Data Wrangling & Species Richness

Jeff Oliver

20 July, 2020

We collect data from a lot of sources, including field observations. Sometimes, though, the way will collect and understand data as humans doesn't work well with the way computers operate. Here we will work through a real data set to use best practices for data organization to analyze species richness and diversity.

Learning objectives

1. Demonstrate how to organize data in 'tidy' format
2. Develop quality control processes for data
3. Visualize measures of species richness and species diversity

---

Getting started

> Happy families are all alike; every unhappy family is unhappy in its own way - Leo Tolstoy

Since we are working in RStudio, start by making a new project from the File menu (File > New Project…). When prompted, select New Directory, then New Project. Name the project sweetwater-analysis. When the project opens, create a folder to hold the data by typing `dir.create(data)` in the Console.

Tidy data

After Wickham 2014, "tidy" data is defined by:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

The survey data we are working with is available as an Excel file for download at https://tinyurl.com/sweetwater-data. (if you want to skip the data wrangling, and work with the "tidy" data, you can download them here). Make sure to move the file to the 'data' folder you created.

Open the file in Excel. What can be done to make these data "tidy"?

First, the data are effectively broken in two, with the most recent observations separate from observations from preceding days. So in some cases, a row has more than one observation. To fix this, we move the cells from the preceding observations to rows after the most recent observations.

We also want to:

- Delete any columns without data
- Delete any rows without data
- Add a column title for Date
- Make sure data for preceding observations lines up with column title (currently the Count column has species names and the Species column has count data)

Let's save this file as a CSV file so we can read it into R. We don't want to overwrite the original data file, so we'll call our file sweetwater-data-clean.csv.

Data wrangling

Now that the data are in CSV format, we can read it into R and take a look. We want to keep a record of all our work, so we are going to type commands into a script file and save the script. Create a new script via the File menu (File > New File > R Script). As we did before, we add a few lines of comments at the top of our script with some relevant information.

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")
```

We can look at the first few rows with the `head` function:

```
head(bird_data)
```

```
##          Date Site Count              Species
## 1 October 4th    1     1         Mallard Duck
## 2               NA     1 Red-winged blackbird
## 3               NA     1      Gila woodpecker
## 4               NA     1                Finch
## 5               NA     1        Mourning Dove
## 6               NA     1          Phainopepla
```

And we see that in rows two through 6 there aren't any values for the Date and Site columns. This is a prime example of the difference between how a human interprets data and how a computer interprets the data. Let's open up our CSV file in Excel again. As humans, we can infer that the value for "Date" should be October 4th all the way to row 35. But the computer is stupid. The computer looks at row 3, sees no date, and infers that the date is *missing*. We need to explicitly tell the computer what the value for date is. Before we do that, we have the opportunity to go ahead and also change the date to the standard ISO date format of "YYYY-MM-DD". In this case, October 4th becomes 2019-10-04.

Go ahead and update all the dates and fill in values appropriately. You should also note that for observations prior to October 10th, there is an offset that (1) needs to be fixed and (2) prompts deletion of now empty rows.

Also, for the 2019-10-04 observations, do the same thing with data in the Site column - that is, fill in values where appropriate. Save the file as a CSV (you can use the same filename, sweetwater-data-clean.csv) and read it into R.

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")
```

Looking at the updated data, we see those first six rows no longer have missing data.

```
head(bird_data)
```

```
##         Date Site Count              Species
## 1 2019-10-04    1     1         Mallard Duck
## 2 2019-10-04    1     1 Red-winged blackbird
## 3 2019-10-04    1     1      Gila woodpecker
```

```
## 4 2019-10-04    1     1               Finch
## 5 2019-10-04    1     1        Mourning Dove
## 6 2019-10-04    1     1          Phainopepla
```

We can also use the `summary` function to get an idea of the data set overall:

```
summary(bird_data)
```

```
##      Date                Site           Count          Species
##  Length:134         Min.   :1.000   Min.   : 1.000   Length:134
##  Class :character   1st Qu.:2.000   1st Qu.: 1.000   Class :character
##  Mode  :character   Median :3.000   Median : 1.000   Mode  :character
##                     Mean   :2.941   Mean   : 4.164
##                     3rd Qu.:4.000   3rd Qu.: 3.000
##                     Max.   :6.000   Max.   :80.000
##                     NA's   :100
```

There are still missing values in the Site column, but that's OK - those are observations from preceding days, which don't have any site data.

Note also though that it is treating "Date" as categorical data when it really should be treated as, well, a date. So we can change the data in the Date column from categories to dates with the `as.Date` function:

```
bird_data$Date <- as.Date(bird_data$Date)
```

Species richness

OK, so now let's actually look at species richness. We are going to look at each date separately, and count the total number of species observed. To do this, we are going to use an additional package called tidyverse; we will need to install the package and load it into memory before we try to use it. First, we check to see if the package is installed by looking at the Packages tab in the lower-right window. If tidyverse *is* installed, we don't need to install it again. If tidyverse is not listed, you can install it through the R console with the command:

```
install.packages("tidyverse")
```

Now that the package is installed, we also have to tell R to load the package into memory, so we add a call to `library` to our script. We often add all `library` commands to the top of our script.

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

library(tidyverse)
bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")

# Convert the Date column to date data type
bird_data$Date <- as.Date(bird_data$Date)
```

You might see some read messages when you load tidyverse. As long as you don't see the word "Error" everything should be fine.

Now we can use some tidyverse functions to calculate the total number of species seen on each day. We use the `group_by` function to first create a different pile of data for each day, then count the number of rows using the `n` function.

