R package for animal behaviour classification from accelerometer data - rabc

Hui Yu[1] and Marcel Klaassen[1]

[1]Deakin University Faculty of Science Engineering and Built Environment

November 21, 2020

Abstract

Increasingly animal behaviour studies are enhanced through the use of accelerometry. To allow translation of raw accelerometer data to animal behaviours requires the development of classifiers. Here, we present the "rabc" package to assist researchers with the interactive development of such animal-behaviour classifiers based on datasets consisting out of accelerometer data with their corresponding animal behaviours. Using an accelerometer and a corresponding behavioural dataset collected on white stork (Ciconia ciconia), we illustrate the workflow of this package, including raw data visualization, feature calculation, feature selection, feature visualization, extreme gradient boost model training, validation, and, finally, a demonstration of the behaviour classification results.

R package for animal behaviour classification from accelerometer data - rabc

Hui Yu[1, 2], Marcel Klaassen[1]

Affiliations

Centre for Integrative Ecology, School of Life and Environmental Sciences, Deakin University, Geelong, Victoria, Australia

Druid Technology Co., Ltd, Chengdu, Sichuan, China

Abstract

Increasingly animal behaviour studies are enhanced through the use of accelerometry. To allow translation of raw accelerometer data to animal behaviours requires the development of classifiers. Here, we present the "rabc" package to assist researchers with the interactive development of such animal-behaviour classifiers based on datasets consisting out of accelerometer data with their corresponding animal behaviours. Using an accelerometer and a corresponding behavioural dataset collected on white stork (*Ciconia ciconia*), we illustrate the workflow of this package, including raw data visualization, feature calculation, feature selection, feature visualization, extreme gradient boost model training, validation, and, finally, a demonstration of the behaviour classification results.

Keywords: animal behaviour classification, accelerometer, XGBoost, data visualization, interactive process

Introduction

Our understandings of animal movement patterns and behaviours continue to rapidly advance with the use of ever smarter and smaller tracking technologies (Ropert-Coudert & Wilson, 2005; Williams et al., 2019). Increasingly, the tracking of animals is also combined with accelerometer (ACC) data collection to study the free-roaming behaviours of animals across a wide range of taxa (Brown, Kays, Wikelski, Wilson, & Klimley, 2013; Shepard et al., 2008). Compared with direct human observation, using ACC to study animal

behaviours has the obvious advantage that it reduces the influence of human presence and also allows the recording of behaviours that would otherwise be hard to observe, away from the human eye (Brown et al., 2013). However, these obvious merits of ACC technology can only be achieved when a reliable behaviour classification model is available that can convert raw ACC data into meaningful behaviour types.

Many studies have already conducted behaviour classification from ACC data (e.g., Nathan et al., 2012). In most cases, ACC data with corresponding behavioural field observations are used to train behaviour classification models (e.g., Kölzsch et al., 2016; Kroschel, Reineking, Werwie, Wildi, & Storch, 2017). However, in some instances the thus developed classifiers that translate ACC data into behaviour types yield only low classification accuracy (Fehlmann et al., 2017). As a general remedy, using fewer behaviour classes and aggregating behaviours usually yields better classification performance (Ladds et al., 2017). However, such grouping of behaviours is typically based on biological or ecological considerations and not necessarily also considering the capacity of ACC data to discriminate between behaviours. It is this often iterative process of combining and splitting behaviours within the behaviour set that the here presented rabc package also endeavours to assist with. In this way, the rabc package allows the user to derive optimal and validated behaviour classifiers suited to their specific research system and questions.

To help biologists translate ACC data into behaviours this package uses XGBoost, which is currently one of the most promising supervised machine learning methods for this specific purpose (Hui et al., in prep). Unlike the web-based tool "AcceleRater" (Resheff, Rotics, Harel, Spiegel, & Nathan, 2014), our rabc package does not focus on providing a "one-stop service" turning raw ACC data into behaviours. Rather, this package focuses on (1) providing interactive visualization tools to its user to assist in handling and interpreting the ACC input data, (2) decide on appropriate behaviour categories for classification as highlighted in the previous paragraph, and (3) reduce ACC data volume efficiently and effectively (through the calculation and selection of a range of features) without compromising behaviour classification performance. In brief, this package endeavours to open the lid of the machine-learning "black-box", allowing the integration of the user's expert knowledge on their own research system in developing advanced behaviour-classification models.

Rabc workflow

The general workflow of the rabc package to transform ACC data using supervised machine learning methods into behaviours is outlined in Fig. 1. The data flow is composed of the following elements (where the numbering refers to the paragraphs where these are being described in detail): 2.1 ACC dataset preparation with behaviour labels; 2.2 ACC visualization; 2.3 Feature calculation; 2.4 Feature selection; 2.5 Feature visualization; 2.6 Model training and valication; 2.7 Classification result check. The rabc package can be installed in Rstudio by "devtools::install_github("YuHuiDeakin/rabc")".

Hosted file

`image1.emf` available at https://authorea.com/users/377610/articles/494253-r-package-for-animal-behaviour-classification-from-accelerometer-data-rabc

Figure 1. The general workflow of the rabc package to develop classifiers for adequately transforming ACC data into behaviours. The various elements in the diagram are numbered according to the paragraphs where these are being described in detail.

