RLab 4: Investigating Relationships

Lab Outcomes

1. Use scatter and line plots to visualize relationship between two parameters
2. Use statistics to determine the strength of a relationship (correlation coefficients and linear regressions)

Today we are going to use the `DaRTS_combined_data.csv` file that we made last day.

Initializing our R Session

Let's start by setting up our R session. These are good steps to take at the start of any R session.

1. Select your R project
2. Open a new script or r-markdown document
3. Save the new script e.g. `Lab04.rmd`
4. Import the libraries we'll use today

```r
library(ggplot2)
```

5. Import our data

```r
DATA <- read.csv("DaRTS_combined_data.csv")
```

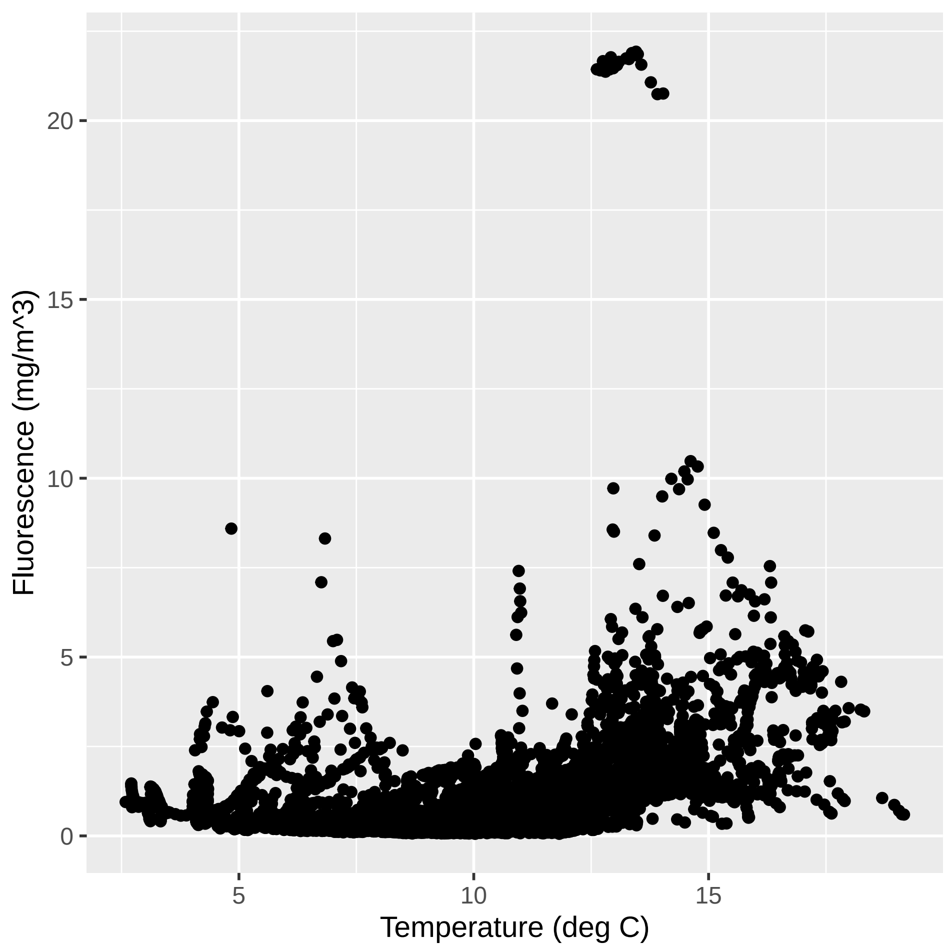
Scatter plots

Scatter plots allow us to quickly understand and visualize relationships.

```r
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)')
```

Example: How is chlorophyll fluorescence related to temperature?

```
## Warning: Removed 802 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



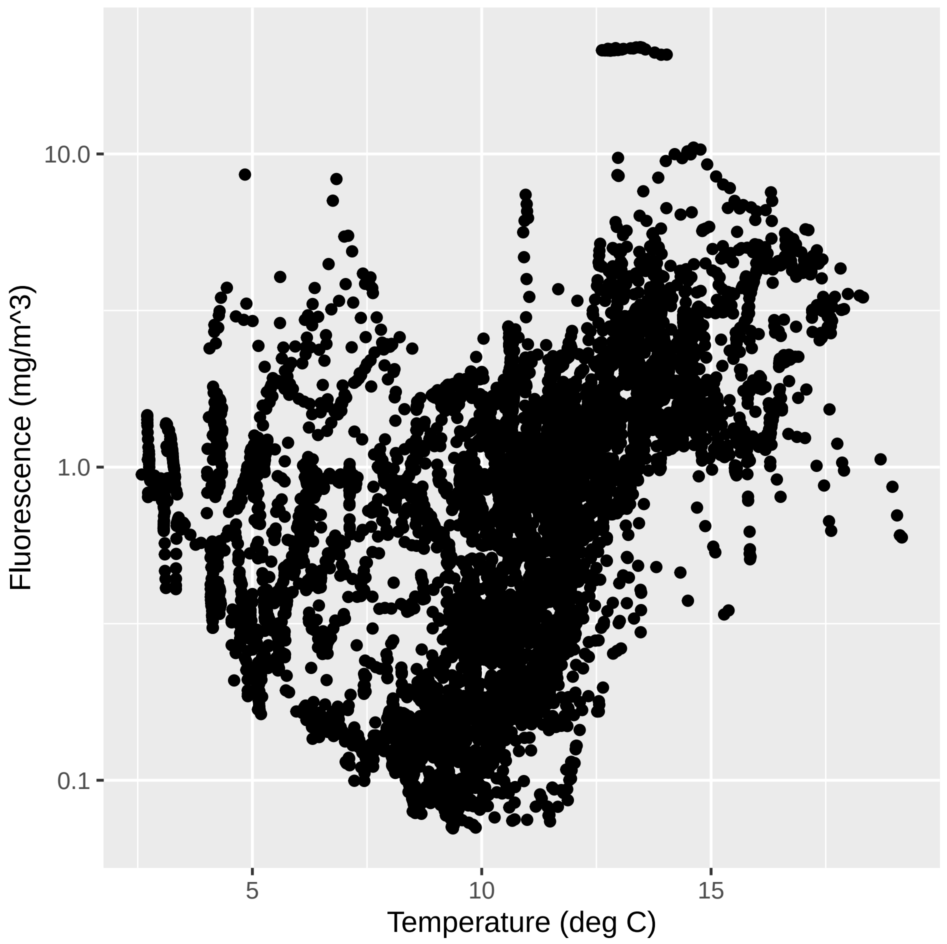
Log-transformations

In the above figure, we have a lot of fluorescence data below 2.5 mg m^{-3} - in fact, the majority of our data are below that value. These data are approximately *log-normally* distributed, that is, they have a lot of observations at the lower values, and fewer at the higher values. This is typical of a lot of geophysical parameters like fluorescence. Hence, we often plot (& perform statistics on) the log-transformed data, so we can better understand the relationship between the parameters.

What is the relationship between temperature and log-transformed fluorescence?

