Lab 1C: Simple Depth Profiles

2024-09-16

Today we are going to work on exploring data that you have recently collected on the first two sampling expeditions for this semester.

Today we will work to:

1. Create and export spreadsheets that are R-readable.
2. Import a spreadsheet into R as a data frame.
3. Plot variables by depth in R using ggplot2.

Formatting Data

Download the spreadsheet from Cruise 1 to your working directory and open it in Excel, Numbers or another spreadsheet viewing software.

- What are the strengths of this excel file?
- What do you see as some weaknesses of it?
- Could you load this sheet into R as-is?

Generally, let's discuss:
* What makes a good spreadsheet? * What are some common pitfalls of spreadsheets?

Spreadsheets in R

General tips for creating an R-readable spreadsheet:
* Keep it simple * One header row * Keep a separate tab or text file describing your data in detail. * Keep your original files!!

Things to consider:

- Split date columns by year, month and day to avoid confusion.
- Include columns for all variables, even if they are split up by sheets (e.g. by station/year etc).
- Keep columns consistent between sheets (to the best of your ability!).
- Avoid complicated headers (minimize keystrokes during analysis!).
- Don't carry out any cell operations within the spreadsheet… save that for R!
- Same applies for plotting!

Final tip: Make a plan about data collection at the *beginning* of a project!

Cleaning up our DARTs data

Note: * Each tab represents a variable (station) * File ID also provides important information (Cruise date, Project ID, Cruise Number)

1. Open a new, empty spreadsheet file, name it appropriately and save to a sea change directory
2. Identify columns and column names
3. Rename columns with simple headers, store new names in new tab
4. Start copying and pasting data over, noting station number as data is transferred
5. Export data as a csv:

Saving as a CSV

If you opened the file in Excel:

1. Go to "File" -> "Save As" and select to save the file as a "CSV (Comma delimited)" file named DaRTS_CTD_data (selecting "yes"" to the windows that pop up)
2. Exit the Excel file (selecting "no" you don't want to save changes if asked.)

If you opened the file in Numbers:

1. Go to "File" -> "Export To" -> CSV
2. Check the box that says "Create a file for every table". You are going to create a CSV file for every sheet (table) in the file. You will be asked to name the folder all the CSV files will be saved in - pick something sensible!
3. Once saved, exit the numbers file.

Loading your data into R

First we want to load the R libraries we'll be using today:

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## 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 error
```

This loads all the functions we will need to load and plot our data.

Let's try to read in the file we've just created. The function to do this is "read.csv"

```r
data <- read.csv("./seachange2024_DARTs_data_240916.csv", header = TRUE)
```

You can see this object show up on the left-hand ‘Environment’ tab in R studio. You can learn more about this object using a couple of different commands. For example, to learn about what the structure of the object is you can type use the function ‘str’

```
str(data)
```

```
## 'data.frame':    1258 obs. of  16 variables:
##  $ year       : int  2024 2024 2024 2024 2024 2024 2024 2024 2024 2024 ...
##  $ month      : int  9 9 9 9 9 9 9 9 9 9 ...
##  $ day        : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ project    : chr  "SeaChange_2024" "SeaChange_2024" "SeaChange_2024" "SeaChange_2024" ...
##  $ cruise_num : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ stn        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ depth      : num  0.78 0.787 0.789 0.79 0.792 0.791 0.79 0.791 0.796 0.801 ...
##  $ temp       : num  16.8 16.8 16.8 16.8 16.8 ...
##  $ salinity   : num  31.3 31.3 31.3 31.3 31.3 ...
##  $ density    : num  22.7 22.7 22.7 22.7 22.7 ...
##  $ par        : num  1400 1700 1670 1790 1640 1660 1490 1560 1430 1550 ...
##  $ fluor      : num  0.978 0.984 1.007 1.011 1.052 ...
##  $ turbidity  : num  1.16 1.15 1.15 1.15 1.15 ...
##  $ o2_umolPerKg: num  200 201 201 201 202 ...
##  $ o2_pctSat  : num  81.7 81.8 81.8 81.8 82.3 ...
##  $ ph         : num  8.02 8.03 8.03 8.03 8.03 ...
```

Other functions you can use to learn about the data object are “colnames” and “nrow”. Feel free to give those a try to see what they do.

Wrangling Data

The first plot we will want to make is a profile of temperature from Cruise 1, Station 1. So we will need to pull out all relevant rows of data that match those parameters. We will do this using a process called *filtering*. We will use the function “filter” from the package “dplyr.” It looks like this:

```
stn1 <- filter(data, cruise_num == 1, stn == 1)
head(stn1)
```

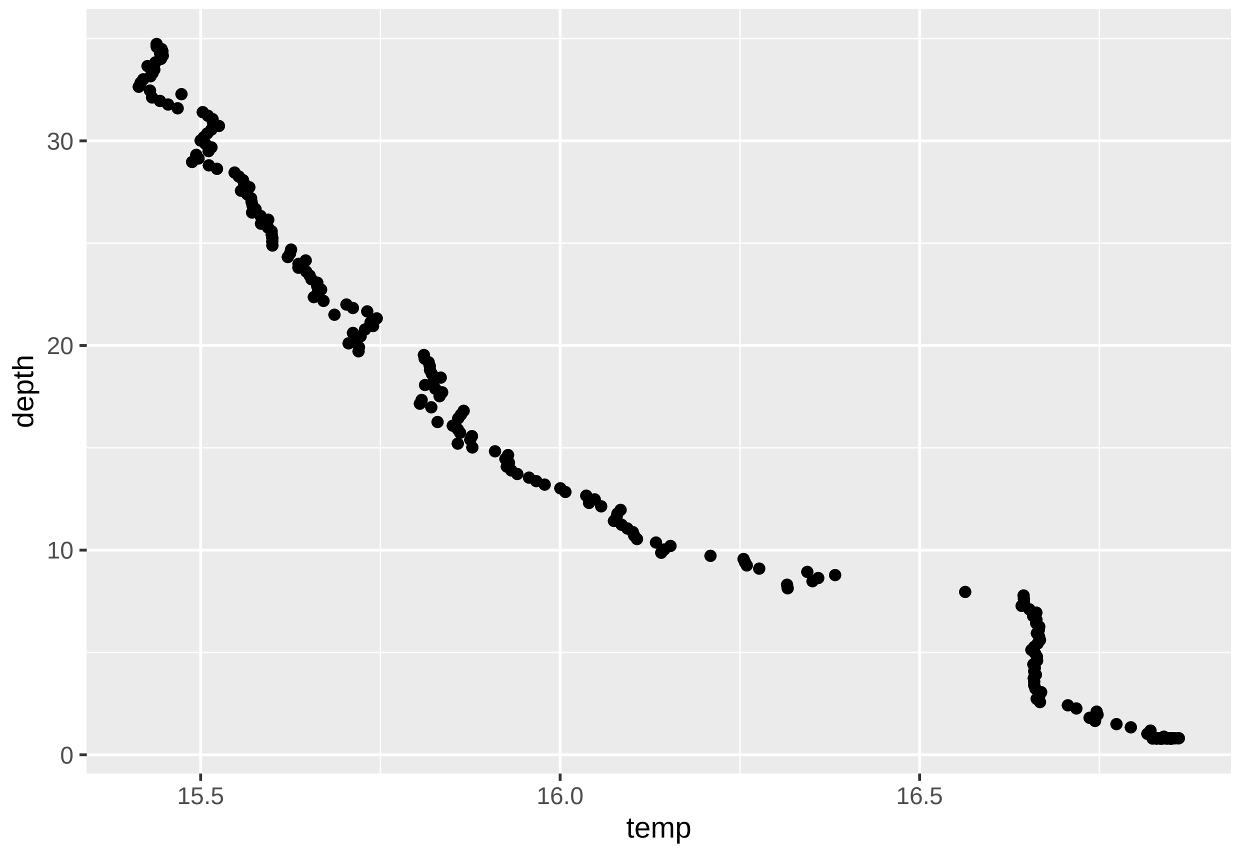
```
##   year month day        project cruise_num stn depth    temp salinity density
## 1 2024     9   5 SeaChange_2024          1   1 0.780 16.8495  31.3065 22.7164
## 2 2024     9   5 SeaChange_2024          1   1 0.787 16.8444  31.3016 22.7138
## 3 2024     9   5 SeaChange_2024          1   1 0.789 16.8368  31.3022 22.7160
## 4 2024     9   5 SeaChange_2024          1   1 0.790 16.8294  31.3028 22.7182
## 5 2024     9   5 SeaChange_2024          1   1 0.792 16.8240  31.3023 22.7190
## 6 2024     9   5 SeaChange_2024          1   1 0.791 16.8298  