3. Worksheet: Basic R

Rich Hull; Z620: Quantitative Biodiversity, Indiana University

23 March, 2021

OVERVIEW

This worksheet introduces some of the basic features of the R computing environment (http://www.r-project.org). It is designed to be used along side the **3. RStudio** handout in your binder. You will not be able to complete the exercises without the corresponding handout.

Directions:

1. In the Markdown version of this document in your cloned repo, change "Student Name" on line 3 (above) with your name.
2. Complete as much of the worksheet as possible during class.
3. Use the handout as a guide; it contains a more complete description of data sets along with examples of proper scripting needed to carry out the exercises.
4. Answer questions in the worksheet. Space for your answers is provided in this document and is indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio (color may vary if you changed the editor theme).
5. Before you leave the classroom today, it is *imperative* that you **push** this file to your GitHub repo, at whatever stage you are. Ths will enable you to pull your work onto your own computer.
6. When you have completed the worksheet, **Knit** the text and code into a single PDF file by pressing the `Knit` button in the RStudio scripting panel. This will save the PDF output in your '3.RStudio' folder.
7. After Knitting, please submit the worksheet by making a **push** to your GitHub repo and then create a **pull request** via GitHub. Your pull request should include this file (**3.RStudio_Worksheet.Rmd**) with all code blocks filled out and questions answered) and the PDF output of `Knitr` (**3.RStudio_Worksheet.pdf**).

The completed exercise is due on **Wednesday, March 24th, 2021 before 12:00 PM (noon)**.

1) HOW WE WILL BE USING R AND OTHER TOOLS

You are working in an RMarkdown (.Rmd) file. This allows you to integrate text and R code into a single document. There are two major features to this document: 1) Markdown formatted text and 2) "chunks" of R code. Anything in an R code chunk will be interpreted by R when you *Knit* the document.

When you are done, you will *knit* your document together. However, if there are errors in the R code contained in your Markdown document, you will not be able to knit a PDF file. If this happens, you will need to review your code, locate the source of the error(s), and make the appropriate changes. Even if you are able to knit without issue, you should review the knitted document for correctness and completeness before you submit the Worksheet. Next to the `Knit` button in the RStudio scripting panel there is a spell checker button (`ABC`) button.

2) SETTING YOUR WORKING DIRECTORY

In the R code chunk below, please provide the code to: 1) clear your R environment, 2) print your current working directory, and 3) set your working directory to your '3.RStudio' folder.

```
# Clear environment:
rm(list=ls())
# Print working directory
getwd()
```

```
## [1] "C:/Users/Rich Hull/GitHub/QB2021_Hull/2.Worksheets/3.RStudio"
```

```
# Set working directory
setwd("C:/Users/Rich Hull/GitHub/QB2021_Hull")
```

3) USING R AS A CALCULATOR

To follow up on the pre-class exercises, please calculate the following in the R code chunk below. Feel free to reference the **1. Introduction to version control and computing tools** handout.

1) the volume of a cube with length, l, = 5 (volume = l^3)
2) the area of a circle with radius, r, = 2 (area = pi * r^2).
3) the length of the opposite side of a right-triangle given that the angle, theta, = pi/4. (radians, a.k.a. 45°) and with hypotenuse length sqrt(2) (remember: sin(theta) = opposite/hypotenuse).
4) the log (base e) of your favorite number.

```
# Calculate volume of cube
vc <- 5^3
print(vc)
```

```
## [1] 125
```

```
# Calculate area of circle
ac <- pi * 2^2
print(ac)
```

```
## [1] 12.56637
```

```
# Calculate length of opp side of right-triangle
lo <- sin(pi/4) * sqrt(2)
print(lo)
```

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

```
# Calculate log (base e) of fav number (7)
lg <- log(7)
print(lg)
```

```
## [1] 1.94591
```

4) WORKING WITH VECTORS

To follow up on the pre-class exercises, please perform the requested operations in the R-code chunks below.

Basic Features Of Vectors

In the R-code chunk below, do the following: 1) Create a vector `x` consisting of any five numbers. 2) Create a new vector `w` by multiplying `x` by 14 (i.e., "scalar"). 3) Add `x` and `w` and divide by 15.

```
# Create vector x
x <- c(1, 2, 3, 4, 5)
print(x)
```

```
## [1] 1 2 3 4 5
```

```
# Create vector w
w <- x * 14
print(w)
```

```
## [1] 14 28 42 56 70
```

```
# Add x and w and divide by 15
wx2 <- (w + x) / 15
print(wx2)
```

```
## [1] 1 2 3 4 5
```

Now, do the following: 1) Create another vector (`k`) that is the same length as `w`. 2) Multiply `k` by `x`. 3) Use the combine function to create one more vector, `d` that consists of any three elements from `w` and any four elements of `k`.

```
# Create k
k <- c(10,15,20,25,30)
print(k)
```

```
## [1] 10 15 20 25 30
```

```
# Multiply k by x
kx <- k * x
print(kx)
```

```
## [1]  10  30  60 100 150
```

```
# Combine
d <- w[c(1:3)]
d[4:7] <- k[c(1:4)]
print(d)
```

```
## [1] 14 28 42 10 15 20 25
```