2.1 ACC dataset preparation and behaviour labels

Segments of continuous ACC data will need to be translated into meaningful behaviours. For raw ACC data segmentation, there are two choices: even-length segmentation and variable-length segmentation (Bom, Bouten, Piersma, Oosterbeek, & van Gils, 2014). Variable-length segmentation requires an algorithm to detect behaviour change points and may thus be prone to error. Even-length segmentation does not require these additional calculations and is therefore much easier to implement. However, even-length ACC segments will inevitably contain behaviour change points (and thus multiple behaviours) affecting down the line processing and behaviour classification. An ACC segment should be sufficiently long to contain enough data to be representative of a behaviour (and, thus, interpretable as a specific behaviour type), whereas

its length should be limited to avoid inclusion of multiple behaviours as much as possible. Regarding the inevitable segments where behaviour transitions take place, we recommend retaining these segments in the model training. Although these data might decrease the accuracy of the classification model, they will make the model more robust and avoid overestimating model performance. The rabc package only supports even-length segmentation data. The input data should be a data.frame or tibble containing raw ACC data including the behaviour associated with the ACC data. For tri-axial ACC data, each row of equal length should be arranged as "x,y,z,x,y,z,. . . ,behaviour", where "behaviour" is the (primary) behaviour observed during that segment. For dual-axial ACC data, it should be arranged as "x,y,x,y,. . . ,behaviour" and for single-axial ACC data as "x,x,. . . ,behaviour".

The here used tri-axial ACC demo dataset from white stork (*Ciconia ciconia*) (data accessible from the AcceleRater website: http://accapp.move-ecol-minerva.huji.ac.il/, see Resheff et al., 2014) was measured at 10.54 Hz. Forty tri-axial measurements, totalling 3.8 seconds, were used to form a behaviour segment. The dataset includes 1746 segments each forming a row in the dataset. Each row contains 121 columns. The first 120 columns are ACC measurements from three orthogonal axes, arranged as x,y,z,x,y,z,. . . ,x,y,z. The final column is of type character containing the corresponding behaviour. The dataset contains 5 different behaviours including "A_FLIGHT" - active flight (77 cases), "P_FLIGHT" - passive filght (96), "WALK" - walking (437), "STND" - standing (863), "SITTING" - sitting (273).

2.2 ACC visualization

Prior to visualising the ACC data, the dataset needs to be sorted by behaviour using the order_acc function. For ACC data visualization, the rabc package uses function dygraph (from package dygraphs) to plot all ACC segments grouped by behaviour. This dynamic mode of presentation is convenient for users to zoom-in or scroll through behaviour segments. It provides the user a visual impression of how the ACC signal generally relates to the different behaviours and can also be used for data quality control (i.e. identifying potentially incorrect segments where ACC and behavioural data do not conform to the general pattern otherwise observed due to, for instance, incorrect behavioural observation). The x axis of this dygraph indicates the row sequence number (i.e. the segment number) of the sorted data.

Plotting the complete white stork ACC dataset using function plot_acc (Fig 2a) and next zooming in on the area around segments 55 to 80 (Fig 2b), it can be seen that the ACC data between segments 60 and 70 is very different from neighbouring segments. Albeit all being labelled as "A_FLIGHT", the ACC data in this range resemble more static behaviours, warranting their scrutiny and, potentially, their relabelling or removal from the dataset.

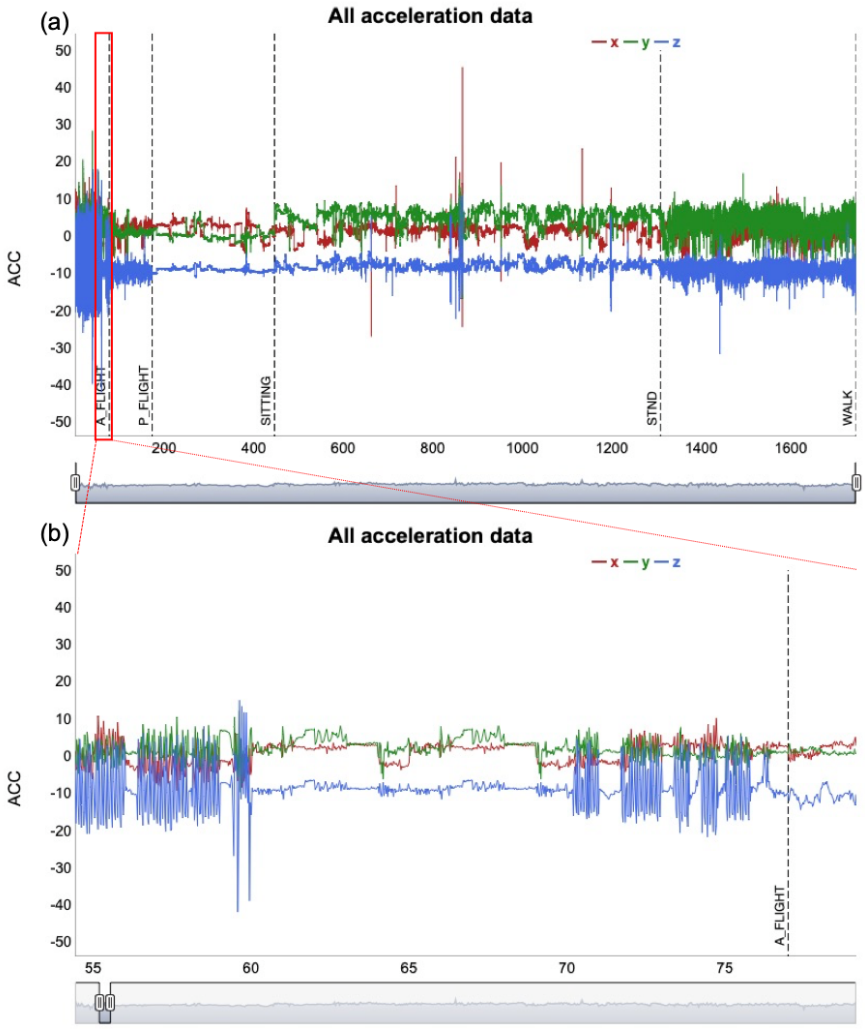


Figure 2. ACC data visualization using a dynamic graph. Panel**a** shows the complete white stork ACC dataset, sorted by behaviour type. The X axis shows the segment numbers of the dataset ordered by behaviour. Vertical dashed lines separate different behaviour types. Panel**b** demonstrates how one can zoom in on specific segment ranges, here from segment 55 to 80.

