Ikäheimonen, Arsi; Triana, Ana M.; Luong, Nguyen; Ziaei, Amirmohammad; Rantaharju, Jarno; Darst, Richard; Aledavood, Talayeh

Niimpy : A toolbox for behavioral data analysis

Published in:
SoftwareX

DOI:
10.1016/j.softx.2023.101472

Published: 01/07/2023

Document Version
Publisher's PDF, also known as Version of record

SoftwareX 23 (2023) 101472

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Niimpy: A toolbox for behavioral data analysis

Arsi Ikäheimonen *, Ana M. Triana, Nguyen Luong, Amirmohammad Ziaei, Jarno Rantaharju, Richard Darst, Talayeh Aledavood

Department of Computer Science, Aalto university, Tietotekniikan laitos, P.O. Box 15400, FI-00076 AALTO, Finland

ARTICLE INFO

Article history:
Received 26 September 2022
Received in revised form 5 June 2023
Accepted 6 July 2023

Dataset link: https://studentlife.cs.dartmouth.edu/dataset.html

Keywords:
Data analysis toolbox
Digital behavioral studies
Mobile sensing
Python package

ABSTRACT

Behavioral studies using personal digital devices typically produce rich longitudinal datasets of mixed data types. These data provide information about the behavior of users of these devices in real-time and in the users' natural environments. Analyzing the data requires multidisciplinary expertise and dedicated software. Currently, no generalizable, device-agnostic, freely-available software exists within Python scientific computing ecosystem to preprocess and analyze such data. This paper introduces a Python package, *Niimpy*, for analyzing digital behavioral data. The Niimpy toolbox is a user-friendly open-source package that can quickly be expanded and adapted to specific research requirements. The toolbox facilitates the analysis phase by offering tools for preprocessing, extracting features, and exploring the data. It also aims to educate the user on behavioral data analysis and promotes open science practices. Over time, Niimpy will expand with new data analysis features developed by the core group, new users, and developers. Niimpy can help the fast-growing number of researchers with diverse backgrounds who collect data from personal and consumer digital devices to systematically and efficiently analyze the data and extract useful information. This novel information is vital for answering research questions in various fields, from medicine to psychology, sociology, and others.

Code metadata

Current code version	1.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00304
Permanent link to Reproducible Capsule	–
Legal Code License	MIT license
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	https://github.com/digitraceslab/niimpy/blob/master/requirements-dev.txt
If available Link to developer documentation/manual	https://niimpy.readthedocs.io/en/latest/
Support email for questions	talayeh.aledavood@aalto.fi

Software metadata

Current software version	1.1
Permanent link to executables of this version	https://github.com/digitraceslab/niimpy/releases/tag/v1.1
Permanent link to Reproducible Capsule	–
Legal Software License	MIT License
Computing platforms/Operating Systems	Any capable of running Python
Installation requirements & dependencies	Python and SciPy stack packages
If available, link to user manual - if formally published, include a reference to the publication in the reference list	https://niimpy.readthedocs.io/
Support email for questions	talayeh.aledavood@aalto.fi

1. Motivation and significance

Digital behavioral studies aim to quantify human behavior continuously in natural living environments using data from personal digital devices (e.g., smartphones and fitness trackers) and

* Corresponding author.
E-mail address: arsi.ikaheimonen@aalto.fi (A. Ikäheimonen).

https://doi.org/10.1016/j.softx.2023.101472

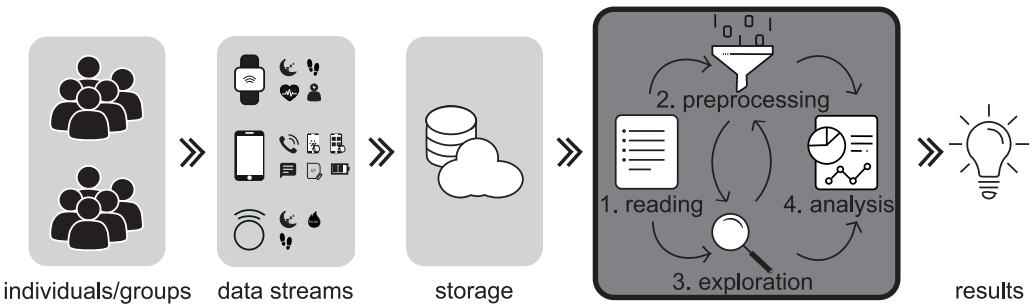

Fig. 1. Schematic of a typical digital behavioral study workflow. Generally, individuals or groups volunteer to gather data. The data streams are collected via devices that use different sensors. The data streams are then stored locally or in the cloud. Next, the data is ready to be analyzed. Niimpy provides tools for this analysis phase via four layers (highlighted in dark gray). Users can (i) read the data and then opt to (ii) preprocess or (iii) explore them. While in these layers, users can switch from preprocessing to exploration as needed, and once they are ready, start the (iv) analysis to obtain the requested results.

online social media platforms the user interacts with [1–4]. Recently, there has been an increasing scientific interest in digital behavioral studies for unobtrusive human behavior monitoring [5–8].

