The Journal of Open Source Software

Devicely: A Python package for reading, timeshifting and writing sensor data

Ariane Sasso[*1], **Jost Morgenstern**[1], **Felix Musmann**[1], **and Bert Arnrich**[1]

1 Digital Health Center, Hasso Plattner Institute, University of Potsdam

DOI: 10.21105/joss.03679

Software

- Review
- Repository
- Archive

Editor: Prashant K Jha

Reviewers:

- @luciorq
- @djmitche

Submitted: 24 August 2021
Published: 14 October 2021

Summary

Wearable devices can track a multitude of parameters such as heart rate, body temperature, blood oxygen saturation, acceleration, blood glucose and many more (Kamišalić et al., 2018). Moreover, they are becoming increasingly popular with a steep increase in market presence in 2020 alone (IDC, 2020). Applications for wearable devices vary from tracking cardiovascular risks (Bayoumy et al., 2021) to identifying COVID-19 onset (Mishra et al., 2020). Therefore, there is a great need for scientists to easily go through data acquired from different wearables and to be able to share them while protecting user privacy. In order to solve this problem and empower scientists working with biosignals, we developed the **devicely** package. It processes the data into a tabular format and contains tools for data de-identification. It allows scientists to focus on what they want: the analysis of biosignals guided by privacy principles.

Related Work

The first example of a package working with wearable data is mhealthtools (Snyder et al., 2020), which is developed in R and focuses on extracting features from sensors such as inertial measurement units (IMUs). Its main difference from **devicely** is firstly the language (R versus Python) and secondly their complementary nature. Mhealthtools offers functionalities for feature extraction and **devicely** is intended to help users in a prior step by reading and writing data from wearables into standardized formats.

There are also packages developed in Python, such as SleepPy (Christakis et al., 2019) which uses raw accelerometer data for assessing sleep quantity and quality. The HRV (Bartels & Peçanha, 2020) package uses CSV and text files or Python iterables such as lists to generate features related to heart rate variability (HRV). GaitPy (Czech & Patel, 2019) accepts input data in a customizable format and is mainly used to extract features for gait analysis. Therefore, packages such as SleepPy, HRV and GaitPy could also be used in a following step to extract features from the output generated by **devicely**.

FLIRT (Maritsch et al., 2020) and wearablecompute (Bent, 2020) are packages that provide ways for reading data from specific wearables such as Empatica E4. They also include functionalities to extract features from electrodermal activity (EDA), acceleration and HRV. The main difference from FLIRT and wearablecompute to **devicely** is the focus on feature extraction versus privacy and data sharing. FLIRT and wearablecompute read the data for extracting features, while **devicely** aims to provide users with a way to read the data, de-identify them as necessary and write them back in a specified format. In this way, researchers can ensure even more data privacy and use the data easily for further analysis and sharing.

*corresponding author

Statement of need

Every wearable vendor has a different data format and reading them is usually a challenge for scientists. Therefore, in order for researchers to be able to use different sensor data in an easy and friendly way we developed the **devicely** package. The package also contains two methods to help with data de-identification, one is called timeshift and the other is a write method. The idea behind them is that researchers can timeshift all their time series data to a different time from the one the actual experiments occurred and then write this new de-identified dataset back to the original or a similar data format. This will empower scientists to maintain user privacy and hopefully share more data to increase research reproducibility.

Design

Different wearables provide incompatible data formats which require specific preprocessing steps. However, it should be easy for scientists to add data from a new wearable to an existing pipeline and easy for developers to add a new wearable to the **devicely** package. To achieve both, **devicely** encapsulates data preparation for each wearable behind three common methods: *read*, *timeshift* and *write*.

After reading, the data is accessible through the reader in common formats such as Pandas DataFrames. De-identification is achieved by timeshifting the data, either by providing a shifting interval or randomly. For writing back de-identified data, **devicely** focuses on keeping a format that can be read again using the same reader class. In almost all cases, this is the same format as the one the wearable originaly provides. This enables sharing data with the community while maintaining user privacy.

Functionalities

All reader classes support three core functions: reading data created by a wearable, timeshifting them and writing them back. To *read* data the corresponding reader class can be initialized using as a parameter a path to the data created by the wearable. After reading, data can be accessed through the reader in convenient formats such as dictionaries and Pandas DataFrames.

After creating a reader object the method *timeshift* can be applied upon it. This assures de-identification by shifting all time-related data points. To control the shifting interval, a parameter can be provided to *timeshift*. If no parameter is provided, the data is shifted by a random time interval to the past. The timeshifted data can be written back using the *write* method.

For all wearables, the written data can be read again using the same reader class. Figure 1 depicts the class structure of the **devicely** package and serves as a guide for future implementation.