Summary Statistics of Vectors

In the R-code chunk below, calculate the **summary statistics** (i.e., maximum, minimum, sum, mean, median, variance, standard deviation, and standard error of the mean) for the vector (v) provided.

```
v <- c(16.4, 16.0, 10.1, 16.8, 20.5, NA, 20.2, 13.1, 24.8, 20.2, 25.0, 20.5, 30.5, 31.4, 27.1)

# Max
maxv <- max(na.omit(v))
print(maxv)
```

```
## [1] 31.4
```

```
# Min
minv <- min(na.omit(v))
print(minv)
```

```
## [1] 10.1
```

```
# Sum
sumv <- sum(na.omit(v))
print(sumv)
```

```
## [1] 292.6
```

```
# Mean
meanv <- mean(na.omit(v))
print(meanv)
```

```
## [1] 20.9
```

```
# Median
medianv <- median(na.omit(v))
print(medianv)
```

```
## [1] 20.35
```

```
# Variance
varv <- var(na.omit(v))
print(varv)
```

```
## [1] 39.44
```

```
# SD
sdv <- sd(na.omit(v))
print(sdv)
```

```
## [1] 6.280127
```

```r
# SE of mean
sem <- function(x){
  sd(na.omit(x))/sqrt(length(na.omit(x)))
}
semv <- sem(v)
print(semv)
```

```
## [1] 1.678435
```

5) WORKING WITH MATRICES

In the R-code chunk below, do the following: Using a mixture of Approach 1 and 2 from the **3. RStudio** handout, create a matrix with two columns and five rows. Both columns should consist of random numbers. Make the mean of the first column equal to 8 with a standard deviation of 2 and the mean of the second column equal to 25 with a standard deviation of 10.

```r
# Make first matrix
j <- c(rnorm(5, mean = 8, sd = 2))
# Make second matrix
z <- c(rnorm(5, mean = 25, sd = 10))
# Combine matrices
k2 <- cbind(j, z)
print(k2)
```

```
##             j         z
## [1,] 7.784896 21.593494
## [2,] 7.274336 18.544983
## [3,] 8.158319 34.879470
## [4,] 8.011252 25.903325
## [5,] 6.307416  9.485453
```

Question 1: What does the `rnorm` function do? What do the arguments in this function specify? Remember to use `help()` or type `?rnorm`.

> Answer 1: The function 'rnorm' generates a normal distribution of values that are equal to a specified mean and standard deviation.

In the R code chunk below, do the following: 1) Load `matrix.txt` from the **3.RStudio** data folder as matrix `m`. 2) Transpose this matrix. 3) Determine the dimensions of the transposed matrix.

```r
# Load matrix
m <- as.matrix(read.table("data/matrix.txt", sep = "\t", header = FALSE))
print(m)
```

```
##       V1 V2 V3 V4 V5
##  [1,]  8  1  7  6  1
##  [2,]  5  5  2  4  1
##  [3,]  2  5  4  3  3
##  [4,]  3  2  5  1  4
##  [5,]  9  9  1  1  2
```

```
##  [6,] 11  8  1  8  8
##  [7,]  2  2  5  8  5
##  [8,]  3  3  6  7  6
##  [9,]  5  5  1  3  6
## [10,]  6  5  9  2  2
```

```
# Transpose matrix
n <- t(m)
print(n)
```

```
##    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## V1    8    5    2    3    9   11    2    3    5     6
## V2    1    5    5    2    9    8    2    3    5     5
## V3    7    2    4    5    1    1    5    6    1     9
## V4    6    4    3    1    1    8    8    7    3     2
## V5    1    1    3    4    2    8    5    6    6     2
```

```
# Dimensions
print(dim(n))
```

```
## [1]  5 10
```

Question 2: What are the dimensions of the matrix you just transposed?

Answer 2: 5 x 10

###Indexing a Matrix

In the R code chunk below, do the following: 1) Index matrix `m` by selecting all but the third column. 2) Remove the last row of matrix `m`.

```
# Index m without third column
mminus3rd <- m[, c(1:2,4:5)]
print(mminus3rd)
```

```
##       V1 V2 V4 V5
##  [1,]  8  1  6  1
##  [2,]  5  5  4  1
##  [3,]  2  5  3  3
##  [4,]  3  2  1  4
##  [5,]  9  9  1  2
##  [6,] 11  8  8  8
##  [7,]  2  2  8  5
##  [8,]  3  3  7  6
##  [9,]  5  5  3  6
## [10,]  6  5  2  2
```

```
# Index matrix m without last column
mminus5th <- m[, 1:4]
print(mminus5th)
```

```
##       V1 V2 V3 V4
##  [1,]  8  1  7  6
##  [2,]  5  5  2  4
##  [3,]  2  5  4  3
##  [4,]  3  2  5  1
##  [5,]  9  9  1  1
##  [6,] 11  8  1  8
##  [7,]  2  2  5  8
##  [8,]  3  3  6  7
##  [9,]  5  5  1  3
## [10,]  6  5  9  2
```

6) BASIC DATA VISUALIZATION AND STATISTICAL ANALYSIS

Load Zooplankton Data Set

In the R code chunk below, do the following: 1) Load the zooplankton data set from the **3.RStudio** data folder. 2) Display the structure of this data set.

