Received: 20 August 2024 | Accepted: 24 November 2024

DOI: 10.1111/2041-210X.14487

Methods in Ecology and Evolution

APPLICATION

AMMonitor 2: Remote monitoring of biodiversity in an adaptive framework in R

Laurence Clarfeld[1] | Caroline Tang[1] | Kaitlin Huber[1] | Cathleen Balantic[2] | Therese Donovan[3]

[1]Vermont Cooperative Fish and Wildlife Research Unit, 302 Aiken Center, University of Vermont, Burlington, Vermont, USA

[2]Natural Sounds and Night Skies Division, National Park Service, Fort Collins, Colorado, USA

[3]U.S. Geological Survey, Vermont Cooperative Fish and Wildlife Research Unit, 302 Aiken Center, University of Vermont, Burlington, Vermont, USA

Correspondence
Therese Donovan
Email: tdonovan@uvm.edu

Funding information
U.S. Geological Survey

Handling Editor: Miguel Acevedo

Abstract

1. Wildlife face a myriad of challenges due to changing climate and land use regimes that necessitate efficient monitoring methods on large geospatial scales. This monitoring is increasingly accomplished with remotely deployed trail cameras and automated recording units.
2. However, remote wildlife monitoring comes with data management challenges, such as storing, organizing and analysing large volumes of digital data. Data management tools can assist biologists in managing projects to reduce the barriers between data collection and dissemination of results.
3. *AMMonitor* is an R package for remote wildlife monitoring with greatly increased functionality in the version 2 release, including a new database structure, support for both photographs and audio monitoring, a rich graphical user interface (Shiny) that includes media labelling tools, integration of machine learning classification outputs and 20 in-depth tutorials.
4. As a fully open-source data management solution, *AMMonitor* is highly accessible, extensible and customizable, making it especially useful for students and users with some coding experience who desire a standardized yet flexible data management framework.
5. *AMMonitor* project data can be released to the public via the USGS ScienceBase data repository, where released projects can be reconstituted by *AMMonitor* functions.

KEYWORDS
acoustic monitoring, *learnr*, machine learning, R, species distribution modelling, trail camera, wildlife monitoring

1 | INTRODUCTION

In a time of rapid global change, effective wildlife monitoring methods are essential for tracking species responses and for making well-informed resource management decisions (Fuller et al., 2020; Polasky et al., 2011). Remote monitoring via camera trapping and audio recorders, collectively referred to as autonomous monitoring units (AMUs), has gained popularity for monitoring multiple taxa in both

terrestrial and marine environments. Audio recorders capture sounds like animal calls, providing insights into species presence, behaviour and communication. In contrast, camera trapping offers visual confirmation, documenting physical traits, interactions and activities. AMUs can provide broad spatiotemporal coverage, capturing signals of target wildlife in the vast troves of digital data they accumulate.

These data present both opportunities and challenges to biologists whose training may not have included the data management tasks associated with AMUs. Even modestly sized monitoring projects can bury researchers under terabytes of data that must be stored, organized and analysed. Shepherding data from the field to the file system and extracting meaningful information about target wildlife from media data can be a bottleneck in monitoring projects. Thus, tools for efficiently moving from data collection to results and analysis are paramount.

Several open-source software options exist for managing monitoring data collected by trail cameras (e.g. see Hendry & Mann, 2017; Ivan & Newkirk, 2016; Niedballa et al., 2016), and online platforms include cloud storage and the ability to run machine learning (ML) models (Ahumada et al., 2020; WildEye, 2023). For acoustic monitoring, software options include both open source (Program, 2014) and subscription-based platforms that feature cloud storage and the ability to run ML models that search for acoustic signals (Aide et al., 2013). Finally, online platforms for managing for both trail camera and acoustic data offer storage, the ability to run ML models, and offer a supporting R package for working with data (MacPhail et al., 2024).

All of these efforts are crucial contributions for tackling the biodiversity crisis the world now faces. However, there is a need for an analysis framework targeted towards individual users, one that synthesizes both camera and acoustic monitoring data, data management, tagging and ML models within a singular framework. AMMonitor is an open-source alternative that can be flexibly used by any number of investigators to address a wide variety of ecological questions and that promotes collaboration through a shared database structure.