2.3 Feature calculation

The next step is to calculate features from the ACC data. A feature is a specific mathematical description (such as e.g. the mean, the standard deviation, etc.) of the ACC signal within a segment, which will form the input to the machine learning models (Brown et al., 2013). Using functions calculate_feature_time and calculate_feature_freq, two basic feature sets are calculated. The first, time-domain feature set, includes: mean, variance, standard deviation, max, min, range and ODBA, where ODBA is short for Overall Dynamic Body Acceleration. This value has been proven to be correlated with the animal's energy expenditure (Wilson et al., 2019). These features are calculated for each ACC axis separately (denoted with prefix x, y, z in the output data frame), except for ODBA, which is calculated using all available axes. The frequency-domain feature set includes: main frequency, main amplitude and frequency entropy. Also, these features are calculated for each ACC axis separately (denoted with prefix x, y, z). Calculations of these features are based

on Fast Fourier Transformation (FFT) of raw ACC data. Frequency entropy here measures unpredictability of the signal. It is worth noting that due to specific ACC sampling settings (e.g., Gilbert et al., 2016), some of the resulting ACC datasets may not have a high enough sampling frequency to log useful frequency information (Nathan et al., 2012). In these cases, it is better not to use frequency-domain features for behaviour classification. In addition, it should be considered that the functions calculate_feature_time and calculate_feature_freq provide an essential but not an exhaustive list of potential features. Since it has been asserted that feature engineering can improve the performance of machine-learning models (Boehmke & Greenwell, 2019), users may consider calculation of custom features. All functions in the rabc package are also able to process custom features after the user has included these in the feature data frame using functions cbind or bind_cols (from the dplyr package).

2.4 Feature selection

Feature selection is the process of selecting a subset of relevant features for use in model building (Chakravarty, Cozzi, Ozgul, & Aminian, 2019). In animal behaviour studies using ACC, tens of features are typically used in model building (e.g., Shamoun-Baranes et al., 2012). Although a relatively small number, compared to many other machine-learning models, there may still be redundancy in the feature set. Redundant features are features that show high correlation with other features and are thus likely to contribute similarly to the behaviour classification model. Redundant features may also be "irrelevant" features that hardly contribute to the classification model. Three aims are being served with feature selection in this package. Firstly, less features will make the model easier to interpret. Indeed, there may for instance be biomechanical connections between features and the ultimate classification model (e.g., Chakravarty et al., 2019). Secondly, fewer features reduce the risk of overfitting and may therewith lead to better behaviour classification from ACC data. Thirdly and finally, because of lower computational requirements in assessing behaviour from ACC data, reduced feature sets have greater potential to be calculated on-board the ACC devices themselves, e.g. on-board of light-weight tracking devices (e.g., Korpela et al., 2020; Nuijten, Gerrits, Shamoun-Baranes, & Nolet, 2020) on which they can either be stored or relayed to receiving stations.

The rabc package's select_features function uses a combination of a filter and a wrapper feature selection method. The filter part removes any redundant features based on the absolute values of the pair-wise correlation coefficients between features. If two features have a high correlation, the function looks at the absolute correlation of each of the two features with all other features and removes the feature with the largest mean absolute correlation value. The threshold correlation coefficient (cutoff) is user-defined with a default "cutoff = 0.9". In the default constellation the filter function is turned off (i.e. "filter = FALSE").

The purpose of the wrapper is to select most relevant features. The wrapper part applies stepwise forward selection (SFS) (Tološi & Lengauer, 2011) using the extreme gradient boosting (XGBoost) model, which is not only used for feature selection but also for the final classification model (see below). XGBoost is a scalable tree boosting method that proved to be better than other tree boosting methods and random forest (Chen & Guestrin, 2016). We also experienced ourselves that XGBoost is fast to train and has good performance with limited numbers of trees. The default limit to the number of features (no_features) is 5 but can be user defined. The no_features also determines how many rounds of SFS are being conducted. In the first round, each feature is individually used to train a classification model by XGBoost. The feature with highest overall accuracy will be kept into the selected feature set. Then, in every following round, each remaining feature will be combined with the selected feature set to train a classification model and the one with the highest accuracy will be kept into the selected feature set. The process will stop when the number of rounds equals the no_features setting.

The select_features function will return a list, of which the first member (i.e., .[[1]]) contains a matrix providing the classification accuracy for each of the features (columns) across all steps (rows, top row being the first step) of the SFS process. Once a feature is selected into the selected feature set, the remaining values in this feature's column are set to zero. The second member of the list (i.e., .[[2]]) contains the names of the selected features in the order in which they were selected in the SFS process. The development of the classification accuracy with each step in the SFS process is plotted with function plot_selection_accuracy

(Fig. 3). In the case of the White Stork dataset, we can see that after the sixth selected feature, "z_variance", there is almost no further improvement in classification accuracy with the addition of more features.

