Survey in Ecological Data Collection Tools & Products

Dr. Katharyn Duffy, Dr. Ben Ruddell

2020-07-31

Contents

Preface

Welcome to our live INF550 course textbook. This live text is a combination of materials, videos, example code, and assignments.

Acknowledgements

As with all open source data and tools, this book is built upon the shoulders of giants who came before me. Special thanks to Megan Jones and Donal O'Leary at Battelle for their support in pulling NEON materials.

Chapter 1

Ecoinformatics Tools

As an Ecoinformatician you *need* to be able to:

1. Pull data from Application Programing Interfaces (APIs)
    - More on this in Chapter 2
2. Organize and document your code and data
3. Version control your code to avoid disaster and make it reproducible
    - For you, your collaborators, and/or the wider community
4. Push your code up to public-facing repositories
5. Pull other's code from public repositories.

More thoughts on the benefits and power of reproducibility can be found here

To be successful, both in this course and in your careers you will need these skills. This is why **they are a requirement** for this course. If you already using these skills on a daily basis, fantastic! If some of this sounds a little greek, I have placed lesson links throughout this chapter so that you can build these skills and be successful in this course.

1.1 Pre-Course Skills & Setup

For the purpose of this course we will largely be using the following tools to access, pull, and explore data:

1. R & Rstudio
2. Git, GitHub, & Atom.io
3. Markdown & Rmarkdown

As such we will need to install and/or update these tools on your personal computer *before* our first day of class. While we chose R for this course, nearly all of the packages and data are fully available and transferable to Python or other languages. If you'd like to brush up on your R skills I highly recommend Data Carpentry Boostcamp's free R for Reproducible Scientific Analysis course.

1.1.1 Installing or Updating R

Please check your version of R. You will need R 3.6.0+

How to check your version in R or RStudio if you already have it:

```
> version

               _
platform       x86_64-apple-darwin15.6.0
arch           x86_64
os             darwin15.6.0
system         x86_64, darwin15.6.0
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```

If you don't already have R or need to update it do so here.

1.1.2 Windows R/RStudio Setup

After you have downloaded R, run the .exe file that was just downloaded Go to the RStudio Download page Under Installers select RStudio X.XX.XXX - e.g. Windows Vista/7/8/10 Double click the file to install it Once R and RStudio are installed, click to open RStudio. If you don't get any error messages you are set. If there is an error message, you will need to re-install the program.

1.1.3 Mac R/RStudio Setup

After you have downloaded R, double click on the file that was downloaded and R will install Go to the RStudio Download page Under Installers select RStudio 1.2.1135 - Mac OS X XX.X (64-bit) to download it. Once it's downloaded, double click the file to install it. Once R and RStudio are installed, click to open RStudio. If you don't get any error messages you are set. If there is an error message, you will need to re-install the program.

1.2 Linux R/RStudio Setup

R is available through most Linux package managers. You can download the binary files for your distribution from CRAN. Or you can use your package manager. e.g. for Debian/Ubuntu

```
run sudo apt-get install r-base
```

and for Fedora

```
run sudo yum install R
```

To install RStudio, go to the RStudio Download page Under Installers select the version for your distribution. Once it's downloaded, double click the file to install it Once R and RStudio are installed, click to open RStudio. If you don't get any error messages you are set. If there is an error message, you will need to re-install the program.

1.2.1 Install basic packages for this course

You can run the following script to make sure all the required packages are properly installed on your computer.

```
# list of required packages
list.of.packages <- c(
  'data.table',
  'tidyverse',
  'jsonlite',
  'jpeg',
  'png',
  'raster',
  'rgdal',
  'rmarkdown',
  'knitr'
)

# identify new (not installed) packages
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

# install new (not installed) packages
if(length(new.packages))
  install.packages(new.packages,
                   repos='http://cran.rstudio.com/')

# load all of the required libraries
sapply(list.of.packages, library, character.only = T)
```

Note: On some operating systems, you may need to install the Geospatial Data Abstraction Library (GDAL). More information about GDAL can be found from

here.

1.3 Installing and Setting up Git & Github on Your Machine

For this course you will need: 1. Git installed on your local machine 2. Very basic bash scripting 3. A linked GitHub account 4. To link RStudio to git via RStudio or Atom.io

As we will be using these skills constantly, they are a *pre-requisite* for this course. If you don't yet have these skills it's okay! You can learn everything that you need to know via the following freely available resources:

*The Unix Shell

*Version Control with Git

*Happy Git with R

If you are learning these skills from scratch I estimate that you will need to devote ~4-6 hours to get set up and comfortable with the various workflows. Alo remember that I have code office hours every week and that Stack Exchange is your friend.

1.4 Installing Atom

Atom.io is a powerful and useful text editor for the follwng reasons:

1. It is langugae agnostic
2. It fully integrates with git and github + You can use it to push/pull/resolve conflicts and write code all in one space.

1.5 Linking RStudio to Git

Happy Git with R has a fantastic tutorial to help you link Rstudio-Git-Github on your local machine and push/pull from or to public repositories.

1.6 How we will be Conducting this Course

At the end of each chapter you will find a set of **Exercises**. At the end of the assigned chapter you will be expected to submit via BBLearn two files: 1. An RMarkdown file with the naming convention: LASTNAME_COURSECODE_Section#.Rmd, and 2. A knitted .PDF with the same naming convention: LASTNAME_COURSECODE_Section#.pdf

To generate these files you have two options:

1. Click on the pencil and pad logo in the top of this text, copy the exercise section code, and drop it into your own .Rmd.
2. Git clone our course Github Repository, navigate to the 'Exercises' folder, and use that .Rmd as a template.

Note: Exercises submited in any other format, or those missing questions will not be graded

To generate your .PDF to upload, in your RMarkdown file simply push the 'Knit' button at the top of your document.

Chapter 2

Introduction to NEON & its Data

Estimated Time: 1 hour

Here we will broadly overview of the National Ecological Observatory Network (NEON). Please carefully read through these materials and links that discuss NEON's mission and design.

Course participants: As you review this information, please consider the final course project that you will work on at the over this semester. At the end of week two, you will document an initial research question or idea and associated data needed to address that question, that you may want to explore while pursuing this course.

2.1 Learning Objectives

At the end of this activity, you will be able to:

- Explain the mission of the National Ecological Observatory Network (NEON).
- Explain the how sites are located within the NEON project design.
- Explain the different types of data that will be collected and provided by NEON.

2.2 The NEON Project Mission & Design

To capture ecological heterogeneity across the United States, NEON's design divides the continent into 20 statistically different eco-climatic domains. Each NEON field site is located within an eco-climatic domain.

2.2.0.1 The Science and Design of NEON

To gain a better understanding of the broad scope fo NEON watch this 4 minute long video.

Please, read the following page about NEON's mission.

2.3 NEON's Spatial Design

2.3.0.1 The Spatial Design of NEON

Watch this 4:22 minute video exploring the spatial design of NEON field sites.

Please read the following page about NEON's Spatial Design:

- Read this primer on NEON's Sampling Design
- Read about the different types of field sites - core and relocatable

2.3.1 NEON Field Site Locations

Explore the NEON Field Site map taking note of the locations of

1. Aquatic & terrestrial field sites.
2. Core & relocatable field sites.

Click here to view the NEON Field Site Map

Explore the NEON field site map. Do the following:

- Zoom in on a study area of interest to see if there are any NEON field sites that are nearby.
- Click the "More" button in the **upper right hand** corner of the map to filter sites by name, site host, domain or state.
- Select one field site of interest.
  - Click on the marker in the map.
  - Then click on the name of the field site to jump to the field site landing page.

Data Tip: You can download maps, kmz, or shapefiles of the field sites here.

2.4 NEON Data

2.4.0.1 How NEON Collects Data

Watch this 3:06 minute video exploring the data that NEON collects.

Read the Data Collection Methods page to learn more about the different types of data that NEON collects and provides. Then, follow the links below to learn more about each collection method:

- Aquatic Observation System (AOS)
- Aquatic Instrument System (AIS)
- Terrestrial Instrument System (TIS) – Flux Tower
- Terrestrial Instrument System (TIS) – Soil Sensors and Measurements
- Terrestrial Organismal System (TOS)
- Airborne Observation Platform (AOP)

All data collection protocols and processing documents are publicly available. Read more about the standardized protocols and how to access these documents.

2.4.1 Specimens & Samples

NEON also collects samples and specimens from which the other data products are based. These samples are also available for research and education purposes. Learn more: NEON Biorepository.

2.4.2 Airborne Remote Sensing

Watch this 5 minute video to better understand the NEON Airborne Observation Platform (AOP).

Data Tip: NEON also provides support to your own research including proposals to fly the AOP over other study sites, a mobile tower/instrumentation setup and others. Learn more here the Assignable Assets programs .

2.4.3 Accessing NEON Data

NEON data are processed and go through quality assurance quality control checks at NEON headquarters in Boulder, CO. NEON carefully documents every aspect of sampling design, data collection, processing and delivery. This documentation is freely available through the NEON data portal.

- Visit the NEON Data Portal - data.neonscience.org
- Read more about the quality assurance and quality control processes for NEON data and how the data are processed from raw data to higher level data products.
- Explore NEON Data Products. On the page for each data product in the catalog you can find the basic information about the product, find the

data collection and processing protocols, and link directly to downloading the data.
- Additionally, some types of NEON data are also available through the data portals of other organizations. For example, NEON Terrestrial Insect DNA Barcoding Data is available through the Barcode of Life Datasystem (BOLD). Or NEON phenocam images are available from the Phenocam network site. More details on where else the data are available from can be found in the Availability and Download section on the Product Details page for each data product (visit Explore Data Products to access individual Product Details pages).

2.4.4 Pathways to access NEON Data

There are several ways to access data from NEON:

1. Via the NEON data portal. Explore and download data. Note that much of the tabular data is available in zipped .csv files for each month and site of interest. To combine these files, use the neonUtilities package (R tutorial, Python tutorial).

2. Use R or Python to programmatically access the data. NEON and community members have created code packages to directly access the data through an API. Learn more about the available resources by reading the Code Resources page or visiting the NEONScience GitHub repo.

3. Using the NEON API. Access NEON data directly using a custom API call.
4. Access NEON data through partner's portals. Where NEON data directly overlap with other community resources, NEON data can be accessed through the portals. Examples include Phenocam, BOLD, Ameriflux, and others. You can learn more in the documentation for individual data products.

2.5 Accessing NEON Data

2.5.1 Via a NEON API Token

NEON data can be downloaded from either the NEON Data Portal or the NEON API. When downloading from the Data Portal, you can create a user account. Read about the benefits of an account on the User Account page. You can also use your account to create a token for using the API. Your token is unique to your account, so don't share it.

While using a token is optional in general, it is required for this course. Using a token when downloading data via the API, including when using the neonU-

tilities package, links your downloads to your user account, as well as enabling faster download speeds. For more information about token usage and benefits, see the NEON API documentation page.

For now, in addition to faster downloads, using a token helps NEON to track data downloads. Using **anonymized** user information, they can then calculate data access statistics, such as which data products are downloaded most frequently, which data products are downloaded in groups by the same users, and how many users in total are downloading data. This information helps NEON to evaluate the growth and reach of the observatory, and to advocate for training activities, workshops, and software development.

Tokens can be used whenever you use the NEON API. In this tutorial, we'll focus on using tokens with the neonUtilities R package.

2.6 Objectives

After completing this activity, you will be able to:

- Create a NEON API token
- Use your token when downloading data with neonUtilities

2.7 Things You'll Need To Complete This Tutorial

You will need a version of R (3.4.1 or higher) and, preferably, `RStudio` loaded on your computer to complete this tutorial.

2.7.1 Install R Packages

- **neonUtilities:** `install.packages("neonUtilities")`

2.8 Additional Resources

- NEON Data Portal
- NEONScience GitHub Organization
- neonUtilities tutorial

If you've never downloaded NEON data using the neonUtilities package before, we recommend starting with the Download and Explore tutorial before proceeding with this tutorial.

In the next sections, we'll get an API token from the NEON Data Portal, and then use it in neonUtilities when downloading data.

2.9 Get a NEON API Token

The first step is create a NEON user account, if you don't have one. Follow the instructions on the Data Portal User Accounts page. If you do already have an account, go to the NEON Data Portal, sign in, and go to your My Account profile page.

Once you have an account, you can create an API token for yourself. At the bottom of the My Account page, you should see this bar:

```
<a href="{{ site.baseurl }}/images/NEON-api-token/get-api-token-button.png">
<img src="{{ site.baseurl }}/images/NEON-api-token/get-api-token-button.png" alt="Accou
```

Click the 'GET API TOKEN' button. After a moment, you should see this:

```
<a href="{{ site.baseurl }}/images/NEON-api-token/account-page-token-view.png">
<img src="{{ site.baseurl }}/images/NEON-api-token/account-page-token-view.png" alt="A
```

Click on the Copy button to copy your API token to the clipboard:

```
<a href="{{ site.baseurl }}/images/NEON-api-token/api-token-copy-button.png">
<img src="{{ site.baseurl }}/images/NEON-api-token/api-token-copy-button.png" alt="Acc
```

2.10 Use API token in neonUtilities

In the next section, we'll walk through saving your token somewhere secure but accessible to your code. But first let's try out using the token the easy way.

First, we need to load the `neonUtilities` package and set the working directory:

```
# install neonUtilities - can skip if already installed, but
# API tokens are only enabled in neonUtilities v1.3.4 and higher
# if your version number is lower, re-install
install.packages("neonUtilities")
# load neonUtilities
library(neonUtilities)
# set working directory
wd <- "~/data" # this will depend on your local machine
setwd(wd)
```

NEON API tokens are very long, so it would be annoying to keep pasting the entire text string into functions. Assign your token an object name:

```
NEON_TOKEN <- "PASTE YOUR TOKEN HERE"
```

Now we'll use the `loadByProduct()` function to download data. Your API token is entered as the optional `token` input parameter. For this example, we'll download Plant foliar traits (DP1.10026.001).

```
foliar <- loadByProduct(dpID="DP1.10026.001", site="all",
                        package="expanded", check.size=F,
                        token=NEON_TOKEN)
```

You should now have data saved in the `foliar` object; the API silently used your token. If you've downloaded data without a token before, you may notice this is faster!

This format applies to all `neonUtilities` functions that involve downloading data or otherwise accessing the API; you can use the `token` input with all of them. For example, when downloading remote sensing data:

```
chm <- byTileAOP(dpID="DP3.30015.001", site="WREF",
                 year=2017, check.size=F,
                 easting=c(571000,578000),
                 northing=c(5079000,5080000),
                 savepath=wd,
                 token=NEON_TOKEN)
```

2.11 Token management for open code

Your API token is unique to your account, so don't share it!

If you're writing code that will be shared with colleagues or available publicly, such as in a GitHub repository or supplemental materials of a published paper, you can't include the line of code above where we assigned your token to `NEON_TOKEN`, since your token is fully visible in the code there. Instead, you'll need to save your token locally on your computer, and pull it into your code without displaying it. There are a few ways to do this, we'll show two options here.

- Option 1: Save the token in a local file, and `source()` that file at the start of every script. This is fairly simple but requires a line of code in every script.
- Option 2: Add the token to a `.Renviron` file to create an environment variable that gets loaded when you open R. This is a little harder to set up initially, but once it's done, it's done globally, and it will work in every script you run.

2.12 Option 1: Save token in a local file

Open a new, empty R script (.R). Put a single line of code in the script:

```
NEON_TOKEN <- "PASTE YOUR TOKEN HERE"
```

Save this file in a logical place on your machine, somewhere that won't be

visible publicly. Here, let's call the file `neon_token_source.R`, and save it to the working directory. Then, at the start of every script where you're going to use the NEON API, you would run this line of code:

```
source(paste0(wd, "/neon_token_source.R"))
```

Then you'll be able to use `token=NEON_TOKEN` when you run `neonUtilities` functions, and you can share your code without accidentally sharing your token.

2.13 Option 2: Save token to the R environment

To create a persistent environment variable, we use a `.Renviron` file. Before creating a file, check which directory R is using as your home directory:

```
# For Windows:
Sys.getenv("R_USER")
# For Mac/Linux:
Sys.getenv("HOME")
```

Check the home directory to see if you already have a `.Renviron` file, **using the file browse pane in RStudio**, or using another file browse method with hidden files shown. Files that begin with `.` are hidden by default, but RStudio recognizes files that begin with `.R` and displays them.

```
<a href="{{ site.baseurl }}/images/NEON-api-token/R-environ-file-browse.png">
<img src="{{ site.baseurl }}/images/NEON-api-token/R-environ-file-browse.png" alt="File
<figcaption>Screenshot of file browse pane with .Renviron file.
</figcaption>
```

If you already have a `.Renviron` file, open it and follow the instructions below to add to it. If you don't have one, create one using File -> New File -> Text File in the RStudio menus.

Add one line to the text file. In this option, there are no quotes around the token value.

```
NEON_TOKEN=PASTE YOUR TOKEN HERE
```

Save the file as `.Renviron`, in the RStudio home directory identified above. Double check the spelling, this will not work if you have a typo. Re-start R to load the environment.

Once your token is assigned to an environment variable, use the function `Sys.getenv()` to access it. For example, in `loadByProduct()`:

```
foliar <- loadByProduct(dpID="DP1.10026.001", site="all",
                        package="expanded", check.size=F,
                        token=Sys.getenv("NEON_TOKEN"))
```

2.13.0.1 Part 2: Pulling NEON Data via the API

This is a tutorial in pulling data from the NEON API or Application Programming Interface. The tutorial uses R and the R package httr, but the core information about the API is applicable to other languages and approaches.

2.14 NEON data

As a reminder, there are 3 basic categories of NEON data:

1. Observational - Data collected by a human in the field, or in an analytical laboratory, e.g. beetle identification, foliar isotopes
2. Instrumentation - Data collected by an automated, streaming sensor, e.g. net radiation, soil carbon dioxide
3. Remote sensing - Data collected by the airborne observation platform, e.g. LIDAR, surface reflectance

This lab covers all three types of data, it is required to complete these sections in order and not skip ahead, since the query principles are explained in the first section, on observational data.

2.15 Objectives

After completing this activity, you will be able to:

- Pull observational, instrumentation, and geolocation data from the NEON API.
- Transform API-accessed data from JSON to tabular format for analyses.

2.16 Things You'll Need To Complete This Tutorial

To complete this tutorial you will need the most current version of R and, preferably, RStudio loaded on your computer.

2.16.1 Install R Packages

- **httr:** `install.packages("httr")`
- **jsonlite:** `install.packages("jsonlite")`
- **dplyr:** `install.packages("dplyr")`
- **devtools:** `install.packages("devtools")`
- **downloader:** `install.packages("downloader")`
- **geoNEON:** `devtools::install_github("NEONScience/NEON-geolocation/geoNEON")`
- **neonUtilities:** `devtools::install_github("NEONScience/NEON-utilities/neonUtilities")`

Note, you must have devtools installed & loaded, prior to loading geoNEON or neonUtilities.

2.16.2 Additional Resources

- Webpage for the NEON API
- GitHub repository for the NEON API
- ROpenSci wrapper for the NEON API (not covered in this tutorial)

2.17 What is an API?

If you are unfamiliar with the concept of an API, think of an API as a 'middle person' that provides a communication path for a software application to obtain information from a digital data source. APIs are becoming a very common means of sharing digital information. Many of the apps that you use on your computer or mobile device to produce maps, charts, reports, and other useful forms of information pull data from multiple sources using APIs. In the ecological and environmental sciences, many researchers use APIs to programmatically pull data into their analyses. (Quoted from the NEON Observatory Blog story: API and data availability viewer now live on the NEON data portal.)

2.18 Anatomy of an API call

An example API call: http://data.neonscience.org/api/v0/data/DP1.10003.001/WOOD/2015-07

This includes the base URL, endpoint, and target.

2.18.1 Base URL:

http://data.neonscience.org/api/v0/data/DP1.10003.001/WOOD/2015-07

Specifics are appended to this in order to get the data or metadata you're looking for, but all calls to this API will include the base URL. For the NEON API, this is http://data.neonscience.org/api/v0 – not clickable, because the base URL by itself will take you nowhere!

2.18.2 Endpoints:

http://data.neonscience.org/api/v0/data/DP1.10003.001/WOOD/2015-07

What type of data or metadata are you looking for?

- ~/**products** Information about one or all of NEON's data products
- ~/**sites** Information about data availability at the site specified in the call
- ~/**locations** Spatial data for the NEON locations specified in the call

- ~/**data** Data! By product, site, and date (in monthly chunks).

2.18.3 Targets:

http://data.neonscience.org/api/v0/data/DP1.10003.001/WOOD/2015-07

The specific data product, site, or location you want to get data for.

2.19 Observational data (OS)

Which product do you want to get data for? Consult the Explore Data Products page.

We'll pick Breeding landbird point counts, DP1.10003.001

First query the products endpoint of the API to find out which sites and dates have data available. In the products endpoint, the target is the numbered identifier for the data product:

```
# Load the necessary libraries
library(httr)
library(jsonlite)
library(dplyr, quietly=T)
library(downloader)

# Request data using the GET function & the API call
req <- GET("http://data.neonscience.org/api/v0/products/DP1.10003.001")
req
```

```
## Response [https://data.neonscience.org/api/v0/products/DP1.10003.001]
##   Date: 2020-07-31 21:03
##   Status: 200
##   Content-Type: application/json;charset=UTF-8
##   Size: 24.2 kB
```

The object returned from `GET()` has many layers of information. Entering the name of the object gives you some basic information about what you downloaded.

The `content()` function returns the contents in the form of a highly nested list. This is typical of JSON-formatted data returned by APIs. We can use the `names()` function to view the different types of information within this list.

```
# View requested data
req.content <- content(req, as="parsed")
names(req.content$data)
```

```
##  [1] "productCodeLong"              "productCode"
##  [3] "productCodePresentation"      "productName"
```

```
##  [5] "productDescription"           "productStatus"
##  [7] "productCategory"              "productHasExpanded"
##  [9] "productScienceTeamAbbr"       "productScienceTeam"
## [11] "productPublicationFormatType" "productAbstract"
## [13] "productDesignDescription"     "productStudyDescription"
## [15] "productBasicDescription"      "productExpandedDescription"
## [17] "productSensor"                "productRemarks"
## [19] "themes"                       "changeLogs"
## [21] "specs"                        "keywords"
## [23] "siteCodes"
```

You can see all of the infoamtion by running the line `print(req.content)`, but this will result in a very long printout in your console. Instead, you can view list items individually. Here, we highlight a couple of interesting examples:

```
# View Abstract
req.content$data$productAbstract
```

```
## [1] "This data product contains the quality-controlled, native sampling resolution d
```

```
# View Available months and associated URLs for Onaqui, Utah - ONAQ
req.content$data$siteCodes[[27]]
```

```
## $siteCode
## [1] "ONAQ"
##
## $availableMonths
## $availableMonths[[1]]
## [1] "2017-05"
##
## $availableMonths[[2]]
## [1] "2018-05"
##
## $availableMonths[[3]]
## [1] "2018-06"
##
## $availableMonths[[4]]
## [1] "2019-05"
##
##
## $availableDataUrls
## $availableDataUrls[[1]]
## [1] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ONAQ/2017-05"
##
## $availableDataUrls[[2]]
## [1] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ONAQ/2018-05"
##
## $availableDataUrls[[3]]
```

```
## [1] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ONAQ/2018-06"
##
## $availableDataUrls[[4]]
## [1] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ONAQ/2019-05"
```

To get a more accessible view of which sites have data for which months, you'll need to extract data from the nested list. There are a variety of ways to do this, in this tutorial we'll explore a couple of them. Here we'll use `fromJSON()`, in the jsonlite package, which doesn't fully flatten the nested list, but gets us the part we need. To use it, we need a text version of the content. The text version is not as human readable but is readable by the `fromJSON()` function.

```
# make this JSON readable -> "text"
req.text <- content(req, as="text")

# Flatten data frame to see available data.
avail <- jsonlite::fromJSON(req.text, simplifyDataFrame=T, flatten=T)
avail
```

```
## $data
## $data$productCodeLong
## [1] "NEON.DOM.SITE.DP1.10003.001"
##
## $data$productCode
## [1] "DP1.10003.001"
##
## $data$productCodePresentation
## [1] "NEON.DP1.10003"
##
## $data$productName
## [1] "Breeding landbird point counts"
##
## $data$productDescription
## [1] "Count, distance from observer, and taxonomic identification of breeding landbirds observe
##
## $data$productStatus
## [1] "ACTIVE"
##
## $data$productCategory
## [1] "Level 1 Data Product"
##
## $data$productHasExpanded
## [1] TRUE
##
## $data$productScienceTeamAbbr
## [1] "TOS"
##
```

```
## $data$productScienceTeam
## [1] "Terrestrial Observation System (TOS)"
##
## $data$productPublicationFormatType
## [1] "TOS Data Product Type"
##
## $data$productAbstract
## [1] "This data product contains the quality-controlled, native sampling resolution d
##
## $data$productDesignDescription
## [1] "Depending on the size of the site, sampling for this product occurs either at e
##
## $data$productStudyDescription
## [1] "This sampling occurs at all NEON terrestrial sites."
##
## $data$productBasicDescription
## [1] "The basic package contains the per point metadata table that includes data pert
##
## $data$productExpandedDescription
## [1] "The expanded package includes two additional tables and two additional fields w
##
## $data$productSensor
## NULL
##
## $data$productRemarks
## [1] "Queries for this data product will return data collected during the date range
##
## $data$themes
## [1] "Organisms, Populations, and Communities"
##
## $data$changeLogs
## NULL
##
## $data$specs
##   specId             specNumber
## 1   3656      NEON.DOC.000916vC
## 2   2565 NEON_bird_userGuide_vA
## 3   3729      NEON.DOC.014041vJ
##
## $data$keywords
##  [1] "vertebrates"           "birds"
##  [3] "diversity"             "taxonomy"
##  [5] "community composition" "distance sampling"
##  [7] "avian"                 "species composition"
##  [9] "population"            "Aves"
## [11] "Chordata"              "point counts"
```

```
## [13] "landbirds"             "invasive"
## [15] "introduced"            "native"
## [17] "animals"               "Animalia"
##
## $data$siteCodes
##    siteCode
## 1      ABBY
## 2      BARR
## 3      BART
## 4      BLAN
## 5      BONA
## 6      CLBJ
## 7      CPER
## 8      DCFS
## 9      DEJU
## 10     DELA
## 11     DSNY
## 12     GRSM
## 13     GUAN
## 14     HARV
## 15     HEAL
## 16     JERC
## 17     JORN
## 18     KONA
## 19     KONZ
## 20     LAJA
## 21     LENO
## 22     MLBS
## 23     MOAB
## 24     NIWO
## 25     NOGP
## 26     OAES
## 27     ONAQ
## 28     ORNL
## 29     OSBS
## 30     PUUM
## 31     RMNP
## 32     SCBI
## 33     SERC
## 34     SJER
## 35     SOAP
## 36     SRER
## 37     STEI
## 38     STER
## 39     TALL
## 40     TEAK
```

```
## 41     TOOL
## 42     TREE
## 43     UKFS
## 44     UNDE
## 45     WOOD
## 46     WREF
## 47     YELL
##                                                                                availableMonths
## 1                                                2017-05, 2017-06, 2018-06, 2018-07, 2019-05
## 2                                                                  2017-07, 2018-07, 2019-06
## 3                                                2015-06, 2016-06, 2017-06, 2018-06, 2019-06
## 4                                     2017-05, 2017-06, 2018-05, 2018-06, 2019-05, 2019-06
## 5                                                2017-06, 2018-06, 2018-07, 2019-06
## 6                                                2017-05, 2018-04, 2019-04, 2019-05
## 7                           2013-06, 2015-05, 2016-05, 2017-05, 2017-06, 2018-05, 2019-06
## 8                                                2017-06, 2017-07, 2018-07, 2019-06, 2019-07
## 9                                                                  2017-06, 2018-06, 2019-06
## 10                                                       2015-06, 2017-06, 2018-05, 2019-06
## 11                                               2015-06, 2016-05, 2017-05, 2018-05, 2019-05
## 12                                               2016-06, 2017-05, 2017-06, 2018-05, 2019-05
## 13                                               2015-05, 2017-05, 2018-05, 2019-05, 2019-06
## 14                                    2015-05, 2015-06, 2016-06, 2017-06, 2018-06, 2019-06
## 15                                               2017-06, 2018-06, 2018-07, 2019-06, 2019-07
## 16                                                       2016-06, 2017-05, 2018-06, 2019-06
## 17                                               2017-04, 2017-05, 2018-04, 2018-05, 2019-04
## 18                                                                 2018-05, 2018-06, 2019-06
## 19                                                       2017-06, 2018-05, 2018-06, 2019-06
## 20                                                       2017-05, 2018-05, 2019-05, 2019-06
## 21                                                                 2017-06, 2018-05, 2019-06
## 22                                                                          2018-06, 2019-05
## 23                                                       2015-06, 2017-05, 2018-05, 2019-05
## 24                                                       2015-07, 2017-07, 2018-07, 2019-07
## 25                                                                 2017-07, 2018-07, 2019-07
## 26                                               2017-05, 2017-06, 2018-04, 2018-05, 2019-05
## 27                                                       2017-05, 2018-05, 2018-06, 2019-05
## 28                                               2016-05, 2016-06, 2017-05, 2018-06, 2019-05
## 29                                                       2016-05, 2017-05, 2018-05, 2019-05
## 30                                                                                 2018-04
## 31                                    2017-06, 2017-07, 2018-06, 2018-07, 2019-06, 2019-07
## 32 2015-06, 2016-05, 2016-06, 2017-05, 2017-06, 2018-05, 2018-06, 2019-05, 2019-06
## 33                                                       2017-05, 2017-06, 2018-05, 2019-05
## 34                                                                 2017-04, 2018-04, 2019-04
## 35                                                                 2017-05, 2018-05, 2019-05
## 36                                                       2017-05, 2018-04, 2018-05, 2019-04
## 37                          2016-05, 2016-06, 2017-06, 2018-05, 2018-06, 2019-05, 2019-06
## 38                          2013-06, 2015-05, 2016-05, 2017-05, 2018-05, 2019-05, 2019-06
```

