

# iEco Lab Data Management Plan

Last Updated: 07 April 2021

## Contents

<b>Glossary</b>	<b>5</b>
<b>Section Overview</b>	<b>6</b>
<b>Pre-Project Planning</b>	<b>8</b>
Summary . . . . .	8
Project Delineation . . . . .	8
Project Roles . . . . .	8
Authorship . . . . .	9
Data Access and Ownership . . . . .	9
<b>Filesystem</b>	<b>10</b>
File Organization . . . . .	10
Data Folder . . . . .	10
Submission Folder . . . . .	11
References Folder . . . . .	11
Code Folder . . . . .	11
Example . . . . .	11
File Naming . . . . .	13
The iEco File Naming Convention . . . . .	13
Other File Types . . . . .	14
File Documentation . . . . .	15
Readme File . . . . .	15
Metadata File . . . . .	16
Changelog File . . . . .	16
File Documentation Naming and Storage . . . . .	17
Version Control . . . . .	17
Versions . . . . .	17
Versioning Software . . . . .	18

<b>Data</b>	<b>20</b>
Data Collection . . . . .	20
Field Data . . . . .	20
Lab Data . . . . .	20
Data Transcription . . . . .	21
Quality Control During Transcription . . . . .	22
Data Proofing . . . . .	25
Data Mining . . . . .	25
Data Formatting . . . . .	26
Data Tables . . . . .	26
Spatial Data . . . . .	30
Audio-Visual Data . . . . .	31
Data Backup . . . . .	32
iEcoLab Backup Strategy . . . . .	32
Final Data Storage and Sharing . . . . .	33
Final Storage . . . . .	33
Sensitive Data . . . . .	34
Sharing Data . . . . .	34
<b>Reference Libraries</b>	<b>35</b>
Zotero Libraries . . . . .	35
Reference Folder Libraries . . . . .	35
<b>Project Packaging</b>	<b>36</b>
R Packages . . . . .	36
Create a R Package . . . . .	36
Build and Call the Package . . . . .	36
R Package Files . . . . .	36
DESCRIPTION & NAMESPACE . . . . .	38
R/ . . . . .	38
vignettes/ . . . . .	39
sandbox/ . . . . .	40
Functions . . . . .	40
Vignettes . . . . .	40
Website . . . . .	41
Using Git . . . . .	41
Version Control . . . . .	41
Pushing Package to github . . . . .	41

<b>Further Resources</b>	<b>42</b>
Useful Code . . . . .	42

*“The goal is to turn data into information, and information into insight.”* — Carly Fiorina

*“It is a capital mistake to theorize before one has data.”* — Sherlock Holmes

*“Without data, you are just another person with an opinion.”* — W. Edwards Deming

*“Data is a precious thing and will last longer than the systems themselves.”* — Sir Tim Berners-Lee

---

### Purpose

A data management plan (DMP) is used to standardize data formatting, naming, organization and storing so that data are accessible, understandable and reproducible. Data are any information collected or created pertaining to a research project (e.g. data tables, analysis scripts, figures, manuscripts, and reference libraries).

The definition of a research project depends on the individual project. Research projects are defined by project leads in collaboration with a PI(s). Data management is project management. An easy guideline to follow is to use the data to define what is a project and what is a subproject. For instance, if you are using data that will be incorporated into many papers, then you would not want a project for each individual paper because then you would have multiple copies of data and keeping good data management practices across all copies would be difficult. In that case each paper would be a subproject under a larger project as defined by the data.

### Updating

The DMP is reviewed regularly because each time a new project is started, the DMP should be referenced. It is updated based on input from iEcoLab team members who are actively managing and creating research projects. All iEcoLab members are encouraged to read this DMP and discuss with their PI(s) if they prefer to follow a different DMP for one project.

You can find the most up-to-date DMP and a quick guide by following the links in the top right corner of this page.

---

# Glossary

**PI(s)** – Principle Investigator(s); Matt Helmus and/or Jocelyn Behm

**iEcoLab** – Integrative Ecology Lab at Temple University

**Research Project** – Defined by the project leads in collaboration with their PI(s). Generally defined by the data collected for each project.

**Project Lead** – The researcher who oversees the day-to-day tasks of the project.

**Data Manager** – The researcher who is in charge of curating the data for the project. This person is often the same as the project lead but can also be any researcher working on the project

---

## Section Overview

### Pre – Project Planning

- Project Delineation: In this section, guidelines for how to delineate projects and subprojects are outlined
- Project Roles: In this section, guidelines for important project leadership roles are outlined
- Authorship: In this section, reasons for the creation of authorship guidelines and a resource to use for the creation of these guidelines is provided
- Data Access and Ownership: In this section, guidelines for the accessibility and ownership of the data produced by research projects are outlined

### Filesystem

- File Organization: In this section, guidelines for how to organize your files in a directory are outlined. File organization should also be strictly adhered to.
- File Naming: In this section, guidelines for how to name files are outlined. Naming conventions should be strictly adhered to.
- File Documentation: In this section, guidelines for how to create readme and meta data files for data and changelogs for versioning are outlined. In addition, information on what to include in these documents are also detailed. At the very least a Meta Data file is required for all data tables.
- Version Control: In this section, guidelines for how to version control your data are described. Specific guidelines for version control of analysis scripts will be described in a later section about Git Version Control.

### Data

- Data Collection: In this section, guidelines for how data from the field, laboratory, and mining digital sources should be collected are described. These guidelines are not for the actual methods and techniques of generating the data but rather how to properly record data and transfer it to a data file on a computer.
- Data Formatting: In this section, guidelines for the formatting of data tables are described and includes guidelines for column headers, text formatting, table structure, and file types.
- Data Backup: In this section, guidelines for how to back up your files are outlined.
- Data Storage and Sharing: In this section, guidelines for how to store and share your finalized data files are outlined.

Reference Libraries: In this section, guidelines for how to create and manage a reference library for a project [Link to Section](#)

### Project Packaging

- R Packaging: In this section, guidelines for how to use an R Package to organize, document, and run your analyses. This includes creating a package for your analyses and vignettes so that collaborators and other researchers can easily understand your code and reproduce your analyses.

- **Git R Script Version Control:** In this section, guidelines for how to how to use Git to version control your R scripts and analysis package.

*Further Resources:* Code, papers and resources for proper data management and hygiene [Link to Section](#)

---

# Pre-Project Planning

## Summary

Pre-project planning is important so that all researchers are on the same page, know what is needed of them, and sets up proper data management practices. During pre-project planning the Data Management Plan should be reviewed and, if necessary, altered to fit the specific project needs. Everyone associated with the project needs to read and accept the Data Management Plan.

Additionally, delineation of projects and subprojects, what researchers will be responsible for, authorship guidelines, and data access and ownership should all be planned out during pre-project planning. These aspects of the project can be changed as needed during the project but only after consultation between the PIs, project lead, and data manager.

---

## Project Delineation

- **Projects should be defined based on logical groupings and can include subprojects**

For larger projects or collaborations, the individual research projects should be determined beforehand. These research projects will be the backbone for your file organization, R packaging, and determining the roles researchers will have for each research project.

The research projects can be delineated via the types of data collected, research questions, or another clear grouping. In addition, research projects can have subprojects used for R packaging and task assignment.

For example, the Caribbean Macrosystems project is one large research project with multiple subprojects. Two main subprojects (CaribMacro and CaribNet) were delineated based on the data used for each and used for R packaging. CaribMacro only uses biodiversity and island characteristic data and CaribNet, while also using data from CaribMacro, also uses detailed shipping network data not needed for CaribMacro. In this case, planned manuscripts or research questions were not appropriate since multiple papers will use these data, and because these projects share data, they are subprojects within the Caribbean Macrosystems project.

Or for graduate student projects, each chapter of a thesis/dissertation can be a separate project or subprojects.

Ultimately, project delineation is based on the preferences of the PIs and Project Lead and should be dependent on the needs and/or characteristics of the research being conducted.

---

## Project Roles

- **Project roles need to be clearly defined for each project and subproject**
- **Project roles should include at least PI point of contact, project lead, and data manager**

For each project, researcher roles need to be defined. Important roles for research projects include PI point of contact, project lead, and data manager. In some cases, a researcher may hold multiple roles. An example of this is graduate student projects where the graduate student will often be both the project lead and data manager. The roles can be the same for the entire project or different between the subprojects.



*PI point of contact:* In the iEcoLab, this person is usually either Dr. Behm or Dr. Helmus but can be both. The PI point of contact will provide the project lead with guidance in the creation of research questions, the methods, and the writing of initial drafts of manuscripts and proposals.

*Project Lead:* the researcher who runs the day-to-day operations of the project (often a graduate student or postdoc). The project lead is also in charge of managing the other researchers on the project by scheduling their hours, giving them tasks, meeting with them regularly, and checking their work.

*Data Manager:* the researcher who is in charge of curating the data produced by the project. The data manager also sets the access to the data (based on instruction from PIs) and is in charge of formatting and properly backing up the data as defined by the Data Management Plan. It is the responsibility of both the data manager and project lead to make sure all researchers are adhering to the Data Management Plan that was agreed upon during pre-project planning. Often the data manager and project lead are the same researcher for this reason.

---

## Authorship

- **Authorship guidelines should be made during pre-project planning**
- **Authorship guidelines need to be agreed upon by the PIs and Project Lead**

Agreed upon authorship guidelines help to avoid issues later on when publishing the results of the project. Setting these up before the start of the project gives the opportunity for all researchers to know what is expected of them if they want to be included as an author on certain papers. Additionally, authorship guidelines allow for practice of ethical authorship.

Typically, project leads are lead authors on papers while the PI point of contact is the last author. However, this does not have to be the case. For more information on ethical authorship see:

Weltzin, J.F., Belote, R.T., Williams, L.T., Keller, J.K. and Engel, E.C. (2006) Authorship in ecology: attribution, accountability, and responsibility. *Front. in Ecol. and the Env.* 4(8):435-441. [LINK](#)

---

## Data Access and Ownership

- **Data should be kept on a Shared Drive since PIs and University own the data**
- **Data access rules need to be agreed upon before data collection begins**

Data should always be kept in a common location accessible to the data manager and PIs. In the iEcoLab we use Google Shared Drives for this. During the pre-project planning, the PIs will create and share with you the shared drive for your folder. If you are associated with Temple University, then these shared drives must be shared to your university account. Access to these data will be controlled by the data manager and rules for which researchers associated with project have access to what data will be determined by both the PI point of contact and project lead.

Shared drives are used in the iEcoLab because the PIs and Temple University own the data. This means that the PIs need to always have access to the data and ultimately control who inside and outside the project has access to it which can be done easily as the owner of the shared drive through sharing permissions. Researchers who wish to have access and use the data should have Google Drive installed on their computer (instructions can be found [HERE](#)).

See the Data Storage and Sharing section for more information on final data storage and sharing.

# Filesystem

## File Organization

- Each project should have a separate folder with at least four subfolders:
  - data
  - submission
  - references
  - R project folder(s)

Proper file organization makes it so that data files are easy to find and their purpose easy to ascertain. Projects should be split into at least four separate folders:

- data
- submission
- references (optional – a Zotero reference library is required instead)
- code

As stated in the beginning of this DMP, a project should be defined by the data. If you are using data for multiple smaller projects (or manuscripts) then you may have subfolders in the project folder for those smaller projects. However, it is recommended that instead of having subfolders for those projects, you should have separate R projects and, therefore, R project folders, for those smaller projects thereby making a new R package for each subproject/manuscript.

