Data visualization tutorial

Experiential Data science for Undergraduate Cross-disciplinary Education

Dr. Kim Dill-McFarland, U. of British Columbia

Contents

Data visualization tutorial

Learning objectives

- Define the grammar of graphics
- Create scatterplots using the `ggplot2` package
- Customize plot color, shape, axes, scales, and other attributes
- Represent subsets of data using facets

Setup

If you would like to follow along, open a new RStudio session and create a Project. Load the data file `Saanich_Data_clean.csv` using the following command in RStudio. If you would like to save code/notes, also start a new R script.

```
dat <- read.csv(
  "https://raw.githubusercontent.com/EDUCE-UBC/workshop_data/master/Saanich_Data_clean.csv")
```

Not sure what a Project is? Be sure to include the "RStudio tutorial" in your materials!

If you would like to learn more about Saanich Inlet and these data, checkout our description as well as how these data were cleaned in our 'Data manipulation in R tutorial'.

Install `tidyverse`

In order to use the functions in the tidyverse like those in the `ggplot2` package, you must first install these packages. (More information available in our 'RStudio tutorial'.)

```
install.packages("tidyverse")
```

*Please note that if you have **R v3.3 or older**, you may not be able to install `tidyverse`. In this case, you need to separately install each package within the tidyverse. This includes:* `readr`, `tibble`, `dplyr`, `tidyr`, `stringr`, `ggplot2`, `purr`, `forcats`

Load `tidyverse`

Installing a package downloads the relevant files onto your computer, but it does not allow R/RStudio to access the functions therein. Thus, you must load packages into RStudio using the `library()` function.

This may seem tedious but it is necessary every time you open a new RStudio session. Otherwise, the program would load every package you had ever downloaded every time it is opened. With 10s of 1000s of packages out there, this would significantly slow down RStudio. Thus, only load the packages you will need for a given project.

For this tutorial, you will need:

R v3.4 or newer

```
library(tidyverse)
```

R v3.3 or older

```
library(readr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.5.2
```

```
library(ggplot2)
```

Graphics with `ggplot2`

ggplot2 is an add-on package to R and part of a suite of data manipulation and visualizations packages called the tidyverse. It is an alternative to base graphics that has become very popular, to the point where it is preferred for most data types.

ggplot is based on the *Grammar of Graphics* (Wilkinson, L. 1999. doi: 10.1007/978-1-4757-3100-2). It provides users with a number of advantages over base R graphics including:

- handsome default settings
- snap-together building block approach
- automatic legends, colors, facets
- statistical overlays like regressions lines and smoothers with confidence intervals

The building block approach works with:

- data: 2D table (`data.frame`) of *variables*
- *aesthetics*: map variables to visual attributes (*e.g.* position)
- *geoms*: graphical representation of data (points, lines, etc.)
- *stats*: statistical transformations to get from data to points in the plot (binning, summarizing, smoothing)
- *scales*: control *how* to map a variable to an aesthetic
- *facets*: juxtapose mini-plots of data subsets, split by variable(s)
- *guides*: axes, legend, etc. reflect the variables and their values

and the idea is to independently specify and combine these blocks to create the plot you want.

To build a plot, you must provide at least:

1. data
2. aesthetic mappings from data variables to visual properties
3. a layer describing how to draw those properties

Data motivation

Before we begin making plots, let's first consider our data and the purpose of this data exploration. Here, we will investigate the relationship between **oxygen (O_2), nitrate (NO_3), and hydrogen sulfide (H_2S)** in Saanich Inlet. We are particularly interested in these relationships, because oxygen is the most energetically favorable terminal electron acceptor for microbial metabolism in this system. However, oxygen is depleted in deeper waters as a result of the seasonal stratification.

So, what are microbes using to survive? As oxygen is depleted, microbes must use alternative terminal electron acceptors. In Saanich Inlet, nitrate is the next best option, so we want to see if nitrate is available to microbes when oxygen is depleted. We are also interested in hydrogen sulfide, because sulfate is the third best terminal electron acceptor in this system. If sulfate is being used, it will be reduced to hydrogen sulfide, and we will see a build up of H_2S when both oxygen and nitrate are depleted.

Let's test these hypotheses by plotting the data!

geom_point

Let's begin by exploring scatterplots in ggplot using `geom_point`.

The first argument of ggplot is the data. We can either specify this argument within the ggplot function

```
ggplot(dat)
```

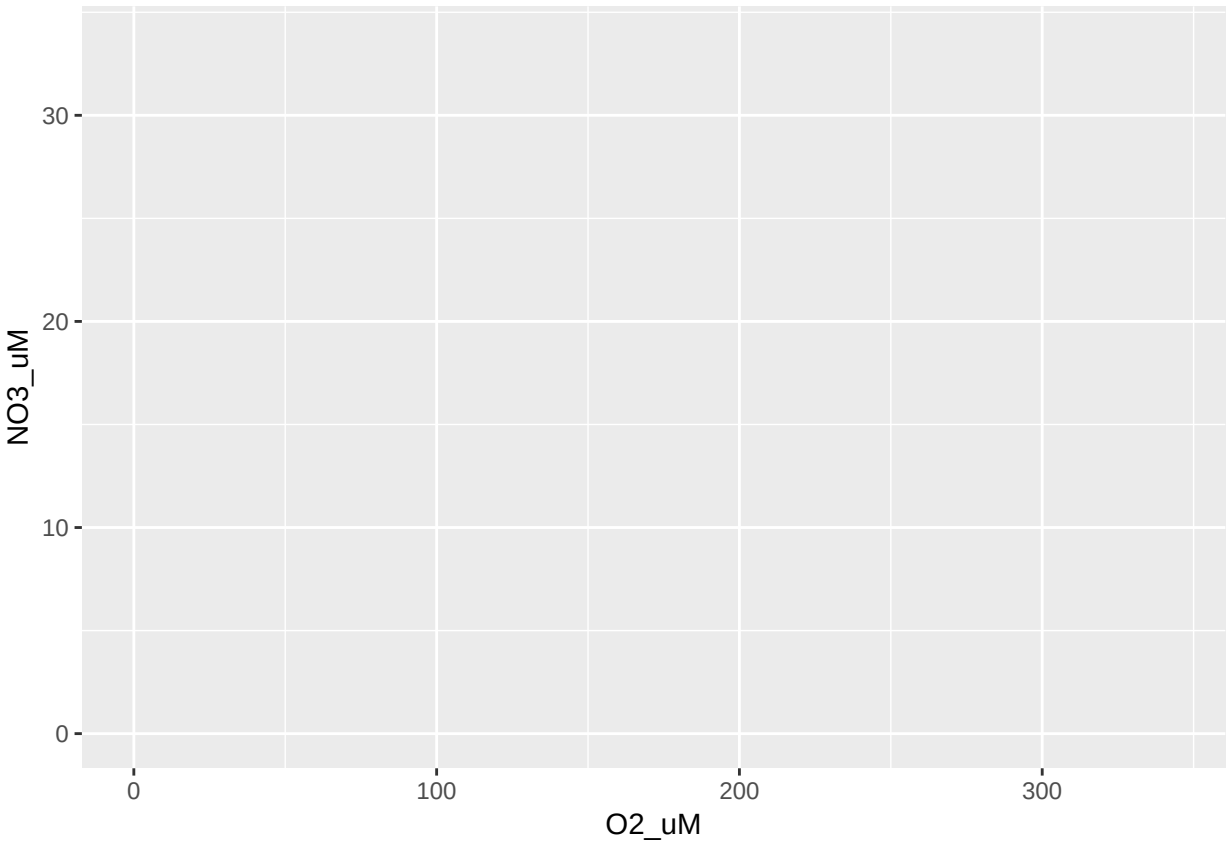
or pipe it in like so.

```
dat %>% ggplot()
```

Not familiar with pipes? Checkout our 'Data manipulation in R' tutorial.

