R Graphics Workshop Part II

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 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 second dataset:

```r
dat = read.csv(file = "zoop_df.csv", header=TRUE)
names(dat)
```

```
## [1] "log_meso_SoDel_cpue" "log_chl_SoDel"       "temp_EaDel"
```

The dataset has $\log_e$-transformed meso zooplankton CPUE in the southern Delta and two environmental variables that a random forest analyses identified as important predictors: $\log_e$-transformed chlorophyll in the southern Delta and water temperature in the eastern Delta.

```r
head(dat)
```

```
##   log_meso_SoDel_cpue log_chl_SoDel temp_EaDel
## 1            7.501418    0.18232156      14.00
## 2            7.501418    0.18232156      14.00
## 3            6.705469    0.00000000      12.20
## 4            6.016194    0.47000363      12.75
## 5            7.351559    0.09531018      15.00
## 6            7.351559    0.09531018      15.00
```

Visualizing Simple Linear Models

Let's develop a simple linear model of meso zooplankton CPUE ($\log_e$-transformed) in the southern Delta as a function of water temperature in the eastern Delta, chlorophyll ($\log_e$-transformed) in the southern Delta, and their interaction.

Mathematically, the model notation would be as follows:

$$\hat{y}_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{1i} \cdot X_{2i} + \varepsilon_i$$

$$\varepsilon_i \sim N(0, \sigma_\varepsilon^2)$$

```r
mod = lm(log_meso_SoDel_cpue ~ temp_EaDel*
     log_chl_SoDel, dat=dat)
summary(mod)
```

```
##
## Call:
## lm(formula = log_meso_SoDel_cpue ~ temp_EaDel * log_chl_SoDel,
##     data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.96699 -0.37789  0.03117  0.43385  1.72996
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               4.337875   0.120316  36.054  < 2e-16 ***
## temp_EaDel                0.180012   0.007063  25.488  < 2e-16 ***
## log_chl_SoDel             0.844929   0.130593   6.470 1.75e-10 ***
## temp_EaDel:log_chl_SoDel -0.022542   0.006959  -3.239  0.00125 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6948 on 767 degrees of freedom
## Multiple R-squared:  0.6806, Adjusted R-squared:  0.6794
## F-statistic: 544.8 on 3 and 767 DF,  p-value: < 2.2e-16
```

Based on the model output above, the fitted model is as follows:

$$\hat{y_i} = 4.337 + 0.180 \cdot Temp_i + 0.845 \cdot log_e Chl_i - 0.023 \cdot Temp_i \cdot log_e Chl_i + \varepsilon_i$$

$$\varepsilon_i \sim N(0, 0.695^2)$$

Interpreting models with an interaction term can be challenging. The best way to understand an interaction term is to visualize the model output. Specifically, simulate across one predictor variable while holding the second predictor variable constant. Let's give this a try. The process involves five steps:

1. generate a sequence across the range of the first variable
2. identify 1-3 values of the second variable to hold constant
3. extract model coefficients (the β values in the equations above)
4. insert the simulated data into the model equation above (in the place of X_n)
5. visualize the model simulations

Let's start with the first step and generate a sequence of values across the observed range of $\log_e$ Chlorophyll:

```r
sim_breaks=100
log_chl_sim = seq(from=min(dat$log_chl_SoDel),
                  to=max(dat$log_chl_SoDel),
                  length.out=sim_breaks)
log_chl_sim
```

```
##   [1] -1.660731207 -1.603287913 -1.545844618 -1.488401324 -1.430958029
##   [6] -1.373514735 -1.316071441 -1.258628146 -1.201184852 -1.143741557
##  [11] -1.086298263 -1.028854969 -0.971411674 -0.913968380 -0.856525085
##  [16] -0.799081791 -0.741638497 -0.684195202 -0.626751908 -0.569308613
##  [21] -0.511865319 -0.454422025 -0.396978730 -0.339535436 -0.282092141
##  [26] -0.224648847 -0.167205552 -0.109762258 -0.052318964  0.005124331
##  [31]  0.062567625  0.120010920  0.177454214  0.234897508  0.292340803
##  [36]  0.349784097  0.407227392  0.464670686  0.522113980  0.579557275
##  [41]  0.637000569  0.694443864  0.751887158  0.809330452  0.866773747
##  [46]  0.924217041  0.981660336  1.039103630  1.096546924  1.153990219
##  [51]  1.211433513  1.268876808  1.326320102  1.383763396  1.441206691
##  [56]  1.498649985  1.556093280  1.613536574  1.670979868  1.728423163
##  [61]  1.785866457  1.843309752  1.900753046  1.958196340  2.015639635
##  [66]  2.073082929  2.130526224  2.187969518  2.245412812  2.302856107
##  [71]  2.360299401  2.417742696  2.475185990  2.532629284  2.590072579
##  [76]  2.647515873  2.704959168  2.762402462  2.819845757  2.877289051
##  [81]  2.934732345  2.992175640  3.049618934  3.107062229  3.164505523
##  [86]  3.221948817  3.279392112  3.336835406  3.394278701  3.451721995
##  [91]  3.509165289  3.566608584  3.624051878  3.681495173  3.738938467
##  [96]  3.796381761  3.853825056  3.911268350  3.968711645  4.026154939
```

We should start simple and simulate a single temperature value. The median temperature value is a good starting point:

```r
med_temp = median(dat$temp_EaDel)
```

Use the ***coefficients( )*** or ***coef( )*** function to extract the model coefficients:

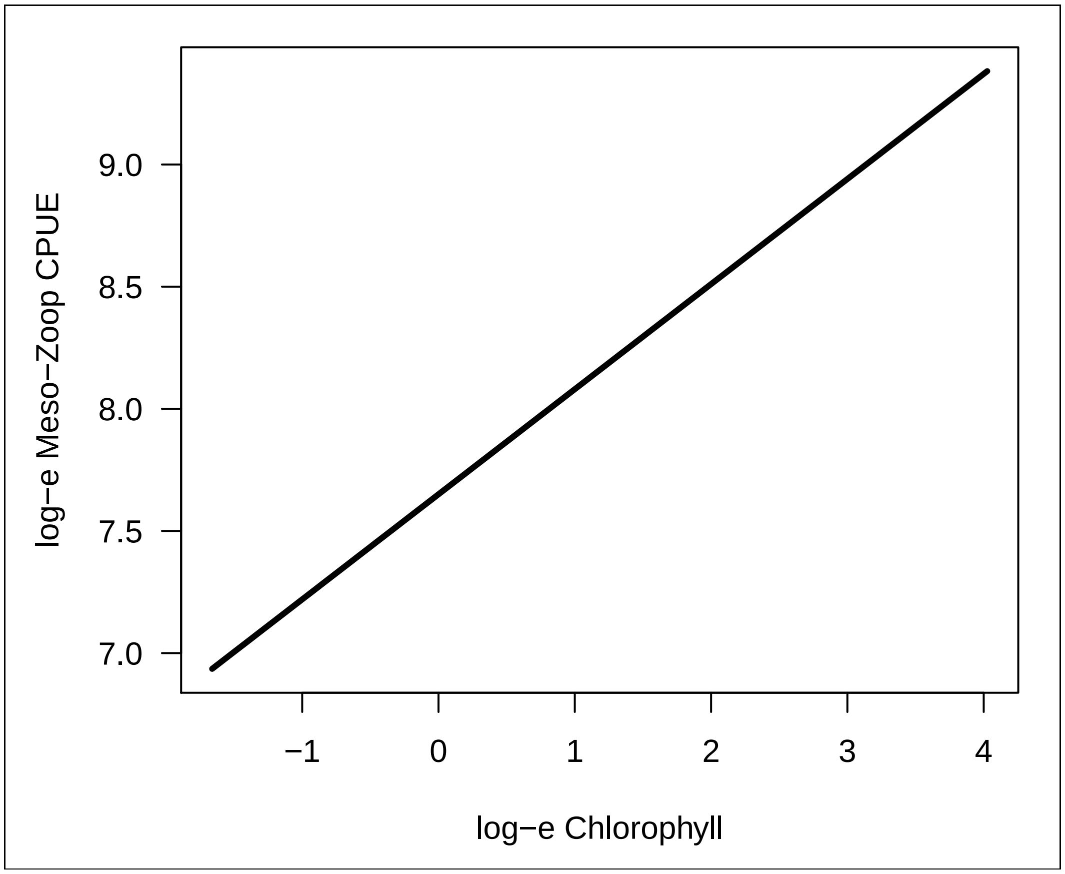
```r
beta_0 = as.numeric(coef(mod)[1])
beta = as.numeric(coef(mod)[2:4])
```

