Received: 9 September 2025 | Accepted: 10 April 2026

DOI: 10.1111/2041-210x.70320

Methods in Ecology and Evolution

APPLICATION

`sampley`: A Python package for sampling visual survey data

Jonathan Syme[1] | Daniel E. Pendleton[1,2] | Erin L. Meyer-Gutbrod[3] | Benjamin Tupper[1] | Nicholas R. Record[1]

[1]Tandy Center for Ocean Forecasting, Bigelow Laboratory for Ocean Sciences, East Boothbay, Maine, USA

[2]NOAA Northeast Fisheries Science Center, Woods Hole, Massachusetts, USA

[3]School of the Earth, Ocean and Environment, University of South Carolina, Columbia, South Carolina, USA

Correspondence
Jonathan Syme
Email: jonathan_syme@outlook.com

Funding information
National Science Foundation, Grant/Award Number: #2307754

Handling Editor: Laura Jane Graham

Abstract

1. Visual surveys are a common method of obtaining data for ecological studies, including studies that develop models (e.g. species distribution models). In the case of transect or non-transect line surveys, data must often be partitioned into spatiotemporal units (i.e. samples) before being modelled. This processing typically follows one of three approaches —the grid, segment and point approaches— each with its own variations.
2. Currently, processing of visual survey data is done in custom scripts that can be time-consuming and technically demanding to develop, which hinders the development of ecological models. This issue is exacerbated if multiple approaches and/or variations are to be applied.
3. Here, we present `sampley`, a user-friendly Python package for processing visual survey data into samples. It operates on various common formats and filetypes, ensuring its applicability to diverse datasets and compatibility with other software. It can process data by multiple variations of the grid, segment and point approaches. As this processing follows a straightforward sequence of stages and involves a limited number of functions, it is easy to learn and use. The combination of these traits will increase efficiency and reproducibility of survey data processing and allow researchers to readily trial different methods.
4. In this paper, we provide a description of the package — the types of data that can be processed, the approaches and variations available and the stages involved in processing. An example application, based on a mock dataset consisting of survey tracks, sightings and in situ environmental data, is also provided with descriptions of each stage.

KEYWORDS
data processing, ecological models, grids, python package, samples, segments, visual survey data

1 | INTRODUCTION

Ecological studies rely on various methods for obtaining data on the occurrence of species. Visual surveys are one common method, having been applied to species great and small in a variety of habitats, from small rainforest birds (Anderson et al., 2015) to the largest marine predators (Derville et al., 2022). In many cases, surveys follow the same basic pattern: observers follow a route with species observations and sometimes environmental data, recorded intermittently.

Data obtained from such surveys can be analysed in numerous ways, including through the use of models, such as species distribution models (Elith & Franklin, 2013; Pasanisi et al., 2024). However, if data from visual surveys are to be modelled, they typically must first be partitioned into distinct spatiotemporal units that can serve as samples in a model (i.e. sampled). For some surveys, samples are predetermined, for example: counts of species within a 1×1 m quadrat in a coastal salt marsh (Baumberger et al., 2012) or along a transect of predetermined length and width on beams of submerged infrastructure (Meyer-Gutbrod et al., 2019). In other studies, however, when surveys are conducted throughout a given area, either along transects or at random, sampling units may be delimited afterwards by dividing survey effort and detections spatially and temporally. In general, this division is achieved by one of three approaches —referred to here as the grid, segment and point approach— each with numerous variations.

The grid approach consists of overlaying a two-dimensional grid, typically rectangular or hexagonal, onto the study area and allocating detections and, optionally, survey effort to the cells that they lie within (Birch et al., 2007) (Figure 1b). Each cell, often within a given temporal period, then serves as a sample. This approach has been used, for example, to quantify the habitat of krill in the Southern Ocean (Merkel et al., 2023) or to investigate the habitat use of coastal dolphins (Giralt Paradell et al., 2019; Syme et al., 2023).

The segment approach involves cutting survey tracks into segments of standardised lengths, which serve as samples (e.g. Becker et al., 2010; Roberts et al., 2016) (Figure 1c). This approach is often used for studies involving distance sampling and density models, where the surveyed area consists of a strip centred along each segment with the width based on a detection function (Stepanuk et al., 2023).

The point approach consists of using the detections as presences and sampling absences from areas that were surveyed but where the species was not detected (e.g. Derville et al., 2016; Rayment et al., 2014; Torres et al., 2008) (Figure 1d). This approach has been used, for example, by Derville et al. (2016) when modelling the habitat distribution of the Māui dolphin (*Cephalorhynchus hectori*). It should be noted that absences differ from background points, which are generated without accounting for detections and which represent general conditions found across the study area (Fernandez et al., 2022; Guillera-Arroita et al., 2015; Phillips et al., 2009).

Given a dataset of survey tracks and observations, samples could be derived via numerous variations of the grid, segment and point approaches with a range of values for key parameters that determine, for example, the scale of the analysis (e.g. cell size and segment length). However, processing data with any given combination of approach, variation and/or parameters requires considerable time, effort and technical skill — demands that only increase if multiple combinations are implemented. Indeed, in modelling studies, more time may be spent on data preparation than on the modelling itself. As a result, studies often implement a single approach with limited exploration of potential variations and parameters (e.g. Pendleton et al., 2020; Syme et al., 2023) even though they may benefit from implementing multiple combinations to determine which is most suitable for the dataset, model and research question. Moreover, researchers tend to rely on their own unpublished custom scripts that are, at best, described cursorily within manuscripts (e.g. Pendleton et al., 2020; Syme et al., 2023), which reduces the reproducibility of studies. These issues could be mitigated with a publicly available package that provides researchers with the functionality to efficiently implement a variety of sampling procedures.

Here, we present, `sampley` (Syme et al., 2025), a Python package that provides a user-friendly way of sampling data obtained from visual transect or non-transect line surveys. As `sampley` follows a simple sequence of steps, it is easy to implement and so will reduce the burden placed on ecologists by the time-consuming process of developing custom scripts for processing survey data. Moreover, it can be implemented repeatedly on the same dataset, allowing the user to make an informed choice of sampling procedure by efficiently applying and comparing numerous combinations of approach, variation and/or parameters — an otherwise onerous task if the user were to develop custom scripts. Finally, as `sampley` follows standardised processes and is publicly available, its implementation can be readily replicated, leading to improved reproducibility. To the best of our knowledge, no equivalent package exists and, by addressing this shortcoming, `sampley` will facilitate and improve the processing of visual survey data and, in turn, the development of ecological models.

