# Lab 12

## Installing Packages

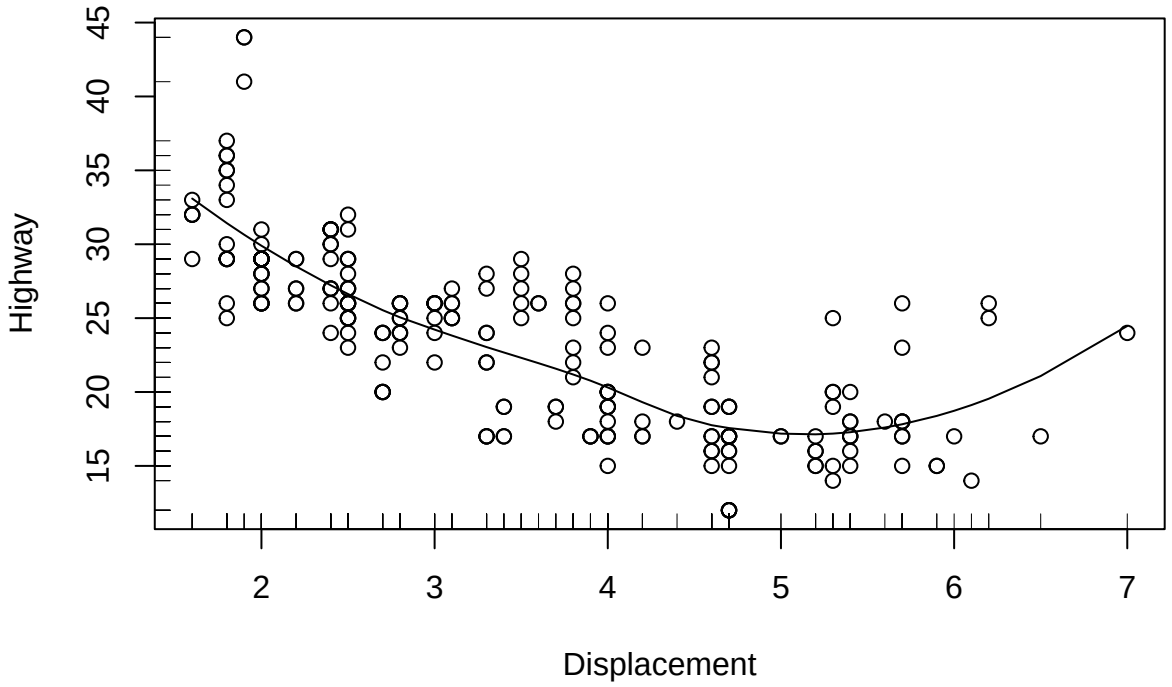
```
# install.packages('ggplot2')
# install.packages('lattice')

library(ggplot2)
library(lattice)
data("mpg")
```

## Background

```
##baseR
## Make a scatterplot
plot(x = mpg$displ, y = mpg$hwy, xlab = "Displacement", ylab = "Highway")
## Add loess curve
lout <- loess(hwy ~ displ, data = mpg)
ord_x <- order(lout$x)
ord_x
```