---

## Data Folder

The data folder will hold all of the various data tables, pictures, GIS data, etc. you are using for the project.

*Root folder:* Only the latest versions of your data should be in the root data folder.

*Subfolders:*

- raw: the original data files. **These data files should never be edited!**
- old: versions of your data.

You can add more subfolders to better organize your data if you wish.

**Note:** this data folder is different than the one that will be in your R project folder (see the **R Packaging** section). The data folder in your R project folder (likely named something like ‘data\_raw’) will only hold the data you are using for your analyses. You can use a separate R script to import the latest versions (or a specific version) of your data into this folder overwriting the data already in that folder (see the **Useful R Code** section). To make this easier, there is a `data_import()` function being written for the `iEcoDMP` package (`iEcoDMP` is currently under construction).

## Submission Folder

The submission folder will hold all all of your files you are using for manuscript creation.

*Root folder:* Files created and used for manuscript creation and presentation

*Subfolders:*

- figures: all figure files
  - tables: all tables and appendices for the publication
  - old: older versions of figures, tables and manuscript
- 

## References Folder

The references folder will hold any important journal articles you are using for your project.

*Subfolders:* created based on subject or manuscript

**Note:** in lieu of this references folder, Zotero reference manager library should be used. See the **Reference Libraries** section for more information.

---

## Code Folder

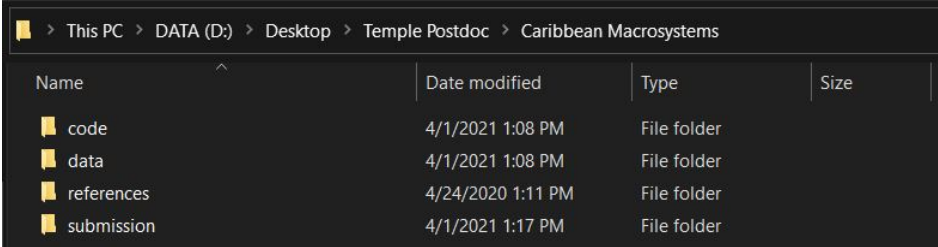
The code folder will hold all scripts and R project folder(s).

*Subfolders:*

- R Project folders (see **R Packaging** section)
  - Any scripts that are not ran in R can be in a different subfolder named for the program/language used. However, these scripts can also be in a subfolder of the R project folder to make uploading to github and version controlling with git easier.
- 

## Example

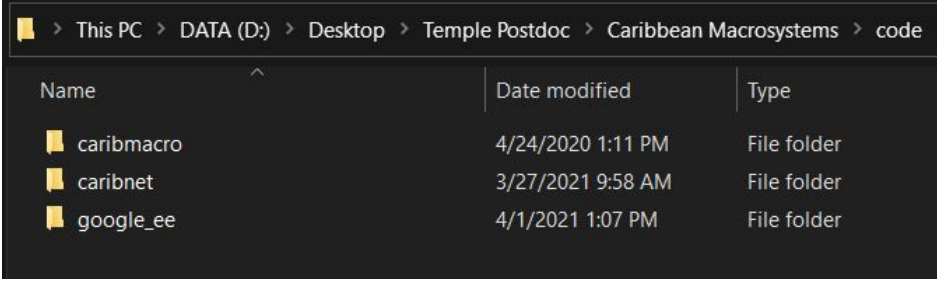
*Project Root Folder*



The screenshot shows a Windows File Explorer window with the address bar displaying the path: This PC > DATA (D:) > Desktop > Temple Postdoc > Caribbean Macrosystems. The main pane shows a table of files and folders.

Name	Date modified	Type	Size
code	4/1/2021 1:08 PM	File folder	
data	4/1/2021 1:08 PM	File folder	
references	4/24/2020 1:11 PM	File folder	
submission	4/1/2021 1:17 PM	File folder	

### Code Folder

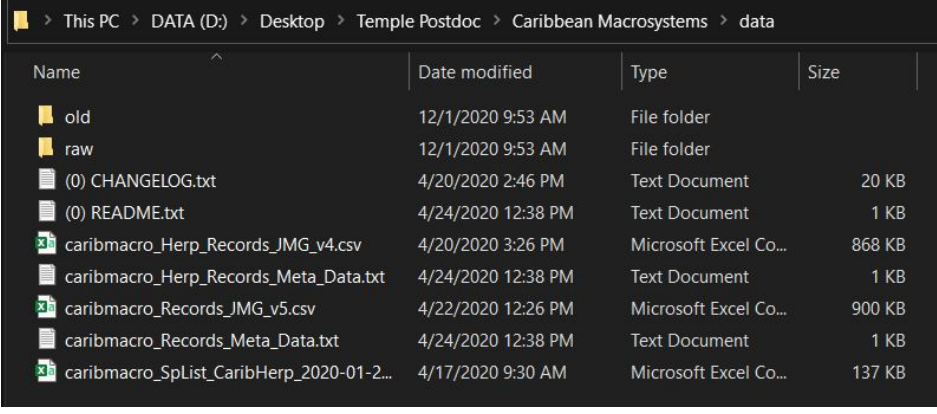


The screenshot shows a Windows File Explorer window with the address bar path: > This PC > DATA (D:) > Desktop > Temple Postdoc > Caribbean Macrosystems > code. The main pane displays a table of files and folders.

Name	Date modified	Type
caribmacro	4/24/2020 1:11 PM	File folder
caribnet	3/27/2021 9:58 AM	File folder
google_ee	4/1/2021 1:07 PM	File folder

- `caribmacro` and `caribnet` are R Project folders
- `google_ee` is java script code used for Google Earth Engine

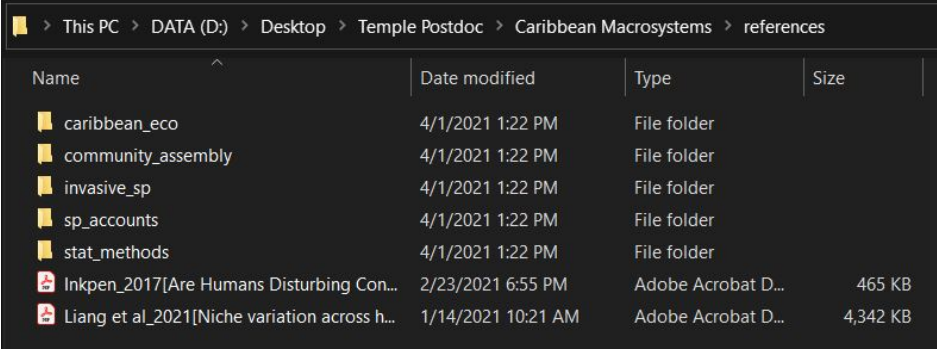
### Data Folder



The screenshot shows a Windows File Explorer window with the address bar path: > This PC > DATA (D:) > Desktop > Temple Postdoc > Caribbean Macrosystems > data. The main pane displays a table of files and folders.

Name	Date modified	Type	Size
old	12/1/2020 9:53 AM	File folder	
raw	12/1/2020 9:53 AM	File folder	
(0) CHANGELOG.txt	4/20/2020 2:46 PM	Text Document	20 KB
(0) README.txt	4/24/2020 12:38 PM	Text Document	1 KB
caribmacro_Herp_Records_JMG_v4.csv	4/20/2020 3:26 PM	Microsoft Excel Co...	868 KB
caribmacro_Herp_Records_Meta_Data.txt	4/24/2020 12:38 PM	Text Document	1 KB
caribmacro_Records_JMG_v5.csv	4/22/2020 12:26 PM	Microsoft Excel Co...	900 KB
caribmacro_Records_Meta_Data.txt	4/24/2020 12:38 PM	Text Document	1 KB
caribmacro_SpList_CaribHerp_2020-01-2...	4/17/2020 9:30 AM	Microsoft Excel Co...	137 KB

### Reference Folder



The screenshot shows a Windows File Explorer window with the address bar path: > This PC > DATA (D:) > Desktop > Temple Postdoc > Caribbean Macrosystems > references. The main pane displays a table of files and folders.

Name	Date modified	Type	Size
caribbean_eco	4/1/2021 1:22 PM	File folder	
community_assembly	4/1/2021 1:22 PM	File folder	
invasive_sp	4/1/2021 1:22 PM	File folder	
sp_accounts	4/1/2021 1:22 PM	File folder	
stat_methods	4/1/2021 1:22 PM	File folder	
Inkpen_2017[Are Humans Disturbing Con...	2/23/2021 6:55 PM	Adobe Acrobat D...	465 KB
Liang et al_2021[Niche variation across h...	1/14/2021 10:21 AM	Adobe Acrobat D...	4,342 KB

### Submission Folder

The screenshot shows a Windows File Explorer window with the address bar path: > This PC > DATA (D:) > Desktop > Temple Postdoc > Caribbean Macrosystems > submission. The main area displays a table of files and folders.

Name	Date modified	Type	Size
figures	4/1/2021 1:09 PM	File folder	
old	4/1/2021 1:09 PM	File folder	
tables	4/1/2021 1:09 PM	File folder	
caribmacro_sar_lin_manuscript_JMG_v2.docx	3/17/2021 7:43 PM	Microsoft Word D...	1,373 KB
caribmacro_sr_drivers_manuscript_JMG_v8.docx	3/10/2021 8:01 PM	Microsoft Word D...	1,206 KB
caribmacro_sr_drivers_manuscript_JMG_v8_MRH....	3/23/2021 11:09 AM	Microsoft Word D...	1,204 KB
caribnet_shipping_manuscript_JEF_v1.docx	3/17/2021 7:43 PM	Microsoft Word D...	1,373 KB

- As you can see from the file names, there are 3 different manuscript drafts in the root folder from 2 different subprojects. You can have these like it is shown or you can have separate subfolders for these manuscripts (recommended).

---

## File Naming

- **File names should be easy to understand, give information about the data, and be consistent**

Surprisingly, file naming is one of the most important aspects of data management. File names should include all the information needed for someone to know what a certain data file is and for what project it is/was used. All components of file names should be in lower case (except for abbreviations and initials) and should use “\_” instead of spaces. NOTE: file names should be unique so that no two files have the same name across the entire project.

---

## The iEco File Naming Convention

*< Project Name > \_ < Data Type > \_ < Author Initials > \_ < Version >*

*Project Name:* The first part of the file name should be the name (or agreed upon abbreviation typically same as R Package name) of the project you are working on. For example, **caribmacro** is the abbreviation (and package name) for the Caribbean herpetology macrosystems project (make sure to check with the PIs about what the name and abbreviation of your project is).

*Data type:* states what the data is. For example, for the occurrence of spotted lanternflies, the data type would be ‘occurrence’. For manuscripts and other types of files this is the name of the specific file. For example, for the manuscript on the drivers of herp species richness in the Caribbean, the data type would be ‘sr\_drivers\_manuscript’.

*Author Initials:* are the three (if applicable) initials of the person who created the data file (and for manuscripts, the person(s) who edited the file). For example, if Matthew Richard Helmus created the spotted lanternfly occurrence data then the author initials should be ‘MRH’.

*Version:* the version of the data and is used for version control. For the raw data the version should be ‘raw’, for the published data the version should be ‘final’, and for all other versions the version should be v0, v1, ... , v{number of version before final}. See the Version Control section for more information about versions.