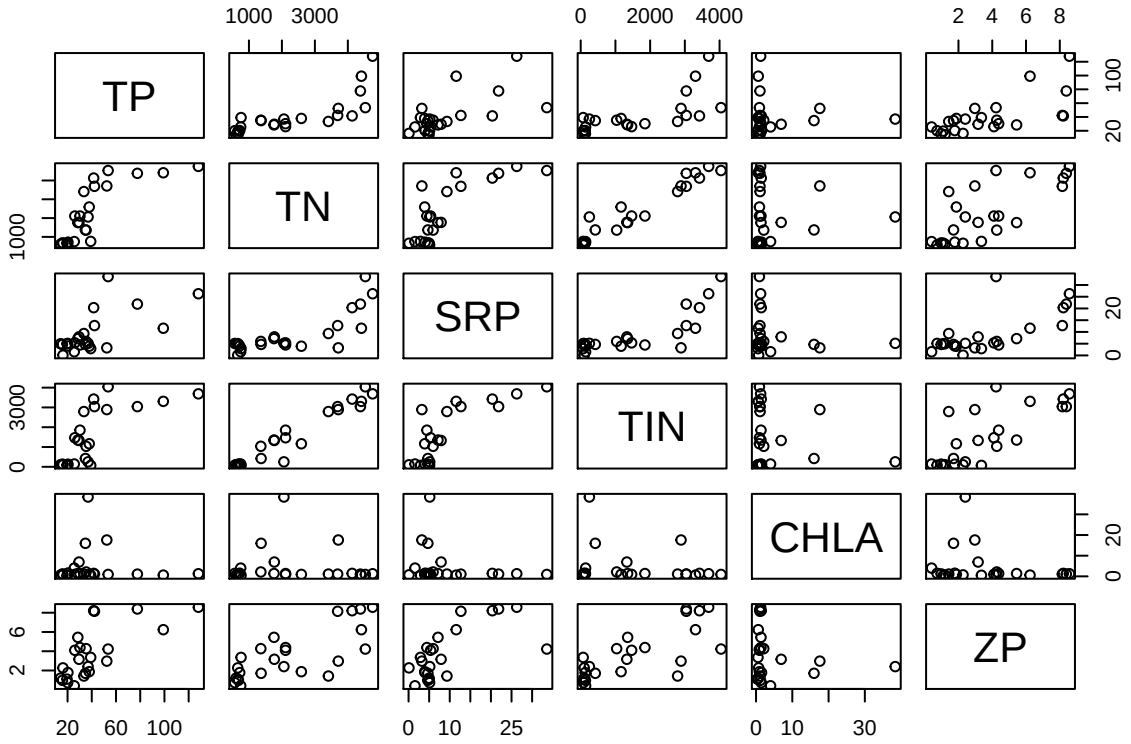
```
# Load data
meso <- read.table("data/zoop_nuts.txt", sep = "\t", header = TRUE)
# Structure
str(meso)
```

```
## 'data.frame':    24 obs. of  8 variables:
##  $ TANK: int  34 14 23 16 21 5 25 27 30 28 ...
##  $ NUTS: Factor w/ 3 levels "H","L","M": 2 2 2 2 2 2 2 2 3 3 ...
##  $ TP  : num  20.3 25.6 14.2 39.1 20.1 ...
##  $ TN  : num  720 750 610 761 570 ...
##  $ SRP : num  4.02 1.56 4.97 2.89 5.11 4.68 5 0.1 7.9 3.92 ...
##  $ TIN : num  131.6 141.1 107.7 71.3 80.4 ...
##  $ CHLA: num  1.52 4 0.61 0.53 1.44 1.19 0.37 0.72 6.93 0.94 ...
##  $ ZP  : num  1.781 0.409 1.201 3.36 0.733 ...
```

Correlation

In the R-code chunk below, do the following: 1) Create a matrix with the numerical data in the `meso` dataframe. 2) Visualize the pairwise **bi-plots** of the six numerical variables. 3) Conduct a simple **Pearson's correlation** analysis.

```
# Create matrix
meso.num <- meso[,3:8]
# Bi-plots
pairs(meso.num)
```




```
# Pearson's correlation
cor1 <- cor(meso.num)
print(cor1)
```

```
##                TP           TN        SRP        TIN         CHLA         ZP
## TP     1.00000000  0.786510407  0.6540957  0.7171143 -0.016659593  0.6974765
## TN     0.78651041  1.000000000  0.7841904  0.9689999 -0.004470263  0.7562474
## SRP    0.65409569  0.784190400  1.0000000  0.8009033 -0.189148017  0.6762947
## TIN    0.71711434  0.968999866  0.8009033  1.0000000 -0.156881463  0.7605629
## CHLA -0.01665959 -0.004470263 -0.1891480 -0.1568815  1.000000000 -0.1825999
## ZP     0.69747649  0.756247384  0.6762947  0.7605629 -0.182599904  1.0000000
```

Question 3: Describe some of the general features based on the visualization and correlation analysis above?