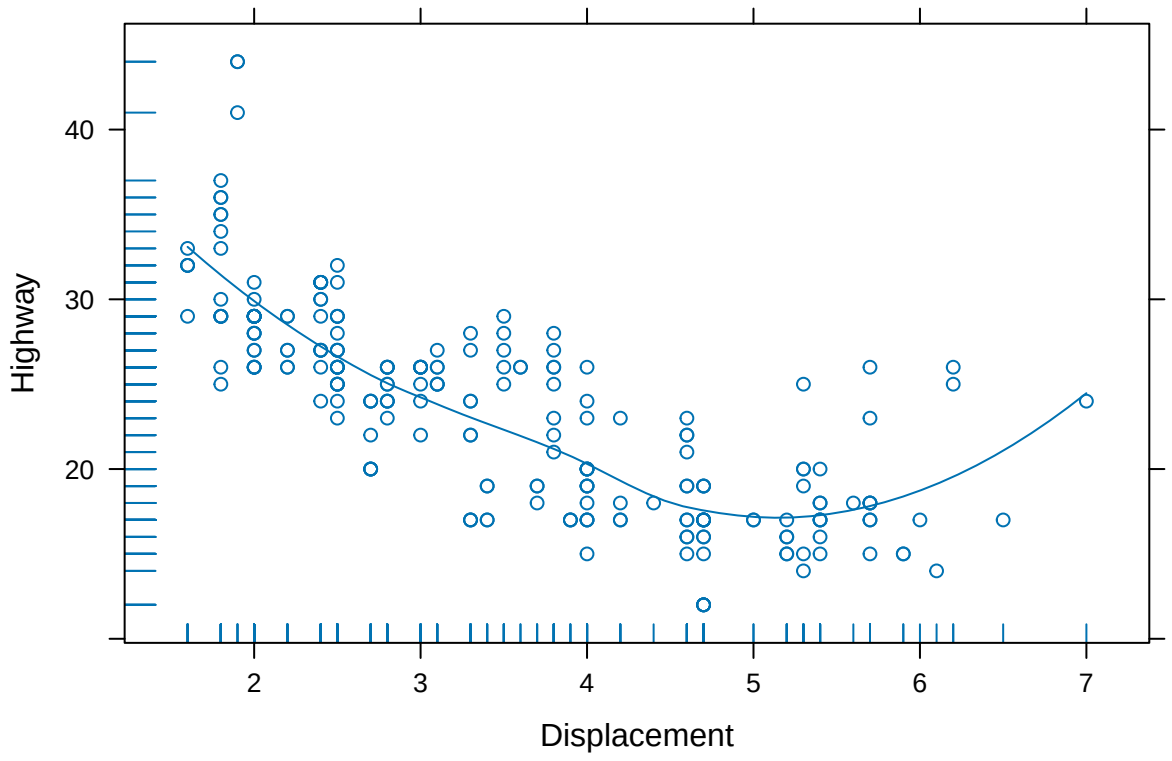
```
##   [1] 100 101 102 103 104   1   2   8   9 105 106 107 194 195 196 197 198 228
##  [19] 229 213 222 223   3   4  10  11 108 116 117 118 119 208 209 210 211 214
##  [37] 215 216 217 224 225 230 231 166 167 180 181 187 188  33  34  38 109 110
##  [55] 111 112 142 143 182 183 189 190 113 114 144 145 160 161 162 163 164 165
##  [73] 168 169 170 171 172 173 218 219 226 227 120 121 122 174 175 201 202 203
##  [91]   5   6  12  13  16 212 220 221 232 233  39 123 148 149 184 185 191 192
## [109]   7  14  15  17  35 155  40  41  42  43  44 115 151 152 193 176 177 204
## [127] 205  36 146 147 150 186  37 234  49  50 124  45  46  47  91  92 156 157
## [145] 158  51  52  58  48  78  79  80  81  93  94 125 131 138 139 153 178 206
## [163] 207  18  84  85 132 133  75  82  86  87  88  95  96  97  98 134 140  53
## [181]  54  55  59  60  61  65  66  67  68  69  70 126 127 128 179 199  83 141
## [199]  56  57  62  71  72  19  20  21  29  30 159  76  77  89  90  99 135 136
## [217] 137 154  22  24  25  31  63  73 129 200  64  74  23 130  26  27  32  28
```

```
lines(lout$x[ord_x], lout$fitted[ord_x])
## Add two rug plots
rug(mpg$displ, side = 1)
rug(mpg$hwy, side = 2)
```




We add everything at once.

```
##Lattice
xyplot(hwy ~ displ, data = mpg,
xlab = "Displacement", ylab = "Highway",
panel = function(x, y) {
panel.xyplot(x = x, y = y) ## Add scatterplot
panel.loess(x = x, y = y, span = 0.75,
family = "gaussian", degree = 2) ## Add loess curve
panel.rug(x = x, y = y) ## Add rug plot
})
```

