Workshop

Exercise # 2: Data visualization

Goal

The goal of this exercise is to gain experience plotting data using `ggplot`. Specifically, we will make a simple map of our study site (Mauch Chunk Lake) and create barplots illustrating catch-per-unit effort for different species over time.

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:

```r
library(dplyr) # data manipulation
library(tidyverse) # data manipulation
library(lubridate) # work with dates
library(kableExtra) # make tables
# New for Ex. 2
library(ggplot2) # plotting data and spatial (simple feature; sf) objects
library(sf) # map creation (simple features)
library(spData) # provides access to polygons of US states
```

Getting started

Open **R** from the `.proj` file associated with this workshop (`R_Workshop.Rproj`) and navigate to the `07_Exercises` –> `Ex_2` folder and open the `exercise_2.R` script. This script contains all the code from Exercise # 1 and you will start Exercise # 2 at the end of that code.

The dataset

The dataset is the same as used for Exercise #1. This exercise will map out our study site and plot the data summary we created (i.e., plot the data frame called `table1`). If you recall, `table1` contained the total catch and total effort for each species and year. We will want to visualize catch-per-unit effort time series for select species. The **R** chunk below shows the first few rows of `table1`.

```
head(table1, 15)
```

```
# A tibble: 15 x 5
# Groups:   year [1]
    year species                n total_catch total_effort
   <dbl> <chr>              <int>       <dbl>        <dbl>
 1  1981 Banded Killifish       1           0          1.4
 2  1981 Black Crappie          1           0          1.4
 3  1981 Bluegill               1           0          1.4
 4  1981 Brown Bullhead         1           0          1.4
 5  1981 Chain Pickerel         1           0          1.4
 6  1981 Channel Catfish        1           0          1.4
 7  1981 Golden Shiner          1           0          1.4
 8  1981 Green Sunfish          1           0          1.4
 9  1981 Largemouth Bass        1          20          1.4
10  1981 Pumpkinseed            1           0          1.4
11  1981 Rock Bass              1           0          1.4
12  1981 Smallmouth Bass        1           0          1.4
13  1981 Walleye                1           1          1.4
14  1981 Yellow Bullhead        1           0          1.4
15  1981 Yellow Perch           1           0          1.4
```

Create a map showing Mauch Chunk Lake

This is a very simple illustration of making a map in **R**, but highlights the basic steps of map creation using the `sf` package (the preferred mapping package) along with `ggplot2` - the preferred package for creating figures/graphics.

Basic mapping steps

1. Convert the data frame (that contains lat/long coordinates) to a spatial object - a simple features (`sf`) object in this case.
2. Set the desired coordinate references system (crs).
3. Use `ggplot` and the `geom_sf` geometric object(s) to create a map.
4. Add any additional desired layers to customize/improve aesthetics of map.

For this example, we only have a single site to map (a single lat/long). The lat/long coordinates for Mauch Chunk Lake are in the original data frame we read into **R** and also in the data frame `dat_tot_catch` we created. Because the lat/long coordinates are repeated for every row in the dataset, we can just select a single row of `dat_tot_catch` for this map. The following code for selecting the first row can be used, where we use square bracket indexing ([rows,columns]) to indicate we want the first row and all columns (`[1, ]`) – leaving the second index for columns empty indicates to select *all* columns.

```r
# Subset first row and all columns (designated as [1,]) of dat_tot_catch
map_dat <- dat_tot_catch[1,]
# Look at map_dat
map_dat
```

```
# A tibble: 1 x 9
    lat  long water_site_survey_id  year species         szclass effort
  <dbl> <dbl>                <dbl> <dbl> <chr>             <dbl>  <dbl>
1  40.8 -75.8                35942  1981 Largemouth Bass     125    1.4
# i 2 more variables: date <dttm>, total_catch <dbl>
```

Again, for simply mapping a single lat/long coordinate, all we really care about in `map_dat` is the lat/long columns. However, if we were plotting multiple sites, the steps we are using are exactly the same.

