Mod2_water

2023-11-14

Background for Module 2: Water Quality

Water is a valuable resource that is important for maintaining many aspects of a community's well being. "Water quality" captures many components of the chemistry, biology, and physics of fresh water. In this module, we will study data about two of these components, water temperature (T) and dissolved oxygen (DO).

Bull trout are a culturally important, but threatened species of fish native to the U.S. Northwest and the Canadian Rockies. They provide an example of how water quality affects community culture, because, to thrive, bull trout require water temperatures below 14 degrees C and dissolved oxygen levels above 6.2 mg/L. Such cold clean water conditions are growing scarcer due to climate warming and pollution.

It is also important to note that water temperature and levels of dissolved oxygen are related. As water temperature increases, dissolved oxygen levels decrease in water. This is concerning as climate warming has the power to increase water temperature, causing dissolved oxygen to dip to unhealthy levels for aquatic life.

[#JMc note: We need to check these threshold values for T and DO.]

In this module, we will use data about water quality trends to help K'avi Fish and Wildlife managers decide which of two streams would be best for a reintroduction of bull trout on tribal land. The tribe only has funding to bring the bull trout back to one stream. They have narrowed the choice of possible locations for the reintroduction to two streams, where temperature and D.O.have been monitored for about 15 years. Neither site is perfect, but they are both pretty good options. Your job is to analyze the temperature and D.O. data to help the tribe decide which is the best site.

The upstream source of Stream A is a groundwater spring. Water in this stream is very clean and temperatures are steady because groundwater temperatures tend to match average yearly temperature.

The upstream source of Stream B is an alpine lake that is fed by a glacier. The glacial melt-water is very cold and clean, but the glacier is shrinking as climate warms, so the cold-water input to the lake is growing smaller over time.

In this module, there are four main coding goals: 1. We will review skills you learned in Module 1, such as calculating summary statistics and making scatter plots.

2. We will learn how to rearrange data into a "tidy" format that makes graphing easier and we will learn how to "debug" code, in other words, we'll learn how to fix common problems that we see when using real world data.
3. We will learn how to make graphs of data over time. These are called "time series" graphs.
4. We will explore how water quality data can help communities make decisions to protect culturally important species.

#INSTRUCTOR'S VERSION: Learning goals and instructor resources
We will reach these goals in six steps 1. On day 1 of this module, we'll start by reviewing the usual steps for getting started: setting the working directory, bringing a dataset into RStudio, etc.

2. Next, we'll "wrangle" the dataset into a format that will make it easier to analyze in R. This includes organizing columns of data in a "tidy" way, merging two datasets, changing the names of variables, etc. Data wrangling is an important component of data science. It's a way of making sure the data you analyze is in a format that is reliable and complete. Students will use new commands, like pivot_longer() and left_join() to make the data "tidy", so it can be more easily reliable. There are many websites that can help you and your students learn these new tools and approaches. Here are a few to get you started: tidying data; pivoting data; joining data.
3. NEW SKILL: students will learn how to handle common code bugs, such as NAs.
4. After step 2, we will have the data in tidy form, and we can use familiar code to learn the summary statistics: means, minima, maxima, etc. We'll end Day 1 of the module using these statistics to make a preliminary assessment of whether Stream A or Stream B would be a better choice for reintroducing bull trout to K'avi land.
4. We will spend Day 2 analyzing patterns in the temperature and DO data from the two streams in more detail. First, we'll use familiar code to make a scatter plot of temperature and DO. This will help us visualize all of the data from both streams, which might give us insights we didn't see from the summary statistics. We can use this scatterplot to learn which stream had more years in the "danger zone" for bull trout, where temperatures were above 14 degrees C or dissolved oxygen was below 6.2 mg/L.
5. Next, we'll learn how to adapt our plotting code to show trends in temperature and dissolved oxygen over the 15 years that the K'avi have been collecting data from these streams. These "time series graphs" tell us whether water quality in the two streams is getting better, getting worse, or staying the same. This shows the adaptability of ggplot, and the ability to add graphical features. Here's a link with more information
6. Finally, you can use all the analyses you have conducted (summary statistics, scatter plots, and time series plots) to make recommendations to the K'avi fisheries managers about which stream is a better bet for supporting healthy bull trout populations on tribal land. How does each analysis help you understand patterns in water quality? Which analysis was most useful?
7. From the scientific analysis perspective, this module highlights why it is useful to visualizing data in multiple ways. There is one question: "Which of two streams has water quality most appropriate to maintain a healthy bull trout population?" But we ask it three times, after seeing summary statistics, then a scatter plot, then time series plots. The summary statistics and the scatter plot show that Stream B generally has better water quality than Stream A, but the time-series graph shows that both temperature and dissolved oxygen are getting worse for bull trout. The question of which stream is a better choice (the one with better values or the one without a dangerous trend) is for the class to discuss.