The second argument is the aesthetics `aes`, where we specify visual attributes of our plot like the x- and y-variables.

```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM))
```



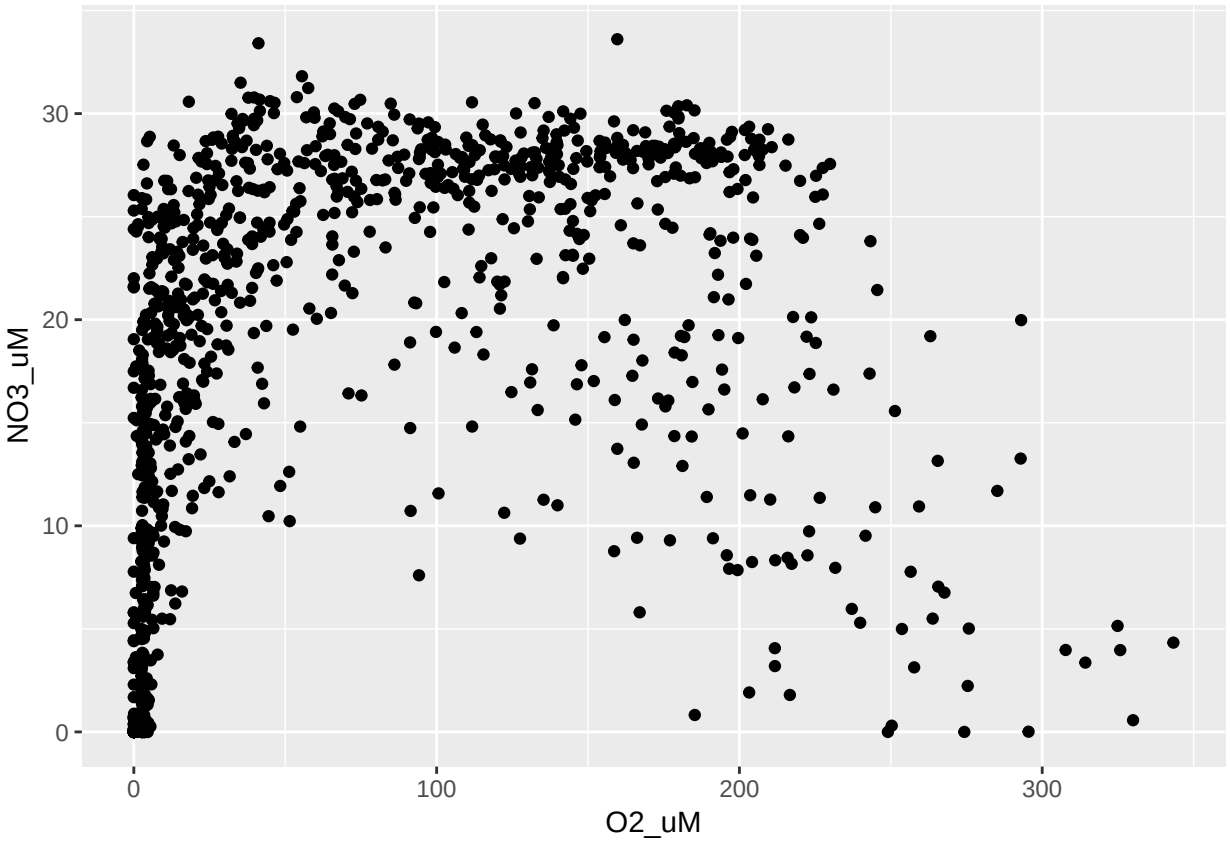
Finally we add the geom to specify how we want to map our data onto these axes. If we don't specify any aesthetics in the geom, it will take them from the main ggplot function.

```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point()
```

But we could specify them in the geom. Importantly, aesthetics in `ggplot` will be applied to all layers while those in a specific geom will only be applied to that layer.

```
dat %>%
  ggplot() +
  geom_point(aes(x=O2_uM, y=NO3_uM))
```

```
## Warning: Removed 131 rows containing missing values (geom_point).
```