Therefore, the file name for the second version of the occurrence data that Matt Helmus made for the SLF project would be:

*SLF\_occurrence\_MRH\_v2*

And the file name for the raw and final versions of the occurrence data Matt Helmus created would be:

*SLF\_occurrence\_MRH\_raw*

*SLF\_occurrence\_MRH\_final*

---

**Dates in file names** If the file is for a specific date, such as the photo back ups of the raw data sheets discussed in the Data Collection section or dictated observation recordings, you should include the YYYY-MM-DD date before the author initials. For example:

*SLF\_survey\_photobackup\_2020-04-24\_MRH\_raw*

If you have multiple date specific data files from a single date, add something to the name to make the two files unique (e.g. site, observer, time, survey number, etc.).

---

The order of the parts of the file name are important for the organization and sorting of files. With the iEco Lab naming convention, all of the SLF data will be grouped, and the data types within the SLF project will be grouped and ordered in ascending version or date.

---

## Other File Types

For some data, such as those created through ArcGIS, file names can have limits for the number characters in the name. In these situations, the Project Lead and the PI(s) should discuss a naming convention for those specific cases. However, one possible alternative is to house these files within a folder that is named according to the naming convention for the lab and then the individual files within the folder can have simpler names and versioning attached to them.

For other common data types the iEco Lab has created the following naming conventions:

File Type	Convention
Base Naming Convention	< <i>Project Name</i> > _ < <i>Data Type</i> > _ < <i>Author Initials</i> > _ < <i>Version</i> >
Audio-Visual Files	< <i>Project Name</i> > _ < <i>Data Type</i> > _ < <i>Date</i> > _ < <i>Version</i> >
Spatial Data	< <i>Project Name</i> > _ < <i>Data Type</i> > _ < <i>Projection</i> > _ < <i>Version</i> >
Manuscripts	< <i>Project Name</i> > _ < <i>Manuscript Name</i> > _ < <i>Author Initials</i> > _ < <i>Version</i> >
Edited Manuscripts	< <i>Project Name</i> > _ < <i>Manuscript Name</i> > _ < <i>Author Initials</i> > _ < <i>Version</i> > _ < <sup>14</sup> <i>Editor Initials</i> >

If a type of data is not in the above table, then you should use the base convention. In all of these conventions, the version is only needed if the data will be edited or modified in any way. This includes modifications within a statistical software, even if the modified data are not saved. Additionally, these conventions can be changed if needed. However, if a naming convention is changed then the new convention(s) must be recorded in a text file within the root project folder.

---

## File Documentation

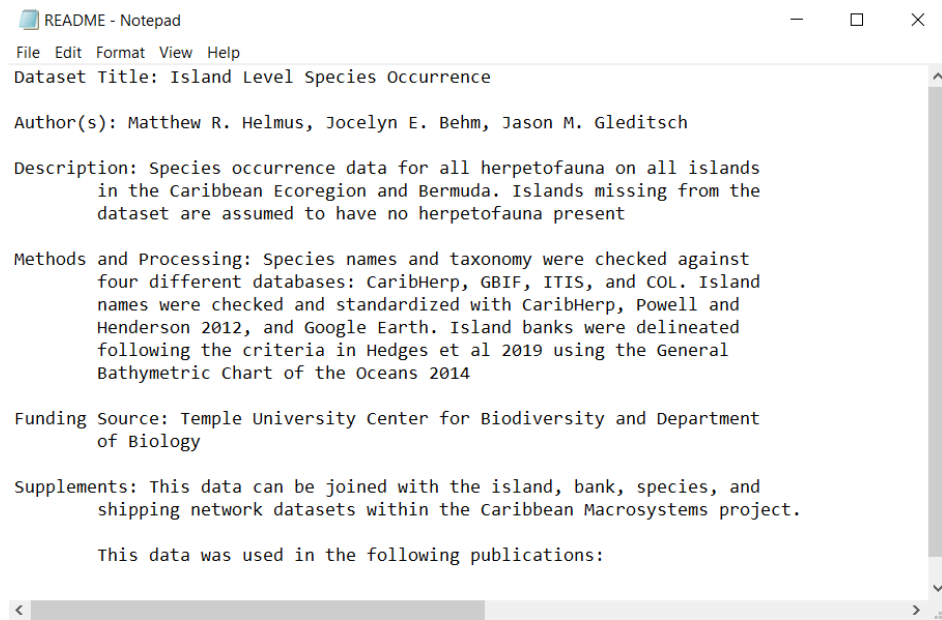
- Documentation files describe the data by stating their authors, methods, editing, and attributes
- The documentation files include Readme, Changelog, and Metadata files

File documentation tells anyone who wants to use the data, including yourself, what the data is and why, where, when, and for whom the data was collected. It will also let the user know how the data were processed. This is done through the inclusion of 3 separate text and/or CSV files. However, it is recommended that these files be text files so that they are easily distinguished from the CSV data files.

---

## Readme File

This file includes the title of the data set, the author(s) of the data, a short description of the data, any important methods used for data collection and processing, funding sources, and any other pertinent information. If your data does not include that many files and types, you only have to create one Readme file for all of the data. However, if you have many data files and/or your data consists of many types, you should make multiple Readme files for each data file (or type if you have a lot of file of the same type such as photographs). Below is an example of a Readme file for the Caribbean Macrosystems project:



## Metadata File

The file that describes the data, including data types, units of measure, factor levels, and a description of what the data represent. It is easiest to create metadata files as a CSV file then changing the file extension to .txt to make these files easily distinguished from the data files. We are currently working on a `metadata()` function in the `iEcoDMP` package to make the creation of these file easy (`iEcoDMP` is currently under construction). Below is an example of a metadata file for the bank data used in the Caribbean Macrosystems project:

Column	Type	Format/Units	Range	Missing Values	Description
bank	Categorical		78 Levels		Names of the banks used in the Caribbean species richness analyses
Area	Numerical	m^2	0.02 -- 110708.6		The land area of the bank
AOE	Numerical	m^2	0.03 -- 223980.93		Area of extent of the bank calculated by taking the area of the minimum cor
Spread	Numerical		0.00423 -- 1388.00		Spread of the islands in the bank calculated by dividing the bank area by the
Number	Numerical		1 -- 1388		Number of islands in the bank
DEM.max	Numerical	m	1 -- 3091.0		The maximum elevation of a bank determined from the SRTM Digital Elevati
DEM.mean	Numerical	m	0.5773 -- 403.4417		The average elevation of a bank determined from the SRTM Digital Elevati
DEM.sd	Numerical	m	0.0 -- 449.248		The standard deviation for the elevation of a bank determined from the SRT
iso.PC1	Numerical		-2.337 -- 8.5506		The scores of the bank along the first principle component from a princip
iso.PC2	Numerical		-3.04 -- 2.0846		The scores of the bank along the second principle component from a princip
min	Numerical	sqrt(m)	99.07 -- 1032.41		The squareroot transformed distance in meters to the nearest bank
main	Numerical	sqrt(m)	162.4 -- 1032.4		The squareroot transformed distance in meters to the mainland
source	Numerical	sqrt(m)	0.0 -- 1221.5		The squareroot transformed distance in meters to am evolutionary source
anthro	Numerical		0.0 -- 0.65361	5	The proportion of the bank that falls into the cropland, urban, and half of th
green	Numerical		0.0 -- 1.0	5	The proportion of the bank that falls into the forest, shrubland, savanna, gra
ships	Numerical		0 -- 1328		The number of ship visits to a bank from outside the bank summed over the
pop_avg	Numerical		0 -- 20237242		The human population of a bank averaged over the years 2000, 2005, 2010,

## Changelog File

A record of the changes you made to the data. They should be organized by version with each change made grouped by which version that change was made on. Changelogs can be made for entire projects, groups of data within a project, or single data files. In the case of entire projects and groups of data within a project, the changelog should be broken into sections for each data file name followed by each version of that file. Even if all of the changes and or edits to your data are made with R code, you should still have a changelog that at least provides the location of the data editing script and a brief description of what is done with that code. An example changelog for the Caribbean Macrosystem biodiversity data is below:

```
* (0) CHANGELOG - Notepad
File Edit Format View Help
#####
caribmacro_Herp_Records_JMG

v0 - created by removing non-Caribbean banks and non-herps from 'caribmacro_Records_JMG_v3.csv'
    -- removed banks = "florida", "hawaii", "mainland", "nicaragua", "okinawa and bonin isl
    -- removed record types = 'snail', 'insect', 'bird', 'mammal', 'sea.turtle'
    - created a 'taxon' column to denote if record was for an amphibian or reptile
v1 - updated the 'genus' and 'species' columns to match 'binomial.cleaned'
v2 - for records GK.0402 and GK.0032 changed bank from jamaica to pedro
    - for records GK.0029, GK.0817, GK.0030, and GK.0031 changed bank from jamaica to morant
v3 - created a bank level status column
    -- in this status column 'X' denotes a genus level id that should be removed from bank
    - for entry GK.0210 changed status from 'N' to 'E'
-----
v4 - recreated from 'caribmacro_Records_JMG_v5.csv'
    -- used same criteria as v0

#####
caribmacro_Splist_CaribHerp_2020-01-27_JMG

v0 - downloaded all Caribbean herp species from caribherp.org on 1-27-2020
    - removed unwanted spaces (random ones ant end of data point) from all columns
    - created SP_ID column for a future unique identifier for each species
      -- currently populated by NAs
    - split off the latin names from the 'SPECIES (AUTHOR(S) AND YEAR)' column into a new 'Latin
    - made new column 'Genus' for each species' genus name
    - made new column 'Species' for each species' species name
    - created all column headers except 'SP_ID' and 'GENUS' and 'SPECIES' (AUTHOR(S) AND YEAR)'
```



**Note:** that in this changelog there are multiple files with each file having its own section for each of its versions (for caribmacro\_Herp\_Records\_JMG there are 4 versions).

Also, note that the file name for the Changelog starts with a “(0)” this is to make the changelog easy to find in a folder with multiple files in it. This can be done for all of the File Documentation files to make them easy to find.

---

## File Documentation Naming and Storage

The documentation files should be kept in the data folder. It is up to you if they are kept with the data or within their own subfolder named ‘File Documentation’ that is in the root data folder. The naming conventions for these files are as follows (there should be ‘\_’ between each part):

Document Type	Naming Convention
Meta Data	< <i>Data File Name</i> > <i>_METADATA</i>
Readme Files	< <i>Project Name</i> > <i>_(0)_README</i>
Project Changelog Files	< <i>Project Name</i> > <i>_(0)_CHANGELOG</i>
Individual Changelog Files	< <i>Data File Name</i> > <i>_CHANGELOG</i>
Individual Readme Files	< <i>Data File Name</i> > <i>_README</i>

The ‘(0)’ is make those files easy to find since they will be at the beginning of the project group when the files are sorted by name.

---

## Version Control

- **Data version control should be done manually by adding the version at the end of the filename**
- **At least three versions of the data are required: raw, v0, and final**

Version control is another essential part of proper data management. With high quality version controlling you will never have to worry about making an error during your data processing because, if you do, you can always revert back to an earlier version of your data. The frequency in which you create new version will largely depend on how you process your data. You should create a new version every time you remove or add a large portion of data as well as every time you edit a large portion of data. Small changes do not require new versions until they sum up to a large portion of the data. However, all changes no matter how small need to be logged in both your file documentation (i.e. Changelog; see File Documentation section) and in the code you wrote for data processing (see [R Packaging] section).