As of now, `map_dat` is still a data frame (among other data classes).

```r
class(map_dat)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

We need to convert `map_dat` to a `sf` object using the `st_as_sf` function and the following code. We need to specify which data frame we are converting, what columns in `map_dat` contain lat/longs, and the desired crs.

```r
# Convert map_dat to a spatial (sf) object (simple feature)
# and set its coordinate references system (crs)
# crs = 4326 = WGS84; WGS84 CRS is often used for lat and long positions
map_dat <- st_as_sf(map_dat, coords = c("long", "lat"), crs = 4326)
class(map_dat)
```

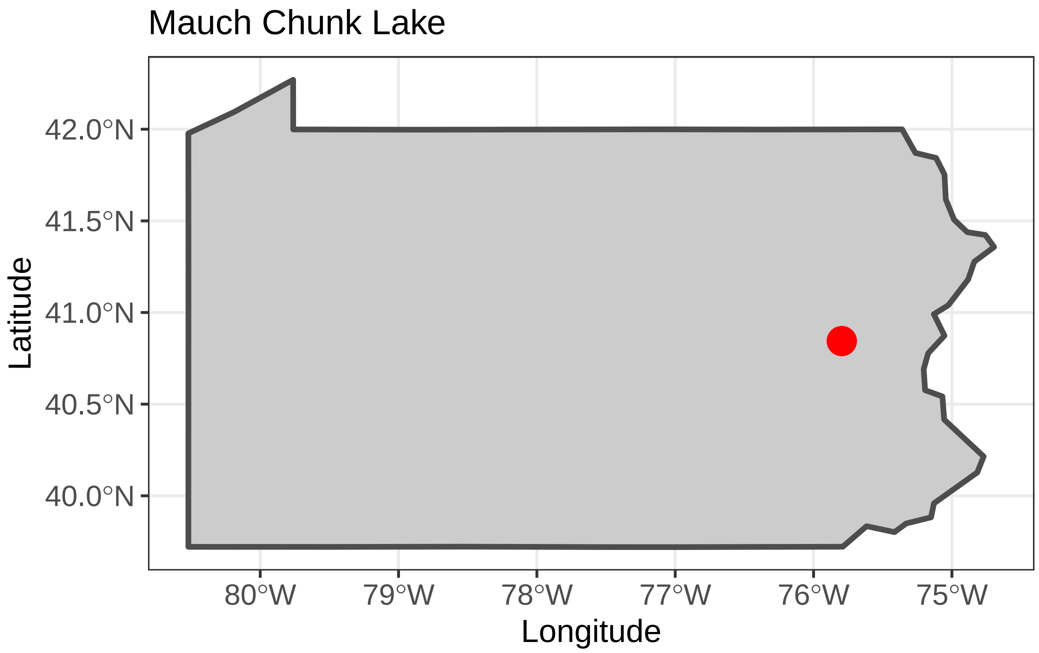
```
[1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

Looking at the class of map_dat, we can now see that it is an `sf` object. Obviously, we don't just want a single lat/long point floating out in space, so we will want to add at least one additional layer to our map – in this case it will be the state of PA. You can easily read in shapefiles or other spatial data into **R**; however, some packages have spatial data we can easily access for mapping. We will use the `spData` package to grab a polygon for the state of PA. The below **R** chunk illustrates the steps to do this.

```
# Grab state boundaries from the spData package (called us_states) and
# transform them to our desired coordinate references system (crs)
# crs = 4326 = WGS84; WGS84 CRS is often used for lat and long positions.
# We will call this new transformed state sf object "us_states2".
us_states2 <- st_transform(us_states, crs = 4326)
# Rename column to "State" in our object us_states2
colnames(us_states2)[2] <- "State"
# Select state(s) of interest - place them in a vector (can be > 1 state)
selectStates <- c("Pennsylvania")
# Subset us_states2 data to grab our selectStates for plotting
# The %in% syntax is matching what is in us_states2$State with
# what is in selectStates (i.e., Pennsylvania, in this case)
us_state_select <- us_states2[us_states2$State %in% selectStates, ]
```