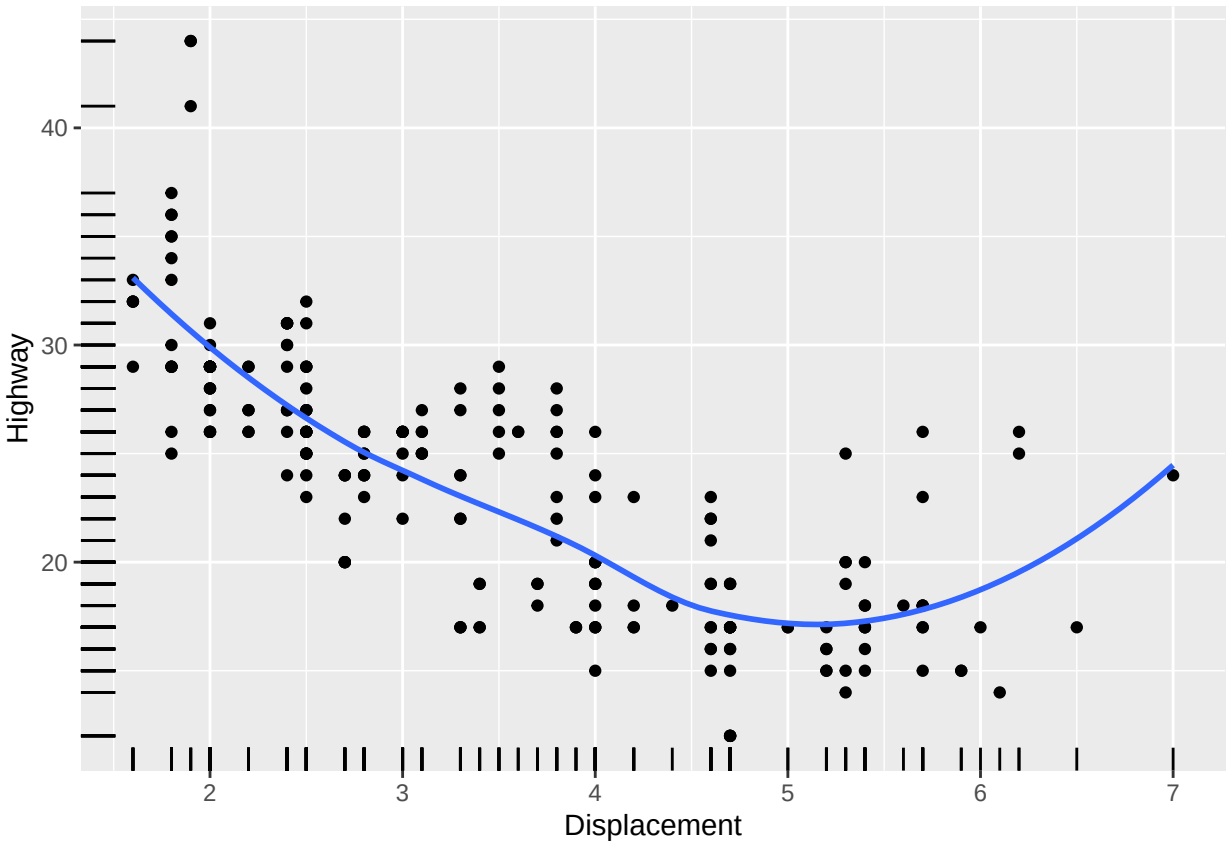



#gg-

Plot2 Scatterplot

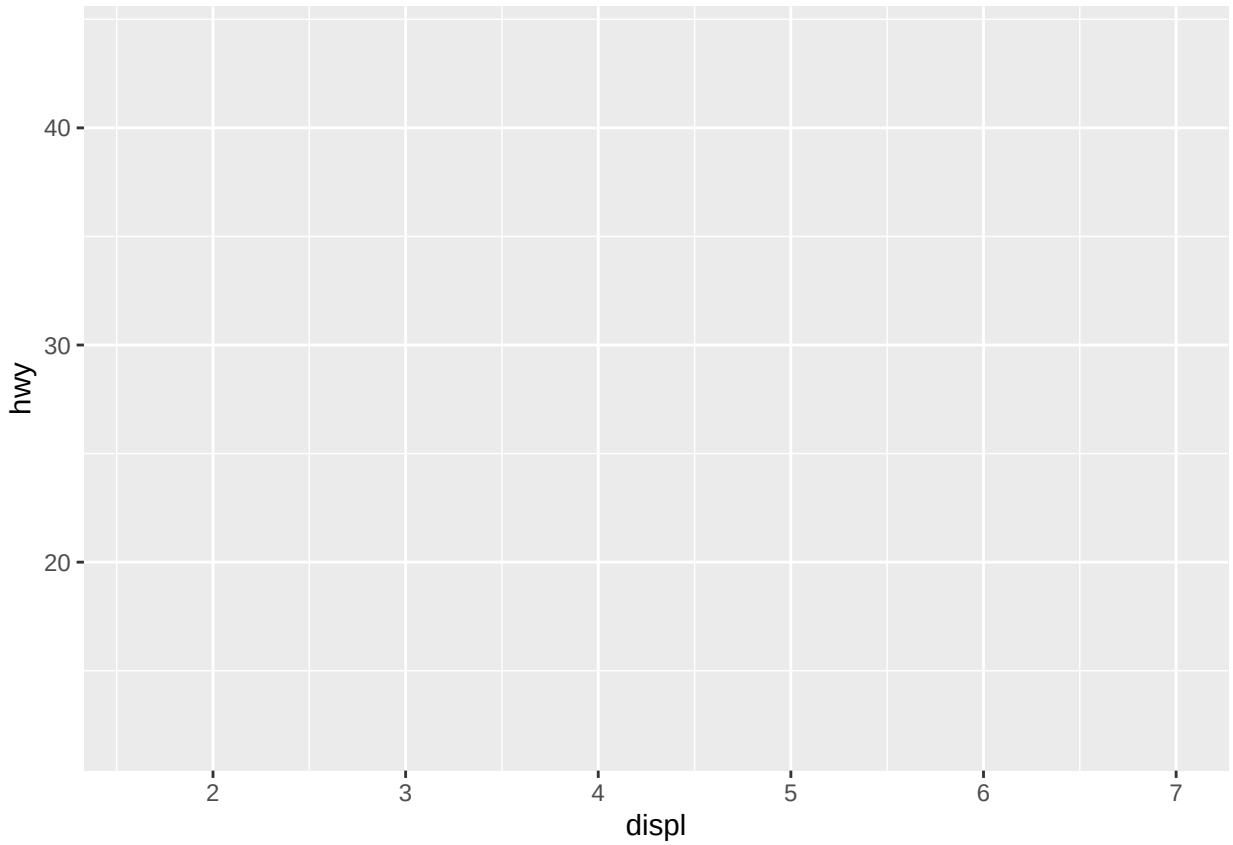
```
## ggPlot2
ggplot(mpg, mapping = aes(x = displ, y = hwy)) + ## set aesthetic maps
geom_point() + ## apply aesthetic maps to
## point objects
geom_smooth(se = FALSE, method = loess) + ## apply to smooth objects
geom_rug() + ## apply to rug objects
xlab("Displacement") +
ylab("Highway")
```

