ggplot Tutorial

Nick Bayley
RMarkdown by Alexzandra Morris

2022-10-14

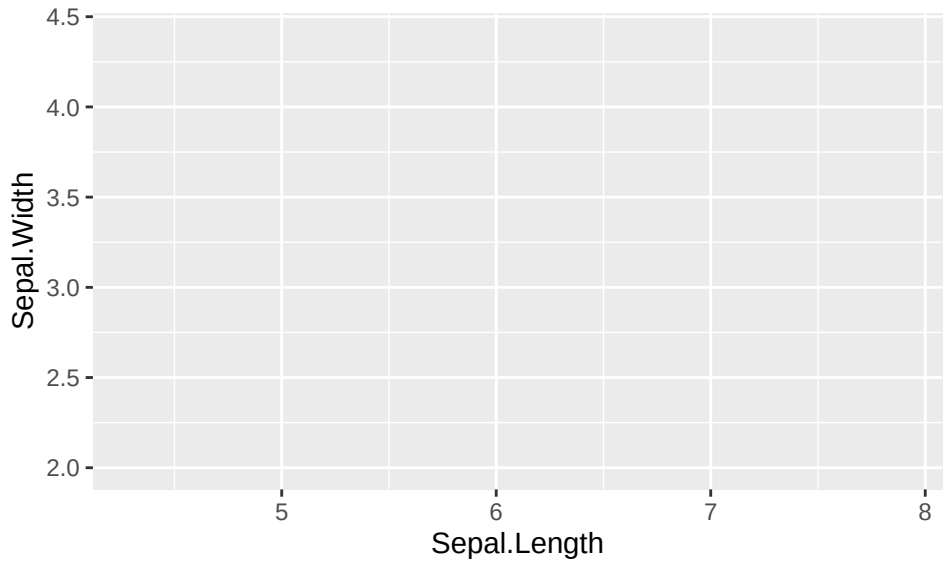
```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

1. The Basics

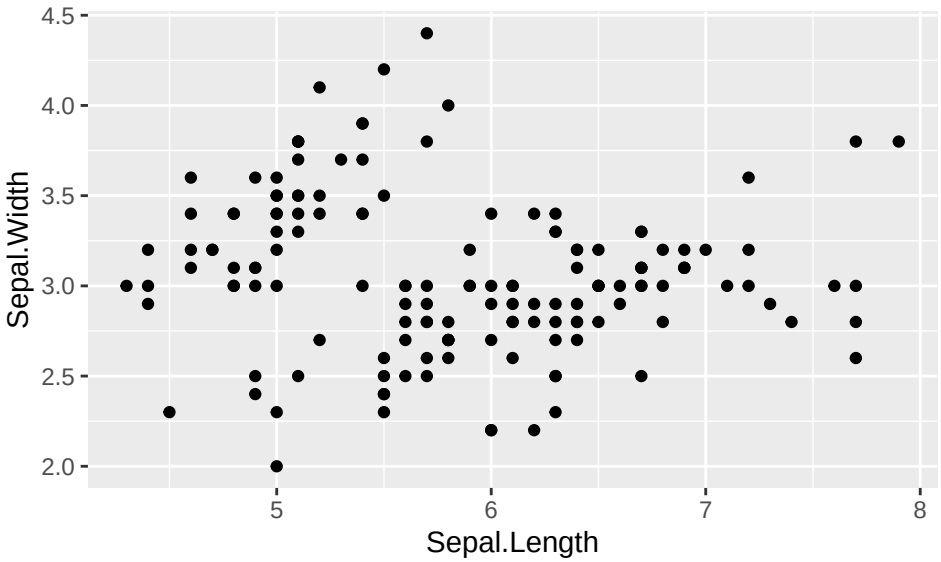
Set up mapping aesthetics by mapping continuous variables (columns in our dataset) to x and y axes.

```
ggplot(data = iris, mapping = aes(x = Sepal.Length, y = Sepal.Width))
```



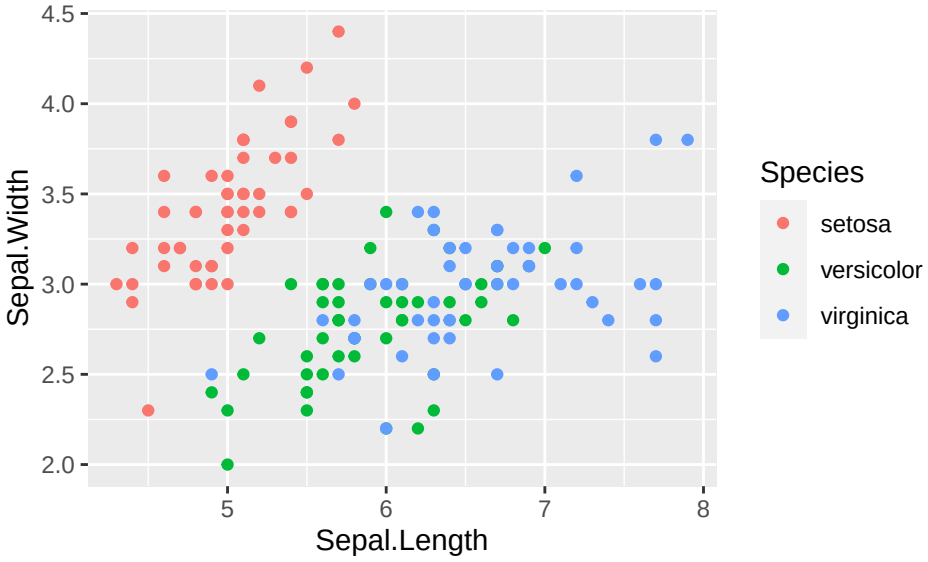
Creating a scatterplot by adding a geom_point layer

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
geom_point()
```



Add color by species of plant to the aesthetics mapping

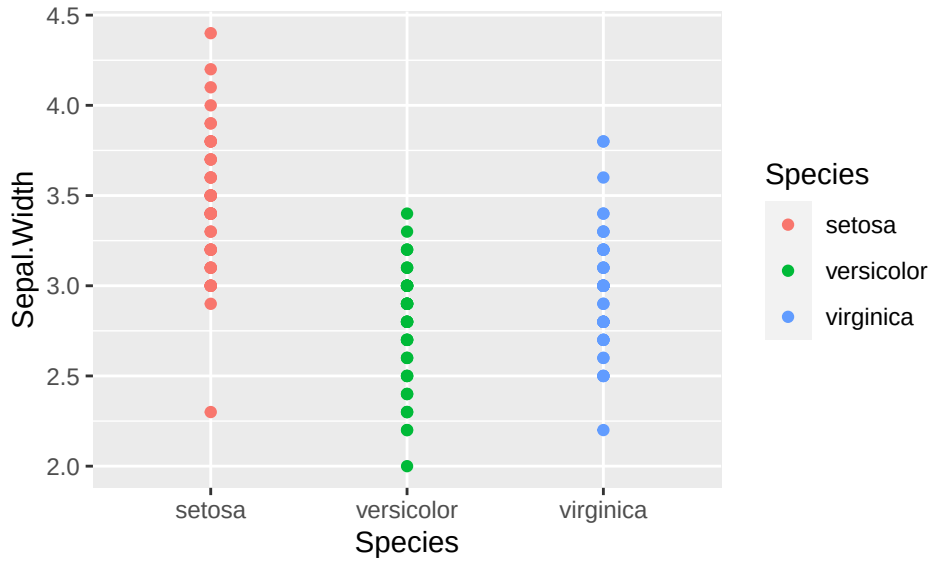
```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
geom_point()
```



Adding multiple layers

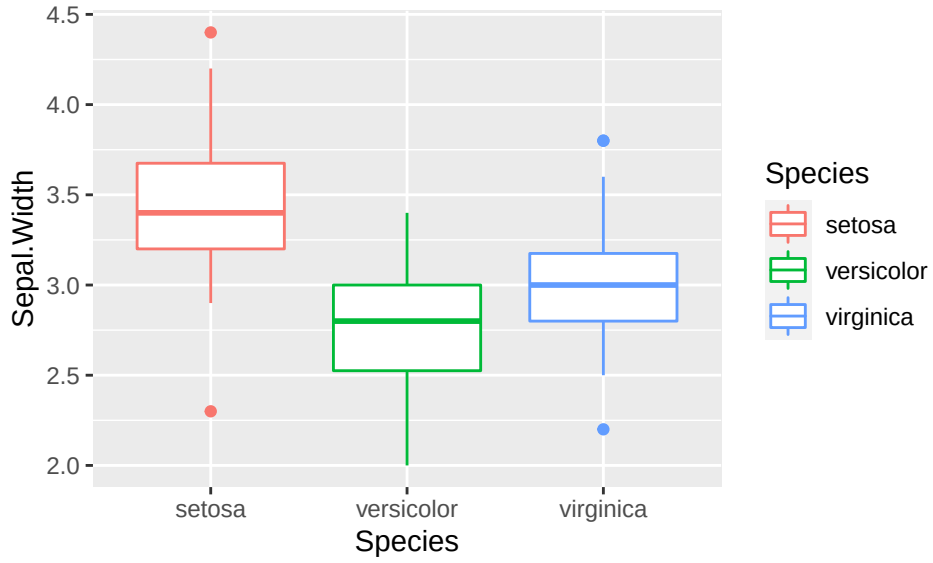
Set up mapping aesthetics with discrete x variable and continuous y variable.

```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_point()
```



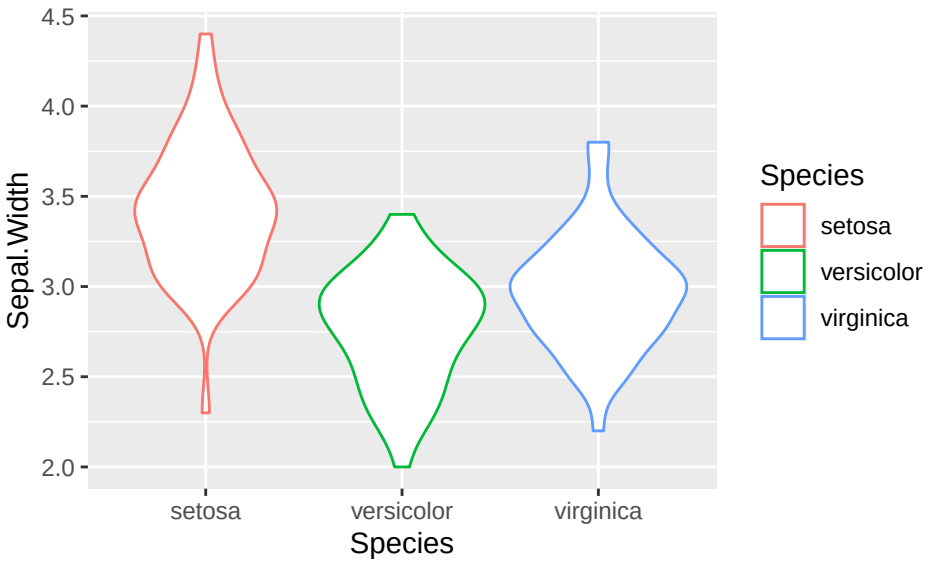
Try a different geometric layer boxplot.

```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_boxplot()
```



