STOICH-Utilities Examples

Chad Petersen

2025-11-12

Setting the path to the data.

```
# Load required packages.
library(tidyverse)
library(stoichUtilities)

# Three options for creating a variable to store the path to the STOICH data.
# With a text string.
basePath <- "C:/Users/chad/Documents/data/STOICH_Release_2025-09-10"
# Building the path starting at your Documents directory.
basePath <- file.path(path.expand("~"), "data", "STOICH_Release_2025-09-10")
# Or if you set the working directory to point to the data.
setwd(basePath)
basePath <- getwd()

# Load the STOICH data (using a predefined path variable)
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# stoichData is now a list of tables. To see the names of those tables use names.
names(stoichData)
```

```
## [1] "metadata"                  "join_tables"
## [3] "tbl_InputFile"             "tbl_Site"
## [5] "tbl_SampleEvent"           "tbl_OrganismStoichiometry"
## [7] "tbl_WaterChemistry"        "tbl_Source"
## [9] "tbl_Contact"
```

```
# Print the first 5 rows of the Source table.
stoichData[["tbl_Source"]] |> slice_head(n=5)
```

```
## # A tibble: 5 x 15
##      Id ContactId Type      Citation DOI   Url   Title FirstAuthor YearPublished
##   <int>     <int> <chr>     <chr>    <chr> <chr> <chr> <chr>               <int>
## 1     1         1 raw_repo~ NEON (N~ 10.4~ http~ aqua~ Neon (Nati~          2025
## 2     2         1 raw_repo~ NEON (N~ 10.4~ http~ chem~ Neon (Nati~          2025
## 3     3         1 raw_repo~ NEON (N~ 10.4~ http~ peri~ Neon (Nati~          2025
## 4     4         2 raw_auth~ <NA>     10.1~ http~ larg~ Camilleri,~          2019
## 5     5         3 raw_auth~ <NA>     10.1~ http~ calc~ Corman, Je~          2016
## # i 6 more variables: Journal <chr>, Publisher <chr>, License <chr>,
## #   LitSurveyId <int>, Organization <chr>, Notes <chr>
```

Exploring the Data

```
# Load required packages.
library(tidyverse)
library(summarytools)
library(DataExplorer)

library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Join the list of tables into one wide table.
stoichTable <- stoichData |>
  stoichUtilities::joinSTOICH()
```

```
# Create a summary of the STOICH data table using summarytools formated for easy viewing (output only s
view(summarytools::dfSummary(stoichTable))
```

dfSummary output

```
# Create a report of the STOICH data table using DataExplorer (output only uses some Source/Contact col
DataExplorer::create_report(stoichTable)
```