AMMonitor (pronounced 'A-M-monitor') stands for 'monitoring for adaptive management' and is an R package of the same name. It is an analysis ecosystem (R + SQLite + media storage) that seamlessly incorporates wildlife monitoring data, ML models and tagging, all of which can feed into species distribution models and decision tools. The analysis ecosystem is characterized by:

1. *Continuous stream of data collection on a variety of wildlife taxa*: AMUs, such as trail cameras or audio recorders allow the remote capture of digital files that form the backbone of wildlife monitoring. AMUs can be deployed at small or large scales, over short- or long-time frames and are inexpensive relative to human-based surveys.
2. *Standardized yet flexible data and metadata infrastructure*: A standardized data infrastructure is critical to store files with strict metadata standards so that data can be easily incorporated into open access repositories or shared between projects.
3. *Rapid analysis*: Vast quantities of AMU data may be collected rapidly, rendering manual inspection for target wildlife species impractical. Accordingly, ML models (developed within *AMMonitor* or external to *AMMonitor*) can be used to automatically detect wildlife species in recordings and photographs, avoiding the time-consuming task of manual annotation. ML outputs can be seamlessly incorporated into species distribution models and updated as new data are collected via continuous monitoring.

Each *AMMonitor* project utilizes the same approach, visualized in Figure 1:

The first iteration of *AMMonitor* (Balantic & Donovan, 2020; Donovan et al., 2021) builds on the R package, *monitoR* (Katz et al., 2016), for acoustic monitoring programmes. The second iteration of *AMMonitor* significantly expands on the first (Clarfeld, Tang, Huber, et al., 2024). Primary developments include: Expanded media support for wildlife monitoring with trail cameras; a user-friendly interface for running analyses and working with data via R Shiny (Chang et al., 2022), including image and audio taggers; a redesigned SQLite database to expand functionality, improve memory usage and performance and maintain compatibility with Postgres for served databases; support for a variety of cloud-based storage options for storing media files using APIs from the Google Drive (D'Agostino McGowan & Bryan, 2023), Microsoft365R (Ooi, 2023) and aws.s3 (Leeper, 2020) packages; functions that enable the creation of no-code cell phone apps, such as Survey123 and Google AppSheet for field data collection; updated R functions that streamline data analysis; improved ability to develop template-based audio ML models; ability to incorporate labels produced by external ML models, such as from Microsoft's MegaDetector (Beery et al., 2019) or Cornell Lab of Ornithology's BirdNET (Kahl et al., 2021); a suite of *learnr* tutorials that introduce users to the *AMMonitor* 2 software and approach; and functions that reconstitute *AMMonitor* projects that have been released on the USGS ScienceBase repository.

2 | THE AMMonitor 2 PACKAGE

2.1 | Getting started

The canonical home of *AMMonitor* is https://code.usgs.gov/vtcfwru/ammonitor, where updated guidance on installation can be found and where users can post issues, create merge requests and download development versions.

The current release can be installed from this site with the following code:

```
# Increase the timeout (for slow downloads)
getOption("timeout")
options(timeout = 300)

# Install with the remotes package
remotes::install_gitlab(
```