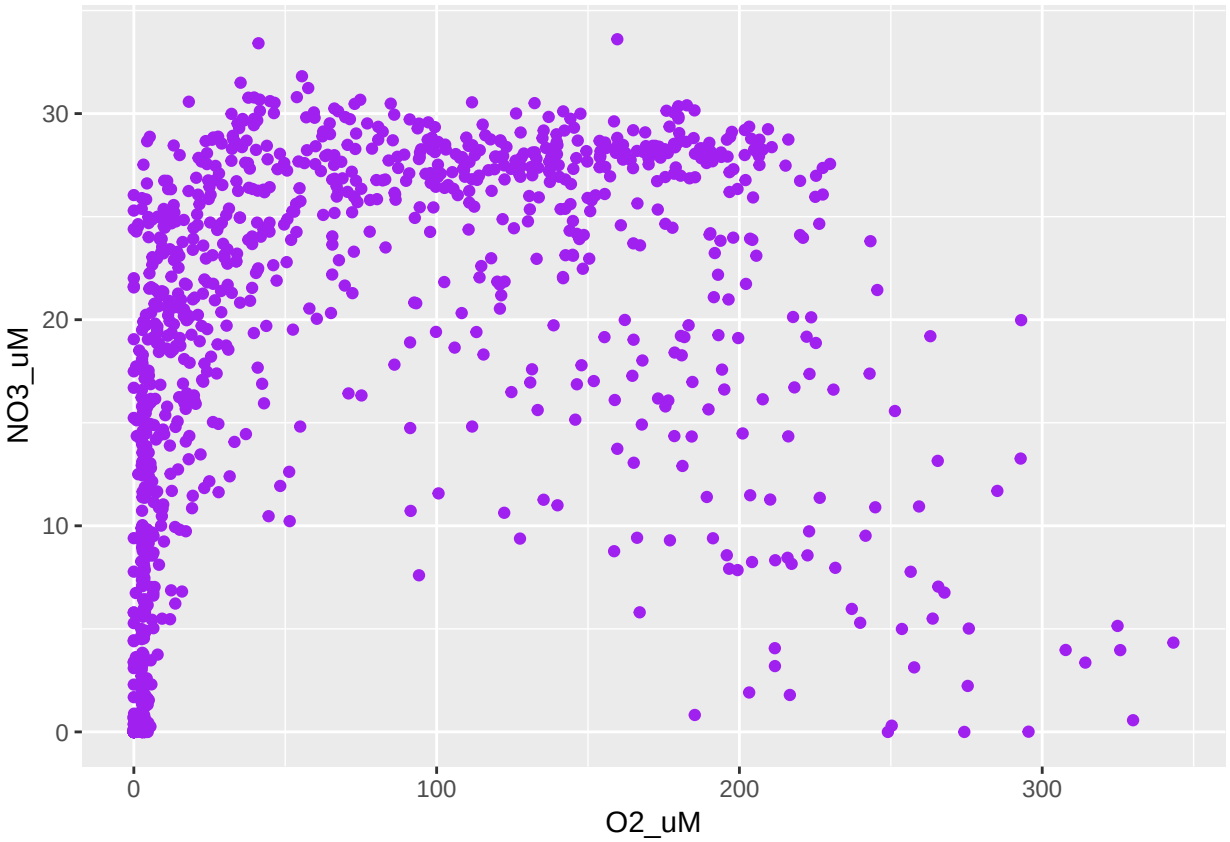


Note that above R code has a warning. Warnings do not always indicate that there is an error in your code. For example, this warning tells us that there is missing data (NA) in our data frame.

Add color

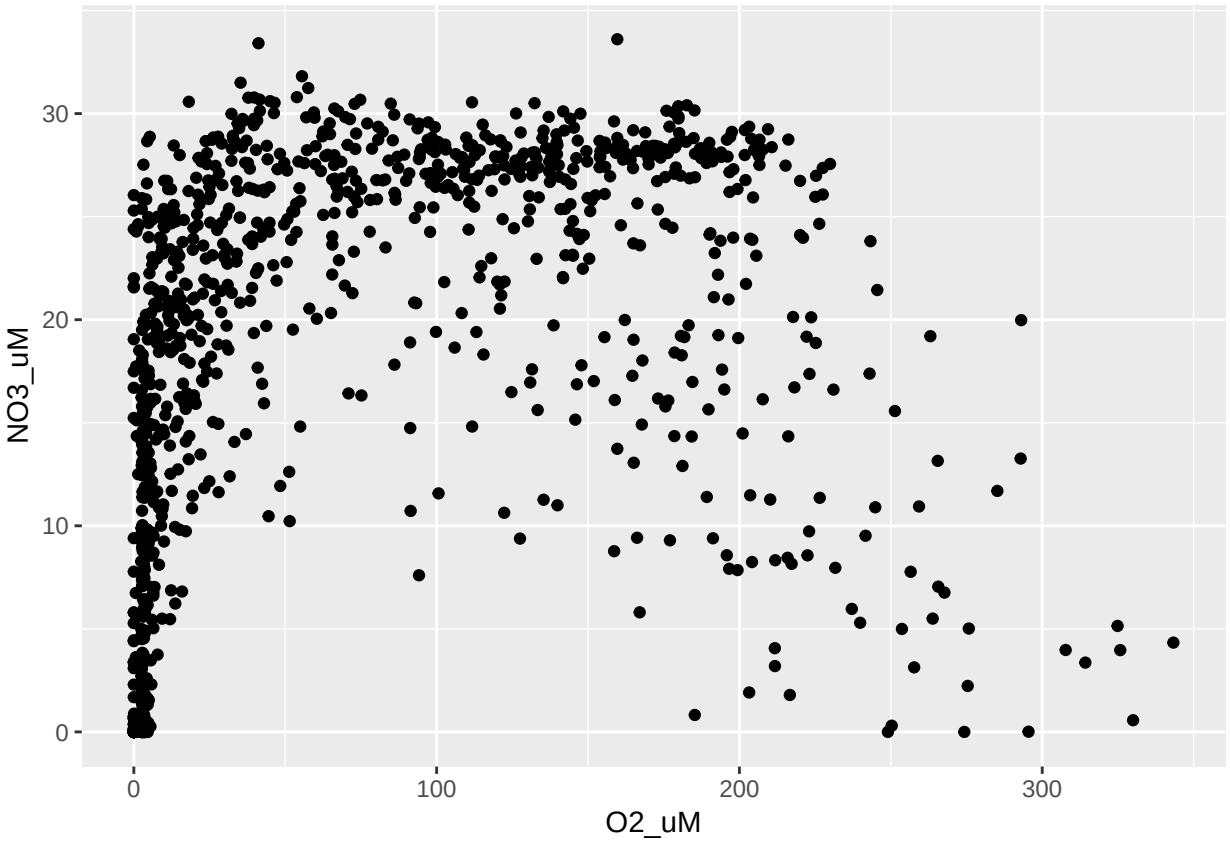
We can begin to customize our plot by adding color. You can simply change the color of the data points like so. *Note that the tidyverse understands both 'color' and 'colour'.*

```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(color="purple")
```



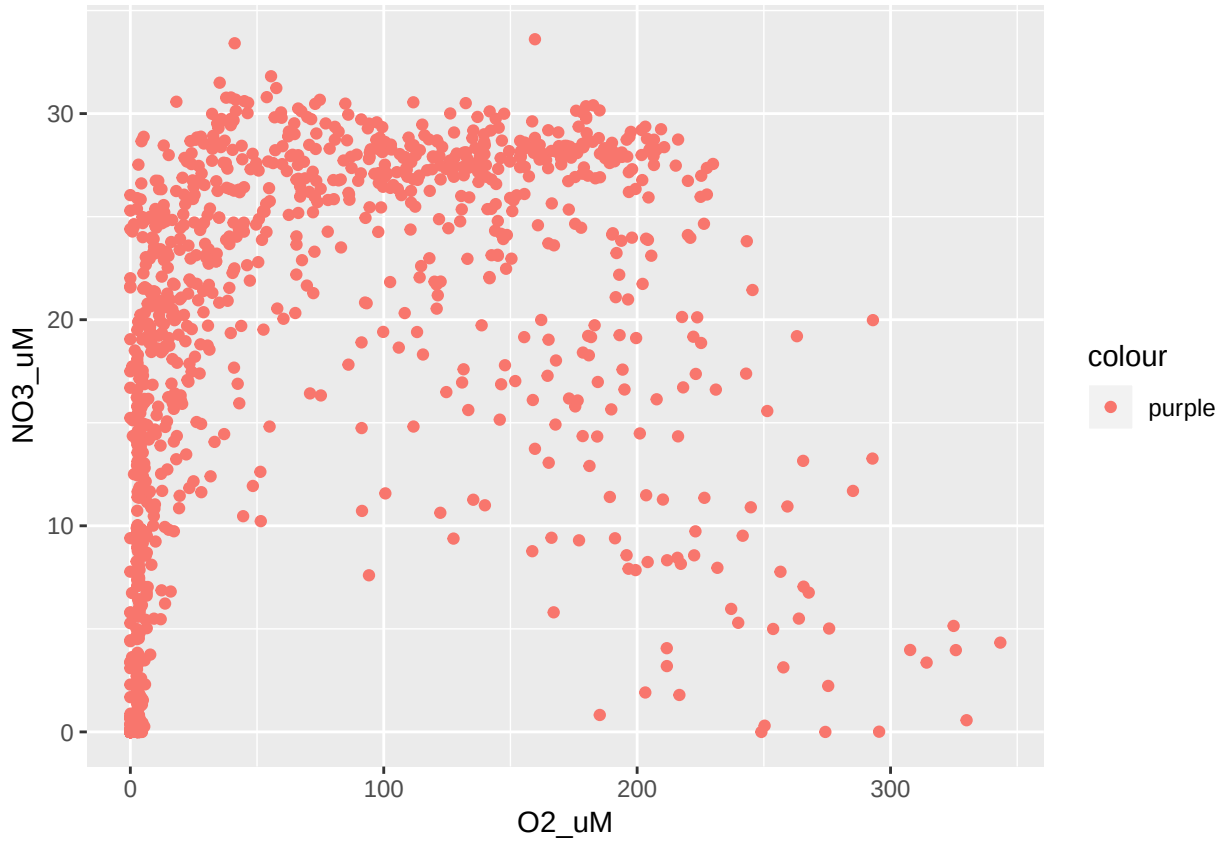
There are two important things to note here. First, since we are changing the color of the points, we must specify the color within `geom_point`. If we were to instead specify color in the main `ggplot` function, there are no points (yet) to color and it fails.

```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM), color="purple") +
  geom_point()
```