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




##ggplot2 Foundation

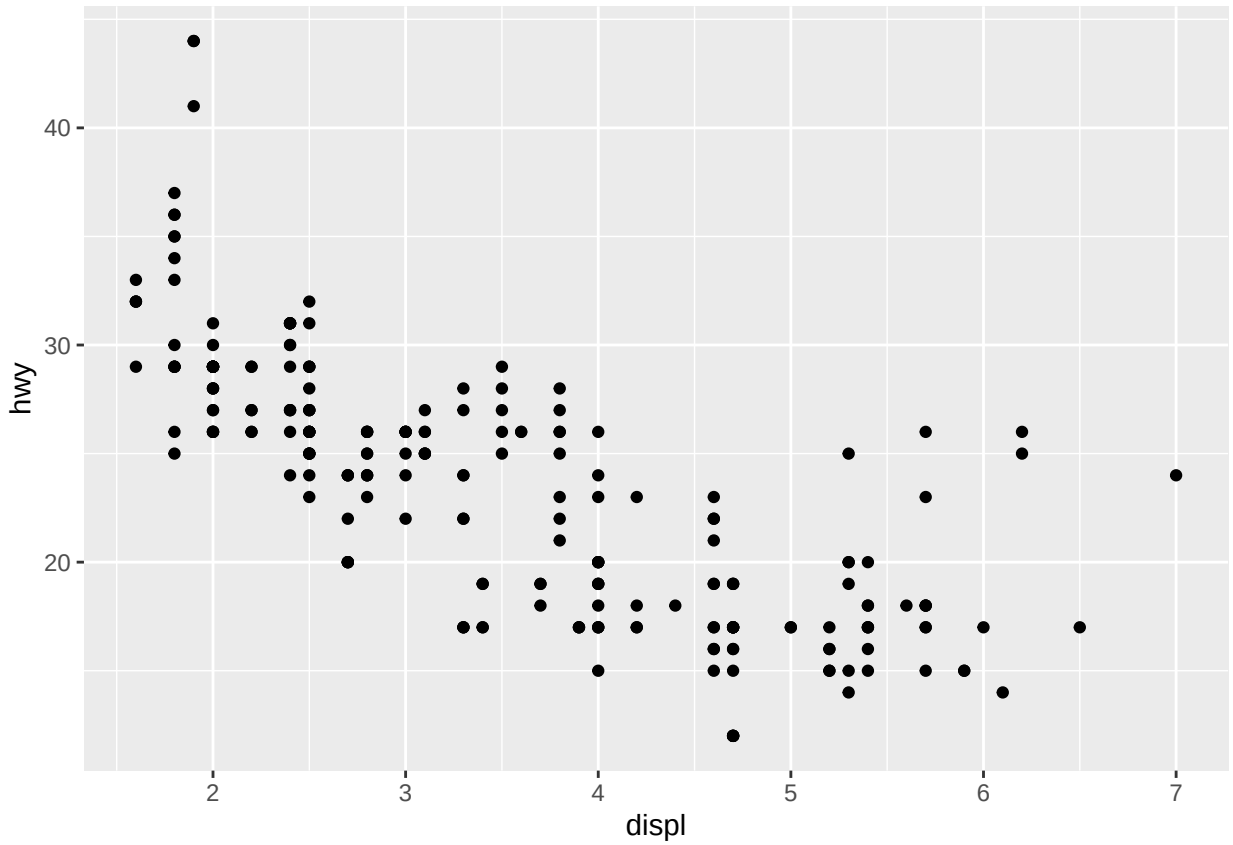
```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))
```



