Meier et al. (2022) Ecosphere - Supporting Information File 2: Example linking NEON TOS Coarse Downed Wood volume and Small Mammal abundance data at plot and site scales

Courtney L. Meier, Katherine M. Thibault, and David T. Barnett

November 18, 2022

Contents

Introduction

The NEON Terrestrial Observation System (TOS) data products are frequently spatially integrated at multiple scales, from the site level to the level of relatively small 'sampling cells' that occur within plots (see Supporting Information File 1, Table 3). In addition, the frequency with which TOS data products are generated at each site varies widely, from multiple bouts per year to once every 5 years. This example is a companion piece to our manuscript, *"Spatial and temporal sampling strategy connecting NEON Terrestrial Observation System protocols,"* and the purpose is to illustrate how a NEON data user can draw scientific insight by working with two TOS data products collected at different spatial scales and at different temporal frequencies.

Here, we focus on the 'Coarse downed wood log survey' data product (DP1.10010.001; generated every 5-years) and the 'Small mammal box trapping' data product (DP1.10072.001; generated multiple times per year). Using R software (https://www.r-project.org/) to download data from the NEON API via the `neonUtilities` package (https://cran.r-project.org/web/packages/neonUtilities/), we show how to link data at both the plot and site spatial scales via the `plotID`. We also develop one method for comparing TOS data across products generated with very different temporal frequencies. For this example, we chose to investigate correlations between 'Coarse downed wood log survey' data and 'Small mammal box trapping' data based on the simple hypothesis that coarse downed wood (CDW) provides habitat for small mammals (MAM) in forested systems, and increasing CDW volume may therefore be correlated with increasing MAM abundance, both within and among sites. Briefly, we first identify colocated mammal grids and Distributed base plots that produce CDW data within a time-frame of interest. Next we generate plot-specific CDW volume estimates, and plot-specific MAM abundance estimates from each bout of mammal sampling within a 3-year window centered on the year that CDW was sampled. We then take the plot-specific maximum MAM abundance from all bouts within the 3-year window and compare with plot-level CDW volume data.

For the code chunks throughout this document, we preface each with a brief summary of the objectives. In addition, within each chunk we employ comments prefaced with the '`###`' symbols to document the purpose

of individual lines of code. The `#-->` symbol denotes an explanation of example code output. A single '`#`' denotes code that has been commented out (e.g., lengthy API calls that need only be run once).

Load data from the NEON Data Portal

- When retrieving data from the NEON API using the `neonUtilities` functions (see code chunks below), use of the `token` argument allows authentication with a user-specific API token. Use of a token is not required, but tokens provide faster download speeds and help NEON to measure data use for reporting to the NSF.
- All TOS data products, including both the 'Coarse downed wood log survey' and 'Small mammal box trapping' data products, may be affected by site management practices and/or natural disturbances. NEON publishes site management data via the 'Site management and event reporting' product (DP1.10111.001; https://data.neonscience.org/data-products/DP1.10111.001). We do not explore the 'Site management and event reporting' product further in this example. However, disturbance and management events like hurricanes, wildfires, and controlled burns can clearly affect both CDW volume and MAM abundance.

Load Coarse Downed Wood Tally Data

Note: The word `'log'` is used frequently in the Coarse Downed Wood dataset, and it refers to a downed piece of wood not a logarithm.

At sites with sufficient logs >= 2 cm diameter, the 'Coarse downed wood log survey' data product, hereafter referred to as 'CDW Tally', is generated from both Distributed and Tower plots on a 5 year interval. However, sampling of Distributed and Tower plots is staggered through time such that each plot type is sampled every 2-3 years within a site on an alternating basis. Distributed plots are allocated across each site in proportion to the area of dominant National Land Cover Database (NLCD) Vegetation Classes, and these plots allow creating statistically robust site-level estimates of both CDW volume (derived from tallies) and small mammal density. The Small Mammal (MAM) data product is generated annually from most TOS sites, and MAM data are collected from grids that are typically colocated with Distributed base plots but not Tower plots. Thus, the first tasks for this example are:

- To identify which NEON sites have produced CDW Tally data.
- To identify several forested sites with recent Distributed plot data for the example, and
- To identify which Distributed plotIDs are colocated with MAM sampling.

The NEON 'Coarse downed wood log survey' data product citation is:

- NEON (National Ecological Observatory Network). Coarse downed wood log survey, RELEASE-2022 (DP1.10010.001). https://doi.org/10.48443/54dd-9407. Dataset accessed from https://data.neonscience.org on November 18, 2022

In the code chunk below, we complete the following:

- Use the `neonUtilities::loadByProduct()` function to programmatically retrieve CDW data using the NEON Data Portal API (commented out), save these data locally (commented out), then read the saved data into R. This approach allows potentially lengthy API calls to be made only once.
- Summarize data to identify 5-6 sites with recent CDW Tally data from Distributed plots that can be paired with MAM data. For demonstration purposes, the goal is to choose a small number of forested sites that span a range of habitats.
- Within each selected site, identify plotIDs in the CDW dataset that are colocated with MAM sampling grids. We use the `geoNEON::getLocByName()` function and the common structure of the `namedLocation` variable across TOS products to identify which CDW sampling locations are also MAM sampling locations.

```
### Retrieve CDW Tally data for last 6 years from NEON Data Portal for all sites and dates

# cdwDP <- neonUtilities::loadByProduct(
#   dpID = "DP1.10010.001",
#   site = "all",
#   startdate = "2016-01",
#   enddate = "2021-12",
#   package = "basic",
#   release = "RELEASE-2022",
#   tabl = "all",
#   check.size = FALSE,
#   token = Sys.getenv('NEON_PAT')
# )


### Extract the field tally data from the download list, and save for quicker read-in
# cdw <- cdwDP$cdw_fieldtally
# saveRDS(cdw, file = "cdw_fieldTally_2016-2021.RDS")


### Read in data saved via neonUtilities code chunk above, then filter to
### Distributed plots and identify plotIDs within scheduled sampling events that
### have generated data. Records with samplingImpractical != "OK" have no data.
cdw <- readRDS(file = "cdw_fieldTally_2016-2021.RDS")

summaryDist <- cdw %>%
  dplyr::filter(plotType == "distributed") %>%
  dplyr::group_by(domainID, siteID, eventID, samplingImpractical) %>%
  dplyr::summarise(plotCount = length(unique(plotID)))


### From 'summaryDist' output, select 6 sites with closed-canopy forest; sites
### are selected based on authors' knowledge of the sites.
theSites <- c("HARV","JERC","STEI","UKFS","RMNP","ABBY")


### Filter CDW data to selected sites, filter to most recent sampling eventID,
### then remove unneeded columns.
tempCDW <- cdw %>%
  dplyr::filter(plotType == "distributed",
                siteID %in% theSites) %>%
  dplyr::group_by(domainID, siteID) %>%
  dplyr::filter(yearBoutBegan == max(yearBoutBegan)) %>%
  dplyr::arrange(domainID, siteID, plotID) %>%
  dplyr::ungroup() %>%
  dplyr::select(-uid,
                -coordinateUncertainty,
                -elevationUncertainty,
                -publicationDate)



### Identify plotIDs in CDW dataset that are colocated with MAM sampling grids.
```

```
### First, create MAM 'namedLocation' value for each CDW namedLocation value
### (e.g., 'HARV_001.mammalGrid.mam'); the goal is to check whether each CDW
### namedLocation has a colocated MAM namedLocation. The structure of the namedLocation
### value for any NEON data product can be determined by downloading data and checking
### the namedLocation field.

tempCDW <- tempCDW %>%
  dplyr::mutate(
    mamNamedLoc = stringr::str_replace(
      string = namedLocation,
      pattern = "basePlot.cdw",
      replacement = "mammalGrid.mam"
    ),
    .after = namedLocation
  ) %>%
  as.data.frame()


### Second, retrieve namedLocation data for each mamNamedLoc value that was constructed
### above; if a CDW namedLocation is also a MAM namedLocation, the
### geoNEON::getLocByName() function will return a value from the NEON API; if a CDW
### namedLocation is not also a MAM namedLocation, the function does not return a value.
### Note: The 'data' argument must be a data.frame() and CANNOT be a tibble (the latter
### is commonly returned by dplyr).

# mamPlots <- geoNEON::getLocByName(
#   data = tempCDW,
#   locCol = "mamNamedLoc",
#   locOnly = TRUE,
#   token = Sys.getenv('NEON_PAT')
# )


### Save data locally for quicker subsequent read-in
# saveRDS(mamPlots, file = "cdw_mam_collocation.RDS")


### Read in previously saved MAM location data
mamPlots <- readRDS(file = "cdw_mam_collocation.RDS")


### Third, join tempCDW data with mamPlots and use 'subtype' column to identify
### colocated sampling.
tempCDW <- tempCDW %>%
  dplyr::left_join(mamPlots %>%
                     select(plotID, subtype),
                   by = "plotID") %>%
  dplyr::select(-mamNamedLoc)
```