```r
# Calculate species richness per day
richness <- bird_data %>%
  group_by(Date) %>%
  summarize(Richness = n())
```

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

Let's look at the output by typing the name of our variable into the console:

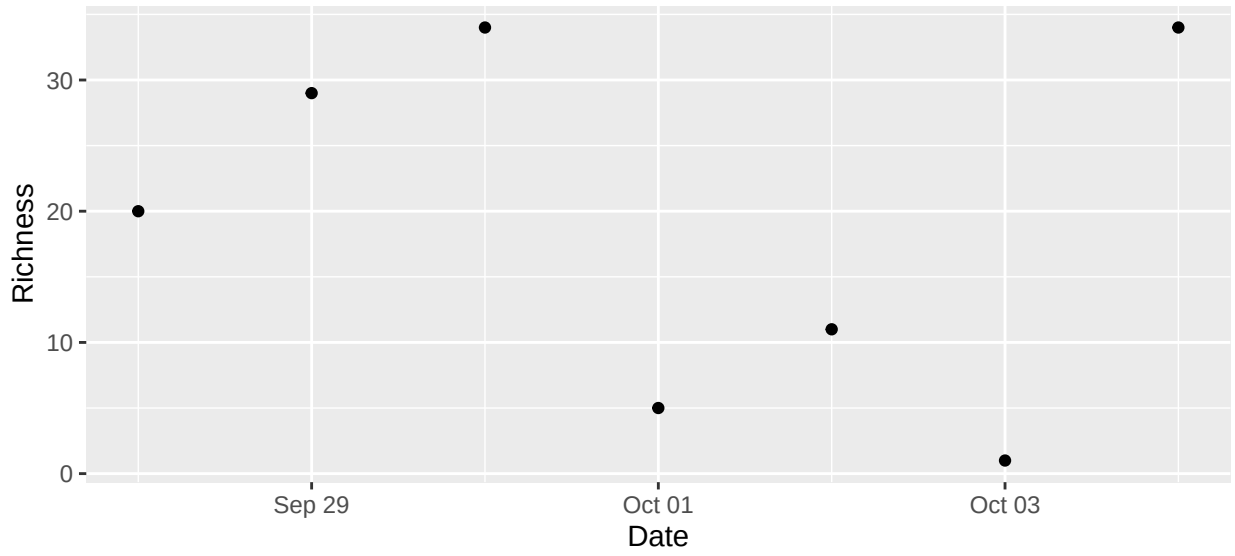
```r
richness
```

```
## # A tibble: 7 x 2
##   Date       Richness
##   <date>        <int>
## 1 2019-09-28       20
## 2 2019-09-29       29
## 3 2019-09-30       34
## 4 2019-10-01        5
## 5 2019-10-02       11
## 6 2019-10-03        1
## 7 2019-10-04       34
```

Pretty cool. We can even plot species richness for each day, too. We are going to use the ggplot2 package again, which comes with tidyverse.

To plot richness for each day,

```r
# Plot daily species richness
richness_plot <- ggplot(data = richness,
                        mapping = aes(x = Date,
                                      y = Richness)) +
  geom_point()
print(richness_plot)
```



Hey, a plot!

Let's take a moment though to think about the data that went into making these plots. In fact, let's look at the first six rows again:

```
head(bird_data)
```

```
##         Date Site Count              Species
## 1 2019-10-04    1     1         Mallard Duck
## 2 2019-10-04    1     1 Red-winged blackbird
## 3 2019-10-04    1     1      Gila woodpecker
## 4 2019-10-04    1     1                Finch
## 5 2019-10-04    1     1        Mourning Dove
## 6 2019-10-04    1     1          Phainopepla
```

Hmmm…row four has "Finch", which isn't a species. At Sweetwater Wetlands, House Finches and Lesser Goldfinches are common, and sometimes other rare finches show up. Another example is on row 17. We can take a look at this part of the data by specifying which rows to show. Here we ask for R to print out rows 15 through 20:

```
bird_data[15:20, ]
```

```
##          Date Site Count                              Species
## 15 2019-10-04    3     1                        Gambils quail
## 16 2019-10-04    3     1                          Phainopepla
## 17 2019-10-04    3     2                             Finch sp
## 18 2019-10-04    3     1                       Northern harrier
## 19 2019-10-04    3     1                     Unidentified raptor
## 20 2019-10-04    3     3 Humming Bird: Rubythroated, Black chin
```

In addition to "Finch sp" we all so see two more examples of observations not identified to species. Because we don't actually know what species these were, we will need to exclude them from our analyses.

Computers are stupid

Or, why quality control matters

We do not actually want to delete these records from our data, but rather we need a means of indicating whether or not they are identified to the species level. This should probably be done in the sweetwater-data-clean.csv file we've been working with. So open that file in Excel and add another column, immediately to the right of Species. Call this one "ID_to_species" and fill in values of 1 for those identified to species (e.g. Gila woodpecker would be scored as a 1) and values of 0 for those not identified to species (e.g. Finch sp and duck sp. would be scored as a 0). Save the file with the same name as before (sweetwater-data-clean.csv).

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")

# Convert the Date column to date data type
bird_data$Date <- as.Date(bird_data$Date)
```

Now we can update our richness calculation to only include observations that were identified to species. We do this by adding a filter to the process. Using the code we had before, before calling `group_by`, we use the `filter` function to only include those observations that have a 1 in the ID_to_species column.

```
# Calculate species richness per day
richness <- bird_data %>%
  filter(ID_to_species == 1) %>%
```

```
  group_by(Date) %>%
  summarize(Richness = n())
```

