R Graphics Workshop Part I

Jereme W Gaeta, PhD Senior Environmental Scientist - Specialist
California Department of Fish and Wildlife Interagency Ecological Program

IEP 2020 Annual Workshop (March 20, 2020)

R Graphics Workshop Outline

The goal of this workshop is to introduce intermediate R users to alternative plotting styles and provide sample code to guide R users in the creation of publishable figures. This workshop will cover the following topics:

- Part I
  - Plotting with dates
  - Axis labels with Greek letters, subscripts, and superscripts
  - Creating a back-transformed second axis
  - Plotting stacked polygons
  - Advanced legends
- Part II
  - Visualizing linear models (working toward transparent visualizations)
  - Visualizing model uncertainty (confidence intervals) using polygons
  - Adding a color scale bar to a plot
  - Formatting and exporting figures for publication

Load and Explore the data

We will be using data from the IEP 'zooper' R Cran Package developed by Sam Bashevkin (Delta Stewardship Council). The first dataset we will be using is a time series dataset of catch per unit effort (CPUE; number per L) of two calanoid (Subclass: Copepoda) species: *Eurytemora affinis* and *Pseudodiaptomus forbesi*. The second dataset is a similar timeseries, but with catch per unit effort (number per L) of all "meso" zooplankton (i.e., zooplankton collected with 150-160 μm mesh).

Let's explore the first dataset:

```
tdat = read.csv(file = "zooper_delta_smelt_food.csv", header=TRUE)
names(tdat)
```

```
## [1] "date"                    "stratum"
## [3] "Eurytemora.affinis"      "Pseudodiaptomus.forbesi"
## [5] "prop_E_affinis"          "prop_P_forbesi"
## [7] "date_2"                  "date_3"
```

The dataset has raw CPUE and proportion of CPUE for each species. The dataset also includes two different date formats for an exercise to follow.

```
head(tdat)
```

```
##         date                      stratum Eurytemora.affinis
## 1 1977-06-06 Cache Slough/Liberty Island          718.51798
## 2 1977-06-21 Cache Slough/Liberty Island          337.56886
## 3 1977-07-06 Cache Slough/Liberty Island          134.71866
## 4 1977-07-20 Cache Slough/Liberty Island           44.34537
## 5 1977-08-03 Cache Slough/Liberty Island          186.80240
## 6 1977-08-18 Cache Slough/Liberty Island           95.88115
##   Pseudodiaptomus.forbesi prop_E_affinis prop_P_forbesi       date_2     date_3
## 1                       0              1              0 Jun 06, 1977 06/06/1977
## 2                       0              1              0 Jun 21, 1977 06/21/1977
## 3                       0              1              0 Jul 06, 1977 07/06/1977
## 4                       0              1              0 Jul 20, 1977 07/20/1977
## 5                       0              1              0 Aug 03, 1977 08/03/1977
## 6                       0              1              0 Aug 18, 1977 08/18/1977
```

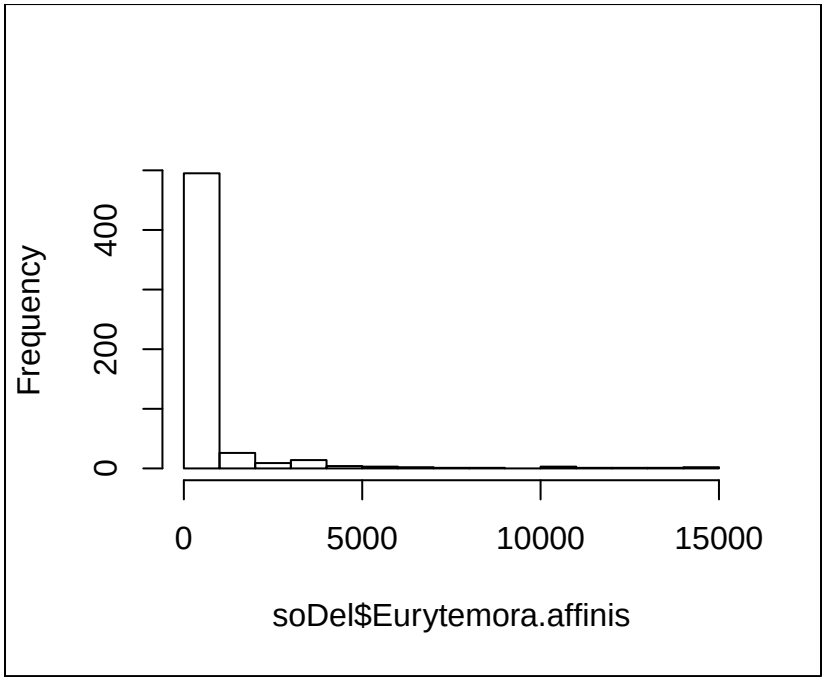
Working with dates

Let's start with just a subset of the data. Subset data from the Southern Delta:

```
soDel = subset(tdat, tdat$stratum=="Southern Delta")
```

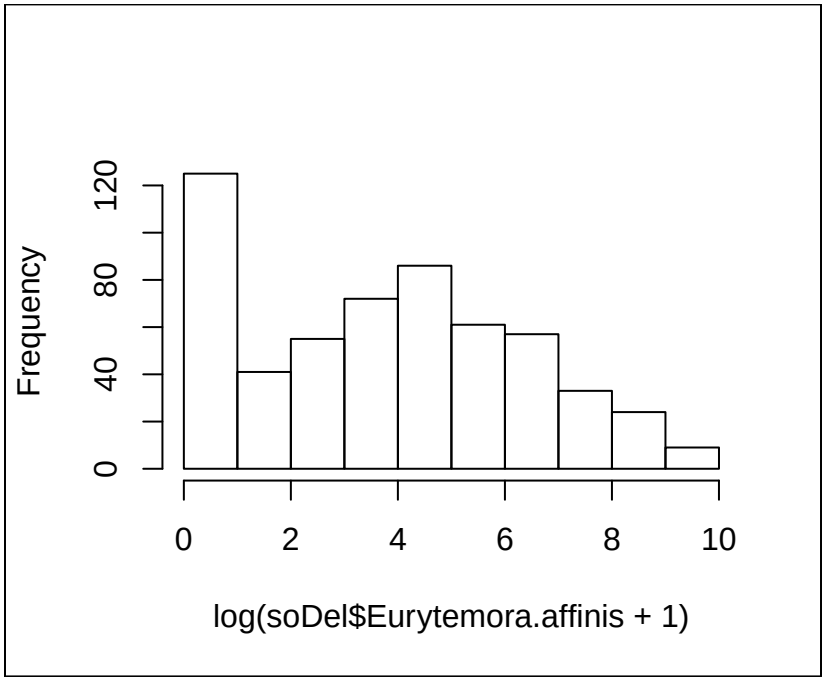
plot a histogram of the CPUE data for one of the species:

```
hist(soDel$Eurytemora.affinis, main="")
box(which="outer")
```



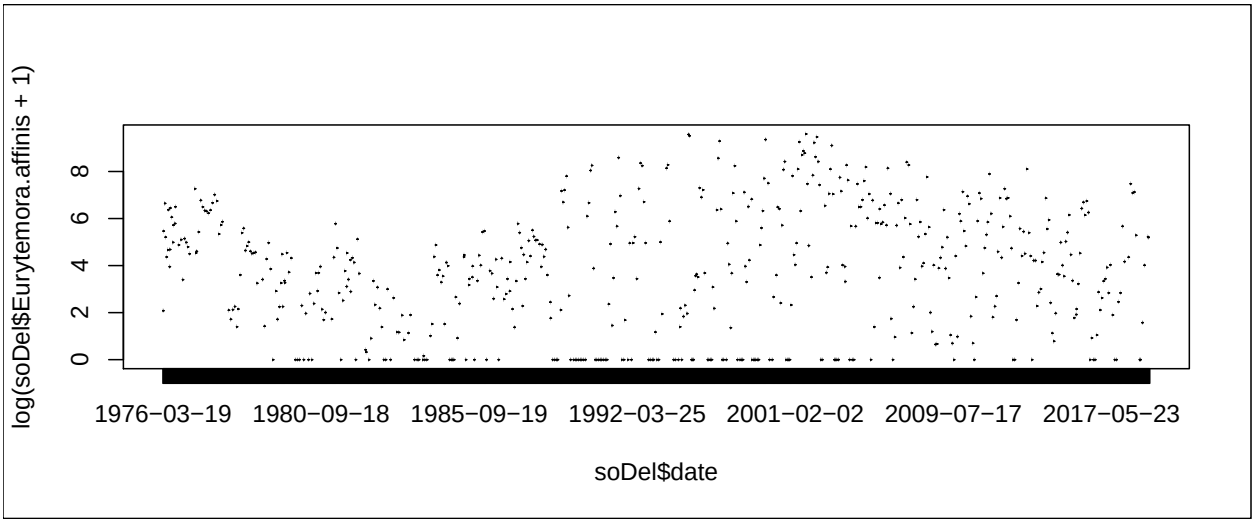
The data are left skewed, so let's $\log_e$-transform the CPUE data to improve our ability to see shifts in CPUE at low values. *Note: the dataset has zero-values for both species, which cannot be $\log_e$-transformed, so we add a constant (e.g., 1) to each value.*

```
hist(log(soDel$Eurytemora.affinis+1), main="")
box(which="outer")
```



Now that the data are transformed, let's plot *Eurytemora affinis* over time.

```
plot(log(soDel$Eurytemora.affinis+1) ~ soDel$date)
box(which="outer")
```



What happened!?! When weird things happen with a variable, I usually start troubleshooting by checking the data classes of the of the variables. We can check each variable directly using the ***class( )*** function or assess the whole dataset using ***str( )***:

```
class(soDel$Eurytemora.affinis)
```

```
## [1] "numeric"
```

```
class(soDel$date)
```

```
## [1] "factor"
```

```
str(soDel)
```