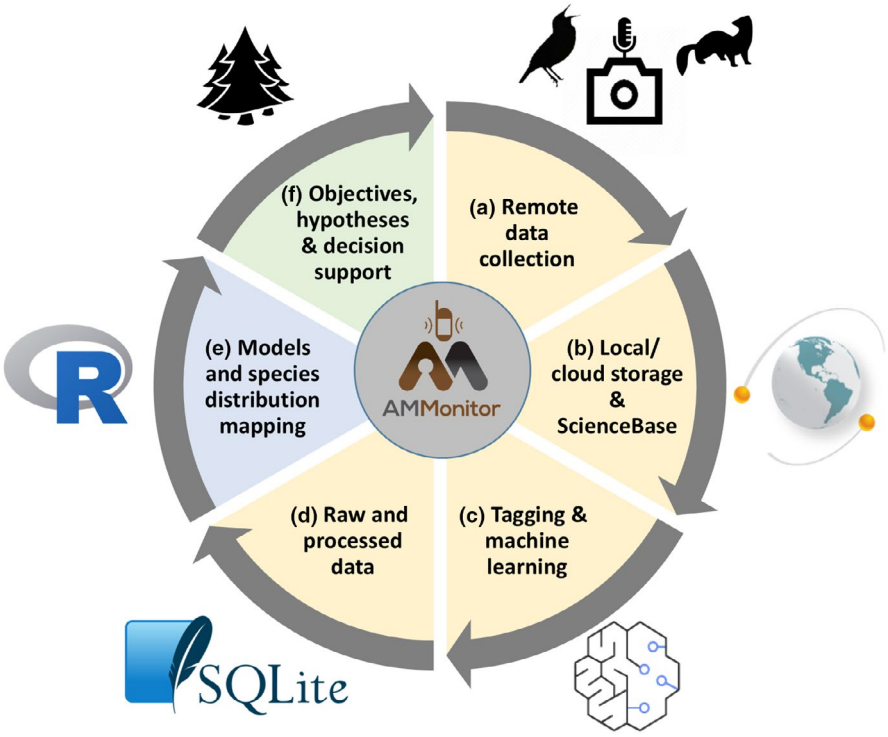

FIGURE 1 Adaptive management cycle employed by *AMMonitor*: (a) Wildlife data are collected by AMUs. (b) R functions upload media files to a local machine or to the cloud. (c) Media files can be tagged by humans and/or analysed with machine learning models that automate the species detection process. (d) Raw and post-processed data are maintained in the project's SQLite database. (e) R functions and scripts permit rapid, reproducible analyses of species distribution models and resulting maps, regularly updated along with metadata as new data are analysed. (f) These products can be the building blocks of decision analysis, continually updated as knowledge evolves, enabling decisions to be made with the most current information.

```
  repo = "vtcfwru/ammonitor@2.1.0",
  auth_token = Sys.getenv("GITLAB_PAT"),
  host = "code.usgs.gov",
  build_vignettes = FALSE,
  dependencies = TRUE,
  upgrade = "never")

# Load the package
library(AMMonitor)
```

AMMonitor requires R version 4.4 or greater. For a full package description, including all dependencies, run utils::packageDescription('AMMonitor').

The function ammCreateDirectories() creates a new *AMMonitor* project, generating an *AMMonitor* project folder within the specified parent directory. For example, the following code will create an *AMMonitor* project called 'myProject' within R's working directory:

```
# Create the AMMonitor directory structure and capture the filepath
fp <- ammCreateDirectories(
  amm_dirname = "myProject",
  filepath = getwd())
```

This project directory contains 16 subdirectories for specific elements of the monitoring framework. Of these, two are required: (1) the 'database' directory, which stores the project's SQLite database, and (2) the 'settings' directory, which stores text files that contain project settings.

The dbCreate() function will create a new SQLite database to track all elements of the monitoring project, including people, equipment, monitoring locations, media file metadata, tags and ML results.

```
# Create a new AMMonitor SQLite database
dbCreate(
  new_db_name = "my_project_AMM.sqlite",
  new_db_filepath = file.path(fp, "database"))
```

From here, the user can begin populating their empty database with project-specific information. Importantly, the database itself does not store media files; instead, it stores metadata about those files. Media files may be stored within the 'photos' and 'recordings' directories in an *AMMonitor* project or on an external drive. Alternatively, *AMMonitor* now supports several cloud storage options for media files.

To assist users who are getting started, *AMMonitor* includes a demo project and database (created with the ammCreateMiniDemo() function) that is populated with metadata and linked media files from a fictional project in Middle-earth. This 'mini demo' will be the basis for workable examples going forward.

```
# Create the AMMonitor demo project and capture the filepath
fp <- ammCreateMiniDemo(filepath = tempdir())
```

2.2 | Interacting with the database

In contrast with the first iteration of *AMMonitor*, which required the user to run commands for the package through function calls, the new *AMMonitor* Shiny app provides an accessible, intuitive graphical

interface for working with the database and running queries, viewing/labelling media data and running mini-apps that perform guided analyses (introduced later). The Shiny app can be launched with the launchApp() function:

```
launchApp(fp)
```

Once launched, a user can log in with a person ID from the database's 'people' table (the 'unknownPerson' ID can be used for new projects; Figure 2).

The left menu provides access to the database, 'mini-apps' that guide users through a given analysis, and photograph and audio tools. As shown, the demo project includes 25 photographs and one audio recording. The database is the backbone of an *AMMonitor* project, and includes 45 tables for storing a wide range of project metadata, including 'equipment', 'locations', 'visits', 'media', 'annotations', 'models' and 'modeloutputs'. In the app's 'database' tab, a user can add/edit/delete database records, although many table entries are added via *AMMonitor* functions. For example, the temporalsGet() function scrapes weather data using the *rnoaa* package (Chamberlain & Hocking, 2023) and inserts them into the 'temporals' table, while the taxaAdd() function uses the *ritis* package (Chamberlain, 2021) to insert taxonomic entries from the Integrated Taxonomic Information System into the 'taxa' table. For some linked tables, related records can be viewed as subtables. For example, Figure 3 shows the 'locations' table, where the 'visits' subtable is nested and shows all field visits associated with 'locationA'. Tables are sortable and filterable. For more advanced querying, a user can use the 'Custom Queries' tab to enter arbitrary SQL code for retrieving records.

