10: Data Visualization

Environmental Data Analytics | Kateri Salk

Spring 2019

LESSON OBJECTIVES

1. Perform advanced edits on ggplot objects to follow best practices for data visualization

SET UP YOUR DATA ANALYSIS SESSION

```
getwd()
```

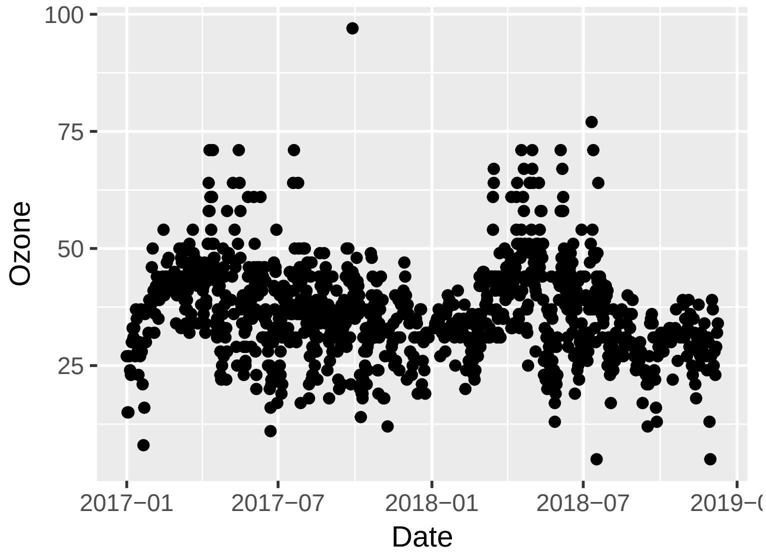
```
## [1] "/Users/ks501/Documents/GithubRepos/ENV872"
```

```
library(tidyverse)

PeterPaul.chem.nutrients <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Proc
PeterPaul.nutrients.gathered <- read.csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Pr
EPAair <- read.csv("./Data/Processed/EPAair_O3PM25_3sites1718_processed.csv")

EPAair$Date <- as.Date(EPAair$Date, format = "%Y-%m-%d")
PeterPaul.chem.nutrients$sampledate <- as.Date(PeterPaul.chem.nutrients$sampledate, format = "%Y-%m-%d")
```

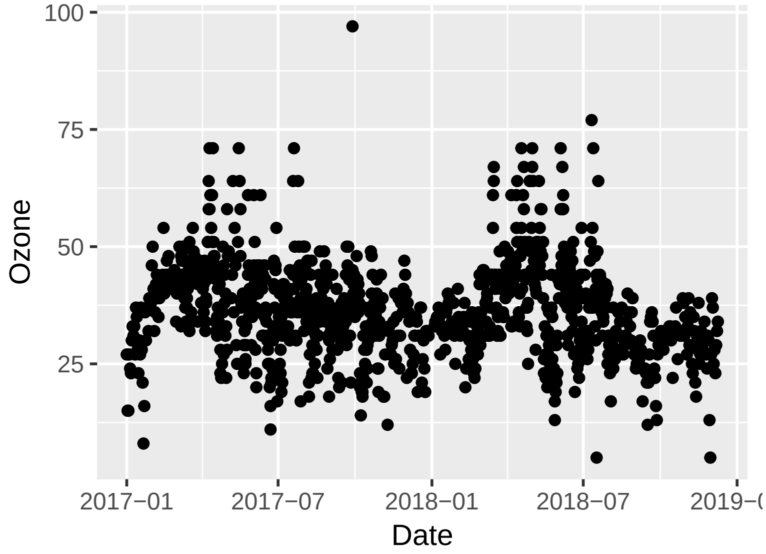
Themes

Often, we will want to change multiple visual aspects of a plot. Ggplot comes with pre-built themes that will adjust components of plots if you call that theme.

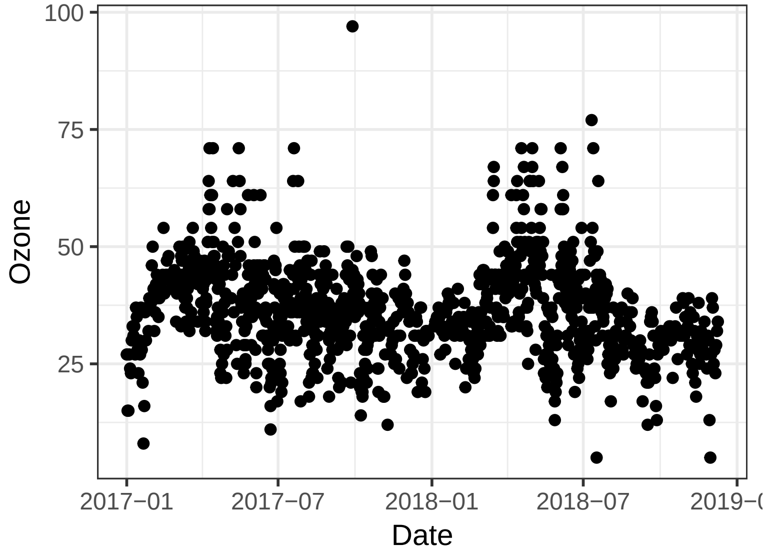
```
O3plot <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot)
```



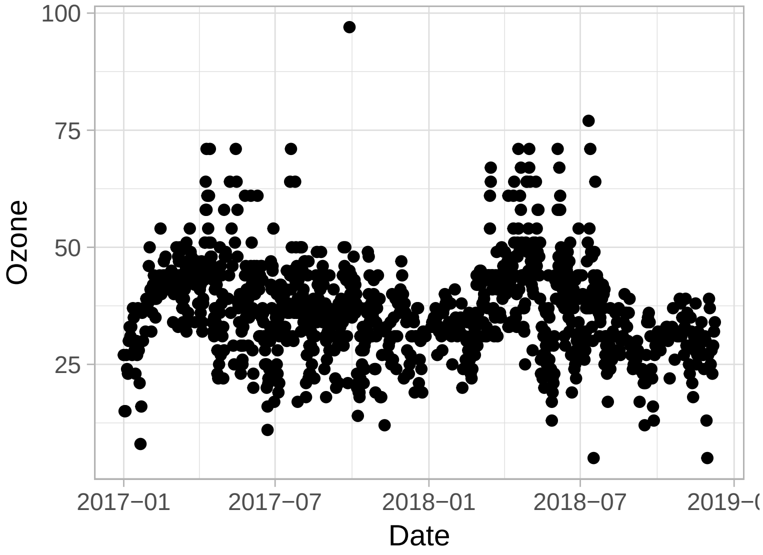
```
O3plot1 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone)) +
  theme_gray()
print(O3plot1)
```