#DAY 1: Loading, merging, and tidying data

Preliminary steps Like we did in Module 1, before we being to analyze our data, we need to do a few steps first.

Clearing the Environment

Before we start to do anything in R, we should clear our environment. As a reminder, your environment is where R stores all the objects and data you were working with since the last time you worked in R. See the code chunk below on how to clear your environment.

```
rm(list = ls())
```

Setting the Working Directory

We will now set our working directory. As you may remember, setting our working directory tells R where to find the files we want to work with on our computer.

[JMc comment: We should decide as a class whether we want to set working directories this way, or just put the data files in the same directory as this .Rmd file. When you knit an R Markdown it looks for the data in that directory. See this link, too: https://github.com/yihui/knitr/issues/277#issuecomment-6528846]

```
current_path = rstudioapi::getActiveDocumentContext()$path
setwd(dirname(current_path))
```

Load in packages

Next, we will load in packages, to get the extra tools we need for this module loaded in R. For this module, we will load in the tidyverse package.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Load in our data

Now it's time to load in our data! In this scenario, water temperature data and dissolved oxygen data were collected and stored in two separate files. Therefore, we will need to load in both files in order to use them.

As a note, the units for temperature are Celsius (C) and the units for dissolved oxygen are mg/L.

```
stream_temp <- read.csv("streams_temperature.csv", header = TRUE, sep =",")
```

```
stream_DO <- read.csv("streams_DissolvedOxygen.csv", header = TRUE, sep =",")
```

Viewing the data

Now that our data is loaded into R, we can explore the data using the view and head commands. Our goal is to make sure things loaded correctly and to get a first look to see if the data make sense.

```
head(stream_DO)
```

```
##   year StreamA StreamB
## 1 2007    7.18      NA
## 2 2008    7.27    8.07
## 3 2009    7.46    7.26
## 4 2010    7.00      NA
## 5 2011    6.53      NA
## 6 2012    7.50    9.50
```

Class Question:

*What do you notice is different between using the command "View" and the command "head"? Which one do you think is more useful?

*It may also be helpful for us to understand how big our dataset is. We can get this number, called the "dimensions" of the data by using the command "dim".

```
dim(stream_temp)
```

```
## [1] 16  3
```

```
dim(stream_DO)
```

```
## [1] 16  3
```

Class Questions:

*How many data entries are in this dataset for water temperature?

*How many data entries are in this dataset for dissolved oxygen?

RECAP: You now reviewed ways to set up your R environment, read in data, and explore your data. These steps are all important for when we analyze our data later in this module.

Tidying datasets for more reliable analysis

As of now, our data for water temperature for Stream A and B exists in a separate data frame from the dissolved oxygen concentrations for Stream A and B. For K'avi Fish and Wildlife managers to truly understand what's going on, they need to analyze water temperature and dissolved oxygen together. To do this, we need to change how our data is set up in each data frame and then merge these data frames together.