Typically, digital behavioral studies produce large, heterogeneous, rich data sets of mixed data types. Multiple stages are required to go from the data produced within such studies to meaningful and useful information. A typical digital behavioral study analysis workflow consists of data collection, storage, and analysis phases. Fig. 1 expands on these details. Data preprocessing and feature extraction are crucial tasks for analyzing the data, yet they often have to be re-implemented for each study. Recycling the implemented tools is challenging due to differences in data types and structures across projects. The lack of established data analysis methods and reusable open-source software form significant barriers to new research [9–11]. Additionally, the lack of reusable methods leads to study results that are not necessarily reproducible and comparable.

The existing software solutions for digital behavioral studies can be categorized by the functionality into three categories: 1) data collection platforms (such as AWARE [12], BEIWE [13], and CARP Mobile Sensing [14]), 2) data analysis frameworks (e.g., The Digital Biomarker Discovery Pipeline (DBDP) [10], Forest [15], and Rapids [9]), and 3) platforms dedicated for study participants and clinicians (e.g., mindLAMP Dashboard [16]). Some solutions focus on certain functionality, while some encompass all three (e.g., HOPES [17], LAMP platform [16], and RADAR-base [18]).

Among the data analysis frameworks, existing software comes with various limitations for behavioral data analysis; the software may not contain a complete suite for data preprocessing and analysis, may be focused on physiological or biomedical data analysis, the code may not be organized coherently for a reproducible analysis pipeline, may be outdated, not actively developed or maintained, tied to some specific data collection platform or device, may not be openly available for researchers, or the software adaptation has a high entry barrier.

The most comprehensive and mature analysis frameworks include DBDP, Forest, Rapids, and Radar-base. These open-source software feature digital behavioral data analysis and are implemented wholly or partly in Python.

DBDP is a platform dedicated to transforming mobile health data into digital biomarkers. The code is organized in modules that are written in R and Python. DBDP provides modules for data aggregation, preprocessing, exploration, and analysis. DBDP supports various wearable devices and data formats, including physiological and behavioral data.

Forest is a library to analyze smartphone-based high-throughput digital phenotyping data. The library is written in Python. It is integrated with the BEIWE back-end and can also be run locally. The system also implements an API for Tableau [19], supporting dashboard visualizations.

Rapids is a framework offering an open-source code for preprocessing, extracting, and visualizing behavioral features from data. It is written in Python and R and supports mobile phones and wearable devices. The code is executed by Snakemake workflow manager [20] and is organized by cookie cutter data science [21] project structure.

Radar-base is a data collection platform built around Confluent/Apache Kafka. It features data preprocessing capabilities and visualization dashboards implemented in Python. Data preprocessing enables compatibility with standard machine learning and data science libraries.

To offer an easily deployable, lightweight analysis framework alternative for researchers already familiar with Python programming, we propose Niimpy: a toolbox for behavioral data analysis. Niimpy is a user-friendly, open-source Python package. It can quickly expand and adapt to specific research questions and workflows and integrates seamlessly into the existing Python scientific computing ecosystem [22]. Niimpy targets a group of users who already use Python for their data analysis needs and are possibly already familiar with other scientific computing libraries in Python. With a minimal barrier to entry, these users can use Niimpy for projects in which sensor data (e.g., from smartphones) from study participants are collected, and the data require preprocessing, exploration, and feature extraction. In the future, later versions of Niimpy will allow them to perform a certain level of analysis on top of these as well. Niimpy is accompanied by comprehensive documentation and examples using real data, facilitating the implementation of the toolbox. The toolbox is not an out-of-the-box software solution but offers a base framework requiring some programming knowledge to use it.

2. Software description

The Niimpy toolbox is a Python package for rich multi-sensor longitudinal behavioral data analysis. Niimpy can preprocess raw sensor data (e.g., GPS coordinates) or work on predefined data summaries (e.g., daily step count). It is designed for small to moderate-sized (order of thousands of participants) studies. Niimpy is built around Pandas [23] and other scientific Python stack libraries. The toolbox requires basic Python programming knowledge. Niimpy comes with comprehensive example notebooks which serve as boilerplate templates for users.

2.1. Design philosophy

Niimpy provides basic behavioral data analysis operations and is a starting point for implementing new analyses. As a single software tool cannot provide everything required for every type