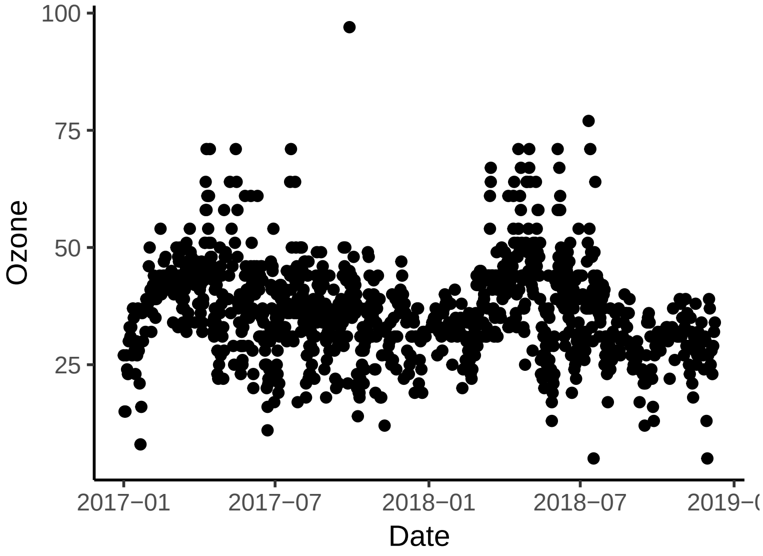
```
O3plot2 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone)) +
  theme_bw()
print(O3plot2)
```



```
O3plot3 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone)) +
  theme_light()
print(O3plot3)
```



```r
O3plot4 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone)) +
  theme_classic()
print(O3plot4)
```



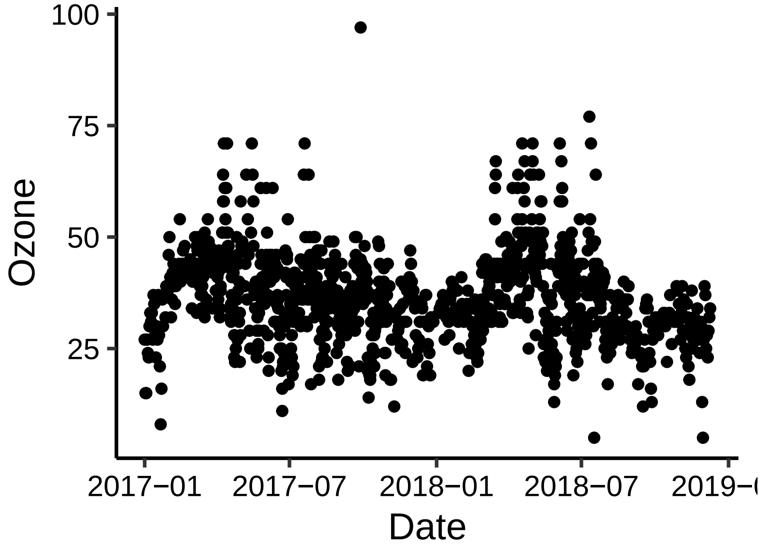
Notice that some aspects of your graph have not been adjusted, including:

- text size
- axis label colors
- legend position and justification

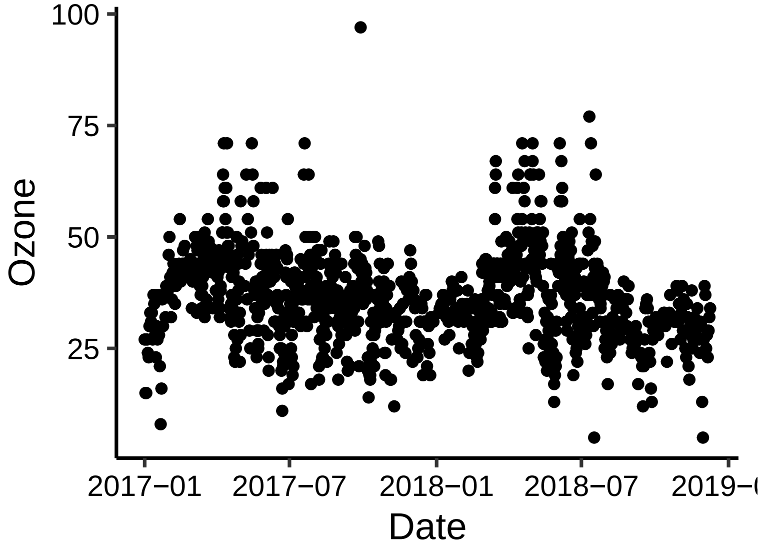
If you would like to set a common theme across all plots in your analysis session, you may define a theme and call up that theme for each graph. This eliminates the need to add multiple lines of code in each plot.

```r
mytheme <- theme_classic(base_size = 14) +
  theme(axis.text = element_text(color = "black"),
        legend.position = "top") #alternative: legend.position + legend.justification
```

```r
# options: call the theme in each plot or set the theme at the start.

O3plot5 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone)) +
  mytheme
print(O3plot5)
```