Secondly, color is outside of the `aes` argument because it is not an aesthetic. Aesthetics *must* refer to some variable in the data. So, if we put the world "purple" as a color aesthetic instead, ggplot things that we want to color our data by the group "purple" and will use its default color to do so.

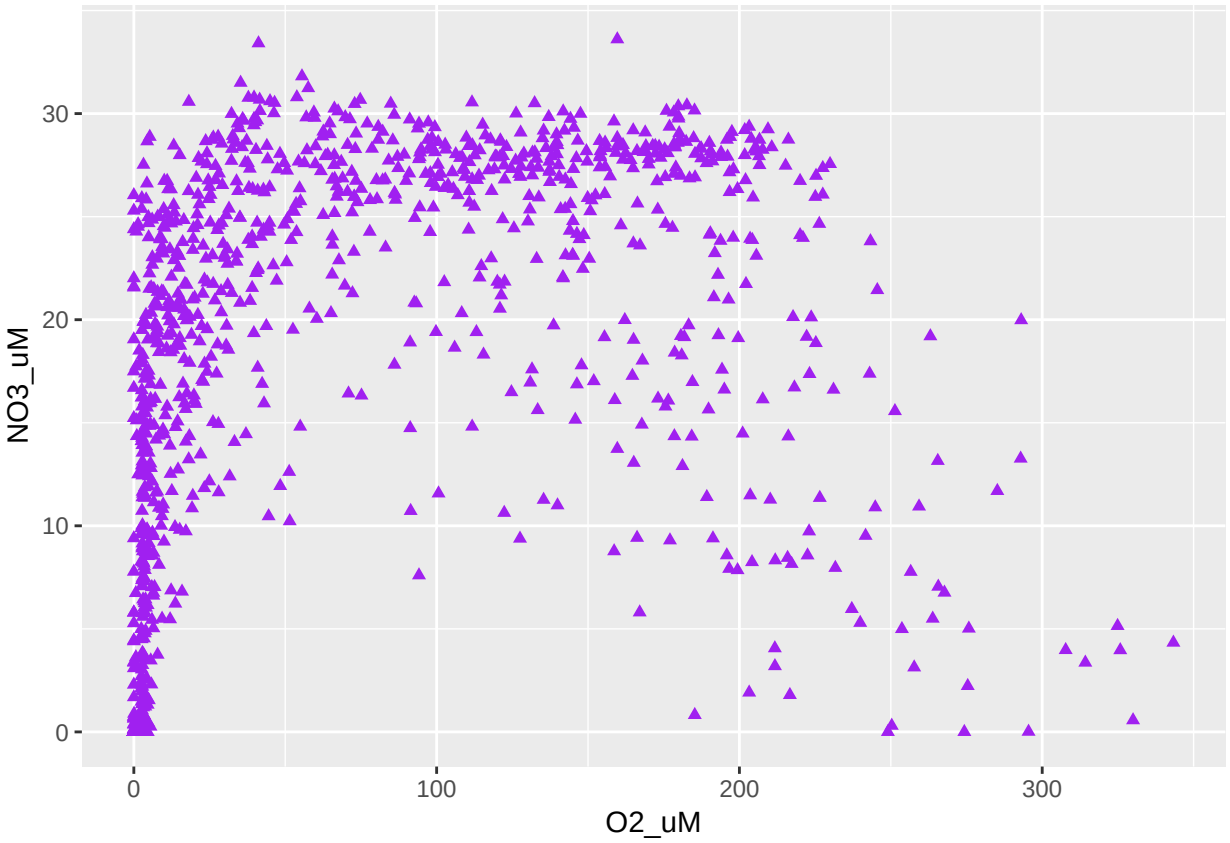
```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(aes(color="purple"))
```



Add shape

Similar to color, we can change the shape of the points. R has a number of shorthand codes for shapes which can be found here.

```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(color="purple", shape=17)
```