2 | DESCRIPTION

The `sampley` package was developed in Python (version 3.12.0: Python Software Foundation, 2023) and relies primarily on the `shapely` (Gillies et al., 2024) and `geopandas` packages (Van den Bossche et al., 2024) that facilitate the manipulation of geospatial data. It is freely available from the Python Package Index where its project page (https://pypi.org/project/sampley/) contains guidance on installation and running as well as links to source code, documentation and the User Manual. As ecologists may be unfamiliar with Python, `sampley` has been designed to be easy to use for those with basic coding knowledge (e.g. knowledge of R) and the project page also contains links to exemplar scripts that provide users with a template for their own data processing. The User Manual also contains guidance, exemplars and links to official documentation concerning installing and running Python and Jupyter. Below, we

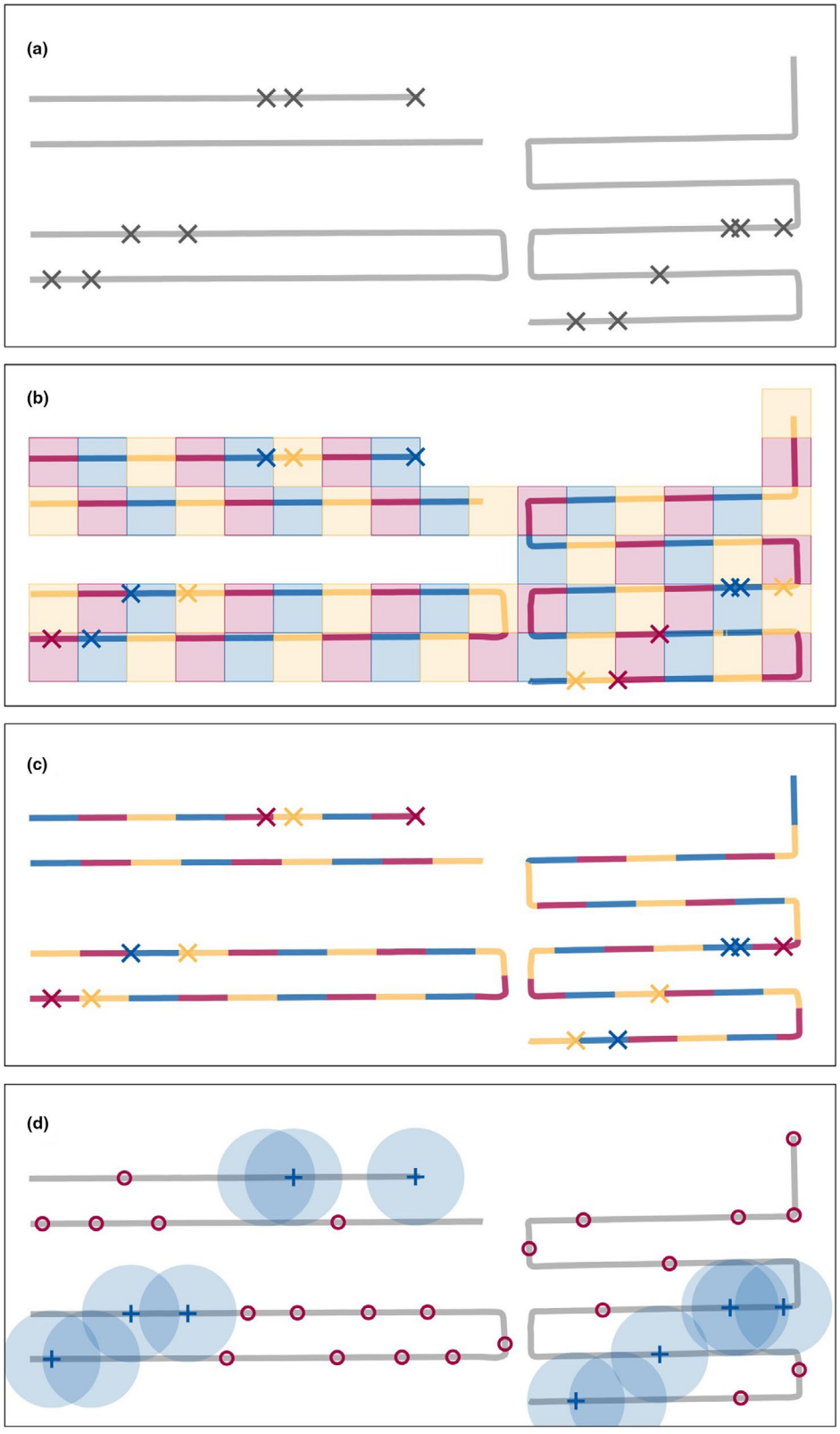

FIGURE 1 Schematic illustration of (a) survey tracks (lines) and sightings (crosses) that have been processed with the (b) grid, (c) segment and (d) point approaches. In the grid approach (b), cells (red, blue and yellow squares) are overlayed and survey tracks and sightings are allocated to the cell that they lie within (illustrated by matching red, blue and yellow). In the segment approach (c), survey tracks are divided into segments (red, blue and yellow lines) to which sightings are allocated (illustrated by matching red, blue and yellow). In the point approach (d), sightings become presences (blue pluses) while absences (red circles) are generated based on survey coverage (grey lines) provided that they are not within presence zones (pale blue circles). Note that some sightings are not contained in the presences as they were removed during spatial thinning, but are nonetheless represented by presence zones.

provide a conceptual description of the package's core functionality but refrain from delving into technical details as these can be found in the User Manual.

2.1 | Stage 1: Importing data

Data processing with `sampley` can be divided into three main stages, the first of which consists of importing data (Figure 2). Broadly speaking, `sampley` requires two types of data: survey tracks and datapoints. Survey tracks simply represent the route taken by the observer(s) and must be in *sections* — stretches of continuous survey effort. It is up to the user to define and delimit sections before inputting data to `sampley`, with the key requirement that each section is continuous. For example, portions of track are often removed (e.g. due to poor weather conditions or observers being off-watch), leaving a series of pieces of track with gaps in between. Consequently, the track is not continuous (as it has gaps), but each of the pieces of track is continuous and so can be a section. Further detail on defining and designating sections can be found in the User Manual. Datapoints represent recordings made during the survey, for example, sightings or in situ measurements of environmental variables. They must be 'on-line' (i.e. on the survey track), with the exception of sightings, which may be 'off-line' (i.e. located some distance from the survey track) if, for example, their location is calculated for distance sampling analysis.