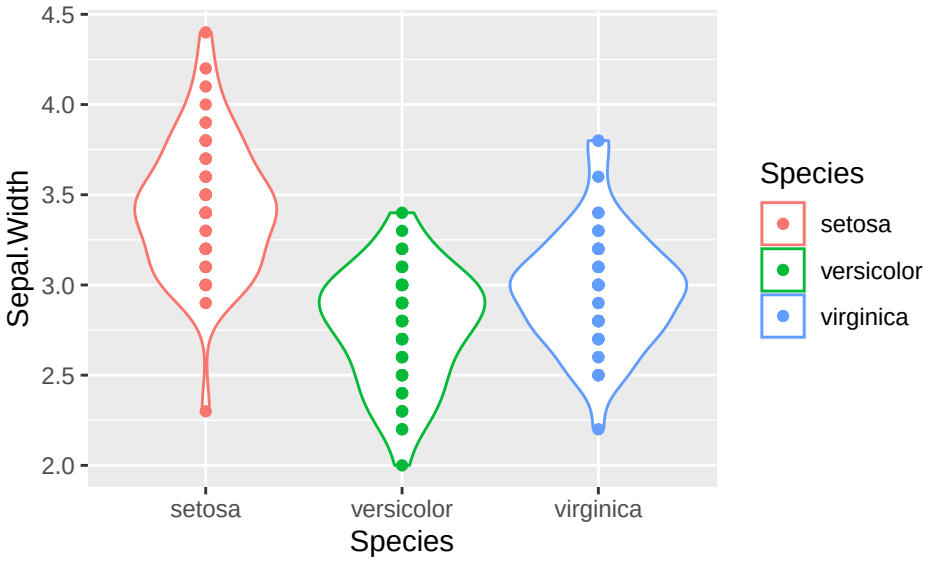
or a violin plot.

```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_violin()
```



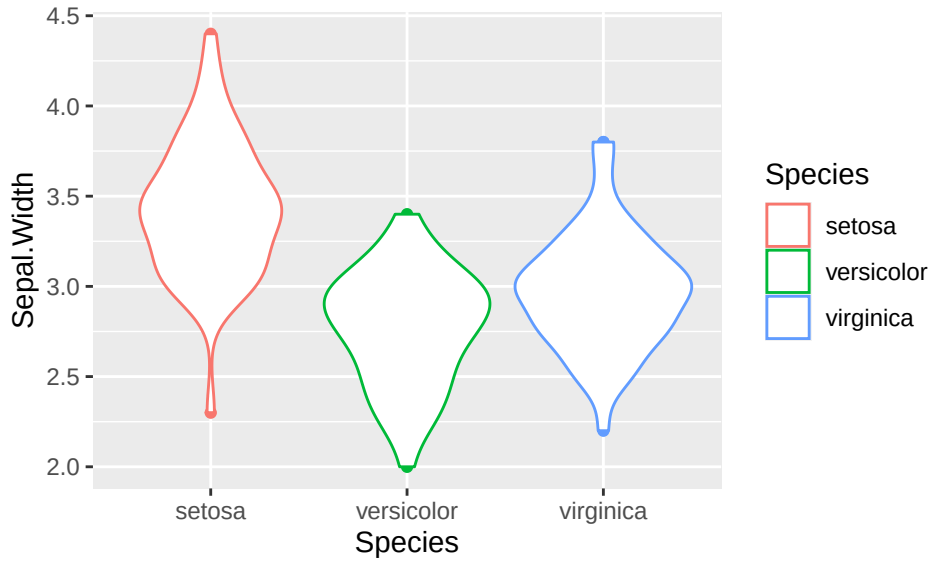
Add second layer for points.

```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_violin() + geom_point()
```



Note that the order of the layers matters! The first ones are plotted first and new layers are added on top (Only the violins are visible)

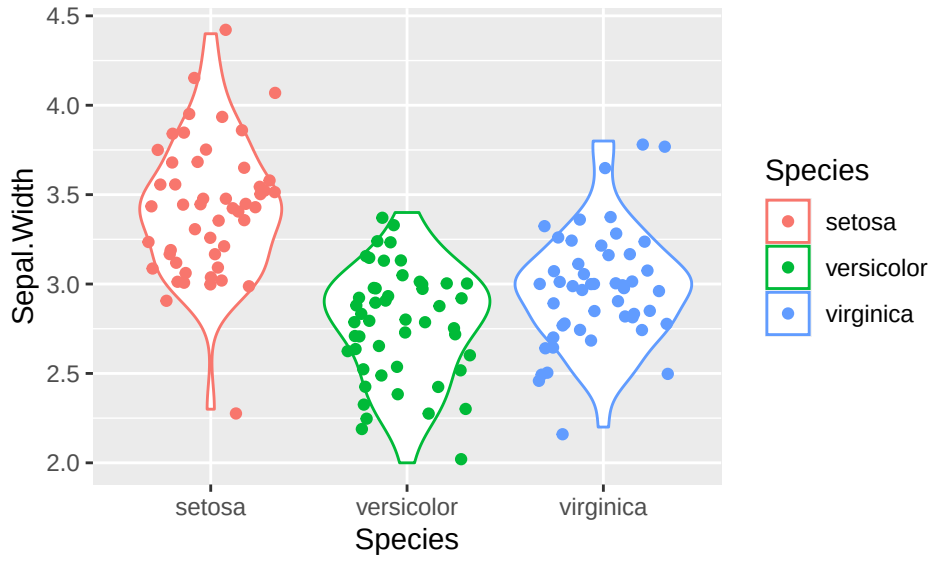
```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_point() + geom_violin()
```



geom_jitter for points with some random noise added.

If the points are overlapping each other, try this.

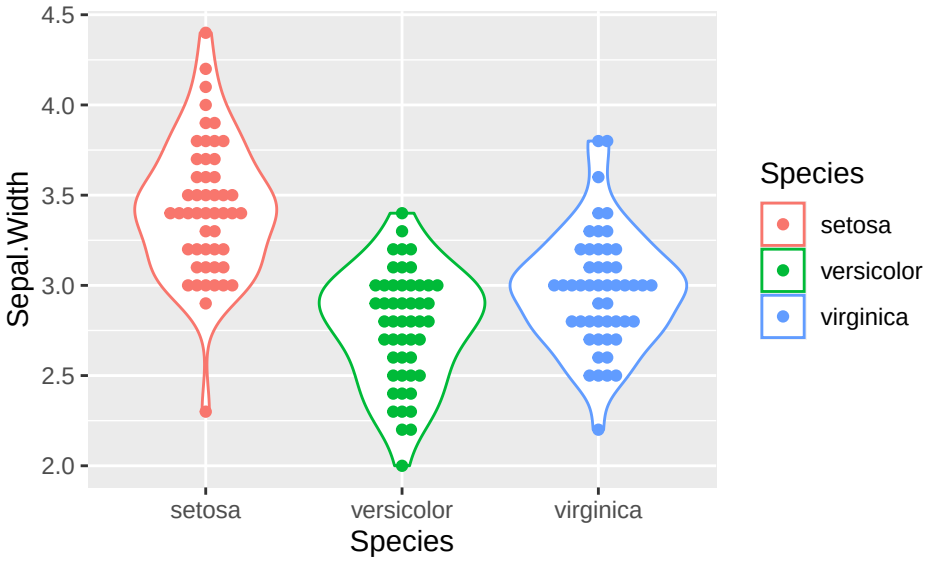
```
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_violin() + geom_jitter(width = 0.33, height = 0.05)
```



Beeswarm Plots: More organized points with random noise

Jittered points help but some points may still be overlapping, so let's use Beeswarm Plots.

```
library(ggbeeswarm)
ggplot(iris, aes(x = Species, y = Sepal.Width, color = Species)) +
geom_violin() + geom_beeswarm(size = 1.5, cex = 1.5)
```

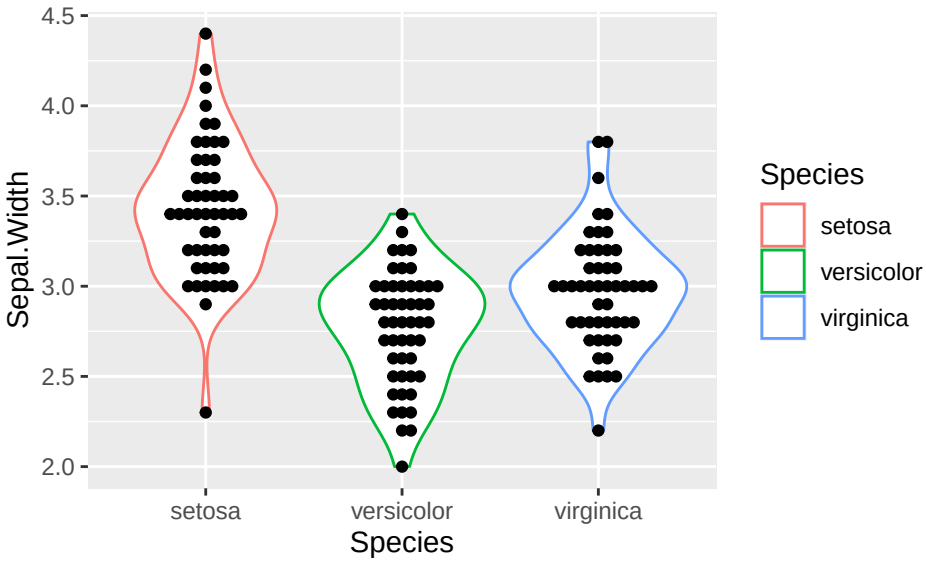


Mapping by layer

Color one specific layer of ggplot

Suppose we only want one layer to be colored, rather than all of them. Here we move mapping of color to a specific layer of the ggplot.

```
ggplot(iris, aes(x = Species, y = Sepal.Width)) +
geom_violin(aes(color = Species)) + geom_beeswarm(size = 1.5, cex = 1.5)
```