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

We can compare the old calculations for richness

```
## # A tibble: 7 x 2
##   Date       Richness
##   <date>        <int>
## 1 2019-09-28       20
## 2 2019-09-29       29
## 3 2019-09-30       34
## 4 2019-10-01        5
## 5 2019-10-02       11
## 6 2019-10-03        1
## 7 2019-10-04       34
```

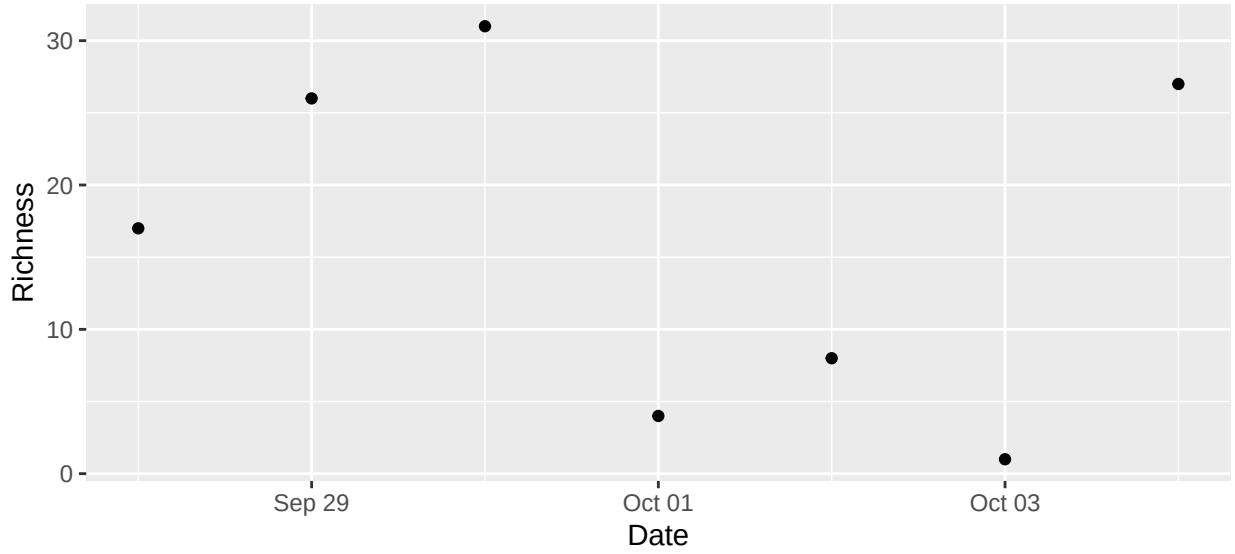
to the new ones:

```
richness
```

```
## # A tibble: 7 x 2
##   Date       Richness
##   <date>        <int>
## 1 2019-09-28       17
## 2 2019-09-29       26
## 3 2019-09-30       31
## 4 2019-10-01        4
## 5 2019-10-02        8
## 6 2019-10-03        1
## 7 2019-10-04       27
```

And note the numbers have dropped a little when we exclude those observations. Now when we run the code for plotting, we see the results changed, too:

```
richness_plot <- ggplot(data = richness,
                        mapping = aes(x = Date,
                                      y = Richness)) +
  geom_point()
print(richness_plot)
```



Let's take a quick look at the species that were observed on October 4, just to double check. We can do that in the console using the pipe (%>%) and the `filter` command (you'll see more than ten rows):

```
bird_data %>% filter(Date == "2019-10-04")
```

```
##          Date Site Count              Species ID_to_species
## 1  2019-10-04    1     1         Mallard Duck             1
## 2  2019-10-04    1     1 Red-winged blackbird             1
## 3  2019-10-04    1     1      Gila woodpecker             1
## 4  2019-10-04    1     1               Finch             0
## 5  2019-10-04    1     1        Mourning Dove             1
## 6  2019-10-04    1     1          Phainopepla             1
## 7  2019-10-04    2     5         Song Sparrow             1
## 8  2019-10-04    2     2      Gila Woodpecker             1
## 9  2019-10-04    2     1         Mallard Duck             1
## 10 2019-10-04    2     1       Lucy's Warbler             1
```

We should see 34 rows. This lines up with our initial plot, that showed 34 species. But look a little closer, notably at rows 1 and 9. Both rows are observations of Mallard Ducks. So this species actually got counted (at least) twice. Because multiple sites were surveyed, We can't just count the number of rows in this data to get the number of species. We should actually do a little more data wrangling to get an accurate count of species for the October 4 sampling. What we are going to do is to create a new data object we'll call `bird_data_sums` that sums up the values for each species on each day.

```
# Sum species counts for each day
bird_data_sums <- bird_data %>%
  filter(ID_to_species == 1) %>%
  group_by(Date, Species) %>%
  summarise(Count = sum(Count))
```

```
## `summarise()` regrouping output by 'Date' (override with `.groups` argument)
```