Manipulating data so that it is formatted in a way that we can easily and reliably analyze is called "data wrangling". In R, we like our data to be "tidy". This means we don't want duplicate columns for the same variable. For example, in the stream_temp dataset, we have two columns for temperature, because we have a column for the water temperature of Stream A and the water temperature of Stream B. In a "tidy" version of thus dataset, we want all the temperature values to be in the same column.

HOWEVER, if we put all the temperature values in the same column, how will we know which stream the measurements came from? Take a moment to think about this, and see the answer below!

ANSWER We would want to make a column called "site" so when we read a row of data, we can see whether the water temperature or dissolved oxygen data was collected in Stream A or Stream B!

If you want to learn more about tidy data, here's a useful website.

But how do we make our data tidy?

First, we will use the pivot_longer() command to reorganize the columns from the temperature dataset, so that there is one column for "site" and one column for "temperature_C" The goal is to make only one column for each variable.

Class Question

*Look back at the current "messy" version of stream_temp. Can you see that there is more than one column with the same variable?

The pivot_longer() command allows us to combine columns, which transforms our dataset into a form with more rows (longer), and fewer columns (less wide). Long form data makes it easier to summarize data and look at trends over time, which allows the tidyverse package to work better.

In the code chunk below, we use pivot_longer() to transform the temperature data. The green #comments describe what each line of code does.

```
# The first line of code tells R that we want to re-define
# the streams_temp dataset with the changes we make below.
# This weird thing %>% is called the "pipe". It says that the code lines that come next,
# which are linked by a "+" at the end of each line,
# will all contribute to a transfomed version of stream_temp.

stream_temp <- stream_temp %>%
# We use the pivot_longer code to tell R which columns we want to change.
  pivot_longer(
    #cols = tells R which columns we want to modify
    cols = c(StreamA, StreamB),
    # names_to tells R that we want the names of the columns we specified
```

```
    # above to be entries in a new column we're creating called "site".
    names_to = "site",
    # Question: Based on what "names_to" means, what do you think the next line
    # of code values_to means?
    # (Hint - searching the internet for the answer is a great way to figure this out.)
    # Looking things up online is a huge part of coding!
    values_to = "temperature_C"
  )
```

To make sure this long command worked the way we intended, we should now view streams_temp:

```
print(stream_temp)
```

```
## # A tibble: 32 x 3
##     year site    temperature_C
##    <int> <chr>           <dbl>
##  1  2007 StreamA          13.2
##  2  2007 StreamB          10.2
##  3  2008 StreamA          12.5
##  4  2008 StreamB          NA
##  5  2009 StreamA          13.9
##  6  2009 StreamB          NA
##  7  2010 StreamA          11.4
##  8  2010 StreamB          11.7
##  9  2011 StreamA          13.0
## 10  2011 StreamB          NA
## # i 22 more rows
```

Great! Our code worked. Now let's do the same for the dissolved oxygen data set. We want to change it in the same way we changed the temperature data set above.

HOWEVER, this time some chunks of code have been left blank. See if you can fill them in based on what you saw in the example above!

NOTE This is filled in for the instructor's version, but will be left blank for students.

```
stream_DO <- stream_DO %>%
  pivot_longer(
    cols = c(StreamA, StreamB),
    names_to = "site",
    values_to = "dissolved_oxygen"
  )
```

Again, we should check our work to make sure that our data looks the way we want it to. We want there to be three columns, one for the year the sample was collected, one for the site name, and one for the dissolved oxygen values (mg/L).

```
print(stream_DO)
```

```
## # A tibble: 32 x 3
##     year site    dissolved_oxygen
##    <int> <chr>              <dbl>
##  1  2007 StreamA             7.18
##  2  2007 StreamB            NA
##  3  2008 StreamA             7.27
##  4  2008 StreamB             8.07
##  5  2009 StreamA             7.46
##  6  2009 StreamB             7.26
```

```
##  7  2010 StreamA             7
##  8  2010 StreamB            NA
##  9  2011 StreamA             6.53
## 10  2011 StreamB            NA
## # i 22 more rows
```

Good work! This data frame looks the way we want it to now as well.