As an aside, rather than programmatically retrieving TOS Spatial Data via the API with the `geoNEON` functions, as is done in the code chunk above, these spatial data may also be manually downloaded:

- First, navigate to https://data.neonscience.org/documents
- Click on 'Spatial Data' and click on the 'All_NEON_TOS_Plots_VX' link to retrieve a zip that

contains a .csv with plot data.
- If this approached is used, one can then read in the included .csv file.
  - For example: `tosPlots <- read.csv(file = "All_NEON_TOS_Plot_Centroids_V8.csv", header = TRUE)`

Load Small Mammal Data

Small Mammal sampling grids (n=3 to n=8 per site) are colocated with Distributed base plots that support multiple TOS protocols (including Coarse Downed Wood), and grids are sampled one to several times per year (ideally, 4-6 times per year). Similar to Distributed base plots, mammal grids are allocated using a stratified random design informed by NLCD Vegetation Class.

The NEON 'Small mammal box trapping' data product citation is:

- NEON (National Ecological Observatory Network). Small mammal box trapping, RELEASE-2022 (DP1.10072.001). https://doi.org/10.48443/h3dk-3a71. Dataset accessed from https://data.neonscience.org on November 18, 2022

The code chunk below accomplishes the following:

- Use the `neonUtilities::loadByProduct()` function to programmatically retrieve MAM data using the NEON Data Portal API (commented out), save these data locally (commented out), then read the saved data into R.
- Retrieve the MAM master taxon table using the NEON API.

```
### Retrieve small mammal box trapping data for same sites and date range as CDW
# mamDP <- neonUtilities::loadByProduct(
#   dpID = "DP1.10072.001",
#   site = theSites, #--> same sites identified for CDW
#   startdate = "2016-01",
#   enddate = "2021-12",
#   package = "basic",
#   release = "RELEASE-2022",
#   tabl = "all",
#   check.size = FALSE,
#   token = Sys.getenv('NEON_PAT')
# )

### Save to RDS format for quicker read-in
# saveRDS(mamDP, file = "mam_allTables_2016-2021.RDS")


### Read in stored MAM data
mamDP <- readRDS(file = "mam_allTables_2016-2021.RDS")


### Convert list tables to dataframes in the Global environment
list2env(mamDP, envir=.GlobalEnv)


### Read in master SMALL_MAMMAL taxon table via the NEON API
### https://www.neonscience.org/resources/learning-hub/tutorials/neon-api-usage
### using the verbose option to get the taxonProtocolCategory field
mam.req <- httr::GET(
  paste0(
    "https://data.neonscience.org/api/v0/taxonomy/?",
```

```
    "taxonTypeCode=SMALL_MAMMAL&verbose=true&offset=0&limit=1000"
  )
)

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

Data QC Checks

Coarse Downed Wood Tally QC Checks

For CDW tally data, the code chunk below performs the following checks:

- Check for duplicates. Removing duplicates is important because log counts are used to estimate CDW volume. As of this writing, we do not use the standardized `neonOS::removeDups()` function to identify CDW duplicates because the primary key in the data needs an update for the function to identify duplicates as intended.
- Remove any tallied logs with a `logDistance` greater than the protocol-specified transect length, and
- Verify that all tallied logs meet 'limiting distance' tally criteria based on reported `equivalentLogDiameter`, `logDistance`, and `volumeFactor` data (see Affleck 2010 for details).
  - The `equivalentLogDiameter` is used so that round and elliptical logs (typically highly decayed) can be assessed with the same criteria. Logs with `logDistance` greater than the limiting distance are removed from the dataset.
  - Because technicians sometimes use look-up tables to determine limiting distance from a subset of `equivalentLogDiameter` values (e.g., when digital ingest devices fail), and look-up tables require choosing the nearest `equivalentLogDiameter` rather than using the actual measured value, it is expected that some logs will have `logDistance` greater than the calculated limiting distance and will be removed.

```
### Check for individualID duplicates based on the 'individualID'. First, construct
### an individualID for logs < 10 cm diameter that are not tagged; smaller logs are
### not tagged and are given a unique temporary logID beginning with 'L' since they
### are less likely to persist over the 5-year CDW Tally measurement interval. Then,
### construct a primary key from plotID, date, lidsAzimuth, and individualID.

tempCDW <- tempCDW %>%
  dplyr::mutate(
    individualID = case_when(
      is.na(individualID) &
        targetTaxaPresent == "Y" ~ paste("NEON.CDW", domainID, plotID, logID, sep = "."),
      TRUE ~ individualID
    ),
    key = paste(plotID, date, lidsAzimuth, individualID, sep = "_")
  )


### Identify duplicates ('dups') based on the 'key' value
cdwDupsKey <- tempCDW %>%
  dplyr::filter(duplicated(tempCDW$key)) %>%
  dplyr::select(key)

cdwDups <- tempCDW %>%
  dplyr::filter(key %in% cdwDupsKey$key)
```

```
#--> Dups results: Only first one listed appears to be a real duplicate based on
#--> examination of other data fields; ABBY records are mis-indentified by the
#--> function as 'dups' because transect was reflected (i.e., not real duplicates).