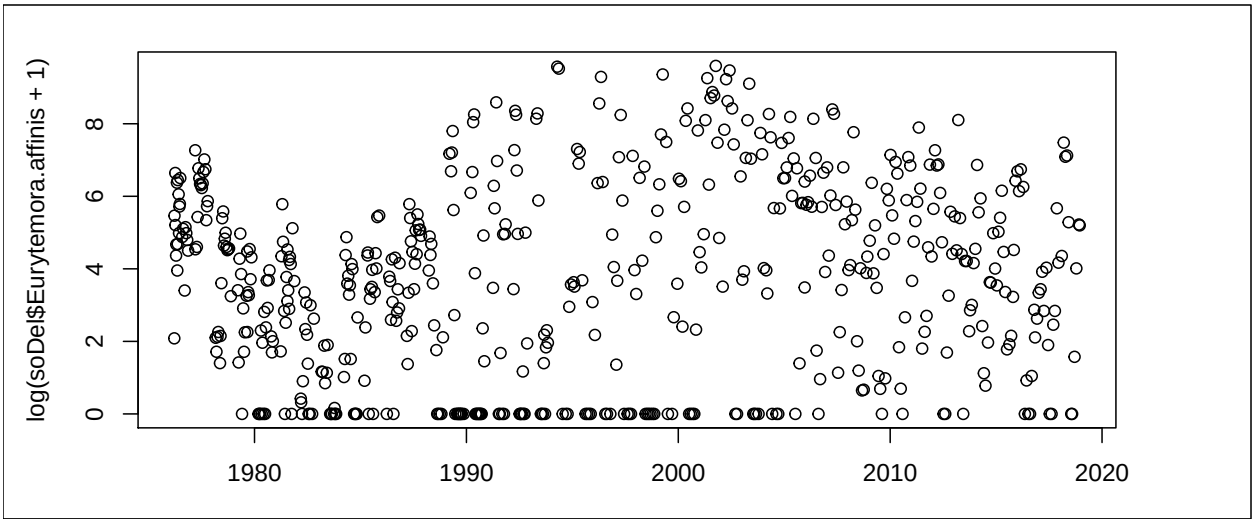
```
## 'data.frame':    563 obs. of  8 variables:
##  $ date                   : Factor w/ 2922 levels "1976-03-19","1976-03-22",..: 1 2 6 8 11 15 16 20 1
##  $ stratum                : Factor w/ 10 levels "Cache Slough/Liberty Island",..: 8 8 8 8 8 8 8 8 8
##  $ Eurytemora.affinis     : num  7.04 235.29 766.27 181.51 77.68 ...
##  $ Pseudodiaptomus.forbesi: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ prop_E_affinis         : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ prop_P_forbesi         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ date_2                 : Factor w/ 2922 levels "Apr 01, 1981",..: 1736 1761 21 40 200 246 1846 190
##  $ date_3                 : Factor w/ 2922 levels "01/03/2001","01/04/2001",..: 407 432 520 539 699 7
```

R thinks all of the dates are factors. So, we need to tell R that these values are dates, not factors. We can do this using the as.Date( ) function. Typing ***?as.Date*** into the console will bring up the help documentation. According to the documentation, we need to set the *format* argument.

A few important date codes (*see* ***help(strptime)*** *for more details and examples*):

- '**%Y**' = Four character year
- '**%y**' = two character year
- '**%m**' = month as decimal number
- '**%b**' = abbreviated month name
- '**%d**' = day of month as decimal number

```
par(mfrow=c(1,1), mar=c(3,4.5,1.5,4.5)+0.1)
plot(log(soDel$Eurytemora.affinis+1) ~ as.Date(soDel$date, format = "%Y-%m-%d"))
box(which="outer")
```

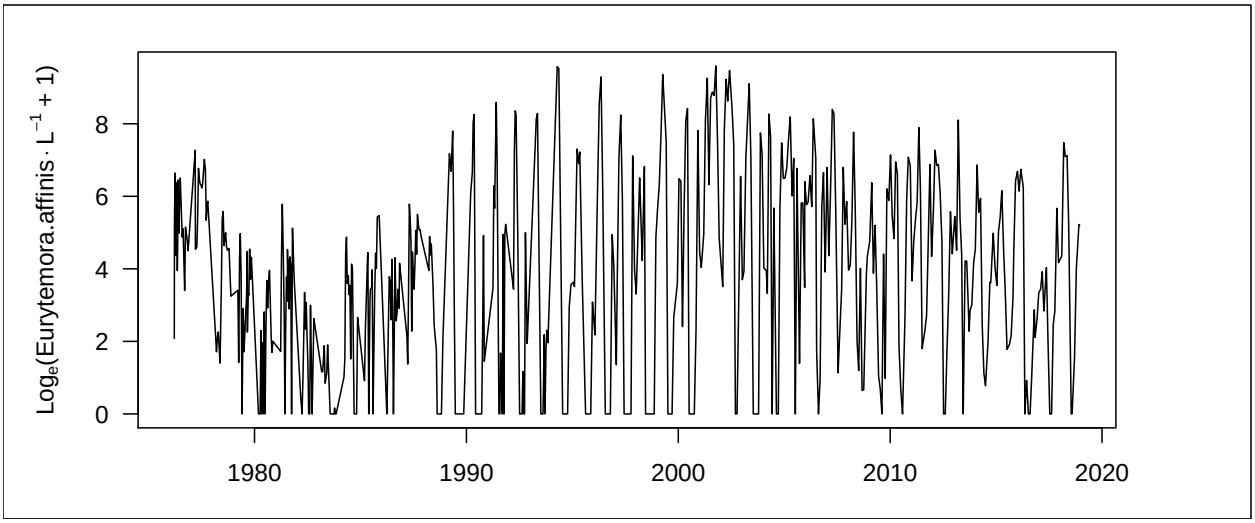


Now let's improve the aesthetics of the plot:

- ***plot( )***
  - **type='l'** = plot points into a line
  - **las=1** = plot axis labels as perpendicular to x axis (my personal recommendation)
- ***mtext( )*** = add an axis label
  - ***bquote( )*** = a function to plot complex text
    - '[ ]' = subscript
    - '^' = superscript
    - '%.%' = centered dot (i.e., multiplication)
    - '*' = no space between elements
    - '~' = add space between elements

  (*see* ***help(plotmath)*** *for more details and examples*)

```
par(mfrow=c(1,1), mar=c(3,4.5,1.5,4.5)+0.1)
plot(log(soDel$Eurytemora.affinis +1) ~ as.Date(soDel$date, format = "%Y-%m-%d"), type='l',
     las=1, ylab = "", xlab="")
mtext(text = bquote(Log[e]*"("*Eurytemora.affinis %.%L^-1~"+ 1)"),
      side=2, line=2.5)
box(which="outer")
```



A final aesthetic option is to add a second, back-transformed y-axis. This involves a two step process:

1. determine the back-transformed values you would like to plot
2. add the axis to the plot

Use ***exp( )*** (*exponentiate*) to back transform the $\log_e$ values, and don't forget to substract the constant of 1.

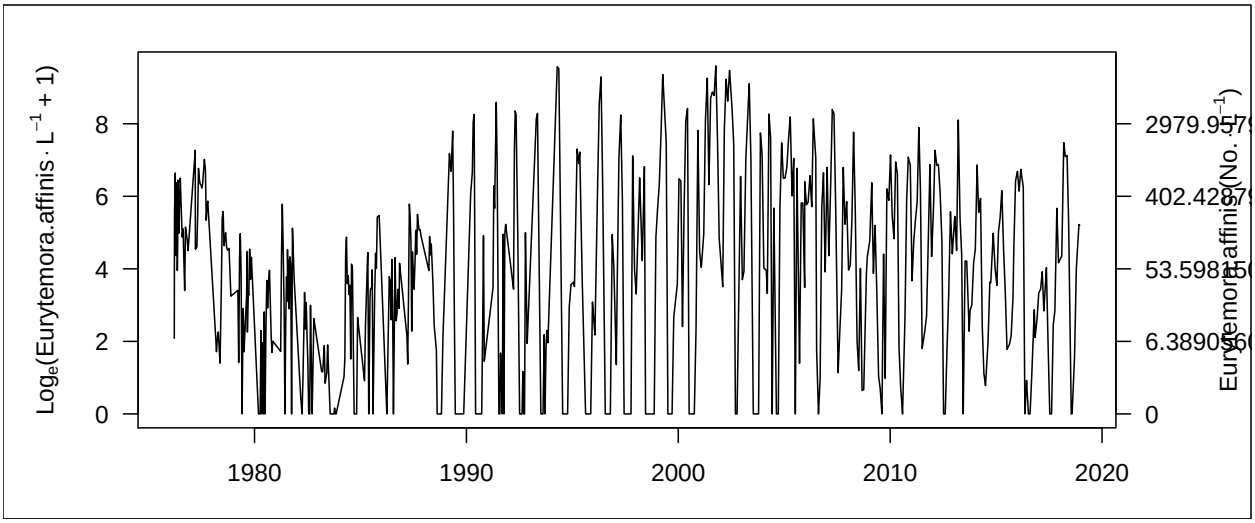
```
exp(seq(from=0, to=10, by=2))-1
```

```
## [1]     0.000000     6.389056    53.598150   402.428793  2979.957987
## [6] 22025.465795
```

We can use the ***axis( )*** function to add an axis

- ***axis( )***
  - **side** = 1: bottom, 2: left, 3: top, 4: right
  - **at** = the $\log_e$-transformed location of axis labels
  - **labels** = the back-transformed axis labels

```
axis(side = 4, at = seq(from=0, to=10, by=2),
     labels=exp(seq(from=0, to=10, by=2))-1,
     las=1)
mtext(text = bquote(Eurytemora.affinis ~"("*No.%.%L^-1*")"),
      side=4, line=3.5)
```