Facets

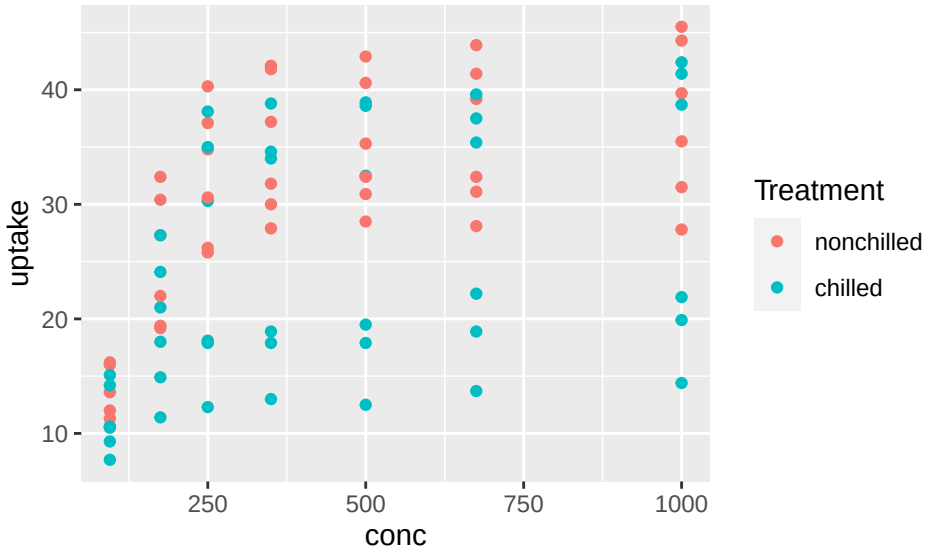
Mapping continuous variables to x and y axes.

Set up mapping aesthetics by mapping continuous variables to x and y axes and coloring by "Treatment".

```
head(CO2)
```

```
Grouped Data: uptake ~ conc | Plant
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
```

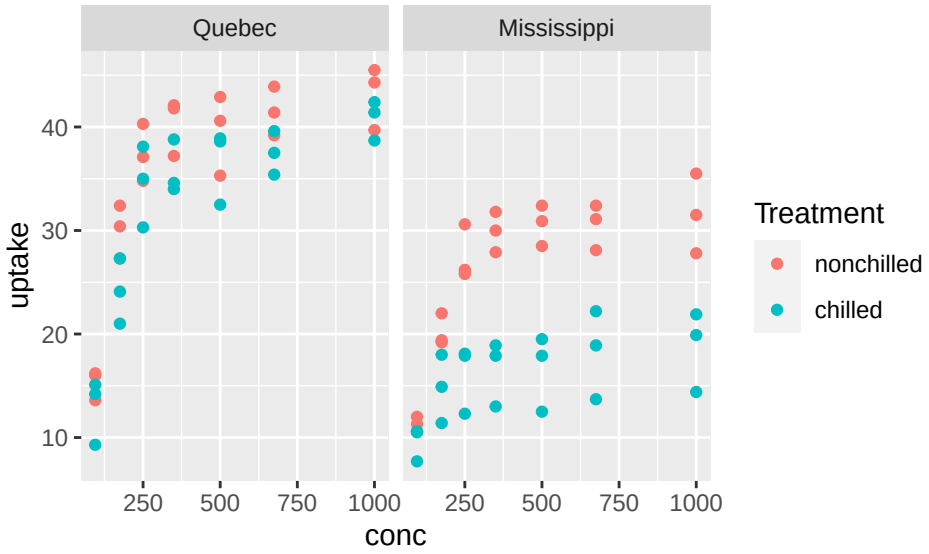
```
gg <- ggplot(CO2, aes(x = conc, y = uptake, color = Treatment)) +
geom_point()
print(gg)
```



Facet Wrap

Our data comes from two different sites. Let's separate the plots by the collection site using facet_wrap.

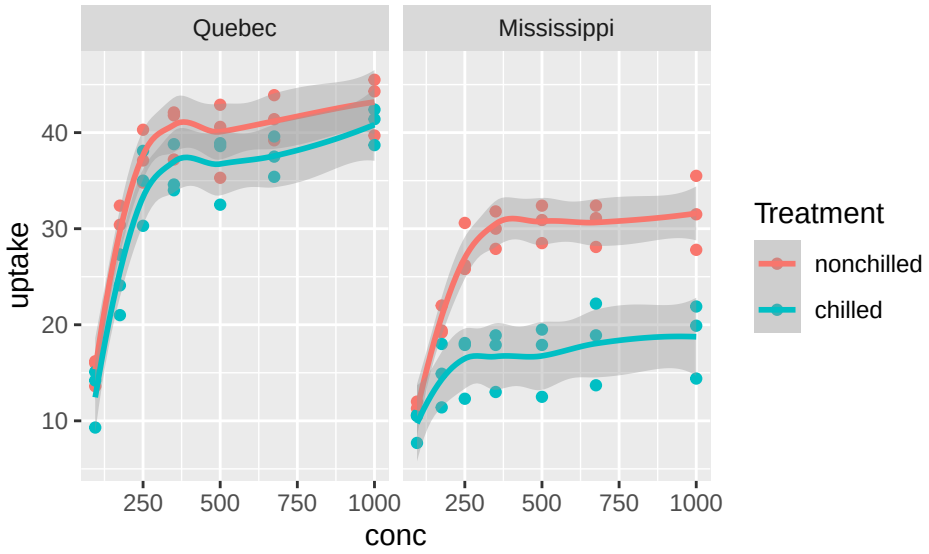
```
gg2 <- gg + facet_wrap(~ Type)
print(gg2)
```



Adding a smooth fitted line to help visualize the trend.

```
gg2 + geom_smooth()
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

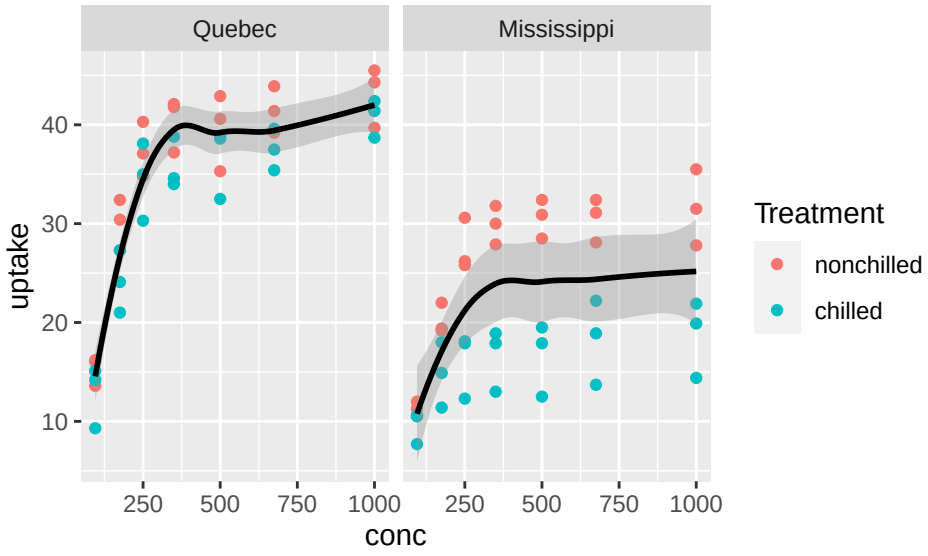


What if we don't want the lines colored?

We can try removing the color mapping in favor of a constant mapping to "black". However, this will cause an issue! The color aesthetic distinguished the two lines we want to draw per facet.