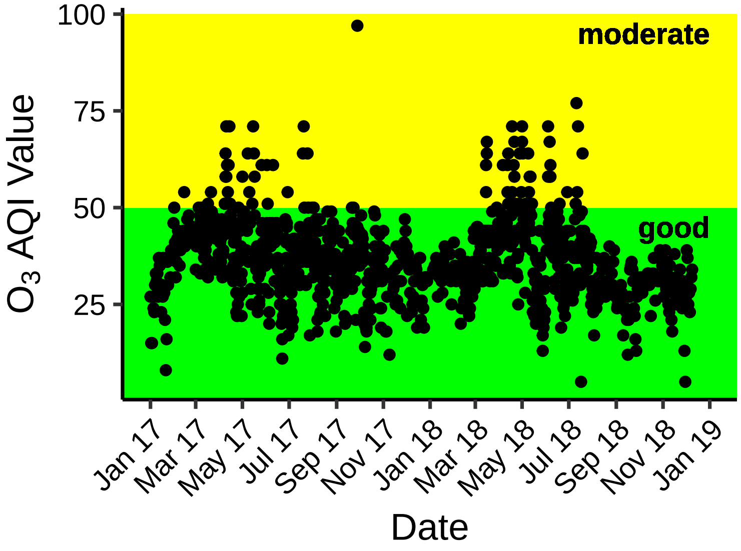
```r
theme_set(mytheme)

O3plot6 <- ggplot(EPAair) +
  geom_point(aes(x = Date, y = Ozone))
print(O3plot6)
```



Adjusting multiple components of your plots

While the theme allows us to set multiple aspects of plots, ggplot allows us to adjust other parts of plots outside of the theme.

```
O3plot7 <- ggplot(EPAair, aes(x = Date, y = Ozone)) +
  geom_rect(xmin = as.Date("2016-01-01"), xmax = as.Date("2020-01-01"),
            ymin = 0, ymax = 50, fill = "green") +
  geom_rect(xmin = as.Date("2016-01-01"), xmax = as.Date("2020-01-01"),
            ymin = 50, ymax = 100, fill = "yellow") +
  geom_point() +
  geom_text(x = as.Date("2019-01-01"), y = 45, label = "good", hjust = 1, fontface = "bold") +
  geom_text(x = as.Date("2019-01-01"), y = 95, label = "moderate", hjust = 1, fontface = "bold") +
  scale_x_date(limits = as.Date(c("2017-01-01", "2018-12-31")),
    date_breaks = "2 months", date_labels = "%b %y") +
  ylab(expression("O"[3]* " AQI Value")) +
  theme(axis.text.x = element_text(angle = 45,  hjust = 1))
print(O3plot7)
```



Color palettes

There are several color palettes that are designed to be more effective than palettes in base R. These include Viridis (https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html) and Color Brewer (http://colorbrewer2.org/). A few rules for choosing colors:

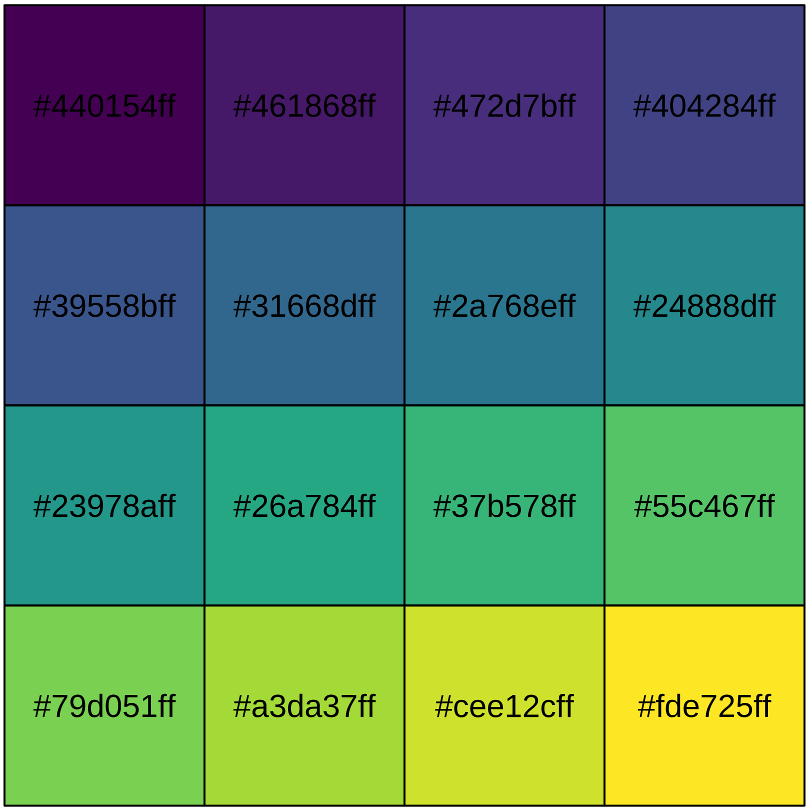
- Consider if your plot needs to be viewed in black and white. If so, choose a sequential palette with varying color intensity.
- Choose a palette that is color-blind friendly
- Maximize contrast (e.g., no pale colors on a white background)
- Diverging color palettes should be used for diverging values (e.g., warm-to-cool works well for values on a scale encompassing negative and positive values)

Perception is key! Choose palettes that are visually pleasing and will communicate what you are hoping your audience to perceive. Hint: base R palettes are not ideal.