Alternatively, R users who prefer a code-based approach instead of a point and click interface can bypass the Shiny dashboard and interact with the database directly. This is achieved by creating a database connection with the dbSetCon() function, and then using *AMMonitor* functions or functions from the package DBI (R Special Interest Group on Databases et al., 2023).

```
# Set a connection the database
conx <- dbSetCon(file.path(fp, "database", "demo.sqlite"))

# Read in the locations table
locations <- DBI::dbReadTable(conn = conx, name = "locations")

# Add code here to work with the locations table in R
head(locations)
```

The 'dbdictionary' table is a crucial table. It stores the database dictionary that provides definitions for all tables and fields within the database. The database format was designed to be flexible to accommodate a wide variety of projects but includes 'core' columns for each table to enforce a standardized naming across projects and facilitate data sharing. However, the database is also extensible for projects that require more customization. For instance, if a user wanted to add a field named 'directions' to the 'locations' table, they could do so with the dbAddUserCol() function, which adds the column and updates the database dictionary.

2.3 | Labelling media content with shiny

A core functionality of *AMMonitor*'s Shiny app is the ability to view, listen to, annotate and verify labels for any photographs or recordings.

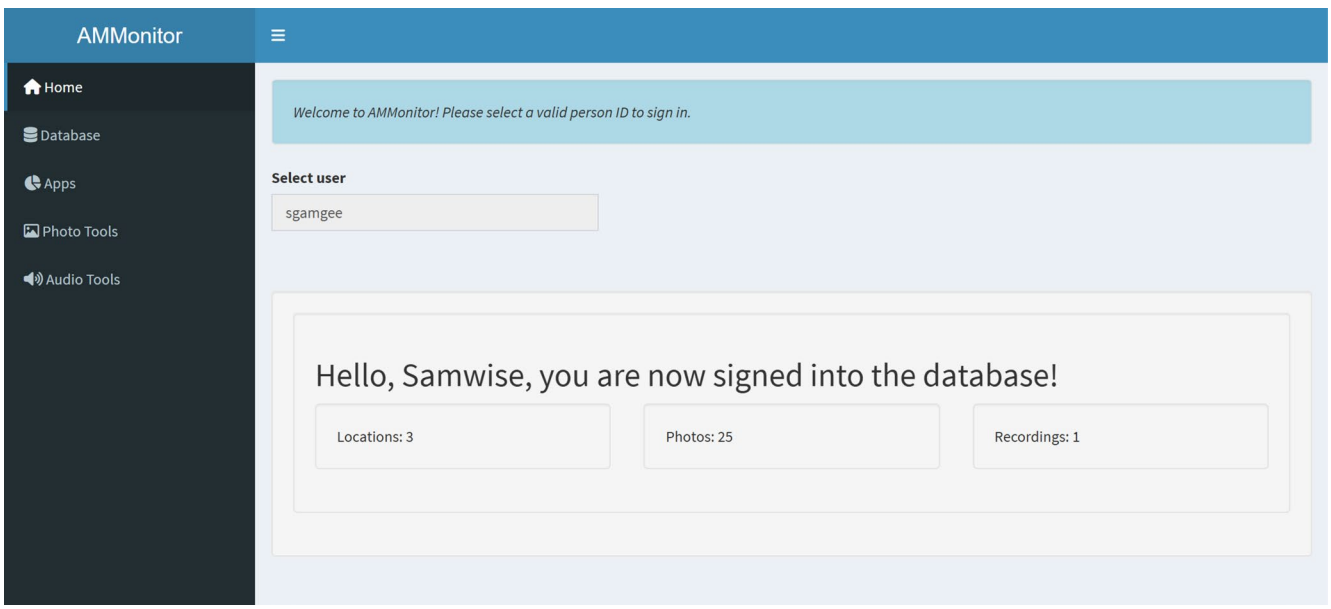

FIGURE 2 *AMMonitor*'s launchApp() function will launch the Shiny application, providing a friendly graphical user interface for working with an *AMMonitor* project.