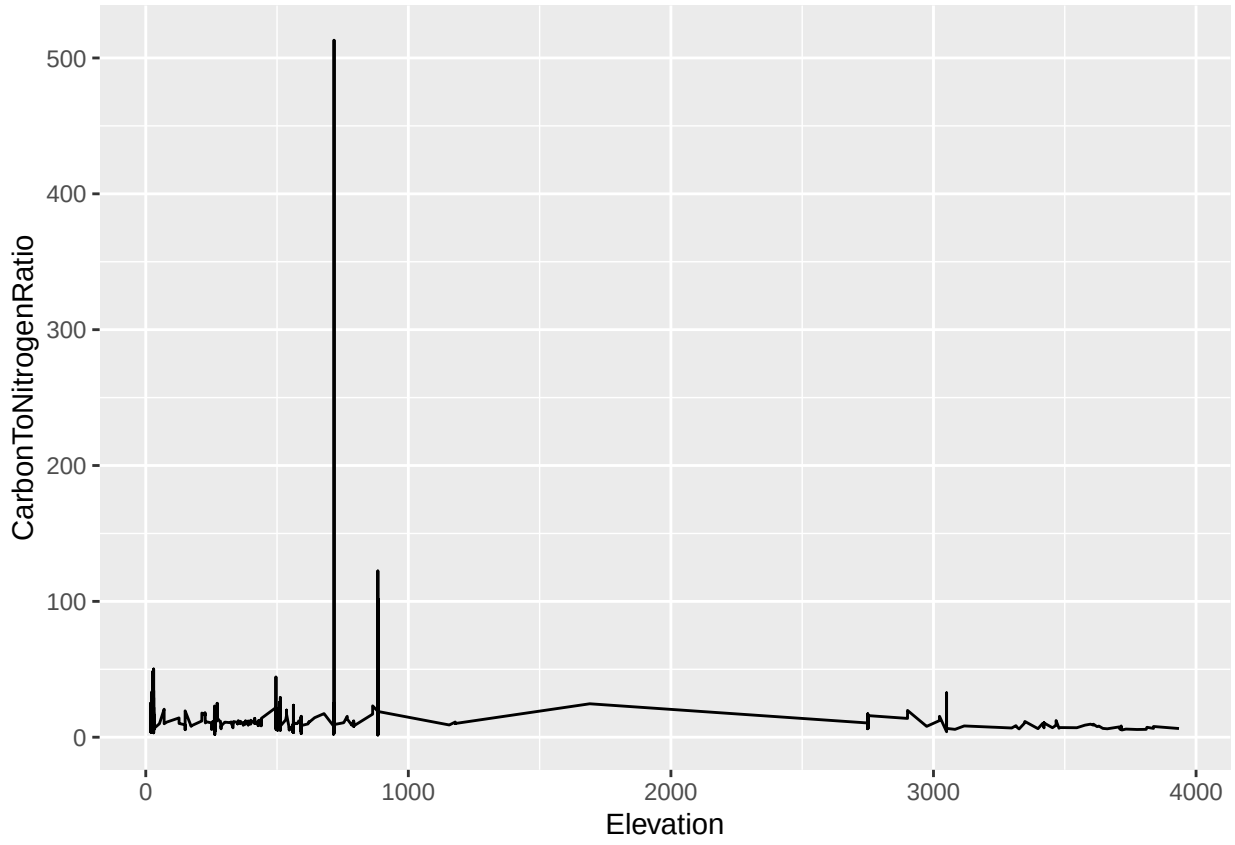
DataExplorer report output

Plotting Example

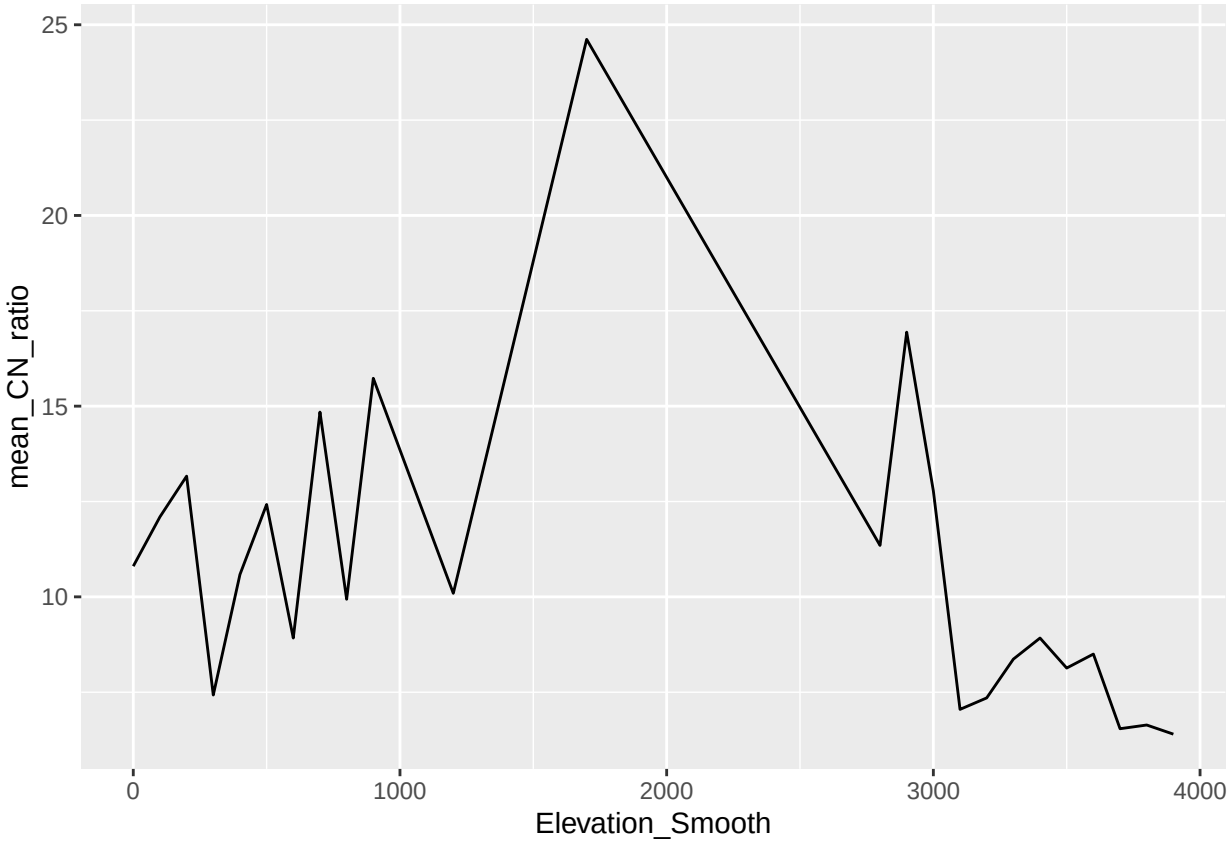
```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Select/filter the data you want.
stoichTable <- stoichData |>
  # Filter the variables we are interested in to remove any rows that don't have values.
  stoichUtilities::filterSTOICH(var="Elevation", val=NA, condition="not equal") |>
  stoichUtilities::filterSTOICH(var="Latitude", val=NA, condition="not equal") |>
  stoichUtilities::filterSTOICH(var="CarbonToNitrogenRatio", val=NA, condition="not equal") |>
  # Filter to select only rows where the Organism Type is "Seston".
  stoichUtilities::filterSTOICH(var="Type.OrganismStoichiometry", val="Seston", condition="equal") |>
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH() |>
  # Select only the columns we are interested in plotting.
```

```
  select(c("Elevation.Site", "Latitude.Site", "CarbonToNitrogenRatio.OrganismStoichiometry")) |>
  # Remove any text with 4 characters or more after a "." at the end ("$") of a string.
  rename_with(~stringr::str_remove(.x, "\\.[a-zA-Z]{4,}$"))

# Plot the elevation versus CN ratio as a line.
ggplot(stoichTable, aes(x=Elevation, y=CarbonToNitrogenRatio)) +
  geom_line()
```




```
# If you want to look at a plot with average CN for elevation bins, you could use the summarize functio
smoothTable <- stoichTable |>
  mutate(Elevation_Smooth = 100*round(Elevation/100)) |>
  group_by(Elevation_Smooth) |>
  summarize(mean_CN_ratio = mean(CarbonToNitrogenRatio))

# Plot the averaged data.
ggplot(smoothTable, aes(x=Elevation_Smooth, y=mean_CN_ratio)) +
  geom_line()
```