Insert the appropriate pieces of code into the equation structure:

```r
med_temp_pred = (beta_0 +
                  beta[1] * med_temp +
                  beta[2] * log_chl_sim +
                  beta[3] * (med_temp*log_chl_sim))
```

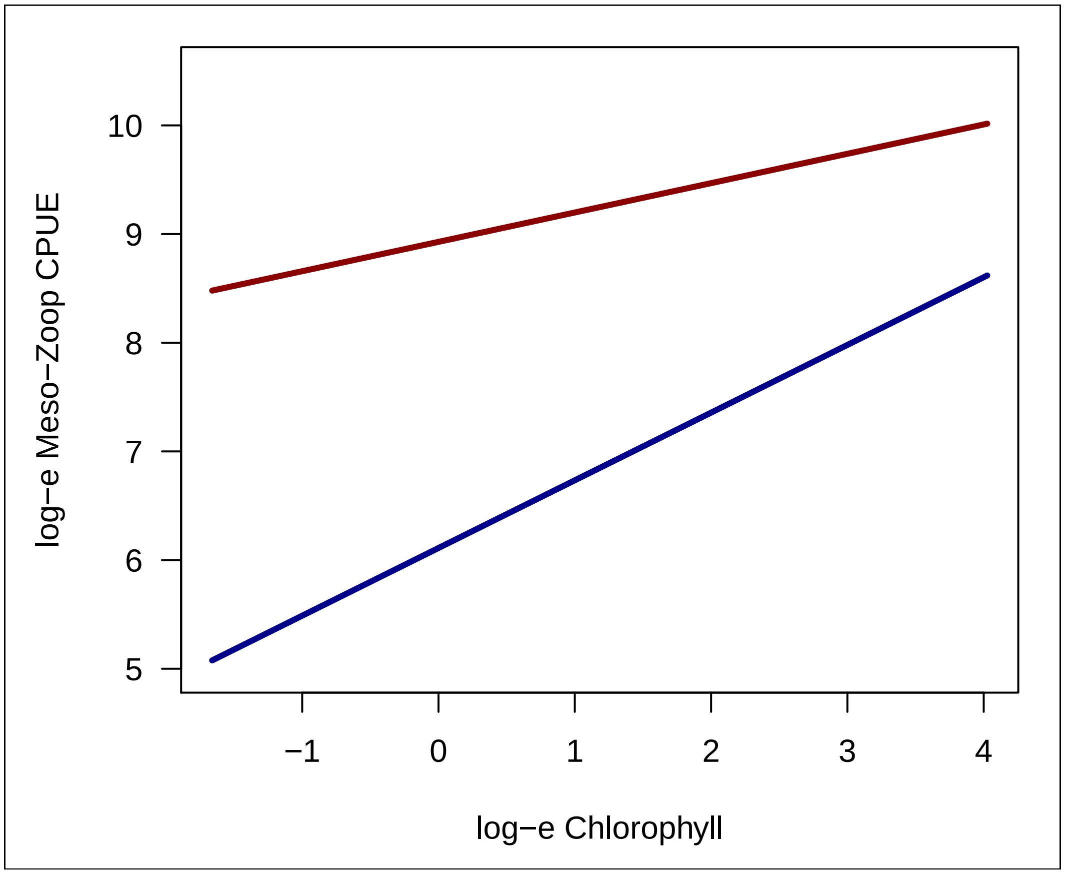
And now, we visualize:

```r
par(mfrow=c(1,1), mar=c(4.5, 4.5, 1, 1)+0.1)
plot(med_temp_pred ~ log_chl_sim,
     type='l', ylab="log-e Meso-Zoop CPUE", xlab="log-e Chlorophyll", las=1,
     col="black", lwd=3)
box(which="outer")
```



To wrap our heads around the interaction, we should simulate and visualize the model output at or near the temperature extremes. I recommend using the 5th and 95th or 10th and 90th percentiles of observed temperatures. You can calculate the percentiles using the ***quantile( )*** function.

```
temp_quants = quantile(dat$temp_EaDel, probs = c(0.05,0.95))
low_temp_pred = (beta_0 +
                  beta[1] * temp_quants[1] +
                  beta[2] * log_chl_sim +
                  beta[3] * (temp_quants[1]*log_chl_sim))
high_temp_pred  = (beta_0 +
                   beta[1] * temp_quants[2] +
                   beta[2] * log_chl_sim +
                   beta[3] * (temp_quants[2]*log_chl_sim))

par(mfrow=c(1,1), mar=c(4.5, 4.5, 1, 1)+0.1)
plot(low_temp_pred ~ log_chl_sim,
     type='l', ylab="log-e Meso-Zoop CPUE", xlab="log-e Chlorophyll", las=1,
     ylim = c(5,10.5), col="darkblue", lwd=3)
lines(high_temp_pred ~ log_chl_sim,
      lty=1, col="darkred", lwd=3)
box(which="outer")
```



We can clearly see that the slope decreases with temperature but the intercept increases. This pattern is the interaction term at play.

Visualizing Model Uncertainty

When simulating an interaction, making statements about significance can be challenging. For example, $\log_e$ meso zooplankton CPUE at cold temperatures may be significantly lower than at high temperatures, but only when $\log_e$ chlorophyll is low. Consequently, visualizing model uncertainty is critical. We can acheive this a number of ways, but let's simulate model uncertainty using the ***Effects( )*** function from the *'effects'* package.

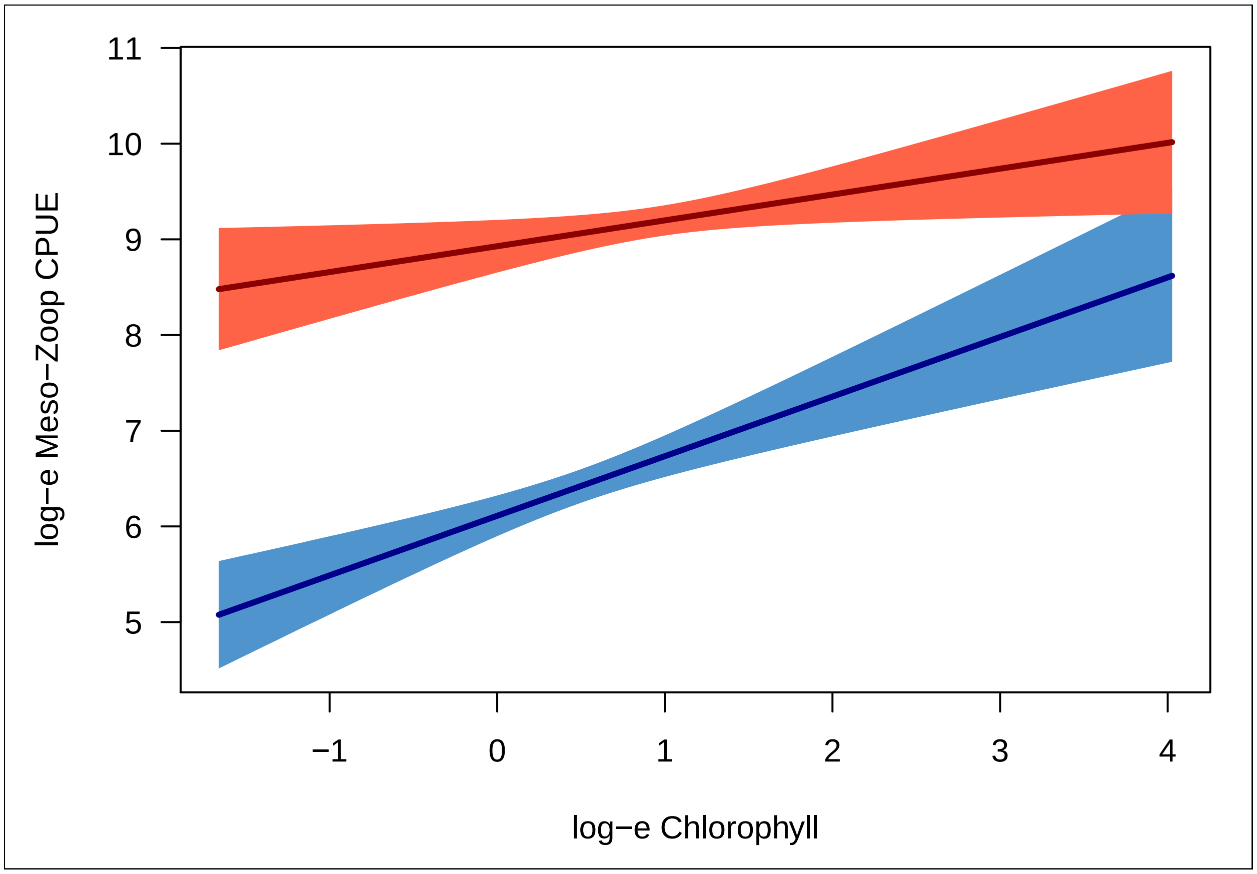
- ***Effects( )***
  - *mod* = the name of the model to simulate
  - *focal.predictors* = a character vector of one or more predictors in the model
  - *xlevels* = a list of values to be used in the simulation
  - *se* = list of elements defining how model uncertainty is determined
    - *type* = algorithm used to calculate uncertainty
      - "pointwise" is the default (AKA Tukey)
      - "Scheffe" is more conservative (AKA simultaneous)
    - *level* = the confidence level of the confidence interval

```r
library('effects')
chl_eff = Effect(mod=mod, focal.predictors = c("log_chl_SoDel" , "temp_EaDel"),
                 xlevels=list(log_chl_SoDel = c(log_chl_sim),
                              temp_EaDel = c(temp_quants[[1]],
                                             temp_quants[[2]])
                              ), se=list(type="Scheffe", level=0.99))
head(data.frame(chl_eff))
tail(data.frame(chl_eff))
```

```
## Loading required package: carData

## lattice theme set by effectsTheme()
## See ?effectsTheme for details.

##   log_chl_SoDel temp_EaDel      fit        se    lower    upper
## 1     -1.660731       9.85 5.076539 0.1534814 4.515235 5.637844
## 2     -1.603288       9.85 5.112320 0.1497764 4.564566 5.660075
## 3     -1.545845       9.85 5.148101 0.1460818 4.613859 5.682344
## 4     -1.488401       9.85 5.183882 0.1423984 4.663110 5.704654
## 5     -1.430958       9.85 5.219663 0.1387272 4.712317 5.727009
## 6     -1.373515       9.85 5.255444 0.1350692 4.761476 5.749412

##     log_chl_SoDel temp_EaDel       fit        se    lower    upper
## 195      3.738938       25.5  9.938114 0.1857253 9.258889 10.61734
## 196      3.796382       25.5  9.953630 0.1893475 9.261158 10.64610
## 197      3.853825       25.5  9.969146 0.1929735 9.263414 10.67488
## 198      3.911268       25.5  9.984662 0.1966031 9.265656 10.70367
## 199      3.968712       25.5 10.000178 0.2002361 9.267885 10.73247
## 200      4.026155       25.5 10.015694 0.2038724 9.270103 10.76129
```

