Movement Ecology
Prying into the intimate secrets of animal lives; software beyond hardware for comprehensive annotation in 'Daily Diary' tags.
--Manuscript Draft--

<table>
<thead>
<tr>
<th>Manuscript Number:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Title:</td>
<td>Prying into the intimate secrets of animal lives; software beyond hardware for comprehensive annotation in 'Daily Diary' tags.</td>
</tr>
<tr>
<td>Article Type:</td>
<td>Software article</td>
</tr>
<tr>
<td>Abstract:</td>
<td>Smart tags attached to freely-roaming animals recording multiple parameters at infra-second rates are becoming commonplace, and are transforming our understanding of the way wild animals operate. However, interpretation of such data is complex and currently limits the ability of biologists to realise the value of their recorded information. This work presents a single program, FRAMEWORK 4, that uses a particular sensor constellation described in the 'Daily Diary' tag (recording tri-axial acceleration, tri-axial magnetic field intensity, pressure and e.g. temperature and light intensity) to determine the 4 key elements considered pivotal within the conception of the tag. These are; animal trajectory, behaviour, energy expenditure and quantification of the environment in which the animal operates. The program takes the original data recorded by the Daily Diary and transforms it into dead-reckoned movements, template-matched behaviours, dynamic body acceleration-derived energetics and position-linked environmental data before outputting it all into a single file. Biologists are thus left with a single data set where animal actions and environmental conditions can be linked across time and space.</td>
</tr>
<tr>
<td>Corresponding Author:</td>
<td>James Stephen Walker, BSc, MRes</td>
</tr>
<tr>
<td></td>
<td>Swansea University</td>
</tr>
<tr>
<td></td>
<td>Swansea, UNITED KINGDOM</td>
</tr>
<tr>
<td>Corresponding Author Secondary Information:</td>
<td></td>
</tr>
<tr>
<td>Corresponding Author's Institution:</td>
<td>Swansea University</td>
</tr>
<tr>
<td>Corresponding Author's Secondary Institution:</td>
<td></td>
</tr>
<tr>
<td>First Author:</td>
<td>James Stephen Walker, BSc, MRes</td>
</tr>
<tr>
<td>First Author Secondary Information:</td>
<td></td>
</tr>
<tr>
<td>Order of Authors:</td>
<td>James Stephen Walker, BSc, MRes</td>
</tr>
<tr>
<td></td>
<td>Mark W Jones</td>
</tr>
<tr>
<td></td>
<td>Robert S Laramee</td>
</tr>
<tr>
<td></td>
<td>Mark D. Holton</td>
</tr>
<tr>
<td></td>
<td>Emily L.C. Shepard</td>
</tr>
<tr>
<td></td>
<td>Hannah J. Williams</td>
</tr>
<tr>
<td></td>
<td>Michael D. Scantlebury</td>
</tr>
<tr>
<td></td>
<td>Nikki J. Marks</td>
</tr>
<tr>
<td></td>
<td>Elizabeth A. Magowan</td>
</tr>
<tr>
<td></td>
<td>Iain E. Maguire</td>
</tr>
<tr>
<td></td>
<td>Ed Grundy</td>
</tr>
<tr>
<td></td>
<td>Agustina Di Virgilio</td>
</tr>
<tr>
<td></td>
<td>Rory P. Wilson</td>
</tr>
<tr>
<td>Order of Authors Secondary Information:</td>
<td></td>
</tr>
</tbody>
</table>

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation
Prying into the intimate secrets of animal lives; software beyond hardware for comprehensive annotation in ‘Daily Diary’ tags.

James S. Walker¹, Mark W. Jones¹, Robert S. Laramee¹, Mark D. Holton², Emily L. C. Shepard³, Hannah J. Williams³, D. Michael Scantlebury⁴, Nikki, J. Marks⁴, Elizabeth A. Magowan⁴, Iain E. Maguire⁴, Ed Grundy³, Agustina Di Virgilio⁵, Rory P. Wilson³.

¹Department of Computer Science, College of Science, Swansea University, Singleton Park, Swansea SA2 8PP, Wales, UK.
²College of Engineering, Swansea University, Singleton Park, Swansea SA2 8PP, Wales, UK.
³Swansea Lab for Animal Movement, Biosciences, College of Science, Swansea University, Singleton Park, Swansea SA2 8PP, Wales, UK.
⁴School of Biological Sciences, Institute for Global Food Security, Queen’s University Belfast, Belfast BT9 7BL, Northern Ireland, UK.
⁵Laboratorio Ecotono, INIBIOMA-CONICET, National University of Comahue, Quintral 1250, (8400) Bariloche, Argentina.

Abstract
Smart tags attached to freely-roaming animals recording multiple parameters at infra-second rates are becoming commonplace, and are transforming our understanding of the way wild animals operate. However, interpretation of such data is complex and currently limits the ability of biologists to realise the value of their recorded information. This work presents a single program, FRAMEWORK 4, that uses a particular sensor constellation described in the ‘Daily Diary’ tag (recording tri-axial acceleration, tri-axial magnetic field intensity, pressure and e.g. temperature and light intensity) to determine the 4 key elements considered pivotal within the conception of the tag. These are; animal trajectory, behaviour, energy expenditure and quantification of the environment in which the animal operates. The program takes the original data recorded by the Daily Diary and transforms it into dead-reckoned movements, template-matched behaviours, dynamic body acceleration-derived energetics and position-linked environmental data before outputting it all into a single file. Biologists are thus left with a single data set where animal actions and environmental conditions can be linked across time and space.