---

## Versions

At the very least you should have 3 versions of your data:

- a raw file that you should never edit (denoted ‘raw’ in the file name)

- an initial version (denoted ‘v0’ in the file name)
- a final version that you used for your publication (denoted ‘final’ in the file name).

Any additional versions created during your data editing and processing should be denoted by v1, ... , v{number of version before final} in the file name. If you think you will have more than ten versions then put a ‘0’ in front of the version number for versions 0 through 9 (i.e. make them version v00, v01, v02, ... , v09). If you move directly from the initial version to your final version you will use for analysis, then you will only have these three versions (raw, v0, and final). However, it is good data hygiene to export several versions from your R code so that any mistakes can be easily diagnosed if the need arises.

The easiest way to do manual versioning as described above is to set data processing goals and for each goal create a new version. For instance, if you are working with spotted lanternfly occurrence data, your first data processing goal may be to check, correct, and standardize every location name. Once you do this you would then create version v1. You then may set the goal to include data from a different state (the data from the other state will have a raw file in the raw data subfolder). Once you do that you would create a new version, v2, and so on. A typical first data processing goal is reformatting the data to adhere to the guidelines set forth by the DMP (see Data Formatting section) meaning that the v0 version does not have to be formatted as the DMP states. The frequency that you make versions is up to the you, but the more versions you make the easier it will be to fix any mistakes made during data processing.

These versioning guidelines are for anything produced during a project which includes, but is not limited to, data tables, GIS data, photographic data, video data, non-R scripts, figures, and manuscripts. Versioning for R scripts will be discussed in the Git R Script Version Control section.

**Example** Here is an example of file versioning that may occur that does not follow typical data versioning: If you are working from multiple data files that will be compiled (e.g. data collected by many researchers), then the compiled data file will be your raw data and have ‘raw’ as the version in the filename. This data file will be saved in the raw subfolder of the data folder of your project and never edited (except maybe if you compile the data again). You will then copy that data file into your root data folder and edit it. These edits will be logged in any code you use AND the changelog. Once you edit a significant portion of data then you will resave the data changing the version in the filename to v1. The v0 version will then be moved to the ‘old’ subfolder of your data folder. You will then edit the v1 version. These steps will be repeated, updating the version number, until you have created your final version. In your data folder this file will still have a version number. This is in case you will continue editing these data after publication of your final version. All older versions are moved to the old files subfolder. The final version will then be saved in the data\_raw subfolder of your R project folder and have ‘final’ as the version in the filename. This is the file you will use in your analyses and upload to a data repository upon publication. If you do all of your editing with R code and do not want to export any incremental versions, then you will load the v0 version into R, edit it, and then export the final version into the root data folder and the data\_raw folder of the R project folder.

**Complex File Types** For more complicated data or filetypes that may have limits on the number of characters (e.g. GIS files), you should have those files in a folder for that specific data file and then name that folder using the file naming convention and version control the folder (or in some cases the simpler file names in the folder).

---

## Versioning Software

If you would like to use a versioning software talk to the PIs to discuss what you would like to use. However, whatever versioning method you use, everyone involved with the data processing for your project should have access to previous versions, or you need to be available as a point of contact for everyone who has data

questions (decisions about data access should be made on a project by project basis with the PIs, see the Pre-Project Planning section).

See [LINK](#) for a list of software. However, use caution since automated versioning of some data types (e.g. GIS data, large data sets, etc.) can require an immense amount of memory which renders their use extremely inconvenient. When looking for versioning software make sure the user has the ability to control when new version are made.

---

# Data

## Data Collection

This section is aimed at providing guidelines for the treatment of data while and after it is collected and before it is transcribed into a computer file. During this period, data is the most vulnerable to loss or other issues that decrease its usefulness. Therefore, proper data management during this period is absolutely essential!

---

### Field Data

- **Data should be recorded with dark pen and never erased or scratched out**

When collecting data, a dark inked pen should be used. This ensures that information will not be lost due to pencil lead being rubbed off. Additionally, pen forces changes made to the data in the field to be tracked on the page (i.e. you cannot erase). If a mistake is made, the mistake should be crossed out with a single line and the initials of the person making the change written near the change. Information should NEVER be scratched out so much that the information being changed cannot be recovered.

If you are worried about rain or water damage to your data sheets, then you should use All-Weather paper and an archival or All-Weather pen. However, these types of pens can be expensive, so a pencil can be used when necessary (with the permission of the PIs or project lead). If a pencil is used, the data recorder needs to press hard enough that the pencil writes as dark as possible. The same guidelines for making changes in the field should be followed so that the changes can be tracked.

After each day of data collection in the field, the data sheets should be legibly photographed or scanned and saved to your project Drive Folder. This provides the most basic back up that you can go to if data is lost.

For audio-visual data, information about the data, such as the date, time, study name, person taking the recording and any other information necessary for the independent comprehension of the data, should be included as a message in the frame for images or at the start of video and audio recording. This makes the file self-identifying in case that information is lost elsewhere. For remotely collected audio-visual data, this can be done immediately after the data is collected (see the Data Formatting section).

---

### Lab Data

- **Data collected in the lab should kept together in a lab notebook and recorded in dark ink**

Data collected in a lab should be collected in Lab Notebooks. Lab Notebooks can be actual notebooks and will be provided by the PI (please ask if they are not readily available). Binders should be avoided when possible in lieu of bound notebooks because pages are easily removed from or fall out of binders. If you are collecting data for the development of a product for patenting, then you are required to use a bound notebook and to review the institution's policy on lab notebooks. Since ecological research rarely produces patented products, we will not give guidelines for keeping a legal lab notebook. However, if you want more information on this, then it is recommended you review "Writing the Laboratory Notebook" by Howard Kanares ([LINK](#)).

Key points in keeping a Lab Notebook:

1. Neat and legible handwriting in dark ink; not pencil if able
2. Procedure/Study title and purpose clearly stated
3. Methods described clearly and succinctly, with errors and steps taken to correct them
4. Calculations performed neatly showing intermediate steps
5. Errors crossed out with a single line, initialed, and briefly explained
6. All pages dated at the top and numbered at the bottom

When making a Lab Notebook, make sure to leave enough room at the beginning for a table of contents. Every new procedure, experiment, notes, calculation, etc. should start on a new page with that page being the front of the right page (i.e. the odd-numbered pages). These new pages will be what is recorded in the table of contents for each procedure, experiments, set of notes, etc. For the sake of being able to easily find a specific date or page number, the date should be recorded at the right-hand side of the top of the page and the page number should be written at the right-hand side of the bottom of the page.

All information about a procedure, experiment, notes, calculation, etc. should be recorded in the Lab Notebook. This includes data recorded on or collected with a computer system. The data should be printed out and taped (never glued) into the Lab Notebook on the appropriate page. These entries should be accompanied by a brief description of what it is. If the data collected by a computer is too large to print and tape into the notebook then the name of the data file and where it is located should be recorded along with the location of any backups/copies made.

As with the collection of field data, information in a Lab Notebook should never be completely removed from the book. Mistakes, such as misspellings, should be crossed out with a single line and initialed. It is also good practice to give a short (only a few words) reason the change was made.

Lab Notebooks should **NEVER** be taken out of the lab or the project lead's possession. Ideally, Lab Notebooks should be kept in a cabinet/drawer in the lab space so that collaborators can easily find them. However, it is also acceptable for lab notebooks to be kept in the office of the project lead. To protect against Lab Notebooks being removed from the lab, digital scans or photographs of the notebook's pages should be created periodically (ideally weekly). This allows for the consultation of the Lab Notebook when not in the lab and acts as a backup of the information in the notebook.

---

## Data Transcription

- **Digitize and back up data soon after collection**

Ideally, the data should be transcribed into a computer each day after collection as well. If that is not possible, then data should be transcribed weekly. This makes transcribing data easier and reduces the chances for transcription errors for multiple reasons. First, data will not pile up, and when data piles up and transcribed all at once, errors are more likely due to rushing and fatigue. Lastly, regular data transcription helps for understanding messy handwriting in the field. The data will still be fresh in the mind of the researcher and any temporary technicians will still be around, both of which are important for deciphering handwriting.

If multiple people are transcribing data, then they should each be working with separate but identically formatted spreadsheets. This reduces the risk of someone accidentally overwriting or deleting data. It is then the responsibility of the data manager or project lead to compile this data in a reproducible way (e.g. with R script or detailed methodology). The compilation of data can occur whenever the data manager deems it necessary.

Once data is transcribed into the computer, it should be immediately backed up as described in the Data Backup section.

## Quality Control During Transcription

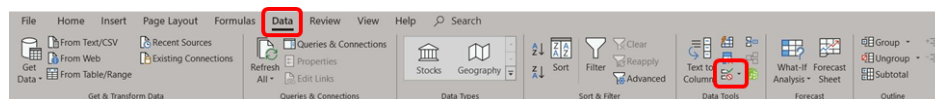
- **Spreadsheets should resemble field data sheets as much as possible**
  - The spreadsheets are then formatted by the Data Manager
- **Use column rules or data entry forms for transcription quality control**

Errors are often introduced into data during the transcription process. This could be due (but not limited to) interpreting messy handwriting, mistyping information, skipping over information, and/or overwriting preexisting data. There are a few steps that can be taken to reduce the risk of these occurring.

The first step that can be taken is to have only one person (preferably the data manager) create the data entry spreadsheets and have everyone use this spreadsheet to enter data. This makes sure that all the spreadsheets will have the same formatting (e.g. file type, column headers, number of columns, etc.). It would then be up to the data manager to disseminate these data entry spreadsheets to the researchers entering data. Additionally, if these spreadsheets are used for the transcription of data from data sheets, the data manager should make them to resemble the data sheets as much as possible, and then when the data has been entered or compiled the data can be formatted correctly (see Data Formatting section).

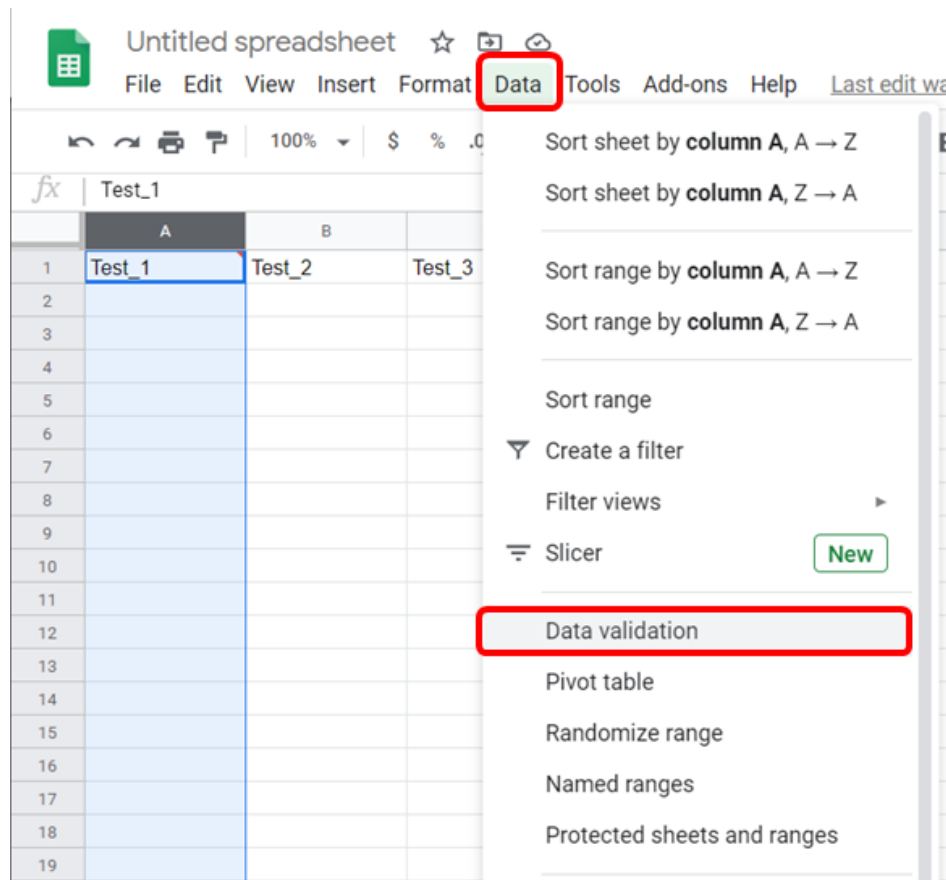
One option to reduce the errors introduced during transcription is setting column rules which reduce the risk of errors due to mistyping, skipping, or erroneous interpretation of information. The simplest way to set rules for columns within a spreadsheet is through data validation tools of the common spreadsheet software. However, these data validation tools are not required and are just recommended if there are many researchers entering data and/or you have researchers with disabilities such as dyslexia or dysgraphia working for you.