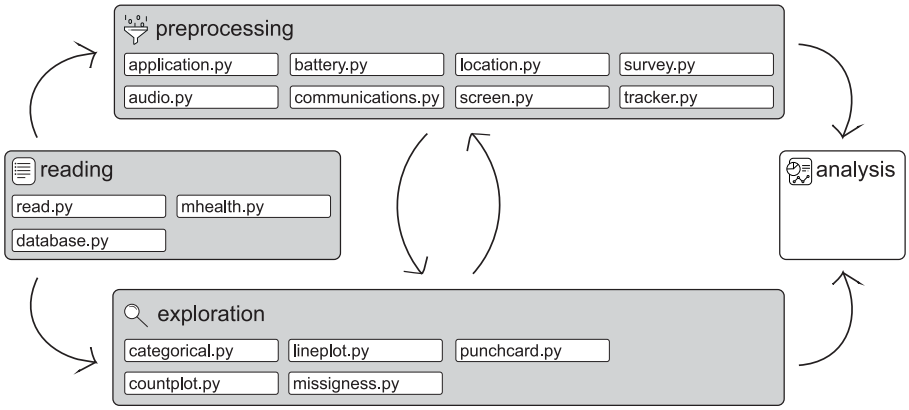

Fig. 2. Block diagram of the niimpy library. Currently, niimpy has several Python functions that can i) read different data formats, ii) preprocess data from eight different sources, and (iii) explore the data using different plots and tools. In the future, Niimpy will also implement functions for data analysis.

of analysis, Niimpy makes it easy to customize analyses while building on existing work.

Niimpy introduces a standard data schema using Pandas DataFrames. The schema facilitates generalizability, guides data structures, and promotes overall reusability. Data processing tools and functions following this schema can be directly incorporated into a Niimpy analysis pipeline. If these add-on analysis functions are generalizable and reusable, we encourage the user to incorporate those into Niimpy. Using existing functionality in Pandas and other Python scientific libraries in the pipeline is straightforward.

Finally, Niimpy uses existing tools as much as possible. The scientific Python ecosystem provides a wide variety of data processing tools, which are used directly. For example, a significant part of data preprocessing can be done using standard Pandas operations, and shortcut functions are not created for this. Instead, shortcuts are created when they can significantly reduce the user's cognitive load or increase the code's readability. Reusing these standard components allows others to begin using Niimpy more quickly and apply the skills learned in Niimpy to other projects.

2.2. Software architecture

The Niimpy software architecture divides into distinct functional layers; 1) reading, 2) preprocessing, 3) exploration, and 4) analysis. Fig. 2 expands on these details, and a succinct summary of the functionalities of each layer is provided in Table 1.

2.2.1. Data reading

The reading layer imports the data from files or other sources, converting the input data to Pandas dataframes with the standardized data schema and doing some minimal data type standardization. Niimpy provides importer functions for CSV and sqlite3 databases, and the current development version supports open mHealth [24] datatypes. However, in many cases, the user loads and converts data into the required dataframe format, predefined by the schema. The schema expects data to be in a tabular (relational) format where a row represents an observation, and columns are properties of observations. This layer is not concerned with the type of sensor or sensor-specific data formats.

2.2.2. Data preprocessing

The preprocessing layer is for data cleaning, filtering, transformation, encoding, and feature extraction. The main focus is on feature extraction functions, while we recommend using existing Python functions for preprocessing when possible. Some preprocessing functions are sensor or device-specific (e.g., Polar tracker feature extraction functions), while some apply generally (e.g., location data feature extraction). Preprocessing functions take in Pandas dataframes and return dataframes and may also require specified column names. Niimpy provides a set of ready-made features for each sensor. Users can extract all the features by default or select the desired ones. Furthermore, users may implement their custom preprocessing functions.

Currently supported sensors and data streams include the Polar fitness tracker, Android and iOS mobile phone sensors (application data, audio, battery, communication, screen), GPS location, and survey streams. For details, refer to appendix Table 2.

2.2.3. Data exploration

The exploration layer produces visual data summaries and assesses the data quality (e.g., missing data, outliers). The module includes functions for plotting categorical data counts and distributions, individual and group-wise observation counts, time series line plots for visualizing trends, cyclicity, and anomalies, punchcard charts for comprehensive surveys, and visualizing missing data. All the functions are implemented using Plotly Python Open Source Library [25]. Plotly enables interactive inspection of different aspects of the data (e.g., specific time range). For more details about exploration module functions, refer to appendix Table 3.

2.2.4. Data analysis

The analysis layer has functions for modeling data and statistical inference. The models and inference are applied to the data to identify relationships (e.g., correlation and association) among the features. This layer will be continuously expanded in the future.

2.3. Unit tests

A set of unit tests accompanies each Niimpy module to ensure it works as intended. The general guideline is to have unit tests for each function with which the toolbox users interact. The toolbox comes with synthetically created and openly available sample data sets for unit tests. More details about the sample data are found on the GitHub repository [26]. Currently, the code test coverage is above 84%.