1. Introduction
The development of hardware that can be attached to animals during their everyday life (Ropert-Coudert and Wilson 2005) has revolutionized our understanding of the biology of wild animals. Indeed, this approach has allowed researchers to look at everything from the behaviour of whales chasing prey at depths of over 1 km underwater (Aguilar Soto, Johnson et al. 2008) to the physiology of geese migrating over the Himalayas (Hawkes, Balachandran et al. 2011). A common feature facilitating these sorts of projects has been the increase in numbers and types of sensors used in smart animal tags, as well as increases in the frequency with which they can be sampled and concomitant increases in memory capacity. Thus, our ability to answer critical questions in biology appears to have been driven to a large extent by advances in technology (Ropert-Coudert and Wilson 2005). These advances in methodology come under two broad areas. One relates to methods that allow tags to be physically attached to animals for increasing lengths of time (e.g. Wilson, Putz et al. 1997; Hooker, Baird et al.
2001; Rowley and Alford 2007) while minimizing animal detriment (Vandenabeele, Shepard et al. 2012; Vandenabeele, Wilson et al. 2013) while the other relates to the physical production of the complex solid-state units in smart tags (e.g. Miller, Johnson et al. 2004).

The excitement at the potential inherent in sophisticated animal tags has, however, been tempered by a new limiting factor. This is a methodology to deal with the problem of the analysis of the high resolution, multiple channel (and therefore multi-dimensional) data acquired by the tags – in short, software (cf. Jonsen, Myers et al. 2003). To be most useful to the community, software to help in the analysis of smart tag-acquired data needs to be able to deal with large quantities of multiple sensor data and, ideally, should be able to merge different derived analytical outputs together into one output file so that various elements derived from the primary data can be interrelated. Currently, the smart tag community has witnessed a number of software innovations, most of which are concerned with determination of behaviour, i.e. from dive profile data (Halsey, Bost et al. 2007) or, most notably, from tri-axial acceleration data (Watanabe, Izawa et al. 2005; Sakamoto, Sato et al. 2009). Analysis of acceleration data is particularly welcome because inspection of raw acceleration values over time to derive behaviours is not particularly intuitive (Shepard, Wilson et al. 2009). Thus, solutions for this have involved a suite of different approaches including cluster analysis (Sakamoto, Sato et al. 2009), support vector machine classification models (Martiskainen, Järvinen et al. 2009) and artificial neural networks (Nathan, Spiegel et al. 2012).

In 2008, Wilson et al. (2008) put forward a concept for a particular sensor configuration within a tag that they called the ‘Daily Diary’ (DD), where analysis yielded value well beyond the simple mathematical sum of its individual parts (Wilson 2008). Specifically, the suggestion advocated the combination of tri-axial accelerometers, tri-axial magnetometers, and pressure and speed transducers together with environmental sensors such as temperature, light and humidity. The theory was that this constellation of sensors, sampled at infra-second rates, would allow tag users to be able to derive four key elements of animal ecology seamlessly. These are: (1) animal trajectory, and therefore position (Wilson, Liebsch et al. 2007) (2) animal behaviour (Shepard, Wilson et al. 2008) (3) energy expenditure (Wilson, White et al. 2006) and (4) the environmental conditions to which the tag carrier is exposed (Wilson, Grémillet et al. 2002). Although this original work did indicate avenues by which these elements might be achieved, there was no specific suggestion of software that might actually do this. In short, currently, although some software is available to help determine some aspects of that advocated by the DD concept (e.g. Bidder, Campbell et al. 2014), there is nothing that binds the concepts together.

This paper describes the structure and functioning of a new software package (DD-FRAMEWORK 4) that allows DD (or similar) users to calculate all four key elements advocated by the DD and then to bind them together into one single output file so that workers can subsequently geo-reference behaviours, energy expenditures and environmental parameters to gain a more holistic picture of how animals react to and within their environment. Specifically, we introduce FRAMEWORK 4, a user-friendly turnkey solution for the analysis of DD data. Using our system, we can seamlessly obtain animal behaviour, animal trajectory, energy expenditure, and environmental conditions all within one application and export them in one merged data file. Our solution requires no knowledge of the underlying processes utilised in the software, or any advanced mathematical and computing skill sets. Our application has been produced with the end-user in mind, utilising wizards and graphical user interfaces wherever possible. We hope this software will assist in
the understanding of wild animal ecology, providing new insights as a result of advanced computing knowledge.

2. FRAMEWORK 4 - software implementation

The software implementation is outlined before specifying each of the software features and the underlying methods by which they operate.

Framework 4 is a desktop application for the Microsoft Windows operating system built using the programming language C++ with the Qt 4.8 framework library. This software has been implemented through a collaboration with the Bioscience and Computer Science departments at Swansea University and is maintained by the first author in this article. We selected to create a desktop application as it allows us to handle large data files directly. Alternatively, utilising a web application would allow access from anywhere on any platform with an active internet connection and a web browser, although this would entail long waiting times while data sets upload, resulting in a reduced ability to handle larger files. Utilising the desktop application, we directly communicate with the CPU for efficient data handling, and the GPU for visualisation purposes.

2.1 Loading Data

The software supports two file formats; comma separated values (CSV) and tabular delaminated formats. These are two of the standard file formats for storing tabular data in text format and are common outputs from commercial animal monitoring tags. The DD exports its data in a binary format which gets segmented into multiple files and converted to tabular delimitated format post-deployment. FRAMEWORK 4 loads and operates on the individual tabulated files.