```
gg2 + geom_smooth(color = "black")
```

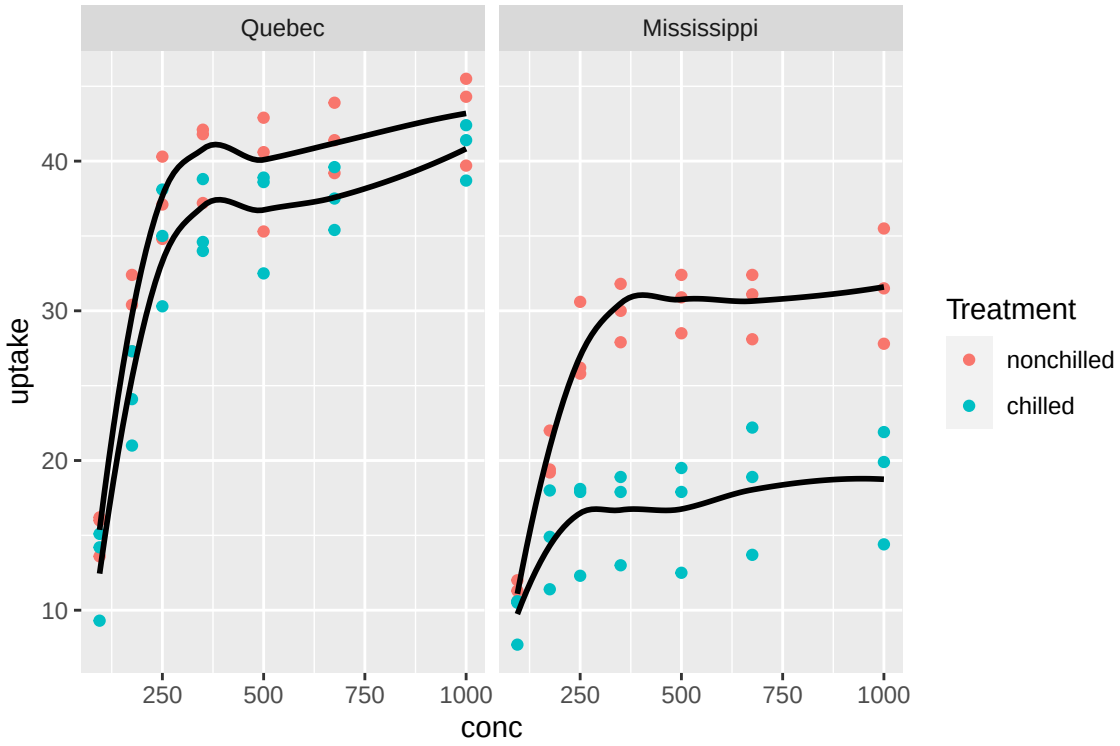
```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Use the group aesthetic mapping when you want to separate mappings without color/fill/etc.

```
gg2 + geom_smooth(aes(group = Treatment), color = "black", se = F)
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

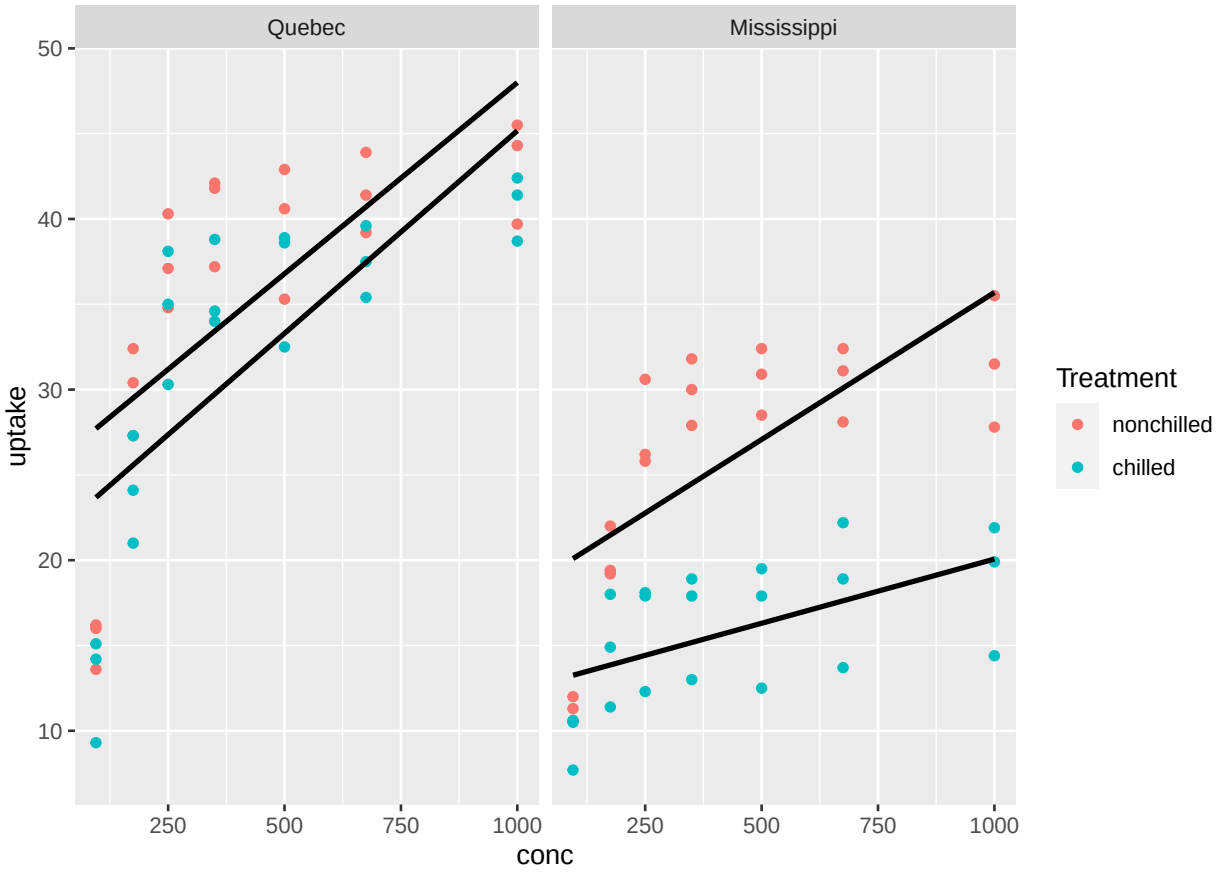


Method Parameter

Although it does not make sense for this data in particular, we can change the method parameter to define what kind of line is fit to the data.

```
gg2 + geom_smooth(aes(group = Treatment), color = "black", se = F, method = "lm")
```

```
`geom_smooth()` using formula 'y ~ x'
```



Barcharts and Stacking

```
head(mtcars)
```

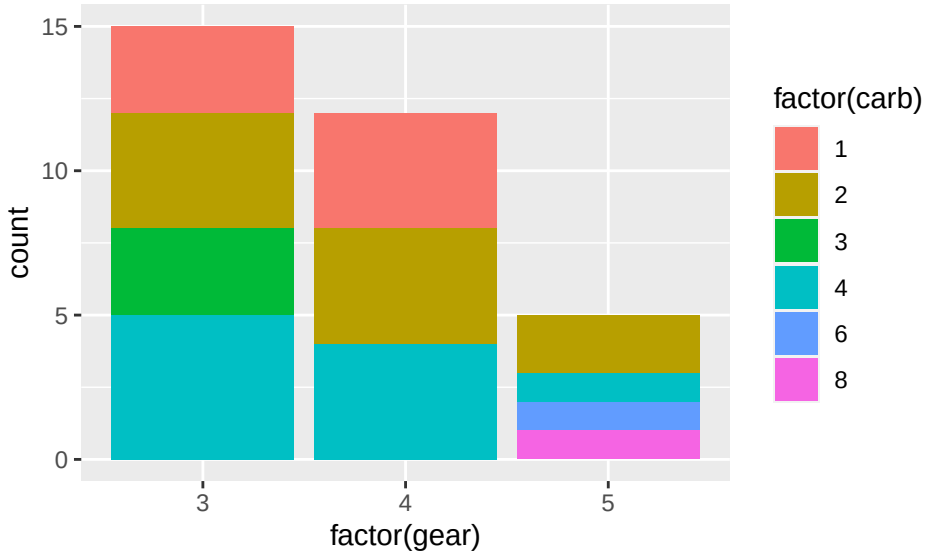
```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Set up mappings

```
gg <- ggplot(mtcars, aes(x = factor(gear), fill = factor(carb)))
```

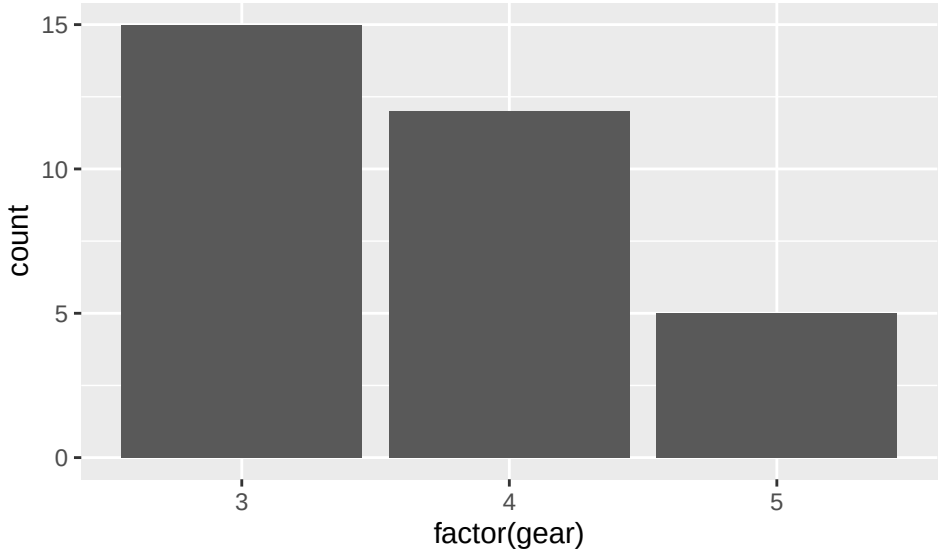
Stacked barchart of counts, colored by the number of carburetors.

```
gg + geom_bar(position = "stack")
```



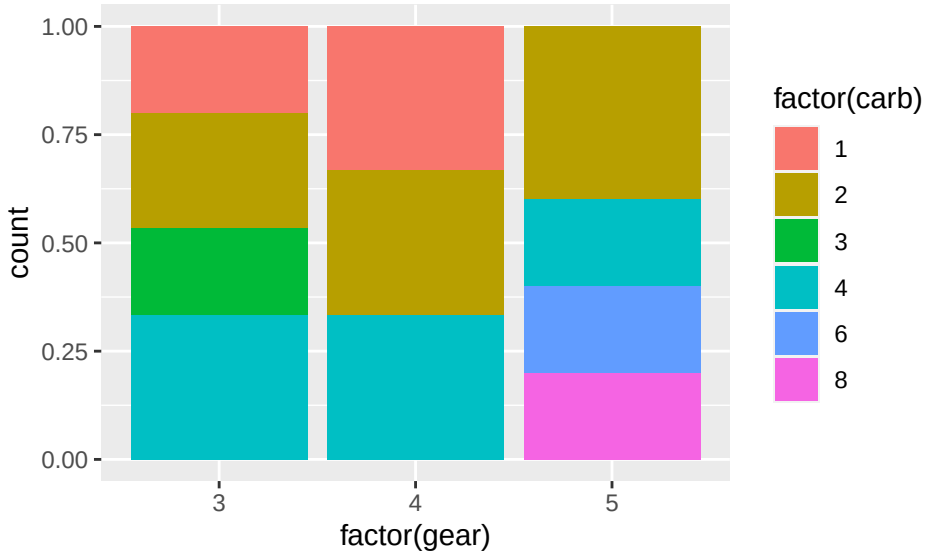
Without the fill parameter, the plot will show the count of each gear.

```
ggplot(mtcars, aes(x = factor(gear))) + geom_bar()
```



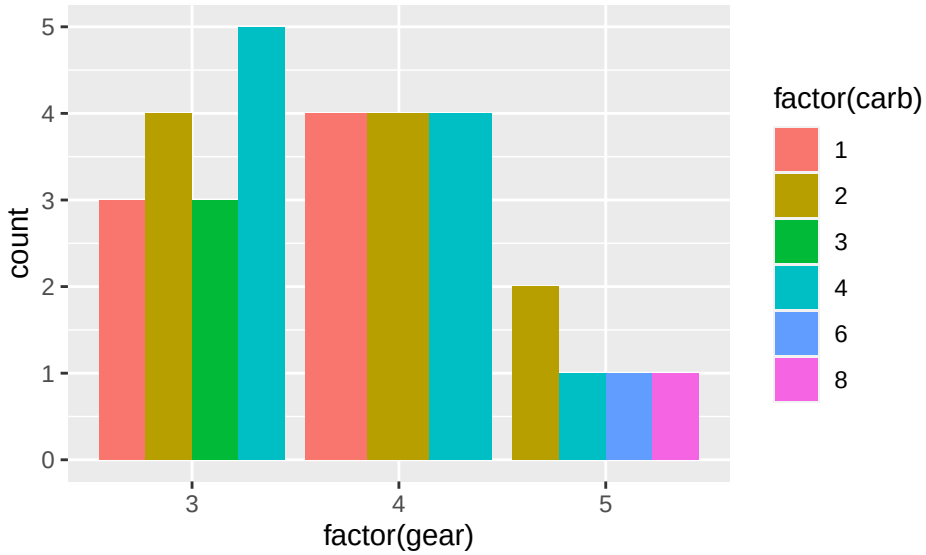
Stacked barchart with relative proportions

```
gg + geom_bar(position = "fill")
```



Dodged barchart with counts

```
gg + geom_bar(position = "dodge")
```



Integrating dplyr

If we want to plot the relative frequencies AND still dodge the fill mapping we will need to use dplyr to calculate the values ourselves instead of leaving it up to R.