NOTE: the model output is a running list of both temperature values (i.e., the model output has 200 rows, one for each sequence). So we can break the output into two tables and then visualize using the ***polygon( )*** function as illustrated in Part I of this workshop.

```r
cold_chl_eff = data.frame(chl_eff)[1:100,]
hot_chl_eff = data.frame(chl_eff)[101:200,]

par(mfrow=c(1,1), mar=c(4.5, 4.5, 1, 1)+0.1)
plot(range(cold_chl_eff$lower, hot_chl_eff$upper) ~ range(log_chl_sim),
     ylab="log-e Meso-Zoop CPUE", xlab="log-e Chlorophyll", las=1,
     col="darkblue", lwd=3, type='n')

polygon(x = c(log_chl_sim, log_chl_sim[100],
              log_chl_sim[100:1], log_chl_sim[1]),
        y = c(cold_chl_eff$lower[1:100], cold_chl_eff$upper[100],
              cold_chl_eff$upper[100:1], cold_chl_eff$lower[1]),
        border=NA, col = "steelblue3")
lines(cold_chl_eff$fit[1:100] ~ log_chl_sim,
      lty=1, col="darkblue", lwd=3)

polygon(x = c(log_chl_sim, log_chl_sim[100],
              log_chl_sim[100:1], log_chl_sim[1]),
        y = c(hot_chl_eff$lower[1:100], hot_chl_eff$upper[100],
              hot_chl_eff$upper[100:1], hot_chl_eff$lower[1]),
        border=NA, col = "tomato1")
lines(hot_chl_eff$fit[1:100] ~ log_chl_sim,
      lty=1, col="darkred", lwd=3)
box(which="outer")
```



Now let's add some aesthetics. Here are the additions I would make:

1. Add second, back transformed x- and y-axes
2. Use the ***bquote( )*** function to add correctly notated axis labels
3. Add a legend identifying explaining the value of the lines (the figure caption should also explain the line and polygon)
4. Use transparent polygons to highlight the overlapping region at high chlorophyll values

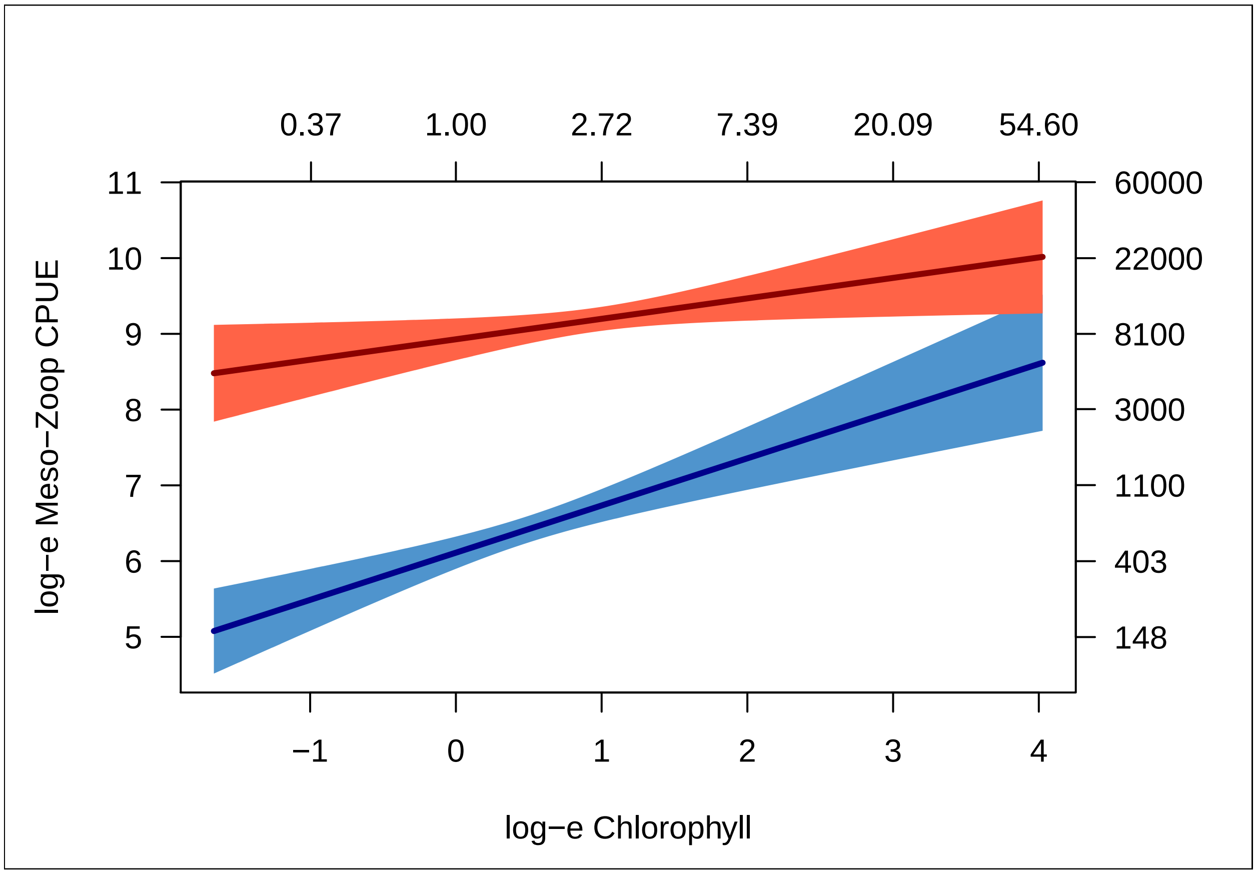
1. Back-transformed axes

Exercise

Add back-transformed x- and y-axes to the figure above (refer to the third section of Part I as a reference).