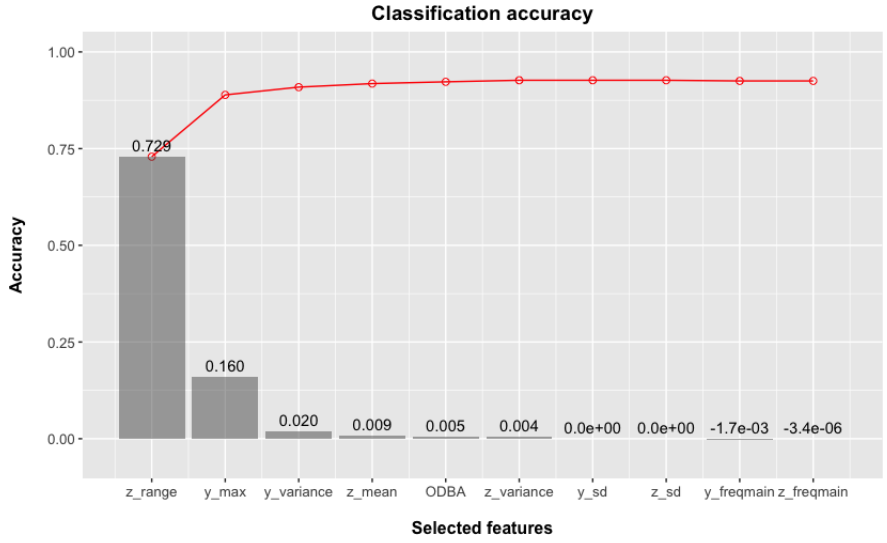


Figure 3. Classification accuracy plot providing an overview of the individual (grey bars) and cumulative (red line and circles) contribution of each feature (in the in which they were selected in the stepwise forward selection (SFS) process).

2.5 Feature visualization

Above, under "Feature selection" we already mentioned the three objectives with feature selection: improving interpretability, reducing overfitting, reducing computational requirements. Visualisation of the features can further assist in deciding on the features to use in the ultimate behaviour classification model, yet its main use is in deciding if any behaviour types should be combined to ultimately improve behaviour classification performance. Alternatively, the visualisation may also lead to considering splitting up existing behaviour types into multiple behaviours. In other words, this visualisation aids in evaluating the behaviour set.

The rabc package offers three ways to visualize features. The first two visualise the features in isolation whereas the third is an integrative approach where entire feature domains are analysed collectively. The first of the visualisation methods, plot_feature, draws individual values of features ordered by behaviour (Fig. 4). The second, plot_grouped_feature, produces a boxplot of a selected feature for all behaviour types, as demonstrated for the ODBA feature in Fig. 5. In the case of the White Stork dataset it suggests clear differentiation of behaviours by ODBA with a trend of ODBA decreasing going from active flight via walking to passive flight, standing and sitting. The third and most important, integrative approach uses Uniform Manifold Approximation and Projection (UMAP).

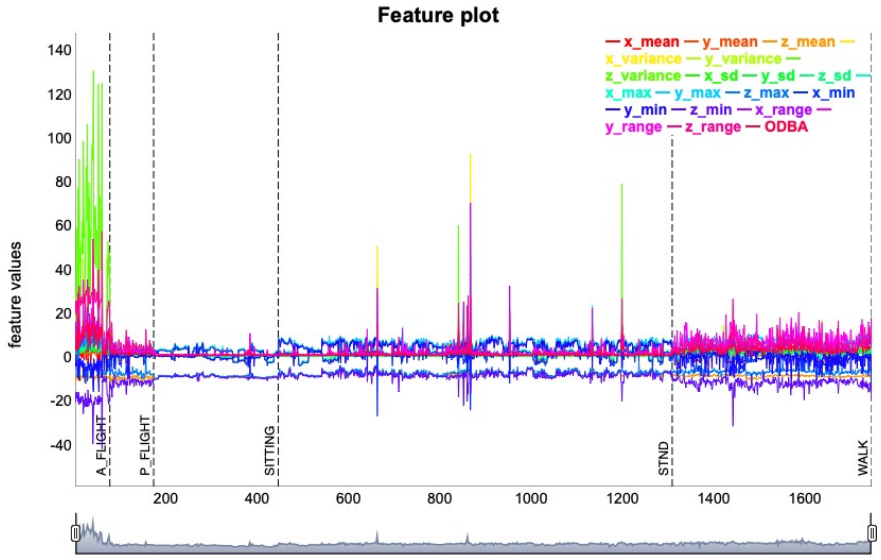


Figure 4. Feature data visualization using a dynamic graph. The features in this plot are calculated by function calculate_feature_time. The X axis shows the segment numbers of the features ordered by behaviour. Vertical dashed lines separate different behaviour types.