Now we can again look at the observations for October 4:

```
bird_data_sums %>% filter(Date == "2019-10-04")
```

```
## # A tibble: 21 x 3
## # Groups:   Date [1]
##    Date       Species              Count
```

```
##    <date>     <chr>                <int>
##  1 2019-10-04 Coopers' hawk            1
##  2 2019-10-04 Curve-bill Thrasher      3
##  3 2019-10-04 Flycatcher vermilion     1
##  4 2019-10-04 Gambils quail            3
##  5 2019-10-04 Gila woodpecker          1
##  6 2019-10-04 Gila Woodpecker          4
##  7 2019-10-04 house finch              3
##  8 2019-10-04 Lucy's Warbler           1
##  9 2019-10-04 mallard                  1
## 10 2019-10-04 Mallard Duck             2
## # ... with 11 more rows
```

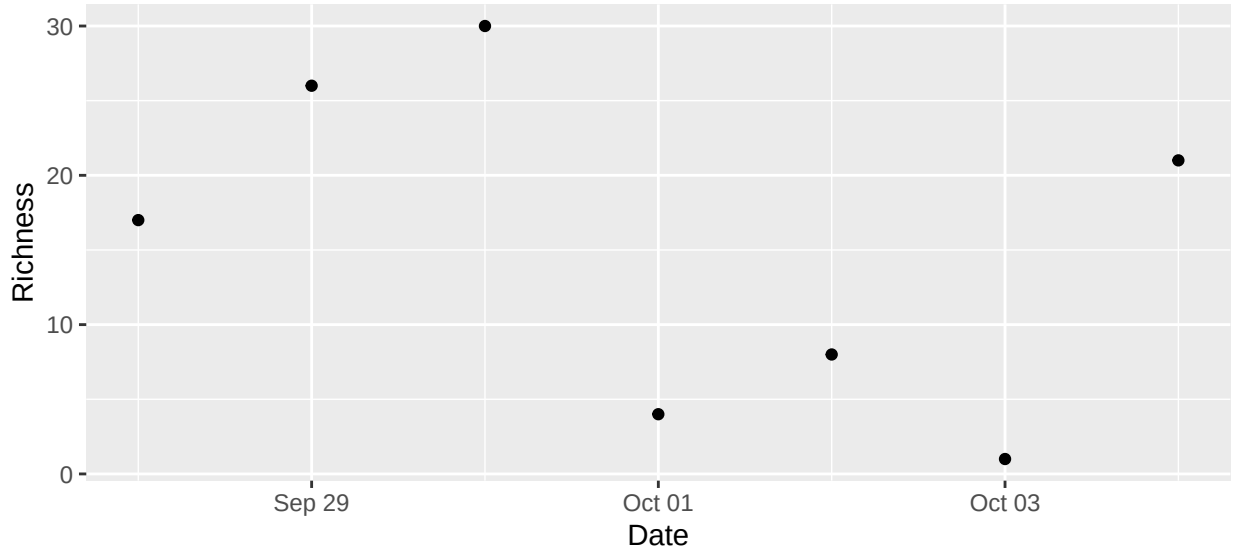
Now this doesn't show us all the data, but looking at the end of the output, we see that R is showing us 10 rows of data, with 11 more rows. So we have gone from 34 species, down to 21 species for October 4.

We can update our code to calculate richness and re-create our plot. Remember to replace `bird_data` with `bird_data_sums` in the first line of the code. We can also remove the `filter` function, because we already filtered out observations that were not identified to species when we created `bird_data_sums`.

```
# Calculate species richness per day
richness <- bird_data_sums %>%
  group_by(Date) %>%
  summarize(Richness = n())
```

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

```
# Plot daily species richness
richness_plot <- ggplot(data = richness,
                        mapping = aes(x = Date,
                                      y = Richness)) +
  geom_point()
print(richness_plot)
```



The only thing to really change in our plot is for October 4. OK, some of you might have noticed this already, but let's look again at those October 4 data:

```
bird_data_sums %>% filter(Date == "2019-10-04")
```

```
## # A tibble: 21 x 3
## # Groups:   Date [1]
##    Date       Species              Count
##    <date>     <chr>                <int>
##  1 2019-10-04 Coopers' hawk            1
##  2 2019-10-04 Curve-bill Thrasher      3
##  3 2019-10-04 Flycatcher vermilion     1
##  4 2019-10-04 Gambils quail            3
##  5 2019-10-04 Gila woodpecker          1
##  6 2019-10-04 Gila Woodpecker          4
##  7 2019-10-04 house finch              3
##  8 2019-10-04 Lucy's Warbler           1
##  9 2019-10-04 mallard                  1
## 10 2019-10-04 Mallard Duck             2
## # ... with 11 more rows
```

So, there's another issue to address. Look at rows 9 and 10. We have a species that is represented twice. Why? Here is a great example of computers only being able to do what you tell them to. We told R to count the total number of individuals for each species. Unfortunately, R sees "mallard" and "Mallard Duck" as two different species.

What do we do?

At this point, the best thing is probably to go into the Excel file and change the species names to be consistent. Now so far, we haven't actually *changed* any of the data - we added a column and we filled in values where appropriate, but we did not change anything. When changing data, you always want to be careful. At this point, we should be keeping track of changes we made to the data. We are going to do this in a separate file, with notes about what we did. So start by creating a new text file (File > New File > Text File) and adding:

```
README for sweetwater species data
2019-11-01
+ Created CSV from original Excel data file
+ Changed to ISO date format
+ Standardized spelling of species' names
```

And save this file as "README.md".