Data can be imported to `sampley` from the following filetypes: GeoPackages (GPKG), Esri shapefiles, comma-separated values files (CSV) and Excel files. Survey tracks can be formatted as linestrings or as trackpoints (i.e. a series of points along a survey track). In the latter case, to ensure that trackpoints are correctly joined to form lines, the trackpoints must have a column of identifiers that detail the section of survey track to which each trackpoint belongs.

Datapoints can be either *continuous* or *sporadic*: continuous datapoints are recorded at frequent, regular intervals (e.g. points recorded by a GPS every 10 s) and can be used to reconstruct a survey track while sporadic datapoints are recorded at infrequent and/or irregular intervals (e.g. sightings or environmental data recorded at the beginning of transects). Survey tracks and datapoints may be contained in separate files or within a single file.

Once imported, the survey tracks and datapoints form the basis of all subsequent processing, yet they remain unchanged. Thus, multiple approaches, variations and/or parameters can be applied successively to the same set of survey tracks and datapoints.

2.2 | Stage 2: Delimiting

The second stage consists of delimiting one or more of the following, depending on the approach(es) used: temporal periods, grid cells, segments, presences and absences (Figure 2). The resulting objects, referred to collectively as *delimiters*, set out various aspects of the samples (e.g. their spatial and temporal extents). There is no limit to the number of delimiters that can be made, thus, the user may implement different approaches, variations and/or parameters. Here, the three approaches require different processes, although they have been standardised to facilitate their application.

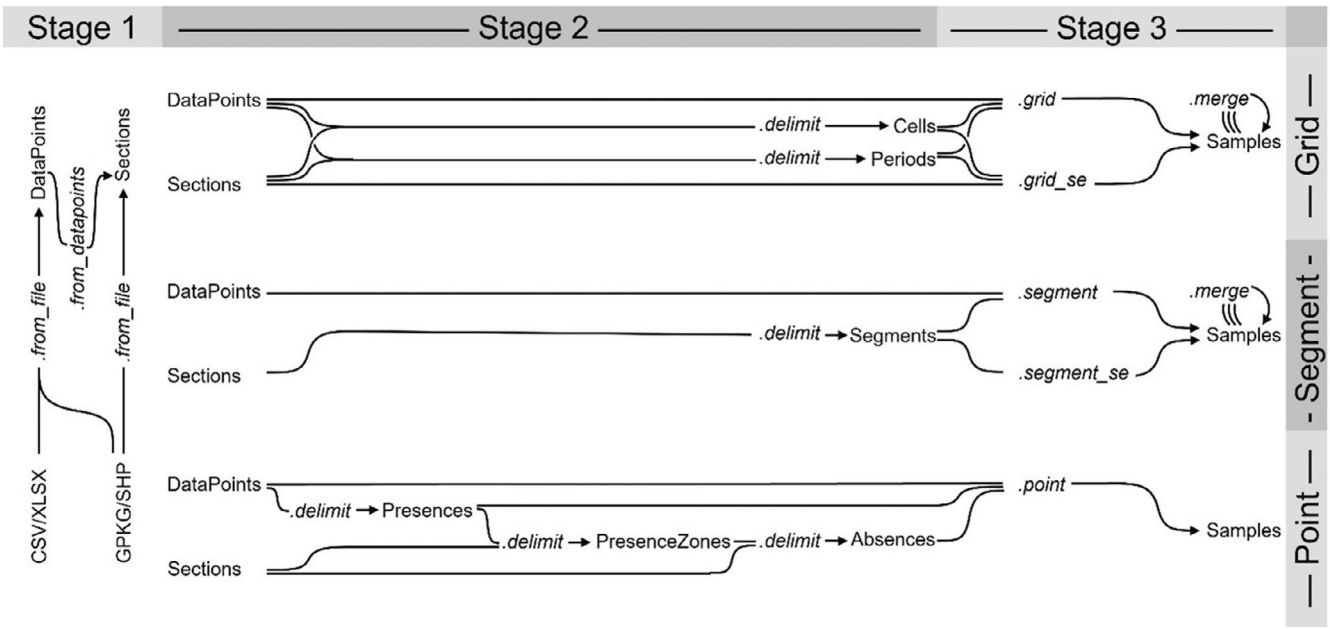

FIGURE 2 A graphical representation of the three Stages of data processing with `sampley`. Objects are written in normal font (e.g. DataPoints) while methods (i.e. functions) are written in italics (e.g. *.delimit*). Stage 1 is the same for all approaches and results in DataPoints and/or Sections objects that, in Stages 2 and 3, are processed via the grid, segment and/or point approach to produce Samples objects that can then be exported. A detailed explanation of this figure can be found in the User Manual.

2.2.1 | Grid approach

To implement the grid approach, cells, either rectangular or hexagonal, must be made. Rectangular cells are widely employed in ecological studies (e.g. Pendleton et al., 2020; Syme et al., 2023) while hexagonal cells, although less frequently used (but see, for example: Giralt Paradell et al., 2019; Viquerat et al., 2022), may offer several advantages (Birch et al., 2007). For both variations, a key parameter is the size of the cells, which is determined by specifying the length of a side.

Additionally, to add a temporal dimension, cells can be combined with temporal periods (e.g. Pendleton et al., 2020; Syme et al., 2023). Regular temporal periods (e.g. 1-year, 2-month, or 8-day periods) can be made with `sampley`, or, alternatively, sections and datapoints can be assigned to periods of custom length (e.g. seasons) prior to importing them.

2.2.2 | Segment approach

Survey tracks may be cut into segments according to the simple, joining and redistribution variations. All three variations require a target length — the length at which or as close to which the segments will be cut. They differ, however, in how they deal with the remainder — the portion of survey track that, almost inevitably, remains after dividing into segments of the target length. The simple variation leaves the remainder as an independent, albeit shorter, segment (e.g. Bedriñana-Romano et al., 2023). In the joining variation, if the remainder is shorter than half the target length, it is joined to a whole segment to make a single longer segment, but if it is longer than half the target length, it is left as an independent segment (e.g. Becker et al., 2022; Derville et al., 2022). For these two variations, the location of the remainder or joined segment, respectively, can be randomised. Finally, in the redistribution variation, the length of the remainder is redistributed equally amongst the segments (e.g. Roberts et al., 2016).