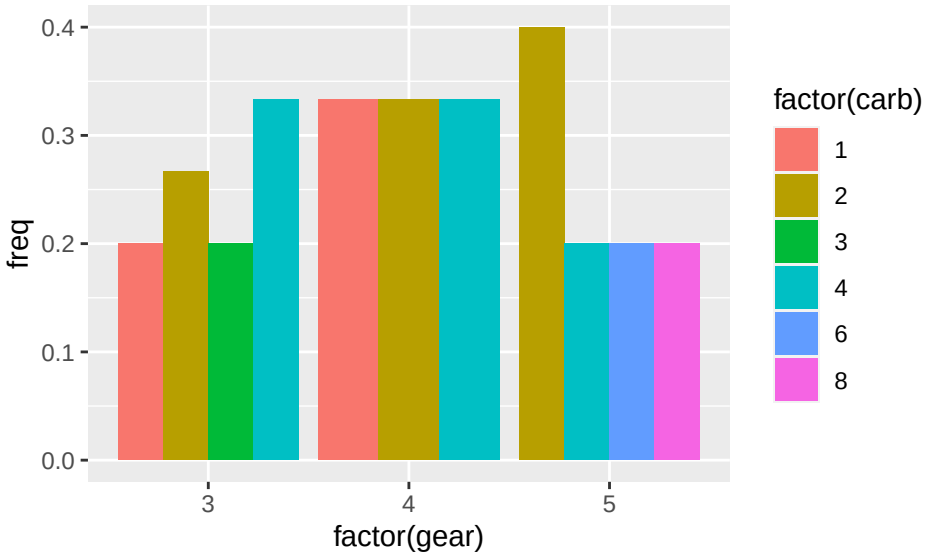
```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Calculate relative frequencies using dplyr piping

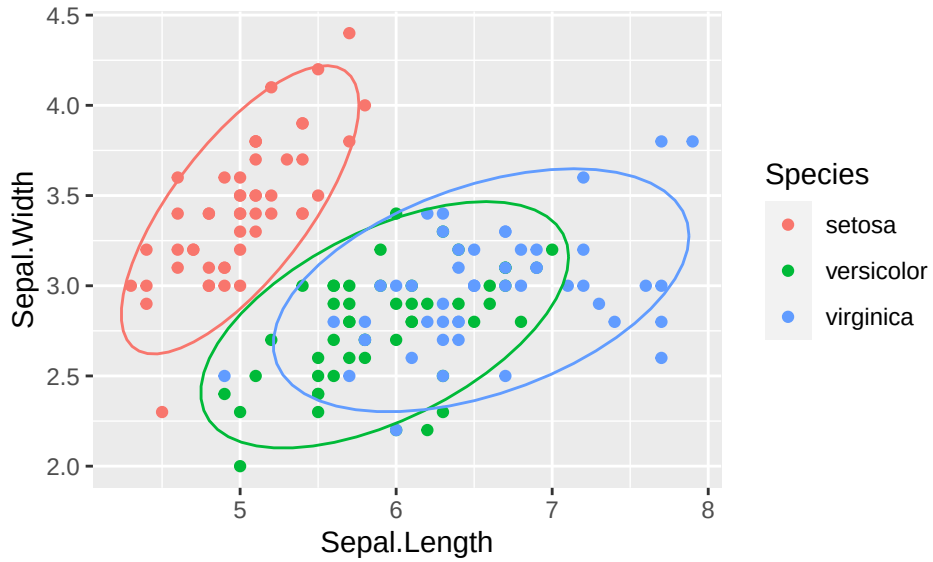
```
mtcars %>%
group_by(gear, carb) %>%
summarize(n = n()) %>%
mutate(freq = n / sum(n)) %>%
ggplot(aes(x = factor(gear), y = freq, fill = factor(carb))) +
geom_bar(stat = "identity", position = "dodge")
```

```
`summarise()` has grouped output by 'gear'. You can override using the
`.groups` argument.
```

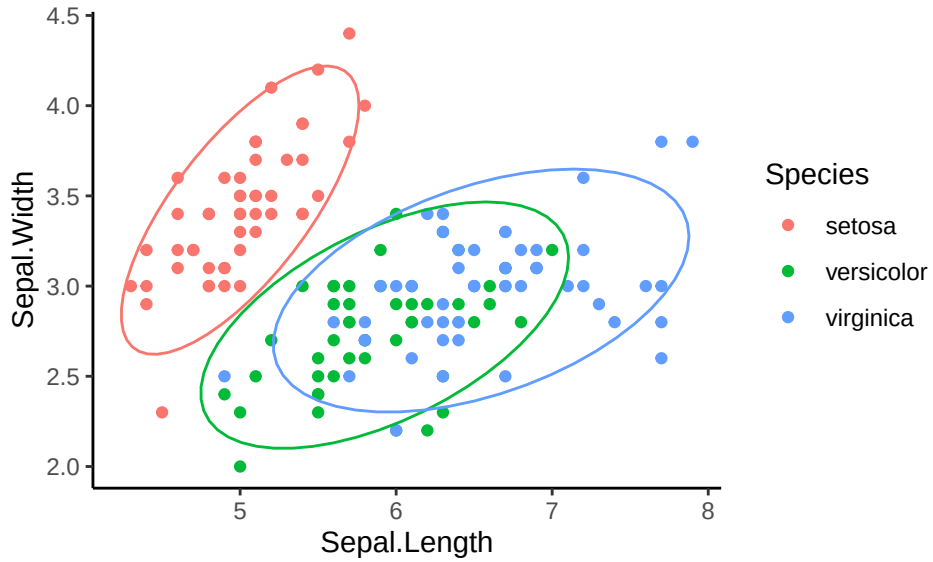


2. Detailing and Saving Plots

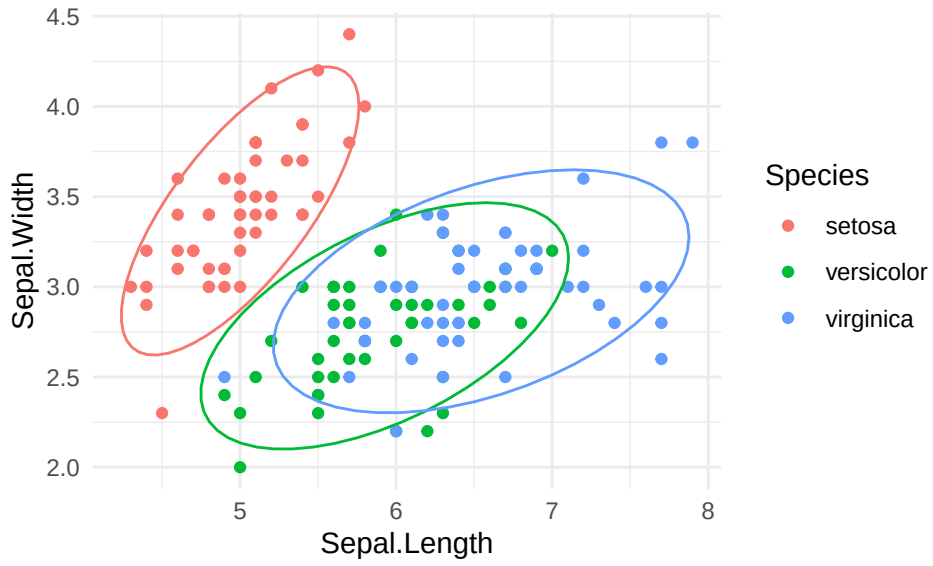
```
gg <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
geom_point() + stat_ellipse(show.legend = F)
print(gg)
```



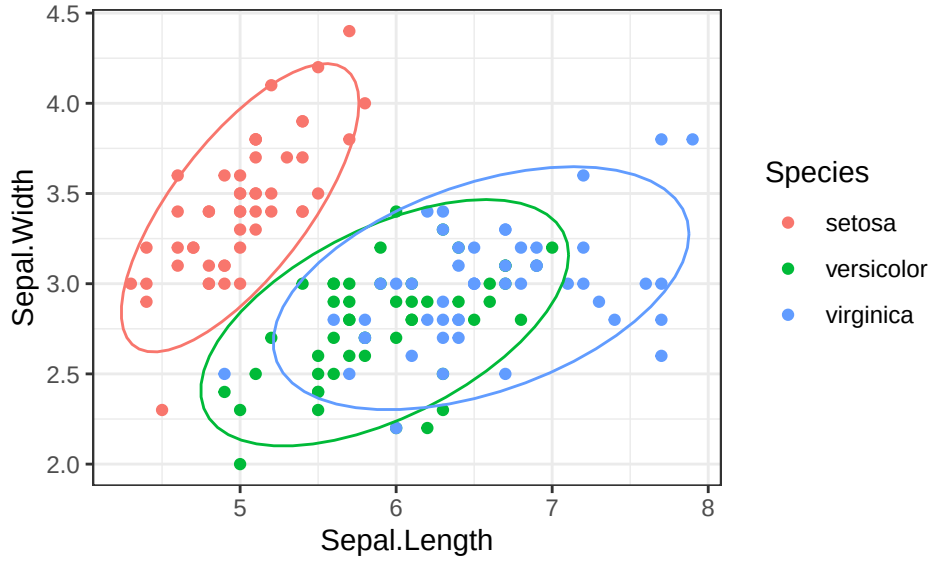