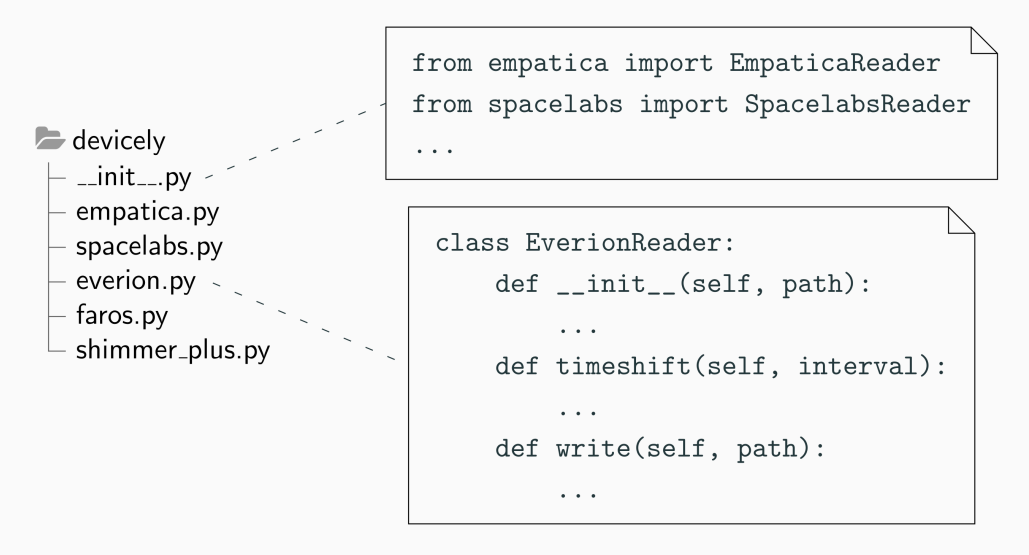

Figure 1: On the left side, the structure of the files in the **devicely** package is depicted. Each device should have a separate file. On the right side at the top we show how to import a class from a device file into ___init___.py. At the right bottom side there is an example of one device class and its methods.

Availability

The software can be obtained through the Python Package Index (PyPI), Conda-forge, Zenodo and on GitHub under the MIT License.

Mention

This package was used in the following paper:

Morassi Sasso A. et al. (2020) HYPE: Predicting Blood Pressure from Photoplethysmograms in a Hypertensive Population. In: Michalowski M., Moskovitch R. (eds) Artificial Intelligence in Medicine. AIME 2020. Lecture Notes in Computer Science, vol 12299. Springer, Cham. https://doi.org/10.1007/978-3-030-59137-3_29

GitHub Repository

Acknowledgements

We acknowledge contributions from Arpita Kappattanavar, Bjarne Pfitzner, Harry Freitas da Cruz, Lin Zhou, Pascal Hecker, Philipp Hildebrandt and Sidratul Moontaha during the genesis and testing of this package.

References

Bartels, R., & Peçanha, T. (2020). HRV: A pythonic package for heart rate variability analysis. *Journal of Open Source Software*, *5*(51), 1867. https://doi.org/10.21105/joss.01867

Bayoumy, K., Gaber, M., Elshafeey, A., Mhaimeed, O., Dineen, E. H., Marvel, F. A., Martin, S. S., Muse, E. D., Turakhia, M. P., Tarakji, K. G., & Elshazly, M. B. (2021). Smart wearable devices in cardiovascular care: where we are and how to move forward. *Nature Reviews Cardiology*. https://doi.org/10.1038/s41569-021-00522-7

Bent, B. (2020). *Wearablecompute is an open source python package containing over 50 data and domain-driven features*. DBDP (Digital Biomarker Discovery Pipeline). https://github.com/brinnaebent/wearablecompute

Christakis, Y., Mahadevan, N., & Patel, S. (2019). SleepPy: A python package for sleep analysis from accelerometer data. *Journal of Open Source Software*, *4*(44), 1663. https://doi.org/10.21105/joss.01663

Czech, M. D., & Patel, S. (2019). GaitPy: An open-source python package for gait analysis using an accelerometer on the lower back. *Journal of Open Source Software*, *4*(43), 1778. https://doi.org/10.21105/joss.01778

IDC. (2020). *Shipments of Wearable Devices Leap to 125 Million Units, Up 35.1% in the Third Quarter, According to IDC*. https://www.idc.com/getdoc.jsp?containerId=prUS47067820

Kamiŝalić, A., Fister, I., Turkanović, M., & Karakatiĉ, S. (2018). Sensors and functionalities of non-invasive wrist-wearable devices: A review. *Sensors (Switzerland)*, *18*(6). https://doi.org/10.3390/s18061714

Maritsch, M., Föll, S., Fleisch, E., Kowatsch, T., Wortmann, F., & Spinola, F. (2020). *FLIRT: Feature generation tooLkIt for weaRable daTa*. Chair of Information Management at ETH Zurich; the Institute of Technology Management at University of St. Gallen. https://github.com/im-ethz/flirt

Mishra, T., Wang, M., Metwally, A. A., Bogu, G. K., Brooks, A. W., Bahmani, A., Alavi, A., Celli, A., Higgs, E., Dagan-Rosenfeld, O., Fay, B., Kirkpatrick, S., Kellogg, R., Gibson, M., Wang, T., Hunting, E. M., Mamic, P., Ganz, A. B., Rolnik, B., … Snyder, M. P. (2020). Pre-symptomatic detection of COVID-19 from smartwatch data. *Nature Biomedical Engineering*, *4*(12), 1208–1220. https://doi.org/10.1038/s41551-020-00640-6

Snyder, P., Tummalacherla, M., Perumal, T., & Omberg, L. (2020). Mhealthtools: A modular r package for extracting features from mobile and wearable sensor data. *Journal of Open Source Software*, *5*(47), 2106. https://doi.org/10.21105/joss.02106