2.2.3 | Point approach

For the point approach, presences (i.e. detections of the species) are used to make presence zones, where each presence zone represents the area around a detection deemed to be occupied by the animal(s) (e.g. Rayment et al., 2014; Torres et al., 2008). Absences are then generated based on the sections and the presences zones by one of two variations. In the along-the-line variation, absences are generated along the survey track while, in the from-the-line variation, they are generated at various distances from the track based on some function (e.g. a detection function).

2.3 | Stage 3: Sampling

The third stage, sampling, consists of assigning data to delimiters to produce samples (Figure 2). This has no effect on the delimiters themselves; thus, multiple sets of data can be assigned successively to the same delimiters.

For the grid approach, datapoints can be assigned by determining which cell they lie within and, optionally, by binning them into corresponding temporal periods. Additionally, values for survey effort, calculated as either length of survey track or area of buffered survey track within each cell (and, optionally, temporal period), can be calculated.

For the segment approach, datapoints can be assigned by matching them to their corresponding segment. Depending on the data, matching may be done on the basis of: Euclidean distance to the segments or their midpoints; datetimes; or distance from the beginning of the sections (a variable that serves as a unique coordinate for positions relative to sections and datapoints). Additionally, survey effort can be calculated for each segment as: length; area, by entering a one-sided stripwidth; or effective area, by entering a one-sided area under a detection function (i.e. effective stripwidth).

For the point approach, presences and absences are concatenated. Data can be assigned from datapoints to presences by matching their IDs (that are generated during data importing) and, if datapoints are continuous, to absences based on their distances from the beginning of the sections. The resulting samples may also be thinned (i.e. some points within close proximity removed) to avoid issues associated with autocorrelation (Dormann et al., 2007).

With all three approaches, the result is a set of samples containing assigned data as well as geographic (i.e. coordinates and geometries) and temporal values (i.e. datetimes).

2.4 | Additional features

All of the objects imported to or made with `sampley` (i.e. survey tracks, datapoints, periods, cells, segments, presences, presence zones, absences and samples), have inbuilt save and open methods so that the user can pause and resume their work at any stage. Moreover, all objects exported from `sampley` are saved as CSV and/or GPKG files — two commonly used filetypes that can be loaded into R, GIS or any other suitable software. Thus, after processing data with `sampley`, the user could, for example, import the samples into R for statistical analysis or import the cells into GIS to make a map. Additionally, all objects, except periods and samples, have inbuilt plot methods, allowing the user to easily conduct visual checks of processing.

3 | EXAMPLE APPLICATION

Here, we provide an example application of `sampley` that applies the segment approach to a mock dataset of survey tracks, sightings and in situ environmental data. The full example is available in the Supporting Information as well as via the project page (https://pypi.org/project/sampley/), which also provides access to the mock data and resulting samples. Additionally, more exemplars illustrating the various approaches and variations applied to the same mock dataset can also be accessed via the project page.

3.1 | Data

For this example, the data are contained in a CSV and consist of continuous datapoints, where each datapoint (i.e. each row) is a point along the survey track as defined by a longitude (column 'lon'), latitude (column 'lat') and datetime value (column 'datetime') (Table 1). Each datapoint has a value for Beaufort sea state (column 'bss') and, if it is a sighting, the number of individuals observed (column 'individuals') (Table 1). Additionally, there is a column, 'section_id', that identifies which section of survey track each datapoint belongs to (Table 1).

3.2 | Stage 1: Importing data

In Stage 1, we import our dataset and make a `DataPoints` object (`u_trackpoints`) with the `from_file` method. We specify the names of the columns containing the coordinates and, as they are latitude and longitude values, EPSG:4326 as the coordinate reference system (CRS). As distances cannot be measured in geographic CRSs, like EPSG:4326, we also set the CRS to be used for processing as EPSG:32619 — a projected CRS whose units are metres. We specify the column containing the datetimes and their timezone as UTC-05:00. Finally, we specify the name of the section ID column.

```
## Stage 1: import data
u_trackpoints = DataPoints.from_file(
    filepath=import_folder+'trackpoints.csv',  # filepath
    x_col='lon',  # name of column containing x coordinates (longitudes)
    y_col='lat',  # name of column containing y coordinates (latitudes)
    crs_import='EPSG:4326',  # CRS of the x and y coordinates
    crs_working='EPSG:32619',  # CRS to be used for processing
    datetime_col='datetime',  # name of column containing datetimes
    tz_import='UTC-05:00',  # timezone of the datetimes
    section_id_col='section_id'  # name of column containing section IDs
)
```

We then enter our `DataPoints` object (`u_trackpoints`) to the `from_datapoints` method to make a `Sections` object (`u_sections`) (Figure 3).

```
## Stage 1: make sections from datapoints
u_sections = Sections.from_datapoints(datapoints=u_trackpoints)
```

3.3 | Stage 2: Delimiting

A `Segments` object is made by entering a `Sections` object and several parameters to the `delimit` method. Here, we firstly make segments with the simple variation (`u_segments_simple`) based on a target length of 10,000 m, with the location of the remainder randomised.

```
## Stage 2: make segments with simple variation
u_segments_simple = Segments.delimit(
    sections=u_sections,  # enter the sections
    var='simple',  # set the variation to simple
    target=10000,  # set the target length to 10000 metres
    rand=True)  # set to randomise the location of the remainder
```

The resulting segments (Table 2) each have a unique ID, a linestring geometry, a midpoint and a date. Additionally, each segment has the ID of the section that it was cut from as well as two values that indicate the distance from the beginning of the section at which the segment begins and ends, respectively. These serve to locate each segment relative to the sections.

By plotting the segments (Figure 4), we see that they are the same length except for the remainders, which are notably shorter. This may be an issue as it decreases the standardisation of segment lengths, with those shorter segments having considerably less survey effort than the others. We could manually remove the remainders; however, this would result in lost data, so instead we opt to implement a different variation.

Using the same method as before, we now make segments with the redistribution variation (`u_segments_redist`). The resulting segments have lengths that vary slightly from the target length but are standardised so that no segment is noticeably shorter or longer than any other (Figure 5).

section_id	lon	lat	datetime	individuals	bss
s001	−68.02000	42.83433	2019-01-25 10:18:13		2
s001	−68.02117	42.83433	2019-01-25 10:18:15		2
s001	−68.02583	42.83433	2019-01-25 10:18:23		2
s001	−68.03133	42.83433	2019-01-25 10:18:32	1	2
s001	−68.03483	42.83433	2019-01-25 10:18:38		2

TABLE 1 The first five rows of the mock data used in the example application of the package **`sampley`**.

