Workshop

Exercise # 1: Data manipulation

Goal

The goal of this exercise is to gain experience manipulating data typically encountered in fisheries research. Specifically, reading data into **R**, "cleaning" raw data into a more **R** user-friendly format, summarizing data, and create a table of fish catch statistics. We will use pipes (`%>%`) and the package `dplyr` for most of the data processing.

Although it is fine to have all the workshop **R** packages installed and loaded into the current session, the specific R packages this exercise uses are:

```
library(dplyr) # data manipulation
library(tidyverse) # data manipulation
library(lubridate) # work with dates
library(kableExtra) # make tables
```

The dataset

The data consist of nighttime boat electrofishing data from Mauch Chunk Lake, PA (Figure 1). This lake is currently managed under a Big Bass regulation and has been sampled periodically since 1981 up until 2022. The number of lake surveys varies among years and there have been a total of 15 species caught over the years, although not all were caught in each survey and in each year. The raw data are well organized, however, the data contains several issues that need to be resolved in order to work easily with them in **R**.

Read in the data

1. Open **R** from the `.proj` file associated with this workshop (`R_Workshop_2024.Rproj`), create a new R script, and save it (give it name of your choice) to the folder called *Ex_1* located in the folder *07_Exercises*. To create a new script, follow File –> New File –> R Script.

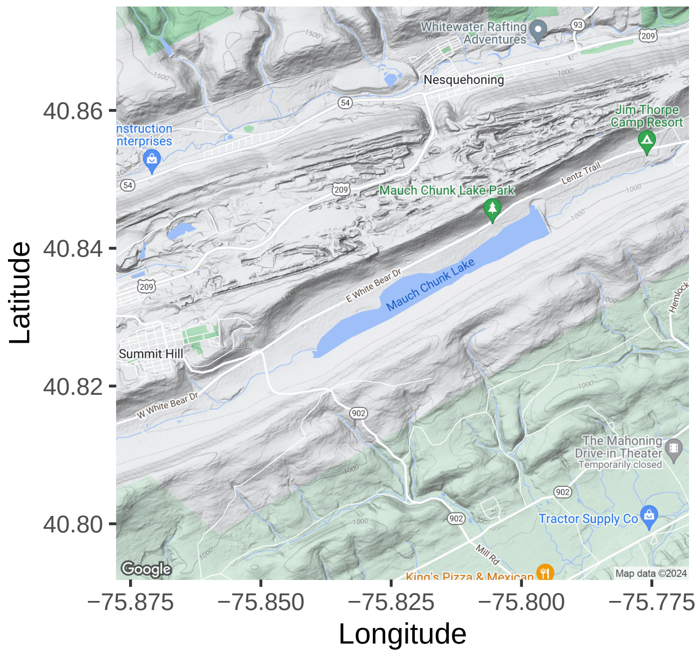

Figure 1: Mauch Chunk Lake, PA

2. Load the required R libraries listed above.
3. We will use the `read_csv` function to read in the data (this function – found in the `readr` package that is loaded with `tidyverse` – is faster at reading in data and has other advantages over the `read.csv` function). Note that the data are in a subdirectory located in the *02_Data* folder. We need to "tell" **R** where to find the data to import. You can use the following code:

```
# Read in data
# The "./../" syntax is backing out 2 directories from our current R script file
# location so we can navigate into the 02_Data folder where our data are located
dat <- read_csv("./../02_Data/Mauch_lake/Mauch_lake_surveys_NBE.csv")
```

4. Once the data are read into R, you can look at the structure of the data frame using the command:

```
str(dat)
```

Although this information is given when using `str`, here is the dimension (# rows and columns) of this data frame.

```
dim(dat)
```

```
[1] 1606   18
```

So, there are 1606 rows and 18 columns.

The column names are:

```r
names(dat)
```

```
 [1] "Water Name"                 "Water Section ID"
 [3] "Unique Water Site ID"       "Survey Site Lat DD"
 [5] "Survey Site Lon DD"         "Water Site Survey ID"
 [7] "Site Date"                  "Year"
 [9] "Month"                      "Fish Species Name"
[11] "Group Size Fish Length"     "szClass"
[13] "Number Caught"              "Survey Purpose"
[15] "Water Site Comment"         "Water Site Survey Comment"
[17] "Description"                "Effort Hours"
```

5. There are several things to note when looking at the structure of the data that we will address in the data cleanup.

    a. Many of the column names have spaces between words, e.g., `Water Name`. Having spaces in column headings is not ideal when working in **R**. It introduces the need for additional syntax that is unnecessarily cumbersome. Note: there is a package called `janitor` that contains useful functions for cleaning column names. However, we will do it 'by hand' here.

    b. There are some columns we will not use or need for this work, so we can remove them to create a "less cluttered" dataset.

    c. There is a column called `Site Date` that **R** has read in as a character (denoted chr in the output). The date includes the month, day, and year and a time stamp (hr:min). We want **R** to recognize dates as a Date and not a character string, since dates are treated differently in many **R** functions.