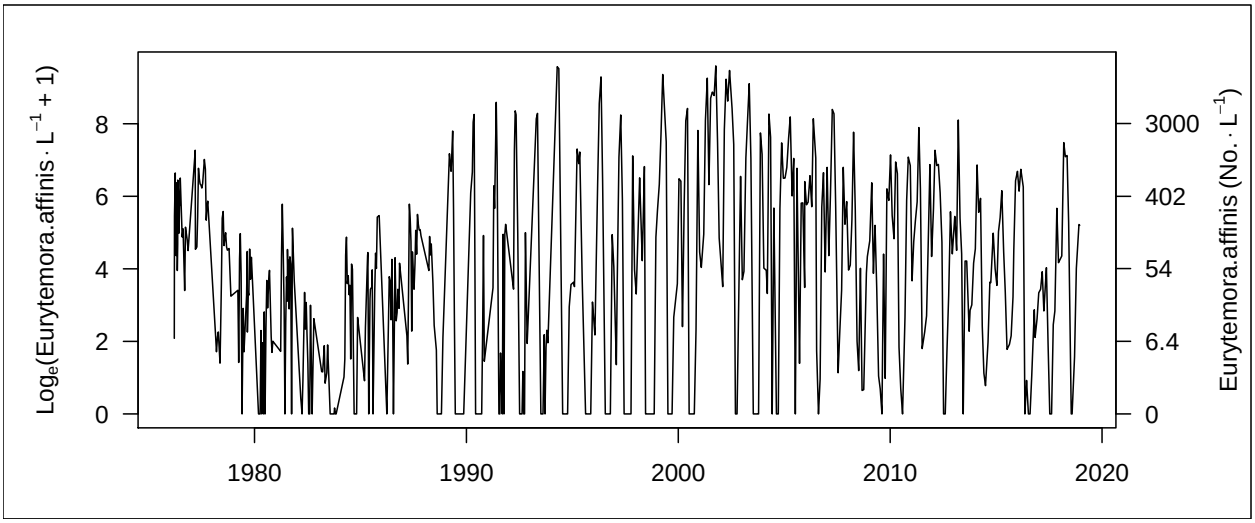


The axis labels are not very pretty, so we can clean this up by selecting rounded values.

```
round(exp(seq(from=0, to=10, by=2))-1, digits=2)
```

```
## [1]     0.00     6.39    53.60   402.43  2979.96 22025.47
```

```
ax_labs = c(0, 6.4, 54, 402, 3000, 22000)
ax_ats = log(ax_labs+1)

par(mfrow=c(1,1), mar=c(3,4.5,1.5,4.5)+0.1)
plot(log(soDel$Eurytemora.affinis +1) ~ as.Date(soDel$date, format = "%Y-%m-%d"), type='l',
     las=1, ylab = "", xlab="")
mtext(text = bquote(Log[e]*"("*Eurytemora.affinis %.% L^-1~"+ 1)"),
      side=2, line=2.5)
axis(side = 4, at = ax_ats,
     labels=ax_labs,
     las=1)
mtext(text = bquote(Eurytemora.affinis ~"("*No.%.%L^-1*")"),
      side=4, line=3.5)
box(which="outer")
```



Exercise

Generate the previous figure using the **date_2** *and/or* **date_3** *columns.*

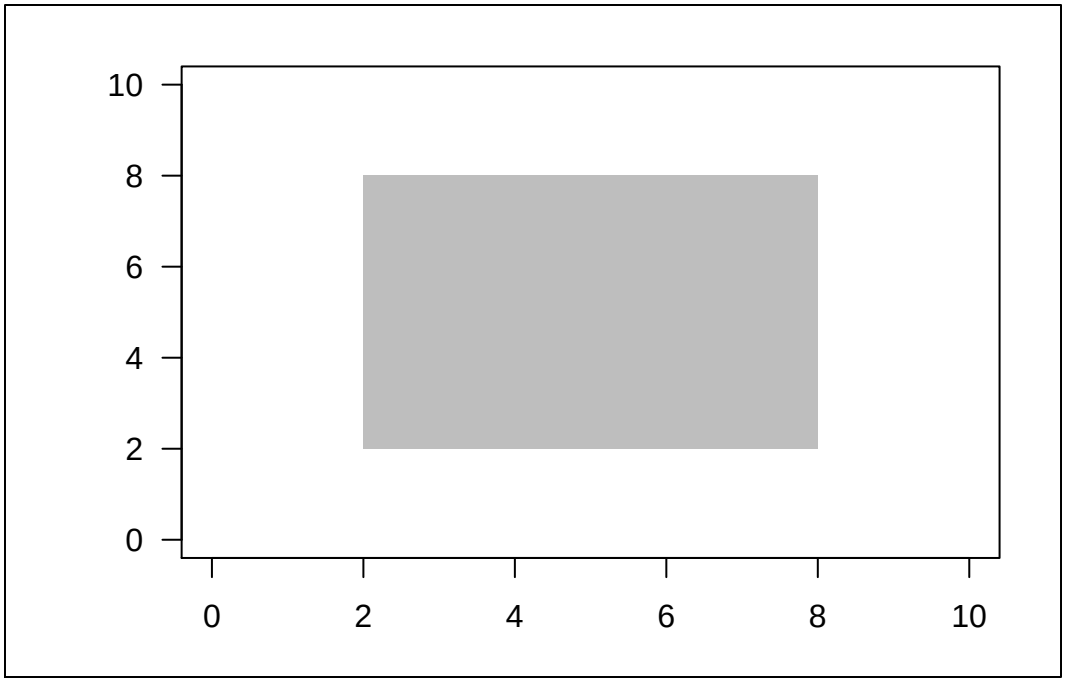
Polygons

Polygons can be thought of as connect-the-dots puzzles in which we tell R the sequence of x-values and y-values of each dot we want to connect. The ***polygon( )*** function has four relevant arguments:

- ***polygon( )***
    1. **x** = The sequence of x-values; the first and last value must be identical to close the polygon
    2. **y** = The sequence of y-values; the first and last value must be identical to close the polygon
    3. **border** = most often set to NULL
    4. **col** = polygon fill color

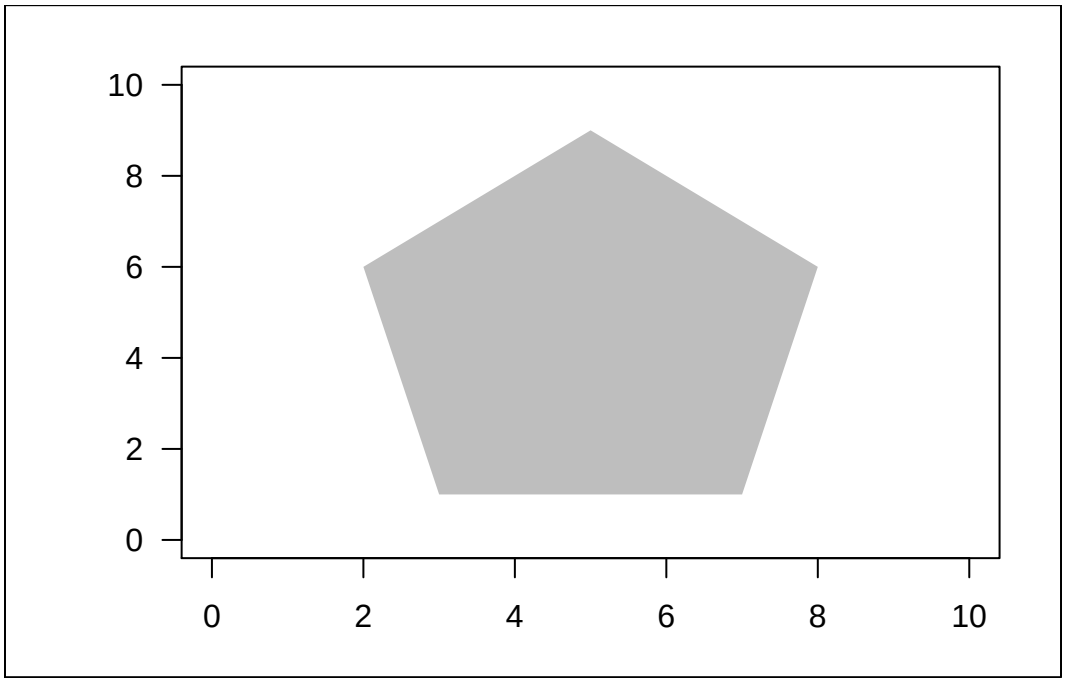
Let's start with a simple square.

```
par(mfrow=c(1,1), mar=c(3,4.5,1.5,1.5)+0.1)
plot(c(0,10), c(0,10), type='n',
     las=1, ylab = "", xlab="")
polygon(x = c(2,8,8,2,2), #bottom left, bottom right, top right, top left, bottom left
        y = c(2,2,8,8,2), #bottom left, bottom right, top right, top left, bottom left
        border=NA,
        col = "gray")
box(which="outer")
```



Exercise

Using the code above as a template, try to replicate the following plot and polygon:



Stacked Polygons

In the section, we are going to visualize both Copepod species (*Eurytemora affinis* and *Pseudodiaptomus forbesi*) in a stacked polygon plot using the ***polygon( )*** function. The process involves three steps:

1. Create a blank plot
2. Add the first polygon, which is a sum of both species
3. Add the second polygon, which is just one of the species

Let's start with creating the summed polygon values and then subsetting to evaluate the Southern Delta:

```
tdat$stacked = rowSums(tdat[,3:4])
tdat[5000:5005,c(3,4,9)]
```

