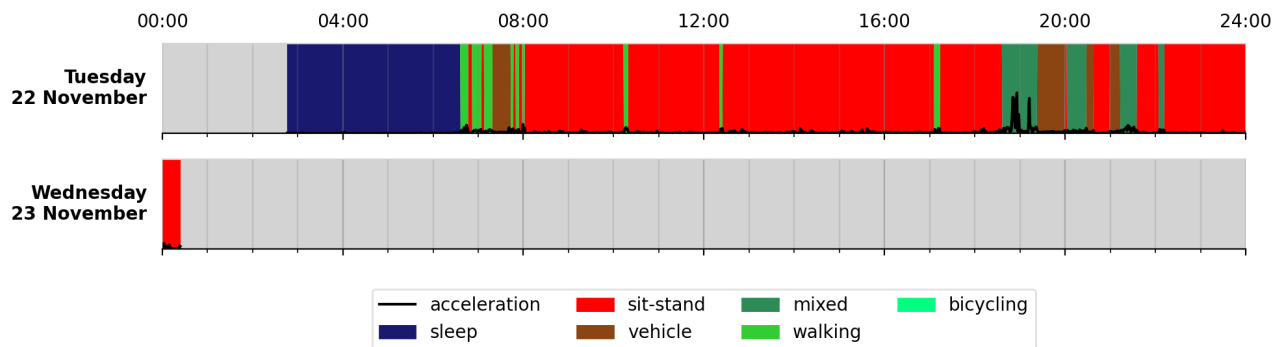


Figure 8: Graph for participant P004 using the generic Willetts (2018) activity model.