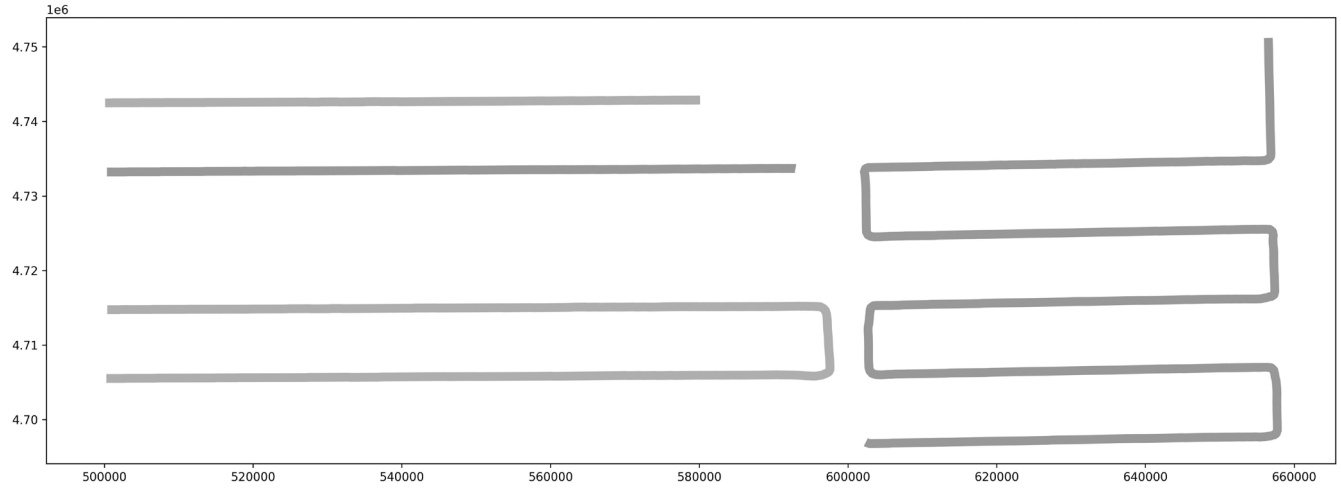

FIGURE 3 Sections of survey track used in the example application of the package `sampley`.

TABLE 2 The first five segments made with the simple variation in the example application of the package **`sampley`**.

segment_id	line	midpoint	date	section_id	dfbsec_beg	dfbsec_end
s01-s10000m	LINESTRING (...)	POINT (...)	2019-01-25	s001	0	10,000.0
s02-s10000m	LINESTRING (...)	POINT (...)	2019-01-25	s001	10,000.0	20,000.0
s03-s10000m	LINESTRING (...)	POINT (...)	2019-01-25	s001	20,000.0	30,000.0
s04-s10000m	LINESTRING (...)	POINT (...)	2019-01-25	s001	30,000.0	40,000.0
s05-s10000m	LINESTRING (...)	POINT (...)	2019-01-25	s001	40,000.0	40,004.6

```
## Stage 2: make segments with redistribution
variation
u_segments_redist = Segments.delimit(
    sections=u_sections,  # enter the sections
    var='redistribution',  # set the variation to
    redistribution
    target=10000)  # set the target length to
    10000 metres
```

3.4 | Stage 3: Sampling

With our chosen variation, the redistribution variation, we now proceed to assign data. Firstly, we make a `Samples` object with the `segment` method and enter our `DataPoints` object (`u_track-points`) as well as our `Segments` object (`u_segments_redist`). We state the columns in the `DataPoints` object that contain data that we want to assign to the segments and how we want to treat them: sum the number of individuals and average the Beaufort sea state. We also specify that the datapoints should be matched to the segments based on their distance from the beginning of the sections — an option made possible by our use of continuous datapoints. The resulting `Samples` object resembles the `Segments` object (Table 2) but with two additional columns: 'individuals' and 'bss'.

```
## Stage 3: make samples from datapoints
u_samples_trackpoints = Samples.segment(
    datapoints=u_trackpoints,  # enter the
    DataPoints
    segments=u_segments_redist,  # enter the
    Segments
    cols={  # set the data columns to assign:
        'individuals': 'sum',  # - sum the number
        of individuals per segment
        'bss': 'mean'},        # - average the BSS
        per segment
    how='dfb')  # match by distance from the be-
    ginning of the section
```

We then use the `segment_se` method to measure survey effort in two different ways: segment length and segment area (where the width is based on a one-sided stripwidth). The resulting `Samples` object resembles the `Segments` object (Table 2) but with two additional columns: 'se_length' and 'se_area'.

```
## Stage 3: make samples of survey effort
u_samples_effort = Samples.segment_se(
    segments=u_segments_redist,  # Segments object
    length=True,  # measure effort as length
    esw=2000  # measure effort as area based on
    stripwidth of 2000 metres)
```