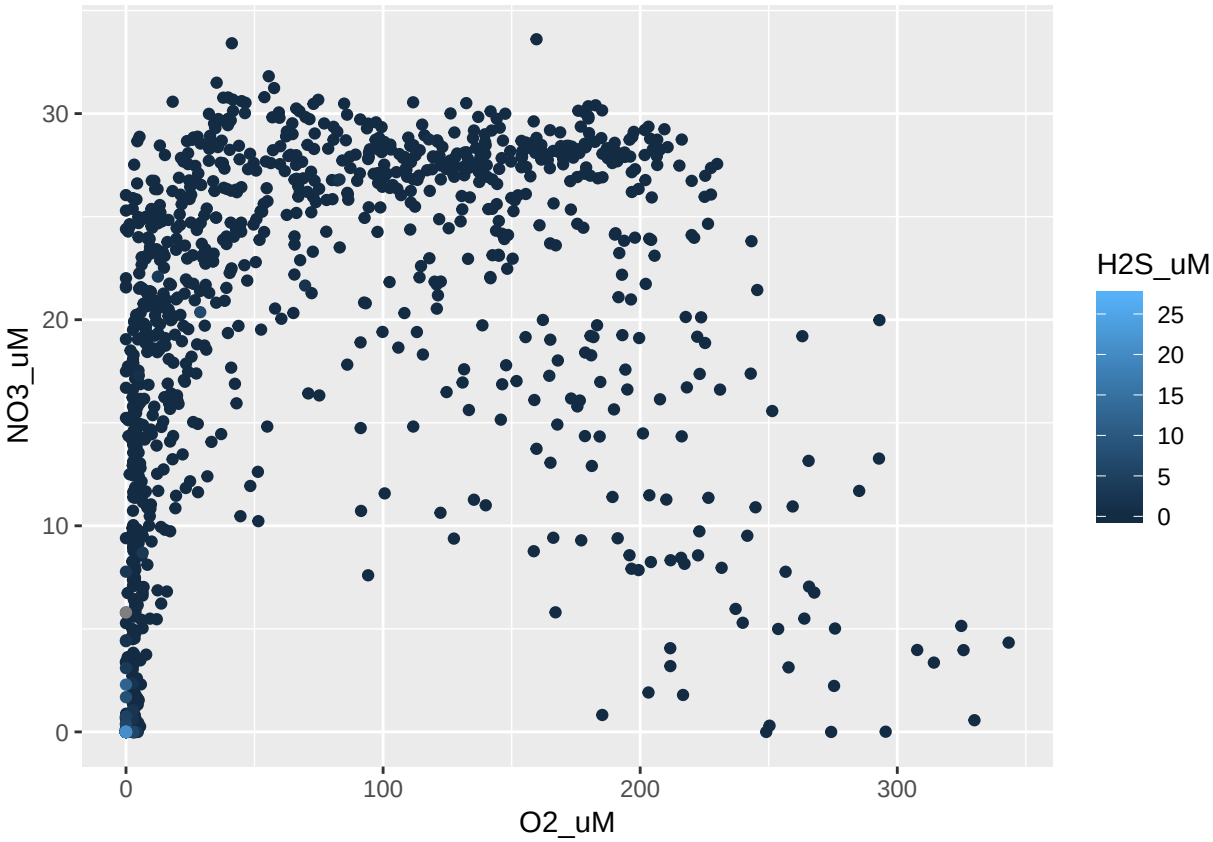
Mapping variables to color

Thinking back to our motivating question in regards to oxygen, nitrate, and hydrogran sulfide in Saanich, perhaps we can get a better understanding of what is happening if we instead plot all 3 variables of interest on 1 plot.

One way to achieve this is to map the third variable as a color *aesthetic* on a plot of the other 2 variables like so.

As before, we want to place the color in the `geom_point` layer since it is the points that we want to color. However, in contrast to changing all points to a single color, using a variable to color the points is an *aesthetic* because it relies on data from the data set. Thus, color must now be contained within `aes`.

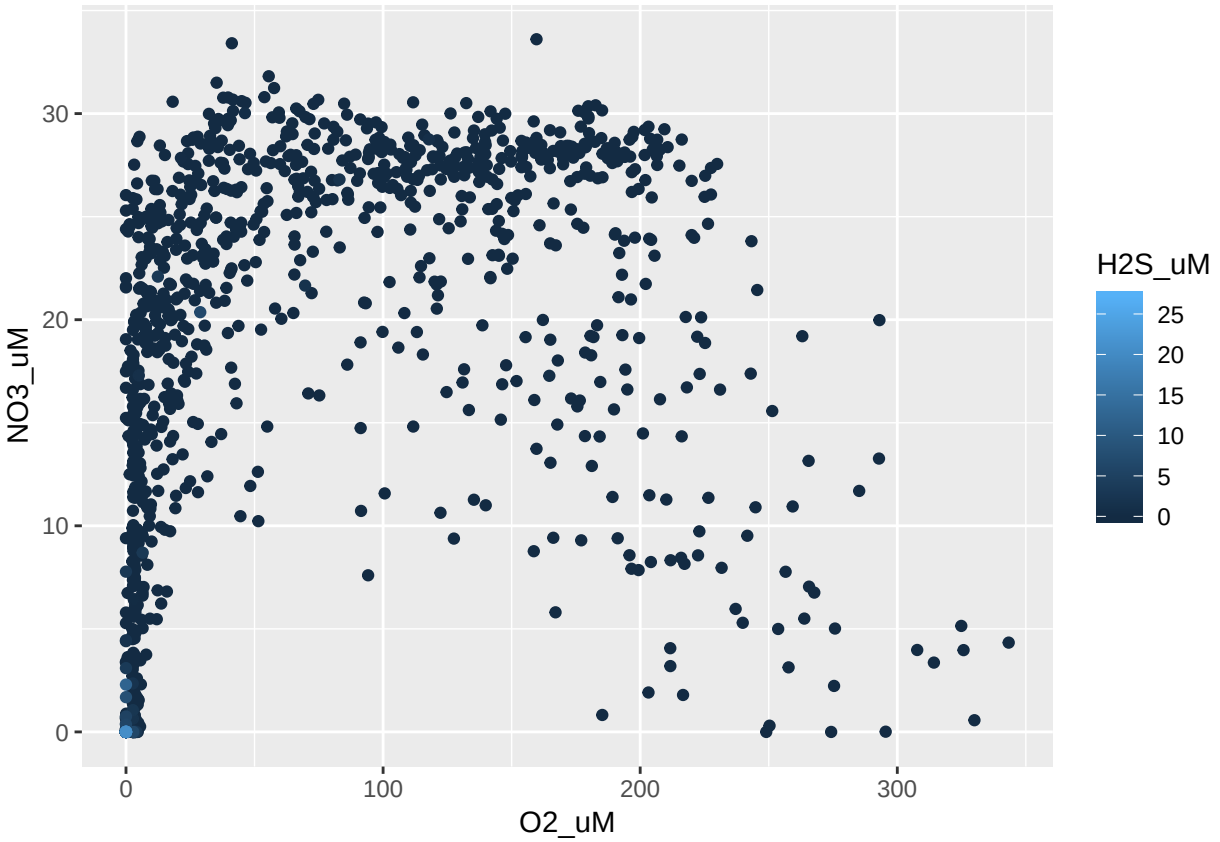
```
dat %>%
  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(aes(color=H2S_uM))
```



However, ggplot plots the observations in the order that they appear in the data frame. Since most of the hydrogen sulfide values are zero, we want to plot the non-zero data *last* so that they are visually on top in the plot.

To do this, we use `arrange` (a `dplyr` verb) by this variable to sort the rows by ascending hydrogen sulfide values. We also remove observations without hydrogen sulfide data (NA) since these will not map to any color. (See the 'Data manipulation in R' tutorial to learn more.)

```
dat %>%
  arrange(H2S_uM) %>%
  filter(!is.na(H2S_uM)) %>%

  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(aes(color=H2S_uM))
```



This is a great example of why we use a pipe to input data into `ggplot`. Doing so allows us to manipulate `dat` using other tidyverse functions and then input these modified data directly into a ggplot without saving an intermediate data frame in the R environment.

Mapping variables to shape

We could also map a third variable to shape. However, this only works for categorical data so mapping to hydrogen sulfide fails.

```
dat %>%
  arrange(H2S_uM) %>%
  filter(!is.na(H2S_uM)) %>%

  ggplot(aes(x=O2_uM, y=NO3_uM)) +
  geom_point(aes(shape=H2S_uM))
```

```
## Error: A continuous variable can not be mapped to shape
```

So, we need to rethink how we are plotting these data if we want to use shape. One option is to instead plot all 3 variables as shapes along depth. We would do so with the following setup. But what are our x and shape variables?

```
dat %>%
  ggplot(aes(x=?, y=Depth_m)) +
  geom_point(aes(shape=?))
```

We actually have multiple variables that we want to input for x (oxygen, nitrate, hydrogen sulfide) and no variable to use for the 3 shape groups we want to see. Thus, we need to `gather` our 3 geochemical variables into a single column.