### Remove second instance of cdwDups$key==
### "STEI_002_2016-05-26_320_NEON.CDW.D05.00375"
cdwRemoveDup <- cdwDups %>%
  dplyr::distinct(key)

tempCDW <- tempCDW %>%
  dplyr::filter(!key %in% cdwRemoveDup$key[1]) %>%
  dplyr::bind_rows(cdwDups[1, ]) %>%
  dplyr::arrange(domainID, siteID, plotID, lidsAzimuth, individualID)


### Volume Factor check: Ensure recorded value matches protocol and flag if no match.
### First, read in CDW volumeFactor and transectLength look-up table derived from
### protocol (NEON.DOC.001711). Note that RMNP has different F-values and transect lengths
### for Distributed and Tower plots due to distant parcels of land used for each plot type
### and different forest types in each parcel. The RMNP values in this table are specific
### to Distributed plots since we wish to compare data with Small Mammal data.

cdwParam <- read.csv(file = "cdw_siteParamLookup.csv", header = TRUE)


### Join look-up table values with CDW data and flag records with incorrect volume factor
### (i.e., f-value). Flagged records are identified with a '1'.
tempCDW <- tempCDW %>%
  dplyr::left_join(cdwParam %>% dplyr::select(siteID, fValue, transectLength),
                   by = "siteID") %>%
  dplyr::mutate(fValQF = case_when(
    is.na(volumeFactor) ~ 1,
    volumeFactor == fValue ~ 0,
    TRUE ~ 1
  ))


### Check records with volumeFactor problems and evaluate
fCheck <- tempCDW %>%
  dplyr::filter(fValQF == 1) %>%
  dplyr::select(domainID, siteID, volumeFactor, fValue, fValQF)

#--> volumeFactor problems identified for HARV and STEI; volumeFactor is supposed to
#--> be assigned at the site level and should not vary from plot-to-plot. The variation
#--> in the data likely reflects a data entry error. We encourage data users to let NEON
#--> staff know about issues they discover like this one:
#--> https://www.neonscience.org/about/contact-us


### Correct the volumeFactor data for HARV, STEI sites based on fCheck results above
tempCDW <- tempCDW %>%
  dplyr::mutate(volumeFactor = as.numeric(volumeFactor)) %>%
```

```
  dplyr::mutate(volumeFactor = case_when(
    siteID == "HARV" & (volumeFactor != 5 | is.na(volumeFactor)) ~ 5,
    siteID == "STEI" & (volumeFactor != 5 | is.na(volumeFactor)) ~ 5,
    TRUE ~ volumeFactor
  ))


### Log Distance tally check: Flag 'logDistance' values that exceed the per-site
### maximum transect length defined in the protocol.
tempCDW <- tempCDW %>%
  dplyr::mutate(
    logDistQF = case_when(
      is.na(logDistance) ~ 0,
      logDistance > transectLength ~ 1,
      TRUE ~ 0
    ))


### Check whether the logDistance is <= limitingDistance, as expected, based on
### equivalentLogDiameter, and flag with '1' if not. The formula to calculate the
### limiting distance is from Affleck 2010, Appendix A:
### limitingDist = ((pi^2)*(roundDiameter^2))/(8*transectNum*fValue), where
### roundDiameter = log round diameter (cm); equal to equivalentLogDiameter in NEON data
### transectNum = number of transects established from survey point (3 for NEON plots),
### fValue = the F value assigned at the site level; volumeFactor field in NEON data
tempCDW <- tempCDW %>%
  dplyr::mutate(
    limitingDist = round(((pi^2)*(equivalentLogDiameter^2))/(8*3*volumeFactor), digits=2)
  ) %>%
  dplyr::mutate(
    limDistQF = case_when(
      is.na(limitingDist) ~ 0,
      logDistance > limitingDist ~ 1,
      TRUE ~ 0
    ))


### Filter out flagged records corresponding to logs that should not have been tallied
filterCDW <- tempCDW %>%
  dplyr::filter(limDistQF == 0,
                logDistQF == 0)
```

Small Mammal Box Trapping Data QC: Check for duplicates

In the code chunk below, we check for duplicates in the `mam_perplotnight` and `mam_pertrapnight` tables using the `neonOS::removeDups()` function:

- `mam_perplotnight`: In this table, duplicate records are defined as those with the same plot and date combination (as captured in an auto-generated nightuid).
- `mam_pertrapnight`: In this table, duplicate records are defined as those with the same nightuid, trap coordinate, and tagID or individualCode; note that the standard `neonOS::removeDups()` function used to remove duplicates cannot account for multiple captures of untagged individuals in a single trap.

```r
### First, check mam_perplotnight table by nightuid using standard removeDups function
mam_plotNight_nodups <- neonOS::removeDups(data = mam_perplotnight,
                                           variables = variables_10072,
                                           table = "mam_perplotnight")

#--> Running this check generated the message: Primary key fields contain NA values
#--> and/or empty strings. Results may be unreliable; check input data carefully.


### To troubleshoot, first query the table to find the problematic records.
### Next, check the mam_pertrapnight table to see if there are corresponding records
### that might have a nightuid
problems <- which(is.na(mam_perplotnight$nightuid))

temp <- mam_pertrapnight %>%
  dplyr::filter(
    plotID %in% mam_perplotnight$plotID[problems] &
      collectDate %in% mam_perplotnight$collectDate[problems]
  )

#--> Since the remaining fields were populated and there are captures for each,
#--> these records appear to represent a valid night of trapping. So, we will assign
#--> custom temporary nightuids in both tables here. As an aside, we encourage users
#--> to let NEON staff know about issues they discover like this one:
#--> https://www.neonscience.org/about/contact-us


### Create custom nightuids for identified duplicates missing primary key value,
### then re-run the neonOS::removeDups() function
mam_perplotnight_adj <- mam_perplotnight %>%
  dplyr::mutate(nightuid = paste(plotID, '_', collectDate, sep = ''))

mam_pertrapnight_adj <- mam_pertrapnight %>%
  dplyr::mutate(nightuid = paste(plotID, '_', collectDate, sep = ''))

mam_plotNight_nodups <- neonOS::removeDups(data = mam_perplotnight_adj,
                                           variables = variables_10072,
                                           table = "mam_perplotnight")

#--> No duplicate key values found


### We now check the mam_pertrapnight table by nightuid and trapcoordinate using the
### standard neonOS::removeDups() function.
### Note: RELEASE 2022 contains ~142K records for theSites downloaded, so this
### operation can take some time. In rare cases, multiple animals can be captured in
### a single trap on the same night and not all receive unique tagIDs or individualCodes.
### This means that the duplicate function is not effective for these records. Here, we
### remove these records prior to running the standard check, and we then add these
### records back to the data frame afterwards.

mam_trapNight_multipleCaps <- mam_pertrapnight_adj %>%
  dplyr::filter(trapStatus == "4 - more than 1 capture in one trap" &
```

```
                  is.na(tagID) & is.na(individualCode))

mam_trapNight_remainingRecords <- mam_pertrapnight_adj %>%
  dplyr::filter(!(uid %in% mam_trapNight_multipleCaps$uid))

mam_trapNight_nodups <- neonOS::removeDups(data = mam_trapNight_remainingRecords,
                                           variables = variables_10072,
                                           table = "mam_pertrapnight")

#--> Output: Two unresolveable duplicates flagged with duplicateRecordQF == 2, which
#--> indicates records that are identical based on primary keys but contain different
#--> values in other fields; see ?neonOS::removeDups documentation for additional
#--> information on duplicate record flagging.


### Follow-up trouble-shooting to deal with unresolveable duplicates: Check for
### unexpected NAs in primary keys
which(is.na(mam_pertrapnight_adj$nightuid))

#--> No NAs found

which(is.na(mam_pertrapnight_adj$trapCoordinate))

#--> No NAs found. The two which() outputs above mean that trapID and individualCode
#--> are the source of the NAs, which is expected based on the protocol.