```r
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  scale_y_log10(limits=c(0.07,22))
```

```
## Warning: Removed 807 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



The above is a great visual tool - but how do we define this relationship scientifically and statistically? We'll look at two different techniques, but often scientists tend to use both of these.

Correlations between two parameters

In everyday speech, *correlation* is often used interchangeably with relationship and is defined as:

> the state or relation of being correlated; specifically : a relation existing between phenomena or things or between mathematical or statistical variables which tend to vary, be associated, or occur together in a way not expected on the basis of chance alone.

How is correlation defined in a mathematical sense? Using *correlation coefficients.*

Correlation coefficients

Correlation coefficients can be used to evaluate the direction and the intensity of the relationship between two variables. There are a range of different correlation coefficients, but the most common of these is the *Pearson correlation coefficient.* The Pearson correlation coefficient, often referred to as simply the correlation coefficient, is given the symbol r and describes the linear correlation between two variables. It takes a value between -1 and 1. The sign of the correlation coefficient describes the direction of the correlation e.g. a positive r indicates a positive relationship between the variables. A value of 0 indicates no linear correlation and values further from 0 in either direction (closer to -1 and 1) indicate stronger linear correlations. See the image below for examples.

[Source: Wikipedia]

Example 1: The game "Guess the Correlation" Try this game: Guess the Correlation

In this game, players are presented with a stream of scatter plots depicting the relationship between two random variables. The aim is to guess the true Pearson correlation coefficient. Guesses over 0.10 from the right answer are not awarded any points and a life is deducted.

Example 2: What is the correlation coefficient between chlorophyll fluorescence and temperature? Here, we are going to use R to calculate the Pearson's correlation coefficient.

```r
# create variables for our two parameters of interest
# NB: we want to the data to be vectors, rather than dataframe columns so we can
# perform our analysis, hence we need to use the dollar sign, $, to index the DATA dataframe
temp <- DATA$Temperature_C
fl <- DATA$Fluorescence

# calculate the correlation coefficient
corValue <- cor.test(temp, fl, method="pearson")

# displaying the correlation coefficient
corValue
```

```
##
##  Pearson's product-moment correlation
##
## data:  temp and fl
## t = 30.905, df = 7525, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3154049 0.3555016
## sample estimates:
##       cor
## 0.3356053
```

The correlation coefficient is the number right at the bottom of the printout, below the word `cor`. In this case, we have a correlation coefficient of 0.336.

But what about all those other bits of information?

The rest of the information tells us if our correlation is significant or not. There's lots of different tests to use, but in this case, we're using a Students t-test. In the t-test, we are testing the following null and alternative hypotheses:

$\mathbf{H}_0$: the correlation coefficient is equal to 0

$\mathbf{H}_A$: the correlation coefficient is not equal to 0

The results of the t-test is a t-score (or t-statistic) and p-value. If the p-value that corresponds to the t-score is less than some threshold (typically 0.05) then we can reject the null hypothesis, and say we have a statistically significant correlation.

Aside: the t-score and p-value are connected through the t-distribution function. We are not covering those details in this course.

Example 2b: What about on our log transformed data? Here, we need to remove the bad data before doing the correlation calculation. To remove the bad data, we're going to make use of *Booleans* and *logical indexing*.

Aside: Logical indexing

In programming, we often need to know if something is true or false. To evaluate is something is true or false, we use Boolean operators, and our resulting data types are called Boolean values. There are only two possible Boolean values: `TRUE` and `FALSE` (in some programming languages, these are represented by 1 (= `TRUE`) and 0 (= `FALSE`)).

Booleans are often paired with indexing in a process known as logical indexing. We've actually been doing some indexing already, we've been doing it using some functions that hide the process of indexing, and we've done it ourselves a couple of times, we just didn't give it that name.