Merging datasets

So far, we have separate datasets for temperature and DO. We want to combine them into one tidy dataset, so we can analyze trends in temperature and dissolved oxygen at the same time for both streams.

To do this, we will use a command in tidyverse called "left_join". Left join means we want to put one data frame next to the the other. In other words, we want to match up the rows in the data frame, so it becomes wider, not longer. This is because we want to preserve the columns from the two data frames we are combining.

```
# Here, we are using left_join to put the streams_temp data frame
# next to the streams_D data frame.
streams <- left_join(stream_temp,stream_DO)
```

```
## Joining with `by = join_by(year, site)`
```

Great! Now let's check to make sure that we successfully merged our two data frames.

```
print(streams)
```

```
## # A tibble: 32 x 4
##     year site    temperature_C dissolved_oxygen
##    <int> <chr>           <dbl>            <dbl>
##  1  2007 StreamA          13.2              7.18
##  2  2007 StreamB          10.2             NA
##  3  2008 StreamA          12.5              7.27
##  4  2008 StreamB          NA                8.07
##  5  2009 StreamA          13.9              7.46
##  6  2009 StreamB          NA                7.26
##  7  2010 StreamA          11.4              7
##  8  2010 StreamB          11.7             NA
##  9  2011 StreamA          13.0              6.53
## 10  2011 StreamB          NA               NA
## # i 22 more rows
```

Great job data wrangling! We now have a single tidy dataset that we can use to analyze water quality and determine which stream is best for introducing bull trout to tribal land.

Summary statistics

We know that bull trout prefer cold clean water. Temperatures above 14 degrees C and dissolved oxygen levels below 6.2 mg/L tend to be bad for bull trout, though these are only general approoximations and the healthiness of stream water for bull trout depends on many factors.

We'll start our analysis of water quality by using the statistical summary commands that will be familiar from Module 1. First, we'll calculate the mean (or average) water temperature

```
mean(streams$temperature_C)
```

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

Ah! Our code didn't work. R gave us an error message. Error messages are a normal part of programming. Although they might seem surprising or frustrating, you can think about them as a challenge for you to figure

out.

Here, R is telling you that the dataset has missing data, which is indicated in R by "the"NA", which means "not available". As you may have already noticed when looking at the "streams" dataset, you'll see that there are, indeed, some NA values.

It is very common to have missing data in datasets. Sometimes the data contains NAs in them by the scientists that entered the data (as in our case), but R will also enter NA in a row if there was no data there to begin with (it was left blank when the data was collected).

The reason this causes R to throw an error message is because the command mean() assumes that there are actual numbers in each cell that R is computing the mean of. R is telling you that it doesn't know how to calculate the value of something called "NA".

In the next section, we'll go over some easy ways to fix this problem in our "streams" data set by telling R to ignore the NAs.

The command na.omit() deletes any row that has NAs in it. The code below doesn't change the streams dataset, it just prints streams without any rows that have NAs.

```
na.omit(streams)
```

```
## # A tibble: 27 x 4
##     year site    temperature_C dissolved_oxygen
##    <int> <chr>           <dbl>            <dbl>
##  1  2007 StreamA          13.2             7.18
##  2  2008 StreamA          12.5             7.27
##  3  2009 StreamA          13.9             7.46
##  4  2010 StreamA          11.4             7
##  5  2011 StreamA          13.0             6.53
##  6  2012 StreamA          13.1             7.5
##  7  2012 StreamB          11.1             9.5
##  8  2013 StreamA          14.5             6.16
##  9  2013 StreamB          11.4             8.78
## 10  2014 StreamA          13.4             7.05
## # i 17 more rows
```

Great! Now let's try to run our command to get the mean for temperature again. We don't want to remove these rows from the streams data permanently, so we'll nest the na.omit() command within the mean() command, allowing us to calculate the mean of all the temperatures for both streams,

```
mean(na.omit(streams$temperature_C))
```

```
## [1] 12.63276
```

Our code worked! Now you've learned how to handle NAs in a dataset in R. Let's carry on with our summary statistics.