We incorporate an import wizard in the application to import CSV and tabular delimited files. Here, the user can specify the names and data types of each attribute, which are used as a reference to them throughout the application.

3. Derivation of animal trajectory

3.1 Principles behind dead-reckoning

The way in which animals use the environment is fundamental to understanding their behavioural ecology (Börger et al. 2008) and, as such, many different systems have been developed to examine animal movements (see e.g. Wikelski et al. 2007). A relatively recent addition to the field is dead-reckoning (Wilson et al. 2008), which has particular value in not being dependent on transmission technology and being able to resolve movement at infra-second scales. Dead-reckoning operates on the basis that the position of an animal at any time ‘t’ can be derived knowing the position at the animal at a previous time ‘t-1’ and the distance and heading taken between the two time intervals. Dead-reckoning has received little interest until now, partially because early systems for dead-reckoning were crude (Ioale et al., 1994; Wilson and Wilson, 1988; Wilson et al., 1991) but with the development of sensors and techniques, headings can now be computed to within 1° utilising accelerometer and magnetometer sensors (Caruso, 2000; Wilson et al., 2008).

The earth’s magnetic field is constructed of field lines approximating a magnetic dipole. Each field line originates at a point near the magnetic South Pole and terminates at a point near the
magnetic North Pole. The relationship between magnetic north and geographic north is defined by a declination angle which varies across the earth’s surface and with time. This angle can be obtained from a reference table provided by the National Geophysical Data Center (http://www.ngdc.noaa.gov/geomag-web/). Measuring the strength and direction of the lines utilising the tri-axial magnetometer can be used to obtain a relative compass heading.

In Framework 4, we introduce a user-friendly wizard for performing dead-reckoning on data with tri-axial magnetometer and tri-axial accelerometer components (Fig. 1). Dead-reckoning is subject to cumulative errors and, as the heading and speed are estimates, any systematic deviations from the actual heading and speed will lead to increasing errors. To assist in reducing such errors, we incorporate a hybrid GPS and dead-reckoned approach which utilises the GPS fixes as the actual position and forces the dead-reckoned path to go through these. This resets any cumulative error but does not factor in any GPS errors. We now outline each of the components of the Dead-reckoning wizard which are modular so that the user can select which analysis steps are required. The steps are; (i) GPS import for synchronising GPS data with the DD data (ii) alignment correction for the accelerometer and magnetometer coordinate frames (iii) heading derivation from the magnetometer channels (iv) VeDBA for obtaining an estimate of the speed (v) constant speed options if necessary (vi) dead-reckoning to combine the heading and speed information to derive an animal trajectory and (vii) dead-reckoning correction using GPS points to eliminate drift in the final dead-reckoned path.

![DR Wizard](image)

**Figure 1 – The Dead-reckoning wizard features a number of modular classification steps. These are; GPS Import, Alignment, Heading, VeDBA, Constant Speed, Dead-reckoning, and Dead-reckoning with GPS Correction.**

### 3.2.1 GPS Import

FRAMEWORK 4 is set up to load data from separate GPS and DD files, although both are assumed time-synchronised. In this step, the user can import the GPS data and merge it back in to the complete data set. The same import wizard is used as in the file importer to import the GPS data. The user imports the data, then selects the data attributes for the available time fields in both the DD and GPS data sets (i.e. day, hours, minutes, and seconds). The wizard
matches the time index columns and inserts the GPS data fields wherever a matching data item with the same time stamp is found.

3.2.2 Alignment
Within the DD, the accelerometer and magnetometer coordinate systems are not necessarily aligned such that the three axes of the magnetometer and accelerometer in the same direction. Appreciation of their relative positions is, however, vital for computing the heading and this is applied using rotational information from the accelerometer and applying it to the magnetometer channels. To deal with this, the program asks the user to specify the orientation of the device. As the user selects different orientations, the image of the device is updated appropriately (Fig. 2). After the correct settings have been chosen, the sensors attributes are transposed so they align correctly with each other.

3.2.3 Heading
Deriving the heading necessitates that the data attributes be sanitised. This is done via; magnetometer calibration, pitch and roll computation, coordinate frame adjustments, and finally the heading derivation. These processes are executed in the background of the wizard and the user is exposed to certain parameters and results which allow them to select the correct attributes and associated settings for each stage in the wizard. For clarity we now detail the methods for each process.

3.2.3.1 Ellipse fitting
The measurements obtained by commercial magnetometer sensors are corrupted by several errors. Proper calibration of the magnetometer is required to obtain high accuracy heading measurements. Errors can are usually introduced by instrumental errors, such as scale factors, non-orthogonality between axes, offsets and magnetic deviations caused by perturbations and interference with the magnetic field lines.

Magnetic measurements are subject to sources of error primarily caused by hard iron and soft iron deposits acting on the magnetic field. Rotating a magnetometer around 360 degrees in all orientations under no sources of error should produce a perfect sphere centred on the origin. Hard iron effects are created by objects which produce a magnetic field with a constant bias in the output, resulting in a sphere displaced from the origin. Soft iron deposits are caused by ferrous materials which are more permissive to the magnetic field influencing the magnetic field as it passes through, via distortion or stretching. This distorts the sphere into an ellipse as hard iron errors are independent of the orientation of the device and can therefore occur across the sphere. Hard iron distortions are caused by metals such as nickel and iron and commonly have a much larger contribution towards the total error (Cai et al. 2001).
We utilise the state-of-the-art error model presented by Renaudin et al. (2010) consisting of
an ellipsoid-fitting algorithm based on an adaptive least squares estimator which calibrates
the magnetometer readings for both instrumental errors and magnetic deviations.