2.4. Expanding the toolbox

Niimpy can serve as a base for developing new analysis methods by leveraging the lower functional layers. The toolbox is intended to be expanded according to different research needs. We encourage users to contribute reusable and well-generalizable functions, which are accompanied by test functions to the toolbox. The toolbox structure is fully modular; therefore, adding a

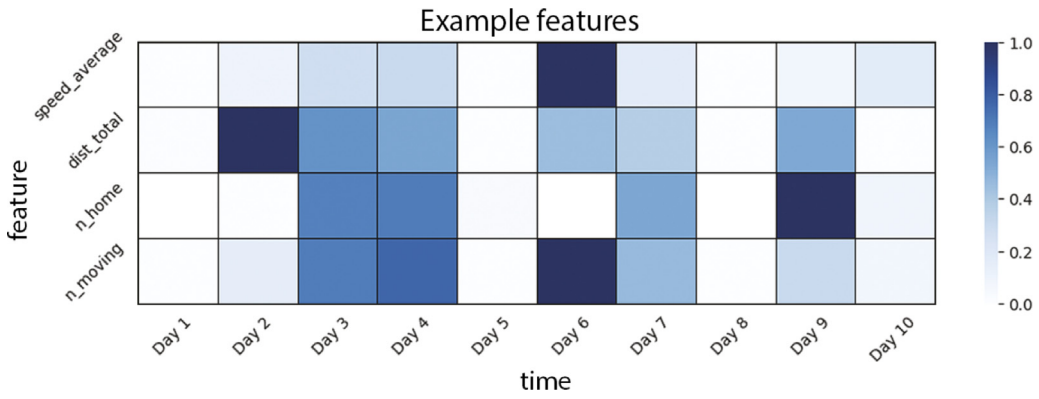

Fig. 3. Heatmap visualization presenting significant places-based and movement-based features for ten days, highlighting movement patterns, including those days in which the participant has traveled longer distances, moved faster or has stayed at home more time. Here the y-label *n_home* represents the time spent at home, *n_moving* the time spent moving from place to place, *speed_average* the average speed, and *dist_total* the total distance traveled. The figure was created using the Seaborn data visualization library [27].

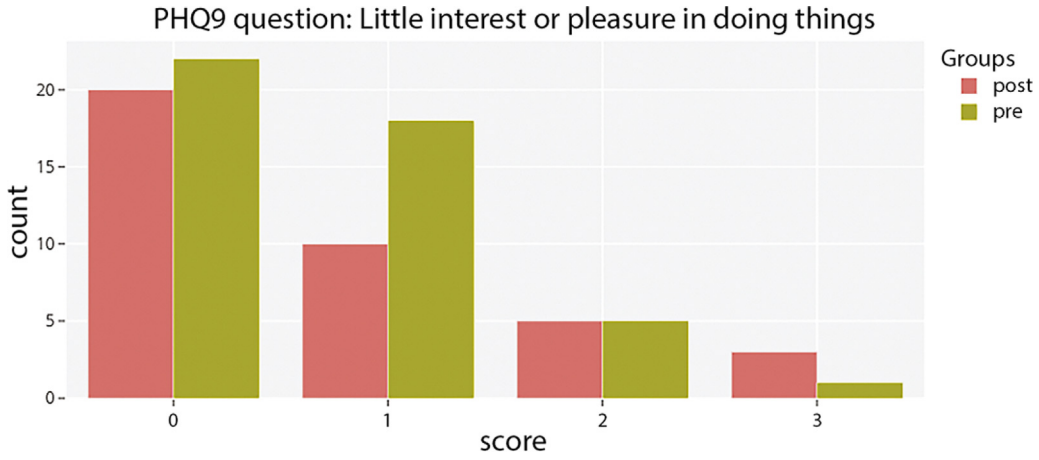

Fig. 4. Barplot presenting of pre- and post-study PHQ9-1 question score distributions. The visualization reveals minor differences between pre- and post-study survey score distributions.

new feature is straightforward. New features should be added via GitHub pull requests, following the instructions provided in the project code repository. The instructions include required data schema, basic design guidelines, unit test requirements, instructions for documentation and demonstrative notebooks, and instructions for adding example datasets. Inputs and outputs are Pandas dataframes that follow standard formatting so that Niimpy fits into the Python scientific stack, allowing the users to use Niimpy in their custom data processing workflows.

3. Illustrative examples

This section provides illustrative examples of toolbox usage. These examples elucidate how to use functions from different toolbox layers to read the data from various sources, transform it into Niimpy-compatible format, extract features, and visualize them. These features carry useful information for analyzing people's behavioral patterns and changes in their behavior. For further examples, refer to Niimpy's documentation [26]. The first example, code listing 1, offers a snapshot of location features extraction from GPS data from a single participant. The data was collected via an Android phone using the AWARE framework [12]. Initially, the data is loaded (line 5), followed by a noise reduction step through data downsampling (lines 8–11). Finally, significant places-based and movement-based features are extracted (line 14). Selected resultant features are displayed in a heatmap, as seen in Fig. 3.

Code listing 2 demonstrates a second example with survey data from the StudentLife dataset [28], specifically featuring PHQ-9 questionnaire [29] responses. The Kaggle API facilitates access to the data [30] (lines 8–10), followed by data download and questionnaire data extraction (lines 13–16). Next, the code transforms the data into Niimpy-compatible format by renaming columns (lines 19–25). Then the code restructures the data for better readability and analysis: shorter PHQ9 question numbers replace the column (lines 28–29), the data format transforms from wide to long (lines 32–33), and numerical format answers replace the original questionnaire responses (lines 36–38). Visualizing this processed data (lines 41–50) provides insight into pre- and post-study score distributions, as demonstrated in Fig. 4.