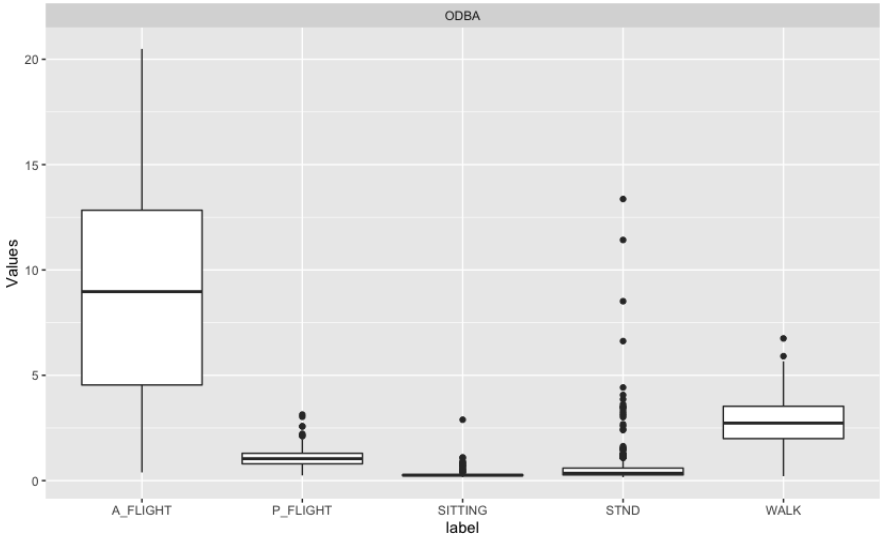
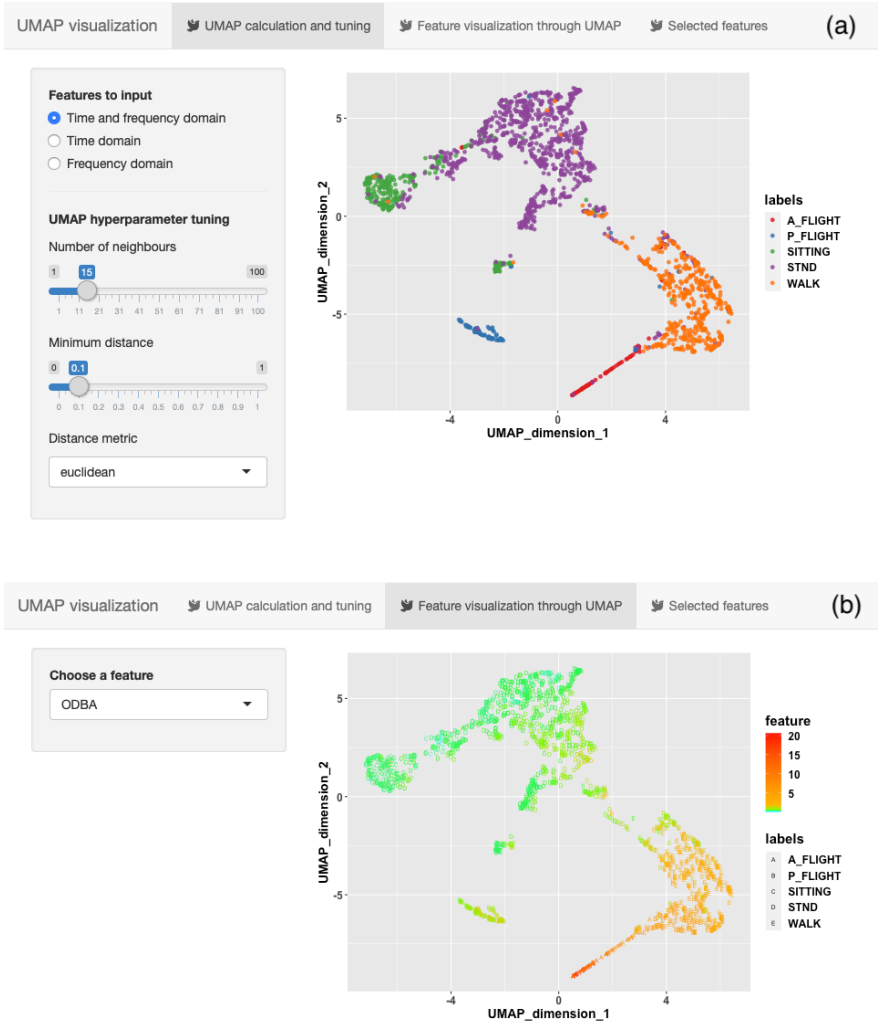


Figure 5. Boxplot of the feature ODBA.

UMAP is a very powerful nonlinear dimensionality-reduction technique, which is also highly suitable for high-dimensional data visualization (McInnes, Healy, Saul, & Grossberger, 2018) and we will here use it to transform and visualise collections of features in a two-dimensional plot. UMAP has already found its niches in bioinformatics, material sciences and machine learning (McInnes et al., 2018). Within the broad field of biology, it has been used in bioacoustics studies (e.g., Sainburg, Theilman, Thielk, & Gentner, 2019), but it has rarely been used in animal behaviour studies. In the rabs package we use UMAP to plot the different behaviours, represented by differently coloured symbols in the two-dimensional space. The optimal scenario to which one strives is to obtain a representation where each behaviour forms an isolated cluster of symbols within this two-dimensional space. In this way, UMAP provides an indication of how the final classification

model will perform; isolated behaviour clusters indicating high classification accuracy. There where overlaps in clusters exists, researchers may wish to consider grouping certain behaviours because they may not be adequately separated using ACC data. Conversely, there where behaviours are spread out over a plot, having those behaviours reclassified in multiple behaviour types, may be a possibility.

We made the UMAP visualization into a Shiny App to facilitate user interaction. There are three tabs in the Shiny App, representing three functions. Tab 1: "UMAP calculation and tuning" – assists with evaluating whether ACC features adequately represent behaviours. Tab 2: "Feature visualization through UMAP" – can show how feature values vary across the two-dimensional UMAP plot. Tab 3: "Selected features" – assists with evaluating the performance of selected features in differentiating between the different behaviours. In Fig. 6 we show screenshots of the three UMAP tabs, loaded with the time and frequency domain features from the white stork dataset. It shows that the different behaviours separate generally well (Fig. 6a), suggesting that there is good potential to develop a satisfactory performance behaviour classification model. In the next tab (Fig.6b), we selected the ODBA feature, the plot showing how its value varies across the different behaviour types with active flight having distinguishably high ODBA values followed by walking, then passive flight, standing and sitting. Finally, in the third tab (Fig.6c), we only selected the six features identified by function select_features to form a new UMAP plot. We can see that these features can preserve the manifold structure of the different behaviours. The demo of this Shiny App can be access through < https://huiyu-deakin.shinyapps.io/rabc_UMAP/>.