Hint: you can use the following code to help you create labels with the same significant digits:

```
axis(side = 3, at = xlab_at, labels = sprintf("%.2f", xlab_lab), las=1)
```

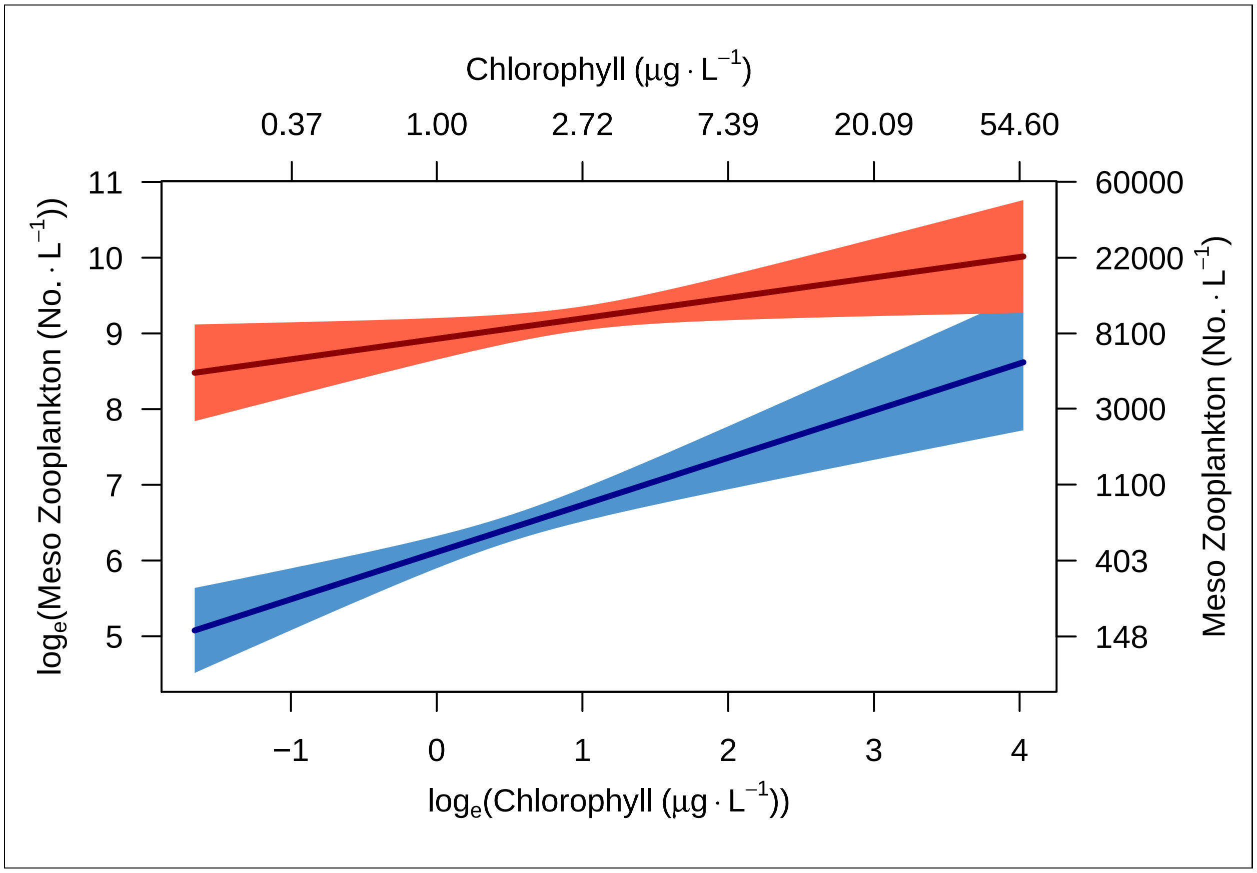


2. Axis labels

As in Part I, we will use the ***bquote( )*** function as an argument within the ***mtext( )*** function to add appropriate axis labels.

- ***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
    - 'mu' = Greek symbol
    - 'degree' = degree symbol
    - *see **help(plotmath)** for more details and examples*

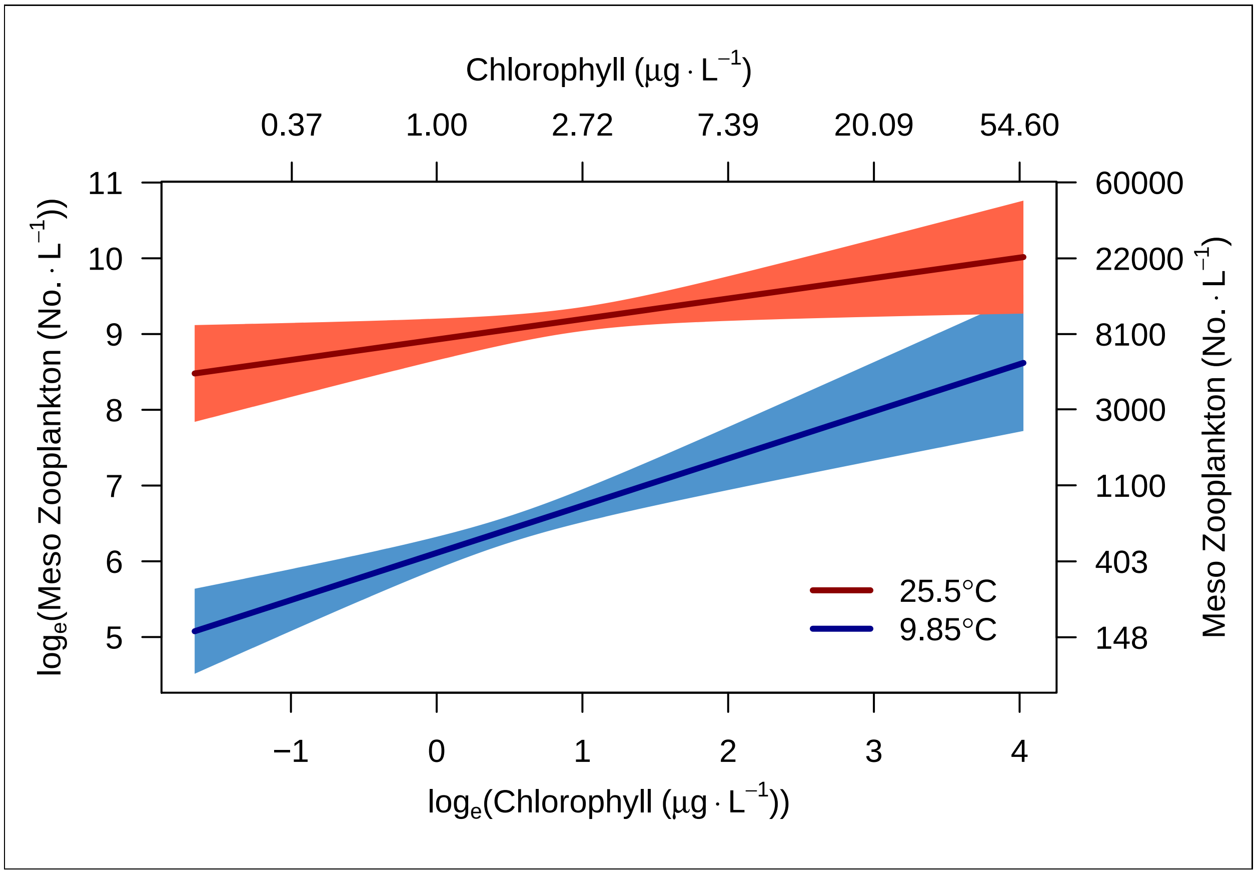
```
mtext(text = bquote(log[e]*"("*Chlorophyll~"("*mu*g%.%L^-1*")"*")"),
      side = 1, line=2.25)
mtext(text = bquote(Chlorophyll~"("*mu*g%.%L^-1*")"),
      side = 3, line=2.25)
mtext(text = bquote(log[e]*"("*Meso~Zooplankton~"("*No.%.%L^-1*")"*")"),
      side = 2, line=2.5)
mtext(text = bquote(Meso~Zooplankton~"("*No.%.%L^-1*")"),
      side = 4,line = 3.75, las=0)
```



3. Add a legend

```
legend("bottomright",
       legend = c(as.expression(bquote(25.50*degree*C)),
                  as.expression(bquote(9.85*degree*C))),
       cex = 1, bty='n', xpd=NA, horiz=FALSE, inset=0.05,
       lty=1, lwd=3, col=c("darkred", "darkblue"))
```

NOTE: we have to use ***as.expression( )*** to convert the bquote style into legend text