Finally, the third code listing 3 provides an example of extracting features from synthetic activity tracker data, modeled after the Polar activity tracker data schema [31]. The process begins with loading the data (line 7) and converting the index into the Pandas DateTime format (line 8). Then the code extract features (lines 11–12) and generate visualizations (lines 15–16). Fig. 5 offers a glance at the step count data of a single subject over three days.

```
1 import niimpy
2 import niimpy.preprocessing.location as
      nilo
3
4 # Load location sample data
```

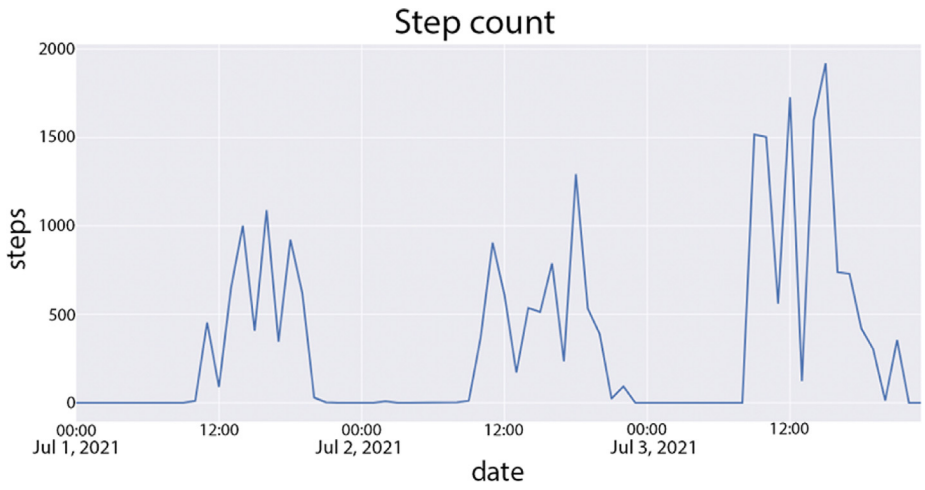

Fig. 5. Lineplot visualization presenting a single subject's hourly aggregated step count patterns for three consecutive days. Inactivity times can be directly seen from this data, as well as variation in more active days.

```
data = niimpy.read_csv(niimpy.sampledata.LOCATION_FILE,tz='et')

# Reduce noise by downsampling the data using the median
binned_data = niimpy.util.aggregate(data, freq='5T',

    method_numerical='median')
# Reset index and drop rows with missing values
binned_data = binned_data.reset_index(0).dropna()

# Extract the features
all_features = nilo.extract_features_location(binned_data)
```

Listing 1: Illustrative code example to extract features from GPS-location data.

```
import zipfile
import pandas as pd
from niimpy.preprocessing import survey
from niimpy.exploration.eda import categorical
from kaggle.api.kaggle_api_extended import KaggleApi

# Authenticate and download dataset using Kaggle API
api = KaggleApi()
api.authenticate()
api.dataset_download_files('dartweichen/student-life', path=".")

# Extract the specific file from the downloaded zip
archive = zipfile.ZipFile('student-life.zip', 'r')
with archive.open(f"dataset/survey/PHQ-9.csv") as survey_file:
    # Read the csv file into a pandas DataFrame
    survey_data = pd.read_csv(survey_file)

# Rename question columns by adding a '.' to the end
new_columns = [b + '.' if not b.endswith('.') else b for b in survey_data.columns[2:11]]
old_columns = survey_data.columns[2:11] # old column names
mapping = dict(zip(old_columns, new_columns)) # create a dictionary
survey_data.rename(columns=mapping, inplace=True) # rename the columns

# Rename the 'uid' column to 'user'
survey_data.rename(columns={"uid": "user",},inplace=True)

# Remap column names to shorter identifiers based on Niimpy provided mappings
col_id = {**survey.PHQ9_MAP}
selected_cols = [col for col in survey_data.columns if col in col_id.keys()]

# Reshape data to long format and encode responses numerically
transformed_df = pd.melt(survey_data, id_vars=['user', 'type'], value_vars=selected_cols, var_name='question', value_name='raw_answer')
transformed_df['id'] = transformed_df['question'].replace(survey.PHQ9_MAP)

# Convert survey answers to numerical format
transformed_df['answer'] = survey.survey_convert_to_numerical_answer(
    transformed_df, answer_col = 'raw_answer', question_id = 'id',
    id_map={"PHQ9": survey.PHQ9_ANSWER_MAP}, use_prefix=True)

# Visualize distribution of responses to the first question
fig = categorical.questionnaire_grouped_summary(
    transformed_df,
    question='PHQ9_1',
    group='type',
```

```
    title='PHQ9 question: Little interest or pleasure in doing things',
    xlabel='score',
    ylabel='count',
    width=800,
    height=400)
fig.show()
```

Listing 2: Illustrative code example for acquiring data using Kaggle API, preprocessing, and visualizing the data

```
import pandas as pd
import niimpy.preprocessing.tracker as tracker
import niimpy.exploration.eda.lineplot as lineplot
from niimpy import config

# Load tracker data and convert index to datetime
data = pd.read_csv(config.STEP_SUMMARY_PATH, index_col=0)
data.index = pd.to_datetime(data.index)

# Extract daily step distribution features
f = tracker.tracker_daily_step_distribution
step_distribution = tracker.extract_features_tracker(data, features={f: {}})

# Visualize hourly-aggregated step count for a single user
lineplot.timeplot(data, users=["wiam9xme"], columns=["steps"],
title="Step count", xlabel="date", ylabel="steps", resample='H')
```

Listing 3: Illustrative code example to extract and visualize activity tracker data.