Now we actually have to do the standardization of the species names, so open up the sweetwater-data-clean.csv file in Excel. What's the best way of going about changing names? We could search for non-standard spellings like "Gambils quail" but that assumes we know all the alternative spellings. What might help our cause is to sort by the Species column so things like "mallard" and "Mallard Duck show up next to one another. **Note**: this sorting is going to rearrange the order of the rows, so we should *only* do this if the order of the rows does not matter. Remember the original data? Where the date was only shown in the first row? If we had sorted those data it would have been *bad*. Why? Because in that case, order *did* matter. But by using tidy data formats, we can re-order rows without losing any information for a single observation.

So go ahead and change names so they are consistent. This includes

- consistent spelling (Gambel's vs. Gambils),
- consistent capitalization (Gila Woodpecker vs. Gila woodpecker), and
- constent use of punctuation (Red-winged Blackbird vs. Redwing black bird and Cooper's Hawk vs. Coopers' Hawk).

Save the file to sweetwater-data-clean.csv and go back into R.

Re-run the line at the top of our script with the `read.csv` command:

```
bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")
bird_data$Date <- as.Date(bird_data$Date)
```

Re-run the line counting the number of each species per day:

```
# Sum species counts for each day
bird_data_sums <- bird_data %>%
  filter(ID_to_species == 1) %>%
  group_by(Date, Species) %>%
  summarise(Count = sum(Count))
```

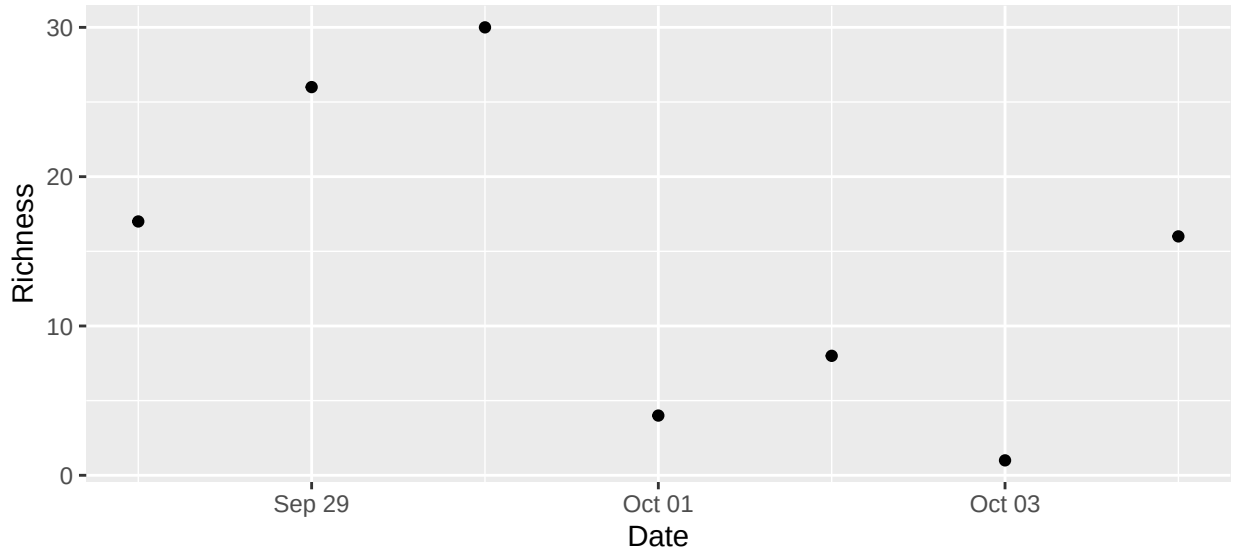
```
## `summarise()` regrouping output by 'Date' (override with `.groups` argument)
```

And calculate richness for each day and draw the plot

```
# Calculate species richness per day
richness <- bird_data_sums %>%
  group_by(Date) %>%
  summarize(Richness = n())
```

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

```
# Plot daily species richness
richness_plot <- ggplot(data = richness,
                        mapping = aes(x = Date,
                                      y = Richness)) +
  geom_point()
print(richness_plot)
```



Whew!

So at this point, let's think about what we've done in R. We really haven't had to do too much, we wrote code to read in the file, count the total number of each species seen each day, calculate species richness, and plot species richness. Most of our time was really spent wrangling the data into a format that could be used to analyze with a computer. This isn't unexpected. Data wrangling can take up as much as 80% of a data scientist's time (See Furche et al. 2016. doi 10.5441/002/edbt.2016.44).

Species diversity

Now we are ready for species diversity! We are going to use Shannon's index (H) of species diversity:

$$H = -\sum_{i=1}^{s} p_i \ln p_i$$

which is the sum over s species, where p_i is the proportion individuals of the i^th species (n_i) out of the total number of inviduals (N), or:

$$p_i = \frac{n_i}{N}$$

To calculate this measure of diversity, we are going to use the vegan package, which does *not* come with tidyverse, so we'll have to install it with:

```
install.packages("vegan")
```

Add the `library` command to the top of our script:

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

library(tidyverse)
library(vegan)
bird_data <- read.csv(file = "data/sweetwater-data-clean.csv")

# Convert the Date column to date
bird_data$Date <- as.Date(bird_data$Date)
```

We are going to use functions that come with the vegan package to help with these calculations. In order to use those functions, we need to convert the `bird_data_sums` from the "long" format it is now to "wide" format. That is, we need to change from:

Date	Species	Count
2019-10-04	Abert's Towhee	1
2019-10-04	Gila Woodpecker	2
2019-10-03	Abert's Towhee	1
2019-10-03	Greater Roadrunner	1

To:

Date	Abert's Towhee	Gila Woodpecker	Greater Roadrunner
2019-10-04	1	2	0
2019-10-03	1	0	1

Where

- each date corresponds to a single row
- each species corresponds to a single column

We can do this with the `pivot_wider` function that comes with tidyverse. Here we tell R to use the values

in the Species column as new column names, and the numbers in the Count column to use as values.