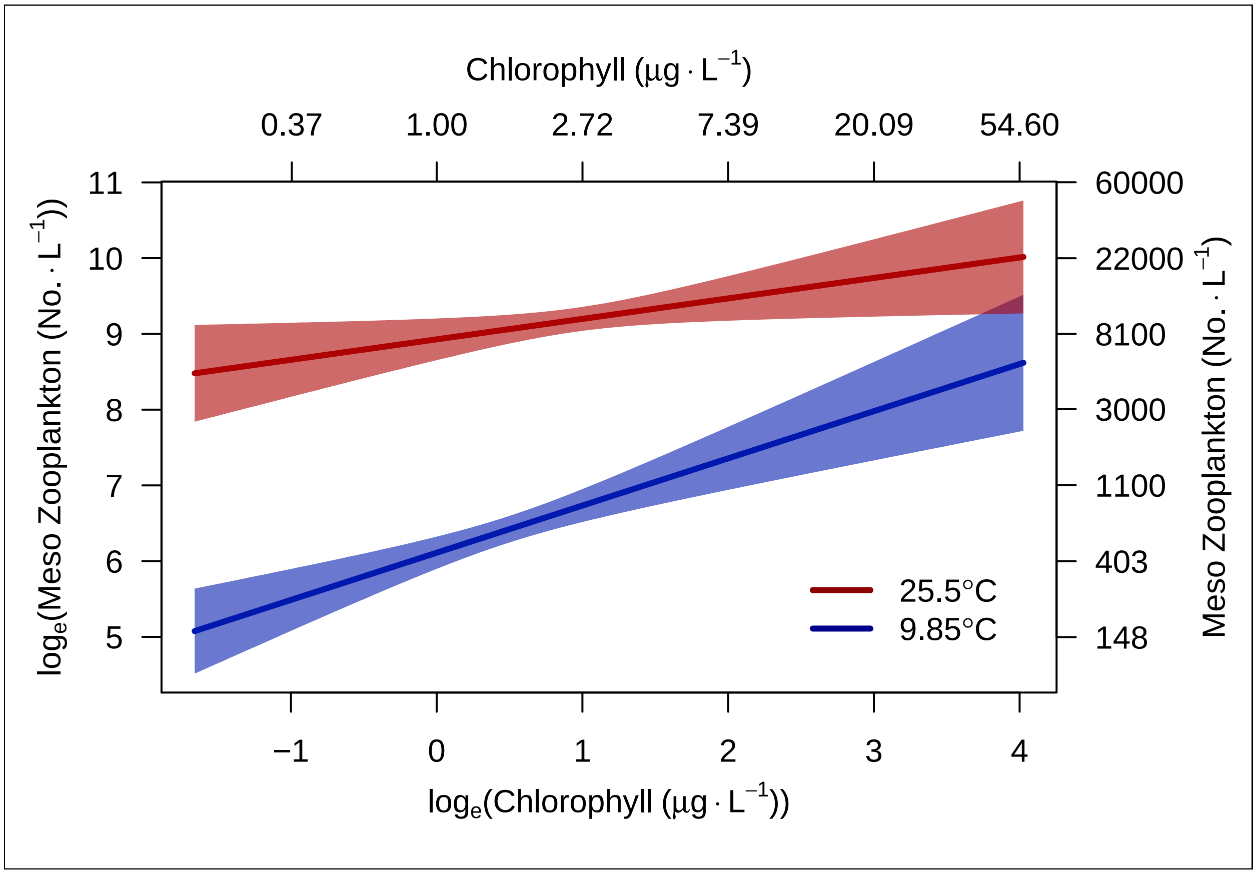


4. Plotting transparencies

Two useful functions for plotting gray-scale and colored transparent elements are ***gray( )*** and ***rgb( )***, respectively. The key argument for both functions is *alpha*. First use https://www.colorspire.com/rgb-color-wheel/ to identify rgb-values of the color of your choice. Then explore how adjusting the *alpha* argument changes element transparency.

Let's start by reducing the *alpha* argument within the ***rgb( )*** function to plot transparent polygons. *Note*: refer to Part I for a description of the ***rgb( )*** function.

```
polygon(x = c(log_chl_sim, log_chl_sim[100],
              log_chl_sim[100:1], log_chl_sim[1]),
        y = c(hot_chl_eff$lower[1:100], hot_chl_eff$upper[100],
              hot_chl_eff$upper[100:1], hot_chl_eff$lower[1]),
        border=NA, col = rgb(174,1,1,maxColorValue = 255, alpha=150))
lines(hot_chl_eff$fit[1:100] ~ log_chl_sim,
      lty=1, col=rgb(174,1,1,maxColorValue = 255), lwd=3)
```



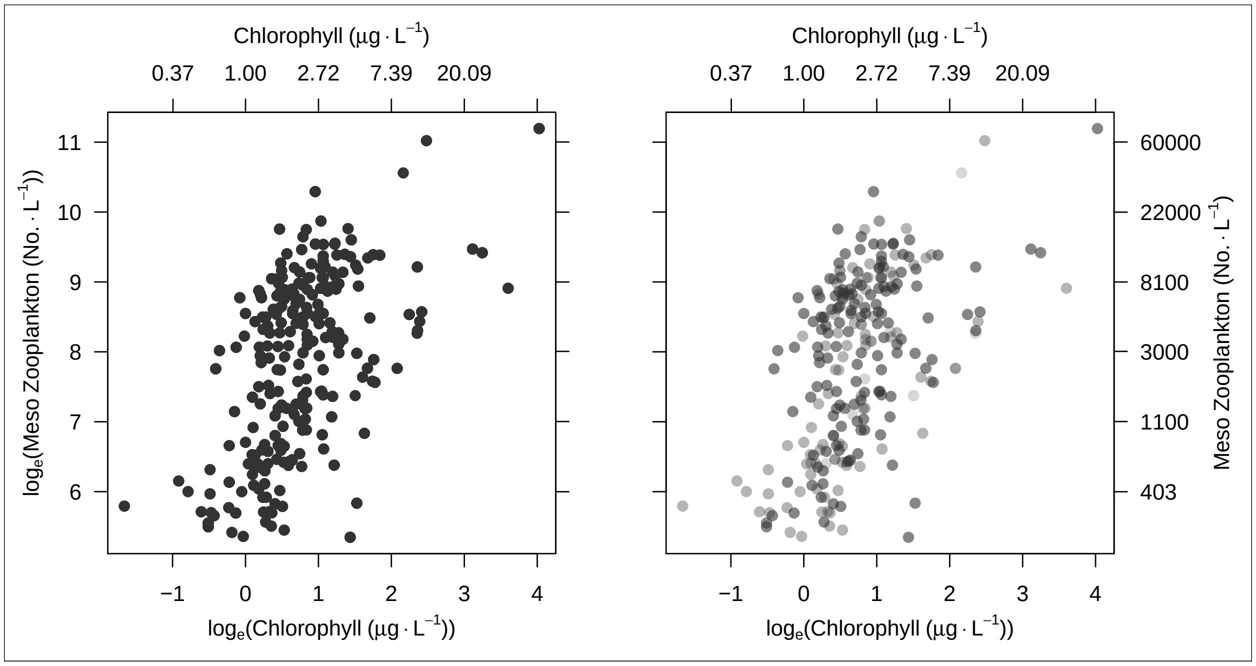
Plotting a high number of observations can be challenging as many observations are similar, resulted in certain values being visually underrepresented as they are plotted on top of one another or resulting in a data cloud that looks like a one imensional data shadow for datasets with numerous values. However, we can use the ***gray( )*** function to help reduce or eliminate these common plotting issues.

- ***gray( )***
  - *level* = value 0:1; where 0=black, 1=white
  - *alpha* = transparency 0:1; 0=transparent, 1=opaque

I usually start with *gray(level=0.2, alpha=0.2)*. Let's take a look:

```
par(mfrow=c(1,2), mar=c(4,1.85,4,2.15)+0.1, oma=c(0,2,0,3))
plot(log_meso_SoDel_cpue ~ log_chl_SoDel, data=dat,
     pch=19, col=gray(level = 0.2),
     ylab="", xlab="", las=1)

plot(log_meso_SoDel_cpue ~ log_chl_SoDel, data=dat,
     pch=19, col=gray(level = 0.2, alpha = 0.25),
     ylab="", xlab="", las=1)
```

When we are not using transparent points (left panel), we see six observations in the upper right. However, when we use transparent points (right panel), we see that there are actually >10 observations with many of them having identical or very similar values for zooplankton and chlorophyll.