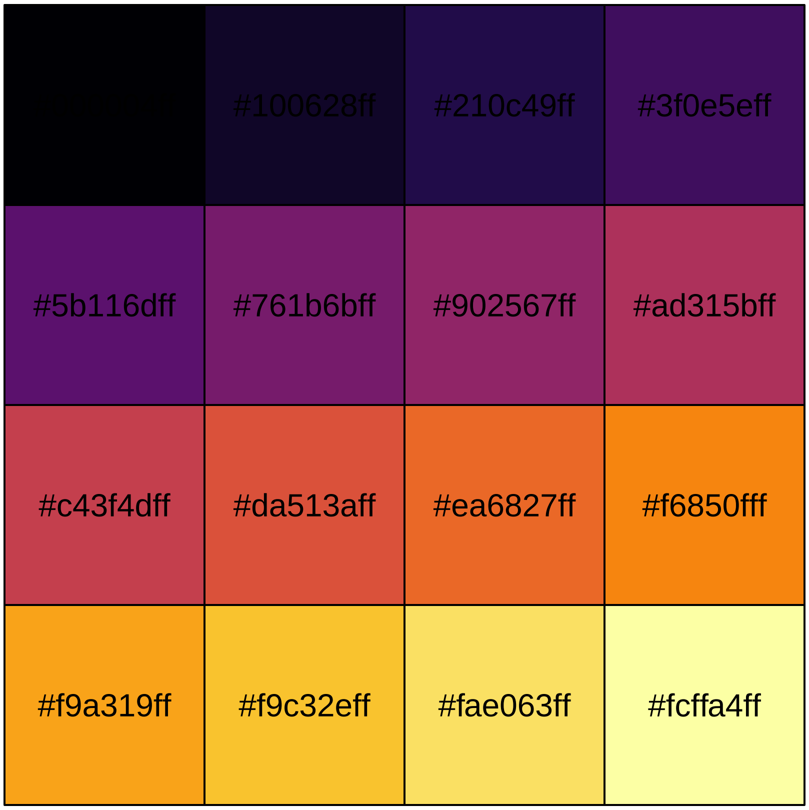
```r
#install.packages("viridis")
#install.packages("RColorBrewer")
#install.packages("colormap")
library(viridis)
```

```
## Loading required package: viridisLite
```

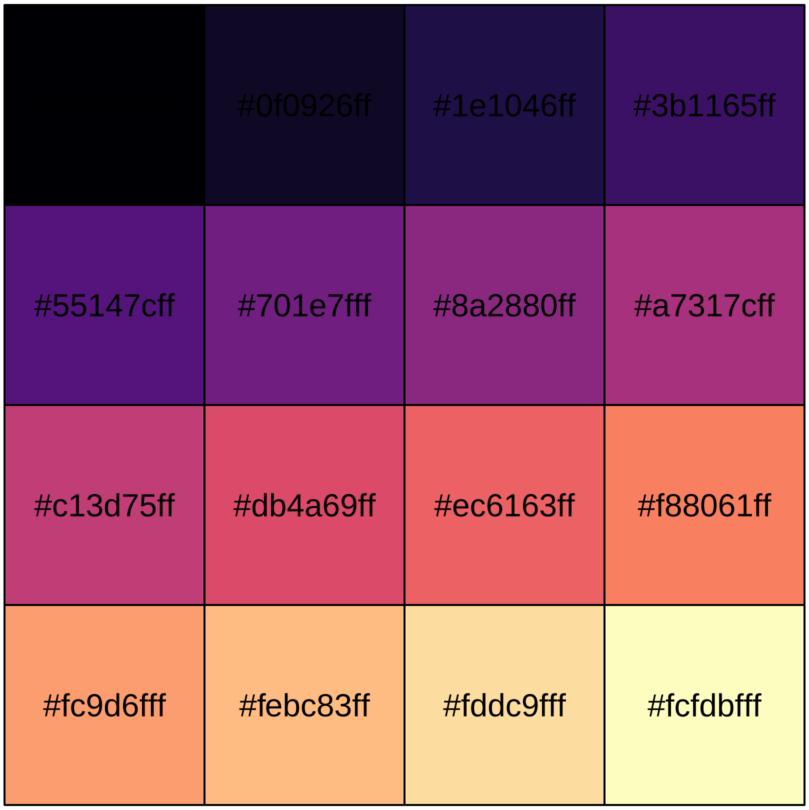
```r
library(RColorBrewer)
library(colormap)

scales::show_col(colormap(colormap = colormaps$viridis, nshades = 16))
```



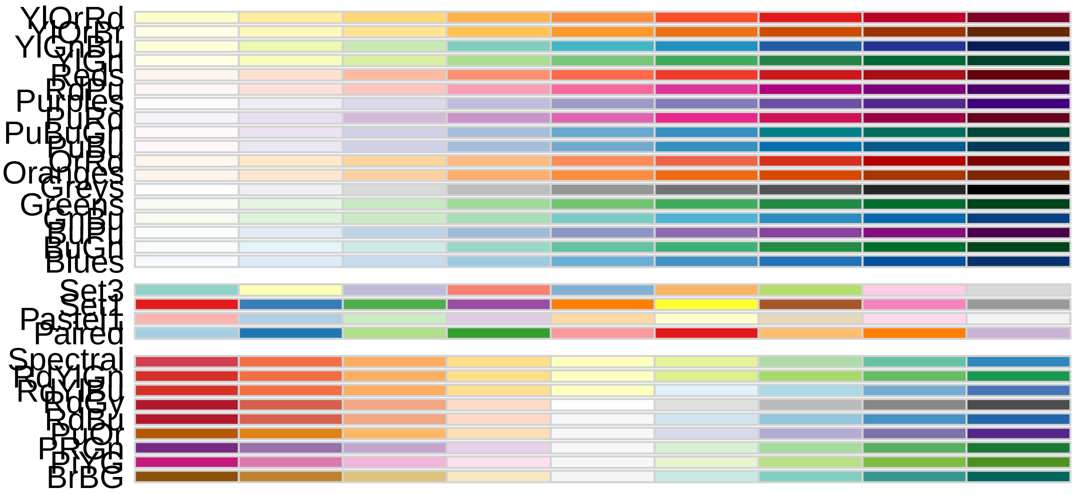
```r
scales::show_col(colormap(colormap = colormaps$inferno, nshades = 16))
```