```
gg + theme_classic()
```




```
gg + theme_minimal()
```




```
gg + theme_bw()
```



Themes

Some other themes I like to use are theme_bw() and theme_minimal() link to list of themes: https://ggplot2.tidyverse.org/reference/ggtheme.html

```
library(RColorBrewer)
```

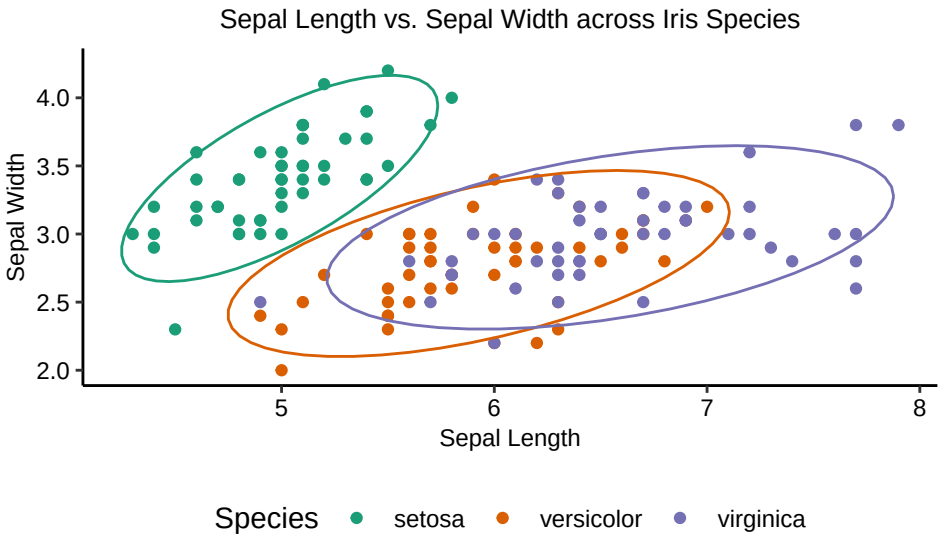
Incorporating many elements

preset theme + custom theme (custom elements must come after preset!!) setting axis limits with xlim/ylim changing labels changing colors with RColorBrewer

```
pretty_gg <- gg +
theme_classic() +
theme(axis.text = element_text(size = 9, color = "black"),
legend.position = "bottom", plot.title = element_text(hjust=0.5, size = 10),
axis.title = element_text(size = 9)) +
ylim(2, 4.25) +
labs(x = "Sepal Length", y = "Sepal Width", fill = "Species",
title = "Sepal Length vs. Sepal Width across Iris Species") +
scale_color_brewer(palette = "Dark2")
print(pretty_gg)
```

```
Warning: Removed 1 rows containing non-finite values (stat_ellipse).
```

```
Warning: Removed 1 rows containing missing values (geom_point).
```



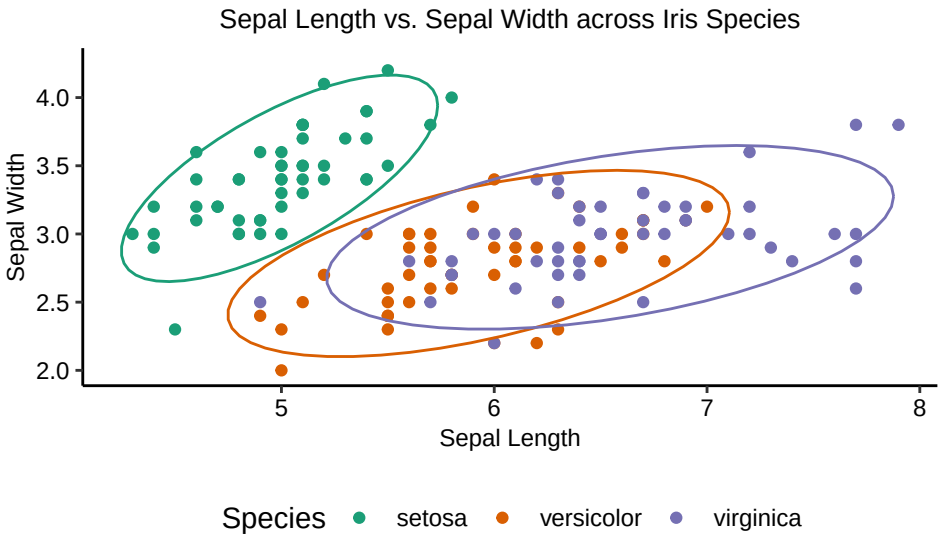
Saving Images

Save image as .png using "cairo" graphics device ("cairo" for windows, "Xlib" or "quartz" for mac) you may need to install the Cairo package with install.packages("Cairo") [this may also work for mac]

```
png(filename = "Iris Sepal Length vs Sepal Width.png", res = 300, type = "Xlib",
    height = 1200, width = 1200)
print(pretty_gg)
```

```
Warning: Removed 1 rows containing non-finite values (stat_ellipse).
```

```
Warning: Removed 1 rows containing missing values (geom_point).
```




```
dev.off()
```

```
pdf
  2
```

Without setting type to cairo, the resulting image quaility will be lower.

This particularly affects curved lines and circles

```
png(filename = "Iris Sepal Length vs Sepal Width no Cairo.png", res = 300, #won't be as clear
    height = 1200, width = 1200)
print(pretty_gg)
```

```
Warning: Removed 1 rows containing non-finite values (stat_ellipse).
```

```
Warning: Removed 1 rows containing missing values (geom_point).
```

```
dev.off()
```

```
pdf
  2
```

Set the units to inches for ease of use.

```
png(filename = "Iris Sepal Length vs Sepal Width.png", res = 300, type = "cairo",
    units = "in", height = 4, width = 4)
print(pretty_gg)
```

```
Warning: Removed 1 rows containing non-finite values (stat_ellipse).

Warning: Removed 1 rows containing missing values (geom_point).
```

```
dev.off()
```

```
pdf
  2
```

Another way to save images is with the ggpubr package.

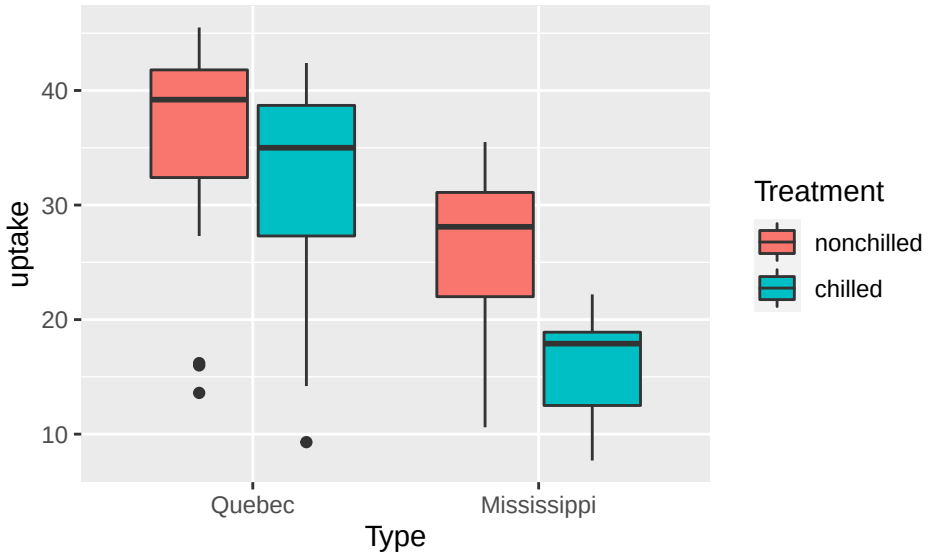
```
library(ggpubr)
ggsave("Iris Sepal Length vs Sepal Width.png", pretty_gg, dpi = 300, type = "cairo",
    height = 4, width = 4)
```

```
Warning: Using ragg device as default. Ignoring ‘type‘ and ‘antialias‘ arguments

Warning: Removed 1 rows containing non-finite values (stat_ellipse).

Warning: Removed 1 rows containing missing values (geom_point).
```

Ordering factors in boxplot/barchart

Simple boxplot

```
ggplot(CO2, aes(x = Type, y = uptake, fill = Treatment)) + geom_boxplot()
```



Create new factor combining Treatment and Type.

This allows us to use more colors if needed.

```
CO2$interaction <-  factor(paste0(CO2$Treatment, CO2$Type),
    levels = c("nonchilledMississippi", "chilledMississippi",
    "nonchilledQuebec", "chilledQuebec"))
```

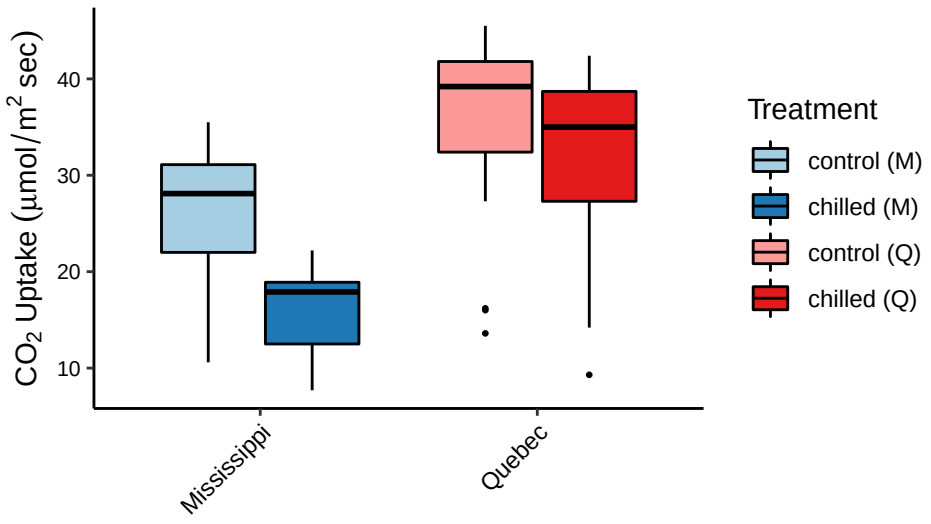
Customized ggplot boxplot

1. Setting x axis order of factors
2. Manually changing fill palette, still using RColorBrewer brewer.pal()
3. Changing labels
4. Preset theme + custom theme

```
boxplot_gg <- ggplot(CO2, aes(x = Type, y = uptake, fill = interaction)) +
geom_boxplot(color = "black", outlier.size = 0.5) +
scale_x_discrete(limits = c("Mississippi", "Quebec")) +
scale_fill_manual(values = brewer.pal("Paired", n = 9)[c(1,2,5,6)],
    labels = c("control (M)", "chilled (M)", "control (Q)", "chilled (Q)")) +
labs(x = "", y = expression(CO[2]~Uptake~(mu*mol/m^2~sec)), fill = "Treatment") +
theme_classic() +
theme(axis.text.x = element_text(size = 9, color = "black", angle = 45, hjust = 1),
    axis.text.y = element_text(size = 8, color = "black"))
```

Works the same as print(boxplot_gg) if you have a graphics device open.