```
bird_data_wide <- bird_data_sums %>%
  pivot_wider(names_from = Species,
              values_from = Count)
```

Let's look at the first five columns of the "wide" formatted data:

```
bird_data_wide[, 1:5]
```

```
## # A tibble: 7 x 5
## # Groups:   Date [7]
##   Date       `Bell's Vireo` `Black-throated Gray Wa~ `Blue-gray Gnatcat~ `Brewer's Blackbi~
##   <date>              <int>                    <int>               <int>              <int>
## 1 2019-09-28              1                        1                   1                 36
## 2 2019-09-29             NA                       NA                  NA                 NA
## 3 2019-09-30             NA                       NA                  NA                 NA
## 4 2019-10-01             NA                       NA                  NA                 NA
## 5 2019-10-02             NA                       NA                  NA                 NA
## 6 2019-10-03             NA                       NA                  NA                 NA
## 7 2019-10-04             NA                       NA                  NA                 NA
```

For the diversity calculations, it only works on count data, and our data include one column of dates. We create a new variable, `species_counts` which has all the data except the first column (the Date column), then we calculate species richness and look at the results by running the name of the variable (`bird_diversity`):

```
species_counts <- bird_data_wide[, -1]
bird_diversity <- diversity(x = species_counts)
bird_diversity
```

```
## [1] NA NA NA NA NA NA NA
```

All the values are `NA` because the input data (`bird_data_wide`) had `NA` values for every Date/Species combination with zero observations. In this case, we need to replace all those `NA` values with 0.

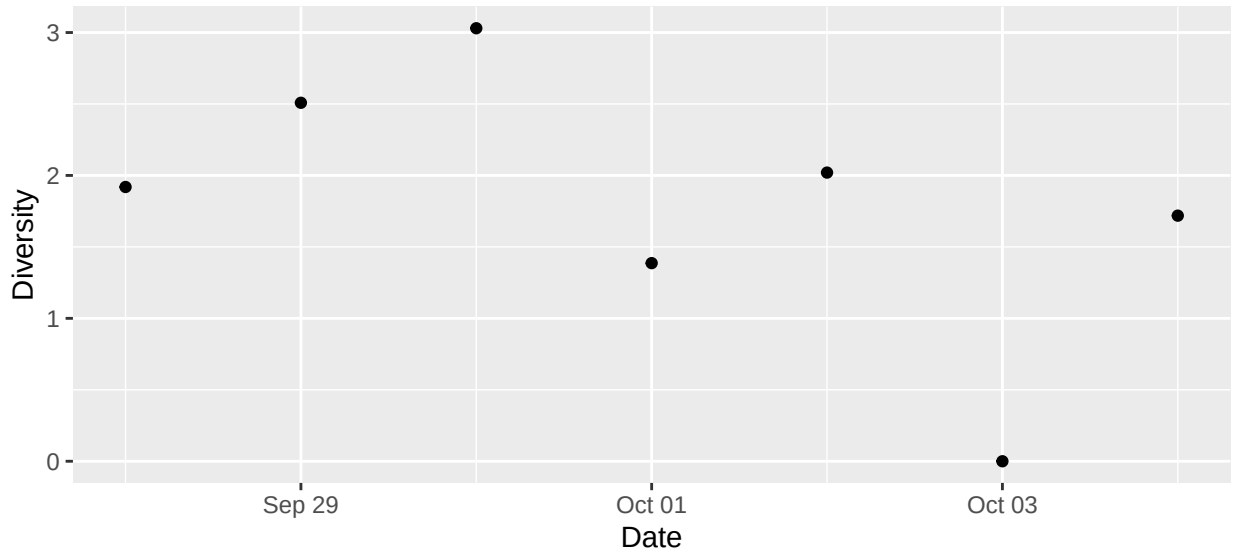
```
bird_data_wide[is.na(bird_data_wide)] <- 0
```

We re-run the diversity calculations after converting those `NA` values to zero. We then overwrite `bird_diversity` with a data frame that contains the Date and the results of the diversity calculations:

```
# Calculate species diversity on count data
species_counts <- bird_data_wide[, -1]
bird_diversity <- diversity(x = species_counts)

# Add date
bird_diversity <- data.frame(Date = bird_data_wide$Date,
                             Diversity = bird_diversity)
```

And plot diversity:

```
# Plot species diversity of each date
diversity_plot <- ggplot(data = bird_diversity,
                         mapping = aes(x = Date, y = Diversity)) +
  geom_point()
print(diversity_plot)
```



Starting clean

Setup

If you want to skip all the data wrangling and start with data that are already in tidy format, download the data here.

Be sure you set up a new project as described in Getting started, above.

Move the file you downloaded to the 'data' folder.

In order to save our work, we will use an R script, which is a text file where we place all our R commands. Create a new script through the file menu (File > New File > R Script). We start our script with a header of comments:

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01
```

We're going to use some R code from packages that do not come with base R. Run these two lines in the console:

```
install.packages("tidyverse")
install.packages("vegan")
```

We also need to tell R in our script that we want to use those packages; we load them into memory with the `library` command. Generally, we try to add all the calls to `library` to the beginning of our code, so if someone opens the script, they can immediately see which additional packages are required.

```
# Analyze sweetwater bird species richness
# Jeff Oliver
# jcoliver@email.arizona.edu
# 2019-11-01

# Load additional packages
library(tidyverse)
```

```
library(vegan)

# Read in data
bird_data <- read.csv(file = "data/sweetwater-data-clean04.csv")

# Convert the Date column to date data type
bird_data$Date <- as.Date(bird_data$Date)
```

The last line reads the data into R. You can look at these data by running `head(bird_data)` in the Console.