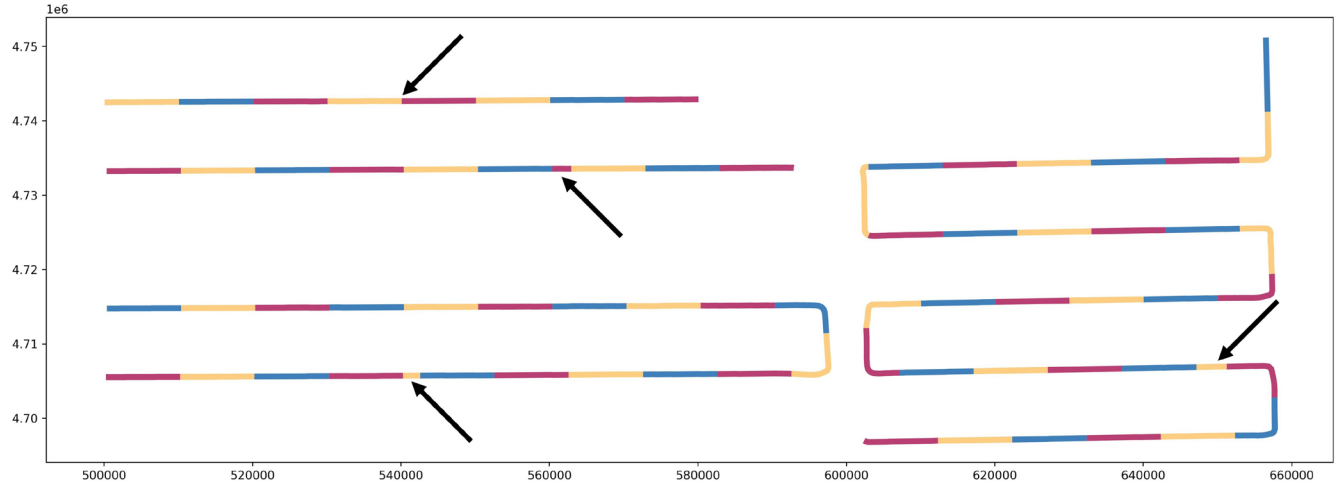

FIGURE 4 Segments made with the simple variation in the example application of the package `sampley`. The remainders (indicated with black arrows) are placed randomly within each section. Note that the remainder in the first section (top left) is so short that it is barely visible.

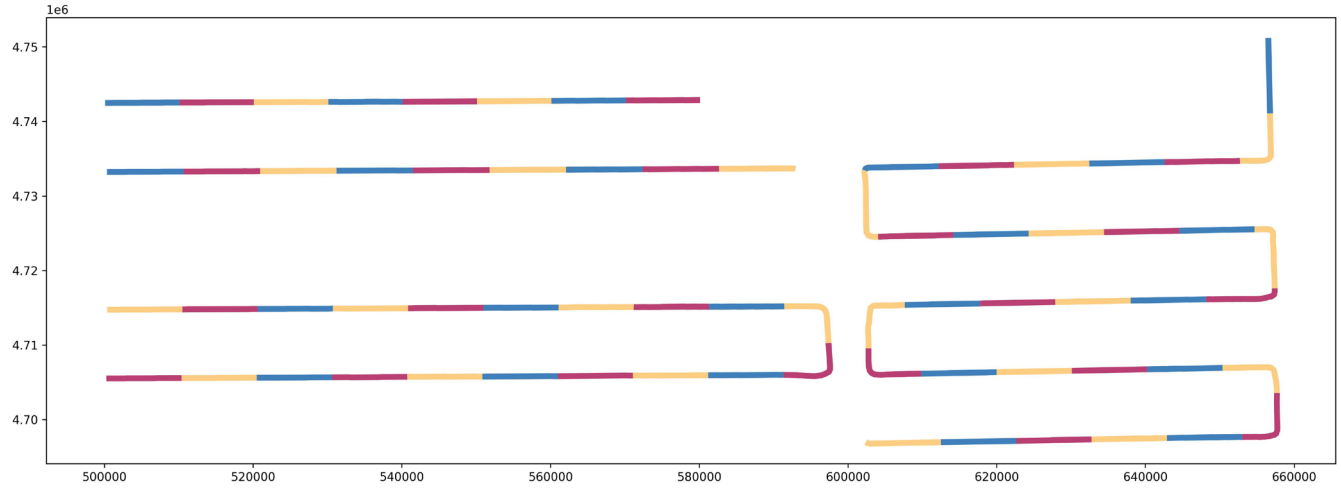

FIGURE 5 Segments made with the redistribution variation in the example application of the package `sampley`. Note that there are no remainders and that segment lengths are standardised.

We now have two `Samples` objects that we combine with the `merge` method to produce our final samples (`u_samples`) which contains all four additional columns: 'individuals', 'bss', 'se_length' and 'se_area' (Table 3).

```
## Stage 3: make final samples by merging samples
from datapoints and samples of survey effort
u_samples = Samples.merge(
    trackpoints=u_samples_trackpoints,  # samples
    from datapoints
    effort=u_samples_effort)  # samples of survey
    effort
```

Before saving the samples, we reproject them to EPSG:4326 (i.e. latitude-longitude coordinates) and we extract the coordinates of the midpoints, which are put into two new columns: 'midpoint_lon' and 'midpoint_lat' (Table 3). The samples are saved as a CSV, which, along with reprojecting and extracting the coordinates, will facilitate subsequent processing outside of `sampley` (e.g. extraction of data from remote datasets in GIS or statistical analysis in R).

```
## Stage 3: additional processing
u_samples.reproject(crs_target='EPSG:4326')  # re-
project to EPSG:4326
u_samples.coords()  # extract coordinates of the
midpoints

## Stage 3: save the samples
u_samples.save(
    folder=export_folder,  # export folder
    filetype='csv'  # set filetype to CSV
)
```

TABLE 3 The first five samples made with the redistribution variation of the segment approach in the example application of the package **sampley**. Note that some values have been abbreviated or rounded for display.

segment_id	line	midpoint	midpoint_lon	midpoint_lat	date	section_id	dfbsec_beg	dfbsec_end	individuals	bss	se_length	se_area
s01-r10000m	LINESTRING (...)	POINT (...)	−68.081	42.835	2019-01-25	s001	0	10,000.6	1	2.0	10,000.6	4.000232×10^7
s02-r10000m	LINESTRING (...)	POINT (...)	−68.203	42.835	2019-01-25	s001	10000.6	20,001.2	0	2.0	10,000.6	4.000232×10^7
s03-r10000m	LINESTRING (...)	POINT (...)	−68.326	42.835	2019-01-25	s001	20001.2	30,001.7	2	2.0	10,000.6	4.000232×10^7
s04-r10000m	LINESTRING (...)	POINT (...)	−68.448	42.835	2019-01-25	s001	30001.7	40,002.3	5	2.85	10,000.6	4.000232×10^7
s05-r10000m	LINESTRING (...)	POINT (...)	−68.571	42.836	2019-01-25	s001	40,002.3	50,002.9	0	3.0	10,000.6	4.000232×10^7

4 | CONCLUSION

Processing visual survey data with `sampley` requires only a few functions organised into three straightforward stages and so should not take long for the user to learn and implement. Additionally, its user-friendly nature, along with the guidance on setting up Python and `sampley` provided in the User Manual and the exemplar notebooks which can serve as templates, will make it accessible to those who are unfamiliar with Python.