### Identify duplicates resulting from > 1 animal captured in a single trap on the
### same night.
dupCheck <- mam_trapNight_nodups %>%
  dplyr::filter(duplicateRecordQF == 2 &
                  trapStatus != "4 - more than 1 capture in one trap")

#--> No records in the set; unresolved duplicates will not impact the analyses


### Add multiple capture records back to dataset
mam_trapNight_nodups <- mam_trapNight_nodups %>%
  dplyr::mutate(collectDate = as.Date(collectDate)) %>%
  dplyr::bind_rows(mam_trapNight_multipleCaps)
```

Analysis of Spatially Integrated Data Products at Plot and Site Scales

In this section, we generate Coarse Downed Wood volume estimates and MAM abundance estimates at both plot and site scales to demonstrate the spatial integration of NEON TOS data products. Because CDW volume is generated every 5-years and does not change rapidly, barring fires or other site management activities, we compare with MAM data collected the same year as the CDW data +/- 1 year. For example, for HARV CDW volume data collected in 2019, we compare to MAM abundance data from 2018-2020. While we have selected a 3-year MAM window centered on the CDW sampling year, other researchers may select a different approach based on the questions being addressed.

Create Coarse Downed Wood Volume Datasets

Using the LIDS method (Affleck 2008, 2010), CDW volume is calculated at the plot scale as:

`CDW volume density = F * n` (m^3/hectare);

where `F` is the site-specific volume factor (an approximate inverse to plot size), and `n` is the number of qualifying logs tallied across all three transects originating from the centroid of the plot. Importantly, at the plot scale, CDW volume density is colocated with MAM abundance data at a maximum of n=6 plots per site (for `theSites` included in this analysis).

At the site scale, mean CDW volume (m^3/hectare) is simply the mean of the plot scale values; the mean site-scale parameter is robust because of the spatially-balanced, stratified-random nature of the TOS Spatial Design. When comparing CDW volume and MAM abundance at the site scale, CDW data from all Distributed plots are used for the calculation.

In the code chunk below, we carry out the following steps:

- Remove records with `samplingImpractical` not equal to "OK" and keep records with `samplingImpractical` equals "NA". The latter is important because the `samplingImpractical` field was introduce in 2020 and older data are not populated with a value.
- Use the `targetTaxaPresent` field to identify only those records associated with a tallied log. For transects within a plot that have no logs, field staff create records with `targetTaxaPresent == N`.
- Calculate CDW volume density for those plotIDs that have logs.
- Because plotIDs that have no logs tallied for any of the the three transects contribute meaningful zeroes to the dataset, we separately identify these plots and manually add a CDW volume density of '0' to the dataset.
- Use plot-level CDW volume density data to calculate mean site-level CDW volume density and standard deviations.

```
### Calculate CDW volume for each plotID in the cleaned dataset. First, focus
### on plots with targetTaxaPresent == "Y" --> i.e., at least one qualifying log was
### tallied within the plot. To do this, remove records for transects with no logs,
### and remove records with samplingImpractical != "OK", then count the number of
### logs per plot and calculate CDW volume.
plotCdwVol <- filterCDW %>%
  dplyr::filter(is.na(samplingImpractical) |
                  samplingImpractical == "OK") %>%
  dplyr::filter(targetTaxaPresent == "Y") %>%
  dplyr::group_by(domainID, siteID, volumeFactor, eventID, plotID) %>%
  dplyr::summarise(logCount = n()) %>%
  dplyr::mutate(cdwVol = volumeFactor * logCount)


### Second, identify plots where all three transects do not have logs and bind to
### add these important zeroes back to the log count output from above
noLogs <- filterCDW %>%
  dplyr::filter(targetTaxaPresent == "N") %>%
  dplyr::group_by(domainID, siteID, volumeFactor, eventID, plotID) %>%
  dplyr::summarise(transectCount = n()) %>%
  dplyr::filter(transectCount == 3) %>%
  dplyr::mutate(logCount = 0,
                cdwVol = 0) %>%
  dplyr::select(-transectCount)

plotCdwVol <- plotCdwVol %>%
  dplyr::bind_rows(noLogs) %>%
```

```
  dplyr::arrange(domainID, siteID, eventID, plotID)


### Add back mammalGrid collocation data that was dropped during grouping and
### summarizing operations immediately above
temp <- tempCDW %>%
  dplyr::select(plotID, subtype) %>%
  dplyr::distinct()

plotCdwVol <- plotCdwVol %>%
  dplyr::left_join(temp, by = "plotID")


### Display table of plot-level CDW volume estimates
knitr::kable(
  x = plotCdwVol %>%
    dplyr::ungroup() %>%
    dplyr::rename(
      "Log Count" = logCount,
      "CDW Vol (m3/ha)" = cdwVol,
      "Plot subtype" = subtype
    ) %>%
    dplyr::select(-eventID) %>%
    dplyr::slice_head(n = 5),
  row.names = FALSE,
  caption = "Example subset of CDW volume density estimates by plotID for selected sites.
  The 'Plot subtype' column indicates whether a given CDW plot is colocated with a Small
  Mammal sampling grid."
)
```

Table 1: Example subset of CDW volume density estimates by plotID for selected sites. The 'Plot subtype' column indicates whether a given CDW plot is colocated with a Small Mammal sampling grid.

domainID	siteID	volumeFactor	plotID	Log Count	CDW Vol (m3/ha)	Plot subtype
D01	HARV	5	HARV_001	5	25	mammalGrid
D01	HARV	5	HARV_002	3	15	NA
D01	HARV	5	HARV_004	5	25	NA
D01	HARV	5	HARV_005	3	15	NA
D01	HARV	5	HARV_006	6	30	mammalGrid

```
### Finally, use plot-level CDW volume density calculated immediately above to
### calculate mean site-level CDW and standard deviations.
siteCdwVol <- plotCdwVol %>%
  dplyr::group_by(domainID, siteID, eventID) %>%
  dplyr::summarise(
    meanCdwVol = round(mean(cdwVol, na.rm = TRUE), digits = 1),
    sdCdwVol = round(sd(cdwVol, na.rm = TRUE), digits = 1),
    nCdwVol = n()
  )
```

```
### Display table of site-level CDW volume estimates
knitr::kable(
  x = siteCdwVol %>%
    dplyr::ungroup() %>%
    dplyr::rename(
      "Mean CDW Vol (m3/ha)" = meanCdwVol,
      "StdDev CDW Vol" = sdCdwVol,
      "Plot Number" = nCdwVol
    ),
  row.names = FALSE,
  caption = "CDW volume density estimates by siteID for selected sites and eventIDs."
)
```

Table 2: CDW volume density estimates by siteID for selected sites and eventIDs.

domainID	siteID	eventID	Mean CDW Vol (m3/ha)	StdDev CDW Vol	Plot Number
D01	HARV	CDW.2019.HARV	23.8	14.9	20
D03	JERC	CDW.2018.JERC	6.6	8.2	19
D05	STEI	CDW.2016.STEI	27.8	23.1	20
D06	UKFS	CDW.2018.UKFS	20.0	24.2	20
D10	RMNP	CDW.2020.RMNP	37.9	29.4	19
D16	ABBY	CDW.2019.ABBY	197.2	118.5	20

Create Small Mammal Density datasets

In this section, we employ the minimum number known alive (MNKA) approach to estimate total MAM abundance - e.g., as defined in Slade & Blair (2000). This approach assumes that a marked individual is present at all sampling points between its first and last capture dates, even if it was not actually captured in those interim trapping sessions. To make the dataset meet this assumption, we add those implicit records to the dataset to make them explicit.

Note: At the time of writing it is necessary to create Small Mammal `eventIDs` for legacy data collected prior to 2021. For the Small Mammal product, `eventIDs` correspond to discrete sampling bouts that represent a group of consecutive nights of trapping scheduled around a particular new moon. For the NEON 'RELEASE-2023' data, the omission of `eventID` from legacy data will be fixed and the code below that creates `eventIDs` will be moot.

In the code chunk below, we carry out the following steps:

- Use `mam_plotNight` data to create an `eventID` to group records according to unique sampling events. We assign the same `eventID` to each record with an `endDate` no more than 10 days later than the previously created record.
- Rectify the `trapStatus` field in the `mam_trapNight` table for records of captured animals that do not have the correct value.
- Filter the `mam_trapNight` data to retain records with the desired `trapStatus`, and records that have `taxonIDs` in the list of target taxa downloaded with the NEON Data Portal API. The latter step removes taxa from the dataset for which the MAM trapping protocol was not designed (e.g., shrews, weasels, etc.).
- Use `mam_trapNight` data and a `for()` loop to add in missing trap nights so that the data meet the assumptions described in Slade & Blair (2000) for calculating MNKA. The `for()` loop is commented out and the resulting dataset is read back in due to the inefficient computation.

- Use the `nightuid` to join the prepared `mam_trapNight` dataset with `mam_plotNight` to bring in the previously calculated `eventID`. The `eventID` is required for downstream computational steps.