Now we have our lat/long point for Mauch Chunk Lake in the sf object `map_dat` and our polygon of PA in our `sf` object `us_state_select`. We can use the `ggplot` along with `geom_sf` to add our spatial data as follows:

```
# Create map of PA (us_state_select) and our single point
# for Mauch Chunk Lake (map_dat)
ggplot() +
  geom_sf(data = us_state_select, color = "gray30", lwd=1, fill="grey80") +
  geom_sf(data=map_dat, shape=16, size = 5, colour="red") +
  labs(title="Mauch Chunk Lake", y="Latitude", x="Longitude") +
  theme_bw() +
  theme(axis.text = element_text(size = 11),
        axis.title = element_text(size = 12))
```



Plot catch-per-unit effort over time

For this plotting example, we will use the `table1` summary we created in Exercise # 1 (**R** chunk below).

```
head(table1, 5)
```

```
# A tibble: 5 x 5
# Groups:   year [1]
   year species               n total_catch total_effort
  <dbl> <chr>             <int>       <dbl>        <dbl>
1  1981 Banded Killifish      1           0          1.4
2  1981 Black Crappie         1           0          1.4
3  1981 Bluegill              1           0          1.4
4  1981 Brown Bullhead        1           0          1.4
5  1981 Chain Pickerel        1           0          1.4
```

We will start with plotting a time series of catch-per-unit effort for a single species. We will focus on largemouth bass. Let's look at the class for `table1`.

```
class(table1)
```

```
[1] "grouped_df" "tbl_df"     "tbl"        "data.frame"
```

Notice that one of the class types is `grouped_df` – a grouped data frame. If you recall from Exercise # 1, we usually want to ungroup our data frames after they are grouped to perform some calculations/summaries. In this case, we did not ungroup `table1`, but we will want to do so before getting this data frame in shape for plotting. Recall that we can use the `ungroup()` function to do this.

The **R** code to prep our data for plotting a time series of largemouth bass (lmb) catch-per-unit effort is as follows. We will call our new data frame for plotting `dat_lmb_plot`.

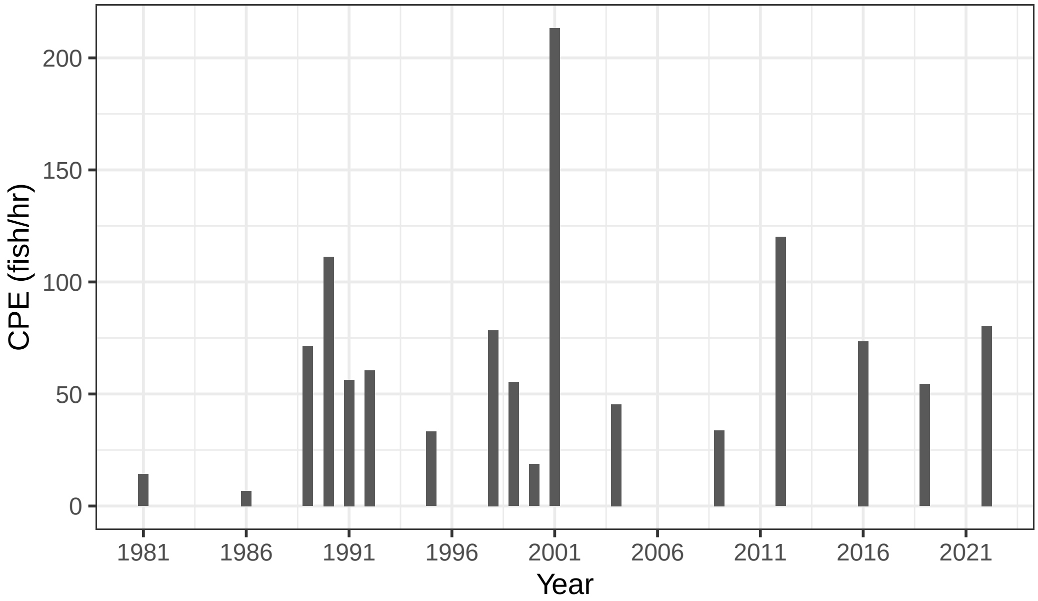
```
dat_lmb_plot <- ungroup(table1) %>%
  filter(species == "Largemouth Bass") %>%
  mutate(cpe = total_catch/total_effort) %>%
  complete(year = min(year):max(year))
```