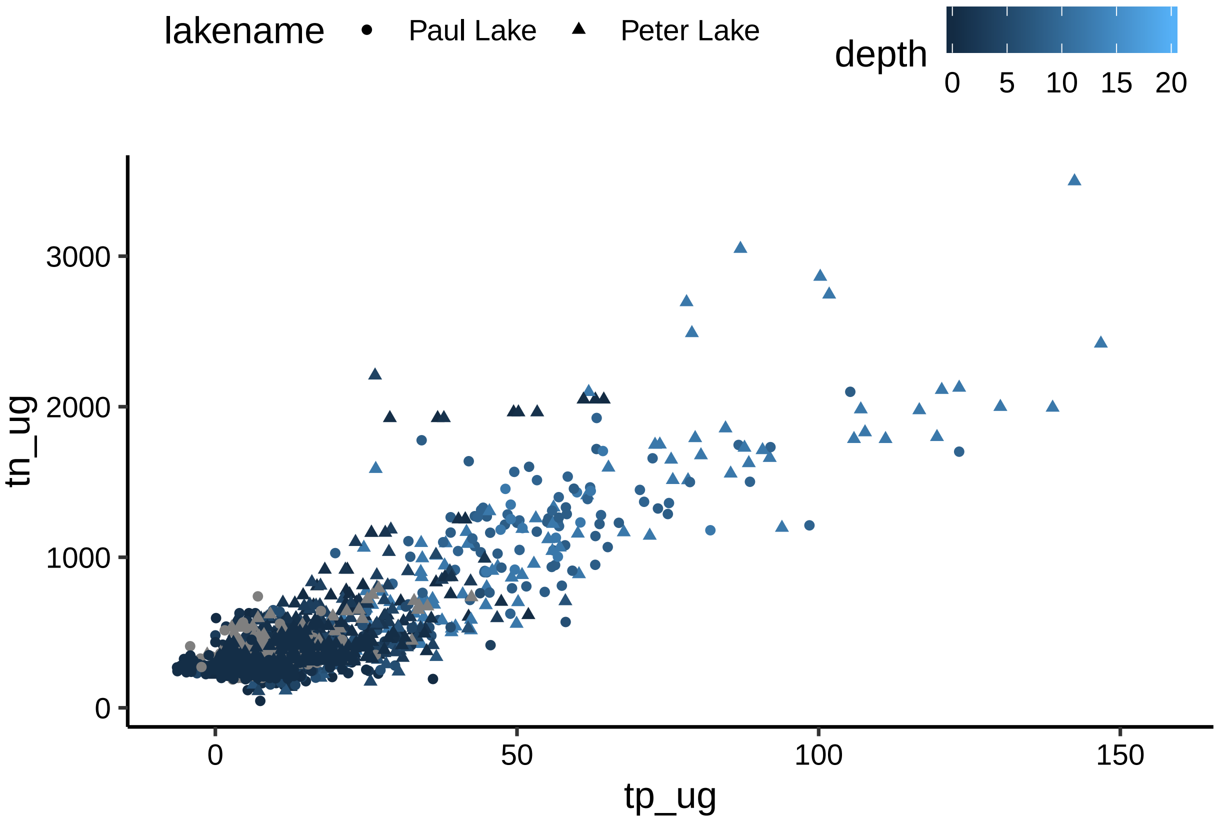
```
scales::show_col(colormap(colormap = colormaps$magma, nshades = 16))
```