```
### Create eventID grouping variable using mam_plotNight data and add to mam_trapNight
### Assign the same eventID to each record with an endDate no more than 10 days later
### than the previously created record. Below, we use the 'data.table::shift()'
### function to return the previous row value for the specified column (i.e., endDate)
mam_plotNight_nodups <- mam_plotNight_nodups %>%
  dplyr::mutate(year = as.integer(format(collectDate, "%Y")),
                .before = eventID) %>%
  dplyr::group_by(siteID, year) %>%
  dplyr::mutate(
    priorEndDate = data.table::shift(endDate, fill = endDate[1]),
    diffDays = difftime(endDate, priorEndDate, units = "days"),
    boutIncrement = ifelse(diffDays >= 10, 1, 0),
    boutNum = cumsum(boutIncrement),
    .after = endDate
  ) %>%
  dplyr::group_by(siteID, year, boutNum) %>%
  dplyr::mutate(
    weekBoutBegan = lubridate::isoweek(min(endDate)),
    eventID = case_when(
      is.na(eventID) ~ paste(siteID, year, weekBoutBegan, sep = "."),
      TRUE ~ eventID
    ),
    .after = boutNum
  ) %>%
  dplyr::ungroup() %>%
  dplyr::select(-priorEndDate, -diffDays, -boutIncrement, -boutNum)


### Add nightuid and eventID from mam_plotNight_nodups to mam_trapNight_nodups
mam_trapNight_nodups <- mam_trapNight_nodups %>%
  dplyr::left_join(mam_plotNight_nodups %>%
                     dplyr::select(nightuid, eventID),
                   by = "nightuid")


### Create and finalize Small Mammal capture dataset for selected sites. First,
### check to ensure all captures have the correct 'trapStatus' - i.e., check if tagIDs
### exist but trap status does not include "capture".
problemRecords <- mam_trapNight_nodups %>%
  dplyr::filter(!is.na(tagID),
                !grepl("capture", trapStatus))


### If nrow(problemRecords) > 0, update the corresponding trapStatus fields to
### "5 - capture" or, in the rare case where there are multiple captures in one trap,
### "4 - more than 1 capture in one trap" - this rare case does not occur in the
### current dataset, so it is not addressed here.
mam_trapNight_nodups <- mam_trapNight_nodups %>%
  dplyr::mutate(trapStatus = case_when(
    uid %in% problemRecords$uid ~ "5 - capture",
    TRUE ~ trapStatus
```

```
  ))


### Next, create the list of target taxa from taxon list downloaded from the API in
### a previous code chunk.
targetTaxa <- mam.list$data %>%
  dplyr::filter(taxonProtocolCategory == "target") %>%
  dplyr::select(taxonID)


### Next, subset the trapping data to only the capture records, i.e., the records that
### describe a captured small mammal, including only those taxa for which the trapping
### protocol is designed. Retain only those columns needed for the MNKA calculation.
captures <- mam_trapNight_nodups %>%
  dplyr::filter(grepl("capture", trapStatus) &
                  taxonID %in% targetTaxa$taxonID) %>%
  dplyr::select(uid, nightuid, plotID, collectDate, tagID)


### Next, per Slade & Blair (2000), generate a column of all the unique tagIDs
### included in the dataset, and create an empty data frame to populate with 'for()'
### loop output.
uTags <- captures %>%
  dplyr::select(tagID) %>%
  dplyr::filter(!is.na(tagID)) %>%
  dplyr::distinct()

capsNew <- dplyr::slice(captures, 0)


### Use the 'captures' data derived from mam_trapNight, and for each tagged
### individual, add a record for each night of trapping done in the plots
### in which it was captured between the first and last dates of capture.
# for (i in uTags$tagID) {
#   temp <- captures %>% filter(tagID == i)
#   firstCap <- as.Date(min(temp$collectDate), "YYYY-MM-DD", tz = "UTC")
#   lastCap <- as.Date(max(temp$collectDate), "YYYY-MM-DD", tz = "UTC")
#   possibleDates <- seq(as.Date(firstCap), as.Date(lastCap), by="days")
#   plots <- unique(temp$plotID)
#   potentialNights <- mam_plotNight_nodups %>%
#     dplyr::filter(as.character(collectDate) %in% as.character(possibleDates) &
#              plotID %in% plots) %>%
#     dplyr::select(nightuid,plotID, collectDate) %>%
#     dplyr::mutate(tagID = i)
#   temp2 <- dplyr::left_join(potentialNights, temp)
#   capsNew <- dplyr::bind_rows(capsNew, temp2)
# }

### Save a copy to avoid time-consuming 'for()' loop processing
# saveRDS(capsNew, file = "mam_final_dataset.RDS")


### Read in processed capsNew dataset
```

```
capsNew <- readRDS(file = "mam_final_dataset.RDS")


### Add untagged individuals back to the dataset
capsNew <- captures %>%
  dplyr::filter(is.na(tagID)) %>%
  dplyr::bind_rows(capsNew)


### Add eventID to enable summarizing by trapping bouts (includes up to 3 nights
### of trapping for some plots in each bout, 1 night for other plots).
capsNew <- capsNew %>%
  dplyr::left_join(mam_plotNight_nodups %>%
                     dplyr::select(eventID, nightuid),
                   by = "nightuid") %>%
  dplyr::relocate(eventID, .after = collectDate) %>%
  dplyr::arrange(eventID, plotID, collectDate)
```

In the code chunk below, we define a function to calculate MNKA for each plot within each bout:

- The function requires the `capsNew` data set generated in the code chunk above as an input.
- In addition, in the event that the `capsNew` data contain records from more plots than are needed for the MNKA estimation, the function requires a list of plots of interest (`plotsOI`) to filter the input dataset to only those plots for which MNKA outputs are desired.