**Excel** In Excel this is done by highlighting a column (that already has a header) and then clicking the “Data Validation” icon in the “Data Tools” section of the “Data” tab as shown in the figure below.



The rules for a column can be a list of values (for nominal data), a range of values that can or cannot include decimals (for numerical data), dates with a certain range, times within a certain range, or a text string of a certain length. These rules can be made using any logical operators (i.e. “between”, “not between”, “equal to”, “not equal to”, “less than”, “greater than”, “less than or equal to”, or “greater than or equal to”). They can also ignore blank cells or return an error if nothing is entered (recommended, see Data Formatting section). For nominal data, the list of values can also be displayed as a drop down menu for each cell further controlling the entry of data. However, it is important to note that in certain software, like Excel, these rules may not be case sensitive. You also can typically specify the type of error that occurs (e.g. warning or a stop error) and the message displayed if an entry breaks these rules to give the transcriber further information.

**Google Sheets** In Google Sheets this is done in a similar fashion by clicking “Data” in the top menu and then “Data Validation” in the dropdown menu that appears as in the below figure.

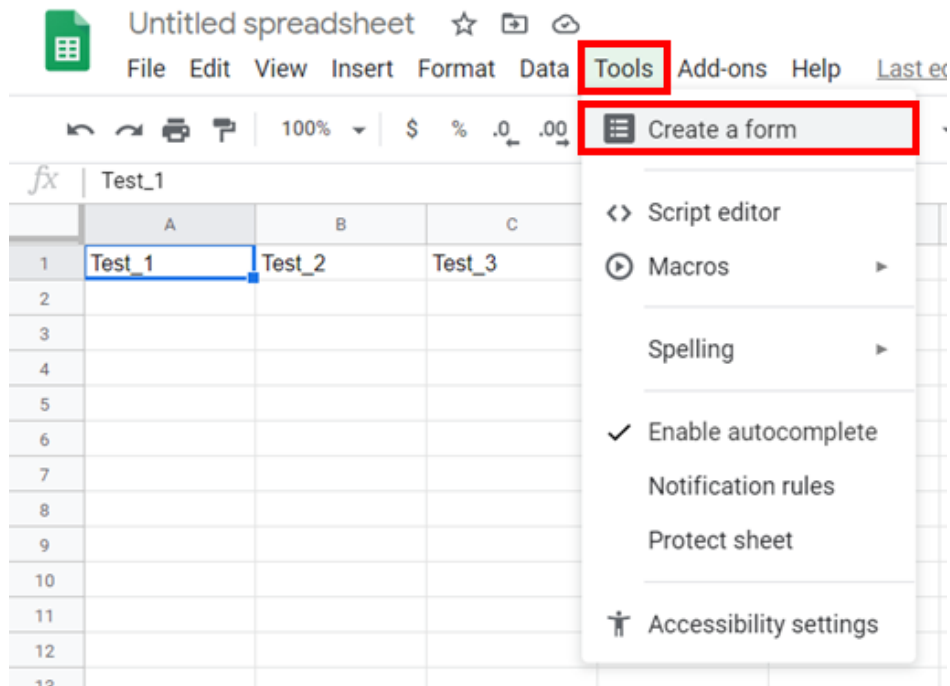


**Google Forms** Unfortunately, these rules do not protect data that has already been entered unless the data manager regularly locks the cells with data already entered in them. To make sure this doesn't happen while keeping the rules in place a data entry form should be used. This can be done in both Excel and Google Sheets. However, it is a little bit easier to do with Google Sheets.

Data entry forms for Google Sheets is done through Google Forms which can be created with the new button in Google Drive:



Or by opening a new Google Sheets and clicking the “Tools” in the top menu and “Create a Form” in the drop-down menu as in the figure below.



A new Google Form will be created that will create a new Google Sheets file. The Google Forms are organized by questions and will make a column in the new Google Sheets file for each question with a header equal to the question title. For each question you can have drop down menus for nominal data or other data validation rules similar to those in Google Sheets depending on the type of data that should be entered as the “Short answer” option for the question. For more advanced Google Sheets users more data friendly entry form can be created that may require a little bit of JavaScript coding and instructions can be found [HERE](#).

**Excel Forms** In Excel, you have to add the “Form” button to the Quick Access toolbar which can be done for the current spreadsheet or all spreadsheets. This is done by clicking “File” then “Options” and then select “Quick Access Toolbar” in the pane on the left-hand side of the new window that pops up. You will now see two selection panes: one to select commands and one that shows the commands already shown in the Quick Access Toolbar. Most likely you will have to select “All Commands” in the drop-down menu under “Choose commands from:” and once you do that you should be able to find and select “Form...”. With “Form...” selected you then click the “Add »” button and then the “OK” button. You should now see the “Forms” button (shown below; although the colors may be different) at the top of the window of your spreadsheet.



Now when researcher go to enter in data, they should highlight all by clicking Ctrl+A and then clicking the “New” button to enter a new row of data. They can ‘Tab’ through the entry fields and hit enter to add the new row of data. NOTE: if the “New” button is not clicked when the form is opened it will overwrite the existing data. Any data validation rules you set for the columns will be inherited to the entry form.

Additionally, if your spreadsheet is saved onto OneDrive or SharePoint, then you can create a form that is browser based similar to that of Google Forms.



**Relational Database Form GUI** If you are using a relational database, an entry graphical user interface (GUI) can be created in Access that can check entries against entries in the table(s) for data quality control. These types of data entry GUIs can also populate the related table when necessary based on the record you are entering.

**Schema** Schema can also be used for quality control. Schema are files that link to your tabular data files (e.g. csv files) that assign attributes to the cells of the table, and therefore, can check for inconsistencies. However, schema require programming knowledge and may not be well suited for every type of data. For more information on schema check EML's website and the File Documentation section.

---

## Data Proofing

- **A secondary check by someone who did not enter the data should be done**

After data has been entered, someone who did not enter the data should compare the hand-written data sheet(s) to the entered data. When doing this, there should be a proofing column(s) in the data file that gets checked off once the data has been checked, and has notes for corrections and/edits that were made as well as a column for the reviewer's names or initials. Again this is most important for when there are many researchers entering data but will always help to catch and correct any errors early. If there is only one researcher entering data, then having that researcher go back and recheck the data they have already entered at a later date (e.g. a week later) is a easy way to do data proofing even on a small project.

If there is too much data for the data proofing stage to be reasonably done, then a random subset of the data entered by each data transcriber should be proofed.

Further proofing will also be done during data formatting (see the Data Formatting section) and should be always be done before analysis.

---

## Data Mining

- **Work from master source list and compile and backup regularly**

Data mining is the practice of obtaining data from large sources of data, and therefore, when data mining, it is inefficient to record data in a written format. For the purposes of this section we are also including obtaining data from scientific literature, online polls, social media, and any other digital sources of data.

These types of data are often collected using a team of researchers. Therefore, it is crucial to be organized and well documented, so you do not repeat data collection and/or lose data. Working from a master list of data sources (e.g. publications, websites, surveys, etc.) allows for researchers to sign up for the mining of certain sources, and therefore, more researchers than what is prescribed by the protocols will not collecting data from the same source. Additionally, researchers collecting data should be working with separate spreadsheets to avoid accidental deletion or overwriting of data.

The separate spreadsheets will then be compiled by the data manager. Since data mining does not have paper copies of the data, the compilation of data should occur at frequent intervals so that back-ups of the data, as described in the Data Backup section, can be made during collection.

With the creation of cloud-based file storing services, the collection of data through data mining can occur completely on the cloud. By using cloud-based services researchers can mine data from sources using their

personal computers while still giving access to the data manager for frequent data compilation. Therefore, cloud-based data mining is encouraged. In this framework, the data lead will share a folder with all of the researchers on the project and the researchers would each have a sub folder containing the spreadsheet(s) they are working from. The master list(s) for the researchers to sign up for sources and subfolder(s) for the sources would be located in the root folder (i.e. the shared folder). The folder for the compiled data should not be in this shared folder to avoid accidental deletion and editing by the researchers. The only people who should have access to this folder are the project lead, data manager, and/or the PI(s) as decided by the PI(s) and project lead.

---

## Data Formatting

This section provides the guidelines for how to build and format data tables so that they are easily manipulated, edited, analyzed, and most importantly understood. In addition, we will discuss the proper formatting of other types of data as well such as spatial and audio-visual data. The formatting outlined in this section should be established well before the final version of the data is made meaning that the early files that are being manipulated/edited do not necessarily have to be formatted this way. However, it is highly recommended that the data (especially data tables) be formatted following these guidelines during the first edit (i.e. the edit(s) creating version 1).

- **You should always check your data to see if it readable with open source software**
- 

## Data Tables

- **Data tables should be saved as CSV files and be in long format**
- **Column headers are required, should be consistent, and easy to understand (see iEco Header Conventions)**
- **All text should be formatted consistently with dates in the YYYY-MM-DD format**
- **All cells should have values and if a value is missing then an NA should be entered**

All data tables should be saved as a comma delimited file (never tab delimited) such as a CSV (comma separated values) file. The reason for this is that Excel files may not be able to be easily read by researchers without Microsoft Office. This is also the same for other proprietary file types such as Access. If you are using a relational database software, such as Access, then the various tables should also be saved as CSV files. This is also true for data stored in R file types (e.g. Rdata), because not everyone knows how to use R.

Even though CSV and comma delimited text files are essentially the same, CSV files are the preferred format for data tables over text files because the .csv extension provides an easy way to sort or filter out the data files since the documentation files (see the File Documentation section) are often saved as text file (.txt extension). Additionally, CSV files are readable by most software and easy to load into most computer programs.

Data tables should always be in a long format unless a long format is too cumbersome (e.g. large community data). A long-formatted data table is a table where one row corresponds to the minimum observable unit of data (e.g. a single trial for an individual in a repeated behavioral assay) that has multiple rows for grouping factors (e.g. individual) and single columns for data (e.g. choice). However, even if long formats seem too cumbersome, long formats are better than wide for many reasons. Specifically, long formats allow for the easy conversion to other formats. Long formats also allow for the easy aggregation, subsetting, and analysis.