4. Impact

Quantifying digital behavioral data yields information about the study participants' behavioral patterns, changes in patterns, and differences between groups. This information is beneficial for predicting future changes in a person's well-being or clinical conditions. It may also yield new insights into theoretical models of human behavior. However, running similar studies in different places and populations is key to producing reliable and validated results. The studies need to be reproducible in terms of study protocols and data analysis. Niimpy can facilitate this by making the data analysis workflow consistent from one study to another and making the data analysis more accessible for a wider group of researchers. The resulting information can be used for well-being applications encouraging improved health behavior or may help develop novel, efficient healthcare solutions.

The Niimpy toolbox helps researchers to analyze digital behavioral data. For this data, preprocessing and feature extraction are critical tasks and have the highest barrier to entry in the analysis. Thus, preprocessing functionalities are the most prominent feature of the toolbox, setting it apart from other data analysis software tools. Niimpy provides tools and comprehensive examples of how to conduct the analysis.

Niimpy's development was motivated by the needs of the Mobile Monitoring of Mood (MoMo-Mood) pilot [32] and the main study [33], which collected various types of data from different devices and different groups of patients with mental disorders as well as a control group. The toolbox is actively used for these and other similar studies (e.g., the cor:ona study [8]) and is consciously extended based on the needs of different projects. In the future, we will continue to develop and add new features to the toolbox actively. While analyzing data sets acquired from existing and new digital behavior studies, we will incorporate a layer for analysis functions into the toolbox. Further, the toolbox will be used for educational purposes in courses and workshops covering the topic of *digital health and human behavior* at Aalto University, Finland, and possibly internationally in the future.

5. Conclusion

We have released a Python package, Niimpy toolbox, for digital behavioral data analysis. The toolbox is intended for data scientists and provides data loading, preprocessing, feature extraction, and visualization tools. Niimpy includes comprehensive documentation and examples covering toolbox functionality and educating users about digital behavioral studies.

The toolbox contributes to the scientific community by facilitating digital behavioral data preprocessing and analysis. Niimpy offers an adaptable data analysis framework enabling replicable and transparent results.

As the toolbox is still under development, more advanced analysis features will be incorporated into it over time. In the near future, we will incorporate readers for the Open mHealth data format [34] as well as data from the LAMP platform [16]. Our work promotes open science; thus, we encourage researchers to adopt and contribute to the toolbox with new analysis features.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We used synthetic data, part of the data is confidential, and part open-source: https://studentlife.cs.dartmouth.edu/dataset.html

Acknowledgments

We thank Professor Jari Saramäki for providing valuable feedback. We also thank Aalto Science-IT for providing computational resources and Aalto Research Software Engineers for their support. We thank Anna Hakala for their help with the project in its early stages. TA acknowledges the support of Professor Erkki Isometsä and other collaborators in the MoMo-Mood project, which has motivated the creation of the Niimpy toolbox.

Appendix

A.1. Toolbox layers by functionality

A.2. Toolbox preprocessing features

A.3. Exploration module description

Table 1
Layer functionality summary. For a detailed reference, see Niimpy documentation [26].

Layer Name	Functionality
Reading	Read data from stream or database into a dataframe
Preprocessing	Cleaning, encoding, feature extraction, integration, normalization, reduction, transformation
Exploration	Summary statistics, data visualization, data quality assessment
Analysis	Data modeling, statistical inference

Table 2
Preprocessing sub-module feature summary table. For a detailed reference, see Niimpy documentation [26].

Sub-module	Input time series	Features
Fitness tracker	Stepcount	Daily step count: mean, mean standard deviation, min, max, distribution per hour
Audio	Decibels, frequency, silence indicator	*Time window based features*: count silent, count speech, count loud, min freq, max freq, mean freq, median freq, std freq, min db, max db, mean db, median db, std db
Application	Application package name	Count, duration
Battery	Status, charge level	Shutdown event timestamp, datapoint occurrence, datapoint gaps, battery charge difference
Communication	Call or message events, including duration, type	Total call duration, mean call duration, median call duration, call duration std, call count, outgoing–incoming call ratio, sms count
Location	Latitude, longitude	*Distance based features*: total distance, variance, log variance, average speed, speed variance, max speed, location bin count. *Significant place related features*: static point count, moving point count, static bin count, max distance from home, number of significant places, number of rarely visited places, number of transitions between significant places, bin count in the top1, top2, top3, top4, and top5 cluster, normalized entropy
Screen	Screen status (including on, off, unlock, and lock events)	Screen off timestamp, screen event count, screen event duration, min screen event duration, max screen event duration, median screen event duration, mean screen event duration, screen event duration std, screen first unlock timestamp
Survey	Survey questions, including question ID, answer	Survey score summation: min, max, mean*, std* (∗: for applicable data)

Table 3
Exploration module function summary (∗: available for applicable data types). For a detailed reference, see Niimpy documentation [26].