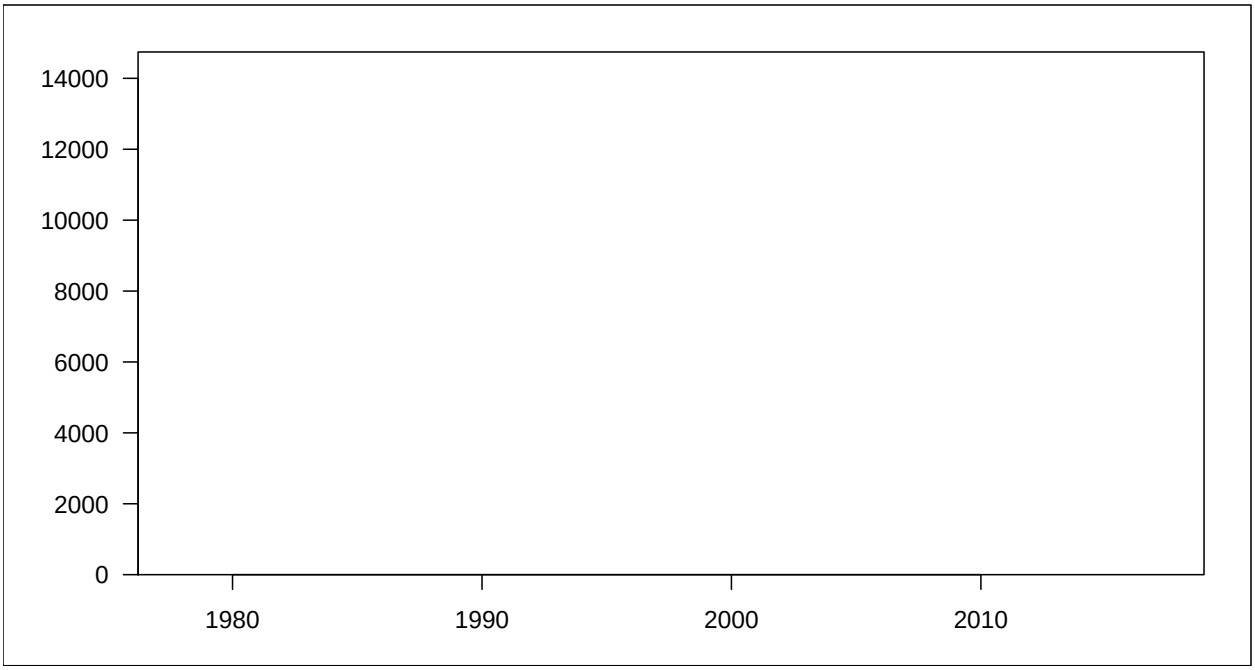
```
##      Eurytemora.affinis Pseudodiaptomus.forbesi    stacked
## 5000          843.14821               0.6764699  843.82468
## 5001          302.96745               0.0000000  302.96745
## 5002           94.02376               0.0000000   94.02376
## 5003         1202.70648              20.5950323 1223.30151
## 5004         1541.79291              11.8464268 1553.63934
## 5005          371.42099               8.7196935  380.14068
```

```
soDel = subset(tdat, tdat$stratum=="Southern Delta")
```

Now, create a blank plot using range of values to be plotted. A few relevant ***plot( )*** arguments include:

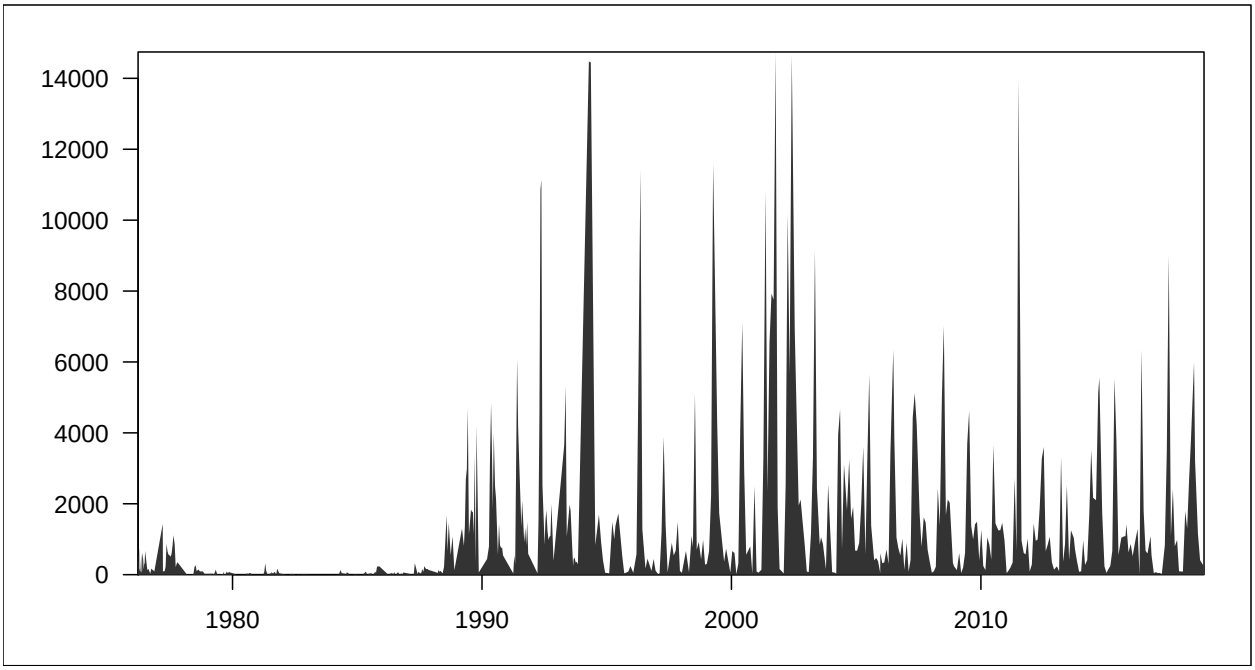
- ***plot( )***
  - **type='n'** = 'n' tells R to plot axes, but not to plot data
  - **xaxs='i'** = R usually adds a buffer to axis-limits; this forces R to constrain to the exact range of x
  - **yaxs='i'** = same as above, but for the y-axis

```
par(mfrow=c(1,1), mar=c(3,4.5,1.5,1.5)+0.1)
plot(x = range(as.Date(soDel$date, format = "%Y-%m-%d")),
     y = range(soDel$stacked),
     type='n', las=1, ylab="", xaxs='i', yaxs='i')
box(which="outer")
```



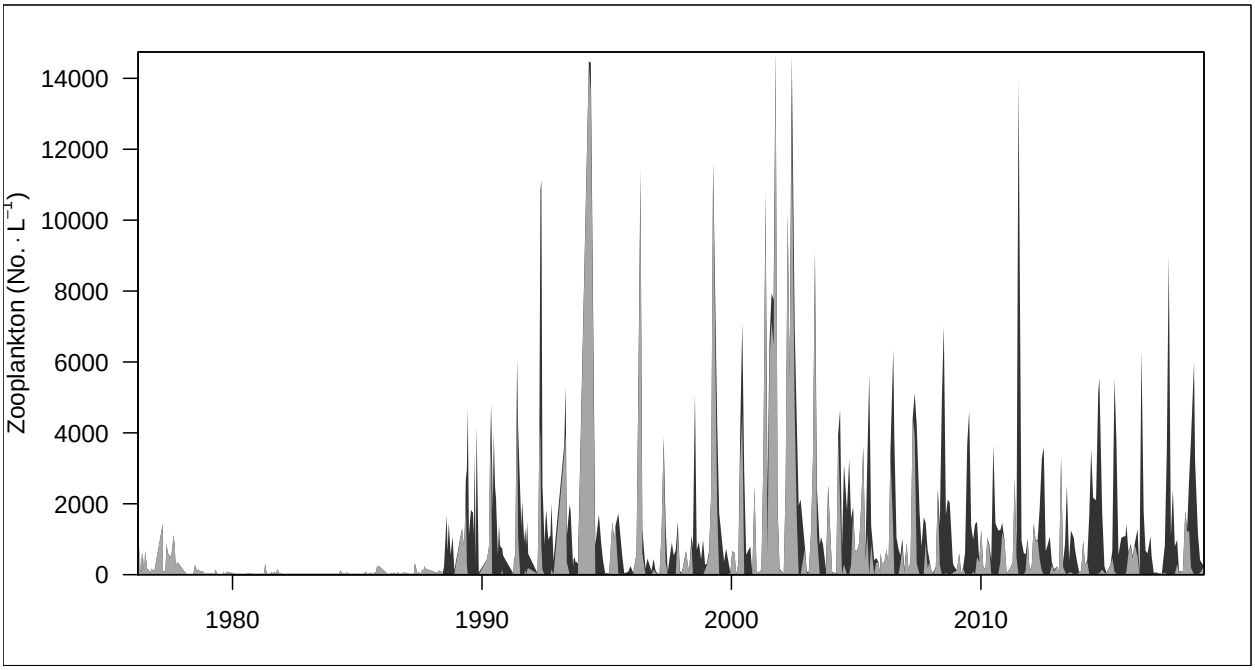
Next, lets connect-the-dots of the stacked data.

```
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$stacked,
              0,0,
              soDel$stacked[1]),
        col="gray20",  border=NA)
```



If we overlay the connect-the-dots data for just one species, the difference (remaining dark gray) will represent the other species.

```
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$Eurytemora.affinis,0,0,soDel$Eurytemora.affinis[1]),
        col=rgb(53,151,143,maxColorValue = 255),  border=NA)
mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.5)
box(which="plot")
box(which="outer")
```