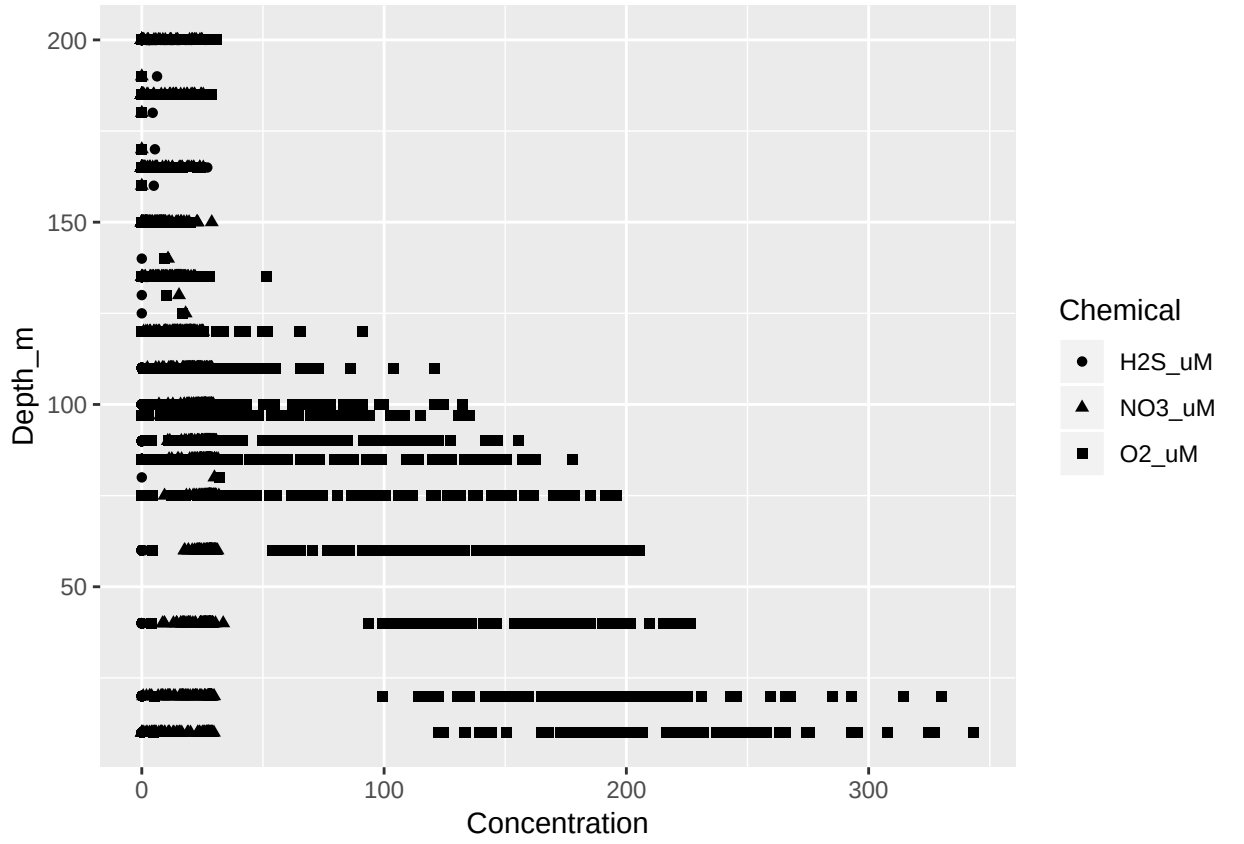
```
dat %>%
  select(Depth_m, O2_uM, NO3_uM, H2S_uM) %>%
  gather(key="Chemical", value="Concentration", O2_uM, NO3_uM, H2S_uM)
```

```
##   Depth_m Chemical Concentration
## 1      10    O2_uM       225.237
## 2      20    O2_uM       220.985
## 3      40    O2_uM       201.769
## 4      60    O2_uM       197.947
## 5      75    O2_uM       193.762
## 6      85    O2_uM       150.404
```

More on ***gather*** *in 'Data manipulation in R' tutorial*

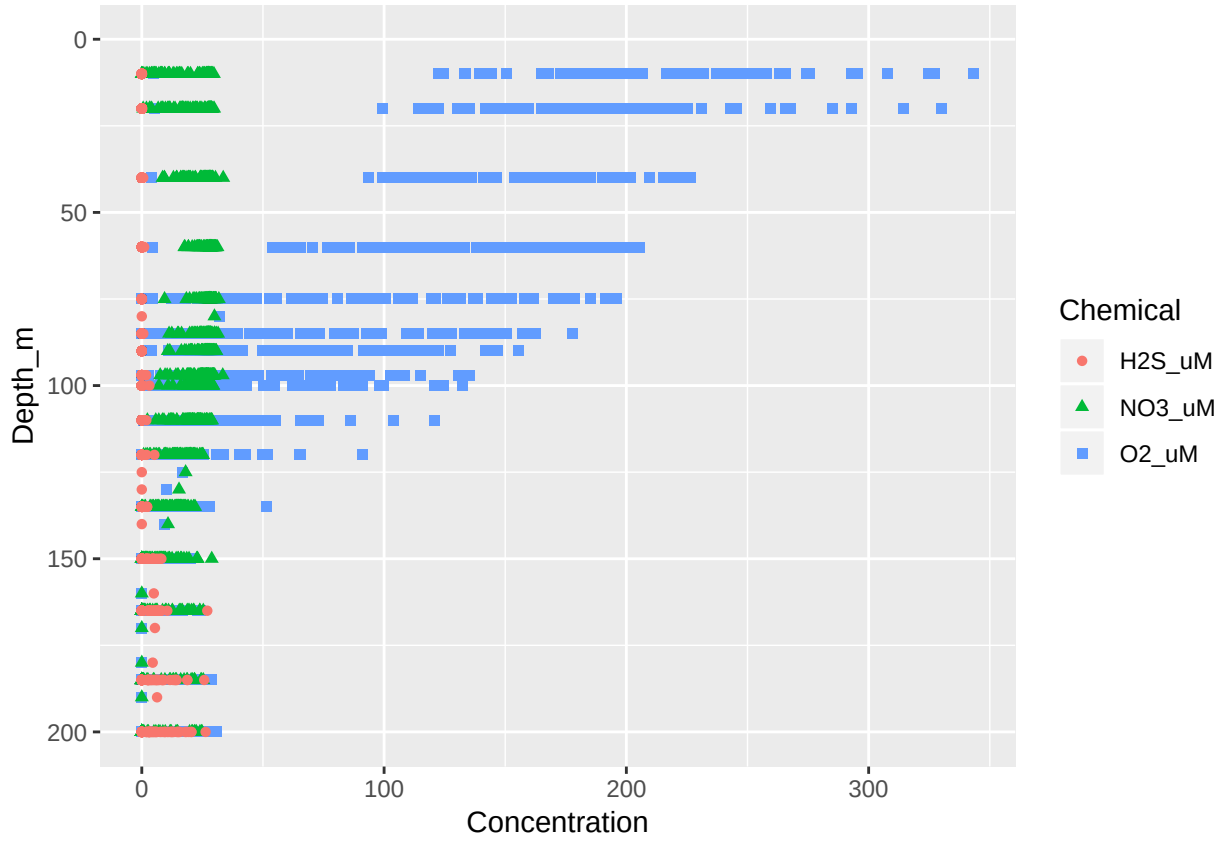
If we pipe these modified data into our plot, we now have individual variables to use as x and shape.

```
dat %>%
  select(Depth_m, O2_uM, NO3_uM, H2S_uM) %>%
  gather(key="Chemical", value="Concentration", O2_uM, NO3_uM, H2S_uM) %>%

  ggplot(aes(x=Concentration, y=Depth_m)) +
  geom_point(aes(shape=Chemical))
```