Sub-module	Data Type	Visualization type	Usage
Categorical plot	Categorical	Barplot	Observation counts and distributions
Count plot	Categorical/Numerical	Barplot/Boxplot*	Observation counts and distributions
Lineplot	Numerical	Lineplot	Trend, cyclicity, patterns
Punchcard	Categorical/Numerical	Heatmap*	Temporal patterns of counts or values
Missingness	Categorical/Numerical	Barplot/Heatmap*	Missing data patterns

References

[1] Marsch LA. Digital health data-driven approaches to understand human behavior. Neuropsychopharmacology 2021;46(1):191–6. http://dx.doi.org/10.1038/s41386-020-0761-5, URL https://www.nature.com/articles/s41386-020-0761-5. Number: 1 Publisher: Nature Publishing Group.

[2] Onnela J-P. Opportunities and challenges in the collection and analysis of digital phenotyping data. Neuropsychopharmacology 2021;46(1):45–54. http://dx.doi.org/10.1038/s41386-020-0771-3, URL https://www.nature.com/articles/s41386-020-0771-3. Number: 1 Publisher: Nature Publishing Group.

[3] Insel TR. Digital phenotyping: Technology for a new science of behavior. JAMA 2017;318(13):1215–6. http://dx.doi.org/10.1001/jama.2017.11295.

[4] Aledavood T, Hoyos AMT, Alakörkkö T, Kaski K, Saramäki J, Isometsä E, et al. Data collection for mental health studies through digital platforms: requirements and design of a prototype. JMIR Res Protocols 2017;6(6):e110, Publisher: JMIR Publications Inc., Toronto, Canada.

[5] Barnett I, Torous J, Staples P, Sandoval L, Keshavan M, Onnela J-P. Relapse prediction in schizophrenia through digital phenotyping: a pilot study. Neuropsychopharmacology 2018;43(8):1660–6. http://dx.doi.org/10.1038/s41386-018-0030-z, URL https://www.nature.com/articles/s41386-018-0030-z. Number: 8 Publisher: Nature Publishing Group.

[6] Huckins JF, daSilva AW, Wang W, Hedlund E, Rogers C, Nepal SK, et al. Mental health and behavior of college students during the early phases of the COVID-19 pandemic: Longitudinal smartphone and ecological momentary assessment study. J Med Internet Res 2020;22(6):e20185. http://dx.doi.org/10.2196/20185, URL https://www.jmir.org/2020/6/e20185. Company: Journal of Medical Internet Research Distributor: Journal of Medical Internet Research Institution: Journal of Medical Internet Research Label: Journal of Medical Internet Research Publisher: JMIR Publications Inc., Toronto, Canada.

[7] Berrouiguet S, Ramírez D, Barrigón ML, Moreno-Muñoz P, Carmona Camacho R, Baca-García E, et al. Combining continuous smartphone native sensors data capture and unsupervised data mining techniques for behavioral changes detection: A case series of the evidence-based behavior (eB2) study. JMIR Mhealth Uhealth 2018;6(12):e197. http://dx.doi.org/10.2196/mhealth.9472.

[8] Luong N, Barnett I, Aledavood T. The impact of the COVID-19 pandemic on daily rhythms. 2023, arXiv preprint arXiv:2303.04535.

[9] Vega J, Li M, Aguillera K, Goel N, Joshi E, Khandekar K, et al. Reproducible analysis pipeline for data streams: Open-source software to process data collected with mobile devices. Front Digit Health 2021;3. URL https://www.frontiersin.org/article/10.3389/fdgth.2021.769823.

[10] Bent B, Wang K, Grzesiak E, Jiang C, Qi Y, Jiang Y, et al. The digital biomarker discovery pipeline: An open-source software

platform for the development of digital biomarkers using mHealth and wearables data. J Clin Transl Sci 2021;5(1). http://dx.doi.org/10.1017/cts.2020.511, URL https://www.cambridge.org/core/journals/journal-of-clinical-and-translational-science/article/digital-biomarker-discovery-pipeline-an-opensource-software-platform-for-the-development-of-digital-biomarkers-using-mhealth-and-wearables-data/A6696CEF138247077B470F4800090E63. Publisher: Cambridge University Press.

[11] Onnela J-P, Dixon C, Griffin K, Jaenicke T, Minowada L, Esterkin S, et al. Beiwe: A data collection platform for high-throughput digital phenotyping. J Open Source Softw 2021;6(68):3417. http://dx.doi.org/10.21105/joss.03417, URL https://joss.theoj.org/papers/10.21105/joss.03417.