While `sampley` provides the functionality, the choices of approach, variation and parameters remain at the user's discretion. These choices should be weighed carefully, given their potential influence on samples and related issues. For example, segmenting can increase the uniformity of survey effort across samples (Becker et al., 2010; Roberts et al., 2016); grid cells can decrease the risk of spatial autocorrelation (Aiello-Lammens et al., 2015); while the generation of absences can be restricted to avoid zero-inflation (Barbet-Massin et al., 2012). By incorporating multiple variations of the three approaches, `sampley` allows users to trial different options on the same dataset, which will help to inform these choices. This would otherwise be impractical, given the time required to develop custom scripts for multiple variations and approaches.

As there is no package currently available for ecologists to apply a suite of sampling procedures to visual survey data, there is no basis for comparing `sampley` with current workflows. However, implementing `sampley` will be more efficient for individual researchers than developing custom scripts, will facilitate comparisons of sampling approaches, and will improve reproducibility of studies. This will allow ecological modellers to make more informed decisions when processing data and to focus their attention on other aspects of their studies, such as modelling.

Although `sampley` is designed for visual survey data, it requires only a survey track and datapoints collected along it. Thus, it may be applicable to similar data from other sources (e.g. data from unmanned vehicles or acoustic surveys), assuming they also consist of a track and datapoints. Moreover, at a base level, `sampley`, like most geospatial software, operates on points, lines and polygons, and so may be applicable to diverse unrelated situations. For example, the functionality provided for the grid, segment and point approaches could be used, respectively, to assign observations of a species to polygons that represent different habitat types, to divide a river into various reaches of a standardised length to be used as management zones or to generate points along a satellite tag track.

In summary, we believe that `sampley` will assist researchers by allowing them to process survey data in a more efficient and reproducible way and to make better-informed decisions when deriving samples for modelling.

AUTHOR CONTRIBUTIONS

Jonathan Syme and Daniel E. Pendleton conceived the idea and designed the package. Jonathan Syme developed the package with

advice from Daniel E Pendleton, Nicholas R Record and Benjamin Tupper. Jonathan Syme led the writing of the manuscript. All authors contributed critically to the drafts and gave final approval for publication.

ACKNOWLEDGEMENTS

We would like to thank Doug Sigourney and Julia Stepanuk for testing the package during its development and providing valuable feedback. Support for this work was provided by NSF Grant ORCC #2307754.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest regarding this work.

PEER REVIEW

The peer review history for this article is available at https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210x.70320.

DATA AVAILABILITY STATEMENT

Data are available via https://doi.org/10.5281/zenodo.19616964 (Syme et al., 2025).

ORCID

Jonathan Syme https://orcid.org/0000-0001-8390-8468

REFERENCES

Aiello-Lammens, M. E., Boria, R. A., Radosavljevic, A., Vilela, B., & Anderson, R. P. (2015). spThin: An R package for spatial thinning of species occurrence records for use in ecological niche models. *Ecography*, *38*(5), 541–545. https://doi.org/10.1111/ecog.01132

Anderson, A. S., Marques, T. A., Shoo, L. P., & Williams, S. E. (2015). Detectability in audio-visual surveys of tropical rainforest birds: The influence of species, weather and habitat characteristics. *PLoS One*, *10*(6), e0128464. https://doi.org/10.1371/journal.pone.0128464

Barbet-Massin, M., Jiguet, F., Albert, C. H., & Thuiller, W. (2012). Selecting pseudo-absences for species distribution models: How, where and how many? *Methods in Ecology and Evolution*, *3*(2), 327–338. https://doi.org/10.1111/j.2041-210X.2011.00172.x

Baumberger, T., Croze, T., Affre, L., & Mesléard, F. (2012). Co-occurring species indicate habitats of the rare *Limonium girardianum*. *Plant Ecology and Evolution*, *145*(1), 31–37. https://doi.org/10.5091/plecevo.2012.685

Becker, E. A., Forney, K. A., Ferguson, M. C., Foley, D. G., Smith, R. C., Barlow, J., & Redfern, J. V. (2010). Comparing California current cetacean–habitat models developed using in situ and remotely sensed sea surface temperature data. *Marine Ecology Progress Series*, *413*, 163–183. https://doi.org/10.3354/meps08696

Becker, E. A., Forney, K. A., Miller, D. L., Barlow, J., Rojas-Bracho, L., Jorge Urbán, R., & Moore, J. E. (2022). Dynamic habitat models reflect interannual movement of cetaceans within the California current ecosystem. *Frontiers in Marine Science*, *9*, 829523. https://doi.org/10.3389/fmars.2022.829523

Bedriñana-Romano, L., Viddi, F. A., Artal, O., Pinilla, E., & Hucke-Gaete, R. (2023). First estimate of distribution, abundance, and risk of encounter with aquaculture vessels for the rare Chilean dolphin in the entire northern Chilean Patagonia. *Aquatic Conservation: Marine and Freshwater Ecosystems*, *33*(12), 1535–1551. https://doi.org/10.1002/aqc.4012

Birch, C. P. D., Oom, S. P., & Beecham, J. A. (2007). Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological Modelling*, *206*(3), 347–359. https://doi.org/10.1016/j.ecolmodel.2007.03.041

Derville, S., Barlow, D. R., Hayslip, C., & Torres, L. G. (2022). Seasonal, annual, and decadal distribution of three Rorqual whale species relative to Dynamic Ocean conditions off Oregon, USA. *Frontiers in Marine Science*, *9*, 868566. https://doi.org/10.3389/fmars.2022.868566

Derville, S., Constantine, R., Baker, S., & Oremus, M. (2016). Environmental correlates of nearshore habitat distribution by the critically endangered Māui dolphin. *Marine Ecology Progress Series*, *551*, 261–275. https://www.int-res.com/abstracts/meps/v551/p261-275/

Dormann, C. F., McPherson, J. M., Araújo, M. B., Bivand, R., Bolliger, J., Carl, G., Davies, R. G., Hirzel, A., Jetz, W., Daniel Kissling, W., Kühn, I., Ohlemüller, R., Peres-Neto, P. R., Reineking, B., Schröder, B., Schurr, F. M., & Wilson, R. (2007). Methods to account for spatial autocorrelation in the analysis of species distributional data: A review. *Ecography*, *30*(5), 609–628. https://doi.org/10.1111/j.2007.0906-7590.05171.x

Elith, J., & Franklin, J. (2013). Species distribution modeling. In S. A. Levin (Ed.), *Encyclopedia of biodiversity* (2nd ed., pp. 692–705). Academic Press. https://doi.org/10.1016/B978-0-12-384719-5.00318-X