```
boxplot_gg
```



```
ggsave("CO2 Uptake by Location by Treatment.png", boxplot_gg, dpi = 300, type = "cairo",
    height = 3, width = 3)
```

```
Warning: Using ragg device as default. Ignoring ‘type‘ and ‘antialias‘ arguments
```

3. Wide vs. Long Data Format

```
head(USPersonalExpenditure)
```

```
                      1940   1945  1950 1955  1960
Food and Tobacco    22.200 44.500 59.60 73.2 86.80
Household Operation 10.500 15.500 29.00 36.5 46.20
Medical and Health   3.530  5.760  9.71 14.0 21.10
Personal Care        1.040  1.980  2.45  3.4  5.40
Private Education    0.341  0.974  1.80  2.6  3.64
```

Cast as data frame for compatiblity with ggplot.

R doesn't like numeric column names, and adds an X to the beginning.

```
df <- data.frame(USPersonalExpenditure)
```

Use melt to transform the data into long format.

```
library(reshape)
```

```
Attaching package: 'reshape'

The following object is masked from 'package:dplyr':

    rename
```

```
USPE_melt <- melt(USPersonalExpenditure)
```

```
Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

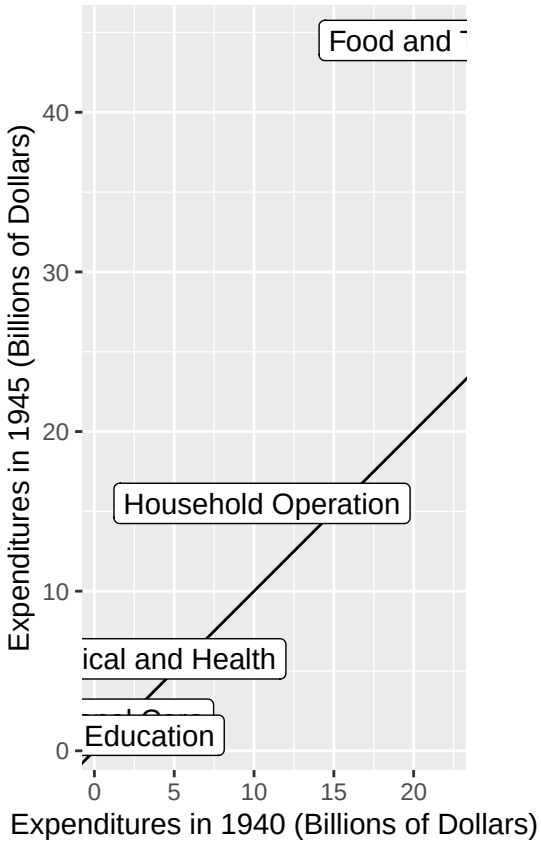
Add some descriptive column names.

```
colnames(USPE_melt) <- c("variable", "year", "value")
```

Scatterplot comparing expenditures in 1940 and 1945 with wide format data.

Here, coord_fixed() forces the axes to the same scale. geom_label adds text labels to the plot, but they are cutoff by the plot margins.

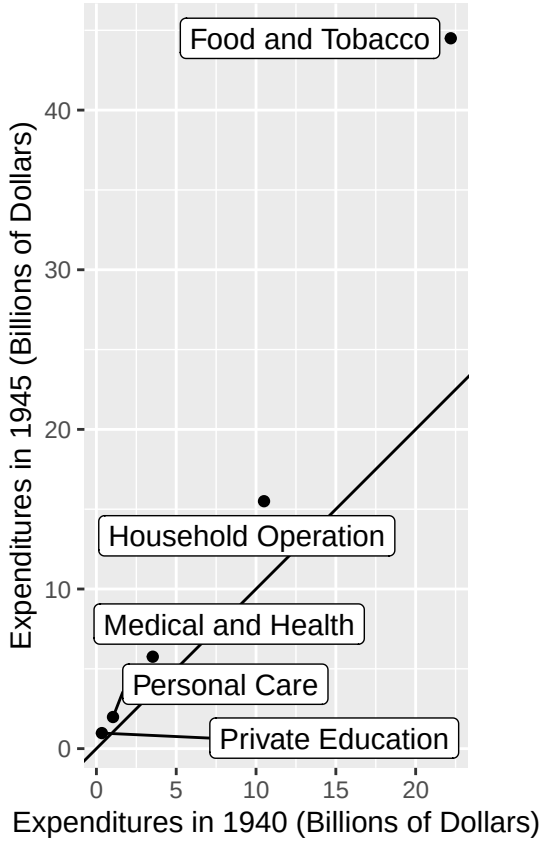
```
ggplot(df, aes(x = X1940, y = X1945)) +
geom_point() +
geom_abline() +
geom_label(aes(label = rownames(df))) +
coord_fixed() +
labs(x = "Expenditures in 1940 (Billions of Dollars)",
    y = "Expenditures in 1945 (Billions of Dollars)")
```



We can use geom_label_repel (and geom_text_repel) to add non-overlapping labels.

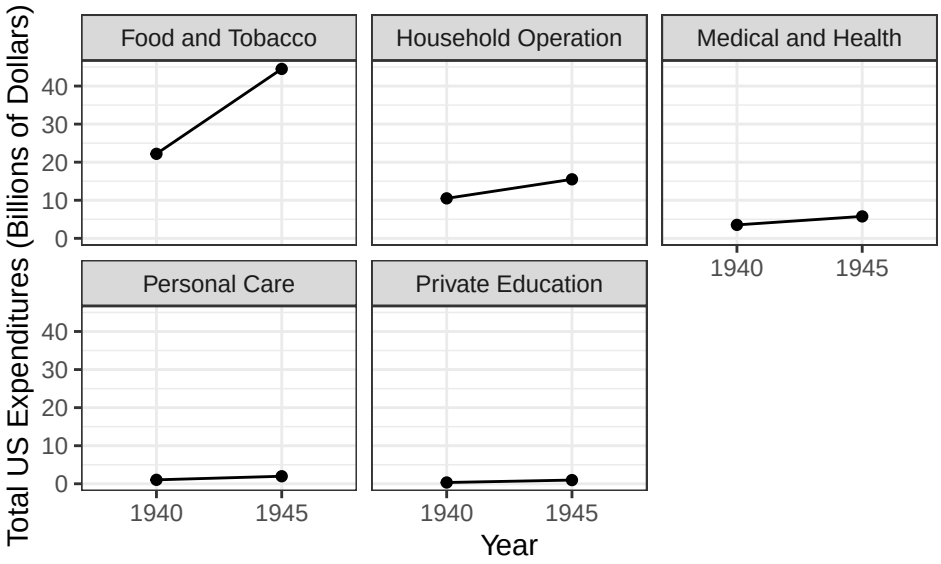
THIS ONE IS MUCH BETTER. NO OVERLAP.

```
library(ggrepel)
ggplot(df, aes(x = X1940, y = X1945)) +
geom_point() +
geom_abline() +
geom_label_repel(aes(label = rownames(df)), force = 20) +
coord_fixed() +
labs(x = "Expenditures in 1940 (Billions of Dollars)",
    y = "Expenditures in 1945 (Billions of Dollars)")
```



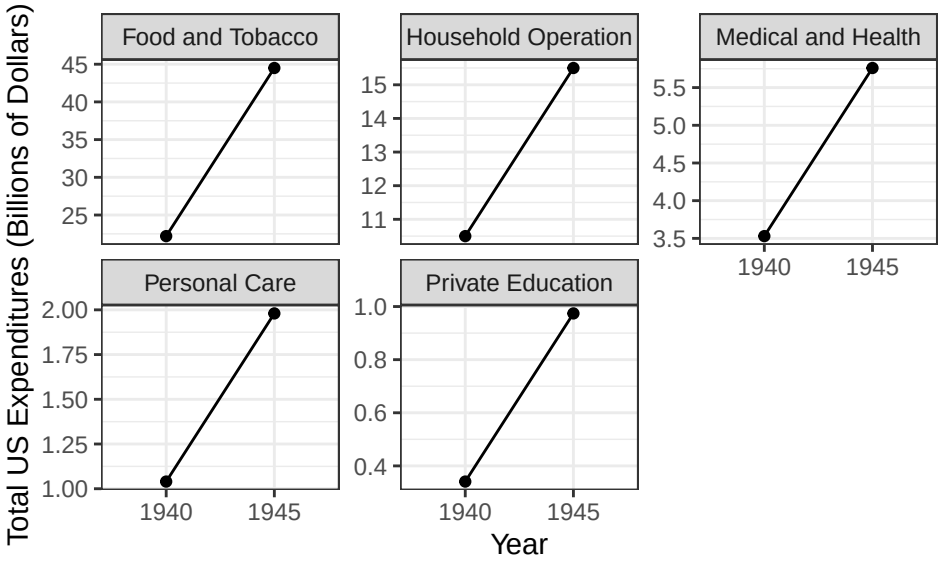
Another way to visualize the difference in expenditures with long format data.

```
USPE_melt %>%
filter(year %in% c(1940, 1945)) %>%
ggplot(aes(x = factor(year), y = value, group = 1)) +
geom_line() +
geom_point() +
facet_wrap(~ variable) +
labs(x = "Year", y = "Total US Expenditures (Billions of Dollars)") +
theme_bw()
```