Below is an example of data in a long format:

Date	Site	Species	Sex	Count
04/24/2020	A	Grackle	M	2
04/24/2020	A	Grackle	F	1
04/24/2020	A	House Finch	M	7
04/24/2020	A	House Finch	F	9
04/24/2020	A	American Robin	M	4
04/24/2020	A	American Robin	F	3
04/24/2020	B	Grackle	M	0
04/24/2020	B	Grackle	F	2
04/24/2020	B	House Finch	M	10
04/24/2020	B	House Finch	F	15
04/24/2020	B	American Robin	M	4
04/24/2020	B	American Robin	F	6

Every data table is REQUIRED to have column headers. Column headers should be short and easy to understand. Avoid using symbols and abbreviations whenever possible. For example, instead of using ‘dc’ to denote the distance to Cuba, use ‘distance\_cuba’. Abbreviations are helpful, especially during analysis coding, but can lead to difficulties in comprehension. Make sure that if you use abbreviations you use common, easy to understand abbreviations. For instance, instead of ‘distance\_cuba’ we could use ‘dist\_cuba’ since “dist” is an often-used abbreviation for distance. Here at the iEco Lab we prefer you to not use abbreviations except for instances when the column header is very long (a general guideline is that headers should be no more than 15 characters and therefore ‘distance\_cuba’ is not too long). Additionally, for column headers, spaces should be replaced with an underscore (i.e. “\_”). Some analysis software and R functions cannot handle spaces in column headers, which can cause the data to erroneously load or return errors. Column header conventions should be made for individual projects with multiple data tables or for whole labs. Column header conventions help to make data easily combined for analysis and understood.

The iEco column header conventions can be found [HERE](#). These conventions can be updated as needed, so if you are creating a data table and the convention for a type of data in a column of your table has not been made, feel free to make the convention for the lab (remember, with great power comes great responsibility).

Column headers also should not include units. The units of measurement should be included in you meta data for that file (see File Documentation section). If you have multiple columns that have the same data but different units that are easy to convert (e.g. m and km) then delete one of those columns and convert in your analysis code if needed. The only exception to not having units in the header is if you have the same data with different units that are not easily converted between each other and are on very different scales (e.g. atmospheric pressure, temperature), or if for some reason you need both metric and imperial units (e.g. for making maps for US and non-US users).

The format of any text (including numbers) in your data table should be as basic as possible. The file types that should be used (i.e. csv or text files) strip all formatting from any text. This can cause issues if you use special characters, symbols, and fonts. Therefore, all special characters and symbols should be avoided. Additionally, most analysis software do not know how to read special characters or symbols.

Here is a list of acceptable symbols and the top 11 worst symbols:

#### Acceptable Symbols :

Name	Symbol
Underscore	_
Dash	-
Veritcle Line	
Semicolon	;

Name	Symbol
Period	.

*Symbols to Avoid at All Cost :*

Name	Symbol
Dollar Sign	\$
Question Mark	?
Percen Sign	%
Number Sign	#
Commercial At	@
Comma	,
Colon	:
Backslash (Reverse Solidus)	\
Quotation Mark	"
Apostrophe	'
Accent	‘

For text strings (column headers and factors with more than 1 character) avoid using capitalized letters where possible. If you do use capital letters, make sure you are consistent with how you use them. For example, if you have binomial species names and you capitalize the genus but not the species, then make sure you use that same framework throughout. In the above example the first letter of each word in the species name is capitalized throughout that column. However, a good rule of thumb to follow is to never use capitalization where possible.

Entry IDs should be included in all tables you produce. Entry IDs are unique identifiers for each entry (i.e. row) in a data table. This allows for easy tracking of specific changes you make to the data as well as allowing for the easy creation of relational keys between your data tables for easy merging. Avoid just using simple row numbers for these IDs. Good entry IDs are alpha-numeric text strings that provides some information about the entry, allows for easy sorting, and should be the same number of characters for each entry. These IDs should not be typed in with the data and be simple (few numbers and characters) or complex. It is often easiest to make IDs by concatenating multiple columns together with a separating symbol.

For example, a good entry ID for the first entry in the above table would be “20200424\_A\_02\_M” which tells us that entry is for site A, the second species alphabetically, is male, and was made on a particular survey day. We could also use that ID to sort so that the sites and species within those sites are alphabetical from the earliest survey to the last.

Entry IDs should also be unique to the data table meaning that the ID scheme should not be repeated across tables that may be later joined. The entry ID should be created at some point before the final version of the data is created. The timing at which the entry ID is made depends on the specific needs of the project and data editing. Once the entry ID is made then it is set for the entire lifetime of the data and should not be change. However, it is recommended to create the entry ID after all of the large edits to the data are made. This is because if large sections of data are deleted then you will have large gaps in the entry ID series. Do not worry about adding data after an entry ID is created since it is easy to just continue the series where you left off. Importantly, entry IDs should never be recycled even if an entry has been deleted. This could make following the change logs difficult.

Dates in data tables should be in the format of YYYY-MM-DD, and no other format should be used. In the above table the format used for the date is MM/DD/YYYY. This format is bad for multiple reasons. First,

not every culture and, therefore, their computers use that format. In many regions, the day comes before the month (i.e. DD/MM/YYYY). This can cause issues for dates like 4/5/2020. Is that April 5th or is it May 4th? Computers in different cultures will also ask that question and may just assume that the first number is the day or month depending on its default setting. The year-month-day format with dashes in between the year, month, and day is understood by all computers and software. Additionally, by having dates in the YYYY-MM-DD you can easily sort your data by date. When including years in dates they always should be 4 digits so that 2020 is not confused with 1920. If you like having the character form of months (e.g. March or Mar) in your data tables, then you should have that separate from the date in a new column.

Missing data should be handled the same throughout the data table. This includes any comment columns you have in the table. **NEVER** leave cells blank for missing data. Blank cells have an ambiguous meaning since they could mean that there was no data for that cell, the cell was accidentally deleted, or it was erroneously skipped over during transcription. Instead, use something like 'NA' to denote missing data. It is preferred to use 'NA' to denote missing data since it is read by R software in that way. No matter what is the cause of the missing data (e.g. data not recorded, data not able to be recorded, etc.), missing data should always be represented by an NA. If it is important for the project to differentiate between different causes of NAs then this should be recorded in a 'notes' or 'comments' column. If a specific value is wanted to denote a certain cause of missing data (e.g. NR for data not recorded), then that should be a decision made by the PI and Project Lead. However, be careful when reading this data into statistical software since they will not know what that value (e.g. NR) means and will likely read it as a character making the whole column a character variable.

When creating data tables, the most important thing is the data to be consistent and the nature of the data recorded in the meta data file (see File Documentation section). For example, species names should always be either only common or only scientific in a single column, and in comment columns, if a comment means the same thing, it should be kept the same (e.g. "date not recorded" is always entered, and never "date was not recorded", "date missed", "unrecorded date" and so on). Further, in comments DO NOT use commas to separate clauses. This is because in CSV files, commas are used to separate each field (i.e. column).

Lastly, make sure that there are no leading or trailing spaces in your data. This can cause some software to mess up your data or treat text strings as separate levels of a factor even if they are identical except for a trailing or leading space.

Now if we rework the above example table into the ideal formatting it will look like:

entry_id	date	site	common	sex	count
20200424_a_02_m	2020-04-24	a	grackle	m	2
20200424_a_02_f	2020-04-24	a	grackle	f	1
20200424_a_03_m	2020-04-24	a	house finch	m	7
20200424_a_03_f	2020-04-24	a	house finch	f	9
20200424_a_01_m	2020-04-24	a	american robin	m	4
20200424_a_01_f	2020-04-24	a	american robin	f	3
20200424_b_02_m	2020-04-24	b	grackle	m	0
20200424_b_02_f	2020-04-24	b	grackle	f	2
20200424_b_03_m	2020-04-24	b	house finch	m	10
20200424_b_03_f	2020-04-24	b	house finch	f	15
20200424_b_01_m	2020-04-24	b	american robin	m	4
20200424_b_01_f	2020-04-24	b	american robin	f	6

Now this table does not have any uppercase letters, has an informative alpha-numeric entry ID, the data is formatted correctly, and the headers adhere to the iEcoLab Header Conventions (i.e. 'Species' was changed to 'common' and there are no uppercase letters).

## Spatial Data

- **Spatial data should be a CSV, shapefile, GeoTIFF, or netCDF file depending on the data type**

Spatial data is complex and requires thorough documentation in order to be read properly without distortion. The actual formatting of the data itself is often dictated by the type of data they are. This means that one overarching rule for spatial may not be applicable for the formatting of spatial data. At the bare minimum, spatial data needs to have the spheroid, datum, and when applicable the projection well documented for each file. Many of the file types used for spatial data have this information included in the file itself or in auxiliary files. However, spatial data stored as a CSV often does not include this information, so it needs to be supplied in a user made documentation file (see File Documentation section).

Point and vector (e.g. polygons) data types can be stored as CSV and shapefiles. However, CSV files are better suited for point data than for vector data. For point data, the coordinates should be kept in two columns named 'latitude' and 'longitude' for lat-long coordinates or 'northing' and 'easting' for UTM coordinates. For UTM coordinates the UTM quadrat or zone needs to be documented which can be done in the CSV file in a new column if working in multiple quadrats/zones or in the file's metadata file. However, it is recommended to use lat-long coordinates for final versions of the data that will be shared.

For vector data, CSV file can be used but they can be cumbersome to work with. Therefore, shapefiles are recommended since they are readable with open-source software like QGIS. Shapefiles are nice in that they automatically create the necessary documentation for geographic systems to properly read them. However, this is done through multiple auxiliary files which can be easily lost if one is not careful with file management. It is recommended that each shapefile and its auxiliary files are kept in individual folders which can be turned into a zip folder for easy sharing. Keeping these files together also aids in version control when creating and editing vector data and avoids the need to long file names for which some geographic software has limits. Point data can also be saved as shapefiles if wanted but CSV files are often smaller. KML files (i.e. Google Earth files) should be avoided for the storing and sharing of these kinds of data since it is difficult to read outside of Google's systems.

Raster data should be saved as a GeoTIFF file(s). GeoTIFFs are tif images that are georeferenced and are tagged with the important georeferencing information (i.e. spheroid, datum, projection, etc.). These types of files can be read by most software and programming languages and is in continual development meaning that any compatibility issues for common software or programming languages will most likely be addressed in future releases. GeoTIFF files can support both single and multi-band raster data (e.g. RGB aerial images). For multi-band raster data, the individual bands can be stored as individual tif images or can be combined into one file. If you store the bands as individual files, then you should follow the same folder organization that was described for shapefiles in the above paragraph.

Spatial data with high dimensionality such as raster data with bands for different data types should be saved as netCDF files. This file type is a scientific standard used for storing and sharing multidimensional spatial data and recommended by NASA and other government agencies and can be read by most programming languages and open source programs (i.e. panoply). However, they can be difficult to work with so these types of files should only be used if needed due to the dimensionality of the data making the use of the other file types too cumbersome.

For more information on proper spatial data management see:

Ramapriyan, H. K., and P. J. T. Leonard. 2020. Data Product Development Guide (DPDG) for Data Producers version 1. NASA Earth Science Data and Information System Standards Office, 9 July 2020. [LINK](#)

## Audio-Visual Data

- **Avoid proprietary formats and include data information in the recording or picture file**

Audio-visual data should only be stored in common file types that are able to be read by most software. These file types include MP3 and WAV for audio files, JPEG and TIFF for images, and MPEG formats (i.e. MPEG and MPEG-4: .mpg and .mp4 file extensions respectively) for video.