```
### Define function to calculate MNKA for each plot within bout; 'plotsOI' is a required
### input vector of plotIDs for which MNKA per plot per bout is desired.

mnka_per_plot_per_bout <- function(capture_data, plotsOI) {
  caps <- capture_data %>%
    dplyr::filter(plotID %in% plotsOI)

  ids_by_plot_bout <- capture_data %>%
    dplyr::group_by(eventID, plotID) %>%
    dplyr::distinct(tagID)

  mnka_by_plot_bout <- ids_by_plot_bout %>%
    dplyr::group_by(eventID, plotID) %>%
    dplyr::count() %>%
    dplyr::mutate(
      siteID = stringr::str_extract(eventID, "^[A-Z]{4}"),
      year = as.numeric(stringr::str_extract(eventID, "20[0-9]{2}")),
      .before = eventID
    ) %>%
    dplyr::ungroup() %>%
    as.data.frame()

  return(mnka_by_plot_bout)
}

#--> Output is MNKA (count/hectare), based on the 1 ha size of the NEON trapping grid
```

In the code chunk below, we carry out additional MAM calculations:

- We use the MNKA small mammal density function defined immediately above to calculate MNKA at

the plot scale.

- We identify MAM data within 3-year windows by site based on the year that CDW volume density was sampled at each site.
- Within the 3-year window, we retain the maximum MAM abundance observed for each plot to compare to plot-level CDW volume density data.
  - Because CDW volume does not change rapidly, and MAM abundance may change substantially both within and among years, we generate the maximum small mammal abundance for a given plot for a three-year window centered on the year that logs were tallied for CDW.
  - Researchers may choose any number of different methods to integrate TOS data products with different temporal sampling frequencies (below, we also calculate mean, median, and minimum per-plot MAM abundance).
  - Additional methods beyond those shown here will all give a different statistical result.
- Plot-level MNKA MAM density data are used to calculate mean site-level MAM MNKA abundances. We also calculate several other MAM MNKA-based abundance metrics to illustrate the many choices available when working with these data.

```
### Calculate MNKA per plot per bout for all data in processed capsNew dataset
plotBoutMNKA <- mnka_per_plot_per_bout(capture_data = capsNew,
                                       plotsOI = unique(capsNew$plotID))


### Create table of MAM data within 3-year windows by site based on year of CDW volume
### sampling.
mamSiteYears <- plotCdwVol %>%
  dplyr::ungroup() %>%
  dplyr::distinct(domainID, siteID, eventID) %>%
  dplyr::mutate(
    cdwYear = as.numeric(stringr::str_extract(eventID, "20[0-9]{2}")),
    minMamYear = cdwYear - 1,
    maxMamYear = cdwYear + 1
  )


### Calculate plot-specific MNKA across bouts: Join plotBoutMNKA and mamSiteYears,
### filter to records within 3-year window centered on CDW measurement year, and
### determine maximum small mammal abundance for each plot across all sampling
### events, as well as other per plot metrics.
plotMNKA <- plotBoutMNKA %>%
  dplyr::left_join(mamSiteYears %>%
                     dplyr::select(siteID,
                                   minMamYear,
                                   maxMamYear),
                   by = "siteID") %>%
  dplyr::filter(year >= minMamYear,
                year <= maxMamYear) %>%
  dplyr::group_by(siteID, plotID) %>%
  dplyr::summarise(
    plotMeanAbundance = round(mean(n), digits = 1),
    plotMedAbundance = median(n),
    plotMinAbundance = min(n),
    plotMaxAbundance = max(n),
    plotBoutCount = n()
  ) %>%
  dplyr::ungroup()
```

```
### Display table of plot-level MNKA parameters
knitr::kable(
  x = plotMNKA %>%
    dplyr::ungroup() %>%
    dplyr::rename(
      "Mean Abundance (count/ha)" = plotMeanAbundance,
      "Median Abundance (count/ha)" = plotMedAbundance,
      "Min Abundance (count/ha)" = plotMinAbundance,
      "Max Abundance (count/ha)" = plotMaxAbundance,
      "# Bouts Sampled" = plotBoutCount
    ) %>%
    dplyr::slice_head(n = 5),
  row.names = FALSE,
  caption = "Example subset of Small Mammal abundance metrics (count/ha) by plotID for
  selected sites for a 3-year window centered on the year that CDW volume was measured."
)
```

Table 3: Example subset of Small Mammal abundance metrics (count/ha) by plotID for selected sites for a 3-year window centered on the year that CDW volume was measured.

siteID	plotID	Mean Abundance (count/ha)	Median Abundance (count/ha)	Min Abundance (count/ha)	Max Abundance (count/ha)	# Bouts Sampled
ABBY	ABBY_002	9.2	9.0	2	18	10
ABBY	ABBY_004	32.0	31.5	9	58	10
ABBY	ABBY_007	7.1	6.0	3	11	9
ABBY	ABBY_010	8.5	7.5	2	26	10
ABBY	ABBY_014	4.2	5.0	1	10	9

```
### Calculate MNKA metrics per site based on plot-level data:
### siteMeanAbundance = mean abundance across plots within a site
### siteMedAbundance = median abundance across plots within a site
### siteMinAbundance = average minimum abundance across plots within a site
### siteMaxAbundance = average maximum abundance across plots within a site
siteMNKA <- plotMNKA %>%
  dplyr::group_by(siteID) %>%
  dplyr::summarise(
    siteMeanAbundance = round(mean(plotMeanAbundance), digits = 1),
    siteMedAbundance = round(median(plotMedAbundance), digits = 1),
    siteMinAbundance = round(mean(plotMinAbundance), digits = 1),
    siteMaxAbundance = round(mean(plotMaxAbundance), digits = 1)
  ) %>%
  dplyr::ungroup()


### Display table of site-level MNKA parameters
knitr::kable(
  x = siteMNKA %>%
    dplyr::ungroup() %>%
    dplyr::rename(
```

```
      "Mean Abundance (count/ha)" = siteMeanAbundance,
      "Median Abundance (count/ha)" = siteMedAbundance,
      "Average Min Abundance (count/ha)" = siteMinAbundance,
      "Average Max Abundance (count/ha)" = siteMaxAbundance
    ),
  row.names = FALSE,
  caption = "Small Mammal abundance metrics (count/ha) by siteID for selected sites
  for a 3-year window centered on the year that CDW volume was measured."
)
```

Table 4: Small Mammal abundance metrics (count/ha) by siteID for selected sites for a 3-year window centered on the year that CDW volume was measured.

siteID	Mean Abundance (count/ha)	Median Abundance (count/ha)	Average Min Abundance (count/ha)	Average Max Abundance (count/ha)
ABBY	13.5	8.2	4.2	26.3
HARV	10.4	8.5	1.0	28.7
JERC	17.8	14.5	4.7	41.3
RMNP	14.1	8.0	1.8	37.0
STEI	20.7	15.8	3.5	52.8
UKFS	8.5	5.8	1.5	22.7

Create unified CDW and MAM datasets

To create plot-level and site-level data sets suitable for plotting and correlation analyses, in the code chunk below we join the CDW volume and MAM abundance data at both plot and site scales.

```
### Join plot-level CDW and MAM datasets
plotDF <- plotCdwVol %>%
  dplyr::left_join(plotMNKA %>%
                     dplyr::select(-plotBoutCount),
                   by = c("siteID", "plotID")) %>%
  dplyr::filter(!is.na(plotMaxAbundance))