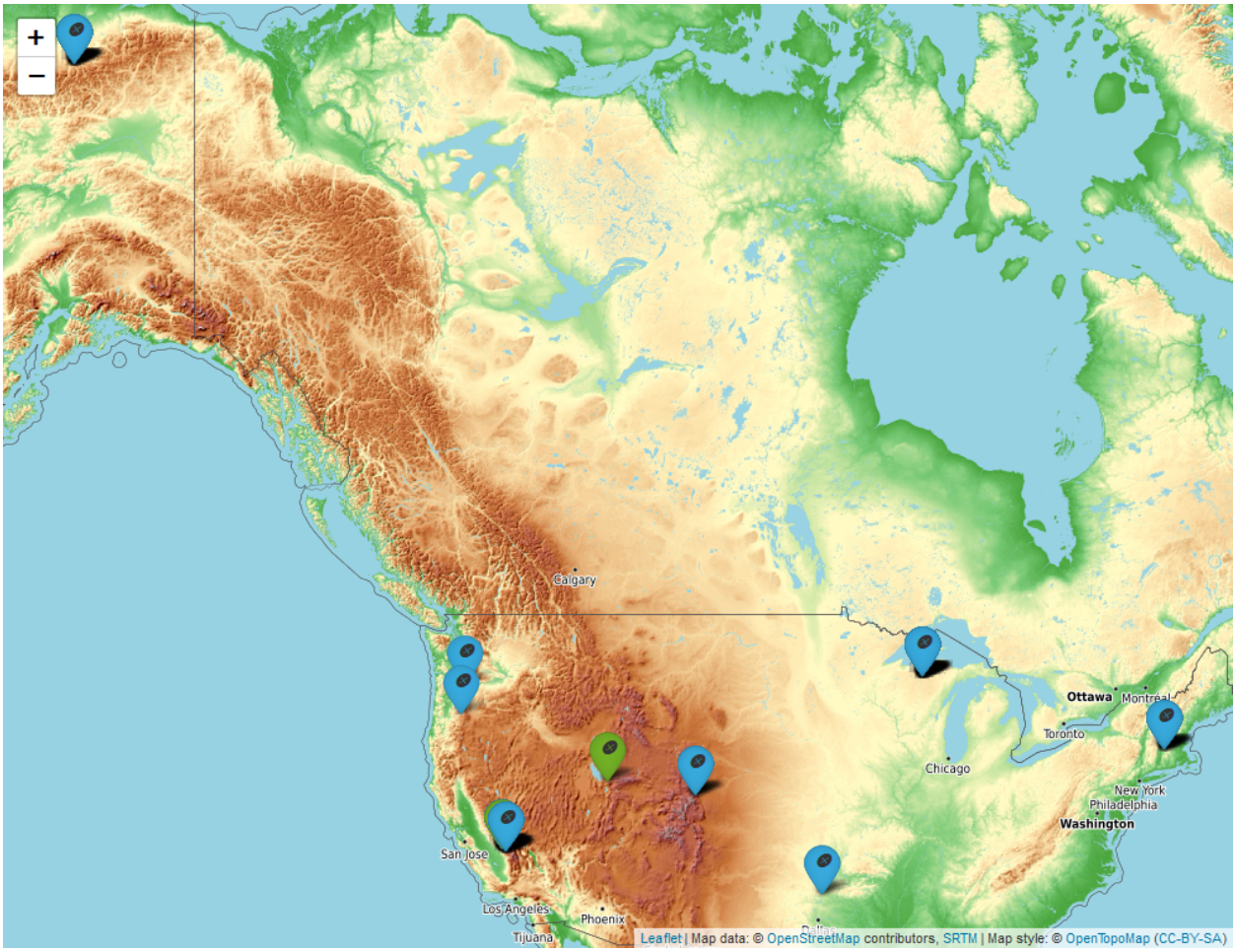
Mapping Example

```
library(tidyverse)
library(stoichUtilities)
library(sf)
library(leaflet)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Select/filter the data you want.
stoichTable <- stoichData |>
  # Remove any locations without GPS data
  stoichUtilities::filterSTOICH(var="Longitude", val=NA, condition="not equal") |>
  stoichUtilities::filterSTOICH(var="Latitude", val=NA, condition="not equal") |>
  # Filter to select three orders of Bryophytes.
  stoichUtilities::filterSTOICH(var="TaxonomicOrder", val=c("hypnales", "marchantiales", "polygonales")
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH() |>
  # From here we pull out just the data we are interested in: GPS & taxonomy and shorten the variable n
  # Shorten the variable names.
  rename(Latitude=Latitude.Site, Longitude=Longitude.Site, TaxonomicOrder=TaxonomicOrder.OrganismStoichi
```

```r
  # Select the columns we are interested in.
  select(c("Latitude", "Longitude", "TaxonomicOrder"))

# Initiate the leaflet plot.
myPlot <- leaflet::leaflet() |>
  # Select the background map style.
  leaflet::addProviderTiles(providers$OpenTopoMap)

# Create colored markers (one for each order selected).
blueIcon <- awesomeIcons(icon = 'ios-close', iconColor = 'black', library = 'ion', markerColor = "blue")
greenIcon <- awesomeIcons(icon = 'ios-close', iconColor = 'black', library = 'ion', markerColor = "green
orangeIcon <- awesomeIcons(icon = 'ios-close', iconColor = 'black', library = 'ion', markerColor = "ora

# Add markers to the plot.
myPlot <- myPlot |>
  # Select (filter) only one order from the data.frame or table for each marker color.
  leaflet::addAwesomeMarkers(data = data.frame(stoichTable |> filter(TaxonomicOrder == "hypnales")), lng
  leaflet::addAwesomeMarkers(data = data.frame(stoichTable |> filter(TaxonomicOrder == "marchantiales"))
  leaflet::addAwesomeMarkers(data = data.frame(stoichTable |> filter(TaxonomicOrder == "polygonales")),
```

```r
# Display the plot.
myPlot
```