Wrangle the data

Next we need to sum the counts for each species for each day:

```
# Sum species counts for each day
bird_data_sums <- bird_data %>%
  filter(ID_to_species == 1) %>%
  group_by(Date, Species) %>%
  summarise(Count = sum(Count))
```

```
## `summarise()` regrouping output by 'Date' (override with `.groups` argument)
```

With the daily totals for each species, now we can calculate species richness (here we are measuring daily richness as the total number of species each day).

Richness

```
# Calculate species richness per day
richness <- bird_data_sums %>%
  group_by(Date) %>%
  summarize(Richness = n())
```

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

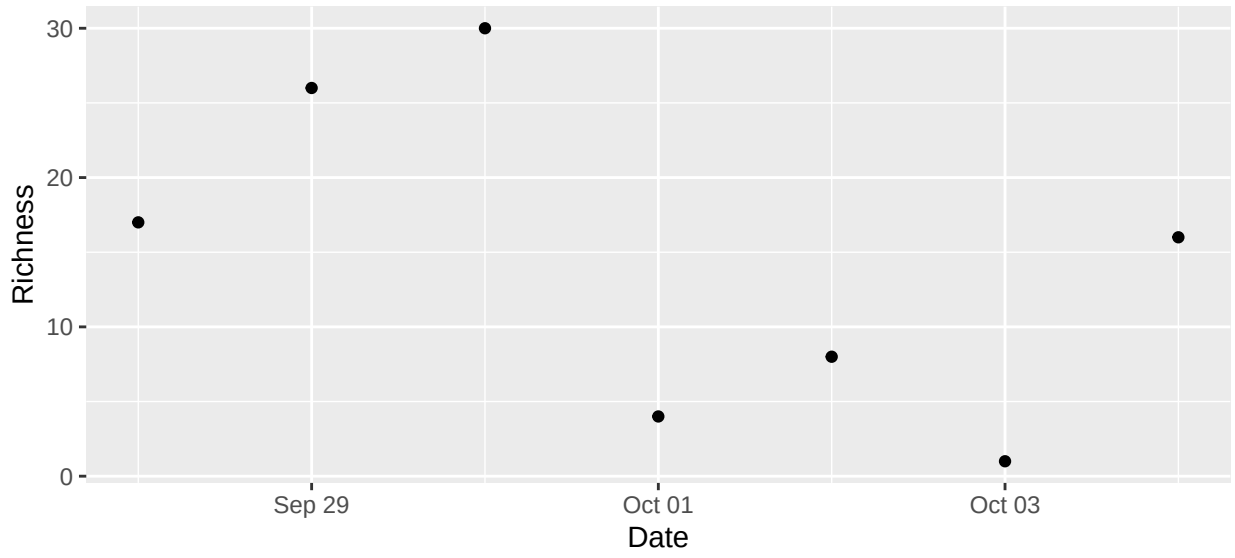
We can investigate the richness data by typing the name of the variable into the console:

```
> richness
```

```
## # A tibble: 7 x 2
##   Date       Richness
##   <date>        <int>
## 1 2019-09-28       17
## 2 2019-09-29       26
## 3 2019-09-30       30
## 4 2019-10-01        4
## 5 2019-10-02        8
## 6 2019-10-03        1
## 7 2019-10-04       16
```

Now that we have species richness calculated for each day, we can use the ggplot2 package, which comes with tidyverse, to plot the data:

```
# Plot daily species richness
richness_plot <- ggplot(data = richness,
                        mapping = aes(x = Date,
                                      y = Richness)) +
  geom_point()
print(richness_plot)
```



Diversity

For species diversity, we are going to use, H, Shannon's index of species diversity:

$$H = -\sum_{i=1}^{s} p_i \ln p_i$$

which is the sum over s species, where p_i is the proportion individuals of the i^th species (n_i) out of the total number of inviduals (N), or:

$$p_i = \frac{n_i}{N}$$

We are going to use functions that come with the vegan package to help with these calculations. In order to use those functions, we need to convert the `bird_data_sums` from the "long" format it is now to "wide" format. That is, we need to change from:

Date	Species	Count
2019-10-04	Abert's Towhee	1
2019-10-04	Gila Woodpecker	2
2019-10-03	Abert's Towhee	1
2019-10-03	Greater Roadrunner	1

To:

Date	Abert's Towhee	Gila Woodpecker	Greater Roadrunner
2019-10-04	1	2	0
2019-10-03	1	0	1

Where

- each date corresponds to a single row

- each species corresponds to a single column

We use the `pivot_wider` function of tidyverse to accomplish this. We also need to replace NA values produced in this process to zeros:

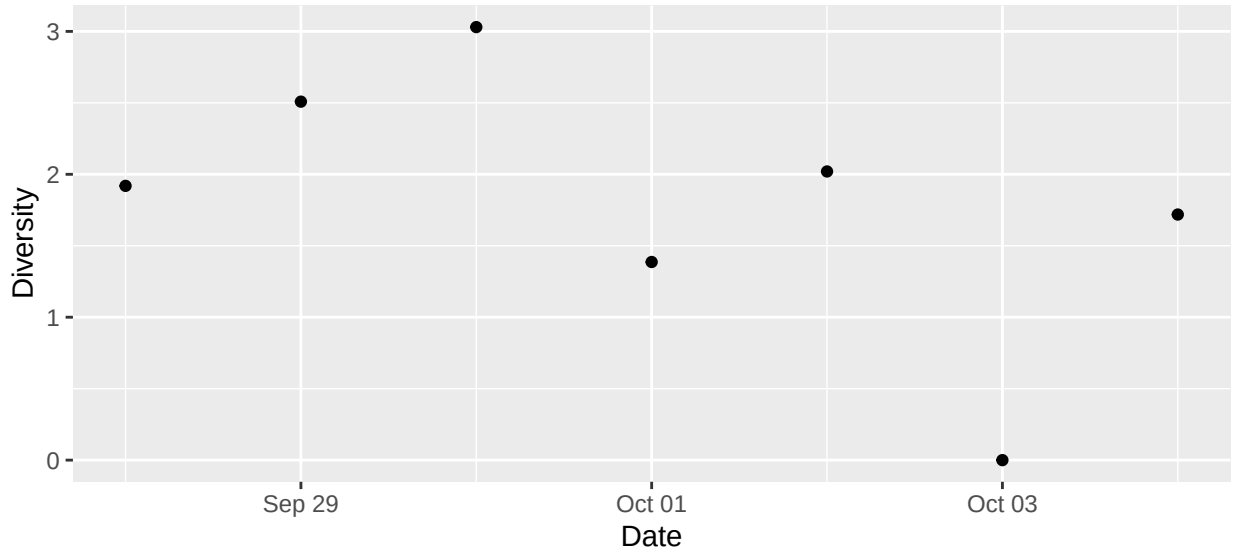
```
# Convert to wide format for species diversity calculation
bird_data_wide <- bird_data_sums %>%
  pivot_wider(names_from = Species,
              values_from = Count)

# Replace NA values with zeros
bird_data_wide[is.na(bird_data_wide)] <- 0
```

The diversity calculation only works on count data, and our data include one column of dates. We create a new variable, `species_counts` which has all the data except the first column (the Date column), then we calculate species richness and look at the results by running the name of the variable (`bird_diversity`):

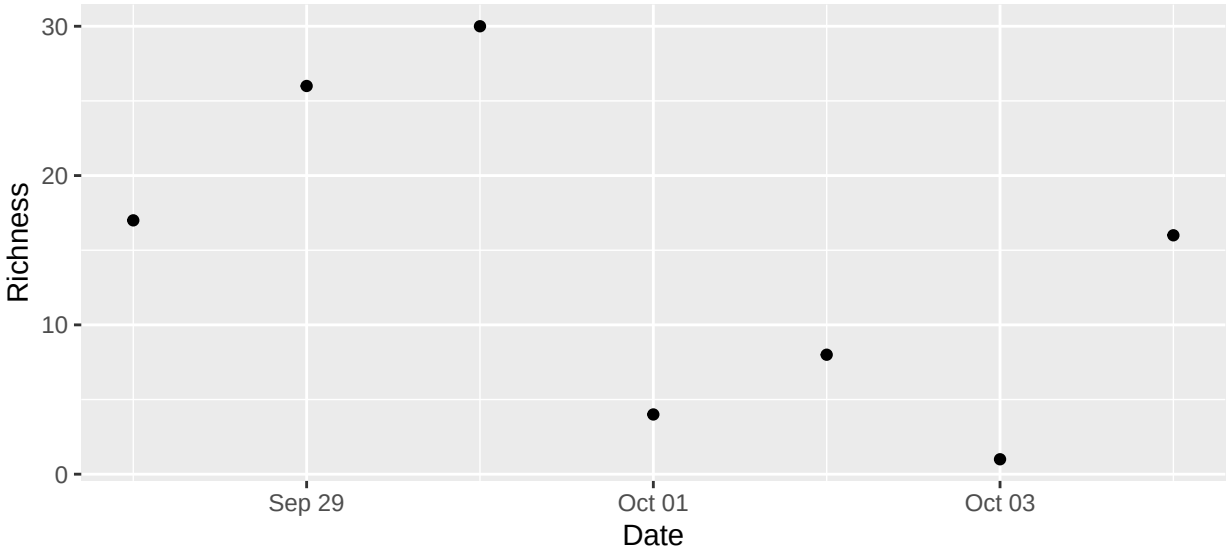
```
# Calculate species diversity on count data
species_counts <- bird_data_wide[, -1]
bird_diversity <- diversity(x = species_counts)

# Add date
bird_diversity <- data.frame(Date = bird_data_wide$Date,
                         Diversity = bird_diversity)
```

Now that we have species diversity, we can plot it using commands similar to those we used for species richness:

```
# Plot species diversity of each date
diversity_plot <- ggplot(data = bird_diversity,
                         mapping = aes(x = Date, y = Diversity)) +
  geom_point()
print(diversity_plot)
```



Recall the plot for richness:



Question: Why is species richness of October 4 higher than richness of October 2, but species diversity is *lower* on October 4 than on October 2?

Additional resources

- Wickham, H. 2014. Tidy data. *The Journal of Statistical Software* **59** http://www.jstatsoft.org/v59/i10/
- A nice introduction to calculating measures of species diversity
- A PDF version of this lesson

Back to learn-r main page

Questions? e-mail me at jcoliver@email.arizona.edu.