3.2.3.2 Pitch and Roll Computation
In order to determine the heading of a device affixed to an animal, the magnetometer should
ideally have an x y plane that is parallel to the earth’s surface, something that is not always
the case. Correction for this requires a correction for tilt and this can be achieved by using the
static acceleration by deriving a moving mean from each of the acceleration axes (Shepard et
al. 2008). For a sample $S_j$, with a smoothing window of $w$ this is given by:

$$S_t = \frac{1}{w} \sum_{j=i-w/2}^{i+w/2} S_j$$

Device pitch and roll are given by the arctangent of the relevant smoothed x, y, and z values.

$$Roll (\theta) = (\arctan2(S_x, \sqrt{S_y \cdot S_y + S_z \cdot S_z}) \cdot \frac{180}{\pi}$$

$$Pitch (\theta) = (\arctan2(S_y, \sqrt{S_x \cdot S_x + S_z \cdot S_z}) \cdot \frac{180}{\pi}$$

5.2.3.3 Coordinate frame adjustment
Once derived, device attitude via pitch and roll can be used to rotate magnetometer
measurements. Here, magnetometer x, y, z values are rotated by the inverse of the pitch ($\theta$)
and roll ($\phi$) to access the rotated column vector $m'$ using;

$$R_x (\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{pmatrix}$$

$$R_y (\phi) = \begin{pmatrix} \cos \phi & 0 & -\sin \phi \\ 0 & 1 & 0 \\ \sin \phi & 1 & \cos \phi \end{pmatrix}$$

$$m' = R_x(\theta)R_y(\phi) \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}$$

3.2.3.4 Heading derivation
The compass heading (H) with respect to magnetic north is determined using the x and y tilt-
and error-corrected magnetometer components utilising:

$$H = \left(\arctan2(m_y, -m_x)\right) \cdot \frac{180}{\pi}$$

3.3.4 Using Vectorial Dynamic Body Acceleration (VeDBA) for Speed
VeDBA can be used for predicting speed in terrestrial animals (Bidder et al. 2012). FRAMEWORK 4 calculates VeDBA using:

$$VeDBA = \sqrt{DA_x^2 + DA_y^2 + DA_z^2}$$

where $DA_x$, $DA_y$, and $DA_z$ are the dynamic acceleration values derived by taking the absolute values of running means of the raw acceleration values of each of the 3 orthogonal measurement axes from the corresponding raw acceleration values. The program allows users to select the length of the running mean although a default of 2 s is used (Shepard et al. 2009). Animal speed ($s$) can then be computed by applying a speed coefficient $m$ and adding a constant $c$ to the VeDBA value. The speed coefficient and offset is adjusted in the dead-reckoning wizard page. In addition, we utilise a threshold $t$ whereby, if the VeDBA falls below this value, the VeDBA is set to zero.

$$s = \begin{cases} 
(VeDBA \cdot m) + c & \text{if } s > t \\
0 & \text{else}
\end{cases}$$

3.3.5 Constant Speed
For volant species, the dynamic acceleration is unlikely to correspond to the speed of the animal, invalidating VeDBA in this context. Until a satisfactory measure of speed is derived (such as a pitot tube), we suggest using a constant speed, with options to correct this using the GPS data later.

3.3.6 Dead-reckoning
Dead-reckoning combines the speed and heading to compute a trajectory for the given data with calculations computed in Latitude and Longitude to compensate for the spherical nature of the earth. Utilising Cartesian coordinates for dead-reckoning introduces error by computing trajectories based on linear distances.

There are a number of parameters which must be defined first by the user. These are; (i) an initial start position defining the point where the path starts (if GPS data is given, the start coordinates are taken from this), (ii) a number of speed parameters for the VeDBA threshold and speed coefficients (see Bidder et al. 2012, this volume) and (iii) a heading offset corresponding to the declination angle obtained from the NGDC website. The computed path is shown alongside the map so that the user can interactively adjust parameters and see the result on the generated path.

To compute the path, the speed must be converted to radial distance in terms of the radius of the earth $R$ ($6.371 \times 10^6$ m). This is calculated as.

$$q = \frac{d}{R}$$

The Latitude and Longitude at time $T_i$ (where $T_0$ is equal to the starting point of the track) can be calculated as follows.

$$Lat_i = \arcsin(\sin(Lat_{i-1}) \cdot q + \cos(Lat_{i-1}) \cdot \sin \cdot \cos H)$$
The complete set of latitude and longitude points defines the trajectory of the animal movement.

3.3.7 Dead-reckoning with GPS Correction

The dead-reckoning solution can be subject to cumulative errors from the estimates of derived heading and speed. Even small, but systematic, errors in these channels will accumulate over time and can increase the resulting error correspondingly. To overcome these problems, the program utilises a dead-reckoning and GPS correction algorithm to correct the heading and speed of the obtained dead-reckoned trajectory using the aligned GPS trajectory information. This resets the accumulated error at each GPS fix.

We correct the heading and speed by adjusting the length and heading of the dead-reckoned path such that they align to the same positions along each fix along the GPS path. For this, we define two equations; the heading between two points and the distance.