4.4 UKBB Data Graphs

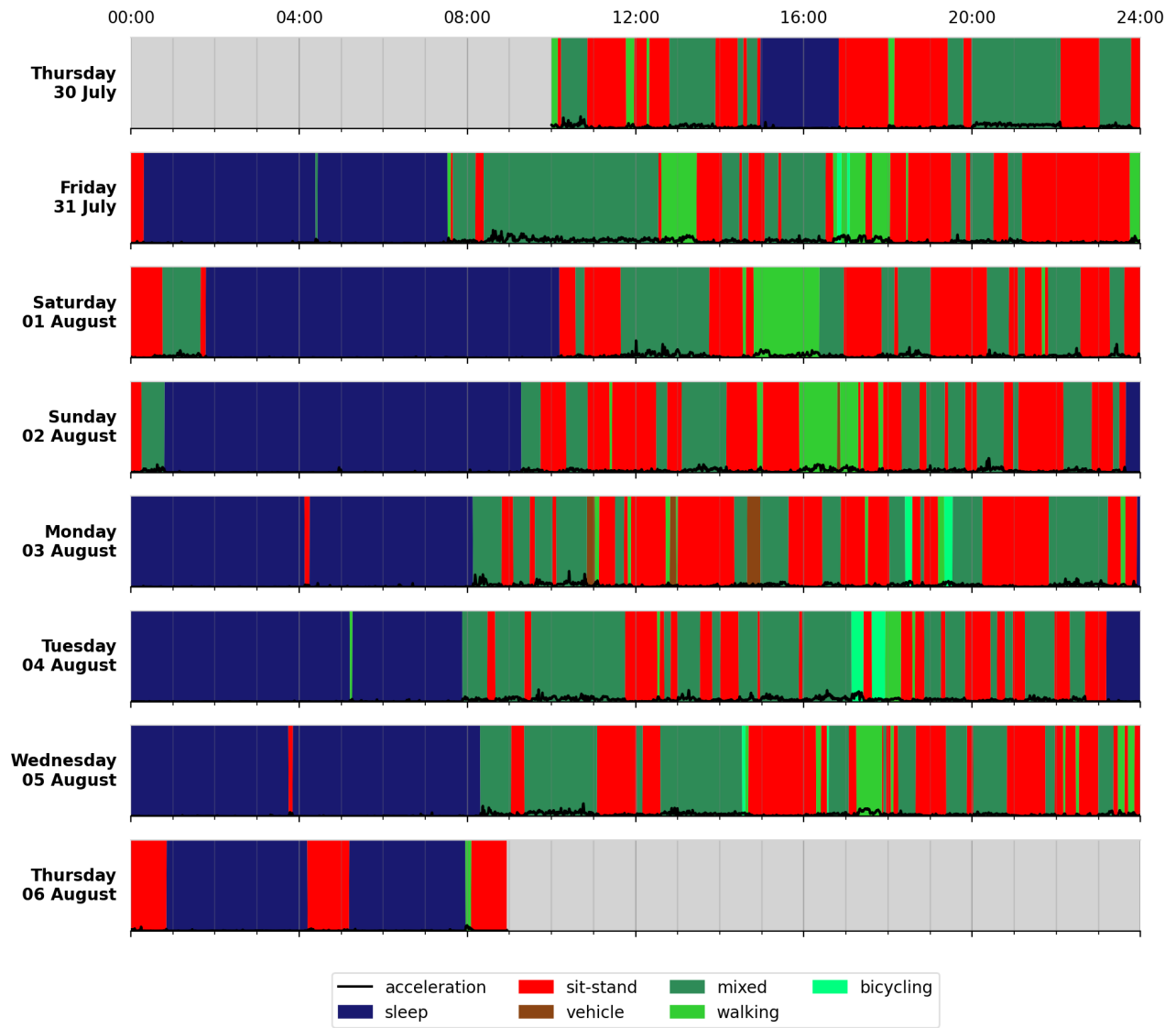


Figure 9: *Graph for participant #1000168*

Annex 4 - Screenshots of errors encountered using the trainClassification command

```
Training...
[Parallel(n_jobs=10)]: Using backend ThreadingBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done 30 tasks | elapsed: 3.0s
[Parallel(n_jobs=10)]: Done 180 tasks | elapsed: 15.2s
[Parallel(n_jobs=10)]: Done 430 tasks | elapsed: 34.8s
[Parallel(n_jobs=10)]: Done 780 tasks | elapsed: 1.0min
[Parallel(n_jobs=10)]: Done 1000 out of 1000 | elapsed: 1.3min finished
Cross-predicting to derive the observations for HMM...
[Parallel(n_jobs=5)]: Using backend LokyBackend with 5 concurrent workers.
[Parallel(n_jobs=5)]: Done 5 out of 10 | elapsed: 2.4min remaining: 2.4min
[Parallel(n_jobs=5)]: Done 10 out of 10 | elapsed: 4.7min finished
Training HMM...
Models saved to results/model.tar
Output trained model written to: results/model.tar
Testing on participant(s): 22.0,54.0
```

Figure 10: Screenshot of the output while running the code.

```
ValueError                                Traceback (most recent call last)
Cell In[15], line 3
      1 # script provided by Shing
----> 3 trainClassificationModel(
      4     '/home/yacine/accel/full_epoch_df_3.csv', # path to the dataset file
      5     participantCol="participant", # column with participant IDs
      6     labelCol="label", # column with labels
      7     metCol="MET", # column with MET values, if None, MET will not be used
      8     featuresTxt="/home/yacine/accel/bespoke/features.txt", # file with list of features to use
      9     nTrees=1000, # random forest parameters: number of trees in the forest
     10     maxDepth=10, # random forest parameters: maximum depth of the tree
     11     minSamplesLeaf=1, # random forest parameters: minimum number of samples required to be at a leaf node
     12     cv=None, # whether to report cross-validation results (None, 5, 10, etc.)
     13     testParticipants="22.0,54.0", # comma separated list of participants to use for testing (as given in the 'particip
antCol' column)
     14     outDir="results/", # directory where to save results
     15     nJobs=10 # number of parallel jobs to run
     16 )

File ~/local/lib/python3.10/site-packages/accelerometer/classification.py:215, in trainClassificationModel(trainingFile, labelCol, participantCol, annotationCol, metCol, featuresTxt, nTrees, maxDepth, minSamplesLeaf, cv, testParticipants, outDir, nJobs)
    213 print('Testing on participant(s):', testParticipants)
    214 Xtest, Ytest = test[featureCols].to_numpy(), test[labelCol].to_numpy()
--> 215 Ypred = model.predict(Xtest)
    216 YpredHmm = viterbi(Ypred, hmmParams)
    217 test['predicted'] = YpredHmm

File ~/local/lib/python3.10/site-packages/sklearn/ensemble/_forest.py:808, in ForestClassifier.predict(self, X)
    787 def predict(self, X):
    788     """
    789     Predict class for X.
    790
    (...)
    806     The predicted classes.
    807     """
--> 808     proba = self.predict_proba(X)
    809     if self.n_outputs_ == 1:
    811         return self.classes_.take(np.argmax(proba, axis=1), axis=0)
```

Figure 11: First part of the error message.

```

File ~/.local/lib/python3.10/site-packages/sklearn/ensemble/_forest.py:850, in ForestClassifier.predict_proba(self, X)
    848 check_is_fitted(self)
    849 # Check data
--> 850 X = self._validate_X_predict(X)
    851 # Assign chunk of trees to jobs
    852 n_jobs, _, _ = _partition_estimators(self.n_estimators, self.n_jobs)

File ~/.local/lib/python3.10/site-packages/sklearn/ensemble/_forest.py:579, in BaseForest._validate_X_predict(self, X)
    578 """
    579 Validate X whenever one tries to predict, apply, predict_proba."""
    580 check_is_fitted(self)
--> 579 X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr", reset=False)
    580 if issparse(X) and (X.indices.dtype != np.intc or X.indptr.dtype != np.intc):
    581     raise ValueError("No support for np.int64 index based sparse matrices")

File ~/.local/lib/python3.10/site-packages/sklearn/base.py:566, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_params)
    564     raise ValueError("Validation should be done on X, y or both.")
    565 elif not no_val_X and no_val_y:
--> 566     X = check_array(X, **check_params)
    567     out = X
    568 elif no_val_X and not no_val_y:

File ~/.local/lib/python3.10/site-packages/sklearn/utils/validation.py:805, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    803 n_samples = _num_samples(array)
    804 if n_samples < ensure_min_samples:
--> 805     raise ValueError(
    806         "Found array with %d sample(s) (shape=%s) while a"
    807         " minimum of %d is required%s."
    808         % (n_samples, array.shape, ensure_min_samples, context)
    809     )
    811 if ensure_min_features > 0 and array.ndim == 2:
    812     n_features = array.shape[1]

ValueError: Found array with 0 sample(s) (shape=(0, 70)) while a minimum of 1 is required.

```

Figure 12: Second part of the error message.