31.2936 22.7110
##    par  fluor turbidity o2_umolPerKg o2_pctSat    ph
## 1 1400 0.9783    1.1572      200.339    81.712 8.024
## 2 1700 0.9838    1.1543      200.690    81.845 8.025
## 3 1670 1.0066    1.1543      200.575    81.786 8.027
## 4 1790 1.0112    1.1543      200.642    81.802 8.029
## 5 1640 1.0515    1.1543      201.918    82.314 8.030
## 6 1660 1.0506    1.1514      203.769    83.073 8.031
```

The double equals sign is a *conditional*, it’s telling R to go through the data frame and identify rows where the condition of equality is met for those two variables.

Plotting Data

Next, let's plot the temperature variable. We're going to use a package called 'ggplot2'… we will go into greater depth in future R sessions about how exactly ggplot is working, for now, here's the code to plot a temperature profile:

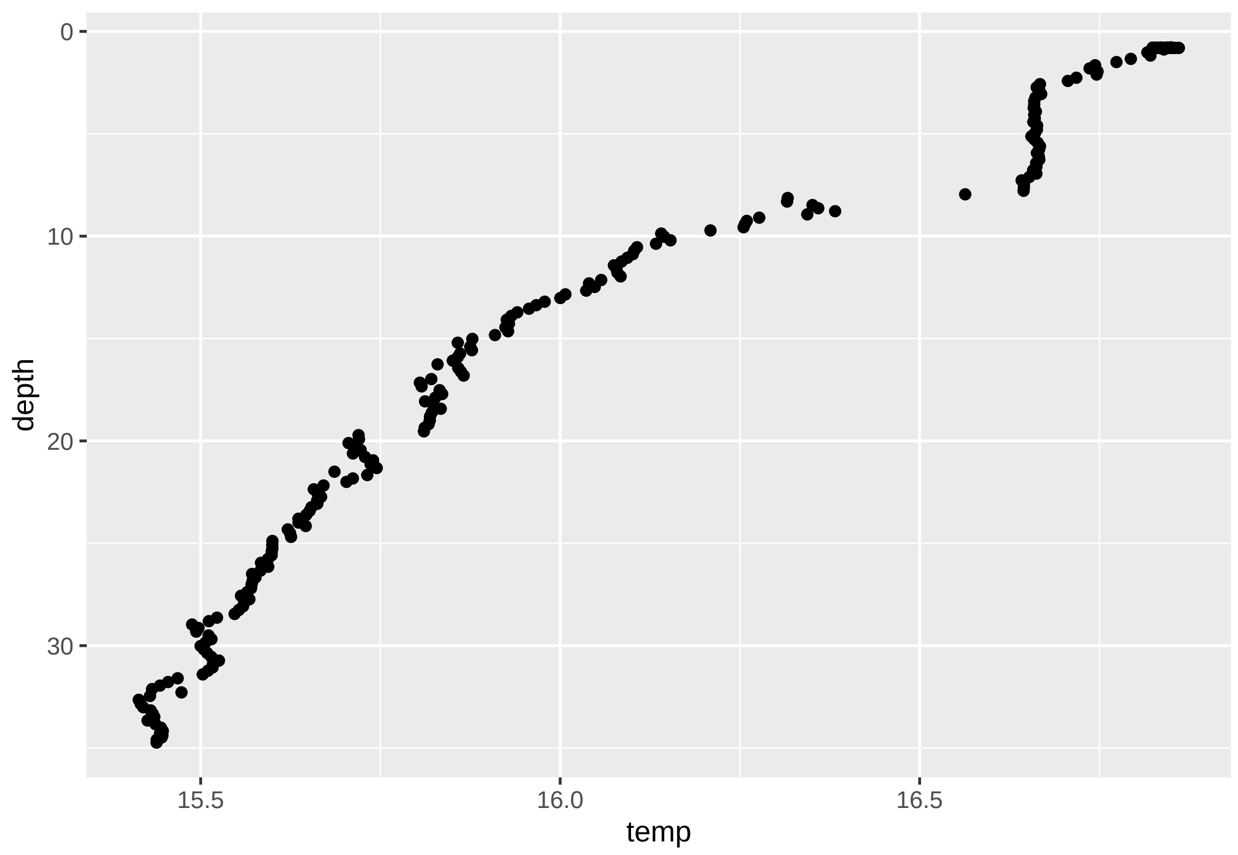
```
ggplot(stn1, aes(x = temp, y = depth)) + geom_point()
```



How does this look? What's wrong with this plot?

In oceanography, we like to orient ocean data from the surface on the top of the y axis and depth at the bottom of the y axis, so we need to add a line to our plot:

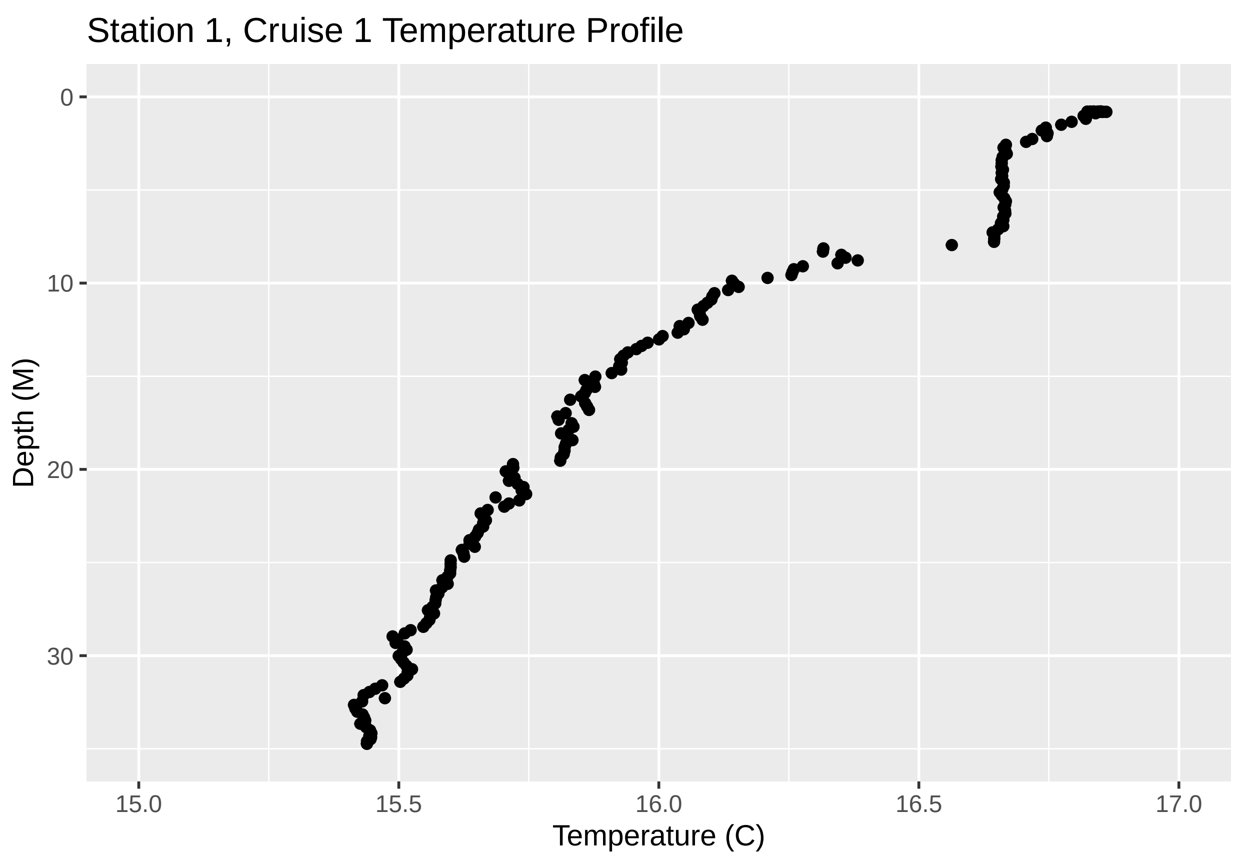
```
ggplot(stn1, aes(x = temp, y = depth)) +
  geom_point() +
  scale_y_reverse()
```



Next, let's make the axis labels more informative:

```
ggplot(stn1, aes(x = temp, y = depth)) +
  geom_point() +
  scale_y_reverse() +
  xlab('Temperature (C)') +
  ylab('Depth (M)') +
  xlim(15, 17) +
  ylim(35, 0) +
  ggtitle('Station 1, Cruise 1 Temperature Profile')
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```



Saving figures

To save a figure, use 'ggsave':

```
ggsave('st1_temp_profile.png', width = 4, height = 6)
```

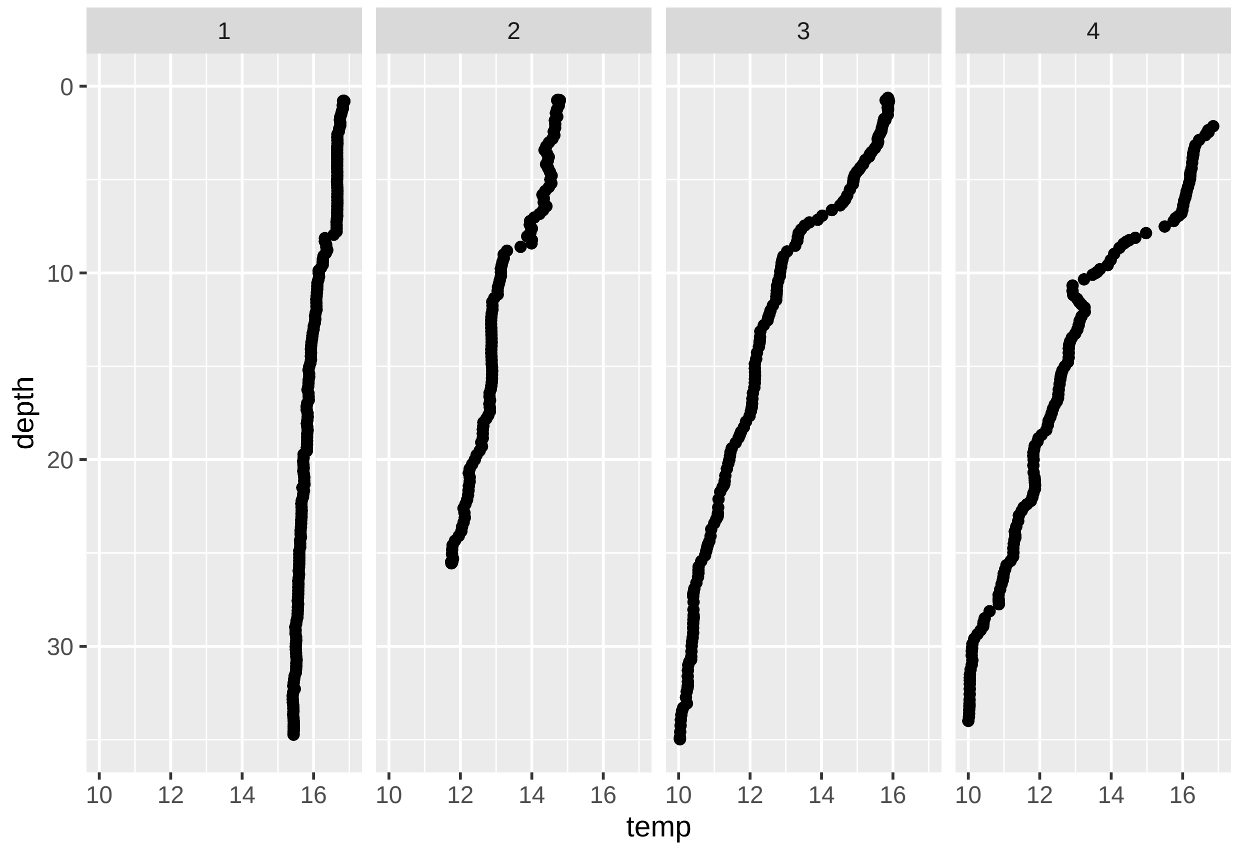
Plotting data from multiple stations

Faceting

```
ggplot(data, aes(x = temp, y = depth)) +
  geom_point() +
  facet_wrap(~stn, ncol = 4) +
  scale_y_reverse() +
  ylim(35, 0) + xlim(10, 17)
```