### Join site-level CDW and MAM datasets
siteDF <- siteCdwVol %>%
  dplyr::left_join(siteMNKA,
                   by = "siteID")
```

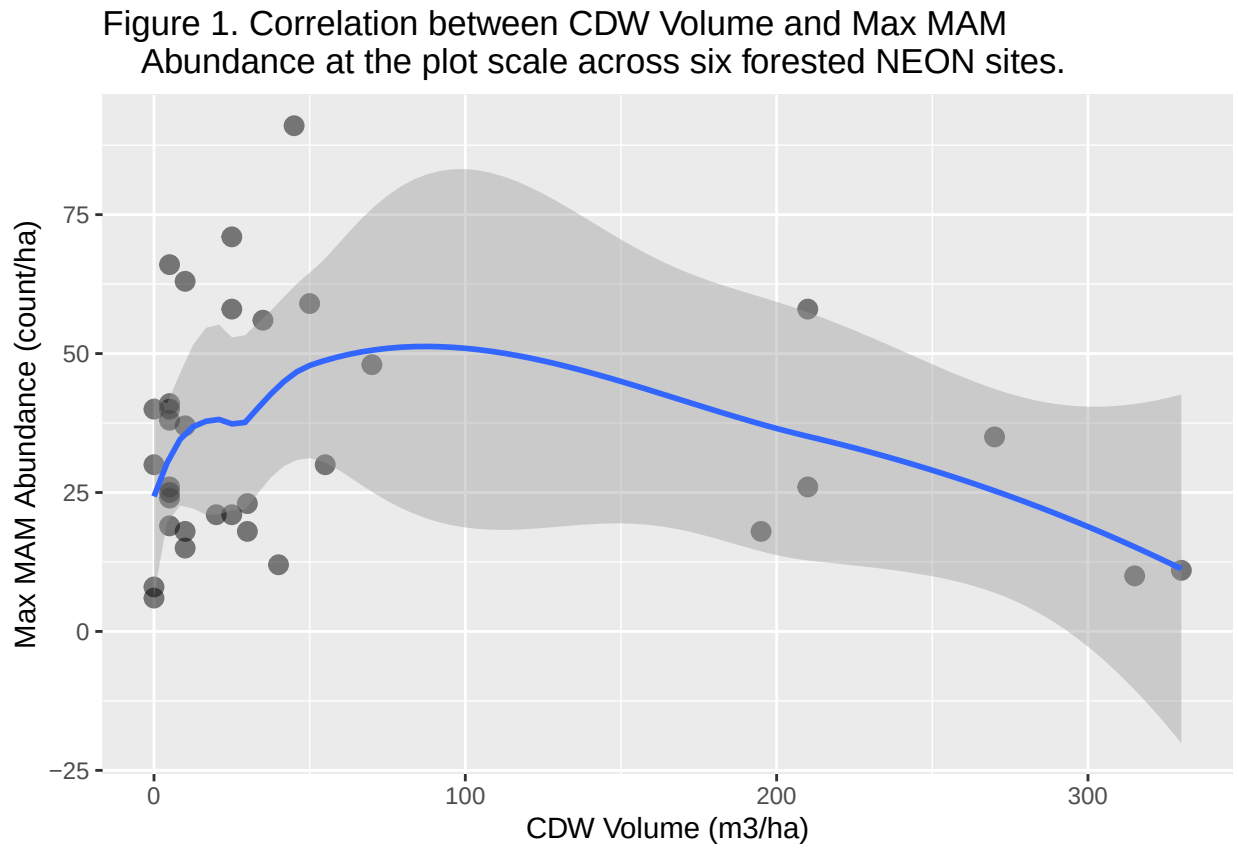
Create graphs of CDW and MAM data at plot and site spatial scales

We first examine the potential relationship between CDW volume and MAM abundance using data from all spatially colocated plots from all of the six forested sites we selected (HARV, JERC, STEI, UKFS, RMNP, and ABBY).

```
### Create graph of CDW volume vs. MAM maximum abundance across all plots and sites
plotsGraph <- ggplot2::ggplot(data = plotDF,
                              mapping = aes(x = cdwVol,
                                            y = plotMaxAbundance)) +
  ggplot2::geom_point(alpha = 0.5,
```

```
                      size = 3) +
  ggplot2::geom_smooth() +
  ggplot2::labs(
    x = "CDW Volume (m3/ha)",
    y = "Max MAM Abundance (count/ha)",
    title = "Figure 1. Correlation between CDW Volume and Max MAM
    Abundance at the plot scale across six forested NEON sites."
  )

plotsGraph
```



Plot-level results across all sites:

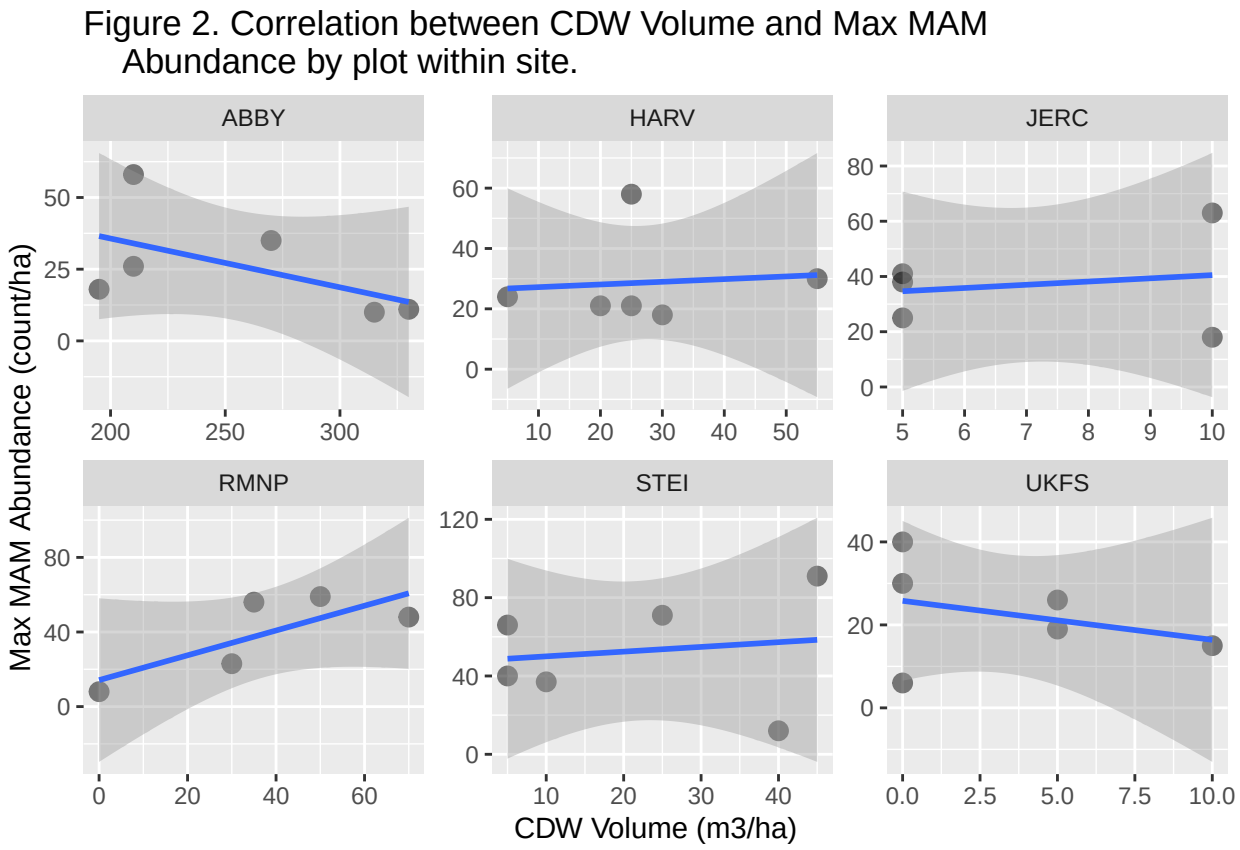
- Across a selection of close-canopy forested NEON sites, measurements of CDW volume (m^3/ha) and MAM abundance (count/ha) reveal no clear correlation at the colocated plot-scale.
- The Pearson's correlation coefficient is: -0.16.

In the next code chunk below, we check for correlations between CDW volume and Small Mammal abundance within sites using colocated plot-scale data.