Indexing is the process of pulling out a subset of data from a larger group of data. That can be pulling out an individual column from a data frame, like we did in Lab 3 when we were working with the dates:

```
# one way
DATA$Date

# another way:
DATA[['Date']]

# another way
DATA['Date']
```

Or indexing can be pulling out one or more elements from a list (or vector, or matrix) - we've done this a couple of times when we wanted to print out just the first 10 depths from our data frame, or the first date in our data frame:

```
depths <- DATA$Depth

depths[1:10]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Even the `dplyr` function `filter` (which we've been using to subset our whole data frame based on e.g. a particular year) uses indexing.

But you can also index a list using another list of values e.g. say we want the 5th, 8th, 15th and 30th depth:

```
# make our list of indices
ind <- c(5,8,15,30)

# index depth with our indices
depths[ind]
```

```
## [1]  5  8 15 30
```

```
# index temperature with our indices
temp[ind]
```

```
## [1] 16.11212 15.98892 15.90805 15.62558
```

So what about logical indexing? In logical indexing, our indices are Boolean values i.e. either `TRUE` or `FALSE`. For example, if we want to pull out all the depths less than 10 m:

```
ind <- depths < 10

# print out the first 50 elements to see what this object looks like
ind[1:50]
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE
```

```
# Then index the depths object
depths[ind]
```

```
##    [1]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##   [25]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##   [49]  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
##   [73]  9  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  1  2  3
##   [97]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [121]  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  7  8  9  1  2  3  4  5
##  [145]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
##  [169]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
##  [193]  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5
##  [217]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
##  [241]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
##  [265]  9  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [289]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [313]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [337]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [361]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [385]  4  5  6  7  8  9  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1
##  [409]  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7
##  [433]  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4
##  [457]  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1
##  [481]  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7
##  [505]  8  9  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5
##  [529]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
##  [553]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
##  [577]  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5
##  [601]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
##  [625]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
##  [649]  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5
##  [673]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
##  [697]  3  4  5  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [721]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [745]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [769]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [793]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [817]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [841]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [865]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [889]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [913]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
##  [937]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
##  [961]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
##  [985]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
## [1009]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
## [1033]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
## [1057]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
## [1081]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
## [1105]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
## [1129]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
## [1153]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
## [1177]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
## [1201]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
## [1225]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
## [1249]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
## [1273]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  0  1  2
## [1297]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5  6  1
## [1321]  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7
## [1345]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6
## [1369]  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3
## [1393]  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9
## [1417]  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
## [1441]  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2
## [1465]  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8
## [1489]  9  1  2  3  4  5  6  7  8  9  1  2  3  4  5  6  7  8  9 NA NA NA  2  2
## [1513]  2 NA  2  2  2 NA NA NA NA  2 NA NA  2  2  2  2  2  2  2  2  2 NA NA NA
## [1537] NA NA NA NA NA NA  2 NA NA  2 NA NA NA NA NA  2  2  2  2  2  2  2  2  2
## [1561]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  5  2  5  2  5  2  5
```

Here, we've pulled out only the elements in our depths list that have a value of less than 10.

```r
templog <- DATA$Temperature_C
fllog <- log10(DATA$Fluorescence)

# removing the NaNs and Infs:
# 1) make a boolean list of the good values
nonans <- is.finite(fllog)
# 2) index temp and fl objects using our boolean list
tempnonan <- templog[nonans]
flnonan <- fllog[nonans]

# calculate the correlation coefficient
corValue <- cor.test(tempnonan, flnonan)

# displaying the correlation coefficient
corValue
```

Back to Example 2b: what is the correlation coefficient on the log-transformed chl and temperature data?

```
##
##  Pearson's product-moment correlation
##
## data:  tempnonan and flnonan
## t = 42.267, df = 7525, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4195766 0.4560948
## sample estimates:
##       cor
## 0.4380164
```

So we end up with a slightly higher correlation coefficient between the log-transformed fluorescence data and temperature (than for the measured fluorescence data and temperature).

Linear regressions

The relationship between two variables can be explained through a simple linear regression (here *simple* refers to the fact that two variables are being related). The main outputs from this kind of analysis are a line or curve of best fit and various *summary statistics* or *goodness-of-fit statistics* or *explanatory variables* which define how good the linear regression is. This analysis can be extended to describe relationships between multiple variables via linear multiple regression analysis.

Best fit line

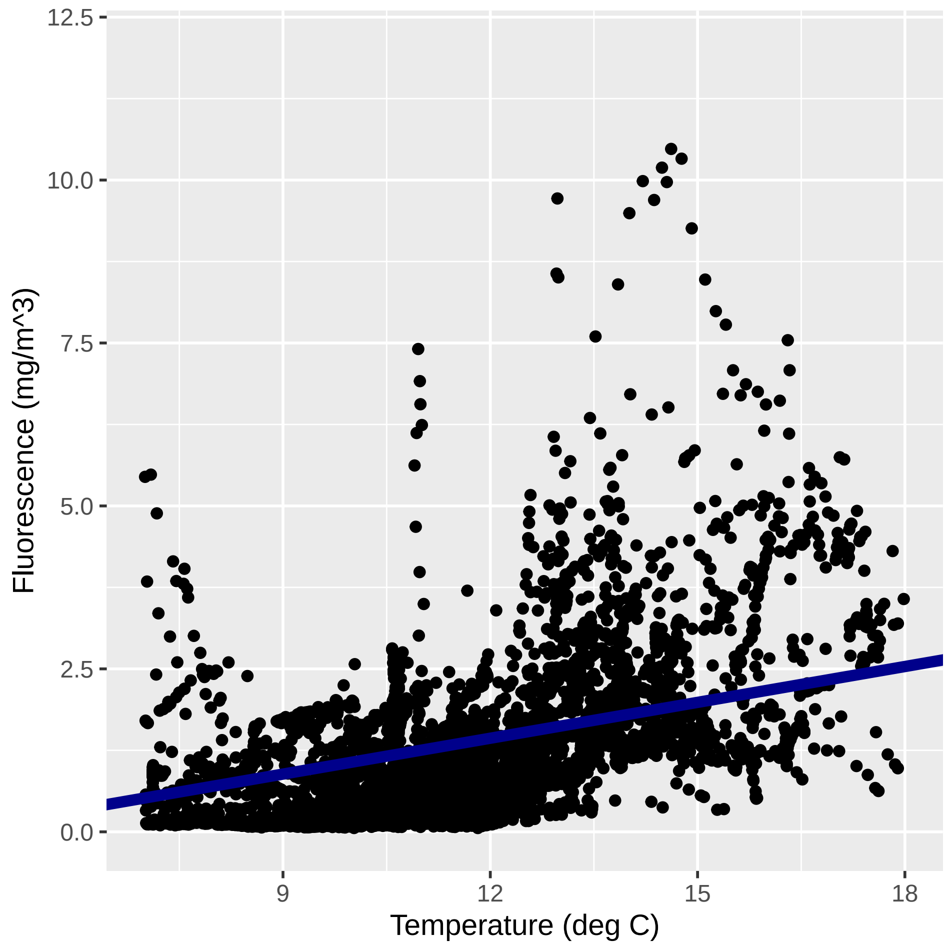
Example 1: What is the line of best fit between the chlorophyll fluoroescence and temperature? We need to fit a linear model to our data:

```
# fitting a linear model between the two variables
model1 <- lm(fl ~ temp)

# assigning the slope and intercept from the model to variables
intercept <- model1$coefficients[1]
slope <- model1$coefficients[2]

# adding the line to our plot
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  geom_abline(slope=slope,intercept=intercept, size=2, color = "darkblue") +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  xlim(7,18) + ylim(0,12)
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
## Warning: Removed 1863 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



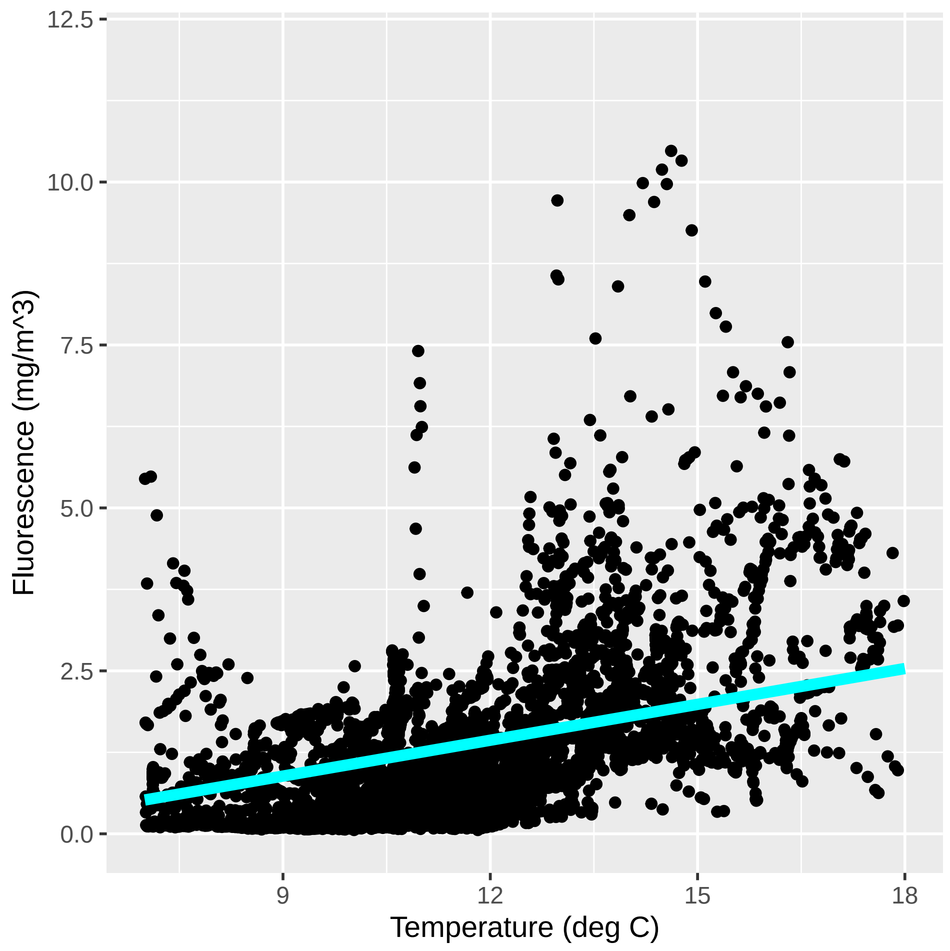
In the above, we used `geom_abline` to add our best fit line, but there are some other ways to do this:

```r
# fitting a linear model between the two variables
model1 <- lm(fl ~ temp)

# assigning the slope and intercept from the model to variables
intercept <- model1$coefficients[1]
slope <- model1$coefficients[2]

# adding the line to our plot
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  stat_function(fun = function(x) intercept + slope * x, size = 2, color = 'cyan') +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  xlim(7,18) + ylim(0,12)
```

```
## Warning: Removed 1863 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



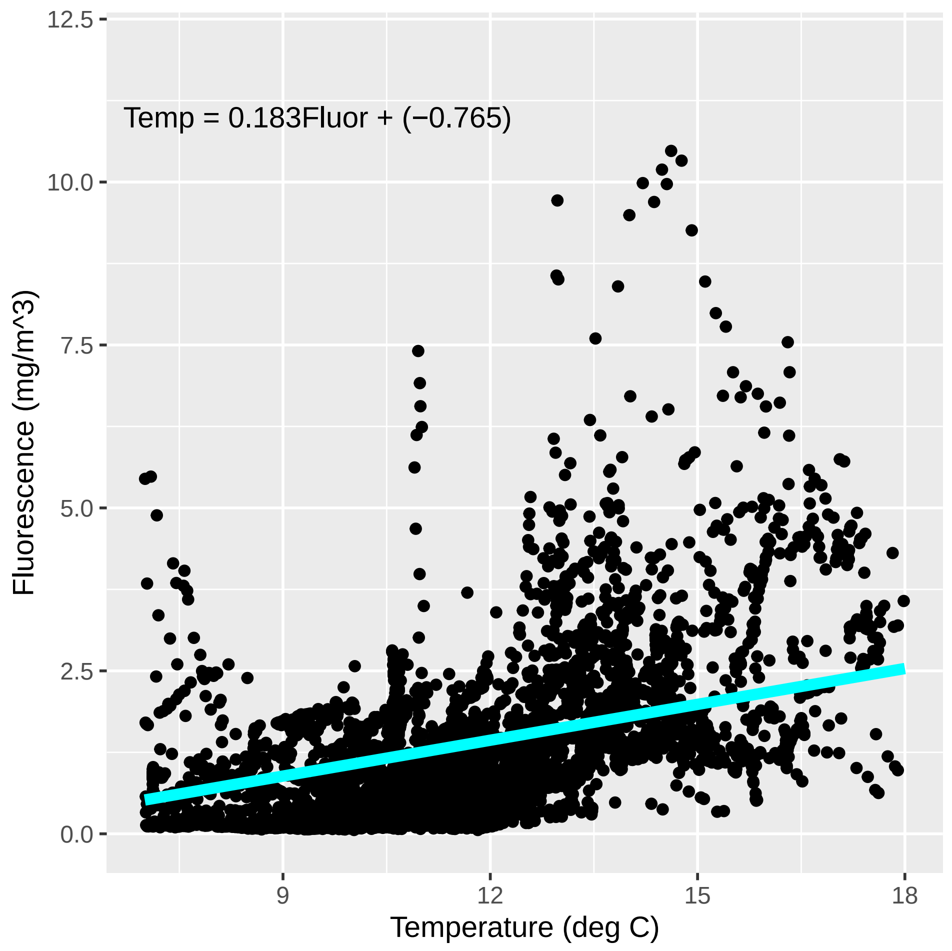
We can also add our best fit line equation to the figure:

```
# fitting a linear model between the two variables
model1 <- lm(fl ~ temp)

# assigning the slope and intercept from the model to variables
intercept <- model1$coefficients[1]
slope <- model1$coefficients[2]

# adding the line to our plot: note we've added the line by three different methods
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  stat_function(fun = function(x) intercept + slope * x, size = 2, color = 'cyan') +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  xlim(7,18) + ylim(0,12) +
  annotate("text", x = 9.5, y = 11, label = paste("Temp = ", format(slope, digits=3), "Fluor + (", forma
```

```
## Warning: Removed 1863 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



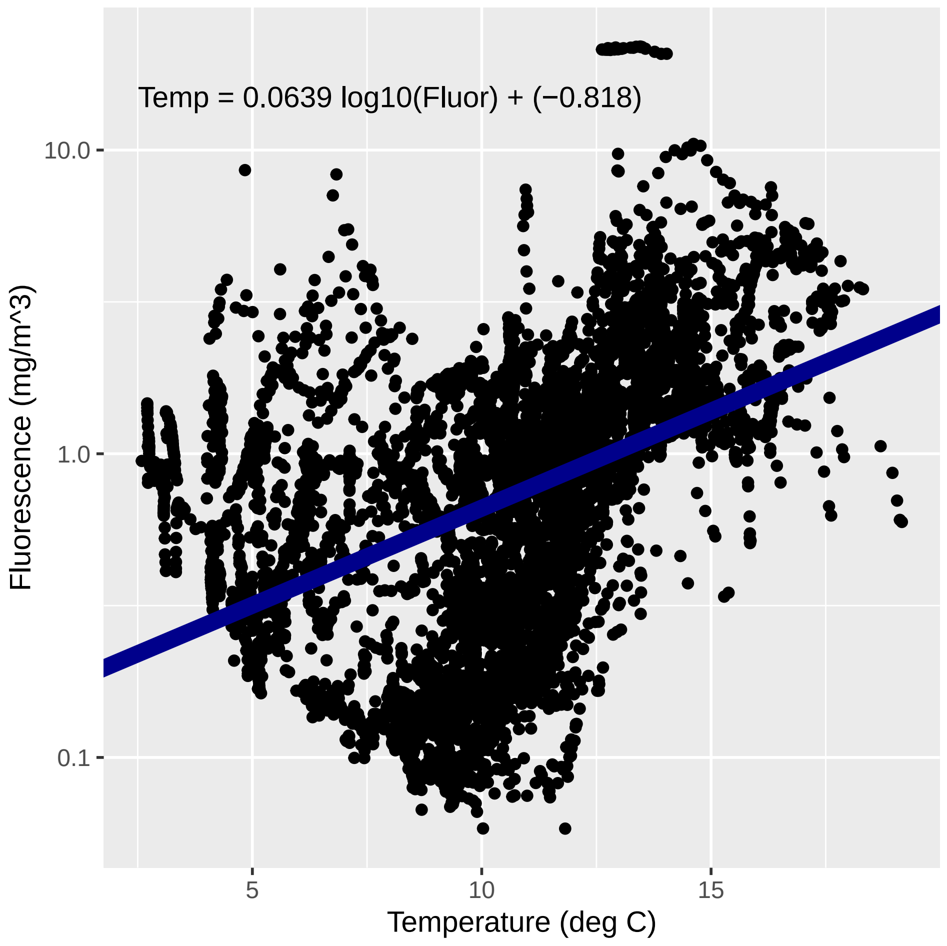
```r
# fitting a linear model between the two variables
modelLog <- lm(flnonan ~ tempnonan)

# assigning the slope and intercept fromt the model to variables
interceptLog <- modelLog$coefficients[1]
slopeLog <- modelLog$coefficients[2]

# adding the line to our plot
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  geom_abline(slope=slopeLog,intercept=interceptLog, size=3, color = "darkblue") +
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  scale_y_log10() +
  annotate("text", x = 8, y = 15, label = paste("Temp = ", format(slopeLog, digits=3), " log10(Fluor) +
```