For Images, JPEG is recommended for photos taken in the field or lab if the loss of quality will not diminish the data in the photograph (e.g. high contrast images, animal images, etc.). JPEG files are compressed and the more they are manipulated the lower the quality of the image will be. However, because they are compressed, they are usually smaller in size and therefore take up less memory. Additionally, JPEG are readable on any operating system, can have metadata embedded in the file, and the loss of quality due to compression can be mitigated with high quality cameras. Other file types for images such as RAW file types, PNG, GIF, etc. should be avoided due to their compression techniques (for all but RAW) and the fact that they are more difficult to read, often requiring proprietary software.

The TIFF file type should largely be reserved for images where the loss of quality will greatly diminish the data in the image. TIFF files do not lose information when they are compressed, manipulated, copied, re-saved, etc. This makes them great for figures, maps, and other complex images. However, because TIFF files do not lose information during compression, they can be quite large and take up a lot of memory. When using TIFF files for figures they should be saved at least at 300dpi (i.e. 300 dots per inch) since most publishers require a figure to be a minimum of 300dpi resolution (some require 600dpi).

Like images, there are many file formats available for video. However, many of them are proprietary and require specific software to read. MPEG is readable by most software and retains video quality while keeping file sizes relatively small. Therefore, MPEG formats are recommended over MOV or WMV which were made specifically for Apple and Microsoft products, respectively. Some digital video recorders (DVRs) record video in H.264 and should be avoided when possible. H.264 video formats are typically used for constant recording because the file sizes are extremely small which can be advantageous for 24hr monitoring of, for example, a nest. However, the H.264 video format requires expensive software to read, and even though there is a free program to work with these files, it is extremely difficult to work with.

Often the type of audio-visual file format is dictated by the hardware used to record it and the quality of the recorder trumps the recommended file types. Therefore, it is important to make sure that you are able to read the file types that specific hardware records before purchasing the hardware.

Outside of file type, the formatting of audio-visual data is dependent on the use of the data. However, whenever recording audio-visual data, you should take the appropriate steps to make sure the recordings and/or images are of the highest quality possible. There are many tutorials online for how to use various hardware and the best practices for recording certain types of data.

A few practices that are recommended for all types of audio-visual data is to include the necessary information to comprehend the data in the file and in a separate data table that included the file name in a column. The bare minimum amount of information should be recorded is the date, time, study name, person taking the recording. However, any other pertinent information for the comprehension of what the data show or is for should also be included. This information can take the form of a label in the frame of a photograph, a message board at the start of video, or a spoken message at the start of an audio recording. For data that are remotely taken (e.g. camera trap images/videos), this information can be tagged to the video or image through many kinds of image or video editing software. To do this, we recommend Adobe Bridge for tagging any audio-visual data file with pertinent information because it is free to download and easy to use for editing a media file's embedded metadata. The information in the data table and the labels, messages, and tags are important since it protects from information loss during the editing, coping, and/or sharing of data.

## Data Backup

- Data should be backed-up weekly using the 3-2-1 strategy:
  - 3 copies of your data
  - 2 copies stored locally on different devices
  - 1 copy stored off site (i.e. on the cloud)
- Data should ideally be backed up weekly and at least monthly

By far the most important aspect of proper data management is backing up your data files. Without proper backup practices you risk losing all of your data if something happens to your computer. Additionally, because you are a part of a larger research group, if something happens to you, such as moving on to a new research group, the PIs and researchers will still have access to your data that you collected for them.

---

### iEcoLab Backup Strategy

It is required that you practice the 3-2-1 backup strategy here in the iEco Lab. The 3-2-1 backup strategy states that you should have at least 3 copies of your data, 2 of those copies should be stored locally, and 1 copy should be stored off site.



Local storage needs to be stable and can include your desktop computer AND an external hard drive or flash drive, which is preferred; or two different computers; or multiple external hard drives. These two copies



should be on two separate devices (i.e. do not have these two copies on the same computer or hard drive). Additionally having them separate from your primary computer will help you get into a physical habit of backing up data. Local storage does not include locally synced folders such as Dropbox, Box, and Google Drive for Desktop. These folders are still linked to the cloud-based versions of the data and, therefore, are not stable.

Off site storage is a cloud-based storage system. Here at the iEco Lab we use Google Drive for this. You should have been or will be invited by the PIs to a shared drive they created for your project. All of your data needs to be backed up to this shared drive. If you are already working out of a shared google drive or folder, ownership of the drive or folder should be transferred to the PI(s) of your project.

However, we would like you to use the copies stored in the shared drive as your primary copies that you edit and work with. That would mean that the copies on your desktop computer and external hard drive would be your backup copies. The location of these backup files needs to be recorded in the project folder.

The person in charge of backing-up your project's data should be the data manager, and proper backup protocols require your data to be backed-up to the various places you have stored copies at least weekly. A calendar reminder to for this.

There are free and paid software that will back up files for you on a regular basis. If you would like to use these then make sure the software name, frequency of backup, locations of back-ups, and a link to the software information page is recorded in the project folder.

---

## Final Data Storage and Sharing

- **Final data storage should be on an online data repository**
- **Data sharing permissions is determined by the Project Lead and the PI(s)**

---

## Final Storage

Once your data are finalized please make sure the final files are uploaded to the shared drive for your project in the R project folder. Upon leaving the lab, any researchers that authored a data file are required to meet with the PIs to make sure the final data are stored in a location the PIs can access and what (if any) repository the data should be uploaded to. The PIs will then back it up as needed after this.

When your paper(s) that use data you created are published, you should upload just the data used for that paper to an online repository. Many journals require this to be done when they accept your paper for publication. If the journal has a preferred repository to be used or has a deal for free storage with a certain repository, then please use that repository. Otherwise, we would like you to use Dryad for your data (discuss this with the PIs to be sure and determine how payment will be processed). Unfortunately, github and gitlab are not appropriate to use as an official data repository because they are not stable and therefore is not accepted by most journals and funding agencies.

## **Sensitive Data**

Sensitive data, such as location data for endangered species at risk from poachers or personal information of human subjects, should be cleaned of any information that could be used for unwelcome purposes before publicly sharing. This may include changing site names, removing GPS points, removing identifying information of human subjects, and so on. Before sharing any data, make sure to check with the PIs to determine the sensitivity of the information within the data you are sharing.

---

## **Sharing Data**

Sharing the final or any version of the data files that you create or work with is strictly forbidden without the written consent of the PIs. Written consent can be just an e-mail response but refrain from only using verbal consent. The PIs, and in some cases the University or funders, are owners of the data, and therefore, the data should not be shared with anyone without their consent.

---

## Reference Libraries

- Reference libraries should be created through Zotero. Contact Dr. Helmus for access
- 

### Zotero Libraries

A single reference library should be made for each project. Here at the iEco Lab, we would like you to use Zotero to build your reference libraries. Zotero is nice since it links with Microsoft Word allowing for the easy conversion of citations and references between different styles. In the iEco Lab, you will need to contact Dr. Helmus for access to a Zotero library that has unlimited size. The PI(s) and Project Lead will then determine which researchers working on the project will have access to the library.

For more information on Zotero reference libraries watch VIDEO (1.75hrs) or visit WEBSITE

---

### Reference Folder Libraries

However, libraries can also be built through folders broken up by subject, manuscript, or some other logical category within the ‘references’ subfolder of the project folder. PDF files of papers should be named in the following way:

*< Author(s) > \_ < Year > [< BriefDescription >]*

For papers with more than one author follow in text citation rules. For example:

*Weterings et al\_2019[House geckos and epidemics].pdf*  
*Perella and Behm\_2020[Impact and Spread of Carib Exotic Geckos].pdf*  
*Janzen\_1985[On Ecological Fitting].pdf*

---

# Project Packaging

## R Packages

- R packages should be made for better storage, communication, and sharing of code and analyses

When organizing your analyses for a research project, it's important to maximize two aspects: reproducibility and ease of sharing. This will make the life of your collaborators, and crucially the life of “future you”, much easier. R packages provide a great tool to create collaborative and reproducible code. An R package bundles together the data your project requires, custom-made functions that can be used throughout your code, vignettes that describe your analyses step-by-step, and a thorough documentation that helps you and others to reproduce your analyses in the future. In practice, creating an R package boils down to:

1. organizing your project folders and files in a clever and consistent way
2. documenting thoroughly your steps and the tools you use.

---

## Create a R Package

UNDER CONSTRUCTION

---

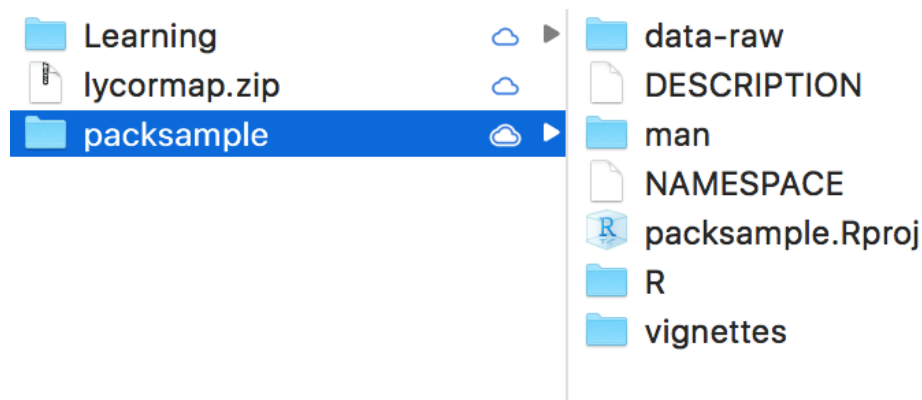
## Build and Call the Package

Once you have populated your package (even just a little), you are ready to build it. This can be done directly from RStudio, inside the R project containing your package. If this is your first package even, go to Build > Configure Build Tools... Here you should tick the checkbox *Generate documentation with Roxygen* and then tick *Build & Reload* in the window this prompts. You can now build your package. Go to Build > Build and Reload (or Clean and Rebuild, depending on the version). This will build your package and install it. You can now load it using `library(packagename)` and start calling your functions.

---

## R Package Files

The basic structure of an R package is simple. Above is an example of a mock package called “packsample”. Here we will take a quick look at the components of this package (and of any package), some of which will be explained more in depth later on.