```
### Create graph of CDW volume vs. MAM maximum abundance across plots within site
sitePlotsGraph <- ggplot2::ggplot(data = plotDF,
                                  mapping = aes(x = cdwVol,
                                                y = plotMaxAbundance)) +
  ggplot2::geom_point(alpha = 0.5,
                      size = 3) +
  ggplot2::geom_smooth(method = lm) +
  ggplot2::facet_wrap(facets = ~ siteID,
```

```
                      ncol = 3,
                      scales = "free") +
  ggplot2::labs(
    x = "CDW Volume (m3/ha)",
    y = "Max MAM Abundance (count/ha)",
    title = "Figure 2. Correlation between CDW Volume and Max MAM
    Abundance by plot within site."
  )

sitePlotsGraph
```



Plot-level results within site:

- Within a selection of close-canopy forested NEON sites, CDW volume (m^3/ha) and MAM abundance (count/ha) show positive, negative, and weak correlations.
- Positive correlation: RMNP, $r_{Pearson} = 0.77$).
- Negative correlation: ABBY, $r_{Pearson} = -0.55$).
- Weak correlations (HARV, JERC, STEI, UKFS).

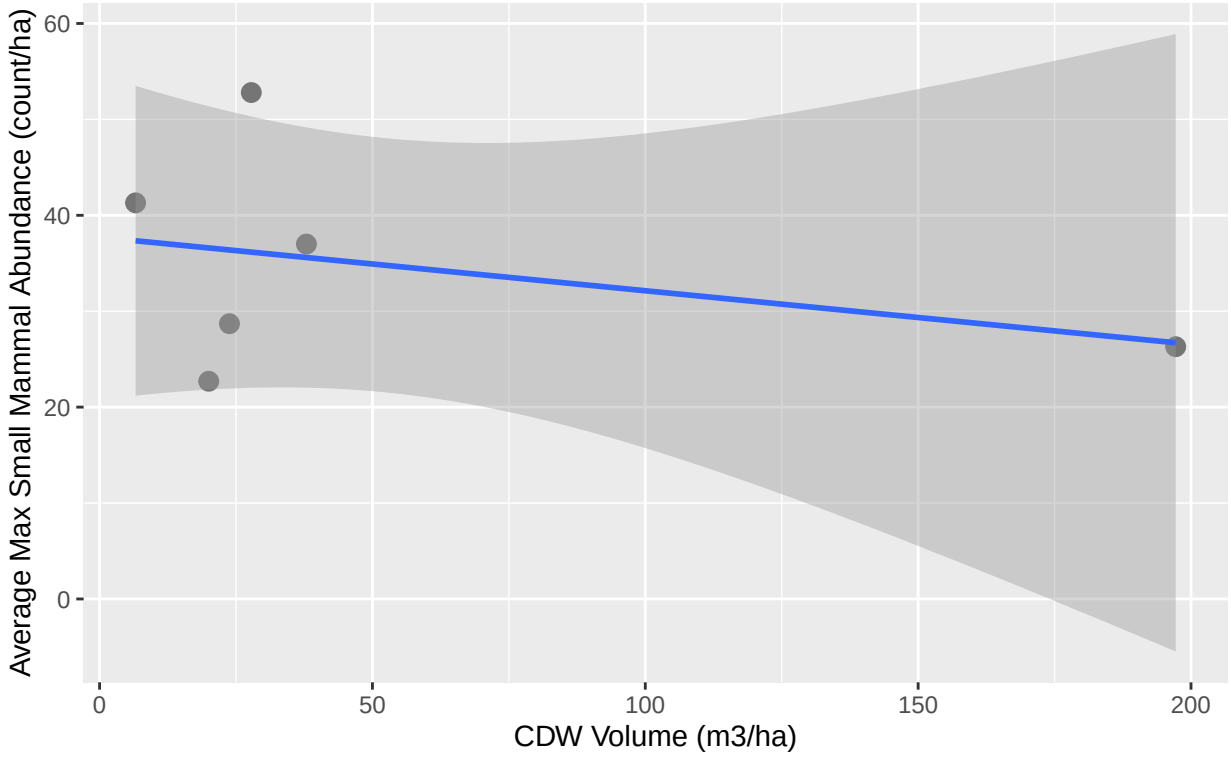
In the final code chunk below, we check for correlations between mean CDW volume and average maximum MAM abundance at the site scale.

```
### Create graph of CDW volume vs. MAM maximum abundance across sites
sitesGraph <- ggplot2::ggplot(data = siteDF,
                              mapping = aes(x = meanCdwVol,
                                            y = siteMaxAbundance,)) +
  ggplot2::geom_point(alpha = 0.5,
                      size = 3) +
```

```
  ggplot2::geom_smooth(method = lm) +
  ggplot2::labs(x = "CDW Volume (m3/ha)",
                y = "Average Max Small Mammal Abundance (count/ha)",
                title = "Figure 3. Correlation between CDW Volume and average max Small Mammal
    Abundance across a selection of forested NEON sites.")

sitesGraph
```

Figure 3. Correlation between CDW Volume and average max Small Mamm Abundance across a selection of forested NEON sites.



Site-level results:

- CDW volume (m^3/ha) and MAM abundance (count/ha) show a weak, negative correlation at the site scale, $r_{Pearson}$ = -0.36.

Conclusions

- Coarse Downed Wood volume (m^3/ha) and Small Mammal abundance (count/ha) can each be calculated at the plot-level, and spatially colocated plots are identified by a shared `plotID`.
  - It is possible to identify shared `plotID` values programmatically using the `geoNEON::getLocByName()` function (lines 183-188), or by downloading the CDW and Small Mammal datasets and joining via the `plotID` (lines 912-917).
  - Note that more `plotID` values may be identified as colocated via the `geoNEON` function than via the table joining method due to the fact that `geoNEON::getLocByName()` identifieds all `plotIDs` that have ever been used for a given protocol, and not just those locations that are currently in use.
- Plot-scale spatial collocation of TOS data products allows data users to investigate patterns within NEON sites.

- All NEON TOS data products are colocated at the site-scale. Site-level estimates may be useful for investigating patterns at very large spatial scales.
- Numerous methods for integrating data collected with different temporal frequencies are possible. Researchers should carefully evaluate methods suitable to the question of interest.

References

Affleck, D.L.R. (2008) A line intersect distance sampling strategy for downed wood inventory. *Canadian Journal of Forest Research*, 38, 2262-2273.

Affleck, D.L.R. (2010) On the efficiency of line intersect distance sampling. *Canadian Journal of Forest Research*, 40, 1086-1094.

Slade, N.A. and Blair, S.M. (2000) An empirical test of using counts of individuals captured as indices of population size. *Journal of Mammalogy*, 81, 1035–1045.

Reproducibility

Code was developed and tested using `R v4.2.1`, and key packages with versions as shown below. The code in this example may generate errors if key package versions differ from those indicated. Source code is available via GitHub: https://github.com/NEONScience/ecosphere-tos-2022

Table 5: Key package versions used to develop and test example code.

Package	Version	Depends	Built
data.table	1.14.6	R (>= 3.1.0)	4.2.0
devtools	2.4.5	R (>= 3.0.2), usethis (>= 2.1.6)	4.2.0
geoNEON	1.0.0.9100	R (>= 3.5.0)	4.2.0
knitr	1.41	R (>= 3.3.0)	4.2.1
neonOS	1.0.0	R (>= 3.6)	4.2.0
neonUtilities	2.2.0	R (>= 3.4.0)	4.2.0
tidyverse	1.3.2	R (>= 3.3)	4.2.0