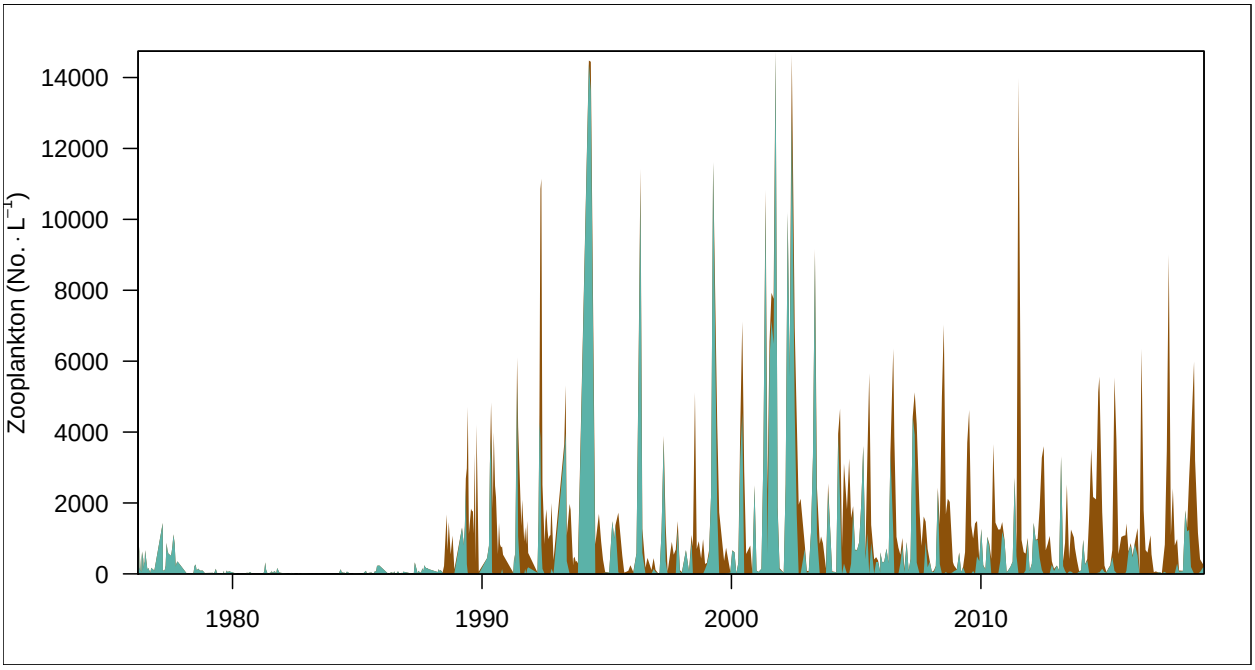
Gray-scale is always best for print publications, but a little (well thought out) flair is suitable for presentations and online publications. Prior to selecting colors, one must consider the audience, particularly if R users are state of federal employees given the American Disabilities Act. Your best option for final color selection is https://webaim.org/resources/contrastchecker/. However, a great starting point is http://colorbrewer2.org/; this website has options for "colorblind safe" as well as "photo copy safe". Finally, a very useful tool (particularly when not worrying about contrasting colors or ADA-compliance) is https://www.colorspire.com/rgb-color-wheel/.

Ok, let's add some color using the ***rgb( )*** function. Relevant arguments include:

- ***rgb( )***
  - **red** = 0:255
  - **green** = 0:255
  - **blue** = 0:255
  - **maxColorValue** = 255; 255 is the standard for rgb values. However, the three numerical values are all relative to the value you set here (i.e., you can set maxColorValue=1 and then set the red, green, and blue arguments to a value from 0 to 1)
  - **alpha** = transparency value. We will circle back to this in part II of the workshop

```
par(mfrow=c(1,1), mar=c(3,4.5,1.5,1.5)+0.1)
plot(x = range(as.Date(soDel$date, format = "%Y-%m-%d")),
     y = range(soDel$stacked),
     type='n', las=1, ylab="", xaxs='i', yaxs='i')
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$stacked,
              0,0,
              soDel$stacked[1]),
        col=rgb(140,81,10,maxColorValue = 255),  border=NA)
```

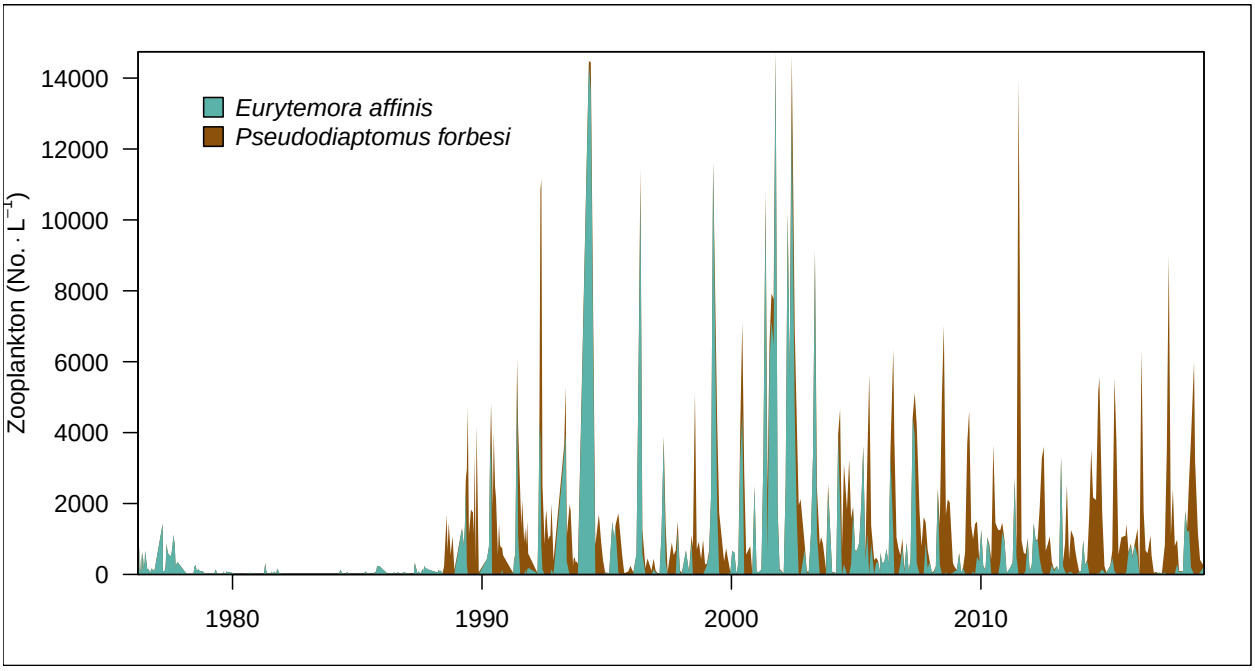
```
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$Eurytemora.affinis,0,0,soDel$Eurytemora.affinis[1]),
        col=rgb(91,178,168,maxColorValue = 255),  border=NA)
mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.5)
box(which="plot")
box(which="outer")
```



Adding a plot legend will make for a good final touch. The ***legend( )*** function has numerous relevant arguments. I will lay out a few:

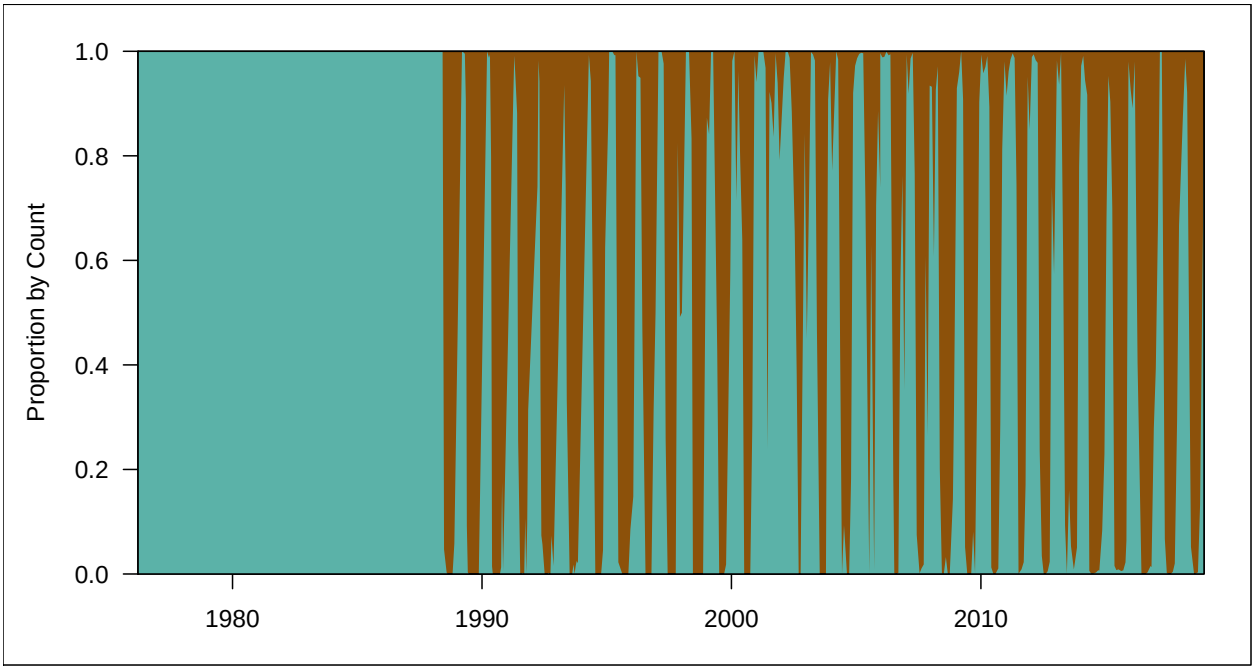
- ***legend( )***
  - **x, y** = you can set the exact x-y coordinates ***OR*** use a keyword (usually my preference)
    - *keywords* are "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center"
  - **pch** = point style; see *fill* below
  - **fill** = using the fill command will plot a small square with the color(s) stated here. Alternatively, I prefer to use the filled square option of pch=22 and set the point background color using pt.bg. This allows me to control the size of the filled square
  - **pt.cex** = size of the pch point
  - **pt.bg** = the background color if using pch = 21:25
  - **inset** = distance the legend is inset from the axes
  - **bty** = option to plot (bty='o') or not plot (bty='n') a box around the legend
  - **text.font** = legend font (3 = italics for species names)

```
legend("topleft", legend = c("Eurytemora affinis",
                             "Pseudodiaptomus forbesi"),
       pch=22, pt.cex=2,
       pt.bg = c(rgb(91,178,168,maxColorValue = 255),
                 rgb(140,81,10,maxColorValue = 255)),
       inset=0.05, bty='n', text.font=3)
```



Exercise

Using the code above as a template, try to replicate the following figure. *Hint: you will have to do some data manipulation and/or troubleshooting. . .*