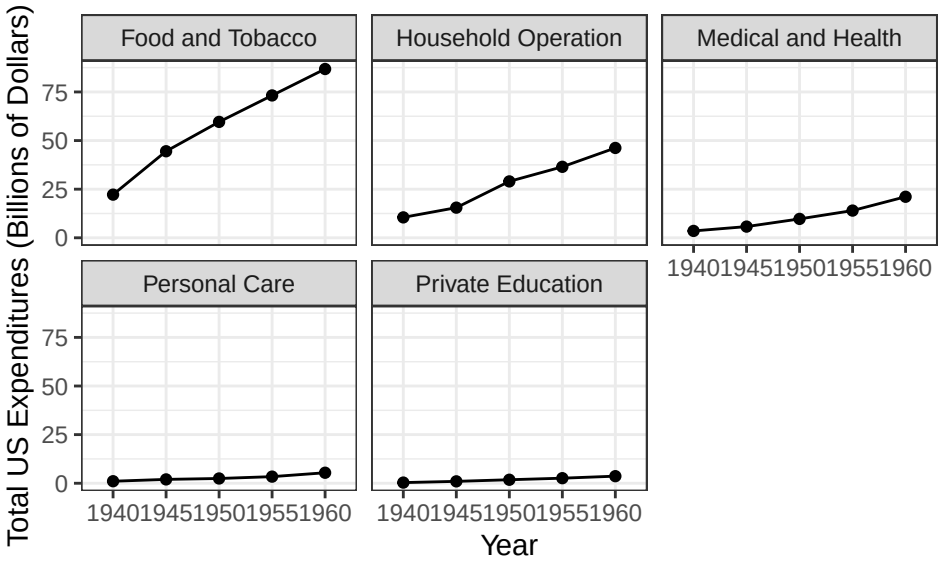
If we scale the data independently it may become less intuitive.

```
USPE_melt %>%
filter(year %in% c(1940, 1945)) %>%
ggplot(aes(x = factor(year), y = value, group = 1)) +
geom_line() +
geom_point() +
facet_wrap(~ variable, scale = "free_y") +
labs(x = "Year", y = "Total US Expenditures (Billions of Dollars)") +
theme_bw()
```



Long format data enables us to look at even more data points.

```
ggplot(USPE_melt, aes(x = factor(year), y = value, group = 1)) +
geom_line() +
geom_point() +
facet_wrap(~ variable) +
labs(x = "Year", y = "Total US Expenditures (Billions of Dollars)") +
theme_bw()
```



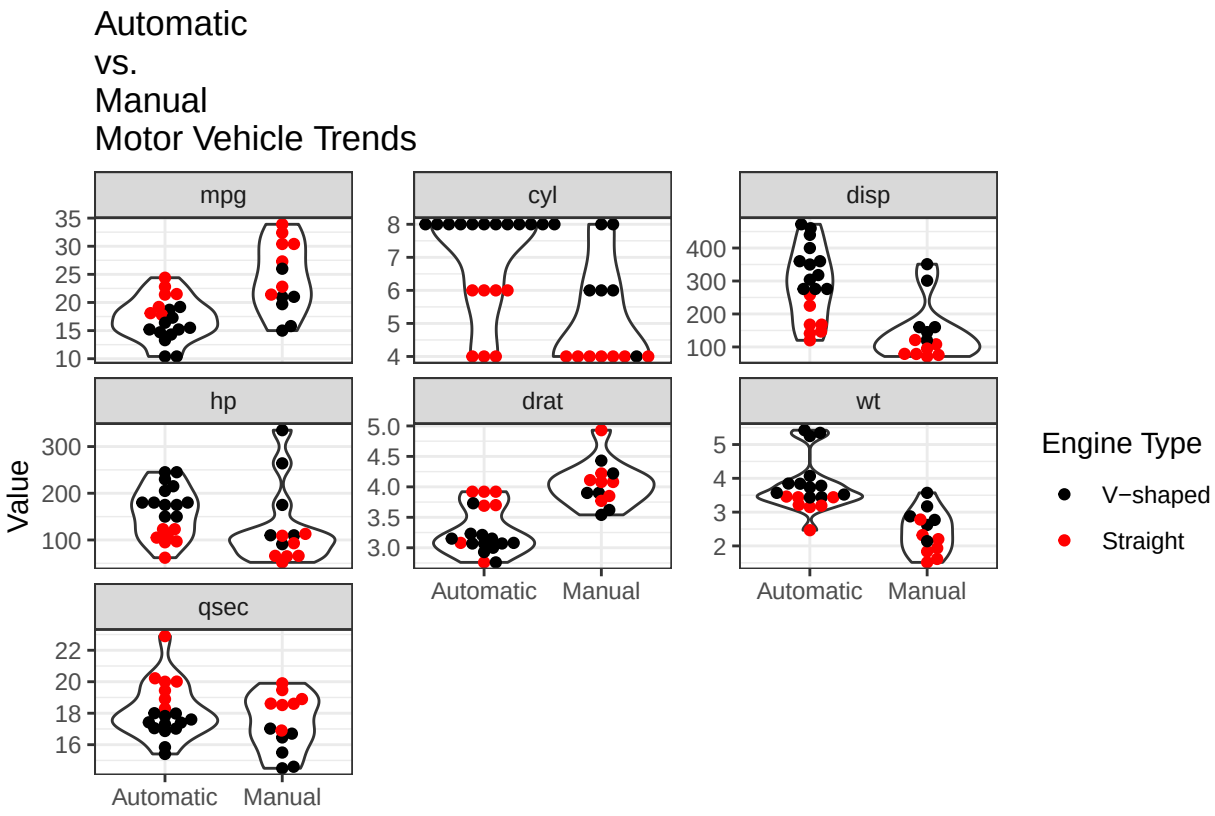
Specifying certain columns you do not want in long format.

These wil be for factor levels you plan on using later while plotting.

```
mtcars_melt <- melt(mtcars, id.vars = c("vs", "am", "gear", "carb"))
```

Another example combining many layers and parameters.

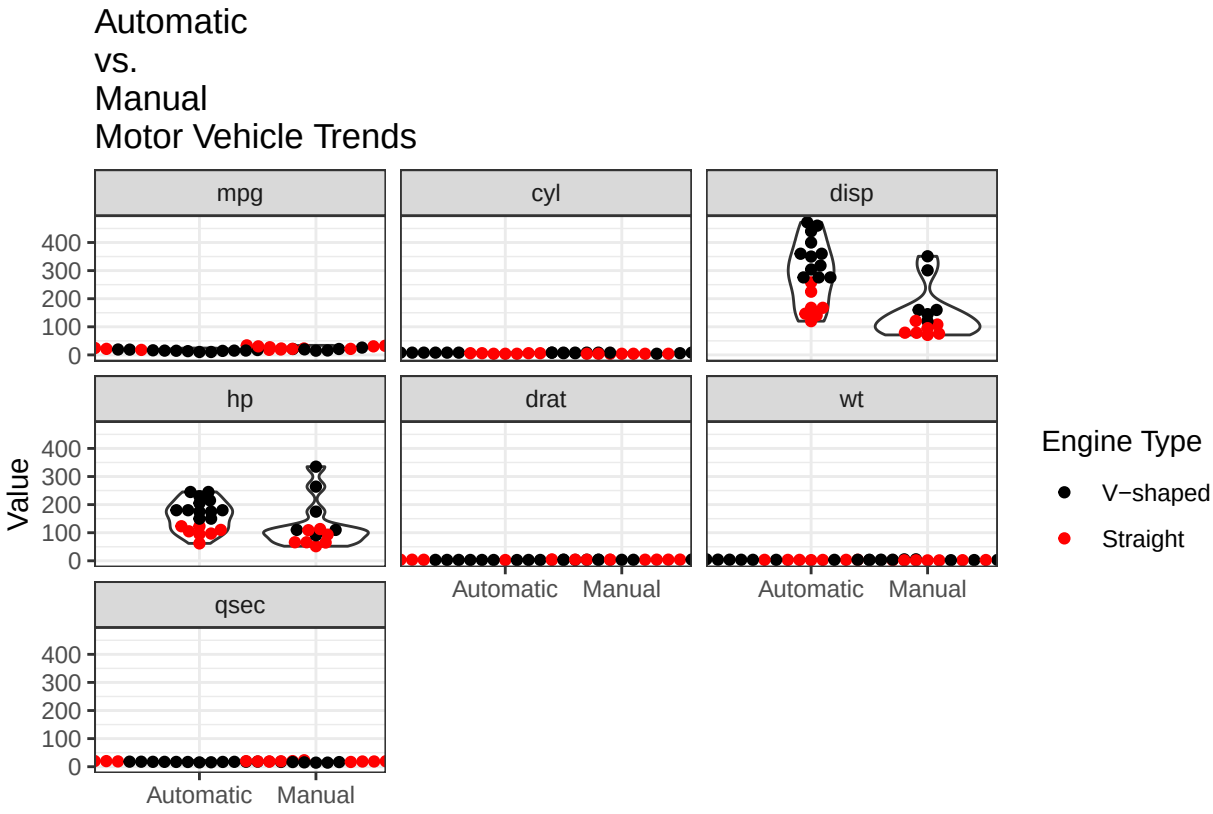
```
ggplot(mtcars_melt, aes(y = value, x = factor(am))) +
geom_violin() +
geom_beeswarm(aes(color = factor(vs)), cex = 5) +
facet_wrap(~ variable, scale = "free_y") +
theme_bw() +
scale_x_discrete(limits = factor(c(0,1)), labels = c("Automatic", "Manual")) +
labs(x = "", y = "Value", color = "Engine Type", title = "Automatic\nvs.\nManual\nMotor Vehicle Trends")
scale_color_manual(values = c("black", "red"), labels = c("V-shaped", "Straight"))
```



Here, independent scaling of the y-axis is preferred.

If we don't some variables can't be viewed properly.

```
ggplot(mtcars_melt, aes(y = value, x = factor(am))) +
geom_violin() +
geom_beeswarm(aes(color = factor(vs)), cex = 5) +
facet_wrap(~ variable) +
theme_bw() +
scale_x_discrete(limits = factor(c(0,1)), labels = c("Automatic", "Manual")) +
labs(x = "", y = "Value", color = "Engine Type", title = "Automatic\nvs.\nManual\nMotor Vehicle Trends")
scale_color_manual(values = c("black", "red"), labels = c("V-shaped", "Straight"))
```


Automatic
vs.
Manual
Motor Vehicle Trends
mpg
cyl
disp
hp
drat
wt
qsec
Value
400
300
200
100
0
Automatic
Manual
Engine Type
V−shaped
Straight