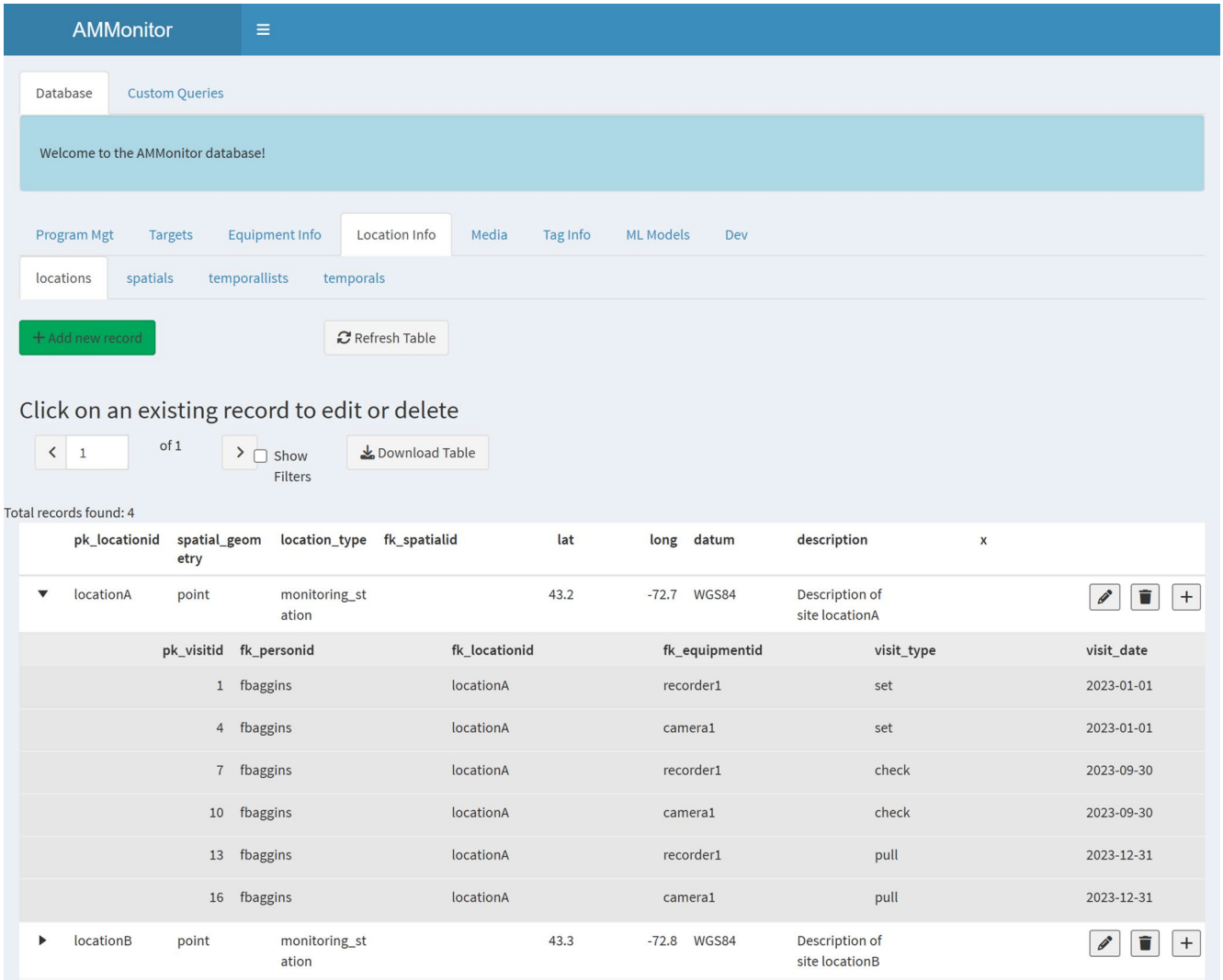

FIGURE 3 *AMMonitor*'s database front-end, illustrating a view of the locations table, showing a subtable with each visit for a selected location.

For this, *AMMonitor*'s photograph and audio tools allow users to interact with media files stored locally or in the cloud (Figure 4).

AMMonitor allows taxonomic labels to be applied to media with bounding boxes and audio timestamps (e.g. there is a chickadee at minute 1.20 in this audio file). Taxonomic labels can be further described with 'adjectives' defined in the other database tables (e.g. age, sex, number of individuals). Non-taxonomic and media-level labels are also supported.

2.4 | AMMonitor mini-apps

In addition to the database front-end and media tools, *AMMonitor's* mini-apps provide users with guided, step-by-step analyses. For example, the 'registerVisit' mini-app consists of seven tabs that guide users through the process of uploading new media files and registering the media metadata into the database (Figure 5). All mini-apps are built using *shinymgr*, an R package for building, managing, and stitching shiny modules into reproducible workflows (Clarfeld, Tang, & Donovan, 2024). This not only gives *AMMonitor*'s mini-apps a standardized, sequential, tab-based approach, but also makes *AMMonitor* easily extensible, with the ability for developers to add new mini-apps created with *shinymgr* and further increase the functionality of *AMMonitor*. These new contributions can be officially incorporated into the *AMMonitor* package via a merge request at https://code.usgs.gov/vtcfwru/ammonitor.