Data cleanup

In addition to the items listed in #5 above, there was also a note in the dataset that there are a few duplicate surveys (`Water Site Survey ID` is the column containining the unique survey ID) – and the duplicates will need to be removed. The duplicate survey IDs that need to be excluded are `Water Site Survey ID` *35947* and *35971*.

Here is the workflow we will follow to *clean* the data (this is one of several that could be used and we are breaking this up for teaching purposes - it could be done more concisely):

1. Remove unwanted columns
2. Rename some columns and, whild doing so, remove the space between words
3. Remove duplicate surveys
4. Create new `Date` column
5. Remove old date column

6. Regardless of how columns were renamed, make all column headings lower case for ease of coding

Tips: Useful R functions

- We will use pipes (`%>%`) to accomplish steps 1 – 5.
- When there are column names that have spaces – like in our dataset – the column names need to be referenced using single quotes, such as `` `Water Name` ``. This is done to indicate that the two words are part of the same name and the space is not indicating two different column names. RStudio usually adds these single quotes automatically when you are typing the name of the column. Some **R** users might not mind this syntax, but I rather not have the extra quotes - so we will remove spaces in column names.
- The `select()` function returns a subset of columns You can select to retain specific columns as in `dat %>% select(site_id, field_notes)` - which will retain the columns named `site_id` and `field_notes`. We can also exclude columns through the use of a minus sign (`-`) placed before the column name(s) to exclude. For example, `dat %>% select(-c(site_id, field_notes))` will select all columns in the data frame dat *except* columns called `site_id` and `field_notes`. i.e., we are excluding the columns named `site_id` and `field_notes`.
- The `rename()` function is used to rename columns and has the following sytax: `rename("new_column_name" = "old_column_name")` (works with and without quotes around column names). Multiple columns can be renamed at the same time by placing a comma between columns to be renamed.
- The `rename_with()` function renames columns using a function. For example, it is often convenient to have all column names in lower case (this makes it easier to type and reference columns without worrying about letter case). This can be accomplished by `rename_with(tolower)`. The `tolower` function puts all text "to lower" case.
- The `filter()` function subsets rows in a data frame by testing against a conditional statement. For example, if we wanted to retain only rows (observations of fish) with total lengths > 200 mm, we could write `filter(fish_length > 200)`
  - **Commonly used conditional statements:** `<` (less than); `>` (greater than); `==` (equal to); `!=` (not equal to); `<=` (less than or equal to); `>=` (greater than or equal to); `&` (and); `|` (or); `is.na()` (is NA [missing]); `!is.na()` (is not NA [missing])
- The `mutate()` function adds new columns (or overrides the values of an existing column) thereby "mutating" the contents and dimensions of the data frame. For example, if I wanted to create a new column (I will call it `log_length`) that was the natural log of a column `fish_length` I could write `mutate(log_length = fish_length)`
- The `group_by()` function create sub-groups based on specified column(s), and allows us to run subsequent functions (chained together using pipes) on the sub-groups. For example, if we wanted to calculate the total catch of fish by year and species we could write code shown in the R chunk below. Notice the use of the `ungroup()` function after

we calculate our total catch by year and species. When running chains it is good practice to ungroup the data after the `group_by()` operations are run.

- The `arrange()` function sorts (orders) the data frame according to values in one or more columns from lowest –> highest (by default). For example, `arrange(year, species)` would order the data frame according to year and then species.
- The `summarize()` function creates summary statistics from a data frame. See R chunk example below.

```
# group_by() illustration

tot_catch <- dat %>%
  group_by(year, species) %>% # group data
  mutate(total_catch = sum(number_caught)) %>% # sum catch according to group_by()
  ungroup() %>% # ungroup data
```

```
# summarize() illustration

tot_catch2 %>%
  group_by(year, species) %>%
 summarize(n = n(), mean_catch = mean(total_catch, na.rm=T),
          mean_effort = mean(effort, na.rm=T))
# Note the use of "na.rm=T" arguement in the mean function
# this is necessary to ignore missing (NA) values when
# calculating the mean (used in other summary stat functions too)
```