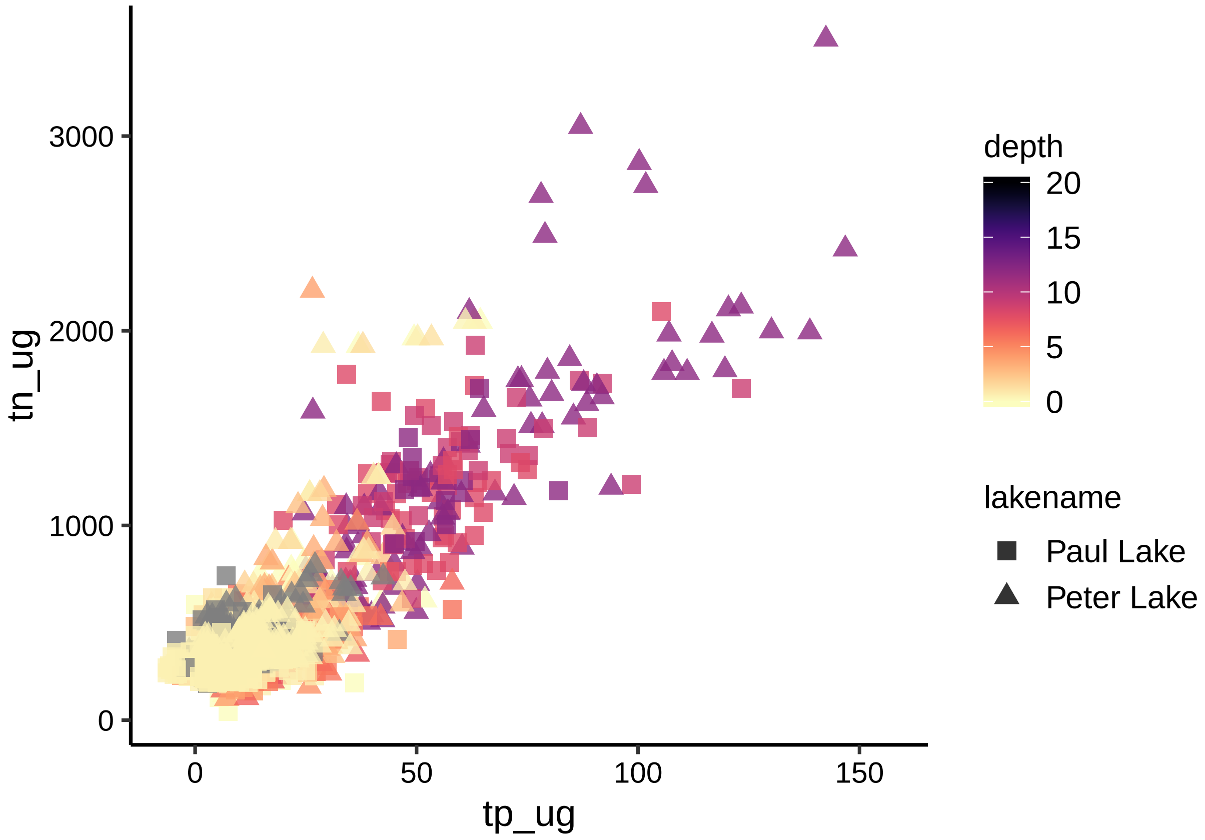
```
display.brewer.all(n = 9)
```



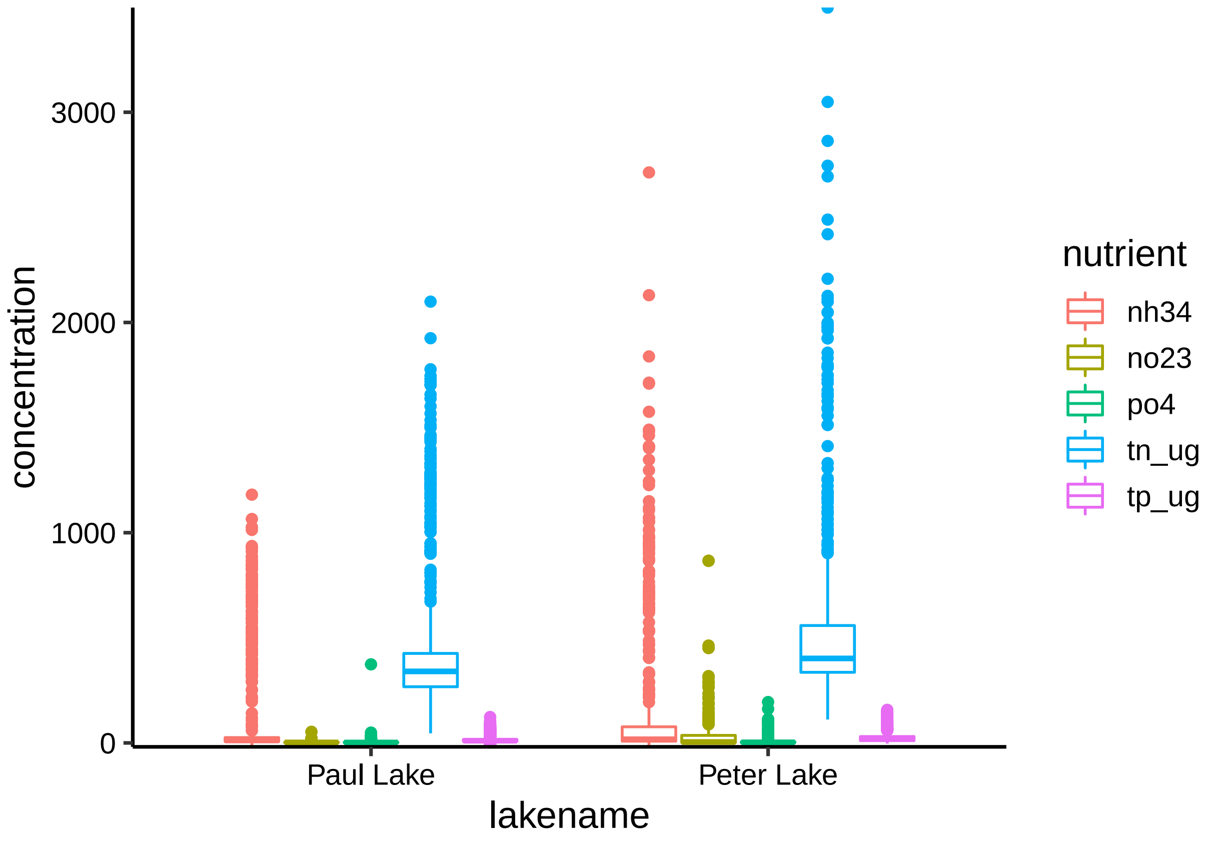
```r
NvsP <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth, shape = lakename)) +
  geom_point()
print(NvsP)
```



```
# let's first make the plot look better.
# change your axis labels to reflect TN and TP in micrograms per liter.
# change your legend labels
NvsP2 <-
  ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = tn_ug, color = depth, shape = lakename)) +
  geom_point(alpha = 0.8, size = 3) +
  #xlab() +
  #ylab() +
  # change your legend labels here
  scale_shape_manual(values = c(15, 17)) +
  #scale_color_distiller(palette = "Blues", direction = 1) + # use scale_color_brewer for discrete vari
  scale_color_viridis(option = "magma", direction = -1) +
  theme(legend.position = "right",
        legend.text = element_text(size = 12), legend.title = element_text(size = 12))
print(NvsP2)
```