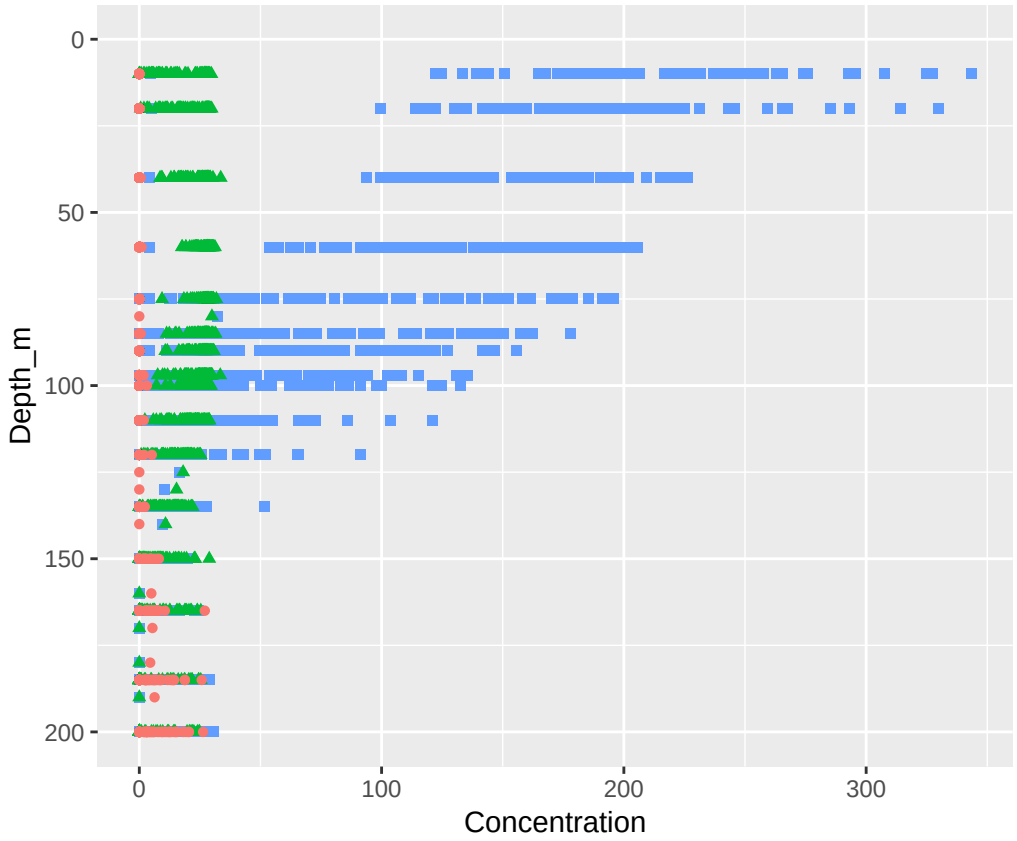
We could further customize this plot by reversing the y-axis (since we traditionally think of depth going down) and adding color on top of shape!

```
dat %>%
  select(Depth_m, O2_uM, NO3_uM, H2S_uM) %>%
  gather(key="Chemical", value="Concentration", O2_uM, NO3_uM, H2S_uM) %>%

  ggplot(aes(x=Concentration, y=Depth_m)) +
  geom_point(aes(shape=Chemical, color=Chemical)) +
  scale_y_reverse(limits=c(200, 0))
```



Facets

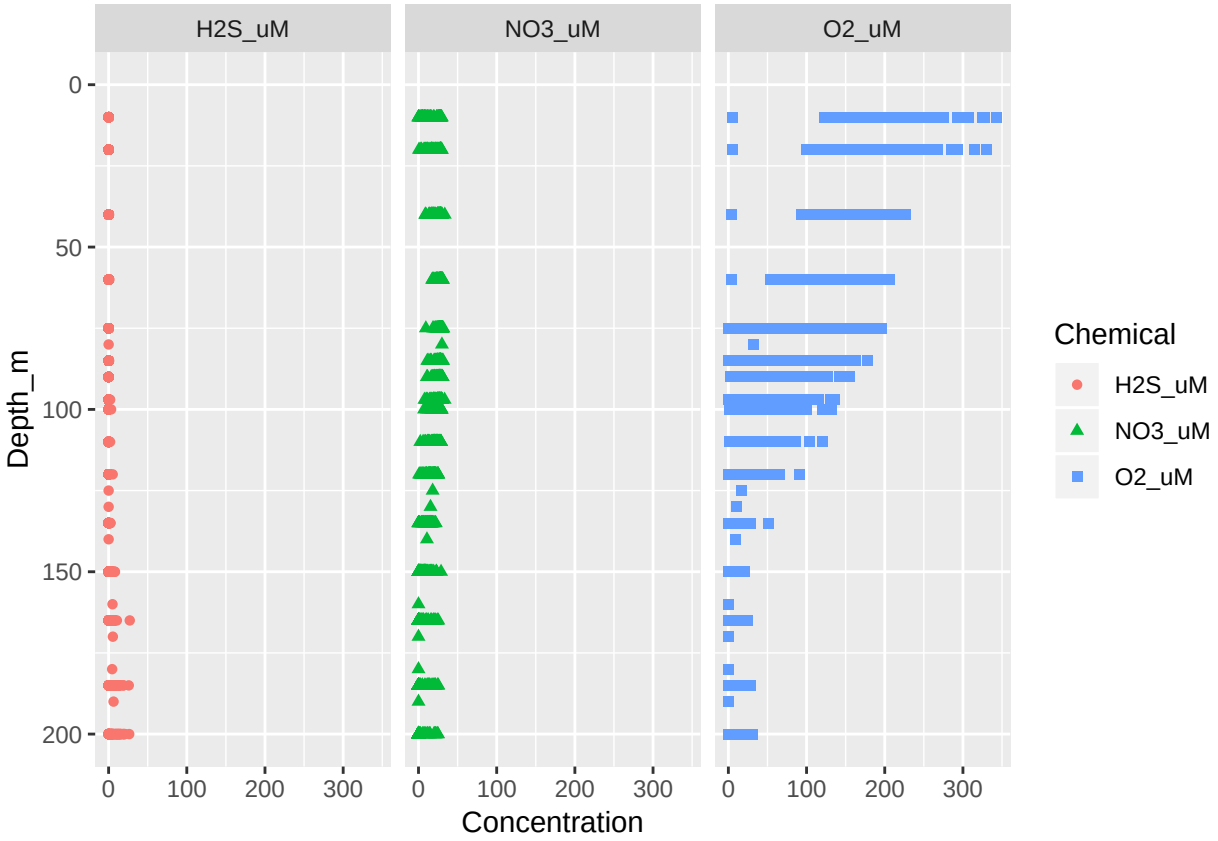
Rather than putting a lot of information into a single graphic, we can split the graphic into panels by certain features. This is called faceting.