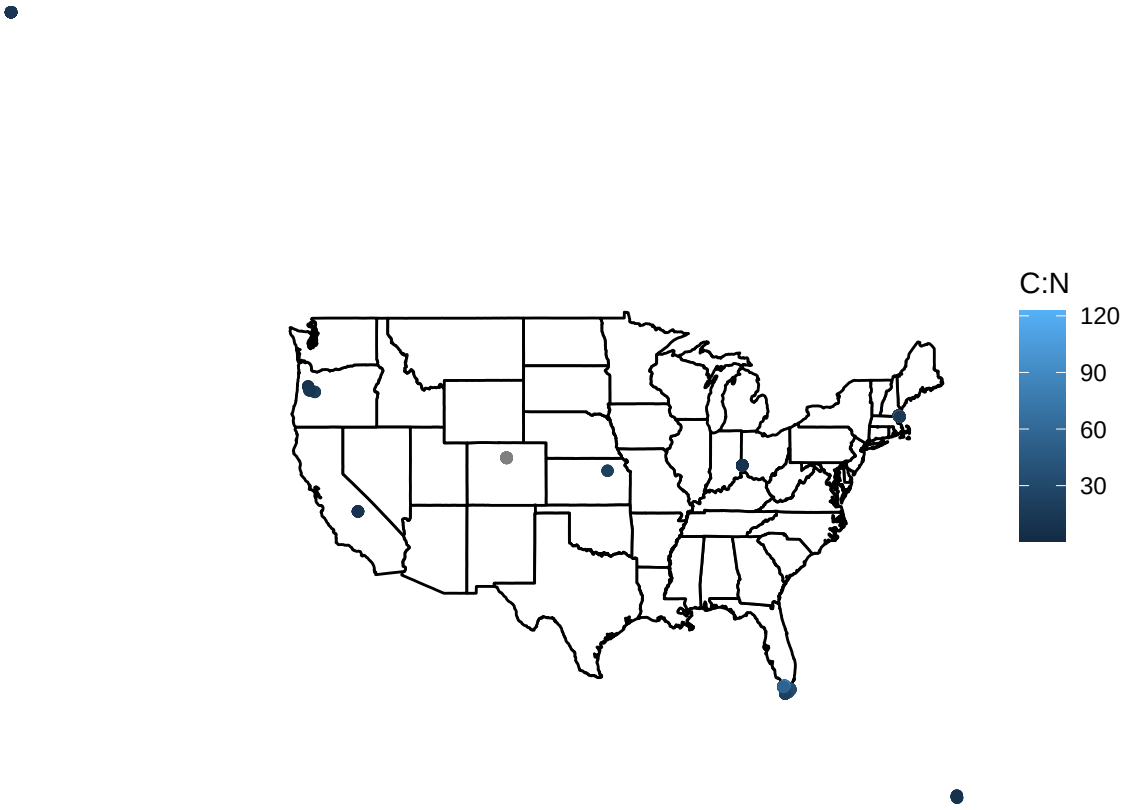
Mapping Example 2

```
library(tidyverse)
library(lubridate)
library(maps)
library(gganimate)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Select/filter the data you want.
stoichTable <- stoichData |>
  stoichUtilities::filterSTOICH(var="Type.OrganismStoichiometry", val="Seston", condition="equal") |>
  stoichUtilities::filterSTOICH(var="Country", val="United States Of America", condition="equal") |>
  stoichUtilities::joinSTOICH()

# Load the state borders
state_info <- map_data("state")
```

```
# Map the state borders
seston_map <- ggplot(data = state_info, mapping = aes(x=long, y=lat, group=group)) +
  geom_polygon(color = "black", fill = "white") +
  coord_quickmap() +
  theme_void() +
  # Add the Seston data to the map of state borders and set the color using the C:N ratio.
  geom_point(data = stoichTable, aes(x = Longitude.Site, y = Latitude.Site, color=CarbonToNitrogenRatio
  labs(color="C:N")

# Add animation using the year the sample was taken
map_with_animation <- seston_map +
  transition_time(SampleYear.SampleEvent) +
  ggtitle('Year: {frame_time}',
          subtitle = 'Frame {frame} of {nframes}')

# Calculate the number of years there was sample data to determine the number of frames for the animati
num_years <- max(stoichTable$SampleYear.SampleEvent) - min(stoichTable$SampleYear.SampleEvent) + 1

# Create the animation.
animate(map_with_animation, nframes = num_years)
```

```
## Warning: Removed 750 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Filtering Example

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)

# Select/filter the data you want.
stoichTable <- stoichData |>
  # Filter to select only rows where the EcosystemType is "Stream".
  stoichUtilities::filterSTOICH(var="EcosystemType", val="Stream", condition="equal") |>
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH() |>
  View()

# Only displaying the first 4 rows.
```

```
## # A tibble: 4 x 127
##   Id.Contact FirstName.Contact LastName.Contact Email.Contact      Id.Source
##        <int> <chr>             <chr>            <chr>                  <int>
```

```
## 1         14 Ada               Pastor           ada.pastor@udg.edu        20
## 2         14 Ada               Pastor           ada.pastor@udg.edu        20
## 3         14 Ada               Pastor           ada.pastor@udg.edu        20
## 4         14 Ada               Pastor           ada.pastor@udg.edu        20
## # i 122 more variables: Type.Source <chr>, Citation.Source <chr>,
## #   DOI.Source <chr>, Url.Source <chr>, Title.Source <chr>,
## #   FirstAuthor.Source <chr>, YearPublished.Source <int>, Journal.Source <chr>,
## #   Publisher.Source <chr>, License.Source <chr>, LitSurveyId.Source <int>,
## #   Organization.Source <chr>, Notes.Source <chr>, Id.SampleEvent <int>,
## #   SampleDay.SampleEvent <int>, SampleMonth.SampleEvent <int>,
## #   SampleYear.SampleEvent <int>, Id.Site <int>, Habitat.SampleEvent <chr>, ...
```