```
## 39                                    2015-06, 2016-07, 2017-06, 2018-06, 2019-05
## 40                                             2017-06, 2018-06, 2019-06, 2019-07
## 41                                                      2017-06, 2018-07, 2019-06
## 42                                             2016-06, 2017-06, 2018-06, 2019-06
## 43                                                      2017-06, 2018-06, 2019-06
## 44                                    2016-06, 2016-07, 2017-06, 2018-06, 2019-06
## 45                                    2015-07, 2017-07, 2018-07, 2019-06, 2019-07
## 46                                                      2018-06, 2019-05, 2019-06
## 47                                                               2018-06, 2019-06
##
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
## 25
## 26
## 27
## 28
## 29
## 30
## 31
## 32 https://data.neonscience.org/api/v0/data/DP1.10003.001/SCBI/2015-06, https://data.neonscien
## 33
## 34
## 35
## 36
```

```
## 37
## 38
## 39
## 40
## 41
## 42
## 43
## 44
## 45
## 46
## 47
```

The object contains a lot of information about the data product, including:

- keywords under `$data$keywords`,
- references for documentation under `$data$specs`,
- data availability by site and month under `$data$siteCodes`, and
- specific URLs for the API calls for each site and month under `$data$siteCodes$availableDataUrls`.

We need `$data$siteCodes` to tell us what we can download. `$data$siteCodes$availableDataUrls` allows us to avoid writing the API calls ourselves in the next steps.

```
# get data availability list for the product
bird.urls <- unlist(avail$data$siteCodes$availableDataUrls)
length(bird.urls) #total number of URLs
```

```
## [1] 204
```

```
bird.urls[1:10] #show first 10 URLs available
```

```
##  [1] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ABBY/2017-05"
##  [2] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ABBY/2017-06"
##  [3] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ABBY/2018-06"
##  [4] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ABBY/2018-07"
##  [5] "https://data.neonscience.org/api/v0/data/DP1.10003.001/ABBY/2019-05"
##  [6] "https://data.neonscience.org/api/v0/data/DP1.10003.001/BARR/2017-07"
##  [7] "https://data.neonscience.org/api/v0/data/DP1.10003.001/BARR/2018-07"
##  [8] "https://data.neonscience.org/api/v0/data/DP1.10003.001/BARR/2019-06"
##  [9] "https://data.neonscience.org/api/v0/data/DP1.10003.001/BART/2015-06"
## [10] "https://data.neonscience.org/api/v0/data/DP1.10003.001/BART/2016-06"
```

These are the URLs showing us what files are available for each month where there are data.

Let's look at the bird data from Woodworth (WOOD) site from July 2015. We can do this by using the above code but now specifying which site/date we want using the `grep()` function.

Note that if there were only one month of data from a site, you could leave off

the date in the function. If you want date from more than one site/month you need to iterate this code, GET fails if you give it more than one URL.

```
# get data availability for WOOD July 2015
brd <- GET(bird.urls[grep("WOOD/2015-07", bird.urls)])
brd.files <- jsonlite::fromJSON(content(brd, as="text"))

# view just the available data files
brd.files$data$files
```

```
##                               crc32
## 1  e0adb3146b5cce59eea09864145efcb1
## 2  4438e5e050fc7be5949457f42089a397
## 3  d84b496cf950b5b96e762473beda563a
## 4  6d15da01c03793da8fc6d871e6659ea8
## 5  f37931d46213246dccf2a161211c9afe
## 6  e67f1ae72760a63c616ec18108453aaa
## 7  df102cb4cfdce092cda3c0942c9d9b67
## 8  e67f1ae72760a63c616ec18108453aaa
## 9  2ad379ae44f4e87996bdc3dee70a0794
## 10 d76cfc5443ac27a058fab1d319d31d34
## 11 22e3353dabb8b154768dc2eee9873718
## 12 6d15da01c03793da8fc6d871e6659ea8
## 13 a2c47410a6a0f49d0b1cf95be6238604
## 14 f37931d46213246dccf2a161211c9afe
## 15 6ba91b6e109ff14d1911dcaad9febeb9
## 16 680a2f53c0a9d1b0ab4f8814bda5b399
##                                                                                name
## 1     NEON.D09.WOOD.DP1.10003.001.brd_countdata.2015-07.basic.20191107T152331Z.csv
## 2                   NEON.D09.WOOD.DP1.10003.001.2015-07.basic.20191107T152331Z.zip
## 3                         NEON.D09.WOOD.DP1.10003.001.readme.20191107T152331Z.txt
## 4                      NEON.D09.WOOD.DP0.10003.001.validation.20191107T152331Z.csv
## 5      NEON.D09.WOOD.DP1.10003.001.brd_perpoint.2015-07.basic.20191107T152331Z.csv
## 6                       NEON.D09.WOOD.DP1.10003.001.variables.20191107T152331Z.csv
## 7           NEON.D09.WOOD.DP1.10003.001.EML.20150701-20150705.20191107T152331Z.xml
## 8                       NEON.D09.WOOD.DP1.10003.001.variables.20191107T152331Z.csv
## 9  NEON.D09.WOOD.DP1.10003.001.brd_countdata.2015-07.expanded.20191107T152331Z.csv
## 10        NEON.D09.WOOD.DP1.10003.001.brd_references.expanded.20191107T152331Z.csv
## 11               NEON.D09.WOOD.DP1.10003.001.2015-07.expanded.20191107T152331Z.zip
## 12                     NEON.D09.WOOD.DP0.10003.001.validation.20191107T152331Z.csv
## 13                          NEON.Bird_Conservancy_of_the_Rockies.brd_personnel.csv
## 14  NEON.D09.WOOD.DP1.10003.001.brd_perpoint.2015-07.expanded.20191107T152331Z.csv
## 15          NEON.D09.WOOD.DP1.10003.001.EML.20150701-20150705.20191107T152331Z.xml
## 16                        NEON.D09.WOOD.DP1.10003.001.readme.20191107T152331Z.txt
##      size
## 1  346679
```

```
## 2   67816
## 3   12784
## 4   10084
## 5   23521
## 6    7337
## 7   70539
## 8    7337
## 9  367402
## 10   1012
## 11  79998
## 12  10084
## 13  46349
## 14  23521
## 15  78750
## 16  13063
##
## 1        https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003.00
## 2                      https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SIT
## 3                             https://neon-prod-pub-1.s3.data.neonscience.org/NEON.
## 4                           https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.
## 5         https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003.0
## 6                           https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM
## 7              https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10
## 8                         https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SI
## 9  https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003.001/PROV
## 10        https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003.0
## 11               https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.
## 12                     https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SIT
## 13                          https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DO
## 14 https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003.001/PRO
## 15          https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM.SITE.DP1.10003
## 16                          https://neon-prod-pub-1.s3.data.neonscience.org/NEON.DOM
```

In this output, `name` and `url` are key fields. It provides us with the names of the files available for this site and month, and URLs where we can get the files. We'll use the file names to pick which ones we want.

The available files include both **data** and **metadata**, and both the **basic** and **expanded** data packages. Typically the expanded package includes additional quality or uncertainty data, either in additional files or additional fields than in the basic files. Basic and expanded data packages are available for most NEON data products (some only have basic). Metadata are described by file name below.

The format for most of the file names is:

NEON.[domain number].[site code].[data product ID].[file-specific name]. [date of file creation]

Some files omit the domain and site, since they're not specific to a location, like the data product readme. The date of file creation uses the ISO6801 format, in this case 20170720T182547Z, and can be used to determine whether data have been updated since the last time you downloaded.

Available files in our query for July 2015 at Woodworth are all of the following (leaving off the initial NEON.D09.WOOD.10003.001):

- **~.2015-07.expanded.20170720T182547Z.zip:** zip of all files in the expanded package
- **~.brd_countdata.2015-07.expanded.20170720T182547Z.csv:** count data table, expanded package version: counts of birds at each point
- **~.brd_perpoint.2015-07.expanded.20170720T182547Z.csv:** point data table, expanded package version: metadata at each observation point
- **NEON.Bird Conservancy of the Rockies.brd_personnel.csv:** personnel data table, accuracy scores for bird observers
- **~.2015-07.basic.20170720T182547Z.zip:** zip of all files in the basic package
- **~.brd_countdata.2015-07.basic.20170720T182547Z.csv:** count data table, basic package version: counts of birds at each point
- **~.brd_perpoint.2015-07.basic.20170720T182547Z.csv:** point data table, basic package version: metadata at each observation point
- **NEON.DP1.10003.001_readme.txt:** readme for the data product (not specific to dates or location). Appears twice in the list, since it's in both the basic and expanded package
- **~.20150101-20160613.xml:** Ecological Metadata Language (EML) file. Appears twice in the list, since it's in both the basic and expanded package
- **~.validation.20170720T182547Z.csv:** validation file for the data product, lists input data and data entry rules. Appears twice in the list, since it's in both the basic and expanded package
- **~.variables.20170720T182547Z.csv:** variables file for the data product, lists data fields in downloaded tables. Appears twice in the list, since it's in both the basic and expanded package

We'll get the data tables for the point data and count data in the basic package. The list of files doesn't return in the same order every time, so we won't use position in the list to select. Plus, we want code we can re-use when getting data from other sites and other months. So we select files based on the data table name and the package name.