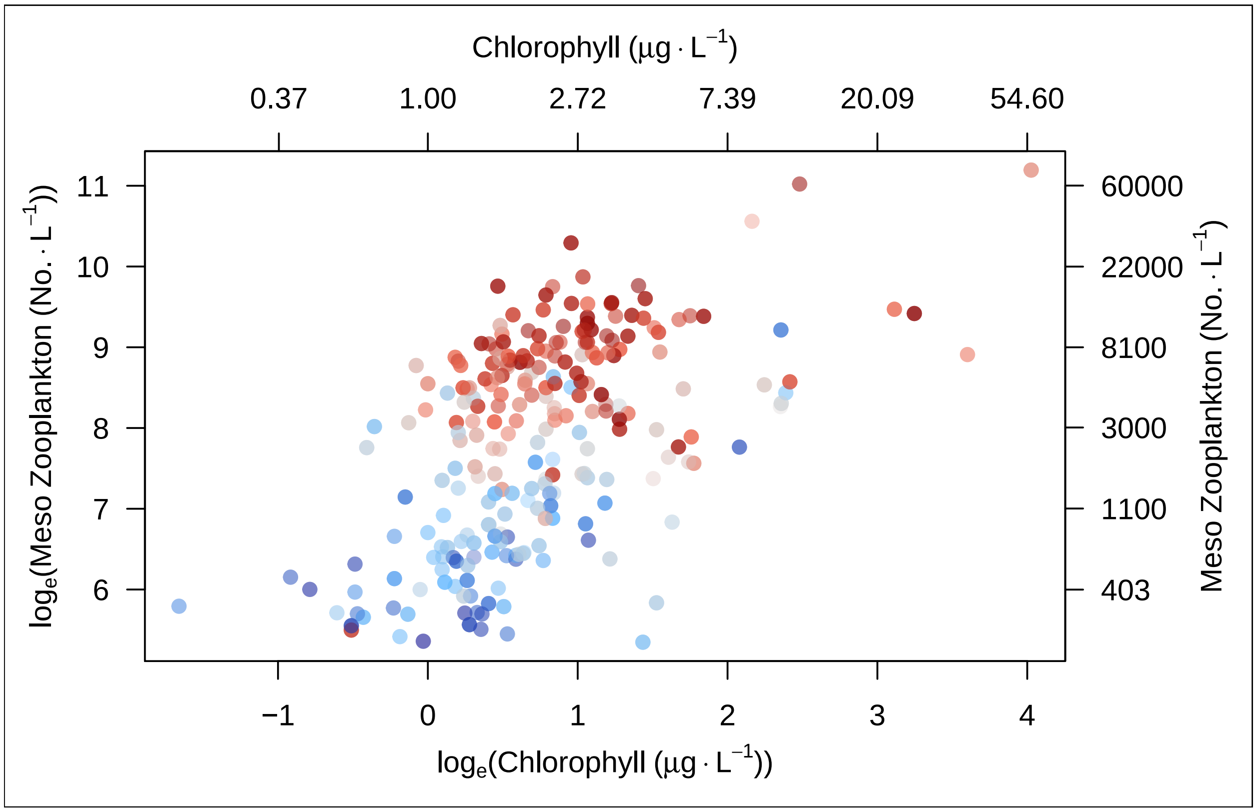
We can add the third dimension of the raw data, temperature in the eastern Delta, to the plot by taking advantage of the ***colorRampPalette( )***, ***col2rbg( )***, and ***rbg( )*** functions. To do so, take the following steps outlined in the code below:

1. Create a custom palette of the colors you want for temperature using a vector of color names in the ***colorRampPalette( )*** function. I chose to transition from darkblue to gray to darkred as temperatures move from cold to hot.
2. Using the palette you create, extract 101 colors. (*Note*: I use 101 colors in this example as we can convert each observation into a percent of total temperature range from 0-100%)).
3. Unfortunately, the custom palette outputs colors in hexadecimal format, which is not suitable for the ***rgb( )*** function. Use ***col2rgb( )*** to create a matrix of rgb-convertent colors.
4. Convert each temperature observation into a percentage of the temperature range using the following simple equation:

$$Temp\ Range\ (\%) = \left(\frac{Observed\ Temp - min(Observed\ Temp)}{max(Observed\ Temp) - min(Observed\ Temp)}\right) \cdot 100$$

   we can then round the percentage of the temperature range observations to the nearest integer and add 1 to match the rgb color matrix (a matrix with 101 rows). We add one because the percentages range from 0 to 100 while the rgb color matrix rows range from 1 to 101.
5. Finally, we use the rounded percentage of the temperature range (plus 1) as an index when calling from the rgb matrix for each observation in the *col* argument of the ***plot( )*** function and specify the transparency using the *alpha* argument in the ***rgb( )*** function.

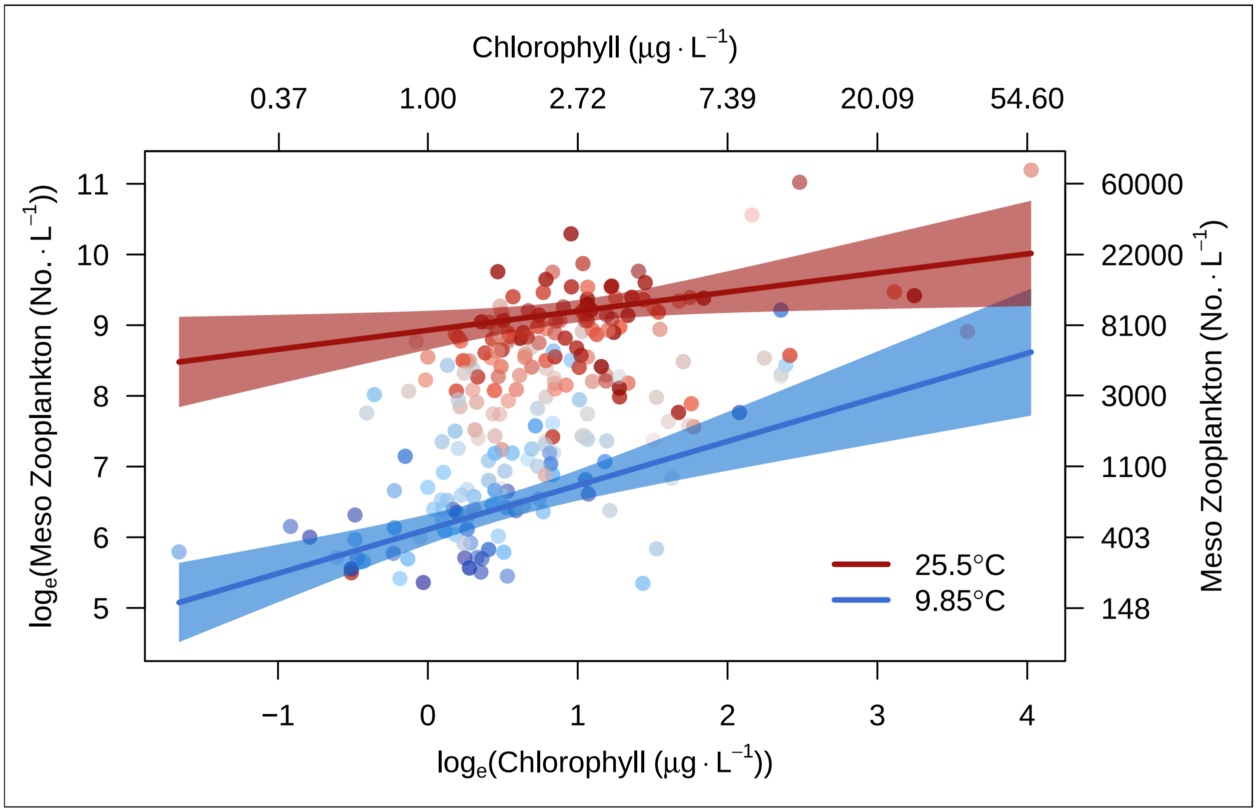
```r
col_range = colorRampPalette(c("darkblue", "steelblue1", "gray85", "tomato2", "darkred"))
col_pal = col_range(101)
rgb_cols=col2rgb(col_pal)

perctage_temp = ((dat$temp_EaDel-min(dat$temp_EaDel))/diff(range(dat$temp_EaDel)))*100
perc_temp = round(perctage_temp) + 1

par(mfrow=c(1,1), mar=c(4,3.85,4,5.15)+0.1)
plot(log_meso_SoDel_cpue ~ log_chl_SoDel, data=dat,
     pch=19, ylab="", xlab="", las=1,
     col = rgb(red=c(rgb_cols[1,perc_temp]),
               green=c(rgb_cols[2,perc_temp]),
               blue=c(rgb_cols[3,perc_temp]),
               maxColorValue=255,
               alpha=255/3))
```



We can overlay the color-scaled observations on the model simulations (with their respective confidence intervals) to convey information about both the observations and the predictions.