3 | TUTORIALS

AMMonitor includes 20 interactive tutorials that can be launched using the package *learnr* (Aden-Buie et al., 2023). Here, we use *learnr*'s available_tutorials() function to display the first 5 tutorials (see S1 for the full list of current tutorials).

FIGURE 4 Audio player (top) and photo viewer (bottom) are some of the media tools that allow *AMMonitor* users to view, listen to, annotate and verify labels for photographs and recordings.

```
tutorials <- learnr::available_tutorials(package = "AMMonitor")
tutorials[order(tutorials$title)[1:5],]
Available tutorials:
* AMMonitor
 - intro               : "AMMonitor 01: Introduction"
  - getting_started : "AMMonitor 02: Getting Started"
  - database          : "AMMonitor 03: Database"
  - dbdictionary      : "AMMonitor 04: The dbdictionary table"
  - dbsummary         : "AMMonitor 05: Summarizing the database"
```

Tutorials can be accessed via the 'Tutorial' tab in RStudio by searching for tutorials with 'AMMonitor' in the title, or can be launched via code with *learnr*'s run_tutorial function. For example, the following code will launch the tutorial that guides users through the creation of a new *AMMonitor* project (Figure 6):

```
learnr::run_tutorial(
  name = "getting_started",
  package = "AMMonitor")
```

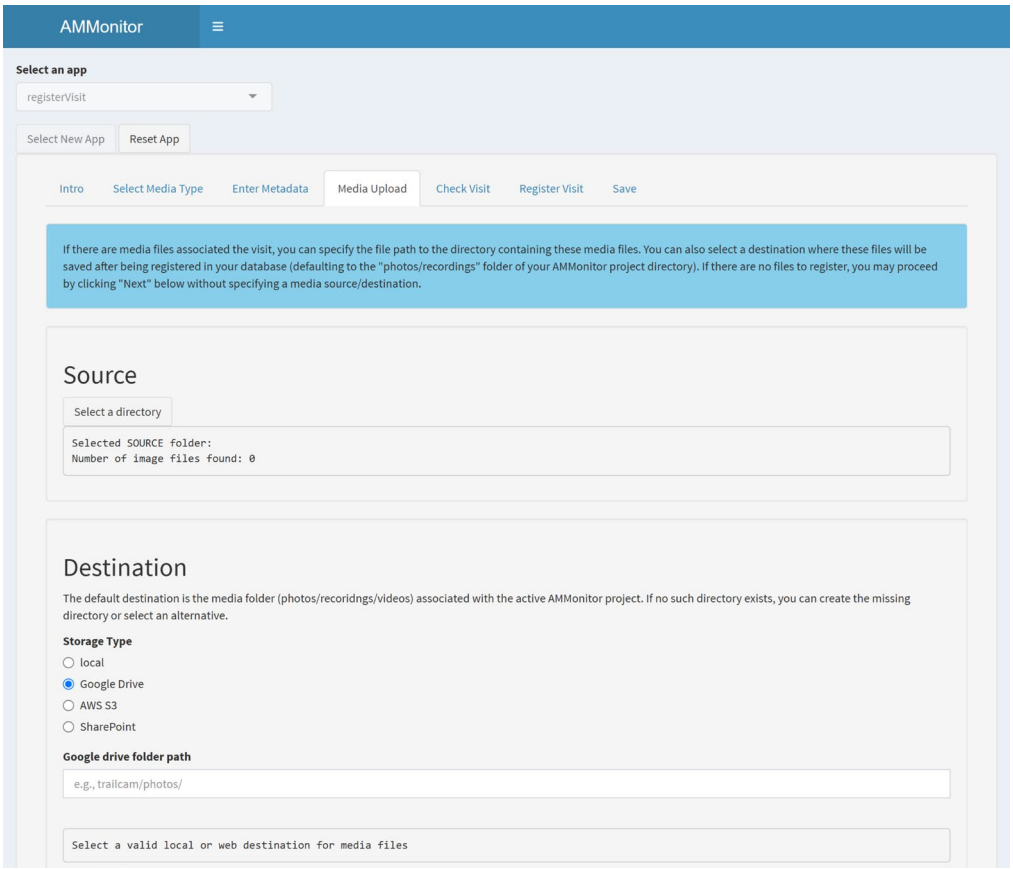

FIGURE 5 Mini-apps guide *AMMonitor* users through various analytical processes. For example, the 'registerVisit' app consists of seven tabs that allows a user to register new visits and media files into their database, with several options for external cloud storage.