> Answer 3: There are some strong correlations and some very weak correlations present in the data, ranging from a high of 0.97 to a low of -0.19.

In the R code chunk below, do the following: 1) Redo the correlation analysis using the `corr.test()` function in the `psych` package with the following options: method = "pearson", adjust = "BH". 2) Now, redo this correlation analysis using a non-parametric method. 3) Use the print command from the handout to see the results of each correlation analysis.

```r
# Download psych package
#install.packages("psych", repos="http://cran.rstudio.com/")
require("psych")
```

```
## Loading required package: psych
```

```
## Warning: package 'psych' was built under R version 3.6.3
```

```r
# Conduct corr.test
cor2 <- corr.test(meso.num, method = "pearson", adjust = "BH")
# Print
print(cor2, digits = 3)
```

```
## Call:corr.test(x = meso.num, method = "pearson", adjust = "BH")
## Correlation matrix
##          TP     TN    SRP    TIN   CHLA     ZP
## TP    1.000  0.787  0.654  0.717 -0.017  0.697
## TN    0.787  1.000  0.784  0.969 -0.004  0.756
## SRP   0.654  0.784  1.000  0.801 -0.189  0.676
## TIN   0.717  0.969  0.801  1.000 -0.157  0.761
## CHLA -0.017 -0.004 -0.189 -0.157  1.000 -0.183
## ZP    0.697  0.756  0.676  0.761 -0.183  1.000
## Sample Size
## [1] 24
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##         TP    TN   SRP   TIN  CHLA    ZP
## TP   0.000 0.000 0.001 0.000 0.983 0.000
## TN   0.000 0.000 0.000 0.000 0.983 0.000
## SRP  0.001 0.000 0.000 0.000 0.491 0.000
## TIN  0.000 0.000 0.000 0.000 0.536 0.000
## CHLA 0.938 0.983 0.376 0.464 0.000 0.491
## ZP   0.000 0.000 0.000 0.000 0.393 0.000
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

```r
# Non-parametric method
cor3 <- corr.test(meso.num, method = "kendall", adjust = "BH")
# Print
print(cor3, digits = 3)
```

```
## Call:corr.test(x = meso.num, method = "kendall", adjust = "BH")
## Correlation matrix
##         TP    TN    SRP   TIN   CHLA     ZP
## TP   1.000 0.739  0.391 0.577  0.044  0.536
## TN   0.739 1.000  0.478 0.809  0.015  0.551
## SRP  0.391 0.478  1.000 0.563 -0.066  0.449
## TIN  0.577 0.809  0.563 1.000  0.044  0.548
## CHLA 0.044 0.015 -0.066 0.044  1.000 -0.051
## ZP   0.536 0.551  0.449 0.548 -0.051  1.000
## Sample Size
## [1] 24
```

```
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
##         TP    TN   SRP   TIN  CHLA    ZP
## TP   0.000 0.000 0.088 0.014 0.899 0.015
## TN   0.000 0.000 0.034 0.000 0.946 0.014
## SRP  0.059 0.018 0.000 0.014 0.899 0.046
## TIN  0.003 0.000 0.004 0.000 0.899 0.014
## CHLA 0.839 0.946 0.760 0.839 0.000 0.899
## ZP   0.007 0.005 0.028 0.006 0.813 0.000
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