The heading ($\theta$) between two points ($\text{Lat}_0$, $\text{Long}_0$) and($\text{Lat}_1$, $\text{Long}_1$) is calculated as so:

$$\text{TrajectoryHeading}(\theta) = \frac{\sin(\text{Lat}_1) - \sin(\text{Lat}_0) \cdot \cos(\text{Lat}_1)}{\cos(\text{Long}_1 - \text{Long}_0) \cdot \sin(\text{Long}_1 - \text{Long}_0) \cdot \cos(\text{Lat}_1)}$$

The distance ($d$) between two points ($\text{Lat}_0$, $\text{Long}_0$) and($\text{Lat}_1$, $\text{Long}_1$) is calculated as so:

$$\text{latitudeDistance} (\text{latD}) = \text{Lat}_1 - \text{Lat}_0$$

$$\text{longDistance} (\text{longD}) = \text{Long}_1 - \text{Long}_0$$

$$\text{arcDist} = \sin\left(\frac{\text{latD}}{2}\right) \cdot \sin\left(\frac{\text{longD}}{2}\right) + \sin\left(\frac{\text{longD}}{2}\right) \cdot \sin\left(\frac{\text{longD}}{2}\right) \cdot \cos(\text{Lat}_0) \cdot \cos(\text{Lat}_1)$$

$$\text{TrajectoryDistance}(d) = R \cdot 2 \cdot \text{atan2}(\sqrt{\text{arcDist}}, \sqrt{1 - \text{arcDist}})$$

The headings and speed between fixes can be adjusted to those of the GPS fixes in an iterative manner. For two sequential GPS fixes there are usually many dead-reckoned fixes in-between. Firstly the heading is adjusted, this consists of adding a constant heading to all the dead-reckoned headings between the GPS fixes.

$$\text{headingCoefficient}(hc) = (\text{gpsHeading} - \text{drHeading})$$

A speed coefficient to adjust the speed by is then computed. This consists of multiplying the speed values (derived from the VeDBA) between the GPS fixes.

$$\text{speedCoefficient}(sc) = \frac{\text{gpsDistance}}{\text{drDistance}}$$

The formulae for dead-reckoning are now adjusted to generate a dead-reckoned GPS-corrected path. The speed coefficient is multiplied by the original speed coefficient, along
with an addition of the heading offset, specific for that section of the track to counter the deviations from the GPS trajectory.

\[
\text{Lat}_i = \sin(\text{Lat}_{i-1}) \cdot \cos(q \cdot sC) + \cos(\text{Lat}_{i-1}) \cdot \sin q \cdot \cos(H + hC)
\]

\[
\text{Lon}_i = \text{Lon}_{i-1} + \arctan(\sin(H + hC) \cdot \sin(q \cdot sC) \cdot \cos(\text{Lat}_{i-1}) + (\cos(q \cdot sC) - \sin(\text{Lat}_{i-1}) \cdot \sin(\text{Lon}_{i}))
\]

The process is repeated by applying the GPS-corrected path in place of the dead-reckoned path utilising the previous formulae. Each iteration makes the path adhere more tightly to the GPS fixes as the heading and speeds are adjusted. We allow the user to repeat the adjustment process a set amount of times or continue until the speed and heading adjustments are under a specified threshold.

The particular advantage of a dead-reckoned track is that it can give very fine detail in the route of an animal and do so without reference to external ground-truthing sources, although confidence in the precision of the route will inevitably decrease with increasing time between ground-truthed points (Bidder et al. this volume). Nonetheless, the approach has particular value in being able allude to trajectories where conventional tracking methods are useless (Fig. 3).

![Dead-reckoned track of a European badger (Meles meles) in Northern Ireland leaving its sleeping quarters (red dashed circle) and moving through the underground sett to emerge at the entrance (yellow circle). The vertical axis representing depth is shown as the pressure difference between the surface and any time underground. The reconstruction assumes that animal speed is directly proportional to VeDBA (Bidder et al. 2012) underground in the same way it is on the surface. If this is not the case, the derived distances will be affected accordingly.](image)

4. Derivation of animal behaviour

4.1 Problems with traditional Machine Learning Techniques
Extracting animal behaviour from the raw DD data is a time-consuming and cognitively demanding process for human analysis. Machine learning classification processes can be applied to identify and label behaviour by executing algorithms which learn from data to discover previously unknown properties (Esling et al. 2012) The learning aspect is typically split into two categories, supervised, and unsupervised learning. Supervised learning algorithms are trained on labelled data to generalize relationships between input and output samples. Conversely, unsupervised learning algorithms operate on unlabelled data to find previously unknown structure, and domain knowledge can then be applied to match found structure to a behaviour.

Supervised techniques build a model from labelled data which generates predictions in response to new data. Traditionally, K-nearest neighbour (K-NN), support vector machines (SVM), and random forests have all been applied to accelerometer data. K-NN is a ‘lazy’ learning method which predicts class membership based on the k closest training examples in the feature space (e.g. Bidder et al. 2014). The SVM algorithm finds a hyperplane which separates the feature space into the classes defined in the training set. Unseen data is assigned to a class based on the hyperplane region under which it falls (e.g. Gao et al. 2013). Random forests are the current state-of-the-art classification method in the data-mining community (e.g. Ellis et al 2014). Random forests construct many decision trees, each modelling the training set, with each tree voting for the resulting classification. A data item is assigned to the class with the most votes.

Cluster analysis is an unsupervised learning technique for exploratory analysis to find natural groupings of data. The most common clustering method used is K-Means. Data is partitioned into k clusters based on the distance to the nearest mean. The disadvantage of k-means is the need to specify how many partitions are in the data prior to clustering. Hierarchical clustering does not require any input parameters, instead grouping data over a variety of different scales to form a cluster tree, often visualized as a dendrogram. The user can choose the level most appropriate to their needs. However, this method is computationally expensive (order N^2).