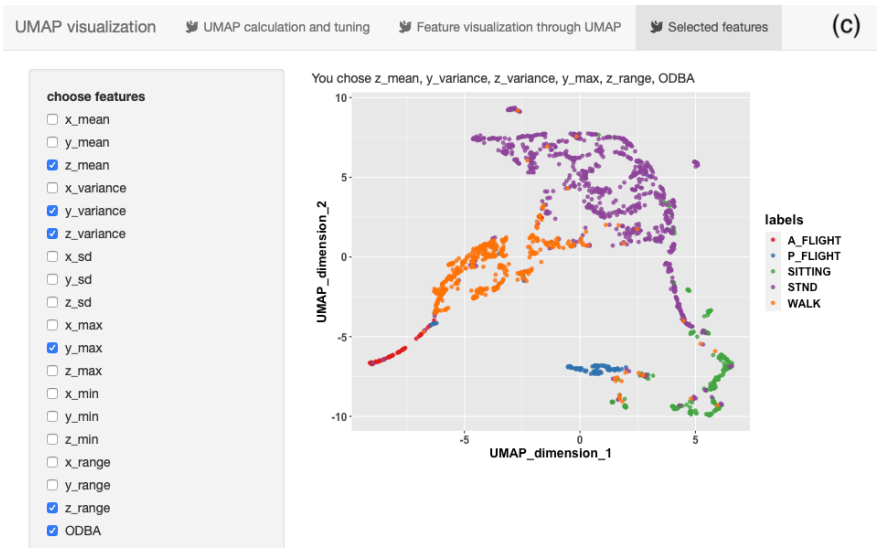


Figure 6. Demonstrations of the three tabs generated by the plot_UMAP function. Tab **a** -UMAP calculation and tuning – evaluates whether ACC features represent behaviours. The "Features to input" section allows users to choose which feature groups to use as input to UMAP. The "UMAP hyperparameter tuning" section allows users to interactively adjust three hyperparameters within the UMAP function to control the two-dimensional clustering. Tab **b** – Feature visualization through UMAP – shows how feature values vary across the two-dimensional UMAP plot. Users can choose which feature to plot by selecting from the drop-box. Tab **c** – Selected features – allows evaluating the performance of selected features in differentiating between the different behaviours. Users can choose which features to input into UMAP by ticking the checkboxes.

2.6 Model training and validation

After feature selection and visualisation (including potential reclassification of behaviour types), the user can train a supervised machine learning model (XGBoost in this package) with the selected, most relevant features through function train_model. Usually, the construction and evaluation of supervised machine learning models includes three steps: (i) machine learning model hyperparameter tuning by cross-validation, (ii) model training with the optimal hyperparameter set, and (iii) evaluating model performance through validation with a test dataset. Function train_model is a wrapper function that utilizes relevant functions from the "caret" package to automatically conduct the three above steps for model construction and evaluation.

Four arguments can be set in the function train_model to control the training and validation processes. Which features to use for model building is set by "df", which in the following example is set to "selection$features[1:6]" (i.e. the first six selected features from the feature selection procedure). The "vec_label" argument is used to pass on a vector of behaviour types. How to select the hyperparameter set is set by "hyper_choice", which has two options: "defaults" will let XGBoost use its default hyperparameters (nrounds = 10) while "tune" will run repeated cross-validations to find a best set (note that the settings for the hyperparameters inside this function are based on our previous experience with a range of different ACC datasets (Hui et al., in prep) and are set at: nrounds = c(5, 10, 50, 100), max_depth = c(2, 3, 4, 5, 6), eta = c(0.01, 0.1, 0.2, 0.3), gamma = c(0, 0.1, 0.5), colsample_bytree = 1, min_child_weight = 1, subsample = 1). Finally, "train_ratio" determines the percentage of data used to train the model, the remainder of the data being used for model validation.

The ultimate output consists out of four parts. The first is a confusion matrix, depicting how well the ultimate behaviour classification model predicts the different behaviours based on the validation part of the dataset only (i.e 25% of the dataset in our stork example using a train_ratio of 0.75). On the diagonal of this table, where the observed behaviour is organised in columns and the predicted behaviour is

organised in rows, the correct predictions are depicted, with all the wrong predictions being off the diagonal. The overall performance statistics are presented next, the meaning of which is explained in detail in <https://topepo.github.io/caret/measuring-performance.html>. The third part of the output, statistics by class, presents a range of performance statistics for the individual behavioural categories, which are explained in detail in <https://topepo.github.io/caret/measuring-performance.html>. Finally, the importance of the various features in producing the behaviour classification model is being presented.

Another way of calculating and visualising the performance of the behavioural classification model makes use of cross-validation using function plot_confusion_matrix. In this case the entire dataset is randomly partitioned into five parts. In five consecutive steps, each of the five parts is used as a validation set, while the remaining four parts are used for model training. This procedure thus resembles a five-fold "classification model training and validation" with a train_ratio of 0.8, be it that in this case the dataset is systematically divided and each point in the dataset is being used for the validation process at some point (see function createFolds in "caret" for more details). Thus, after all five training and validation rounds, all behavioural observations will also have an associated predicted behaviour, which are being stored in the data frame that is being returned by plot_confusion_matrix in addition to a plot of the confusion table (Fig. 7).