```
# Get both files
brd.count <- read.delim(brd.files$data$files$url
```

```
                          [intersect(grep("countdata",
                                          brd.files$data$files$name),
                                     grep("basic",
                                          brd.files$data$files$name))],
                          sep=",")

brd.point <- read.delim(brd.files$data$files$url
                          [intersect(grep("perpoint",
                                          brd.files$data$files$name),
                                     grep("basic",
                                          brd.files$data$files$name))],
                          sep=",")
```

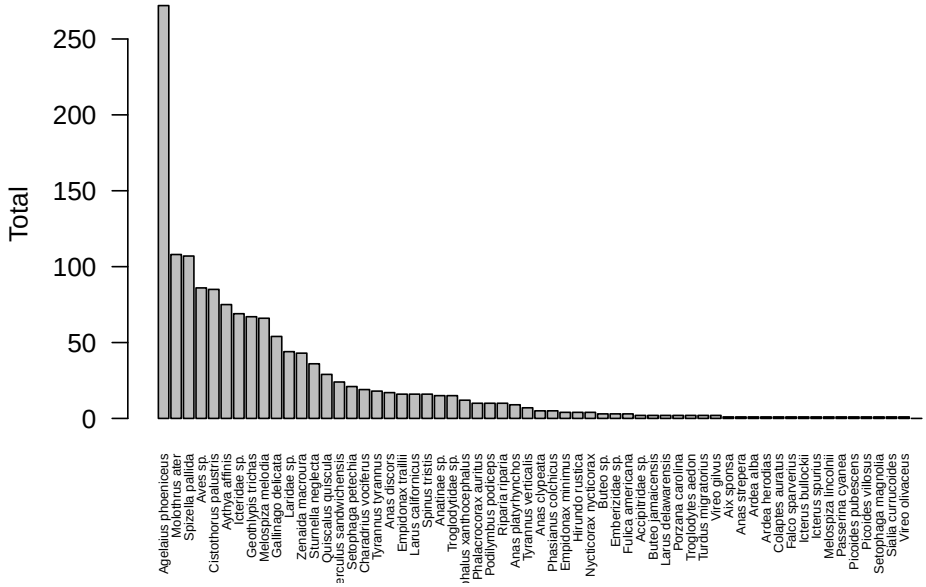
Now we have the data and can access it in R. Just to show that the files we pulled have actual data in them, let's make a quick graphic:

```
# Cluster by species
clusterBySp <- brd.count %>%
  dplyr::group_by(scientificName) %>%
  dplyr::summarise(total=sum(clusterSize, na.rm=T))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# Reorder so list is ordered most to least abundance
clusterBySp <- clusterBySp[order(clusterBySp$total, decreasing=T),]

# Plot
barplot(clusterBySp$total, names.arg=clusterBySp$scientificName,
        ylab="Total", cex.names=0.5, las=2)
```



Wow! There are lots of *Agelaius phoeniceus* (Red-winged Blackbirds) at WOOD in July.

2.20 Instrumentation data (IS)

The process is essentially the same for sensor data. We'll do the same series of queries for Soil Temperature, DP1.00041.001. Let's use data from Moab in March 2017 this time.

```
# Request soil temperature data availability info
req.soil <- GET("http://data.neonscience.org/api/v0/products/DP1.00041.001")

# make this JSON readable
# Note how we've change this from two commands into one here
avail.soil <- jsonlite::fromJSON(content(req.soil, as="text"), simplifyDataFrame=T, flatten=T)

# get data availability list for the product
temp.urls <- unlist(avail.soil$data$siteCodes$availableDataUrls)

# get data availability from location/date of interest
tmp <- GET(temp.urls[grep("MOAB/2017-06", temp.urls)])
tmp.files <- jsonlite::fromJSON(content(tmp, as="text"))
length(tmp.files$data$files$name) # There are a lot of available files
```

```
## [1] 190
```

```
tmp.files$data$files$name[1:10]   # Let's print the first 10
```

```
##  [1] "NEON.D13.MOAB.DP1.00041.001.004.501.030.ST_30_minute.2017-06.expanded.2020062
##  [2] "NEON.D13.MOAB.DP1.00041.001.002.506.030.ST_30_minute.2017-06.expanded.2020062
##  [3] "NEON.D13.MOAB.DP1.00041.001.004.505.001.ST_1_minute.2017-06.expanded.20200620
##  [4] "NEON.D13.MOAB.DP1.00041.001.001.508.001.ST_1_minute.2017-06.expanded.20200620
##  [5] "NEON.D13.MOAB.DP1.00041.001.003.505.030.ST_30_minute.2017-06.expanded.2020062
##  [6] "NEON.D13.MOAB.DP1.00041.001.003.501.001.ST_1_minute.2017-06.expanded.20200620
##  [7] "NEON.D13.MOAB.DP1.00041.001.002.501.030.ST_30_minute.2017-06.expanded.2020062
##  [8] "NEON.D13.MOAB.DP1.00041.001.004.502.001.ST_1_minute.2017-06.expanded.20200620
##  [9] "NEON.D13.MOAB.DP1.00041.001.004.509.001.ST_1_minute.2017-06.expanded.20200620
## [10] "NEON.D13.MOAB.DP1.00041.001.sensor_positions.20200620T070859Z.csv"
```

These file names start and end the same way as the observational files, but the middle is a little more cryptic. The structure from beginning to end is:

NEON.[domain number].[site code].[data product ID].00000. [soil plot number].[depth].[averaging interval].[data table name]. [year]-[month].[data package].[date of file creation]

So **"NEON.D13.MOAB.DP1.00041.001.003.507.030.ST__30__minute.2017-06.expanded.20200620T070859Z.csv"** is the:

- NEON (`NEON.`)
- Domain 13 (`.D13.`)
- Moab field site (`.MOAB.`)
- soil temperature data (`.DP1.00041.001.`)
- collected in Soil Plot 2, (`.002.`)
- at the 7th depth below the surface (`.507.`)
- and reported as a 30-minute mean of (`.030.` and `.ST_30_minute.`)
- only for the period of June 2017 (`.2017-06.`)
- and provided in a expanded data package (`.basic.`)
- published on June 20th, 2020 (`.0200620T070859Z.`).

More information about interpreting file names can be found in the readme that accompanies each download.

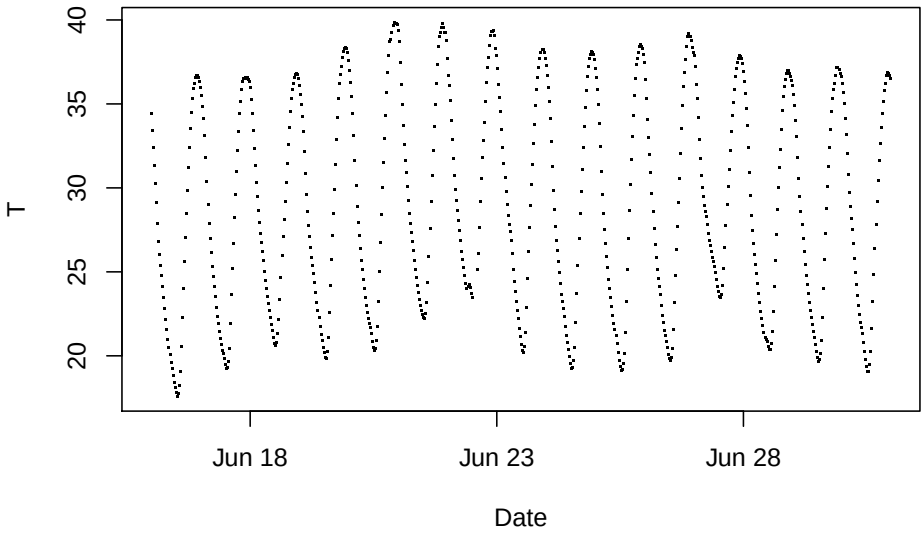
Let's get data (and the URL) for only the 2nd depth described above by selecting `002.502.030` and the word `basic` in the file name.

Go get it:

```
soil.temp <- read.delim(tmp.files$data$files$url
                        [intersect(grep("002.502.030",
                                        tmp.files$data$files$name),
                                   grep("basic",
                                        tmp.files$data$files$name))],
                        sep=",")
```

Now we have the data and can use it to conduct our analyses. To take a quick look at it, let's plot the mean soil temperature by date.

```
# plot temp ~ date
plot(soil.temp$soilTempMean~as.POSIXct(soil.temp$startDateTime,
                                       format="%Y-%m-%d T %H:%M:%S Z"),
     pch=".", xlab="Date", ylab="T")
```



As we'd expect we see daily fluctuation in soil temperature.

2.21 Remote sensing data (AOP)

Again, the process of determining which sites and time periods have data, and finding the URLs for those data, is the same as for the other data types. We'll go looking for High resolution orthorectified camera imagery, DP1.30010, and we'll look at the flight over San Joaquin Experimental Range (SJER) in March 2017.

```
# Request camera data availability info
req.aop <- GET("http://data.neonscience.org/api/v0/products/DP1.30010.001")

# make this JSON readable
# Note how we've changed this from two commands into one here
avail.aop <- jsonlite::fromJSON(content(req.aop, as="text"),
                      simplifyDataFrame=T, flatten=T)

# get data availability list for the product
cam.urls <- unlist(avail.aop$data$siteCodes$availableDataUrls)
```

```
# get data availability from location/date of interest
cam <- GET(cam.urls[intersect(grep("SJER", cam.urls),
                              grep("2017", cam.urls))])
cam.files <- jsonlite::fromJSON(content(cam, as="text"))

# this list of files is very long, so we'll just look at the first ten
head(cam.files$data$files$name, 10)
```

```
##  [1] "17032816_EH021656(20170328190629)-0680_ort.tif"
##  [2] "17032816_EH021656(20170328190117)-0642_ort.tif"
##  [3] "17032816_EH021656(20170328195321)-1085_ort.tif"
##  [4] "17032816_EH021656(20170328182204)-0324_ort.tif"
##  [5] "17032816_EH021656(20170328193045)-0880_ort.tif"
##  [6] "17032816_EH021656(20170328182447)-0358_ort.tif"
##  [7] "17032816_EH021656(20170328185526)-0596_ort.tif"
##  [8] "17032816_EH021656(20170328180003)-0154_ort.tif"
##  [9] "17032816_EH021656(20170328192935)-0864_ort.tif"
## [10] "17032816_EH021656(20170328191703)-0760_ort.tif"
```

File names for AOP data are more variable than for IS or OS data; different AOP data products use different naming conventions. File formats differ by product as well.

This particular product, camera imagery, is stored in TIFF files. For a full list of AOP data products, their naming conventions, and their file formats, see .

Instead of reading a TIFF into R, we'll download it to the working directory. This is one option for getting AOP files from the API; if you plan to work with the files in R, you'll need to know how to read the relevant file types into R. We hope to add tutorials for this in the near future.

To download the TIFF file, we use the `downloader` package, and we'll select a file based on the time stamp in the file name: `20170328192931`

```
download(cam.files$data$files$url[grep("20170328192931",
                                       cam.files$data$files$name)],
         paste(getwd(), "/SJER_image.tif", sep=""), mode="wb")
```

The image, below, of the San Joaquin Experimental Range should now be in your working directory.