Supervised learning algorithms require extensively labelled instances of behaviours. Obtaining this data is time consuming, requires domain expertise, and the undertaking of field studies to gather video-synchronised data. It is obviously not possible to obtain all such data in all cases due to environmental constraints. Secondly, the choice of algorithm and parameters provides its own class of problems. Typically, in this process, the data dimensionality is reduced to a few parameters which contain the relevant information to perform classification; feature extraction. Good classification results rely heavily on the features chosen, however, extracting a desirable feature set is considered more of an art than a science and takes a great amount of skill along with trial and error (Smith 1997). Once the data is classified, if the accuracy is less than desired, decisions must be made as to whether it is useful to invest more time creating additional training input, modify the parameters, or use a different learning algorithm. It is not obvious what the next best step to take is without sufficient knowledge of the underlying algorithms.

4.2 FRAMEWORK 4 approach to behavioural identification.

Unlike the previous methods, our method for identifying animal behaviour can use just one positive example and does not require large extensively labelled collections of behaviours (Walker et al. 2015 – Submitted to The Euro Graphics Conference on Visualization). Complicated parameters are avoided by utilising interactive visual interfaces which draw from the domain expertise when selecting matching behaviours in the input signal. A
feedback loop is incorporated such that the precision and recall can be boosted by applying
the user’s subject knowledge. These features overcome the disadvantages of machine
learning and provide a working solution that can cope with large complex data sets, a vital
element given the rates at which DDs record (e.g. 10 channels, each at 40 Hz). The result is a
system which supports the manual labelling of animal behaviour complimented with a user-
driven approach for the semi-automatic classification of animal behaviour, requiring one
instance of behaviour for the matching process to take place.

The user interface is split into three components (see Fig. 4 for overview). This consists of
the data view at the top, being composed of the data in a stacked time-series graph format.
Coloured segments overlaid on the graph indicate classified animal behaviour. A search panel
is located in the bottom left, within which the user can perform searches on the data utilizing
the template search wizard. Results are shown in this panel for the user to test, reject or
accept results before moving to the appropriate classification in the bottom right panel, where
the classification widget is situated. Classified behaviours are shown to the user in this
tabulated panel. Each tab represents a separate behaviour and contains visualizations of the
 corresponding set of classified instances. The colours assigned to each tab correspond to
those overlaid transparently on the time-series graph.

The user interface is split into three components (see Fig. 4 for overview). This consists of
the data view at the top, being composed of the data in a stacked time-series graph format.
Coloured segments overlaid on the graph indicate classified animal behaviour. A search panel
is located in the bottom left, within which the user can perform searches on the data utilizing
the template search wizard. Results are shown in this panel for the user to test, reject or
accept results before moving to the appropriate classification in the bottom right panel, where
the classification widget is situated. Classified behaviours are shown to the user in this
tabulated panel. Each tab represents a separate behaviour and contains visualizations of the
 corresponding set of classified instances. The colours assigned to each tab correspond to
those overlaid transparently on the time-series graph.

Figure 4 – An image of an overview of the user interface provided in FRAMEWORK4. In the
top there is a stacked time-series graph with labelled behaviours overlaid as transparent
windows. The search widget (bottom left), allows the user to search for behaviours and filter
through found instances by accepting and rejecting. Accepted results are moved into the
classification widget (bottom right). This view encapsulates the behaviour groups and associated classified instances.

The components of our system and each of the processes towards classifying animal behaviour are split into five steps; (1) the user must select a behaviour to find in the data set, (2) matching is performed to find the similarity across the series, (3) A classification wizard allows the user to apply their knowledge to extract matches, (4) Extracted behaviours are presented to the user, (5) The user can improve the accuracy of their results by applying a feedback loop, (6) The classified results are displayed.

4.2.1 Behaviour Selection

The first step in our system is for the user to select a behaviour to classify in the data. There are two methods for this in the application. Firstly, query-by-example, and secondly selecting previously saved behaviour instances from the template database.

Query-by-example allows the direct selection of animal behaviour by drawing a window across the time-series encapsulating a subset of data exemplifying the behaviour the user wishes to search for. After selection, a dialog is then displayed where the user can select the data attributes to utilize for the template. Any data attribute can be used for searching throughout the system (not just the accelerometer component). For example, the magnetometer attribute is useful for finding thermalling cycles in condors, while the pressure component can indicate diving cycles in aquatic species.

Behaviour templates used in the system are stored in a database back-end for future use. This is particularly useful because assigned behaviours can be used to search other files. The database can be set up to deal with behaviour templates assigned to classes of animals. The user can query for all patterns present for a specific animal or select an existing behaviour template previously saved in the database by navigating to the animal of interest and then selecting the appropriate behaviour template.

The signal may be resampled to capture events at different frequencies as some behaviours occur at different speeds, for example running. To capture these events independently of the time duration, we can store and search for the signal at different time-intervals using resampling. Re-sampling is used by specifying an irrational factor consisting of an interpolation factor (amount to up-sample by) and a decimation factor (amount to down-sample by) e.g. resampling by 1/2 will half the sampling rate, while resampling by 2/1 will double the duration.

4.2.2 Template Matching

Supervised machine learning techniques rely on large bodies of labelled data. Such extensively labelled data sets do not exist in our application area. An alternative solution for classification is to use template matching, a process for determining the presence of a known waveform in a larger dataset. In essence, this works by sliding the specified template across the data set, computing the similarity of the template at each position in the data series corresponding to the concordance in fit between template and the sample at the position. The result is a similarity value at each position in the time-series. This allows the user to select a single positive example of a behaviour and search for all occurrences of it in the data.