Guide for cleaning the data following the workflow outlined above

1. Create a new "clean" data frame, called `dat_clean` from the data frame we read into **R** and named `dat`. Then, using a pipe, remove columns we will not need using the `select()` function as shown below. Notice that there is another pipe `%>%` after the `select()` function – this is where you will continue to chain together commands for cleaning in step 2. However, if you run the code up until that last pipe, you can see how the select function removed columns.

```
# Create new data frame for our cleaned data
dat_clean <- dat %>%
  select(-c(`Water Name`, `Unique Water Site ID`,
            `Water Section ID`, `Group Size Fish Length`,
            `Survey Purpose`, `Water Site Comment`,
            `Water Site Survey Comment`, Month, Description)) %>% # remove unwanted columns
```

2. Add the `rename()` function onto the above code, after the last `%>%` to rename columns. We will rename the following columns as:

```
Water_site_survey_ID  = `Water Site Survey ID`,
         Lat = `Survey Site Lat DD`,
         Long = `Survey Site Lon DD`,
         Species = `Fish Species Name`,
         Number_caught = `Number Caught`,
         Effort = `Effort Hours`
```

3. Add a pipe and use the `filter()` function to remove the two duplicate surveys. One way to do this is to simply keep (retain) all the rows *except* the duplicates. E.g., we want all `Water_site_survey_ID`'s that don't equal (!=) 35947 & 35971 (these are the duplicate IDs).
4. Add a pipe and use the `mutate()` function to create a new column called `Date` that uses a function to convert the `Site Date` column into a date data type. Because the dates in `Site Date` are in "month/day/year hour minute" format, we can use the function `mdy_hm` (m = month, d = day, y = year, h = hour, m = minute) in the `lubridate` package for this conversion.
5. Add a pipe and remove the old date column called `Site Date` from the data frame since we have a new, properly formatted one called `Date`.
6. Add a pipe and add `rename_with(tolower)` to convert all column names to lower case.

The structure of dat_clean should look something like this:

```
str(dat_clean)
```

```
tibble [1,571 x 9] (S3: tbl_df/tbl/data.frame)
 $ lat                 : num [1:1571] 40.8 40.8 40.8 40.8 40.8 ...
 $ long                : num [1:1571] -75.8 -75.8 -75.8 -75.8 -75.8 ...
 $ water_site_survey_id: num [1:1571] 35942 35942 35942 35942 35942 ...
 $ year                : num [1:1571] 1981 1981 1981 1981 1981 ...
 $ species             : chr [1:1571] "Largemouth Bass" "Largemouth Bass" "Largemouth Bass" "La
 $ szclass             : num [1:1571] 125 150 275 300 375 450 175 325 350 75 ...
 $ number_caught       : num [1:1571] 1 1 5 10 1 2 1 1 1 2 ...
 $ effort              : num [1:1571] 1.4 1.4 1.4 1.4 1.4 1.4 1.4 2.8 2.8 2.8 ...
 $ date                : POSIXct[1:1571], format: "1981-06-16" "1981-06-16" ...
```

Calculate total catch for each year, survey, and species

Using the data frame `data_clean`, let's create a new data frame that is the total catch of each species caught in each survey and year. We will group by `year`, `water_site_survey_id`, and `species` and use `mutate()` to create a new column called `total_catch` which is the sum of `number_caught`. Because we have grouped first, the sum of total catches will be done for each year, survey, and species. We will call this new data frame `dat_tot_catch`.

The code chunk below illustrates what needs to be done. You need to enter code for `XXX`. Notice that after we calculate total catch we ungroup (`ungroup()`). The `distinct` function is retaining distinct combinations of `year`, `water_site_survey_id`, `species` otherwise we have a data frame with the total catch repeated for each size class (szclass), which we don't want. The .keep_all=TRUE argument is making sure we retain all columns after we retain the distinct rows. We can then get rid of number_caught (which is by size class) since we are only interested in total catch (which was summed across size classes).

```
# Calculate total catch for each year, survey, and species
dat_tot_catch <- dat_clean %>%
  group_by( XXX ) %>% # group data
  mutate( XXX ) %>% # sum over catch for each variable in group_by
  ungroup() %>% # ungroup data
  # retain distinct combos since we don't need size-specific numbers here
  distinct(year, water_site_survey_id, species, .keep_all=TRUE) %>%
  select(-c(number_caught)) # remove number caught, no longer needed
```

Filling in species that were not caught in a given survey and year with 0 catch values

Species that were not caught in a given survey and year are not recorded as zero catch, rather they were not entered into the data set. For example, if in a given surve and year only largemouth bass were caught, largemouth bass would be the only species listed and the other 14 species are simply missing from that survey record. However, we would like to know that, for example, no (zero) black crappie were caught in a given survey and year even if largemouth bass were the only species caught. Thus, we have to impute the missing species names into surveys and years where they were not recorded and give them zero catch values. Here is the code to do this and we will create a new data frame called dat_tot_catch2:

```
dat_tot_catch2 <- dat_tot_catch %>%
   # Select columns of interest for summarizing
  select(water_site_survey_id, year, species, total_catch, effort) %>%
  # Input missing species for surveys and years
  complete(nesting(water_site_survey_id, year), species,  fill = list(total_catch=0)) %>%
  arrange(year, species) # sort by year and species
```