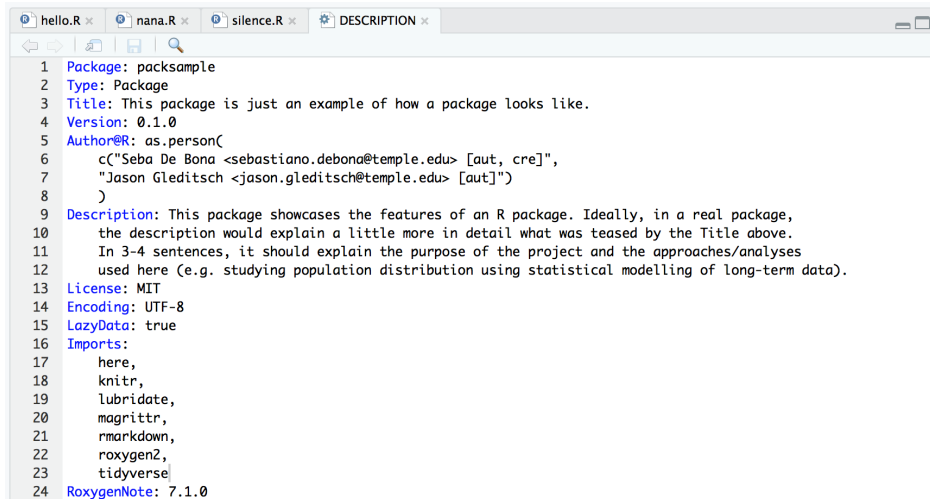
- **DESCRIPTION**: the main text file containing metadata on your R package (e.g. who created the package, who helps maintain it, what are the CRAN packages required for the package to work etc.)
- **NAMESPACE**: contains the names of the functions in your package. We will never manually edit this file, which will be automatically updated using Roxygen (a ready-made package for package development).
- **R/**: a folder containing individual R scripts where the package functions are stored and described.
- **man/**: a folder containing the documentation for our functions which will be used to generate the helpfiles. These will travel with the package and can be accessed by typing “?name\_of\_the\_function” in the console (substitute “name\_of\_the\_function” with the actual name of your function). This folder, like **NAMESPACE**, will be automatically populated through Roxygen.
- **data-raw/**: folder containing all data files (in .csv or .txt format) that are necessary for your analyses. It’s important to avoid naming this folder “data”, since a “data” folder is used to store .Rda files that are shipped with the package (similarly to the data `mtcars` contained in the package “`datasets`”).
- **vignettes/**: this folder is the heart of your analyses. It contains a series of .Rmd vignettes that describe every step of the project, from data tidying and wrangling, to running models or simulations, to displaying the results through tables and plots. The Rmarkdown allows you to create a very thorough documentation, with text interspersed between coding chunks. Also it can be used to generate automatically documentation for your package using “`pkgdown`”.
- **packsample.Rproj**: this is the RStudio project associated with our R package. It contains some meta-data that allows you to always have the same interface every time you work on our package. Open this file anytime you want to work on your package, and it will automatically take you to the right folder and open up the vignettes, scripts and files you last worked on.

When creating a package in RStudio some of the files in the package folder are created automatically. In addition, some files and folders will be automatically populated.

Whether you’re starting your project from scratch, or turning a collection of R scripts into a package form, the process is pretty much the same. To create an R package, open a new RStudio session, go to File > New Project > New Directory and then select “R package”. In the prompted window, you can decide where to place your new package folder and, crucially, a name for your package. For tips on naming and many other useful guidelines, please see Hadley Wickham’s awesome R packages book (freely available [HERE](#)). Also, you can tick the box “create a git repository” for version control (see the section Version Control using git, below). Creating the package this way will automatically create a **DESCRIPTION** and a **NAMESPACE** file, and **R/** and a **man/** folder, and an **.Rproj** file. You can then create a **data-raw/** and **vignettes/** folder manually. All information on how to create and populate an R package can be found in Wickham’s book. Below are some quick tips on how to fill and organize the files and folders in your package. By following the tips below you will have a package structure. Making it into a working package will take very few extra steps.

---

## DESCRIPTION & NAMESPACE



```
1 Package: packsample
2 Type: Package
3 Title: This package is just an example of how a package looks like.
4 Version: 0.1.0
5 Author@R: as.person(
6   c("Seba De Bona <sebastiano.debona@temple.edu> [aut, cre]",
7     "Jason Gleditsch <jason.gleditsch@temple.edu> [aut]")
8 )
9 Description: This package showcases the features of an R package. Ideally, in a real package,
10 the description would explain a little more in detail what was teased by the Title above.
11 In 3-4 sentences, it should explain the purpose of the project and the approaches/analyses
12 used here (e.g. studying population distribution using statistical modelling of long-term data).
13 License: MIT
14 Encoding: UTF-8
15 LazyData: true
16 Imports:
17   here,
18   knitr,
19   lubridate,
20   magrittr,
21   rmarkdown,
22   roxygen2,
23   tidyverse
24 RoxygenNote: 7.1.0
```

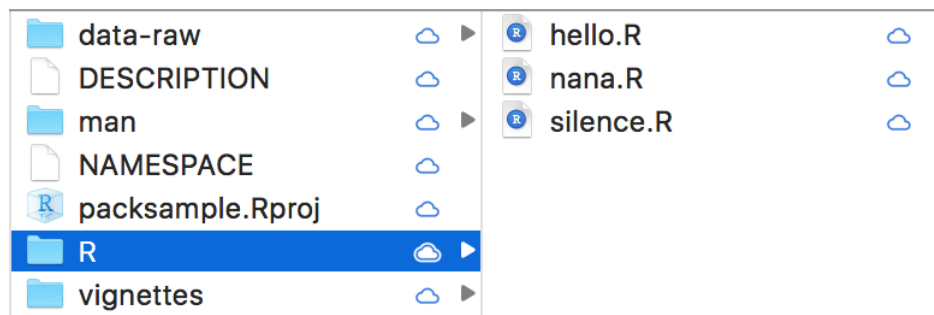
The DESCRIPTION file is made of headers that define different fields. As soon as you create a package, you should update this file. Make sure you fill the *Title* field (one sentence to explain what the project/package does), the *Description* field (a few sentences to expand upon the title) and the *Author@R* field (with your name and the names of your collaborators; “aut” and “cre” defines the roles as Authors and Creator of the package). A key component of the file is the list of Imports, which details all CRAN packages your project (and therefore your R package) depends on. It’s good to keep this up to date anytime you add new dependencies to your project.

As mentioned above, the *NAMESPACE* file will be automatically updated by Roxygen. Just type into it `# Generated by roxygen2: do not edit by hand` and save the file.

---

## R/

This folder contains your custom functions, stored as individual .R scripts. Below is an example of (too) few functions.



Each function has the same structure: a header defining its documentation (where each line starts with `#'`), and a body containing the code that defines the function (as below). The first line of the header is

translated into a short title of the function. The text that follows, separated by an empty line, describes more in detail what the function does. The different headers starting with the at (@) symbol represents parts of the documentation: `@params` defines the arguments of the function; `@return` the output of the function; `@examples` some code to be used to understand how the function operates, etc. Finally, the `@export` is necessary to ensure the function's documentation is written out.

```

1 #'Evaluate if a vector is empty
2 #'
3 #'\code{not_all_na} takes a vector and checks whether at least one element
4 #' is not an \code{NA}. This is particularly useful when selecting non_empty
5 #' columns of a data frame or tibble.
6 #'
7 #'@export
8 #'
9 #'@param x A vector to be evaluated.
10 #'@return A logical value (\code{TRUE} or \code{FALSE}) that answers the question
11 #' "is any element of the vector not an NA.
12 #'@examples
13 #'x1 <- c(0,1,NA,5)
14 #'not_all_na(x1)
15 #'x2 <- rep(NA, 5)
16 #'not_all_na(x2)
17
18
19 nana <- function(x){
20   any(!is.na(x))
21 }
22

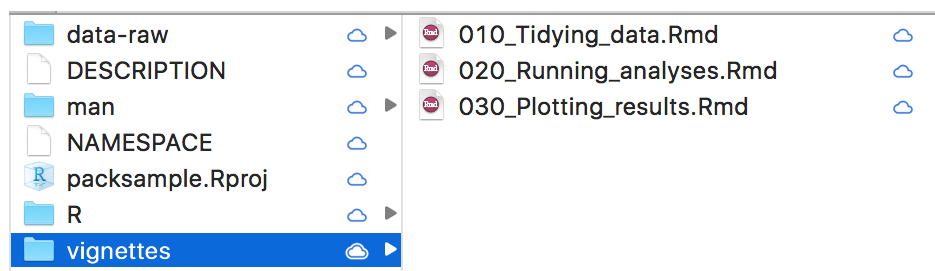
```

## vignettes/

The bulk of a typical CRAN package (like `dplyr` or `lme4`) is made by functions and their documentation. This is due to the fact that most published packages provide a toolkit for R users to wrangle data, run analyses, or visualize data and results. In the case of a private R package for a project, like the one we are describing here, the bulk of the package will be in the form of rmarkdown scripts to tidy data, run analyses, and show the results with pretty plots. The *vignettes/* folder should contain all scripts (in Rmarkdown format) describing and running the analyses. The scripts should be named using the following convention:

*< Number > \_ < Short informative name > .rmd*

starting with a consecutive numbering, followed by a descriptive brief name (separated by underscore (i.e. "\_") and without spaces).



Numbering the scripts in chronological (and logical) order ensures anyone can access your scripts and know where to start and how to proceed. Moreover, it ensures the scripts are run in the correct order when they are automatically run by pkgdown (which we won't go into details about here). Please, make sure you comment each script thoroughly. At last half of your rmarkdown scripts should be in the form of text, with each chunk of code well explained. Always ask yourself the question: "will someone accessing my package with no prior knowledge of it understand what's going on here?". If the answer is "maybe", add more comments. Another good practice is to keep a consistent structure to each vignette. The script should be articulated in three parts:

1. Aim and setup: where the purpose of the vignette is described, the required packages are attached, and the necessary data and/or objects are loaded.
2. Main body: where the actually coding happens (and is thoroughly described).
3. Saving: where all the objects to be required by future vignettes are saved. All saved objects can be stored inside the vignettes folder, or in additional folders "parallel" to the vignettes folder (such as *results/* and/or *figures/*).

---

## sandbox/

This last folder (which you will have to create by hand) becomes extremely useful if you're transforming an existing project into an R package. A good way to do this is to start by creating a new package and then migrating your code, piece by piece, into your package. While you do that, you can copy all of your old project scripts (whether they are in .R or .Rmd format) to your sandbox folder. Then proceed to transform them into either functions (that will go into the *R/* folder) or vignettes. As you transform your old code, remove it from the sandbox. This folder is also useful to host code that has become obsolete or useless, but might be recovered later, or scripts that have not made it into the mainstream of your project yet, but might in the future.

---

## Functions

UNDER CONSTRUCTION

---

---

## Vignettes

UNDER CONSTRUCTION

---

---



## Website

UNDER CONSTRUCTION

---

---

## Using Git

UNDER CONSTRUCTION

---

## Version Control

Version controlling enables the archiving of every change made to the vignettes, functions, and metadata of your package. You should always version control your code, and it's very easy to version control your package using git. Once git is installed, RStudio provides a very intuitive interface to it. Please refer to Hadley Wickham's R packages for a quick guide to using git in conjunction with an R package (in the Best practices section at the end). A good set of tips and guidelines can also be found on the RStudio support page [HERE](#).

---

## Pushing Package to github

UNDER CONSTRUCTION

---

## Further Resources

Borer, E.T., Seabloom, E.W., Jones, M.B. and Schildhauer, M., 2009. Some simple guidelines for effective data management. *The Bulletin of the Ecological Society of America*, 90(2), pp.205-214. [LINK](#)

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L. and Teal, T.K., 2017. Good enough practices in scientific computing. *PLoS computational biology*, 13(6). [LINK](#)

Ramapriyan, H. K., and P. J. T. Leonard. 2020. Data Product Development Guide (DPDG) for Data Producers version1. NASA Earth Science Data and Information System Standards Office, 9 July 2020. [LINK](#)

British Ecological Society: Guide to Reproducible Code

Stanford: Data Best Practices

## Useful Code

A web page with useful code can be found [HERE](#)

---

---

---