```
<a href="https://raw.githubusercontent.com/NEONScience/NEON-Data-Skills/dev-aten/graphi
<img src="https://raw.githubusercontent.com/NEONScience/NEON-Data-Skills/dev-aten/grapl
<figcaption> An example of camera data (DP1.30010.001) from the San Joaquin
Experimental Range. Source: National Ecological Observatory Network (NEON)
</figcaption>
```

2.22 Geolocation data

You may have noticed some of the spatial data referenced above are a bit vague, e.g. "soil plot 2, 4th depth below the surface."

How to get spatial data and what to do with it depends on which type of data you're working with.

2.22.0.1 Instrumentation data (both aquatic and terrestrial)

Stay tuned - spatial data for instruments are in the process of entry into the NEON database.

2.22.0.2 Observational data - Aquatic

Latitude, longitude, elevation, and associated uncertainties are included in data downloads. Most products also include an "additional coordinate uncertainty" that should be added to the provided uncertainty. Additional spatial data, such as northing and easting, can be downloaded from the API.

2.22.0.3 Observational data - Terrestrial

Latitude, longitude, elevation, and associated uncertainties are included in data downloads. These are the coordinates and uncertainty of the sampling plot; for many protocols it is possible to calculate a more precise location. Instructions for doing this are in the respective data product user guides, and code is in the `geoNEON` package on GitHub.

2.22.1 Querying a single named location

Let's look at the named locations in the bird data we downloaded above. To do this, look for the field called `namedLocation`, which is present in all observational data products, both aquatic and terrestrial.

```
# view named location
head(brd.point$namedLocation)
```

```
## [1] WOOD_013.birdGrid.brd WOOD_013.birdGrid.brd WOOD_013.birdGrid.brd
## [4] WOOD_013.birdGrid.brd WOOD_013.birdGrid.brd WOOD_013.birdGrid.brd
## 7 Levels: WOOD_006.birdGrid.brd ... WOOD_020.birdGrid.brd
```

Here we see the first six entries in the `namedLocation` column which tells us the names of the Terrestrial Observation plots where the bird surveys were conducted.

We can query the locations endpoint of the API for the first named location, `WOOD_013.birdGrid.brd`.

```
# location data
req.loc <- GET("http://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd")

# make this JSON readable
brd.WOOD_013 <- jsonlite::fromJSON(content(req.loc, as="text"))
brd.WOOD_013
```

```
## $data
## $data$locationName
## [1] "WOOD_013.birdGrid.brd"
##
## $data$locationDescription
## [1] "Plot \"WOOD_013\" at site \"WOOD\""
##
## $data$locationType
## [1] "OS Plot - brd"
##
## $data$domainCode
## [1] "D09"
##
## $data$siteCode
## [1] "WOOD"
##
## $data$locationDecimalLatitude
## [1] 47.13912
##
## $data$locationDecimalLongitude
## [1] -99.23243
##
## $data$locationElevation
## [1] 579.31
##
## $data$locationUtmEasting
## [1] 482375.7
##
## $data$locationUtmNorthing
## [1] 5220650
##
## $data$locationUtmHemisphere
## [1] "N"
##
## $data$locationUtmZone
## [1] 14
##
## $data$alphaOrientation
```

```
## [1] 0
##
## $data$betaOrientation
## [1] 0
##
## $data$gammaOrientation
## [1] 0
##
## $data$xOffset
## [1] 0
##
## $data$yOffset
## [1] 0
##
## $data$zOffset
## [1] 0
##
## $data$offsetLocation
## NULL
##
## $data$locationProperties
##                              locationPropertyName locationPropertyValue
## 1                    Value for Coordinate source            GeoXH 6000
## 2               Value for Coordinate uncertainty                  0.28
## 3                                Value for Country          unitedStates
## 4                                 Value for County              Stutsman
## 5                 Value for Elevation uncertainty                  0.48
## 6                    Value for Filtered positions                   121
## 7                        Value for Geodetic datum                 WGS84
## 8     Value for Horizontal dilution of precision                     1
## 9                     Value for Maximum elevation                579.31
## 10                    Value for Minimum elevation                569.79
## 11 Value for National Land Cover Database (2001)   grasslandHerbaceous
## 12                      Value for Plot dimensions          500m x 500m
## 13                              Value for Plot ID             WOOD_013
## 14                           Value for Plot size                250000
## 15                         Value for Plot subtype             birdGrid
## 16                            Value for Plot type           distributed
## 17    Value for Positional dilution of precision                   2.4
## 18            Value for Reference Point Position                    B2
## 19                           Value for Slope aspect                238.91
## 20                         Value for Slope gradient                 2.83
## 21                       Value for Soil type order             Mollisols
## 22                        Value for State province                    ND
## 23               Value for Subtype Specification             ninePoints
## 24                                Value for UTM Zone                  14N
```

```
##
## $data$locationParent
## [1] "WOOD"
##
## $data$locationParentUrl
## [1] "https://data.neonscience.org/api/v0/locations/WOOD"
##
## $data$locationChildren
## [1] "WOOD_013.birdGrid.brd.B2" "WOOD_013.birdGrid.brd.A2"
## [3] "WOOD_013.birdGrid.brd.C3" "WOOD_013.birdGrid.brd.A3"
## [5] "WOOD_013.birdGrid.brd.B3" "WOOD_013.birdGrid.brd.C1"
## [7] "WOOD_013.birdGrid.brd.A1" "WOOD_013.birdGrid.brd.B1"
## [9] "WOOD_013.birdGrid.brd.C2"
##
## $data$locationChildrenUrls
## [1] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.B2"
## [2] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.A2"
## [3] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.C3"
## [4] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.A3"
## [5] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.B3"
## [6] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.C1"
## [7] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.A1"
## [8] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.B1"
## [9] "https://data.neonscience.org/api/v0/locations/WOOD_013.birdGrid.brd.C2"
```

Note spatial information under `$data$[nameOfCoordinate]` and under `$data$locationProperties`. Also note `$data$locationChildren`: these are the finer scale locations that can be used to calculate precise spatial data for bird observations.

For convenience, we'll use the `geoNEON` package to make the calculations. First we'll use `getLocByName()` to get the additional spatial information available through the API, and look at the spatial resolution available in the initial download:

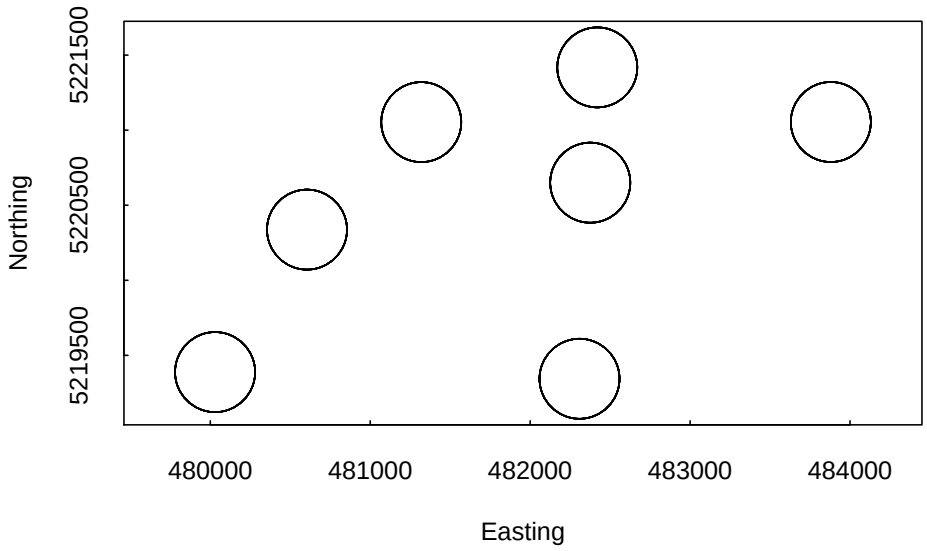
```
# load the geoNEON package
library(geoNEON)

# extract the spatial data
brd.point.loc <- getLocByName(brd.point)
```

```
##
  |
  |                                                                      |   0%
  |
  |=========                                                             |  14%
  |
```

```
  |==================                                                    |  29%
  |
  |============================                                          |  43%
  |
  |======================================                                |  57%
  |
  |================================================                      |  71%
  |
  |==========================================================            |  86%
  |
  |======================================================================| 100%
```

```
# plot bird point locations
# note that decimal degrees is also an option in the data
symbols(brd.point.loc$easting, brd.point.loc$northing,
        circles=brd.point.loc$coordinateUncertainty,
        xlab="Easting", ylab="Northing", tck=0.01, inches=F)
```



And use `getLocTOS()` to calculate the point locations of observations.