*Your turn! How would you figure out what the mean amount of dissolved oxygen is in the water across Stream A and Stream B?

```
#Write your code here!
```

Now lets use another familiar command to show the MEDIAN of temperature measured across all streams and years?

```
median(na.omit(streams$temperature_C))
```

```
## [1] 12.87
```

Class question: What is the median amount of dissolved oxygen across all samples? Try and fill in the code chunk below:

```
#Write your code here!
```

Since our goal is to compare the water quality of Stream A and Stream B, let's temporarily create separate dataframes for each stream using a new command called subset(). We explain the code and the commands we will use in the code chunk below.

```
# Below, we define a new dataframe "StreamA_WQ" as the subset of "streams" that have "StreamA" in the c
StreamA_WQ <- subset(streams, site == "StreamA")

# PRACTICE YOUR OWN CODE:
# See if you can make a new dataframe called "StreamB_WQ"
# that is the subset of "streams" that have "StreamB" in the column "site"
# WRITE YOUR OWN CODE BELOW:
StreamB_WQ <- subset(streams, site == "StreamB")
```

We created "StreamA_WQ" and StreamB_WQ" so that we could use the summarise() command that you saw in Module 1 to compare all the summary statistics for water temperature and dissolved oxygen between the two streams.

```
# print() just prints a line with whatever is inside the parentheses ().
# Here, we just want to separate Stream A statistics from Stream B statistics.
print("StreamA")
```

```
## [1] "StreamA"
```

```
# The summary() command is the same one you used in Module 1.
summary(StreamA_WQ)
```

```
##       year          site           temperature_C   dissolved_oxygen
##  Min.   :2007   Length:16          Min.   :11.36   Min.   :6.030
##  1st Qu.:2011   Class :character   1st Qu.:12.60   1st Qu.:6.478
##  Median :2014   Mode  :character   Median :13.13   Median :7.035
##  Mean   :2014                      Mean   :13.06   Mean   :6.918
##  3rd Qu.:2018                      3rd Qu.:13.38   3rd Qu.:7.287
##  Max.   :2022                      Max.   :14.47   Max.   :7.530
```

```
# Now we do the same thing for Stream B
print("StreamB")
```

```
## [1] "StreamB"
```

```
summary(StreamB_WQ)
```

```
##       year          site           temperature_C   dissolved_oxygen
##  Min.   :2007   Length:16          Min.   :10.20   Min.   :6.630
##  1st Qu.:2011   Class :character   1st Qu.:11.42   1st Qu.:7.300
##  Median :2014   Mode  :character   Median :12.03   Median :8.070
##  Mean   :2014                      Mean   :12.11   Mean   :7.931
##  3rd Qu.:2018                      3rd Qu.:12.87   3rd Qu.:8.350
##  Max.   :2022                      Max.   :13.78   Max.   :9.500
##                                    NA's   :3       NA's   :3
```

Preliminary analysis of water quality data from Stream A and Stream B

On Day 2 of this module, we will explore the water quality data in detail, and use this exploration of data to help inform the best decision about which stream is most promising for bull trout reintroduction. As a reminder, temperatures above 14 degrees C and dissolved oxygen levels below 6.2 mg/L tend to be bad for bull trout.

For now, let's use the information in the summary tables above to get a preliminary sense of which stream might have better water quality for bull trout.

Class Questions

Comparing the two tables, which stream seems better?

Which of the summary statistics do you think is most useful? Is there any other information you'd like to know in order to make the best recommendation to the tribe?