Notice the sequence of operations:

1. ungroup `table1`
2. Filter (select) data for when species is euqal to "Largemouth Bass" – our focal species we want to plot
3. Calculate catch-per-unit effort and call it `cpe`
4. Fill in years that were not sampled so we can plot a complete time series

We can then use the following `ggplot` code for our figure:

```
# Plot cpe over time
ggplot(data=dat_lmb_plot, aes(x=year, y=cpe)) +
  geom_bar(stat="identity", width=0.5) +
  labs(title="", y="CPE (fish/hr)", x="Year") +
  theme_bw() +
  scale_x_continuous(breaks=seq(1981,2022,5))
```



Plot catch-per-unit effort for multiple species

Task: your task is to use the previous two code chunks and modify them to make a plot showing catch-per-unit effort time series for three species.

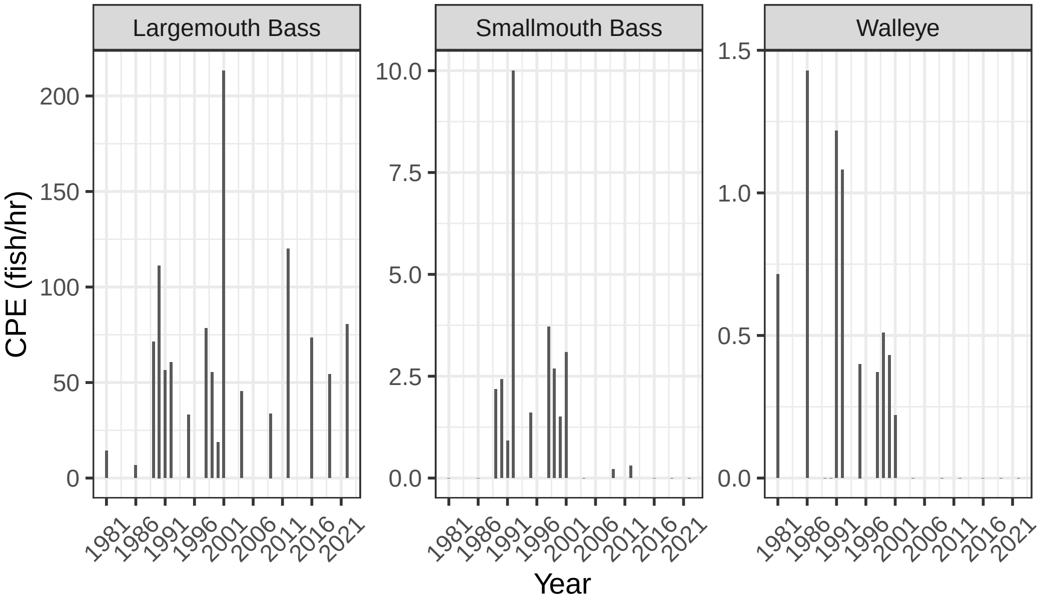
Details:

1. The focal species are Largemouth Bass, Smallmouth Bass, and Walleye
2. To make a multi-panel plot in `ggplot` we can use the following syntax:

```
  facet_wrap(~species, scales = "free_y")
```

Where the `~species` indicates we want a separate panel for each species time series and `scales="free_y"` indicates that each panel will have its own y-axis range.

The final figure will look like this:


Largemouth Bass
Smallmouth Bass
Walleye
CPE (fish/hr)
0
50
100
150
200
0.0
2.5
5.0
7.5
10.0
0.0
0.5
1.0
1.5
1981
1986
1991
1996
2001
2006
2011
2016
2021
Year