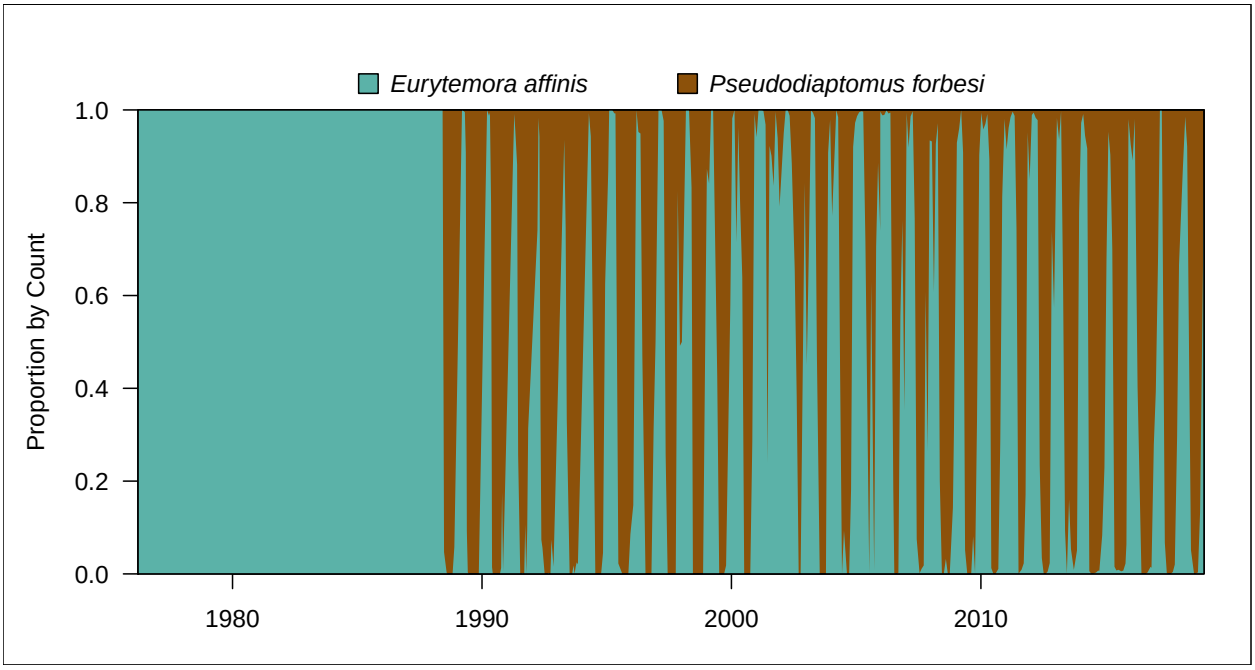
. . . Stacked Polygons continued . . .

Adding a legend to the inside of the plot region would be inappropriate for the figure above, so let's modify a few arguments in the ***legend( )*** function to put the legend outside of the plot region.

- ***legend( )***
  - **inset** = change this from a positive to a negative
  - **xpd** = an argument to specify whether plotting is constrained to the plot region; set to "xpd=NA" to override
  - **horiz** = whether to plot horizontally or vertically

We also need to add a little room to the top margin by altering ***par(****mar=c( )****)***.

```
par(mfrow=c(1,1), mar=c(3,4.5,3.5,1.5)+0.1)
.
.
.
legend("top", legend = c("Eurytemora affinis",
                         "Pseudodiaptomus forbesi"),
       pch=22, pt.cex=2, text.font=3,
       pt.bg = c(rgb(91,178,168,maxColorValue = 255),
                 rgb(140,81,10,maxColorValue = 255)),
       inset=-0.12, bty='n', xpd=NA, horiz=TRUE)
```



Plotting Multiple Panels

Let's create a second subset of the data for Suisun Bay and then plot a stacked barplot for Suisun Bay and the South Delta on the same plot

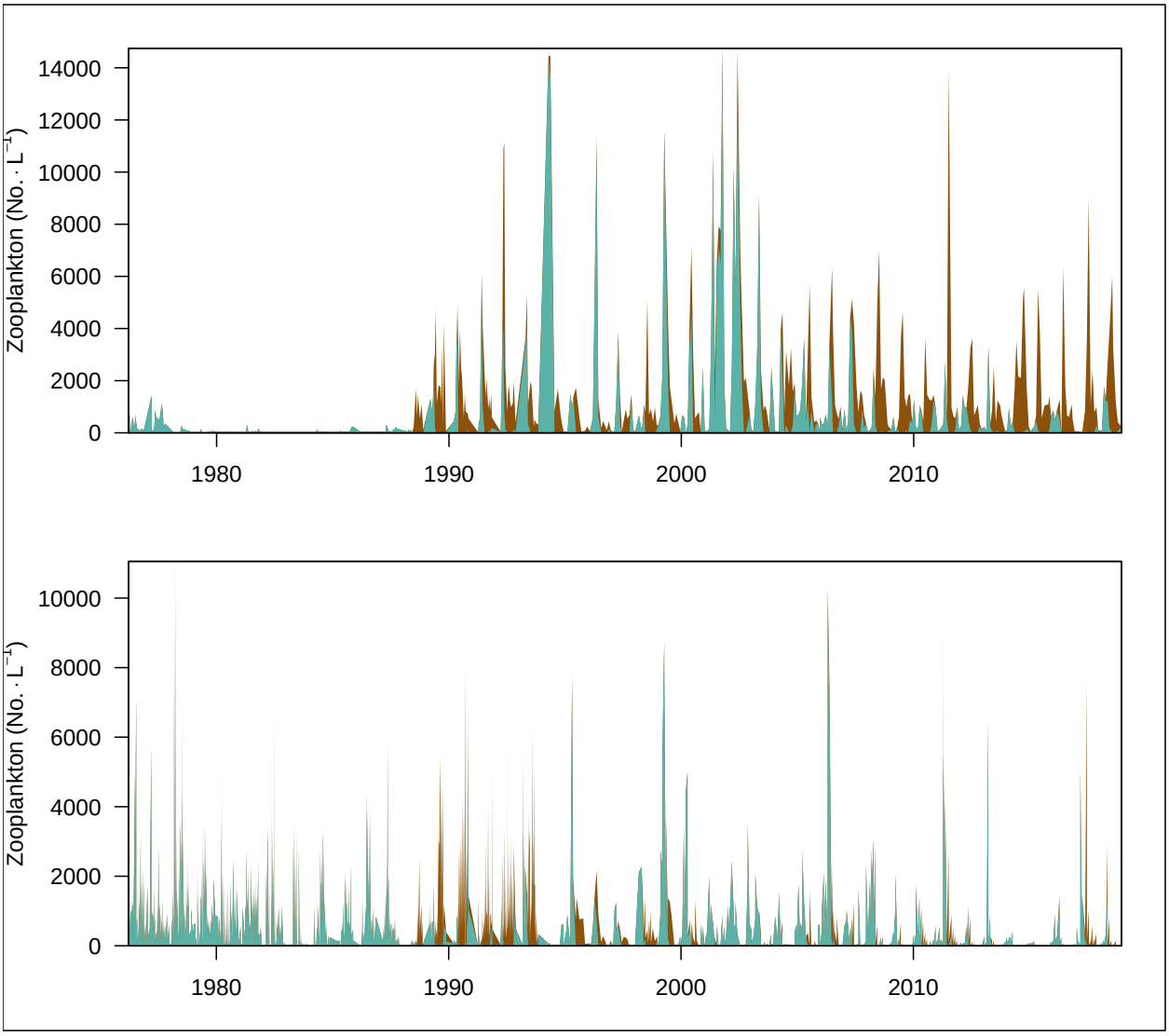
The main function to add multiple plots is *mfrow* argument within the ***par( )*** function:

- ***par( )***
  - **mfrow = c(nr , nc)**
    - *nr* = number of rows
    - *nc* = number of columns

```
suBay = subset(tdat, tdat$stratum=="Suisun Bay")
```

```
par(mfrow=c(2,1), mar=c(3,4.5,1.5,1.5)+0.1)
plot(x = range(as.Date(soDel$date, format = "%Y-%m-%d")),
     y = range(soDel$stacked),
     type='n', las=1, ylab="", xaxs='i', yaxs='i')
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$stacked,
              0,0,
              soDel$stacked[1]),
        col=rgb(140,81,10,maxColorValue = 255),  border=NA)
polygon(x = c(as.Date(soDel$date, format = "%Y-%m-%d"),
              max(as.Date(soDel$date, format = "%Y-%m-%d")),
              min(as.Date(soDel$date, format = "%Y-%m-%d")),
```

```
              min(as.Date(soDel$date, format = "%Y-%m-%d"))),
        y = c(soDel$Eurytemora.affinis,0,0,soDel$Eurytemora.affinis[1]),
        col=rgb(91,178,168,maxColorValue = 255),  border=NA)
mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.5)
box(which="plot")

plot(x = range(as.Date(suBay$date, format = "%Y-%m-%d")),
     y = range(suBay$stacked),
     type='n', las=1, ylab="", xaxs='i', yaxs='i')
polygon(x = c(as.Date(suBay$date, format = "%Y-%m-%d"),
              max(as.Date(suBay$date, format = "%Y-%m-%d")),
              min(as.Date(suBay$date, format = "%Y-%m-%d")),
              min(as.Date(suBay$date, format = "%Y-%m-%d"))),
        y = c(suBay$stacked,
              0,0,
              suBay$stacked[1]),
        col=rgb(140,81,10,maxColorValue = 255),  border=NA)
polygon(x = c(as.Date(suBay$date, format = "%Y-%m-%d"),
              max(as.Date(suBay$date, format = "%Y-%m-%d")),
              min(as.Date(suBay$date, format = "%Y-%m-%d")),
              min(as.Date(suBay$date, format = "%Y-%m-%d"))),
        y = c(suBay$Eurytemora.affinis,0,0,suBay$Eurytemora.affinis[1]),
        col=rgb(91,178,168,maxColorValue = 255),  border=NA)
mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.5)
box(which="plot")
box(which="outer")
```