FIGURE 6 'Getting started' *learnr* tutorial is one of 20 tutorials that teach users about all aspects of AMMonitor.

4 | AMMonitor SCIENCEBASE COMMUNITY

The *AMMonitor* package is fully open-source, developed by USGS to be free and available for any end-users. Consequently, we strongly encourage the practice of releasing datasets (i.e. an *AMMonitor* project) to the public domain. The path for formally releasing or publishing USGS scientific data is called a 'data release', and includes a peer-reviewed process for ensuring that releases meet high-quality standards for rich metadata. *AMMonitor* releases are hosted in the USGS ScienceBase under the *AMMonitor* ScienceBase Community at https://www.sciencebase.gov/catalog/item/6188c0c4d34ec04fc9c4f7a4, where each monitoring project has its own landing page. For example, the 'Middle Earth Wildlife Study' page can be found at https://www.sciencebase.gov/catalog/item/62b9b787d34e8f4977cc9e77. Each project contains metadata about the project itself, and data releases are 'child' items of the project that contain the data as CSV attachments and zipped media for download. The *AMMonitor* ScienceBase Community currently features projects by the US Geological Survey, the US Fish and Wildlife Service, National Park Service, Bureau of Land Management, the USDA Forest Service and state partners. Because each project uses the *AMMonitor* R package and standardized database structure, releases offer an opportunity for pooling data under a unifying framework. *AMMonitor* projects released on ScienceBase can be reconstituted with the sbRehydrate() function.

5 | SUMMARY

AMMonitor 2 provides a substantial expansion over the first iteration with a myriad of new features. *AMMonitor* is especially useful for students and users with some coding experience who desire a standardized yet flexible data management framework. The analysis ecosystem provides several important benefits:

1. *Comprehensiveness*: A one-stop data management portal that allows integration of both audio and camera monitoring.
2. *Standardization*: Field data collection is standardized across projects, ensuring high-quality metadata standards are achieved.
3. *Aggregation*: Since different *AMMonitor* projects use a common format and structure, they can easily be aggregated across individual research projects, agencies and organizations, promoting bottom-up collaboration.
4. *Persistence*: Data releases on ScienceBase under the *AMMonitor* community offer a long-term solution to the storage of monitoring data, ensuring their durability.
5. *Open source and active development*: As an open-source project, contributions from the community are welcome and development of new features is ongoing. Additional functionality to be released in future versions of *AMMonitor* will include adding the ability to display names of taxa in different languages, a streamlined process for verifying ML outputs and interfaces with other monitoring tools.

AUTHOR CONTRIBUTIONS

All authors contributed to the coding of the AMMonitor package under the direction of Therese Donovan. Laurence Clarfeld and Therese Donovan led the writing of the manuscript. All authors contributed critically to the drafts and gave final approval for publication.

ACKNOWLEDGEMENTS

We thank Hardin Waddle, Doug Hynes, and Kayley Dillon for their rigorous review of the package and package tutorials. We thank two anonymous reviewers for their thoughtful critique of this manuscript. Any use of trade, firm or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. The Vermont Cooperative Fish and Wildlife Research Unit is jointly supported by the US Geological Survey, University of Vermont, Vermont Fish and Wildlife Department, and Wildlife Management Institute.

CONFLICT OF INTEREST STATEMENT

No conflicts of interest have been declared.

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.14487.

DATA AVAILABILITY STATEMENT

The AMMonitor code has been officially released by USGS (https://doi.org/10.5066/P13MRDRV) and can be downloaded with the following code:

```
# increase the timeout
getOption('timeout')
options(timeout = 300)

# install with the remotes package
remotes::install_gitlab(
repo = "vtcfwru/ammonitor@2.1.0",
auth_token = Sys.getenv("GITLAB_PAT"),
host = "code.usgs.gov",
build_vignettes = FALSE,
dependencies = TRUE,
upgrade = "never")
```

The main repository is at https://code.usgs.gov/vtcfwru/ammonitor (Clarfeld, Tang, Huber, et al., 2024).

ORCID

Laurence Clarfeld https://orcid.org/0000-0002-3927-9411
Caroline Tang https://orcid.org/0000-0001-7966-5854
Kaitlin Huber https://orcid.org/0009-0008-2513-7456
Cathleen Balantic https://orcid.org/0000-0003-2043-0975
Therese Donovan https://orcid.org/0000-0001-8124-9251

REFERENCES

Aden-Buie, G., Schloerke, B., Allaire, J., & Rossell Hayes, A. (2023). *learnr: Interactive tutorials for R*. CRAN.