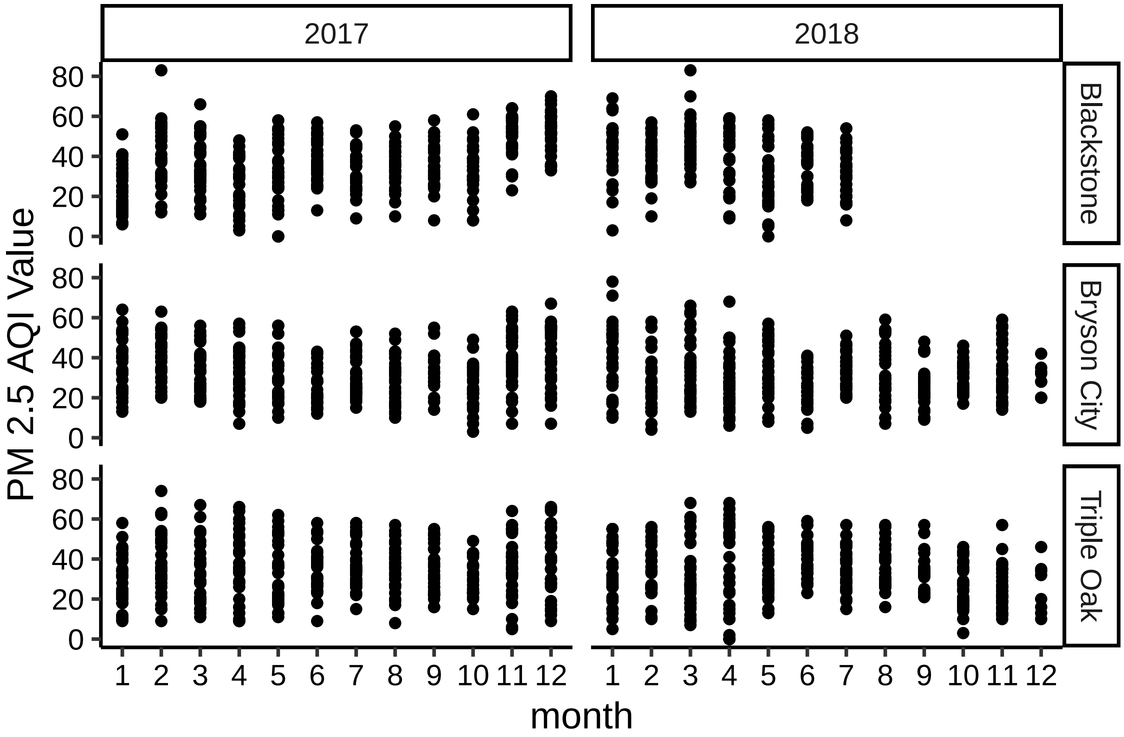
```
# change your y axis label to list concentration in micrograms per liter
# remove your x axis label
# change labels for nutrients in the legend
Nutrientplot <-
  ggplot(PeterPaul.nutrients.gathered, aes(x = lakename, y = concentration, color = nutrient)) +
  geom_boxplot() +
# place your additional edits here
  scale_y_continuous(expand = c(0, 0)) +
  #scale_color_brewer(palette = "YlGnBu") +
  #scale_color_manual(values = c("#7fcdbb", "#41b6c4", "#1d91c0", "#225ea8", "#0c2c84")) +
  #scale_color_viridis(discrete = TRUE) +
  theme(legend.position = "right")
print(Nutrientplot)
```



Adjusting facets

```
PMplot.faceted <-
  ggplot(EPAair, aes(x = month, y = PM2.5)) +
  geom_point() +
  facet_grid(Site.Name ~ year) +
  scale_x_continuous(breaks = c(1:12)) +
  #theme(strip.background = element_rect(fill = "black"), strip.text = element_text(color = "white"))
  ylab(expression("PM 2.5 AQI Value"))
print(PMplot.faceted)
```

```
## Warning: Removed 52 rows containing missing values (geom_point).
```



Multiple plots on a page

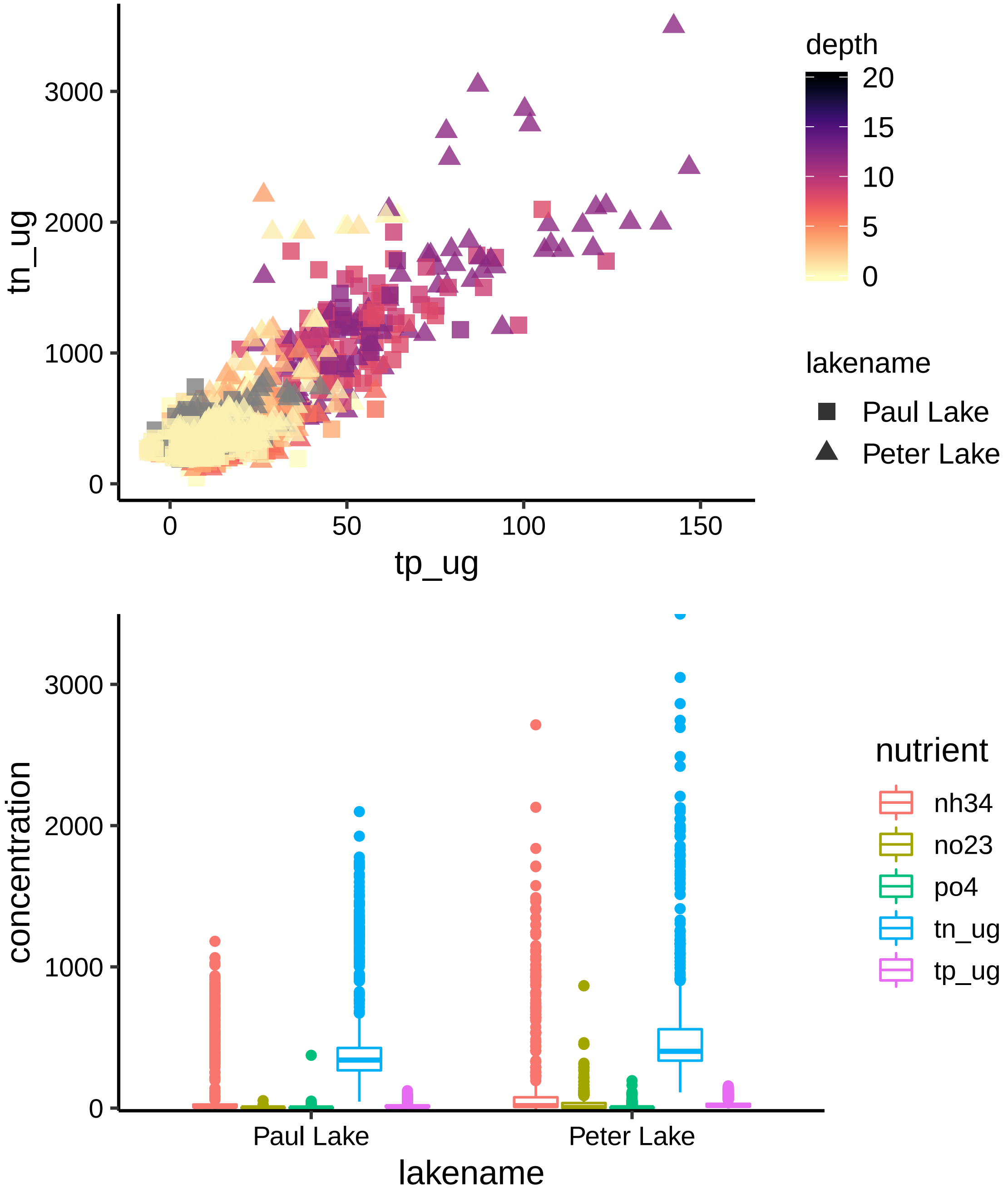
In situations where facets don't fill our needs to place multiple plots on a page, we can use the package `gridExtra` to arrange plots. The `grid.arrange` function is extremely flexible in its ability to arrange plots in specific configurations. A useful guide can be found here: https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html.

```
#install.packages("gridExtra")
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(NvsP2, Nutrientplot)
```