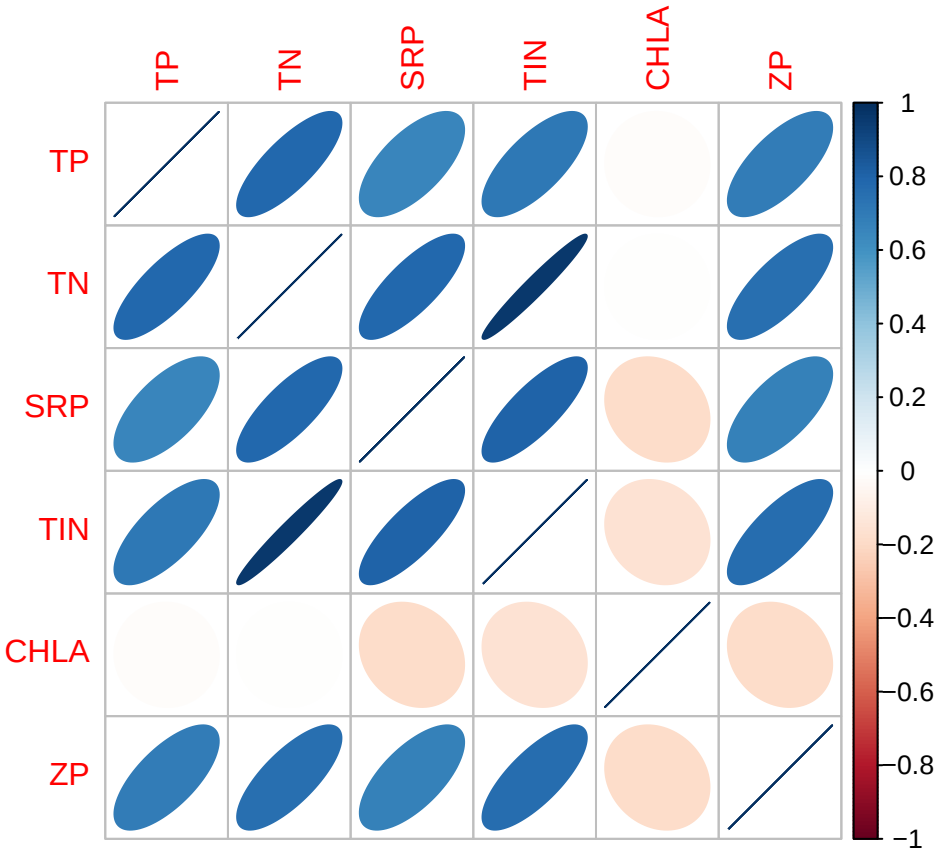
```
# Visualize
#install.packages("corrplot", repos = "http://cran.rstudio.com/")
require("corrplot")
```

```
## Loading required package: corrplot
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
corrplot(cor1, method="ellipse")
```



```
dev.off()
```

```
## null device
##           1
```

Question 4: Describe what you learned from `corr.test`. Specifically, are the results sensitive to whether you use parametric (i.e., Pearson's) or non-parametric methods? When should one use non-parametric methods instead of parametric methods? With the Pearson's method, is there evidence for false discovery rate due to multiple comparisons? Why is false discovery rate important?

> Answer 4: First, yes, the results from the non-parametric tests are generally lower than the parametric tests. Second, if the data being analyzed is normal you use parametric tests and if the data is not normal you use non-parametric tests. Third, yes, the probability values are different from their original values after they are adjusted for multiple tests. The false discovery rate is important because it supplies a statistically easy way to correct for multiple tests, which needs to be done because using the same data to perform multiple tests inflates our significance level to a value greater than 0.05.

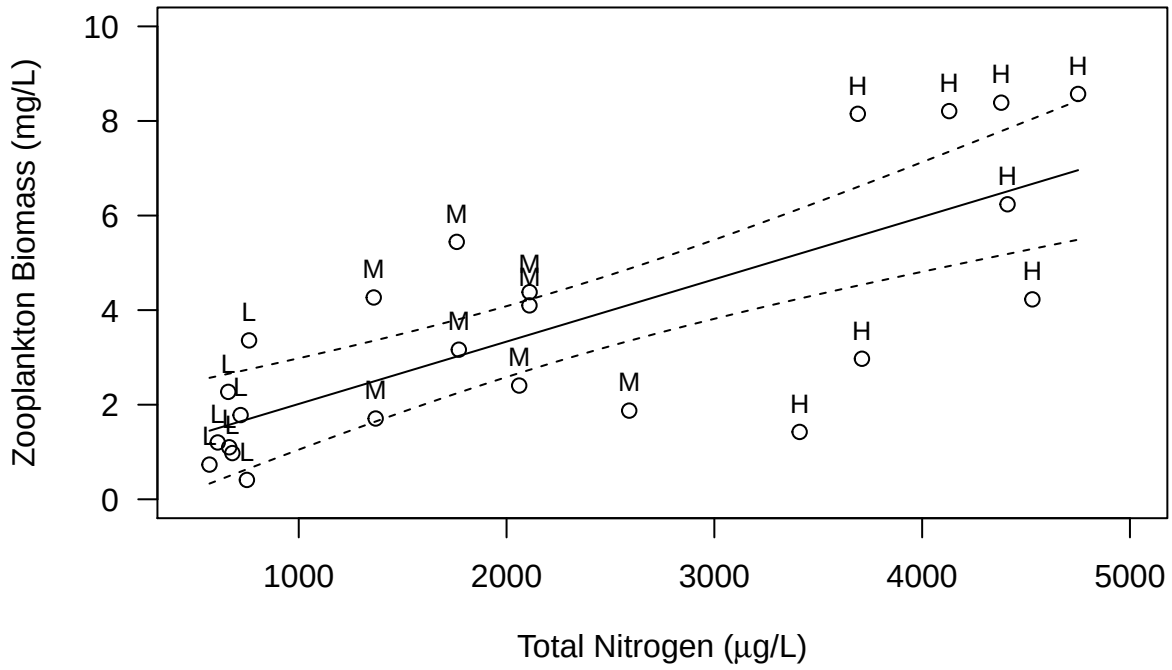
Linear Regression

In the R code chunk below, do the following: 1) Conduct a linear regression analysis to test the relationship between total nitrogen (TN) and zooplankton biomass (ZP). 2) Examine the output of the regression analysis. 3) Produce a plot of this regression analysis including the following: categorically labeled points, the predicted regression line with 95% confidence intervals, and the appropriate axis labels.

```
# Linear regression
fitreg <- lm(ZP ~ TN, data = meso)
# Examine regression model output
summary(fitreg)
```

```
##
## Call:
## lm(formula = ZP ~ TN, data = meso)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7690 -0.8491 -0.0709  1.6238  2.5888
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.6977712  0.6496312   1.074    0.294
## TN          0.0013181  0.0002431   5.421 1.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.75 on 22 degrees of freedom
## Multiple R-squared:  0.5719, Adjusted R-squared:  0.5525
## F-statistic: 29.39 on 1 and 22 DF,  p-value: 1.911e-05
```