Ahumada, J. A., Fegraus, E., Birch, T., Flores, N., Kays, R., O'Brien, T. G., Palmer, J., Schuttler, S., Zhao, J. Y., Jetz, W., & others. (2020). Wildlife insights: A platform to maximize the potential of camera trap and other passive sensor wildlife data for the planet. *Environmental Conservation*, *47*, 1–6.

Aide, T., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., & Alvarez, R. (2013). Real-time bioacoustics monitoring and automated species identification. *PeerJ*, *1*, e103.

Balantic, C., & Donovan, T. (2020). AMMonitor: Remote monitoring of biodiversity in an adaptive framework with r. *Methods in Ecology and Evolution*, *11*, 869–877.

Beery, S., Morris, D., & Yang, S. (2019). Efficient pipeline for camera trap image review. *arXiv preprint arXiv:1907.06772* https://doi.org/10.48550/arXiv.1907.06772

Chamberlain, S. (2021). *Ritis: Integrated taxonomic information system client*. CRAN.

Chamberlain, S., & Hocking, D. (2023). *Rnoaa: 'NOAA' weather data from r*. CRAN.

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2022). *Shiny: Web application framework for r*. CRAN.

Clarfeld, L., Tang, C., & Donovan, T. (2025). Shinymgr: A framework for building, managing, and stitching shiny modules into reproducible workflows. *The R Journal*, *16*, 157-174.

Clarfeld, L., Tang, C., Huber, K., Balantic, C., & Donovan, T. (2024). *AMMonitor: Remote monitoring of biodiversity in an adaptive framework*. U.S. Geological Survey software release.

D'Agostino McGowan, L., & Bryan, J. (2023). *Googledrive: An interface to google drive*. CRAN.

Donovan, T., Balantic, C., Katz, J., Massar, M., Knutson, R., Duh, K., Jones, P., Epstein, K., Lacasse-Roger, J., & Dias, J. (2021). Remote ecological monitoring with smartphones and tasker. *Journal of Fish and Wildlife Management*, *12*, 163–173.

Fuller, A. K., Decker, D. J., Schiavone, M. V., & Forstchen, A. B. (2020). Ratcheting up rigor in wildlife management decision making. *Wildlife Society Bulletin*, *44*, 29–41.

Hendry, H., & Mann, C. (2017). Camelot–intuitive software for camera trap data management. *bioRxiv*, 203216. https://doi.org/10.1101/203216

Ivan, J. S., & Newkirk, E. S. (2016). CPW photo warehouse: A custom database to facilitate archiving, identifying, summarizing and managing photo data collected from camera traps. *Methods in Ecology and Evolution*, *7*, 499–504.

Kahl, S., Wood, C. M., Eibl, M., & Klinck, H. (2021). BirdNET: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, *61*, 101236.

Katz, J., Hafner, S. D., & Donovan, T. (2016). Tools for automated acoustic monitoring within the r package monitoR. *Bioacoustics*, *25*, 197–210.

Leeper, T. J. (2020). *Aws.s3: AWS S3 client package*. CRAN.

MacPhail, A., Becker, M., & Knight, E. (2024). *Wildrtrax: Environmental sensor data management and analytics to and from WildTrax*. R package version 1.3.2. https://abbiodiversity.github.io/wildrtrax/

Niedballa, J., Sollmann, R., Courtiol, A., & Wilting, A. (2016). camtrapR: An r package for efficient camera trap data management. *Methods in Ecology and Evolution*, *7*, 1457–1462.

Ooi, H. (2023). *Microsoft365R: Interface to the 'microsoft 365' suite of cloud services*. CRAN.

Polasky, S., Carpenter, S. R., Folke, C., & Keeler, B. (2011). Decision-making under great uncertainty: Environmental management in an era of global change. *Trends in Ecology & Evolution*, *26*, 398–404.

Program, B. R. (2014). *Raven pro: Interactive sound analysis software, version 1.5*. The Cornell Lab of Ornithology.

R Special Interest Group on Databases (R-SIG-DB), Wickham, H., & Muller, K. (2023). *DBI: R database interface*. CRAN.

WildEye. (2023). *TrapTagger*.

SUPPORTING INFORMATION

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

Data S1. List of current tutorials in the R package, AMMonitor 2.

How to cite this article: Clarfeld, L., Tang, C., Huber, K., Balantic, C., & Donovan, T. (2025). AMMonitor 2: Remote monitoring of biodiversity in an adaptive framework in R. *Methods in Ecology and Evolution*, *00*, 1–9. https://doi.org/10.1111/2041-210X.14487