```
brd.point.pt <- getLocTOS(brd.point, "brd_perpoint")
```

```
##
  |
  |                                                                      |   0%
  |
  |=                                                                     |   2%
  |
  |==                                                                    |   3%
  |
```

```
|===                                                    |   5%
|
|====                                                   |   6%
|
|=====                                                  |   8%
|
|======                                                 |  10%
|
|=======                                                |  11%
|
|========                                               |  13%
|
|=========                                              |  14%
|
|==========                                             |  16%
|
|===========                                            |  17%
|
|============                                           |  19%
|
|=============                                          |  21%
|
|==============                                         |  22%
|
|===============                                        |  24%
|
|=================                                      |  25%
|
|==================                                     |  27%
|
|===================                                    |  29%
|
|====================                                   |  30%
|
|=====================                                  |  32%
|
|======================                                 |  33%
|
|========================                               |  35%
|
|=========================                              |  37%
|
|==========================                             |  38%
|
|===========================                            |  40%
|
```

```
|=========================                              |  41%
|
|==========================                             |  43%
|
|===========================                            |  44%
|
|============================                           |  46%
|
|=============================                          |  48%
|
|==============================                         |  49%
|
|================================                       |  51%
|
|=================================                      |  52%
|
|==================================                     |  54%
|
|===================================                    |  56%
|
|====================================                   |  57%
|
|=====================================                  |  59%
|
|======================================                 |  60%
|
|=======================================                |  62%
|
|========================================               |  63%
|
|=========================================              |  65%
|
|==========================================             |  67%
|
|===========================================            |  68%
|
|============================================           |  70%
|
|=============================================          |  71%
|
|==============================================         |  73%
|
|===============================================        |  75%
|
|================================================       |  76%
|
```

```
  |================================================                 |  78%
  |
  |=================================================                |  79%
  |
  |==================================================               |  81%
  |
  |===================================================              |  83%
  |
  |====================================================             |  84%
  |
  |=====================================================            |  86%
  |
  |======================================================           |  87%
  |
  |=======================================================          |  89%
  |
  |========================================================         |  90%
  |
  |=========================================================        |  92%
  |
  |==========================================================       |  94%
  |
  |===========================================================      |  95%
  |
  |============================================================     |  97%
  |
  |============================================================== |  98%
  |
  |================================================================| 100%
```

```
# plot bird point locations
# note that decimal degrees is also an option in the data
# symbols(brd.point.pt$easting, brd.point.pt$northing,
#         circles=brd.point.pt$adjCoordinateUncertainty,
#         xlab="Easting", ylab="Northing", tck=0.01, inches=F)
```

Now you can see the individual points where the respective point counts were located.

2.23 Taxonomy

NEON maintains accepted taxonomies for many of the taxonomic identification data we collect. NEON taxonomies are available for query via the API; they are also provided via an interactive user interface, the Taxon Viewer.

NEON taxonomy data provides the reference information for how NEON vali-

dates taxa; an identification must appear in the taxonomy lists in order to be accepted into the NEON database. Additions to the lists are reviewed regularly. The taxonomy lists also provide the author of the scientific name, and the reference text used.

The taxonomy endpoint of the API works a little bit differently from the other endpoints. In the "Anatomy of an API Call" section above, each endpoint has a single type of target - a data product number, a named location name, etc. For taxonomic data, there are multiple query options, and some of them can be used in combination. For example, a query for taxa in the Pinaceae family:

http://data.neonscience.org/api/v0/taxonomy/?family=Pinaceae

The available types of queries are listed in the taxonomy section of the API web page. Briefly, they are:

- `taxonTypeCode`: Which of the taxonomies maintained by NEON are you looking for? BIRD, FISH, PLANT, etc. Cannot be used in combination with the taxonomic rank queries.
- each of the major taxonomic ranks from genus through kingdom
- `scientificname`: Genus + specific epithet (+ authority). Search is by exact match only, see final example below.
- `verbose`: Do you want the short (`false`) or long (`true`) response
- `offset`: Skip this number of items in the list. Defaults to 50.
- `limit`: Result set will be truncated at this length. Defaults to 50.

Staff on the NEON project have plans to modify the settings for `offset` and `limit`, such that `offset` will default to 0 and `limit` will default to ∞, but in the meantime users will want to set these manually. They are set to non-default values in the examples below.

For the first example, let's query for the loon family, Gaviidae, in the bird taxonomy. Note that query parameters are case-sensitive.

```
loon.req <- GET("http://data.neonscience.org/api/v0/taxonomy/?family=Gaviidae&offset=0&limit=500"
```

Parse the results into a list using `fromJSON()`:

```
loon.list <- jsonlite::fromJSON(content(loon.req, as="text"))
```

And look at the `$data` element of the results, which contains:

- The full taxonomy of each taxon
- The short taxon code used by NEON (taxonID/acceptedTaxonID)
- The author of the scientific name (scientificNameAuthorship)
- The vernacular name, if applicable
- The reference text used (nameAccordingToID)

The terms used for each field are matched to Darwin Core (dwc) and the Global Biodiversity Information Facility (gbif) terms, where possible, and the matches are indicated in the column headers.

```
loon.list$data
```

```
##   taxonTypeCode taxonID acceptedTaxonID dwc:scientificName
## 1          BIRD    ARLO            ARLO      Gavia arctica
## 2          BIRD    COLO            COLO        Gavia immer
## 3          BIRD    PALO            PALO     Gavia pacifica
## 4          BIRD    RTLO            RTLO     Gavia stellata
## 5          BIRD    YBLO            YBLO      Gavia adamsii
##   dwc:scientificNameAuthorship dwc:taxonRank dwc:vernacularName
## 1                   (Linnaeus)       species        Arctic Loon
## 2                   (Brunnich)       species        Common Loon
## 3                   (Lawrence)       species       Pacific Loon
## 4                (Pontoppidan)       species  Red-throated Loon
## 5                 (G. R. Gray)       species Yellow-billed Loon
##      dwc:nameAccordingToID dwc:kingdom dwc:phylum dwc:class   dwc:order
## 1 doi: 10.1642/AUK-15-73.1    Animalia   Chordata      Aves Gaviiformes
## 2 doi: 10.1642/AUK-15-73.1    Animalia   Chordata      Aves Gaviiformes
## 3 doi: 10.1642/AUK-15-73.1    Animalia   Chordata      Aves Gaviiformes
## 4 doi: 10.1642/AUK-15-73.1    Animalia   Chordata      Aves Gaviiformes
## 5 doi: 10.1642/AUK-15-73.1    Animalia   Chordata      Aves Gaviiformes
##   dwc:family dwc:genus gbif:subspecies gbif:variety
## 1   Gaviidae     Gavia              NA           NA
## 2   Gaviidae     Gavia              NA           NA
## 3   Gaviidae     Gavia              NA           NA
## 4   Gaviidae     Gavia              NA           NA
## 5   Gaviidae     Gavia              NA           NA
```

To get the entire list for a particular taxonomic type, use the `taxonTypeCode` query. Be cautious with this query, the PLANT taxonomic list has several hundred thousand entries.

For an example, let's look up the small mammal taxonomic list, which is one of the shorter ones, and use the `verbose=true` option to see a more extensive list of taxon data, including many taxon ranks that aren't populated for these taxa. For space here, we display only the first 10 taxa:

```
mam.req <- GET("http://data.neonscience.org/api/v0/taxonomy/?taxonTypeCode=SMALL_MAMMAL
mam.list <- jsonlite::fromJSON(content(mam.req, as="text"))
mam.list$data[1:10,]
```

```
##    taxonTypeCode taxonID acceptedTaxonID                dwc:scientificName
## 1   SMALL_MAMMAL    AMHA            AMHA         Ammospermophilus harrisii
## 2   SMALL_MAMMAL    AMIN            AMIN        Ammospermophilus interpres
## 3   SMALL_MAMMAL    AMLE            AMLE         Ammospermophilus leucurus
## 4   SMALL_MAMMAL    AMLT            AMLT Ammospermophilus leucurus tersus
## 5   SMALL_MAMMAL    AMNE            AMNE          Ammospermophilus nelsoni
## 6   SMALL_MAMMAL    AMSP            AMSP              Ammospermophilus sp.
```

```
## 7   SMALL_MAMMAL    APRN            APRN            Aplodontia rufa nigra
## 8   SMALL_MAMMAL    APRU            APRU                  Aplodontia rufa
## 9   SMALL_MAMMAL    ARAL            ARAL                Arborimus albipes
## 10  SMALL_MAMMAL    ARLO            ARLO            Arborimus longicaudus
##    dwc:scientificNameAuthorship dwc:taxonRank
## 1            Audubon and Bachman       species
## 2                        Merriam       species
## 3                        Merriam       species
## 4                        Goldman    subspecies
## 5                        Merriam       species
## 6                           <NA>         genus
## 7                         Taylor    subspecies
## 8                      Rafinesque       species
## 9                        Merriam       species
## 10                          True       species
##               dwc:vernacularName taxonProtocolCategory
## 1      Harriss Antelope Squirrel         opportunistic
## 2        Texas Antelope Squirrel         opportunistic
## 3  Whitetailed Antelope Squirrel         opportunistic
## 4                           <NA>         opportunistic
## 5      Nelsons Antelope Squirrel         opportunistic
## 6                           <NA>         opportunistic
## 7                           <NA>            non-target
## 8                       Sewellel            non-target
## 9               Whitefooted Vole                target
## 10                 Red Tree Vole                target
##    dwc:nameAccordingToID
## 1   isbn: 978 0801882210
## 2   isbn: 978 0801882210
## 3   isbn: 978 0801882210
## 4   isbn: 978 0801882210
## 5   isbn: 978 0801882210
## 6   isbn: 978 0801882210
## 7   isbn: 978 0801882210
## 8   isbn: 978 0801882210
## 9   isbn: 978 0801882210
## 10  isbn: 978 0801882210
##
## 1  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 2  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 3  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 4  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 5  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 6  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 7  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
## 8  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and Geographi
```