[12] Ferreira D, Kostakos V, Dey AK. AWARE: Mobile context instrumentation framework. Front ICT 2015;2. URL https://www.frontiersin.org/article/10.3389/fict.2015.00006.

[13] Torous J, Kiang MV, Lorme J, Onnela J-P. New tools for new research in psychiatry: a scalable and customizable platform to empower data driven smartphone research. JMIR Ment Health 2016;3(2):e16, Publisher: JMIR Publications Inc., Toronto, Canada.

[14] Bardram JE. The CARP mobile sensing framework–A cross-platform, reactive, programming framework and runtime environment for digital phenotyping. 2020, arXiv preprint arXiv:2006.11904.

[15] Onnela-Lab. Onnela-Lab/Forest: Forest is a library for analyzing smartphone-based high-throughput digital phenotyping data. 2023, GitHub. URL https://github.com/onnela-lab/forest. [Accessed 09 March 2023].

[16] Bilden R, dcurrey88, Vaidyam A, Patel S, Meyer A, Scheuer L, et al. BIDMCDigitalPsychiatry/LAMP-platform: release 2023.2.15. 2023, Zenodo. http://dx.doi.org/10.5281/zenodo.7643628.

[17] Wang X, Vouk N, Heaukulani C, Buddhika T, Martanto W, Lee J, et al. HOPES: An integrative digital phenotyping platform for data collection, monitoring, and machine learning. J Med Internet Res 2021;23(3):e23984. http://dx.doi.org/10.2196/23984, URL https://www.jmir.org/2021/3/e23984. Company: Journal of Medical Internet Research Distributor: Journal of Medical Internet Research Institution: Journal of Medical Internet Research Label: Journal of Medical Internet Research Publisher: JMIR Publications Inc., Toronto, Canada.

[18] Ranjan Y, Rashid Z, Stewart C, Conde P, Begale M, Verbeeck D, et al. RADAR-base: open source mobile health platform for collecting, monitoring, and analyzing data using sensors, wearables, and mobile devices. JMIR MHealth UHealth 2019;7(8):e11734.

[19] Tableau: Business intelligence and analytics software. 2023, Tableau. URL https://www.tableau.com/node/62770. [Accessed 09 March 2023].

[20] Köster J, Rahmann S. Snakemake—a scalable bioinformatics workflow engine. Bioinformatics 2012;28(19):2520–2.

[21] Cookiecutter data science. In: Home - Cookiecutter data science. 2023, URL https://drivendata.github.io/cookiecutter-data-science/. [Accessed 09 March 2023].

[22] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. Nature Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.

[23] The pandas development team. pandas-dev/pandas: Pandas. 2020.

[24] Estrin D, Sim I. Open mHealth architecture: an engine for health care innovation. Science 2010;330(6005):759–60.

[25] Plotly Technologies Inc. Collaborative data science. Place: Montreal, QC: Plotly Technologies Inc.; 2015, URL https://plot.ly.

[26] Niimpy: behavioral data analysis — Niimpy dev documentation. 2023, URL https://niimpy.readthedocs.io/en/latest/. [Accessed 09 March 2023].

[27] Waskom ML. Seaborn: statistical data visualization. J Open Source Softw 2021;6(60):3021. http://dx.doi.org/10.21105/joss.03021.

[28] Wang R, Chen F, Chen Z, Li T, Harari G, Tignor S, et al. StudentLife: assessing mental health, academic performance and behavioral trends of college students using smartphones. In: Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing. 2014, p. 3–14.

[29] Kroenke K, Spitzer RL, Williams JB. The PHQ-9: validity of a brief depression severity measure. J Gen Intern Med 2001;16(9):606–13.

[30] Kaggle public API documentation. 2023, URL https://www.kaggle.com/docs/api. [Accessed 09 March 2023].

[31] Polar accesslink API V3. 2023, URL https://www.polar.com/accesslink-api/#polar-accesslink-api. [Accessed 09 March 2023].

[32] Triana AM, Martikkala A, Baryshnikov I, Heikkilä R, Alakörkkö T, Darst RK, et al. Mobile monitoring of mood (MoMo-mood) pilot: A longitudinal, multi-sensor digital phenotyping study of patients with major depressive disorder and healthy controls. Health Informatics; 2020, http://dx.doi.org/10.1101/2020.11.02.20222919, URL http://medrxiv.org/lookup/doi/10.1101/2020.11.02.20222919.

[33] Baryshnikov I, Aledavood T, Rosenström T, Heikkilä R, Darst R, Riihimäki K, et al. Relationship between daily rated depression symptom severity and the retrospective self-report on PHQ-9: A prospective ecological momentary assessment study on 80 psychiatric outpatients. J Affect Disord 2023;324:170–4.

[34] Chen C, Haddad D, Selsky J, Hoffman JE, Kravitz RL, Estrin DE, et al. Making sense of mobile health data: an open architecture to improve individual-and population-level health. J Med Internet Res 2012;14(4):e2152.