A distance measure is used to determine a quantitative value corresponding to similarity or dissimilarity between time-series. Correlation is the optimal technique for detecting a known
waveform in random noise (Smith 1997). In signal processing, it is well known that correlation has a linear complexity frequency space. We utilise correlation and a new fast normalized cross-correlation method for template matching in order to obtain results within real time (seconds) which maintains an interactive implementation. Standard cross-correlation performs matching, taking into account amplitude information, while normalized cross-correlation normalizes the template signal and current area under the template such that amplitude shifts are not taken into account. This is important when performing matching over regions where the represented waveform may be present at different orientations. Resampling the signal allows us to introduce time axis distortion to extract behaviours occurring at different durations.

4.2.3 Classification Wizard

Once the template matching algorithm has been executed, the user is presented with the pattern-matching results in the classification wizard. The classification wizard is used to guide the user through refining a similarity threshold to verify matched signals. The user interactively modifies the threshold value which corresponds to the similarity of extracted matches. Matches are depicted to the user in an intuitive format with interaction to modify the result set according to the user’s domain knowledge.

The classification wizard (Figure 5) is used to find all instances of the specified behaviour in the data series. The aim is to then obtain an appropriate threshold value through the interaction and inspection of visualization which maximizes the number of instances found, while minimizing the number of misclassifications. The similarity threshold is represented as a percentage of the match, with one hundred percent similarity representing an exact match, while zero represents no matching features. The user needs to find an appropriate estimate value using their expert knowledge of behavioural patterns and their occurrences in the data set.

The classification wizard features two views. On the left (Figure 5 (b)) are visualisations to show where matches occur in the data series, while on the right (Figure 5 (c)), all of the extracted matches are overlaid on top of each other to show the variance between matches. The visualisations are updated as the threshold value is refined by adjusting a slider corresponding to the threshold percentage (Figure 5 (a)).
Figure 5 - This figure shows the classification wizard. (a) Illustrates our wizard parameters for dynamically adjusting the threshold. (b) Shows our density based visualizations to gain an understanding of where matches occur in the data series. (c) Shows our overlaid signals visualization of all the extracted matches in a stacked graph format, with one graph for each attribute of the template. The template signal is overlaid in red to show a direct comparison. A yellow to blue color scheme is used, yellow representing low similarity matches, while blue encodes high similarity matches.

The positions where matches occur in the series are depicted using three graphical views (Figure 5 (b)), all of which are aligned with a time-series graph of the data series (Figure 5 (b1)). The confidence of a match visualization (Figure 5 (b2)) depicts a heatmap showing an overview of the pattern matching results to encode where high (blue) and low (yellow) similarity matches occur in the data series. The extracted matches view (Figure 5 (b3)) depicts where the extracted matches occur in the series and is updated as the similarity threshold is adjusted. Finally, a distribution of extracted matches (Figure 5 (b4)) which utilises a histogram, shows the number of matches at each position. The user may refine the result set by rejecting results by clipping rectangular regions of matching results from the data series to reject. This allows the user to reject results based on their knowledge of where they expect results to be present in the data series and the temporal trends expected.

All of the extracted matched signals are overlaid in a stacked time-series graph format, one graph for each data attribute of the pattern (Figure 5 (c)). The user can gain an overview of the general shape of the extracted signals from the graphs. This allows the verification of the shape of extracted matches as most outliers stand out immediately, not fitting into the general shape of the extracted results. The same colour-encoding scheme as the position of matches is
used to encode the strength of a match. As the user adjusts the threshold, results are added or removed. The user can directly see the cause and effect of modifying the threshold on the general shape of matched signals in comparison to the template signal overlaid in red. Results can be rejected in this view by manual selection of lines on the time-series graphs. All results falling within the selection are removed from the result set. This allows the user to filter results that should not be present manually. The user continues adjusting the similarity threshold and rejecting results in the synchronised graphical views until they are satisfied with the results being extracted. The user clicks ‘finish’ and the wizard closes.

4.2.4 Results
After the user has concluded with a good threshold value, the results are extracted and added to the results widget in the bottom left of Figure 5. The user can further inspect the results using our two views. Firstly the sparkline (embedded time-series) display, this puts the classifications in a table format, with each row corresponding to an identified instance of a behaviour visualized using a sparkline. The user can accept or reject results by selecting rows. Secondly, the overlaid plot view overlays the classified instances in a time-series graph. The user can accept or reject results by selection on the time-series. The overlaid plot view is useful where the signals shape is similar amongst results. Conversely, the sparkline display is useful where the behaviour signal varies. Matches displayed in the results view are also shown in the data view overlaid on top of the time-series graph in grey.

4.2.5 Improving Precision and Recall
The variability and inconsistency of animal behaviour makes the automatic labelling of behaviour a challenging task. It is widely accepted in the machine learning community that achieving 100% precision and recall is a difficult, if not impossible, task. From a movement ecology viewpoint, we aspire to close to perfect labelling of behaviour. We incorporate a feedback loop which draws from domain expertise to enhance results. Firstly, the user can provide secondary examples of a behaviour to find more behaviour instances. Secondly, the user can directly manipulate the result set to accept and reject matches. Finally, the user can manually classify behaviour.

Where the user believes the number of found instances to be low, boosting can be used to retrieve more instances. More examples of a behaviour are utilised in the template for searching. This, in effect, widens the search span to find patterns related to the secondary retrieved patterns but may not be directly related to the initial search pattern.