Example 2: What is the line of best fit between the log transformed chlorophyll fluoroescence and temperature?

```
## Warning: Removed 802 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



In this case, the line doesn't go through the main cloud of data - why might this be?

How do linear regressions work? Linear regressions typically use a least squares method to determine the best fit line. The method aims to minimize the sum of the squares of the distance between all the data points and the fitted line (the *residuals*). In our example of fluorescence and temperature, there are a number of points that have low temperature and relatively high fluorescence values i.e. the fall outside of the main cloud of data points. This group of points results in a regression line away from the main cloud of data. If your data looks similar to this, it's a good idea to look at the individual clusters of data points and see what characteristics are similar about them and decide whether they can be safely removed from your analysis. **But remember: you can't remove data without a good reason.** If you do remove some of the data, you need to make it clear in the presentation of your work ***why*** you did it.

Important side note about linear regressions An important point to note is what makes a regression a *linear* regression. **It is not that you fit a straight line through your data.** We could in fact fit a quadratic or higher order polynomials to our data using linear regression techniques (see below):

```r
# fitting a 2nd and 3rd order polynomial between the two variables
model2 <- lm(fl ~ temp + I(temp^2))

# assigning the coefficients from the models to the variables
```

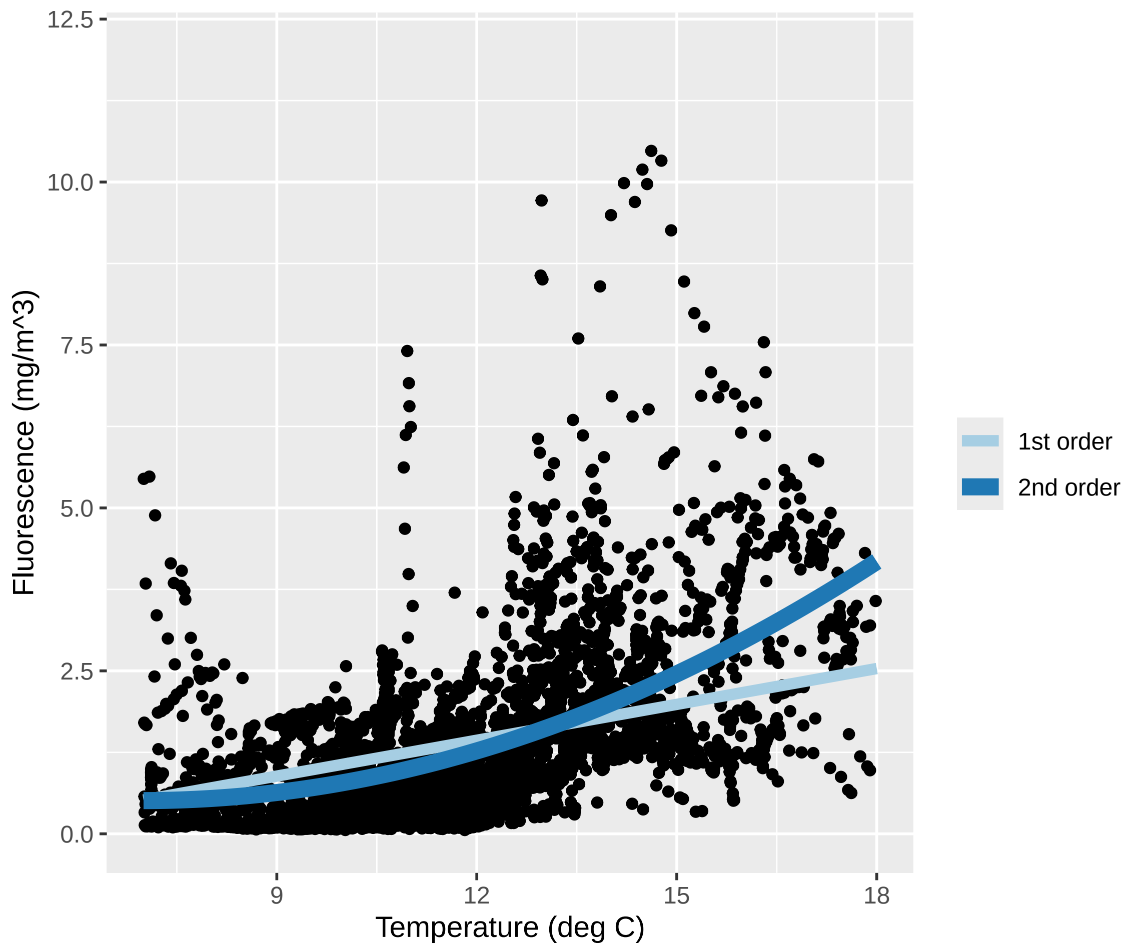
```r
p2 <- model2$coefficients[3]
p1 <- model2$coefficients[2]
p0 <- model2$coefficients[1]

# adding the line to our plot: note only one of the above methods works for curves, stat_function
ggplot(data = DATA) +
  geom_point(mapping = aes(x = Temperature_C, y = Fluorescence)) +
  stat_function(fun = function(x) intercept + slope * x, size = 2, aes(color = '1st order')) +
  stat_function(fun = function(x) p0 + p1 * x + p2 * (x^2), size = 3, aes(color = '2nd order')) +
  scale_colour_manual("",values = c("#a6cee3", "#1f78b4")) + # with Color-hex Color code
  xlab('Temperature (deg C)') +
  ylab('Fluorescence (mg/m^3)') +
  xlim(7,18) + ylim(0,12)
```

```
## Warning in stat_function(fun = function(x) intercept + slope * x, size = 2, : All aesthetics have len
## i Please consider using ‘annotate()‘ or provide this layer with data containing
##   a single row.