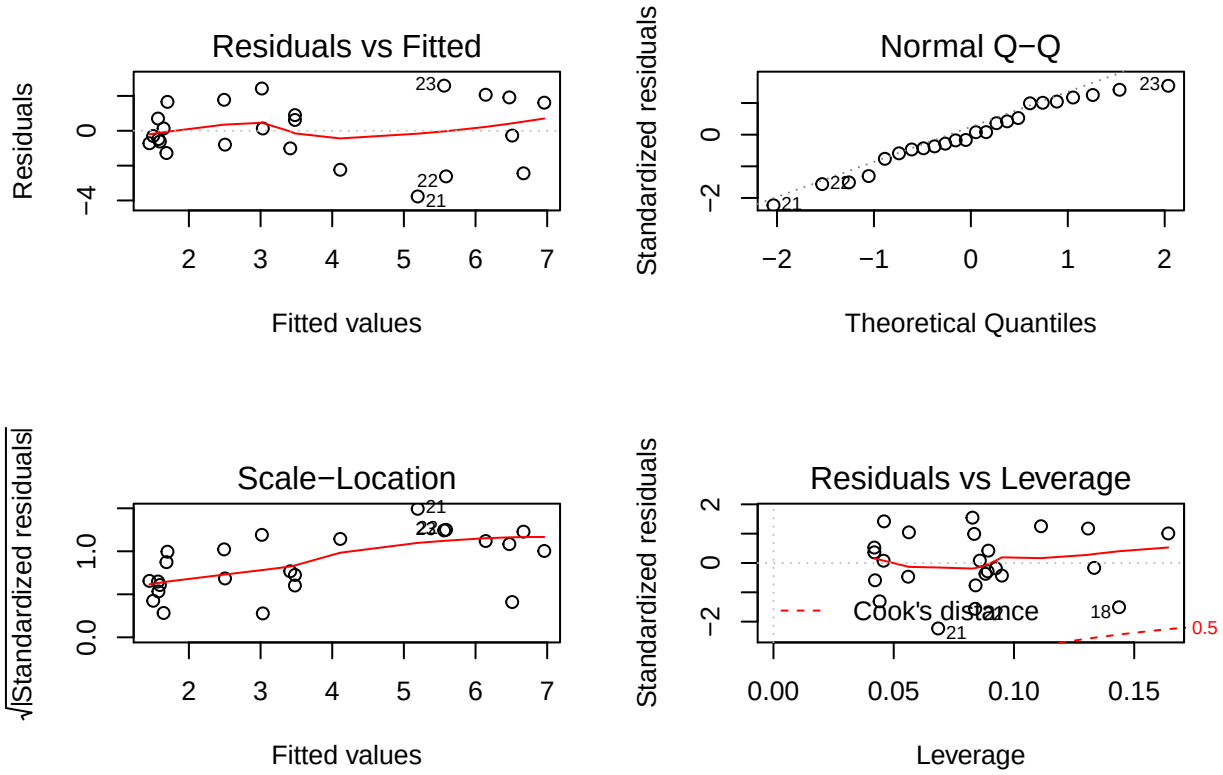
```r
# Plot
plot(meso$TN, meso$ZP, ylim = c(0, 10), xlim = c(500, 5000), xlab = expression(paste("Total Nitrogen ("
# Add text
text(meso$TN, meso$ZP, meso$NUTS, pos = 3, cex = 0.8)
# Add regression line
newTN <- seq(min(meso$TN), max(meso$TN), 10)
regline <- predict(fitreg, newdata = data.frame(TN = newTN))
lines(newTN, regline)
# Add 95% confidence intervals
conf95 <- predict(fitreg, newdata = data.frame(TN = newTN), interval = c("confidence"), level = 0.95, ty
matlines(newTN, conf95[, c("lwr", "upr")], lty = 2, lwd = 1, col = "black")
```



Question 5: Interpret the results from the regression model

Answer 5: There is an intermediate correlation of r squared 0.5525 as there is a general positive correlation observed in the figure; however, several individual points fall outside of the 95% confidence interval lines shown. Below I analyze whether the residuals are normally distributed and homoscedastic.