```
## 9  Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and
## 10 Wilson D. E. and D. M. Reeder. 2005. Mammal Species of the World; A Taxonomic and
##    dwc:kingdom gbif:subkingdom gbif:infrakingdom gbif:superdivision
## 1     Animalia              NA                NA                 NA
## 2     Animalia              NA                NA                 NA
## 3     Animalia              NA                NA                 NA
## 4     Animalia              NA                NA                 NA
## 5     Animalia              NA                NA                 NA
## 6     Animalia              NA                NA                 NA
## 7     Animalia              NA                NA                 NA
## 8     Animalia              NA                NA                 NA
## 9     Animalia              NA                NA                 NA
## 10    Animalia              NA                NA                 NA
##    gbif:division gbif:subdivision gbif:infradivision gbif:parvdivision
## 1             NA                NA                 NA                 NA
## 2             NA                NA                 NA                 NA
## 3             NA                NA                 NA                 NA
## 4             NA                NA                 NA                 NA
## 5             NA                NA                 NA                 NA
## 6             NA                NA                 NA                 NA
## 7             NA                NA                 NA                 NA
## 8             NA                NA                 NA                 NA
## 9             NA                NA                 NA                 NA
## 10            NA                NA                 NA                 NA
##    gbif:superphylum dwc:phylum gbif:subphylum gbif:infraphylum
## 1                NA   Chordata             NA               NA
## 2                NA   Chordata             NA               NA
## 3                NA   Chordata             NA               NA
## 4                NA   Chordata             NA               NA
## 5                NA   Chordata             NA               NA
## 6                NA   Chordata             NA               NA
## 7                NA   Chordata             NA               NA
## 8                NA   Chordata             NA               NA
## 9                NA   Chordata             NA               NA
## 10               NA   Chordata             NA               NA
##    gbif:superclass dwc:class gbif:subclass gbif:infraclass gbif:superorder
## 1               NA  Mammalia            NA              NA              NA
## 2               NA  Mammalia            NA              NA              NA
## 3               NA  Mammalia            NA              NA              NA
## 4               NA  Mammalia            NA              NA              NA
## 5               NA  Mammalia            NA              NA              NA
## 6               NA  Mammalia            NA              NA              NA
## 7               NA  Mammalia            NA              NA              NA
## 8               NA  Mammalia            NA              NA              NA
## 9               NA  Mammalia            NA              NA              NA
## 10              NA  Mammalia            NA              NA              NA
```

```
##    dwc:order gbif:suborder gbif:infraorder gbif:section gbif:subsection
## 1   Rodentia            NA              NA           NA              NA
## 2   Rodentia            NA              NA           NA              NA
## 3   Rodentia            NA              NA           NA              NA
## 4   Rodentia            NA              NA           NA              NA
## 5   Rodentia            NA              NA           NA              NA
## 6   Rodentia            NA              NA           NA              NA
## 7   Rodentia            NA              NA           NA              NA
## 8   Rodentia            NA              NA           NA              NA
## 9   Rodentia            NA              NA           NA              NA
## 10  Rodentia            NA              NA           NA              NA
##    gbif:superfamily    dwc:family gbif:subfamily gbif:tribe gbif:subtribe
## 1                NA     Sciuridae        Xerinae  Marmotini            NA
## 2                NA     Sciuridae        Xerinae  Marmotini            NA
## 3                NA     Sciuridae        Xerinae  Marmotini            NA
## 4                NA     Sciuridae        Xerinae  Marmotini            NA
## 5                NA     Sciuridae        Xerinae  Marmotini            NA
## 6                NA     Sciuridae        Xerinae  Marmotini            NA
## 7                NA Aplodontiidae           <NA>       <NA>            NA
## 8                NA Aplodontiidae           <NA>       <NA>            NA
## 9                NA    Cricetidae    Arvicolinae       <NA>            NA
## 10               NA    Cricetidae    Arvicolinae       <NA>            NA
##           dwc:genus dwc:subgenus gbif:subspecies gbif:variety
## 1  Ammospermophilus         <NA>              NA           NA
## 2  Ammospermophilus         <NA>              NA           NA
## 3  Ammospermophilus         <NA>              NA           NA
## 4  Ammospermophilus         <NA>              NA           NA
## 5  Ammospermophilus         <NA>              NA           NA
## 6  Ammospermophilus         <NA>              NA           NA
## 7        Aplodontia         <NA>              NA           NA
## 8        Aplodontia         <NA>              NA           NA
## 9         Arborimus         <NA>              NA           NA
## 10        Arborimus         <NA>              NA           NA
##    gbif:subvariety gbif:form gbif:subform speciesGroup dwc:specificEpithet
## 1               NA        NA           NA         <NA>            harrisii
## 2               NA        NA           NA         <NA>           interpres
## 3               NA        NA           NA         <NA>            leucurus
## 4               NA        NA           NA         <NA>            leucurus
## 5               NA        NA           NA         <NA>             nelsoni
## 6               NA        NA           NA         <NA>                 sp.
## 7               NA        NA           NA         <NA>               rufa
## 8               NA        NA           NA         <NA>               rufa
## 9               NA        NA           NA         <NA>             albipes
## 10              NA        NA           NA         <NA>         longicaudus
##    dwc:infraspecificEpithet
## 1                      <NA>
```

```
## 2                      <NA>
## 3                      <NA>
## 4                    tersus
## 5                      <NA>
## 6                      <NA>
## 7                     nigra
## 8                      <NA>
## 9                      <NA>
## 10                     <NA>
```

To get information about a single taxon, use the `scientificname` query. This query will not do a fuzzy match, so you need to query the exact name of the taxon in the NEON taxonomy. Because of this, the query will be most useful when you already have NEON data in hand and are looking for more information about a specific taxon. Querying on `scientificname` is unlikely to be an efficient way to figure out if NEON recognizes a particular taxon.

In addition, scientific names contain spaces, which are not allowed in a URL. The spaces need to be replaced with the URL encoding replacement, %20.

For an example, let's look up the little sand verbena, *Abronia minor Standl.* Searching for *Abronia minor* will fail, because the NEON taxonomy for this species includes the authority. The search will also fail with spaces. Search for `Abronia%20minor%20Standl.`, and in this case we can omit `offset` and `limit` because we know there can only be a single result:

```
am.req <- GET("http://data.neonscience.org/api/v0/taxonomy/?scientificname=Abronia%20m
am.list <- jsonlite::fromJSON(content(am.req, as="text"))
am.list$data
```

```
##   taxonTypeCode taxonID acceptedTaxonID    dwc:scientificName
## 1         PLANT   ABMI2           ABMI2 Abronia minor Standl.
##   dwc:scientificNameAuthorship dwc:taxonRank  dwc:vernacularName
## 1                      Standl.       species little sand verbena
##                         dwc:nameAccordingToID dwc:kingdom    dwc:phylum
## 1 http://plants.usda.gov (accessed 8/25/2014)     Plantae Magnoliophyta
##       dwc:class      dwc:order    dwc:family dwc:genus gbif:subspecies
## 1 Magnoliopsida Caryophyllales Nyctaginaceae   Abronia              NA
##   gbif:variety
## 1           NA
```

2.24 Stacking NEON data

At the top of this tutorial, we installed the `neonUtilities` package. This is a custom R package that stacks the monthly files provided by the NEON data portal into a single continuous file for each type of data table in the download. It currently handles files downloaded from the data portal, but not files pulled

from the API. That functionality will be added soon!

For a guide to using `neonUtilities` on data downloaded from the portal, look here.

2.25 Exercises

2.25.1 Computational

2.25.1.1 Part 1: Sign up for and Use an NEON API Token:

Code to come

2.25.2 Written

Question 1: How might or does the NEON project intersect with your current research or future career goals? *(1 paragraph)*

Question 2: Use the map in week 2:Intro to NEON to answer the following questions. Consider the research question that you may explore as your final semester project or a current project that you are working on and answer each of the following questions:

- Are there NEON field sites that are in study regions of interest to you?
- What domains are the sites located in?
- What NEON field sites do your current research or Capstone Project ideas coincide with?
- Is the site or sites core or relocatable?
- Is or are they terrestrial or aquatic?
- Are there data available for the NEON field site(s) that you are most interested in? What kind of data are available?

Question 3: Consider either your current or future research, or a question you'd like to address durring this course:

- Which types of NEON data may be more useful to address these questions?
- What non-NEON data resources could be combined with NEON data to help address your question?
- What challenges, if any, could you foresee when beginning to work with these data?

Question 4: Use the Data Portal tools to investigate the data availability for the field sites you've already identified in the previous questions:

- What types of aquatic or terrestrial data are currently available? Remote sensing data?
- Of these, what type of data are you most interested in working with for your project during this course?
- For what time period does the data cover?
- What format is the downloadable file available in?
- Where is the metadata to support this data?

Intro to NEON Culmination Activity

Write up a 1-page summary of a project that you might want to explore using NEON data over the duration of this course. Include the types of NEON (and other data) that you will need to implement this project. Save this summary as you will be refining and adding to your ideas over the course of the semester.

Chapter 3

Introduction to USA-NPN & its Data

Estimated Time: 2 hours

Course participants: As you review this information, please consider the final course project that you will work on at the over this semester. At the end of this section, you will document an initial research question or idea and associated data needed to address that question, that you may want to explore while pursuing this course.

3.1 Learning Objectives

At the end of this activity, you will be able to:

3.2 USA-NPN Project Mission & Design:

The USA National Phenology Network (USA-NPN) collects, organizes, and shares phenological data and information to aid decision-making, scientific discovery, and a broader understanding of phenology from a diversity of perspectives. The USA National Phenology Network consists of a National Coordinating Office (NCO), thousands of volunteer observers and many partners, including research scientists, resource managers, educators, and policy-makers. Anyone who participates in Nature's Notebook or collaborates with NCO staff to advance the science of phenology or to inform decisions is part of the USA-NPN.

3.3 Vision & Mission

USA-NPN's vision is to provide data and information on the timing of seasonal events in plants and animals to ensure the well-being of humans, ecosystems, and natural resources. To support this and its mission the USA-NPN collects, organizes, and shares phenological data and information to aid decision-making, scientific discovery, and a broader understanding of phenology from a diversity of perspectives.

3.3.1 Relevant documents & background information:

1. USA-NPN Strategic Plan

2. USA-NPN Information Sheet: Tracking seasonal changes to support science, natural resource management, and society

3. 2019 USA-NPN Annual Report

3.4 USA-NPN's Spatial design:

As a citizen-science based platform, the spatial sampling of USA-NPN data is opportunistic, since observations are contributed voluntarily by citizen scientist participants.

3.5 Types of USA-NPN Data:

- Observational - As described in USA National Phenology Network Observational Data Documentation (Rosemartin et al. 2018)
  - Status/intensity
  - Individual phenometrics