RECAP: In this section, we reviewed ways to combine datasets in a "tidy" way, remove NAs, reviewed how to run summary statistics, and introduced a technique to subset data so that we can look at data in different groups that may be useful for analysis.

Day 2: Detailed exploration of the water quality data with graphs.

The summary statistics we generated using the summary() command gave us an overview of how water temperature and dissolved oxygen differ between the two sites. Summary statistics are useful, and tribes are often asked to generate such statistics for reports (for instance the water quality reports tribes for the U.S. Environmental Protection Agency).

However, it is always a good idea to look at your data in many ways, including charts and graphs. Today, we will look at two different graphs of the water quality data, then revisit the question of which stream would be best for the bull trout reintroduction.

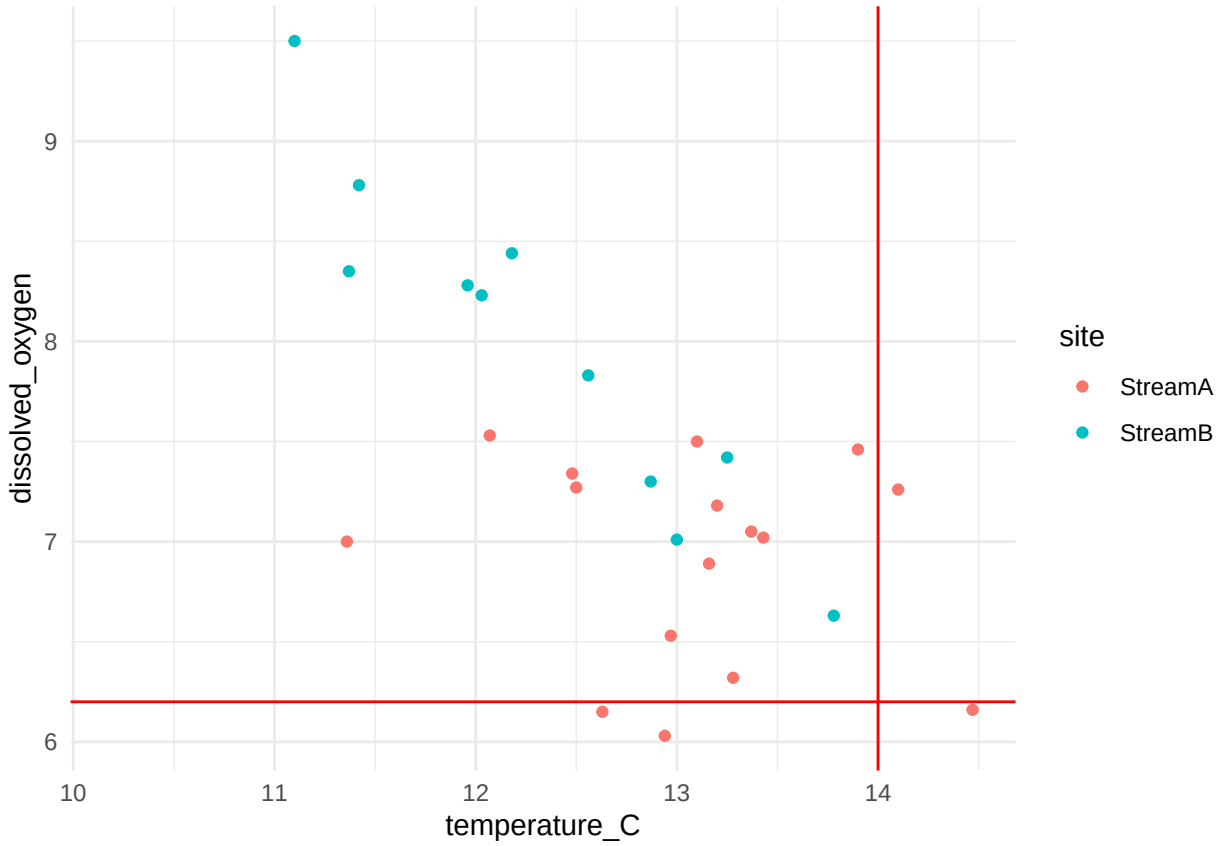
Scatter plot of water temperature and dissolved oxygen

Bull trout prefer water below 14 deg C and above 6.2 mg/L dissolved oxygen, but these thresholds are only part of the story. We can use a scatter plot to look at all the data for both streams to see patterns in the overall temperature and DO values are.