Fernandez, M., Sillero, N., & Yesson, C. (2022). To be or not to be: The role of absences in niche modelling for highly mobile species in dynamic marine environments. *Ecological Modelling*, *471*, 110040. https://doi.org/10.1016/j.ecolmodel.2022.110040

Gillies, S., van der Wel, C., den Bossche, J., Taves, M. W., Arnott, J., Ward, B. C., et al. (2024). Shapely. *Zenodo*. https://doi.org/10.5281/zenodo.13345370

Giralt Paradell, O., Díaz López, B., & Methion, S. (2019). Modelling common dolphin (*Delphinus delphis*) coastal distribution and habitat use: Insights for conservation. *Ocean and Coastal Management*, *179*, 104836. https://doi.org/10.1016/j.ocecoaman.2019.104836

Guillera-Arroita, G., Lahoz-Monfort, J. J., Elith, J., Gordon, A., Kujala, H., Lentini, P. E., McCarthy, M. A., Tingley, R., & Wintle, B. A. (2015). Is my species distribution model fit for purpose? Matching data and models to applications. *Global Ecology and Biogeography*, *24*(3), 276–292. https://doi.org/10.1111/geb.12268

Merkel, B., Trathan, P., Thorpe, S., Murphy, E. J., Pehlke, H., Teschke, K., & Griffith, G. P. (2023). Quantifying circumpolar summer habitat for Antarctic krill and ice krill, two key species of the Antarctic marine ecosystem. *ICES Journal of Marine Science*, *80*(6), 1773–1786. https://doi.org/10.1093/icesjms/fsad110

Meyer-Gutbrod, E. L., Kui, L., Nishimoto, M. M., Love, M. S., Schroeder, D. M., & Miller, R. J. (2019). Fish densities associated with structural elements of oil and gas platforms in southern California. *Bulletin of Marine Science*, *95*(4), 639–656. https://doi.org/10.5343/bms.2018.0078

Pasanisi, E., Pace, D. S., Orasi, A., Vitale, M., & Arcangeli, A. (2024). A global systematic review of species distribution modelling approaches for cetaceans and sea turtles. *Ecological Informatics*, *82*, 102700. https://doi.org/10.1016/j.ecoinf.2024.102700

Pendleton, D. E., Holmes, E. E., Redfern, J., & Zhang, J. (2020). Using modelled prey to predict the distribution of a highly mobile marine mammal. *Diversity and Distributions*, *26*(11), 1612–1626. https://doi.org/10.1111/ddi.13149

Phillips, S. J., Dudík, M., Elith, J., Graham, C. H., Lehmann, A., Leathwick, J., & Ferrier, S. (2009). Sample selection bias and presence-only distribution models: Implications for background and pseudo-absence data. *Ecological Applications*, *19*(1), 181–197. https://doi.org/10.1890/07-2153.1

Python Software Foundation. (2023). *Python language reference version 3.12.0 (3.12.0)*. Python Software Foundation. https://www.python.org/

Rayment, W., Dawson, S., & Webster, T. (2014). Breeding status affects fine-scale habitat selection of southern right whales on their wintering grounds. *Journal of Biogeography*, *42*(3), 463–474. https://doi.org/10.1111/jbi.12443

Roberts, J. J., Best, B. D., Mannocci, L., Fujioka, E., Halpin, P. N., Palka, D. L., Garrison, L. P., Mullin, K. D., Cole, T. V. N., Khan, C. B., McLellan, W. A., Pabst, D. A., & Lockhart, G. G. (2016). Habitat-based cetacean density models for the U.S. Atlantic and Gulf of Mexico. *Scientific Reports*, *6*(1), 22615. https://doi.org/10.1038/srep22615

Stepanuk, J. E. F., Kim, H., Nye, J. A., Roberts, J. J., Halpin, P. N., Palka, D. L., Pabst, D. A., McLellan, W. A., Barco, S. G., & Thorne, L. H. (2023). Subseasonal forecasts provide a powerful tool for dynamic marine mammal management. *Frontiers in Ecology and the Environment*, *21*(3), 117–123. https://doi.org/10.1002/fee.2506

Syme, J., Kiszka, J. J., & Parra, G. J. (2023). Habitat partitioning, co-occurrence patterns, and mixed-species group formation in sympatric delphinids. *Scientific Reports*, *13*(1), 3599. https://doi.org/10.1038/s41598-023-30694-w

Syme, J., Pendleton, D. E., Meyer-Gutbrod, E. L., Tupper, B., & Record, N. R. (2025). sampley: Sample survey data (v0.0.15). https://doi.org/10.5281/zenodo.19616964

Torres, L. G., Read, A. J., & Halpin, P. (2008). Fine-scale habitat modeling of a top marine predator: Do prey data improve predictive capacity. *Ecological Applications*, *18*(7), 1702–1717. https://doi.org/10.1890/07-1455.1

Van den Bossche, J., Jordahl, K., Fleischmann, M., Richards, M., McBride, J., Wasserman, J., Badaracco, A. G., Snow, A. D., Ward, B., Tratner, J., Gerard, J., Perry, M., Hjelle, G. A., Taves, M., ter Hoeven, E., Cochran, M., Bell, R., Bartos, M., Roggemans, P., … Gardiner, J. (2024). geopandas/geopandas: v1.0.1. *Zenodo*. https://doi.org/10.5281/zenodo.12625316

Viquerat, S., Waluda, C. M., Kennedy, A. S., Jackson, J. A., Hevia, M., Carroll, E. L., Buss, D. L., Burkhardt, E., Thain, S., Smith, P., Secchi, E. R., Santora, J. A., Reiss, C., Lindstrøm, U., Krafft, B. A., Gittins, G., Dalla Rosa, L., Biuw, M., & Herr, H. (2022). Identifying seasonal distribution patterns of fin whales across the Scotia Sea and the Antarctic peninsula region using a novel approach combining habitat suitability models and ensemble learning methods. *Frontiers in Marine Science*, *9*, 1040512. https://doi.org/10.3389/fmars.2022.1040512

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

Data S1: Example application of `sampley`.

How to cite this article: Syme, J., Pendleton, D. E., Meyer-Gutbrod, E. L., Tupper, B., & Record, N. R. (2026). **`sampley`**: A Python package for sampling visual survey data. *Methods in Ecology and Evolution*, *00*, 1–11. https://doi.org/10.1111/2041-210x.70320