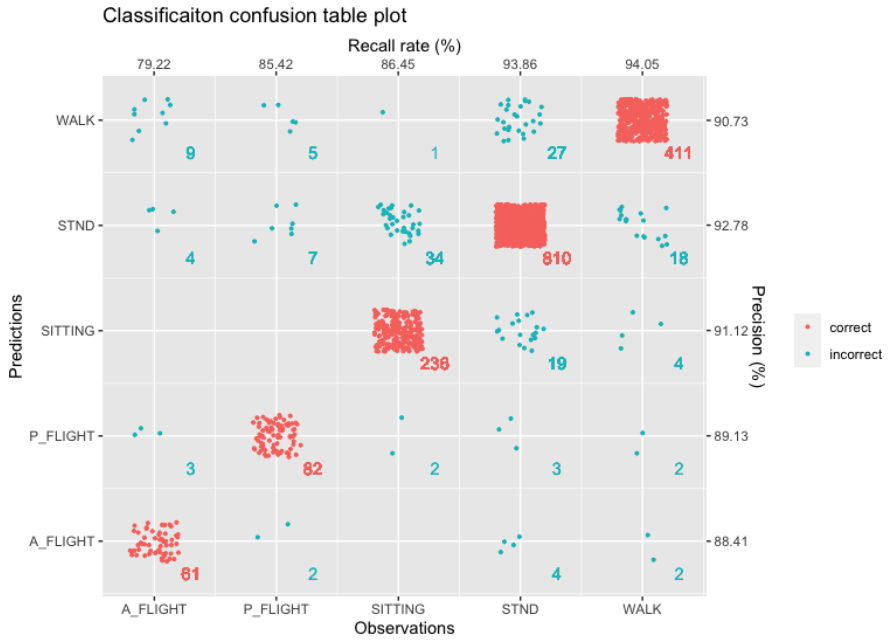


Figure 7. Confusion matrix plot of 5-fold cross-validation results. The dots in the graph are coloured according to the classification results, with blue and red symbols being correct and incorrect classifications, respectively. Sample size for each observation and prediction combination is provided.

2.7 Classification result check

Using the predictions from the behaviour classification model, we can now return to the original ACC data to evaluate which ACC signals lead to correct and incorrect classifications using function plot_confusion_matrix. This function basically uses the same digraph with near identical look to function plot_acc used earlier. The only deviation is that all correct and incorrect predictions (identified using the data frame from function plot_confusion_matrix) are now identified with a solid and a dotted line, respectively, and annotated with labels for the observed and predicted behaviours at the bottom and top of the graph respectively (Fig. 8).

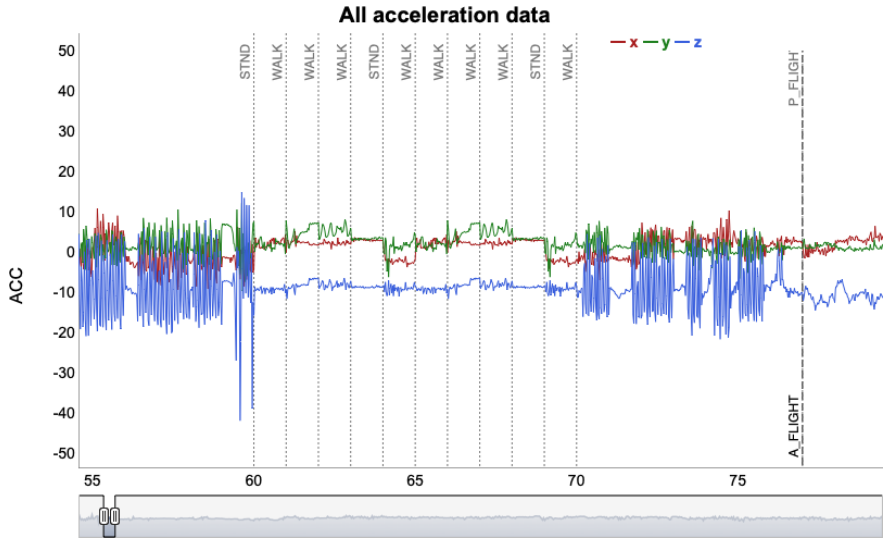


Figure 8. ACC data visualization including behaviour classification results using a dynamic graph. White stork ACC data is shown from segment 55 to 80 (cf Fig. 2b). Vertical black, dashed lines separate different observed behaviour types, while vertical grey dotted lines identify segments that have been incorrectly classified.

Conclusions

As demonstrated, the rabc package aims to assist researchers in developing good animal behaviour classification models in an interactive fashion. As human brains are extremely good at recognizing patterns, the visualization of data and results can greatly assist the workflow towards developing a behaviour classification model. Raw ACC data visualization assists in the detection of aberrant associated behaviour scores. Feature visualization helps researchers to understand how different features distribute across behaviours and whether the current behaviour set potentially needs adjustments, either by combining or by splitting behaviours up. Finally, classification-result visualization assists the understanding of misclassification patterns. Other than the visualization functionalities, this package provides complete functions to perform behaviour classification through XGboost, including: feature calculation, feature selection, model hyperparameter tuning, model training and validation and an output classifier for future ACC data classification.

Given its unique aim and functionality the rabc package will be a valuable addition to the growing array of R packages already available for behaviour and movement analyses (Joo et al., 2020). It is worth noting that the features calculated in this package can be further extended if deemed neccessary. Users can develop additional features and include these in the here described analyses and the ultimate generation of a behaviour classification model. Although we only use XGboost as the supervised machine learning model in this package, users can potentially use the output from the rabc package as input to the "caret" package. This will allow for the use of other machine learning models in generating behaviour classification models such as for example decision tree, support vector machine and random forest.

Data Availability Statement

The white stork dataset used in this paper is accessible from the online software AcceleRater website: http://accapp.move-ecol-minerva.huji.ac.il/. Upom acceptance, the dataset will be archived in Dryad.

Competing Interest Statement

The authors declare that there is no conflict of interest.

Author Contribution Section

Hui Yu: Conceptualization (equal); methodology (lead); software (lead); validation (lead); writing – original draft preparation (lead).**Marcel Klaassen:** Conceptualization (equal); methodology (supporting); software (supporting); writing – review & editing (lead).

Acknowledgements

We gratefully acknowledge Ran Nathan for sharing the dataset used in this study. We thank Batbayar Galtbalt and Tobias Alexander Ross for testing the software and providing feedbacks.

Reference

Boehmke, B., & Greenwell, B. (2019). *Hands-On Machine Learning with R* .