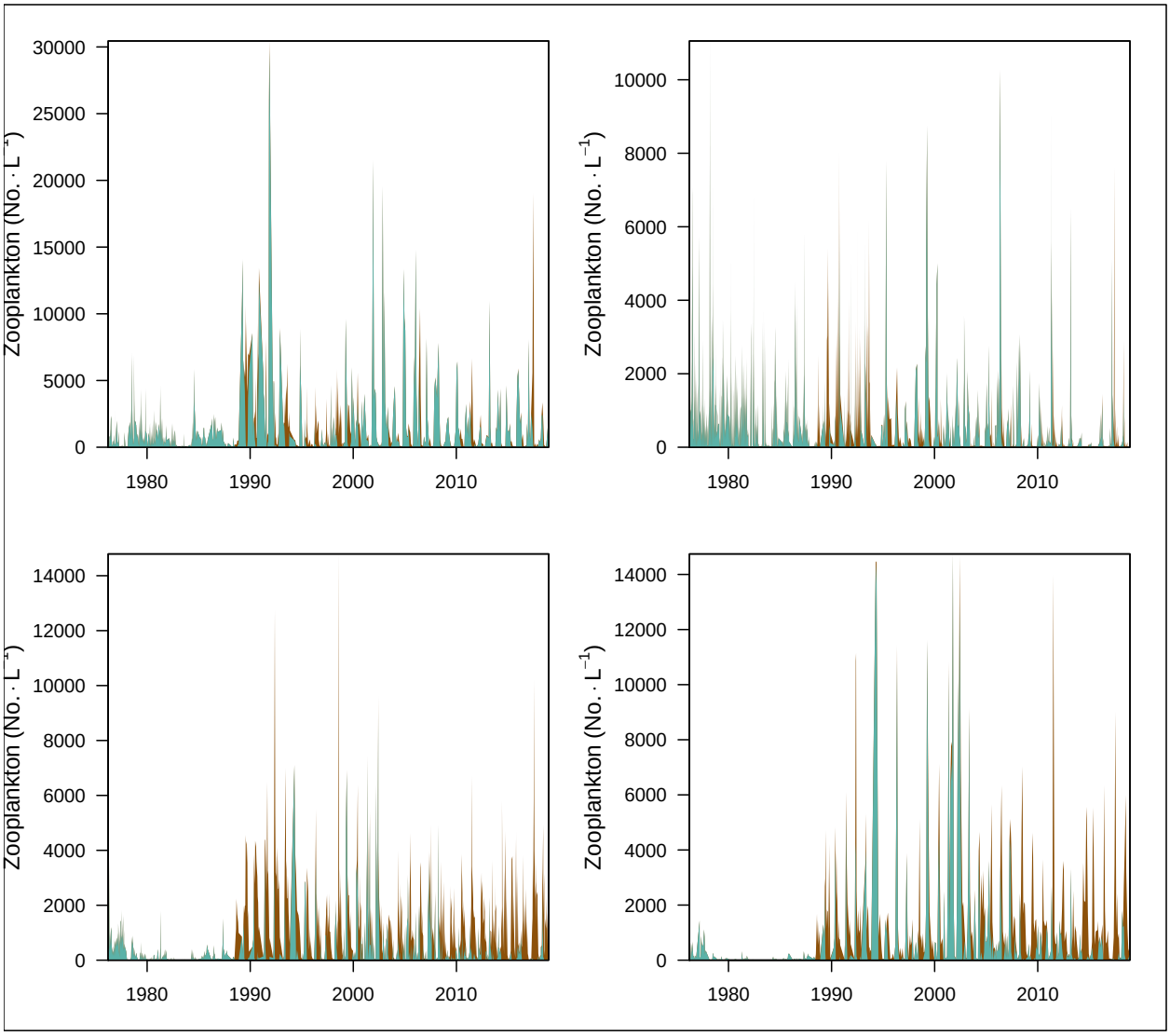


A more efficient option is to plot multiple plots with a loop. The loop involves two main components:

1. Subsetting the data to plot
2. The plot itself

```
subset_element_ids = c("Suisun Marsh",
                       "Suisun Bay",
                       "Lower Joaquin River",
                       "Southern Delta")
par(mfrow=c(2,2), mar=c(3,4.5,1.5,1.5)+0.1)

for(i in 1:length(subset_element_ids)){
  subdat = subset(tdat, tdat$stratum==subset_element_ids[i])

  plot(x = range(as.Date(subdat$date, format = "%Y-%m-%d")),
     y = range(subdat$stacked),
     type='n', las=1, ylab="", xaxs='i', yaxs='i')
  polygon(x = c(as.Date(subdat$date, format = "%Y-%m-%d"),
```

```
              max(as.Date(subdat$date, format = "%Y-%m-%d")),
              min(as.Date(subdat$date, format = "%Y-%m-%d")),
              min(as.Date(subdat$date, format = "%Y-%m-%d"))),
        y = c(subdat$stacked,
              0,0,
              subdat$stacked[1]),
        col=rgb(140,81,10,maxColorValue = 255),  border=NA)
  polygon(x = c(as.Date(subdat$date, format = "%Y-%m-%d"),
              max(as.Date(subdat$date, format = "%Y-%m-%d")),
              min(as.Date(subdat$date, format = "%Y-%m-%d")),
              min(as.Date(subdat$date, format = "%Y-%m-%d"))),
        y = c(subdat$Eurytemora.affinis,0,0,subdat$Eurytemora.affinis[1]),
        col=rgb(91,178,168,maxColorValue = 255),  border=NA)
  mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.5)
  box(which="plot")
}
box(which="outer")
```

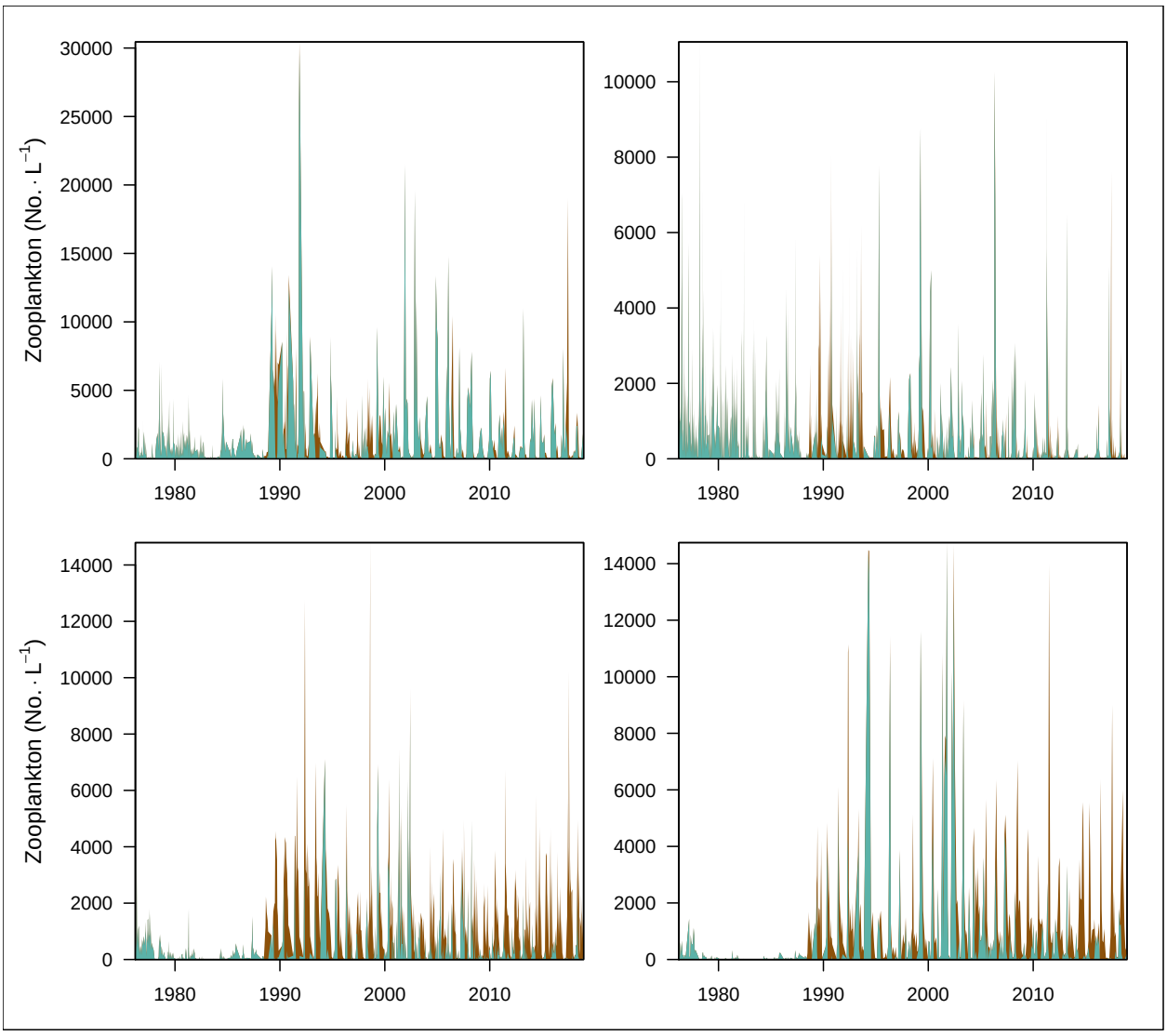


A few asethetic improvements include:

1. only having y-axis labels on the left panels
2. adding a panel label to the same spot in each plot
3. adding a legend above all the panels

Starting with the first improvement, we need to tell R to only add axis labels to the first and third panel. However, to maintain aesthetic integrity of gaps between panels, we need to alter the *mar* argument within the ***par( )*** function as well as the similiar argument *oma*, which sets the outer margin area analagous to *mar*.