We'll use code that is similar to the scatter plot code in Module 1:

```
# The first line says we're making a plot using the streams dataframe.
# We're going to plot temperature_C against dissolved_oxygen.
# And we're going to color the symbols by site, distinguishing StreamA and StreamB.
ggplot(streams, aes(temperature_C, dissolved_oxygen, color = site)) +
  # Then we specify a scatter plot (points)
  geom_point() +
  # Then we'll add red lines depicting the levels of temperature and DO
  # that are dangerous for bull trout.
  geom_hline(yintercept = 6.2, col = "red") +
  geom_vline(xintercept = 14, col = "red") +
    # We'll use the same simple "theme" we used in module 1.
    theme_minimal()
```

```
## Warning: Removed 5 rows containing missing values (`geom_point()`).
```



Class Questions:

Where are the best values of water quality for bull trout? Where are the worst values?

When you ran the code chunk, you might have gotten a warning message that says, "Warning: Removed 5 rows containing missing values (`geom_point()`)." What does that mean? How does this affect your interpretation of the graph?

Does looking at this graph change your opinion about whether Stream A or Stream B will be healthier for bull trout? What other information would help the tribe make this decision?

Time series graphs of water quality

The scatter plot shows more detail than the summary statistics, but there is still more useful information in our data set. K'avi Fish and Wildlife managers have measured temperature and DO for 15 years, which is a long time. Since we know that temperatures are rising, it would be useful to see if these values change over time (we're looking for a trend).

To recap, Stream A is fed by a groundwater spring, and Stream B is fed by an alpine lake that is fed by a glacier. Because there sources of water are different, we can hypothesize that these streams may react differently to climate change.