Bom, R. A., Bouten, W., Piersma, T., Oosterbeek, K., & van Gils, J. A. (2014). Optimizing acceleration-based ethograms: the use of variable-time versus fixed-time segmentation. *Movement Ecology, 2* (1), 6. doi:10.1186/2051-3933-2-6

Brown, D. D., Kays, R., Wikelski, M., Wilson, R., & Klimley, A. P. (2013). Observing the unwatchable through acceleration logging of animal behavior. *Animal Biotelemetry, 1* (1), 20. doi:10.1186/2050-3385-1-20

Chakravarty, P., Cozzi, G., Ozgul, A., & Aminian, K. (2019). A novel biomechanical approach for animal behaviour recognition using accelerometers. *Methods in Ecology and Evolution, 10* (6), 802-814. doi:10.1111/2041-210X.13172

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* , 785-794. doi:10.1145/2939672.2939785

Fehlmann, G., O'Riain, M. J., Hopkins, P. W., O'Sullivan, J., Holton, M. D., Shepard, E. L. C., & King, A. J. (2017). Identification of behaviours from accelerometer data in a wild social primate.*Animal Biotelemetry, 5* (1), 6. doi:10.1186/s40317-017-0121-3

Gilbert, N. I., Correia, R. A., Silva, J. P., Pacheco, C., Catry, I., Atkinson, P. W., . . . Franco, A. M. A. (2016). Are white storks addicted to junk food? Impacts of landfill use on the movement and behaviour of resident white storks (Ciconia ciconia) from a partially migratory population. *Movement Ecology, 4* (1), 7. doi:10.1186/s40462-016-0070-0

Joo, R., Boone, M. E., Clay, T. A., Patrick, S. C., Clusella-Trullas, S., & Basille, M. (2020). Navigating through the r packages for movement. *Journal of Animal Ecology, 89* (1), 248-267. doi:10.1111/1365-2656.13116

Kölzsch, A., Neefjes, M., Barkway, J., Müskens, G. J. D. M., van Langevelde, F., de Boer, W. F., . . . Nolet, B. A. (2016). Neckband or backpack? Differences in tag design and their effects on GPS/accelerometer tracking results in large waterbirds. *Animal Biotelemetry, 4* (1), 13. doi:10.1186/s40317-016-0104-9

Korpela, J., Suzuki, H., Matsumoto, S., Mizutani, Y., Samejima, M., Maekawa, T., . . . Yoda, K. (2020). Machine learning enables improved runtime and precision for bio-loggers on seabirds. *Commun Biol, 3* (1), 633. doi:10.1038/s42003-020-01356-8

Kroschel, M., Reineking, B. r., Werwie, F., Wildi, F., & Storch, I. (2017). Remote monitoring of vigilance behavior in large herbivores using acceleration data. *Anim Biotelemetry, 5* (10). doi:DOI 10.1186/s40317-017-0125-z

Ladds, M. A., Thompson, A. P., Kadar, J.-P., J Slip, D., P Hocking, D., & G Harcourt, R. (2017). Super machine learning: improving accuracy and reducing variance of behaviour classification from accelerometry.*Animal Biotelemetry, 5* (1), 8. doi:10.1186/s40317-017-0123-1

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software, 3* , 861. doi:10.21105/joss.00861

Nathan, R., Spiegel, O., Fortmann-Roe, S., Harel, R., Wikelski, M., & Getz, W. M. (2012). Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures. *Journal of Experimental Biology, 215* (6), 986-996. doi:10.1242/jeb.058602

Nuijten, R. J. M., Gerrits, T., Shamoun-Baranes, J., & Nolet, B. A. (2020). Less is more: On-board lossy compression of accelerometer data increases biologging capacity. *Journal of Animal Ecology, 89* (1), 237-247. doi:10.1111/1365-2656.13164

Resheff, Y. S., Rotics, S., Harel, R., Spiegel, O., & Nathan, R. (2014). AcceleRater: a web application for supervised learning of behavioral modes from acceleration measurements. *Movement Ecology, 2* (1), 27. doi:10.1186/s40462-014-0027-0

Ropert-Coudert, Y., & Wilson, R. P. (2005). Trends and perspectives in animal-attached remote sensing. *Frontiers in Ecology and the Environment, 3* (8), 437-444. doi:Doi 10.2307/3868660

Sainburg, T., Theilman, B., Thielk, M., & Gentner, T. Q. (2019). Parallels in the sequential organization of birdsong and human speech.*Nature Communications, 10* (1), 3636. doi:10.1038/s41467-019-11605-y

Shamoun-Baranes, J., Bom, R., van Loon, E. E., Ens, B. J., Oosterbeek, K., & Bouten, W. (2012). From sensor data to animal behaviour: an oystercatcher example. *Plos One, 7* (5), e37997. doi:10.1371/journal.pone.0037997

Shepard, E. L., Wilson, R. P., Quintana, F., Laich, A. G., Liebsch, N., Albareda, D. A., . . . Myers, A. E. (2008). Identification of animal movement patterns using tri-axial accelerometry. *Endangered Species Research, 10* , 47-60. doi:10.3354/esr00084

Toloşi, L., & Lengauer, T. (2011). Classification with correlated features: unreliability of feature ranking and solutions.*Bioinformatics, 27* (14), 1986-1994. doi:10.1093/bioinformatics/btr300

Williams, H. J., Taylor, L. A., Benhamou, S., Bijleveld, A. I., Clay, T. A., de Grissac, S., . . . Börger, L. (2019). Optimizing the use of biologgers for movement ecology research. *Journal of Animal Ecology, n/a* (n/a). doi:10.1111/1365-2656.13094

Wilson, R., Börger, L., Holton, M., Michael Scantlebury, D., Laich, A., Quintana, F., . . . Shepard, E. (2019). *Estimates for energy expenditure in free-living animals using acceleration proxies: A reappraisal* .