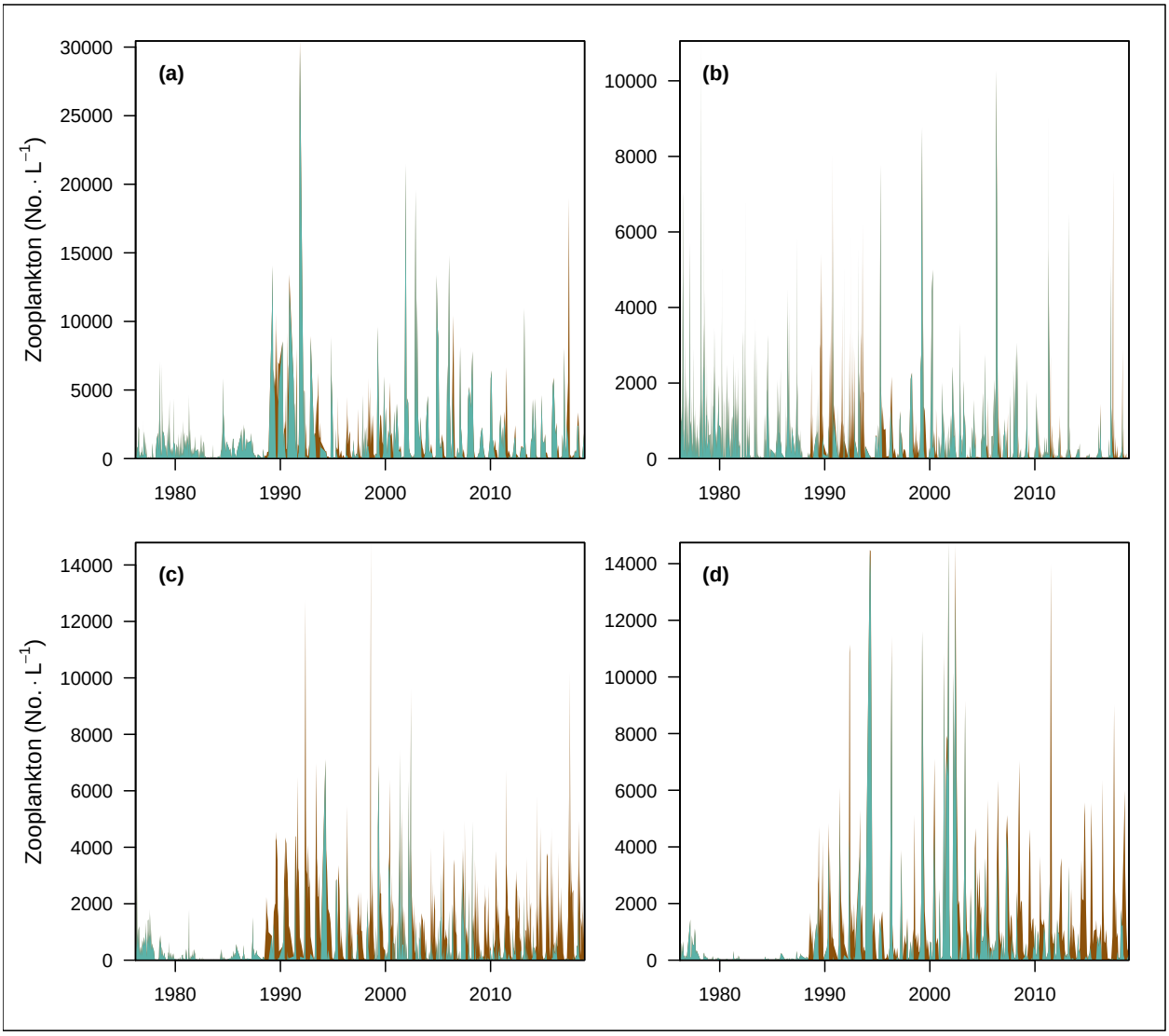
```
par(mfrow=c(1,1), mar=c(3,2.5,0.5,1.5)+0.1, oma=c(0,3.25,1,0))
  ...
for(i in 1:length(subset_element_ids)){
  ...
  if(i %in% c(1,3)){
      mtext(text = bquote(Zooplankton ~"("*No.%.%L^-1*")"),
      side=2, line=3.75)
  }
  ...
```



Next, let's add a plot label to each panel. Running ***par('usr')*** will give you the plot axes ranges. The ***plot_label( )*** function I create below allows you to set the same location for each panel.

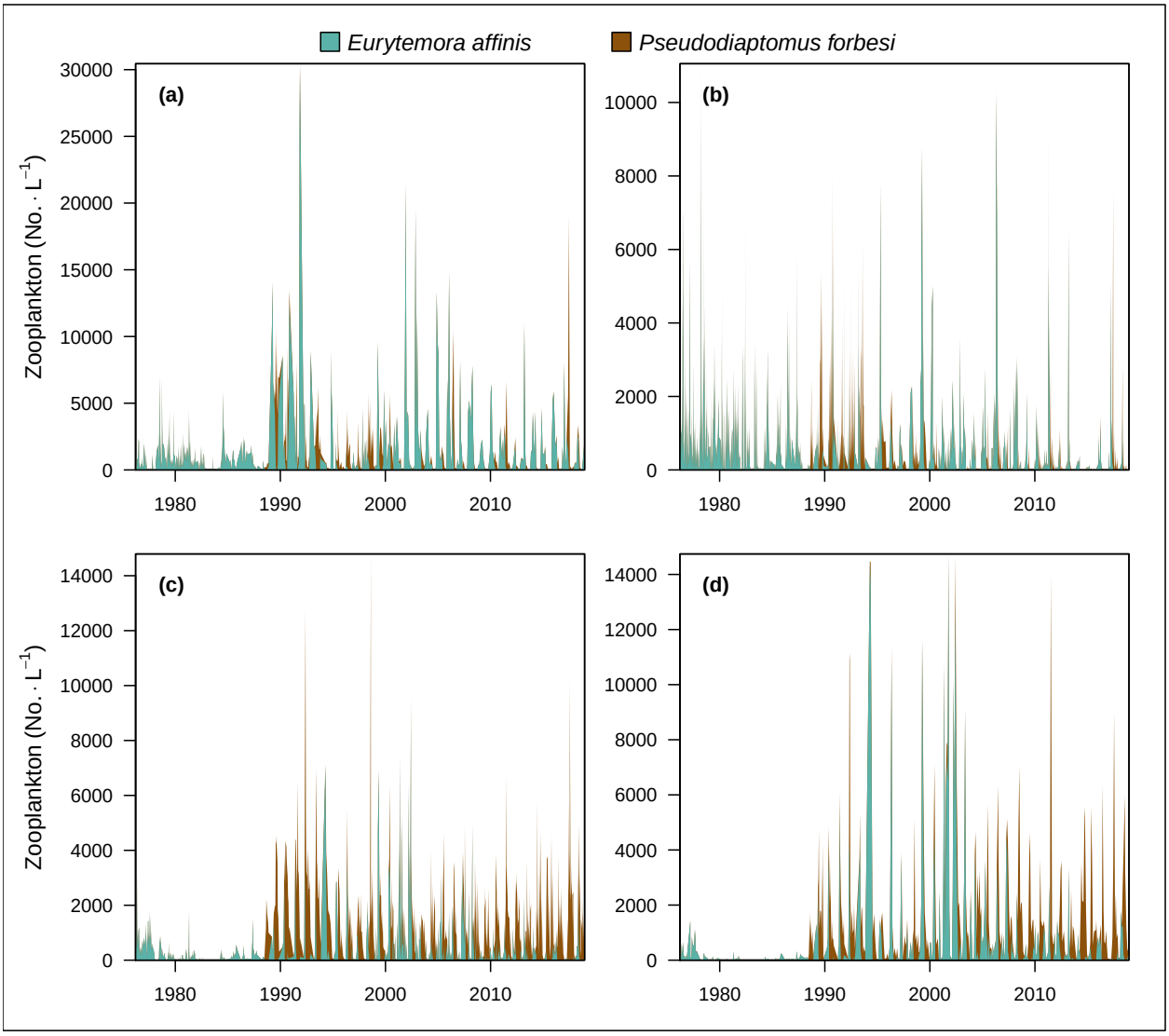
- ***plot_label( )***
  - *lab* = defualt is *(a)*; you can change for each panel
  - *x_prop* = default is 0.08; the label location as a proportion of the x-axis range
  - *y_prop* = default is 0.92; the label location as a proportion of the y-axis range
  - *font_type* = default is 2; 1=normal, 2=bold, 3=italics, 4=bold-italic

```
plot_label_vals = c("(a)", "(b)", "(c)", "(d)")

plot_label = function(lab="(a)", x_prop=0.08, y_prop=0.92,
                      font_type=2, fcex=1.15, usr=par('usr')){
  x_val = usr[1]+(usr[2]-usr[1])*x_prop
  y_val = usr[3]+(usr[4]-usr[3])*y_prop
  text(x = x_val, y = y_val, labels = lab, font = font_type, cex=fcex)
}

...

for(i in 1:length(subset_element_ids)){
  ...
  plot_label(lab = plot_label_vals[i] )
  ...
}
```



Finally, let's add a legend above all panels. Unfortunately, using the ***legend( )*** function after plotting will place a legend above the fourth panel plotted. We need to tell R to overlay a new, blank plot on the current plot and then add a legend above the new blank plot. Place the following code below the current plot script and then add a legend. However, you will need to add room using the *oma* argument in ***par( )***. *Note: if you set the* oma *and* mar *arguments to 0, you will need to adjust the* inset *argument from the legend.*

```
par(fig=c(0,1,0,1), oma=c(0,0,0,0), mar=c(0,0,0,0), new=TRUE)
legend("top", legend = c("Eurytemora affinis",
                         "Pseudodiaptomus forbesi"),
       pch=22, pt.cex=2.5, text.font=3, cex = 1.25,
       pt.bg = c(rgb(91,178,168,maxColorValue = 255),
                 rgb(140,81,10,maxColorValue = 255)),
       bty='n', xpd=NA, horiz=TRUE, inset=-0.0175)
```


Eurytemora affinis
Pseudodiaptomus forbesi
(a)
(b)
(c)
(d)
Zooplankton (No. · L^{-1})
Zooplankton (No. · L^{-1})
30000
25000
20000
15000
10000
5000
0
10000
8000
6000
4000
2000
0
14000
12000
10000
8000
6000
4000
2000
0
1980
1990
2000
2010

The ***layout( )*** function is an alternative to *mfrow* for non-symetrical multipanel layouts. For example:

```
par(mfrow=c(1,1), mar=c(3,2.5,0.5,1.5)+0.1, oma=c(0,3.25,2,0))
nf <- layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE), respect = FALSE)
layout.show(nf)
```