Let's look at dat_tot_catch2:

```
# Look at first few rows of tot_catch2
head(dat_tot_catch2, 10)
```

```
# A tibble: 10 x 5
   water_site_survey_id  year species          total_catch effort
                  <dbl> <dbl> <chr>                  <dbl>  <dbl>
 1                35942  1981 Banded Killifish           0     NA
 2                35942  1981 Black Crappie              0     NA
 3                35942  1981 Bluegill                   0     NA
 4                35942  1981 Brown Bullhead             0     NA
 5                35942  1981 Chain Pickerel             0     NA
 6                35942  1981 Channel Catfish            0     NA
```

```
 7                35942  1981 Golden Shiner              0    NA
 8                35942  1981 Green Sunfish              0    NA
 9                35942  1981 Largemouth Bass           20     1.4
10                35942  1981 Pumpkinseed                0    NA
```

Notice how we now have species that were not caught in a given survey and year entered into our data frame with 0 catch. However, they all have `NA` for effort when they should have the same effort for a given survey and year. Let's fix that as follows:

```
# Lets replace the NA values for imputed species efforts
# to the actual effort of the survey
dat_tot_catch2 <- dat_tot_catch2 %>%
  group_by(water_site_survey_id) %>%
  mutate(effort = replace_na(mean(effort, na.rm=T))) %>%
  ungroup()

head(dat_tot_catch2, 10)
```

```
# A tibble: 10 x 5
   water_site_survey_id  year species          total_catch effort
                  <dbl> <dbl> <chr>                  <dbl>  <dbl>
 1                35942  1981 Banded Killifish           0    1.4
 2                35942  1981 Black Crappie              0    1.4
 3                35942  1981 Bluegill                   0    1.4
 4                35942  1981 Brown Bullhead             0    1.4
 5                35942  1981 Chain Pickerel             0    1.4
 6                35942  1981 Channel Catfish            0    1.4
 7                35942  1981 Golden Shiner              0    1.4
 8                35942  1981 Green Sunfish              0    1.4
 9                35942  1981 Largemouth Bass           20    1.4
10                35942  1981 Pumpkinseed                0    1.4
```

Ok, that looks better. Now let's summarize (calculate) total catch (across surveys for each year and species), sample size (# of surveys in a given year), and total effort. To do this we use `group_by()` again and the summarize function, as follows – and we store this summary in an object called `table1` that we will use to make a table (this is a very large Table (255 rows), so we will just show some of it here).

```
# Summarize mean catch (across surveys for each year and species)
# sample size, and mean effort.
table1 <- dat_tot_catch2 %>%
  group_by(year, species) %>%
 summarize(n = n(), total_catch = sum(total_catch, na.rm=T),
           total_effort = sum(effort, na.rm=T)) %>%
  arrange(year, species)
```

Here is the code to make a Table using the summary `table1` we created:

Table 1: Catch and effort (hrs) summary table (first 20 rows).

Year	**Species**	**n (surveys)**	**Total catch**	**Total effort (hrs)**
1981	Banded Killifish	1	0	1.4
1981	Black Crappie	1	0	1.4
1981	Bluegill	1	0	1.4
1981	Brown Bullhead	1	0	1.4
1981	Chain Pickerel	1	0	1.4
1981	Channel Catfish	1	0	1.4
1981	Golden Shiner	1	0	1.4
1981	Green Sunfish	1	0	1.4
1981	Largemouth Bass	1	20	1.4
1981	Pumpkinseed	1	0	1.4
1981	Rock Bass	1	0	1.4
1981	Smallmouth Bass	1	0	1.4
1981	Walleye	1	1	1.4
1981	Yellow Bullhead	1	0	1.4
1981	Yellow Perch	1	0	1.4
1986	Banded Killifish	1	0	2.8
1986	Black Crappie	1	0	2.8
1986	Bluegill	1	0	2.8
1986	Brown Bullhead	1	0	2.8
1986	Chain Pickerel	1	2	2.8

```
# Create a table of the catch summary
table1[1:20,] %>%
  kbl(caption = "Catch and effort (hrs) summary table (first 20 rows).", digits=2, booktabs
      col.names = c("Year",
                    "Species",
                    "n (surveys)",
                    "Total catch",
                    "Total effort (hrs)")) %>%
  row_spec(0,bold=TRUE) %>%
  kable_classic(full_width = F, html_font = "Cambria")
```