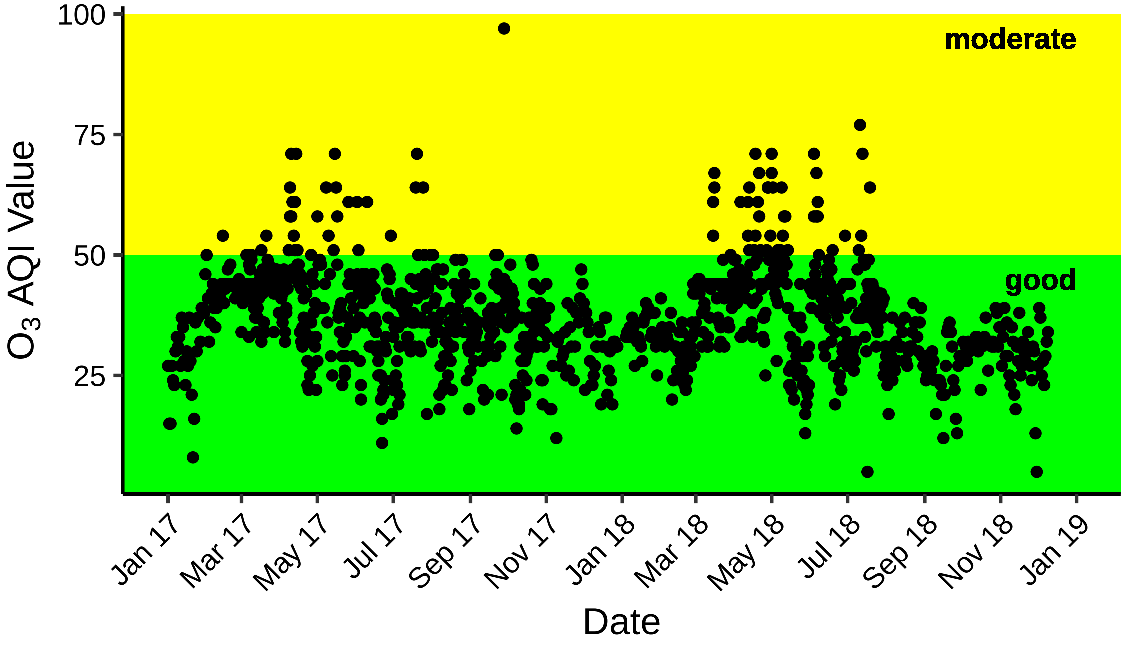
```
## Warning: Removed 21648 rows containing missing values (geom_point).
```

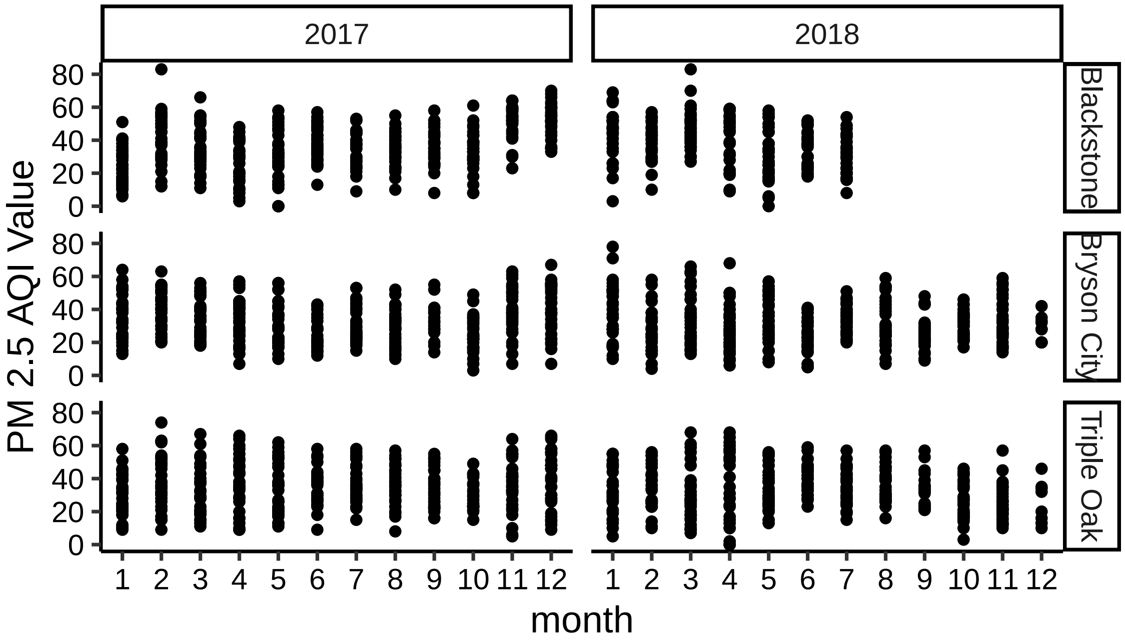


```
grid.arrange(O3plot7, PMplot.faceted)
```

```
## Warning: Removed 868 rows containing missing values (geom_point).
```

```
## Warning: Removed 52 rows containing missing values (geom_point).
```





Saving plots

The `ggsave` function allows you to save plots in jpg, png, eps, pdf, tiff, and other formats. The following information can be supplied:

- filename, with file extension and in quotes (required)
- plot object (required)
- path, with file name
- width, height, units
- resolution (dpi)

For example: `ggsave("PMplot.jpg", PMplot.faceted,  path = "./Output/PMplotjpg", height = 4, width = 6, units = "in", dpi = 300)`