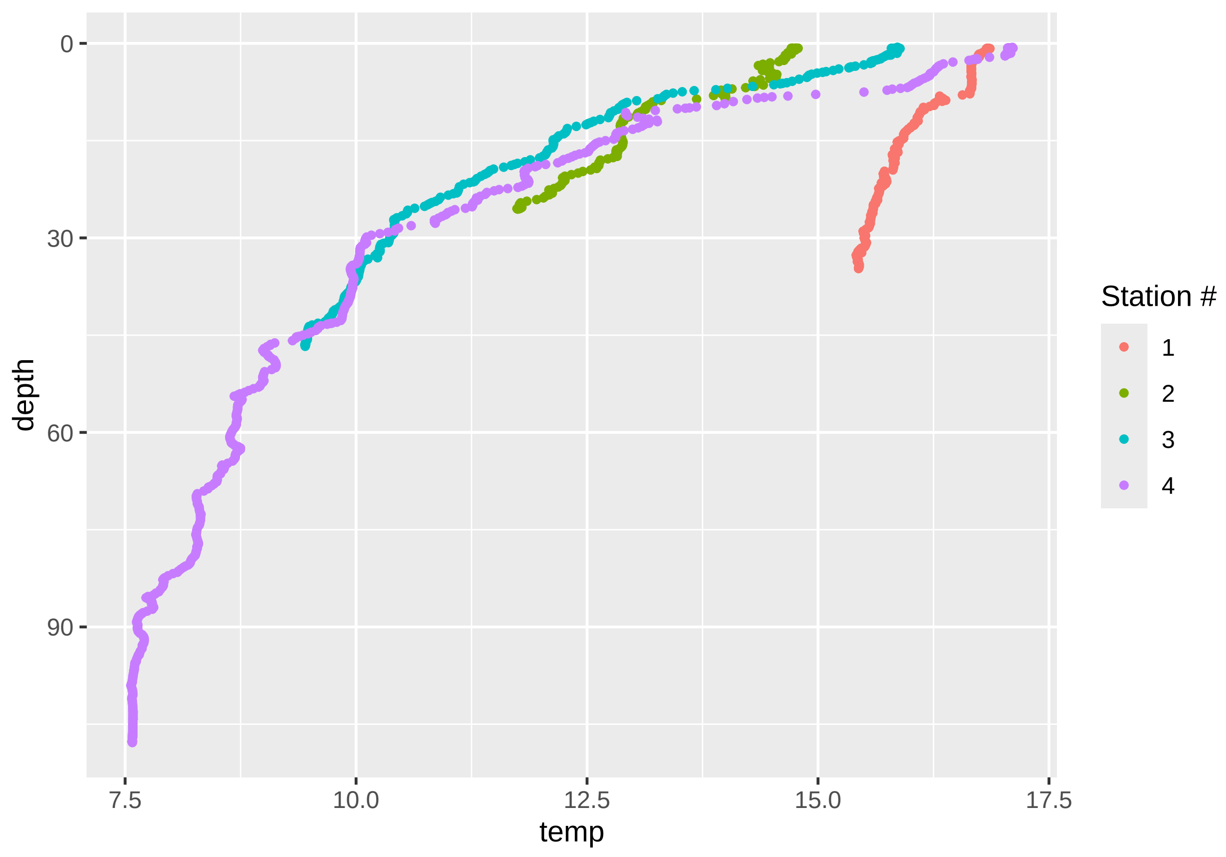
```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```
## Warning: Removed 521 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



```r
ggplot(data, aes(x = temp, y = depth, color = factor(data$stn))) +
  geom_point(size = 1) +
  scale_y_reverse() +
  labs(color = 'Station #')
```

```
## Warning: Use of ‘data$stn‘ is discouraged.
## i Use ‘stn‘ instead.
```



David's Assignment Part 1

1) plot temperature, salinity, density, chlorophyll, O2 (% Saturation) and pH in R with the Depth (m) on the vertical axis in reverse order.

Be sure to label the Axes and put in the correct units (Shown in S1 in the excel sheet).