## Warning in stat_function(fun = function(x) p0 + p1 * x + p2 * (x^2), size = 3, : All aesthetics have
## i Please consider using ‘annotate()‘ or provide this layer with data containing
##   a single row.

## Warning: Removed 1863 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



What makes it a *linear* regression is the coefficients of the fit are linear. In the above plot, the equation for the quadratic is given by $y = 0.036x^2 - 0.490x + 1.700$. This equation is non-linear in the x-variable, but linear in the coefficients. A non-linear example is a Michaelis-Menten curve which is used to describe phytoplankton nutrient uptake. It has an equation of the form $y = \frac{\alpha x}{\beta + x}$.

Summary statistics

There are a variety of different statistics which are used to describe how well the linear regression describes the relationship between the two variables. Here, we will consider the following variables:

1. Coefficient of determination, R^2 : gives a measure of how much of the variability of the x-variable is explained by the y-variable. For simple linear regressions which include the intercept term, $R^2 = r^2$, i.e. the coefficient of determination is equal to the square of the correlation coefficient.
2. Adjusted R^2 : If there are multiple variables or terms included in the relationship (e.g. the quadratic curve has an x term and an x^2 term), then the adjusted R^2 takes into account only the variables or terms which affect the y variable.
3. p-value : describes the probability of the null hypothesis being true. In the case of linear regression analysis, the null hypothesis is there is no relationship between the two variables. If the p-value is found to be below a given significance level, typically taken as 0.05 (or 5%), then the null hypothesis can be rejected i.e. there is a statistically significant relationship between the two variables.

4. Residual standard error or the root mean squared error : in R, these are equivalent. The *standard error* is defined to show how far the sample mean is from the population mean. It is calculated by dividing the *standard deviation* by the square root of the number of data points. The *standard deviation* is a measure of how much the data points differ from the mean. For a linear regression, the *root mean square error* is estimated by taking the square root of the mean of the residuals-squared (or put another way, by dividing the square of the residuals by the number of data points, and taking the square root). By virtue of the least squares fitting procedure used in the linear regression, the mean of the residuals are zero. Hence, mathematically, the root mean square error and the residual standard error are the same, and in this case give an overall idea of how close the data points are to the fitted line.

Example: Does the straight line, quadatratic or the log relationship describe the relationship between chlorophyll fluorescence and temperature best? Let's print the summary statistics from each of our models (or linear regressions)

```
summary(model1)
```

```
##
## Call:
## lm(formula = fl ~ temp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1524 -0.7338 -0.2786  0.3392 20.2237
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.764976   0.065314  -11.71   <2e-16 ***
## temp         0.183383   0.005934   30.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.548 on 7525 degrees of freedom
##   (802 observations deleted due to missingness)
## Multiple R-squared:  0.1126, Adjusted R-squared:  0.1125
## F-statistic: 955.1 on 1 and 7525 DF,  p-value: < 2.2e-16
```

What we see here is

1. Some information about the *residuals* (the difference between the predicted value and the measured value i.e. the distance on the y-axis between a data point and the best fit line).
2. The fit coefficients (e.g. the slope and intercept) and associated statistics. Notice here we have a t-score and `Pr`. `Pr` is the p-value for the t-test to check if that coefficient is statistically different from 0.
3. Residual standard error (as discussed above).
4. The R^2 and adjusted R^2 (as discussed above).
5. The F-statistic and associated p-value. This is a similar test as a t-test, but in the case of a linear regression, looks at the relationship as a whole, not the individual coefficients. Here, the null hypothesis is the model with no independent variables (also known as an intercept-only model) fits the data as well as the regression model. And the alternative hypothesis is the regression model fits the data better than the intercept-only model.

```r
summary(model2)
```

```
##
## Call:
## lm(formula = fl ~ temp + I(temp^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3945 -0.5729 -0.2600  0.2548 20.1945
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.974199   0.138491   14.26   <2e-16 ***
## temp        -0.420414   0.027757  -15.15   <2e-16 ***
## I(temp^2)    0.030159   0.001356   22.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.5 on 7524 degrees of freedom
##   (802 observations deleted due to missingness)
## Multiple R-squared:  0.1673, Adjusted R-squared:  0.1671
## F-statistic: 756.1 on 2 and 7524 DF,  p-value: < 2.2e-16
```

```r
summary(modelLog)
```

```
##
## Call:
## lm(formula = flnonan ~ tempnonan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.17150 -0.29609  0.06867  0.26335  1.44316
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.818338   0.016646  -49.16   <2e-16 ***
## tempnonan    0.063918   0.001512   42.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3946 on 7525 degrees of freedom
## Multiple R-squared:  0.1919, Adjusted R-squared:  0.1918
## F-statistic:  1786 on 1 and 7525 DF,  p-value: < 2.2e-16
```