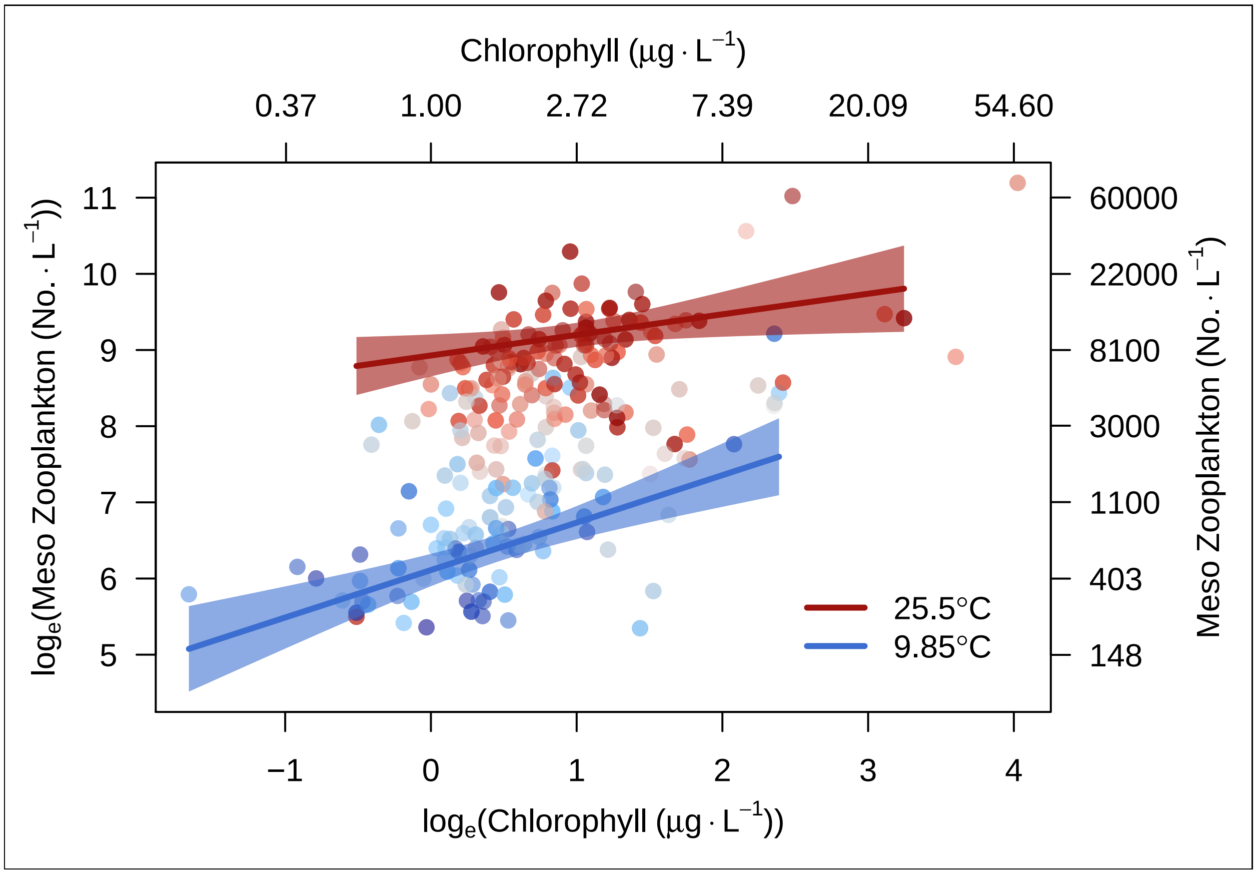
Note: be sure to change the line and polygon to the correct color to match the 5th and 95th percentile of temperatures (9.85°C and 25.50°C, respectively).

```
cold_val=which(abs(9.85-dat$temp_EaDel) == min(abs(9.85-dat$temp_EaDel)))[1]
cold_perc=perc_temp[cold_val]
hot_val=which(abs(25.5-dat$temp_EaDel) == min(abs(25.5-dat$temp_EaDel)))[1]
hot_perc=perc_temp[hot_val]
...
polygon(x = c(log_chl_sim, log_chl_sim[100],
              log_chl_sim[100:1], log_chl_sim[1]),
        y = c(cold_chl_eff$lower[1:100], cold_chl_eff$upper[100],
              cold_chl_eff$upper[100:1], cold_chl_eff$lower[1]),
        border=NA, col = rgb(red=c(rgb_cols[1,5]),
               green=c(rgb_cols[2,cold_perc]),
               blue=c(rgb_cols[3,cold_perc]),
               maxColorValue=255, alpha=150))
lines(cold_chl_eff$fit[1:100] ~ log_chl_sim,
      lty=1, col=rgb(red=c(rgb_cols[1,cold_perc]),
               green=c(rgb_cols[2,cold_perc]),
               blue=c(rgb_cols[3,cold_perc]),
               maxColorValue=255), lwd=3)
...
```



While the previous figure conveys a lot of information, the model simulations are misleading. When comparing the simulated ranges and observed range of chlorophyll at a given temperature, you will notice we appear to be extrapolating (i.e., we do not see high temperature values when chlorophyll is very low nor do we see low temperature values when chlorophyll is high). I recommend constraining the simulated chlorophyll range to the range observed under moderate to high conditions (red points) for the 95th percentile of temperature simulation and constrain similarly under cold temperatures. However, very few, if any, observations are at the exact temperatures we are simulating. So, you need to make a judgement call; just be sure to clearly describe your process in the figure caption and, if appropriate, the methods section. For our example, I will simulate across the range of chlorophyll observed from the minimum through the 25th percentile of temperature for the cold simulation and from the 75th percentile to the maximum temperature for the hot simulation:

```
temp_quartile=quantile(dat$temp_EaDel, probs=c(0.25, 0.75))

cold_log_chl = range(dat$log_chl_SoDel[which(dat$temp_EaDel<=temp_quartile[1])])
hot_log_chl = range(dat$log_chl_SoDel[which(dat$temp_EaDel>=temp_quartile[2])])
cold_log_chl_sim = seq(from=cold_log_chl[1], to=cold_log_chl[2], length.out=100)
hot_log_chl_sim = seq(from=hot_log_chl[1], to=hot_log_chl[2], length.out=100)
cold_chl_eff2 = Effect(mod=mod, focal.predictors = c("log_chl_SoDel" , "temp_EaDel"),
                      xlevels=list(log_chl_SoDel = c(cold_log_chl_sim),
                                   temp_EaDel = c(temp_quants)
                      ), se=list(type="Scheffe", level=0.99))
hot_chl_eff2 = Effect(mod=mod, focal.predictors = c("log_chl_SoDel" , "temp_EaDel"),
                     xlevels=list(log_chl_SoDel = c(hot_log_chl_sim),
                                  temp_EaDel = rev(temp_quants)
                     ), se=list(type="Scheffe", level=0.99))
```



Adding a Color Scale Bar to a Plot

The previous figure is missing only one piece of information that cannot be clearly articulated in the caption: the water temperatures associated with the plotted colors. The code below is a custom function to convert your color palette into a scale bar with an axis that ranges from 0 to 1.

```
color.bar <- function(lut, min, max=-min, nticks=11,
                      ticks=seq(min, max, len=nticks),
                      title='') {
  scale = (length(lut))/(max-min)
  plot(c(min,max), c(0,1), type='n', bty='n',
       xaxt='n', xlab='', yaxt='n', ylab='',
       main=title)
  for (i in 1:(length(lut))) {
    x = (i-1)/scale + min
    rect(0,x, 1, x+1/scale, col=lut[i], border=NA)
  }
}
```

To add a color scale bar to a plot we need to take the following steps:

1. Convert desired dimension of data (eastern Delta water temperature in this case) into equidistance values across the observed range for scale bar axis label values.

2. Convert those axis label values into a proportion from 0 to 1 to match the scale bar range of 0 to 1 using the following simple equation:

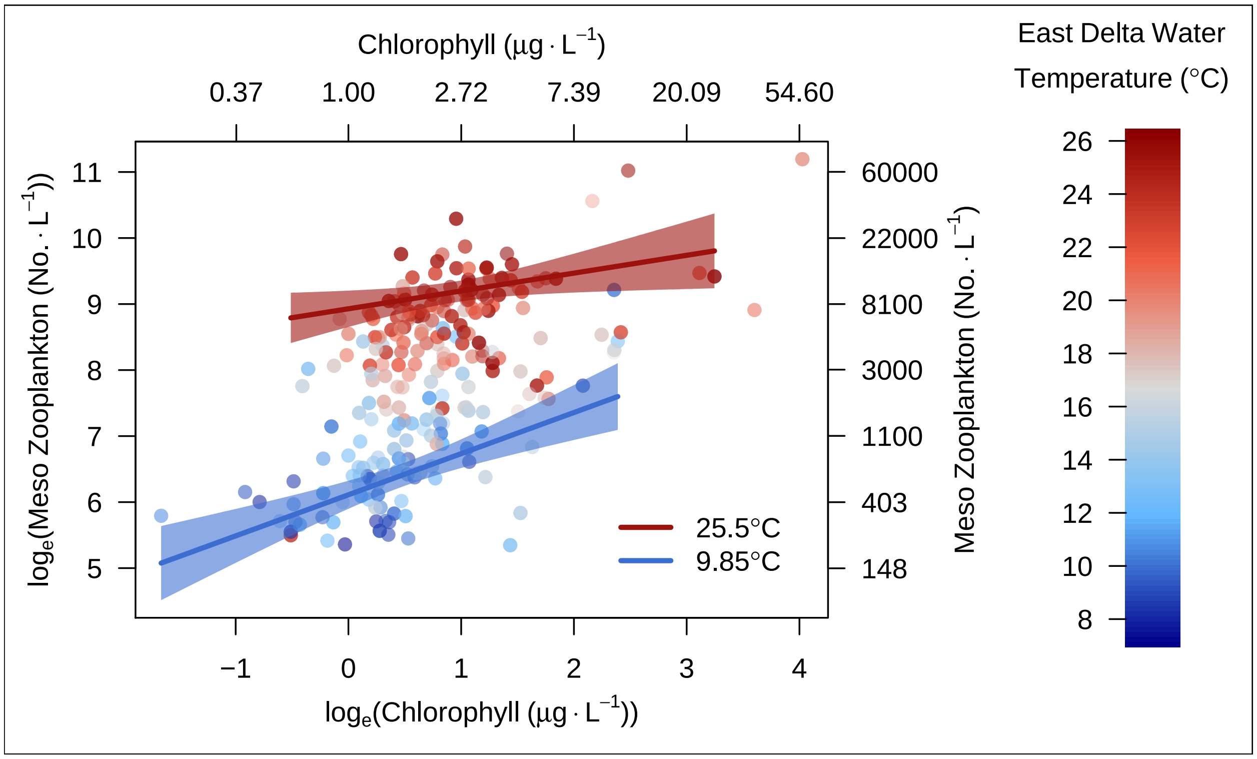
$$Temp\ Range\ Proportion = \left(\frac{Scale\ Bar\ Axis\ Label\ Values - min(Observed\ Temp)}{max(Observed\ Temp) - min(Observed\ Temp)}\right)$$