Summarizing Data

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data
stoichTable <- stoichUtilities::loadSTOICH(dataPath=basePath) |>
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH() |>
  # Use the dplyr package from the tidyverse to select the columns we are interested in.
  dplyr::select(c("Title.Source", "FirstAuthor.Source", "YearPublished.Source", "Journal.Source", "LitSu
  # Shorten column names because I don't want to keep typing ".Source".
  dplyr::rename_all(~stringr::str_remove(.x, "\\.Source")) |>
  # There are probably multiple samples/site so only keep unique entries of sites and sources.
  unique() |>
  # Group the data by Source columns.
  dplyr::group_by(Title, FirstAuthor, YearPublished, Journal, LitSurveyId, Organization) |>
  # Count the number of sites for each group.
  dplyr::summarise(Number_of_Sites = n(), .groups="keep") |>
  View()

# Only displaying the first 4 rows.
```

```
## # A tibble: 4 x 7
##   Title               FirstAuthor YearPublished Journal LitSurveyId Organization
##   <chr>               <chr>               <int> <chr>         <int> <chr>
## 1 a database of west~ Beck, Miri~          2022 global~          NA Université ~
## 2 a stream insect de~ Halvorson,~          2015 freshw~          NA higher_educ~
## 3 alligator pond foo~ Strickland~          2023 <NA>             NA higher_educ~
## 4 allochthonous and ~ Neres-Lima~          2017 freshw~          NA Laboratório~
## # i 1 more variable: Number_of_Sites <int>
```

Search Example

Solution 1: pulling a table from the raw data.

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

# Load the data
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)

# Select just the Site table.
stoichData[["tbl_Site"]] |>
  # Find the row with the maximum Longitude value (drop rows with NA values).
  filter(Longitude == max(stoichData[["tbl_Site"]][["Longitude"]], na.rm=TRUE)) |>
  View()
```

```
## # A tibble: 1 x 9
##      Id Name     Latitude Longitude State Country  EcosystemType Elevation Notes
##   <int> <chr>       <dbl>     <dbl> <chr> <chr>    <chr>             <dbl> <chr>
## 1  3102 deadmans    -41.8      172. <NA>  New Zea~ Stream               NA aqua~
```

Solution 2: working with the joined table.

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

#Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)

stoichTable <- stoichData |>
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH() |>
  # Select just the columns from the Site table.
  select(ends_with(".Site"))

# Find the row with the maximum Longitude value
stoichTable |>
  filter(Longitude.Site == max(stoichTable[["Longitude.Site"]], na.rm=TRUE)) |>
  # There will probably be duplicates, so keep only unique values.
  unique() |>
  View()
```

```
## # A tibble: 1 x 9
##   Id.Site Name.Site Latitude.Site Longitude.Site State.Site Country.Site
##     <int> <chr>             <dbl>          <dbl> <chr>      <chr>
## 1    3102 deadmans          -41.8           172. <NA>       New Zealand
## # i 3 more variables: EcosystemType.Site <chr>, Elevation.Site <dbl>,
## #   Notes.Site <chr>
```

Filter and Summarize Example

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

# Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
#Load the data and summarize elevation data for samples in December with Latitude > 45 degrees.
stoichTable <- stoichData |>
  # Filter just December (month 12) data.
  stoichUtilities::filterSTOICH(var="SampleMonth", val=as.integer(12), condition="equal") |>
  # Filter just samples north of 45 degrees latitude.
  stoichUtilities::filterSTOICH(var="Latitude", val=45, condition="greater than") |>
  # Join all the tables into one large wide table.
  stoichUtilities::joinSTOICH()

# Count the number of rows.
nrow(stoichTable)
```

```
## [1] 68
```

```
# Calculate the mean elevation for the samples.
mean(stoichTable[["Elevation.Site"]], na.rm=TRUE)
```

```
## [1] 448.45
```

Filter and Search Example

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

# Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Load the data and pipe it ("|>" and "%>%" can pass a table into the next function) for further proces
stoichTable <- stoichData |>
  # Filter the STOICH data for any samples in Texas with Isotope13C content and remove any samples with
  stoichUtilities::filterSTOICH(var="State", val="Texas", condition="Equal") |>
  stoichUtilities::filterSTOICH(var="Isotope13C", val=NA, condition="not equal") |>
  stoichUtilities::filterSTOICH(var="TaxonomicSpecies", val=NA, condition="not equal") |>
  # Join all the tables into one large wide table
  stoichUtilities::joinSTOICH() |>
  # Use the dplyr package from the tidyverse to select the columns we are interested in.
  dplyr::select(c("TaxonomicKingdom.OrganismStoichiometry",
```