Let's return to our most recent plot.

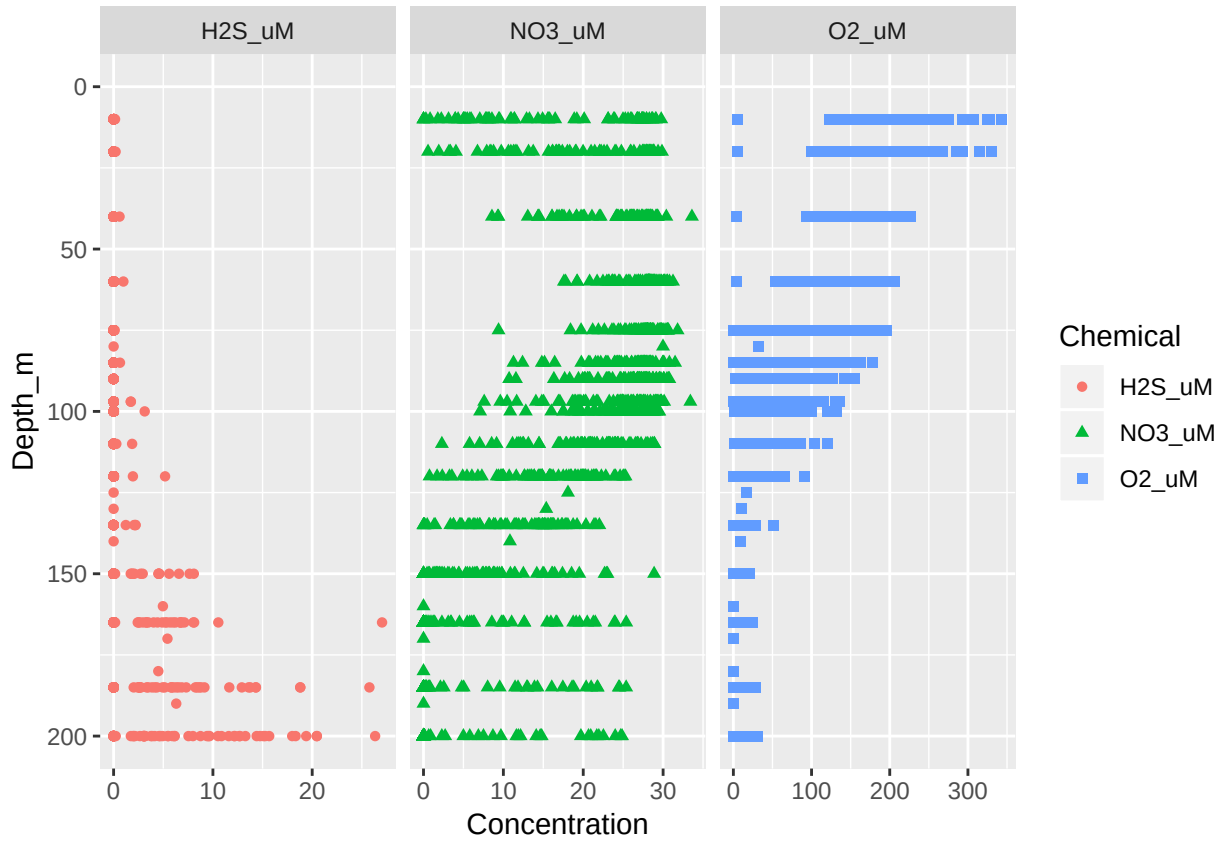
Many of our data points overlap so we may want to separate the geochemical variables into separate plots. Instead of independently creating 3 plots, we can instead add `facet_wrap` of the Chemical variable to the exact same code from above.

```
dat %>%
  select(Depth_m, O2_uM, NO3_uM, H2S_uM) %>%
  gather(key="Chemical", value="Concentration", O2_uM, NO3_uM, H2S_uM) %>%

  ggplot(aes(x=Concentration, y=Depth_m)) +
  geom_point(aes(shape=Chemical, color=Chemical)) +
  scale_y_reverse(limits=c(200, 0)) +
  facet_wrap(~Chemical) ## New facets
```



Since the 3 variables are on somewhat different scales, we can also tell ggplot to scale the x-axes independently within each facet with the `scales` option.

```
dat %>%
  select(Depth_m, O2_uM, NO3_uM, H2S_uM) %>%
  gather(key="Chemical", value="Concentration", O2_uM, NO3_uM, H2S_uM) %>%

  ggplot(aes(x=Concentration, y=Depth_m)) +
  geom_point(aes(shape=Chemical, color=Chemical)) +
  scale_y_reverse(limits=c(200, 0)) +
  facet_wrap(~Chemical, scales="free_x")
```



You can also facet across multiple variables using `facet_grid`.

Further customization

ggplot allows you to complete many, many customizations to plots that we do not have time to go over. Some aspects you may want to explore for the next problem set include:

- **Other geoms:** There are many more geoms outside of a scatterplot. Check them out on the R cheatsheets also available in RStudio under Help > Cheatsheets
- **Themes:** By default, ggplot uses a gray color scheme. However, there are a number of other themes that come with ggplot as well as many more in the additional package ggthemes. You can see many examples from ggplot or ggthemes.
- **Labels:** All aspects of labs can be changed in your plot, including axes, legends, facets, etc. This can be done with `labs()` or in a custom theme with `theme()`
- **Multi-panel figures:** While faceting achieves this for data in a single data frame, you may wish to combine figure from different data sets or different subsets of the same data set. This can be done with a number of packages. We recommend `cowplot` as it easily combines and labels multi-panel figures.
- **Saving plots:** You can simply save you plot in RStudio with the 'Export' button in the 'Plots' tab. However, for a fully reproducible script, checkout the `ggsave()` function!

And sometimes after all this, it's still not perfect. So, don't feel bad if you need to finalize plots for publication in another editor like Illustrator. Sometimes it's a necessary evil.

Additional resources

- R cheatsheets also available in RStudio under Help > Cheatsheets

- ggplot tutorial 1
- ggplot tutorial 2
- ggplot tutorial 3 ***