The results panel provides an effective means to inspect the newly found behaviour classifications. Results are accepted by moving them to an appropriate classification tab in the classification widget, or rejected by clicking the reject button. The reject button removes the result from the panel. The user should aim to keep accepting / rejecting results until this panel is empty.

We appreciate that some instances will never be identified by machine learning and may only be able to be extracted by the domain expert, be that because of a low number of instances of the behaviour, or because of the variability of the animal behaviour. We enable manual labelling in our system so that the user can manually select and classify behaviour regions. To classify a behaviour region in the data manually, the user cuts the time-series graph up into segments. Each cut contains a start and end boundary defined by that of a behaviour instance. Once a behaviour region is cut in the time-series graph, the user drags and drops the time-
series segment into the appropriate classification tab. The cutting and dragging of data samples is similar to that used in video-editing software.

4.2.6 Classified Results
Classified behaviours are shown in two views. Firstly, the classification widget which displays classified behaviour in a corresponding tabulated view. Secondly, classified instances are aligned and overlaid on top of time-series graphs as coloured rectangular regions identifying where in the data a match for the behaviour has occurred. Each behaviour is identified by a unique colour assigned to each classification tab in the classification widget.

A typical output of this process is that, not only can animal behaviour be classified with respect to time, but that the occurrence of different behaviours can be represented on GPS-enabled dead-reckoned animal tracks in an obvious colour scheme (Fig. 6).

![Fig 6. The dead-reckoned trajectory of a cow (Bos taurus) in a field in Northern Ireland over 2 h, colour-coded according to different activities — green = grazing, black = walking, red = lying down.](image)

5. Derivation of animal energy expenditure

5.1 Proxies for energy expenditure.
Since the suggestion by Wilson et al. (2006) that dynamic body acceleration could be used as a proxy for VO$_2$, there have been a number of studies that have confirmed its utility in species ranging from shellfish, through fish, amphibia and reptiles to birds and mammals (see Halsey et al. 2011 for review). Two measures have been used, Overall Dynamic Body Acceleration (ODBA) and Vectorial Dynamic Body Acceleration (VeDBA), which are essentially equivalent in terms of their power to predict VO$_2$ (Qasem et al. 2012) although VeDBA has more utility for predicting speed (Bidder et al. 2012). FRAMEWORK 4 uses VeDBA (see section 3.3.4 for calculation) as a proxy for VO$_2$ (Halsey et al. 2009) and so that plots of animal trajectory can be colour coded accordingly (Fig. 7).
Fig. 7 – The dead-reckoned trajectory of a sheep (Ovis aries) in Argentina over 14 h showing how VeDBA, a metric that correlates linearly with metabolic rate, (ranging from pale green [low values] through yellow to red [high values]) varies with location and track tortuosity. Note how higher track tortuosity is generally linked to lower VeDBA.

6. Derivation of the physical characteristics of the environment

Many animals modulate their behaviour in the environment according to its physical characteristics. For instance, reptiles may associate with areas of high temperature or insolation to warm up (Chelazzi and Calzolai 1986) while many bird species are limited in their foraging capacities by light (e.g. Wanless et al. 1999). Thus, the ability to resolve the geographic position of animals in tandem with environmental variables can help explain the incidence or emergence of particular behaviours (e.g. Wilson et al. 1993). Thus, the particular value of FRAMEWORK4 is that it allows the physical attributes of the environment to be allocated to space (and time) and to be associated with identified behaviours (Fig. 8).
7. Merged file output
The derived analytical attributes from the software can be outputted together into one data file. Exporting data is supported via navigation to the ‘Export’ option in the ‘File’ menu on the main tool bar. Export is undertaken in CSV format where derived attributes are appended as an additional column in the data file alongside the existing data channels. Each behaviour is assigned a unique numerical value where, if a data item falls within a labelled region, it is assigned this value.

8. Obtaining Software
The software is freely available for download from the following web address. The website features instructional videos and documentation on using the software. As the website evolves there will become more documentation and features available. We hope this will be the foundation of a variety of software techniques for animal movement analysis.

http://www.framework4.co.uk

We ask if you utilise the software in a publication that you reference this paper appropriately.

9. Conclusion
This is a first attempt to create a single program that explicitly links space use, movement, behaviour, and energy expenditure in free-living animals together with environmental conditions, doing so using an accessible column-separated format for ASCII-type data. Although there is appreciable room for improvement in many facets of the program at the moment, the aspiration is to progress and refine it to make it as powerful as possible and thereby provide a methodology which will enhance our understanding of the processes that affect the way that animals move within their environment.

Acknowledgements
JSW was funded by an EPSRC doctoral training grant. IEM and EAM were funded by PhD studentships from the Department of Employment and Learning (DEL) and the Department of Agriculture and Rural Development (DARD), Northern Ireland, respectively. HW benefitted from a Swansea University PhD studentship. The Andean condor work was conducted in collaboration with Sergio Lambertucci, it was funded by National Geographic (grant GEFNE89-13 awarded to E Shepard), CONICET (grants PIP-0095, and ANPCyT–PICT-1156 awarded to S Lambertucci) and Swansea University, and data were collected with the help of a large field team.

Declarations – Competing Interests
The author(s) declare(s) that they have no competing interests.

References
Ellis, K, Kerr, J et al. (2014). “A random forest classifier for the prediction of energy expenditure and type of physical activity from wrist and hip accelerometers.”. Physiological measurement, 35(11) 2191.2