```
                  "TaxonomicOrder.OrganismStoichiometry",
                  "TaxonomicFamily.OrganismStoichiometry",
                  "TaxonomicGenus.OrganismStoichiometry",
                  "TaxonomicSpecies.OrganismStoichiometry",
                  "Isotope13C.OrganismStoichiometry"))
```

```
# Find the row with the minimum Isotope13C value
stoichTable |>
  filter(Isotope13C.OrganismStoichiometry == min(stoichTable[["Isotope13C.OrganismStoichiometry"]])) |>
  View()
```

```
## # A tibble: 1 x 6
##   TaxonomicKingdom.OrganismStoic~1 TaxonomicOrder.Organ~2 TaxonomicFamily.Orga~3
##   <chr>                            <chr>                   <chr>
## 1 plantae                          alismatales             potamogetonaceae
## # i abbreviated names: 1: TaxonomicKingdom.OrganismStoichiometry,
## #   2: TaxonomicOrder.OrganismStoichiometry,
## #   3: TaxonomicFamily.OrganismStoichiometry
## # i 3 more variables: TaxonomicGenus.OrganismStoichiometry <chr>,
## #   TaxonomicSpecies.OrganismStoichiometry <chr>,
## #   Isotope13C.OrganismStoichiometry <dbl>
```

Filter, Summarize and Search Example

```
library(tidyverse)
library(stoichUtilities)

basePath <- "C:\..." # Update this with your path to the data

# Load the data.
stoichData <- stoichUtilities::loadSTOICH(dataPath=basePath)
```

```
# Load the data and pipe it ("|>" and "%>%" can pass a table into the next function) for further proces
stoichTable <- stoichData |>
  # Filter the STOICH data for any samples in California with Isotope15N content and remove any samples
  stoichUtilities::filterSTOICH(var="State", val="California", condition="Equal") |>
  stoichUtilities::filterSTOICH(var="Isotope15N", val=NA, condition="not equal") |>
  stoichUtilities::filterSTOICH(var="TaxonomicSpecies", val=NA, condition="not equal") |>
  # Join all the tables into one large wide table
  stoichUtilities::joinSTOICH() |>
  # Use the dplyr package from the tidyverse to select the columns we are interested in.
  dplyr::select(c("TaxonomicKingdom.OrganismStoichiometry",
                  "TaxonomicOrder.OrganismStoichiometry",
                  "TaxonomicFamily.OrganismStoichiometry",
                  "TaxonomicGenus.OrganismStoichiometry",
                  "TaxonomicSpecies.OrganismStoichiometry",
                  "Isotope15N.OrganismStoichiometry")) |>
  # Shorten column names because I don't want to keep typing ".OrganismStoichiometry".
  dplyr::rename_all(~stringr::str_remove(.x, "\\.OrganismStoichiometry")) |>
  # Group the data by Taxonomy.
  dplyr::group_by(TaxonomicKingdom, TaxonomicOrder, TaxonomicFamily, TaxonomicGenus, TaxonomicSpecies)
```

```r
  # Average the N15 Isotope data for each group.
  dplyr::summarise(mean_Isotope15N = mean(Isotope15N), .groups="keep")
```

```r
# Find the row with the maximum average Isotope15N value
stoichTable |>
  filter(mean_Isotope15N == max(stoichTable[["mean_Isotope15N"]])) |>
  View()
```

```
## # A tibble: 1 x 6
## # Groups:   TaxonomicKingdom, TaxonomicOrder, TaxonomicFamily, TaxonomicGenus,
## #   TaxonomicSpecies [1]
##   TaxonomicKingdom TaxonomicOrder TaxonomicFamily TaxonomicGenus
##   <chr>            <chr>          <chr>           <chr>
## 1 plantae          solanales      convolvulaceae  cuscuta
## # i 2 more variables: TaxonomicSpecies <chr>, mean_Isotope15N <dbl>
```