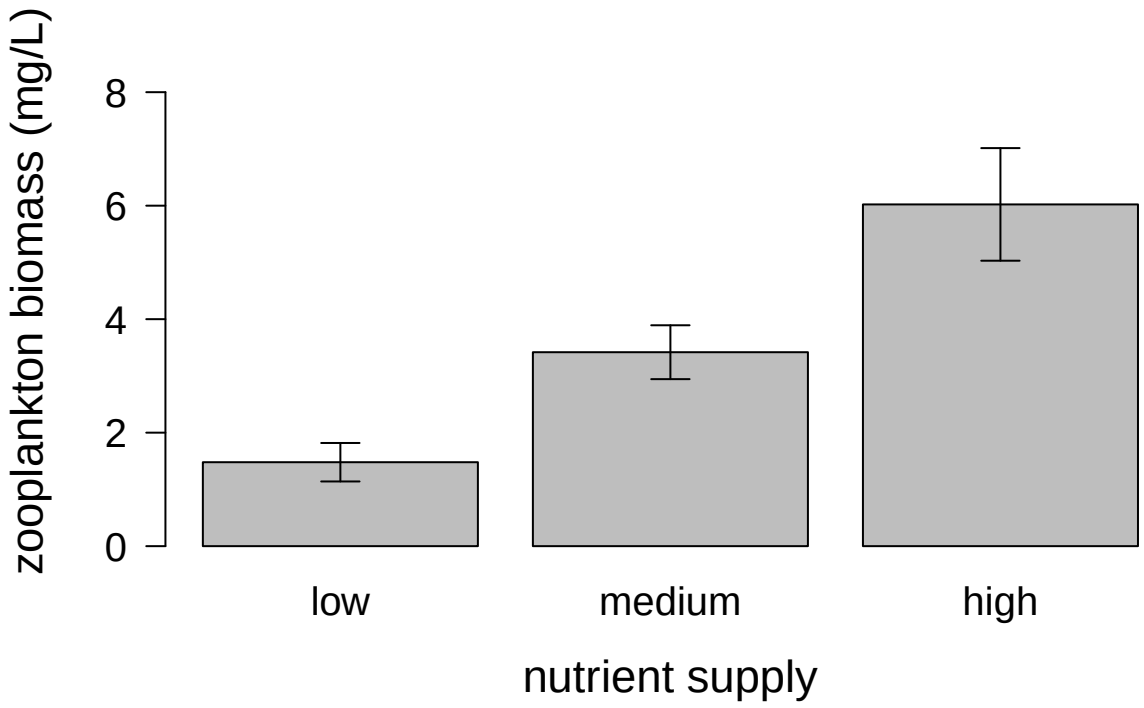
```r
par(mfrow = c(2, 2), mar = c(5.1, 4.1, 4.1, 2.1))
plot(fitreg)
```



Analysis of Variance (ANOVA)

Using the R code chunk below, do the following: 1) Order the nutrient treatments from low to high (see handout). 2) Produce a barplot to visualize zooplankton biomass in each nutrient treatment. 3) Include error bars (+/- 1 sem) on your plot and label the axes appropriately. 4) Use a one-way analysis of variance (ANOVA) to test the null hypothesis that zooplankton biomass is affected by the nutrient treatment.

```
# Order
NUTS <- factor(meso$NUTS, levels = c("L", "M", "H"))
# Calculate means and standard errors
zp.means <- tapply(meso$ZP, NUTS, mean)
sem <- function(x){
  sd(na.omit(x))/sqrt(length(na.omit(x)))
}
zp.sem <- tapply(meso$ZP, NUTS, sem)
# Make barplot
bp <- barplot(zp.means, ylim = c(0, round(max(meso$ZP), digits = 0)), pch = 15, cex = 1.25, las = 1, ce
# Add error bars
arrows(x0 = bp, y0 = zp.means, y1 = zp.means - zp.sem, angle = 90, length = 0.1, lwd = 1)
arrows(x0 = bp, y0 = zp.means, y1 = zp.means + zp.sem, angle = 90, length = 0.1, lwd =1)
```