3. Run the previous plot, but use the *oma* argument in the ***par( )*** function to add room for the scale bar to the right of the plot. *Note*: you may want to increase the plot window width.
4. As with adding a legend outside of a multipanel figure in Part I, generate a blank plotting canvas using the *new=TRUE* argument in the ***par( )*** function. Similar to step 3 above, add space to the left of the plotting window using the *oma* argument to add room for the original plot. Not adding enough room will cause the scale bar to overlay on the original plot. Adding too much space will prevent R from plotting. *Note*: you usually have to add a rediculous amount of right margin space not to overlay the scale bar on the plot, but this is simply a process of trial and error.
5. Use the custom ***color.bar( )*** function with following argument values:

   - *lut* = the name of your color palette ('col_pal' in our case)
   - *min* = the minimum proportion value ('0' in our case)
   - *max* = the maximum proportion value ('1' in our case)

6. Add the appropriate scale bar axis labels.

```r
scale_labs = seq(from = 8, to = 26, by=2)
scale_ats = ((scale_labs - min(dat$temp_EaDel))/diff(range(dat$temp_EaDel)))

par(mfrow=c(1,1), mar=c(4,3.85,4,5.15)+0.1, oma=c(0,0,0,7.5))

# insert plot here

par(fig=c(0,1,0,1), mar=c(4,3.85,4,5.15)+0.1, oma=c(0,29.5,0,0), new=TRUE)
color.bar(col_pal, min=0, max=1)
axis(2,at=scale_ats, labels=scale_labs, lwd=0, lwd.ticks=1, line=-0.1, cex.axis=1, las=1)
mtext(text = bquote(atop(East~"Delta"~Water,Temperature*degree*C)),
      side = 3,line = 0.25, adj=0.7, las=0)
```



Formatting and Exporting High Quality Figures

Base-R has several plot exporting functions such as ***jpeg( )*** and ***tiff( )***; however, these functions have shortcomings when attemping to export a specificed dpi. For example, changing the *res* [resolution] argument in the ***tiff( )*** function will not produce higher quality images (at least not on Macs, I have not attempted on a PC). Consequently, ***pdf( )*** is my preferred plot exporting function. You can open the pdf in a photo editing/viewing software and save in the format and resolution you desire.

- ***pdf( )***
  - *file* = "plot.pdf"
  - *width* = graphic width in inches
  - *height* = graphic height in inches
  - *family* = font family; Helvetica is the default

The plot will write to your working directory. Once you have run all of the appropriate code for your plot, run >*dev.off( )* to close the pdf file.

Note: You can test for the appropriate width and height of your plot using the ***windows( )*** function on a PC and the ***quartz( )*** function on a Mac. This will allow you to assess text, point, and line sizes prior to writing to pdf.

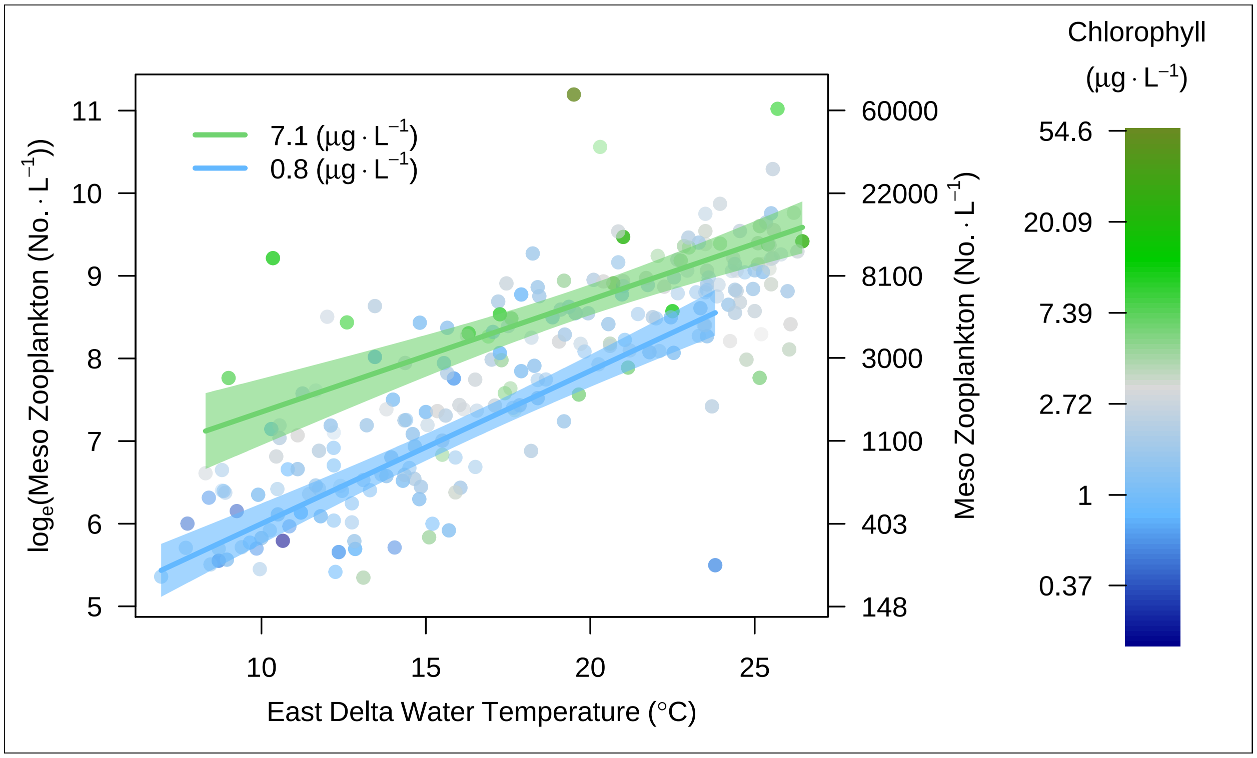
I prefer to comment out (using '#') the ***pdf( )***, ***quartz( )***, ***windows( )***, and ***dev.off( )*** lines of code so as to not accidentally overwrite pdfs or unintentionally close graphics.

```
getwd()
#pdf(file = "test.pdf", width = 6, height=3.5)
#quartz(width = 6, height=3.5)
#windows(width = 6, height=3.5)
```

```
par(mfrow=c(1,2), mar=c(4.5,4.5,3.5,4.5)+0.1)
plot(log_meso_SoDel_cpue ~ log_chl_SoDel, data=dat,
     pch=19, ylab="", xlab="", las=1)

plot(log_meso_SoDel_cpue ~ log_chl_SoDel, data=dat,
     pch=19, col=gray(level = 0.25, alpha = 0.25),
     ylab="", xlab="", las=1)
#dev.off()
```

Bonus Exercise

Generate the figure above, but with eastern Delta temperature as the primary predictor variable (x-axis) and chlorophyll as the third dimension.


Chlorophyll
(µg · L⁻¹)
7.1 (µg · L⁻¹)
0.8 (µg · L⁻¹)
logₑ(Meso Zooplankton (No. · L⁻¹))
Meso Zooplankton (No. · L⁻¹)
East Delta Water Temperature (°C)