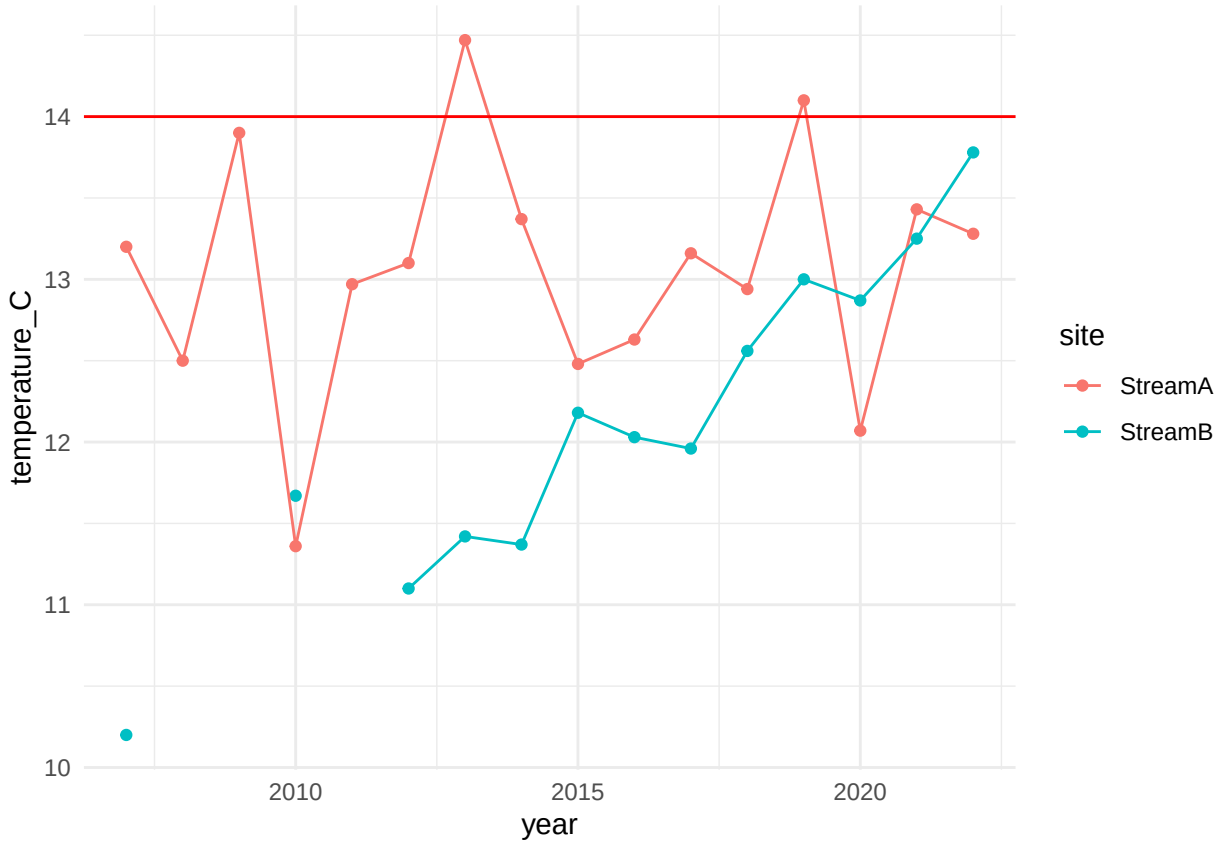
Class Question: Based on what we know about Stream A and Stream B, what would your hypothesis be on how the water quality for each stream may change as global temperatures increase with climate change?

We can assess our hypotheses by plotting water temperature and dissolved oxygen over time. We'll make a small change to our typical ggplot() code to make a "time series" plot, which shows trends across time.

First, we'll plot temperatures from both sites across the 15 years that the tribe has been monitoring water temperatures.

```r
# The first line says we're making a plot using the streams dataframe.
# We're going to plot temperature_C over time (year).
# And we're going to color the symbols by site, distinguishing StreamA and StreamB.
ggplot(streams, aes(year, temperature_C, color = site)) +
  # We start by plotting points (as we did in the scatter plot code)
  geom_point() +
  # Next, we add lines connecting adjacent points in time using geom_line()
  # Note how ggplot() allows you to add features like lines in a step by step way.
  geom_line() +
  # We'll add a line indicating the "danger zone" of temperatures above 14 C.
  geom_hline(yintercept = 14, col = "red") +
  # and we'll use the same theme as always.
  theme_minimal()
```

```
## Warning: Removed 3 rows containing missing values (`geom_point()`).
```



Class Questions:

What do the two colored lines represent?

Why are there breaks in the blue line?

Do the trends over time support your hypothesis about the stability of the temperature of the streams? Why or why not?

Using the code above as an example, see if you can make a similar time-series plot showing changes in dissolved oxygen for both streams over time:

```r
# Write your code here:
```

INSTUCTORS VERSION - this line of writing outline our intent with the plots Although Stream B tends to have had better water quality in the past, it seems that the data support our the hypothesis that the water quality of Stream B is getting worse, while the water quality of Stream A is fairly constant.

STUDENT VERSION

Class Questions:

Why do you think the water quality in Stream B is getting worse?

Which stream do you think will be healthier for bull trout in the future?

If you were using this analysis to make a recommendation to K'avi Fish and Wildlife managers, what would you suggest they consider as they decide which stream should be the site for bull trout reintroductions?

Beyond the case of bull trout reintroductions, what other water quality issues are important to tribes?

How can coding in R help inform tribal decisions about water quality?