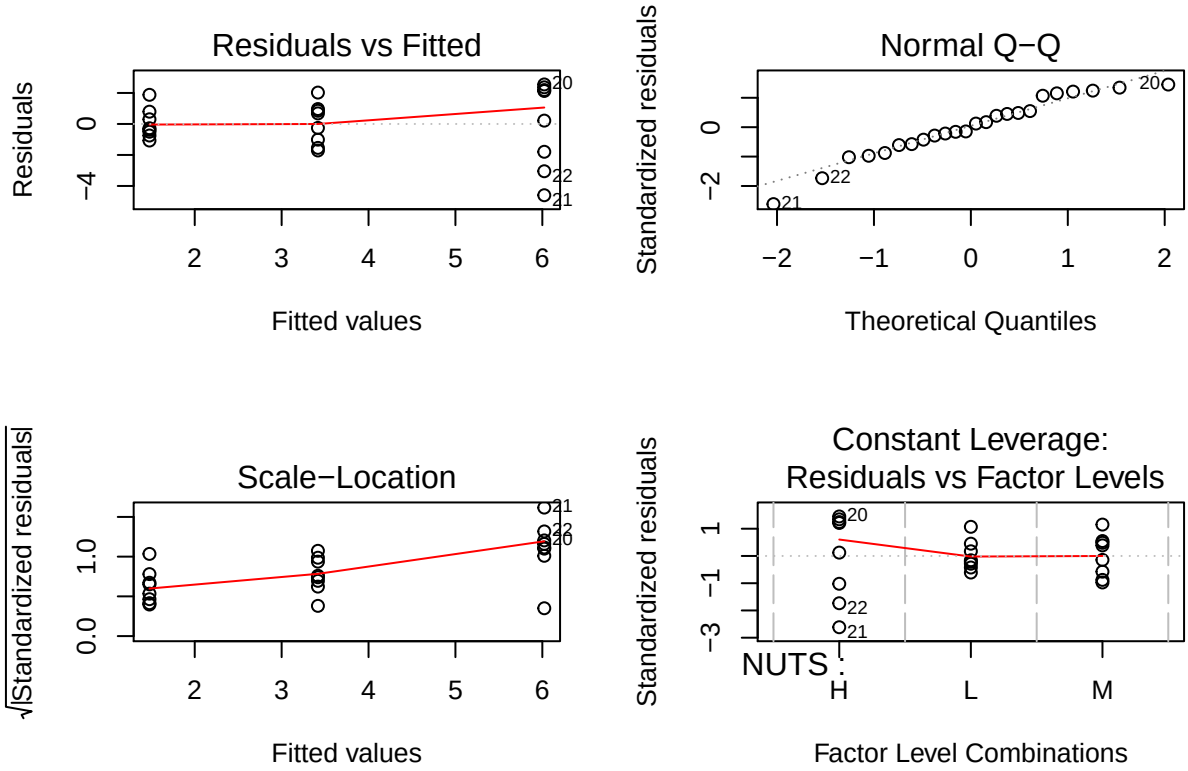

```
# Conduct ANOVA
fitanova <- aov(ZP~NUTS, data = meso)
summary(fitanova)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## NUTS         2  83.15   41.58   11.77 0.000372 ***
## Residuals   21  74.16    3.53
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Posthoc comparison
TukeyHSD(fitanova)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = ZP ~ NUTS, data = meso)
##
## $NUTS
##          diff        lwr        upr     p adj
## L-H -4.543175 -6.9115094 -2.1748406 0.0002512
## M-H -2.604550 -4.9728844 -0.2362156 0.0294932
## M-L  1.938625 -0.4297094  4.3069594 0.1220246
```

```r
# Residuals
par(mfrow = c(2,2), mar = c(5.1, 4.1, 4.1, 2.1))
plot(fitanova)
```



SYNTHESIS: SITE-BY-SPECIES MATRIX

In the R code chunk below, load the zoops.txt data set in your **3.RStudio** data folder. Create a site-by-species matrix (or dataframe) that does *not* include TANK or NUTS. The remaining columns of data refer to the biomass (µg/L) of different zooplankton taxa:

- CAL = calanoid copepods
- DIAP = *Diaphanasoma* sp.
- CYL = cyclopoid copepods
- BOSM = *Bosmina* sp.
- SIMO = *Simocephallus* sp.
- CERI = *Ceriodaphnia* sp.
- NAUP = naupuli (immature copepod)
- DLUM = *Daphnia lumholtzi*
- CHYD = *Chydorus* sp.

Question 6: With the visualization and statistical tools that we learned about in the **3. RStudio** handout, use the site-by-species matrix to assess whether and how different zooplankton taxa were responsible for the total biomass (ZP) response to nutrient enrichment. Describe what you learned below in the "Answer" section and include appropriate code in the R chunk.

```
# Load data
ZOOPS <- read.table("data/zoops.txt", sep = "\t", header = TRUE)
# Remove first two columns
zoops <- ZOOPS[,3:11]
# Sum biomass of each taxa
sums <- colSums(zoops)
# Sum total biomass
biomass <- sum(sums)
# Calculate percentage of biomass each taxon accounts for
biomassperc <- sums/biomass * 1
print(biomassperc)
```

```
##          CAL         DIAP         CYCL         BOSM         SIMO         CERI
## 0.0088404951 0.0103754382 0.0273661039 0.0003063029 0.1213233655 0.0342487751
##         NAUP         DLUM         CHYD
## 0.0001702953 0.0000754328 0.7972937913
```

Answer Section

> Answer 6: Most taxa accounted for a very small percentage of the increase in total biomass. In fact, 7 of the 9 taxa accounted for less than 10% of the biomass increase. Only Chydorus (79.7%) and Simocephallus (12.1%) exceeded this percentage, with Chydorus providing almost 80% of the biomass of the surveyed zooplankton.

SUBMITTING YOUR WORKSHEET

Use Knitr to create a PDF of your completed **3.RStudio_Worksheet.Rmd** document, push the repo to GitHub, and create a pull request. Please make sure your updated repo include both the PDF and RMarkdown files.

This assignment is due on **Wednesday, January 24th, 2021 at 12:00 PM (noon)**.