Aside: Extracting the summary statistics into individual objects

To compare our results between the different models, it can be easier to have all the relevant statistics together in one data frame. To do this, we need to pull the different statistics out of the summaries programmatically:

```r
# assigning summaries to variables
sum_model1 <- summary(model1)
```

```
sum_model2 <- summary(model2)
sum_model3 <- summary(modelLog)

# combining statistics into vectors
rsquare = c(sum_model1$r.squared,sum_model2$r.squared,sum_model3$r.squared)
adjrsquare = c(sum_model1$adj.r.squared,sum_model2$adj.r.squared,sum_model3$adj.r.squared)
std_error = c(sum_model1$sigma,sum_model2$sigma,sum_model3$sigma)

# unfortunately we can't extract the p-value in the same way, but we can calculate it separately
pvalue = c(pf(sum_model1$fstatistic[1],sum_model1$fstatistic[2],sum_model1$fstatistic[3],lower.tail = F
           pf(sum_model2$fstatistic[1],sum_model2$fstatistic[2],sum_model2$fstatistic[3],lower.tail = F
           pf(sum_model3$fstatistic[1],sum_model3$fstatistic[2],sum_model3$fstatistic[3],lower.tail = F


# creating a dataframe containing the model statistics
statsData = data.frame(rsquare,adjrsquare,std_error,pvalue,row.names=c("model1","model2","modelLog"))
statsData
```

```
##            rsquare adjrsquare std_error        pvalue
## model1   0.1126309  0.1125130 1.5481944 1.513019e-197
## model2   0.1673432  0.1671219 1.4998065 6.211599e-300
## modelLog 0.1918583  0.1917509 0.3945698  0.000000e+00
```

Some additional comments

1. Correlation is not causation. Just because a relationship exists between two variables, it does not mean one causes the other to happen.
2. Linear (or non-linear) regression does not tell the whole story of your data. It will give you an idea of the relationship between different variables, but to understand that relationship, you often need additional research, thought and statistical analysis.

Relationships between 3 or more variables

In the above we've focused solely on the relationship between two variables. What happens if we want to consider a third, or even fourth variable?

We can use linear model as we did before, but include the other variables. For example, lets see if there is a relationship between chlorophyll fluorescence, temperature and salinity.

```
templog <- DATA$Temperature_C
fllog <- log10(DATA$Fluorescence)
sal <- DATA$Salinity_PSU

# removing the NaNs and Infs
nonans <- is.finite(fllog)
tempnonan <- templog[nonans]
flnonan <- fllog[nonans]
salnonan <- sal[nonans]

modelLog_ts <- lm(flnonan ~ tempnonan + salnonan)

summary(modelLog_ts)
```

```
##
## Call:
## lm(formula = flnonan ~ tempnonan + salnonan)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5070 -0.2559  0.0415  0.2374  1.1022
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.745492   0.138504   34.26   <2e-16 ***
## tempnonan     0.062826   0.001371   45.82   <2e-16 ***
## salnonan     -0.172025   0.004257  -40.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3577 on 7524 degrees of freedom
## Multiple R-squared:  0.336,  Adjusted R-squared:  0.3358
## F-statistic:  1904 on 2 and 7524 DF,  p-value: < 2.2e-16
```

We have an extra coefficient, so we end up with the following equation (or model):

$\log 10(chlorophyll\ fluorescence) = 0.063 \times Temperature - 0.172 \times Salinity + 4.745$

This model has increased our R^2 values, and reduced the residual standard error compared to when we looked at chlorophyll fluorescence and temperature only.

What about including day of year as well?

```
templog <- DATA$Temperature_C
fllog <- log10(DATA$Fluorescence)
sal <- DATA$Salinity_PSU
doy <- DATA$doy

# removing the NaNs and Infs
nonans <- is.finite(fllog)
tempnonan <- templog[nonans]
flnonan <- fllog[nonans]
salnonan <- sal[nonans]
doynonan <- doy[nonans]

modelLog_tsdoy <- lm(flnonan ~ tempnonan + salnonan +doynonan)

summary(modelLog_tsdoy)
```

```
##
## Call:
## lm(formula = flnonan ~ tempnonan + salnonan + doynonan)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5732 -0.2591  0.0178  0.2414  1.0723
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.101e+00  1.430e-01   21.69   <2e-16 ***
## tempnonan    9.178e-02  1.637e-03   56.07   <2e-16 ***
## salnonan    -1.133e-01  4.513e-03  -25.10   <2e-16 ***
## doynonan    -2.216e-03  7.614e-05  -29.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3391 on 7523 degrees of freedom
## Multiple R-squared:  0.4032, Adjusted R-squared:  0.403
## F-statistic:  1694 on 3 and 7523 DF,  p-value: < 2.2e-16
```

The R^2 values, and reduced the residual standard error have improved even more when the day of year is included in our model.

Assignment

Write up your answers to the following questions in a document and email it to me (julia@bigelow.org) by October 29th. Please also send me any R code (you can attach an R script to an email).

1. Correlation coefficients
   a. What is the correlation coefficient for two variables of your choice?
   b. What is the correlation coefficient when one or both (your choice) of your variables are log-transformed?
   c. Which set of variables are more highly correlated? The log-transformed data or the regular data? Justify your answer in 1 sentence.
2. Linear regressions
   a. Calculate the relationship between your two variables using a linear model. What are the summary statistics of this model? (You can copy and paste the `summary` output of the model into your write-up.)
   b. Calculate the relationship using a linear model on the log-transformed data. What are the summary statistics of this model? (You can copy and paste the `summary` output of the model into your write-up.)
   c. Which model describes your data the best? The model on the log-transformed data or the regular data? Justify your answer in 1 - 2 sentences.
3. Include a third variable in your analysis and calculate the relationship between your three variables using a linear model. You can choose whether or not to log-transform any of your variables. What are the summary statistics of this model? (You can copy and paste the `summary` output of the model into your write-up.)