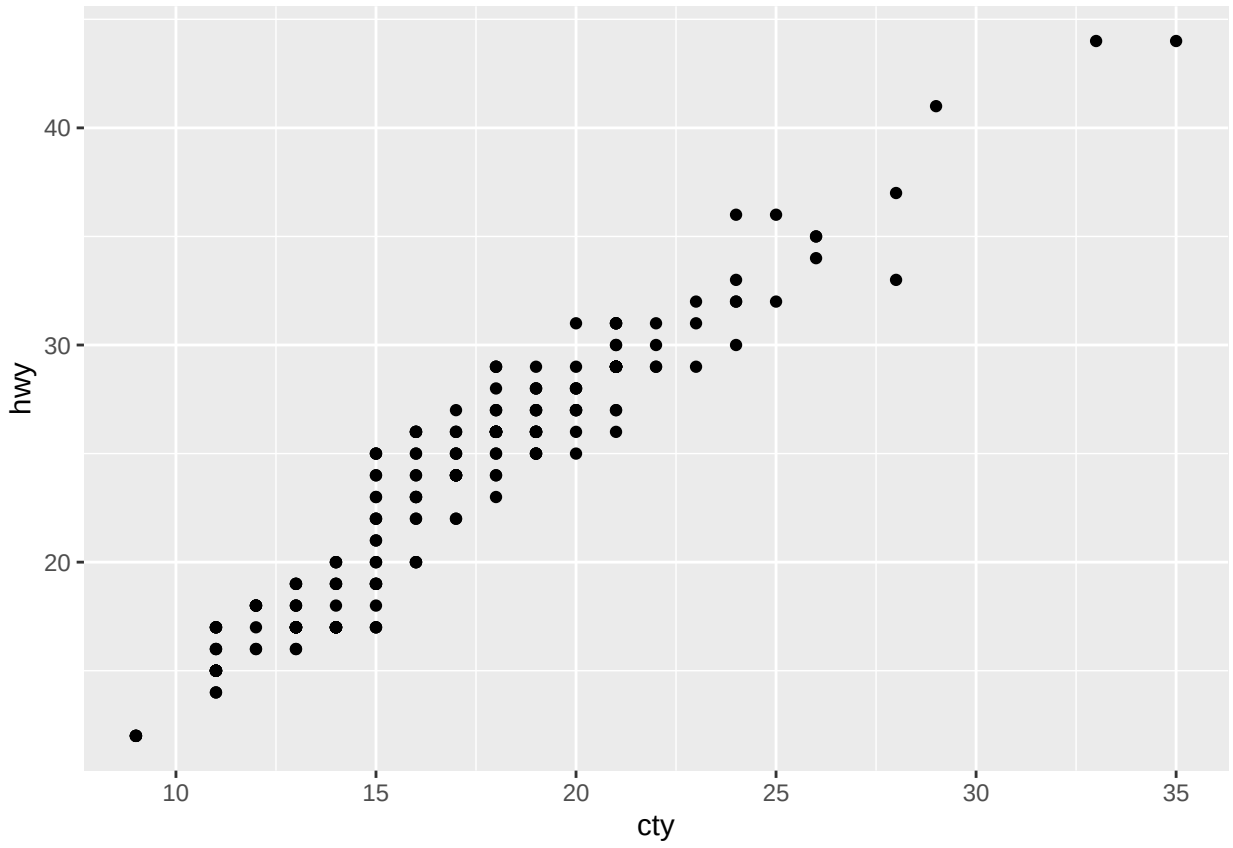

```
#data= data frame you want to use
#mapping: definition for mapping variables onto aesthetics, almost always use aes,
#in this case mapping x and y onto x and y coordinates
```

##Comparing two quantitative variables

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() # indicates scatterplot
```

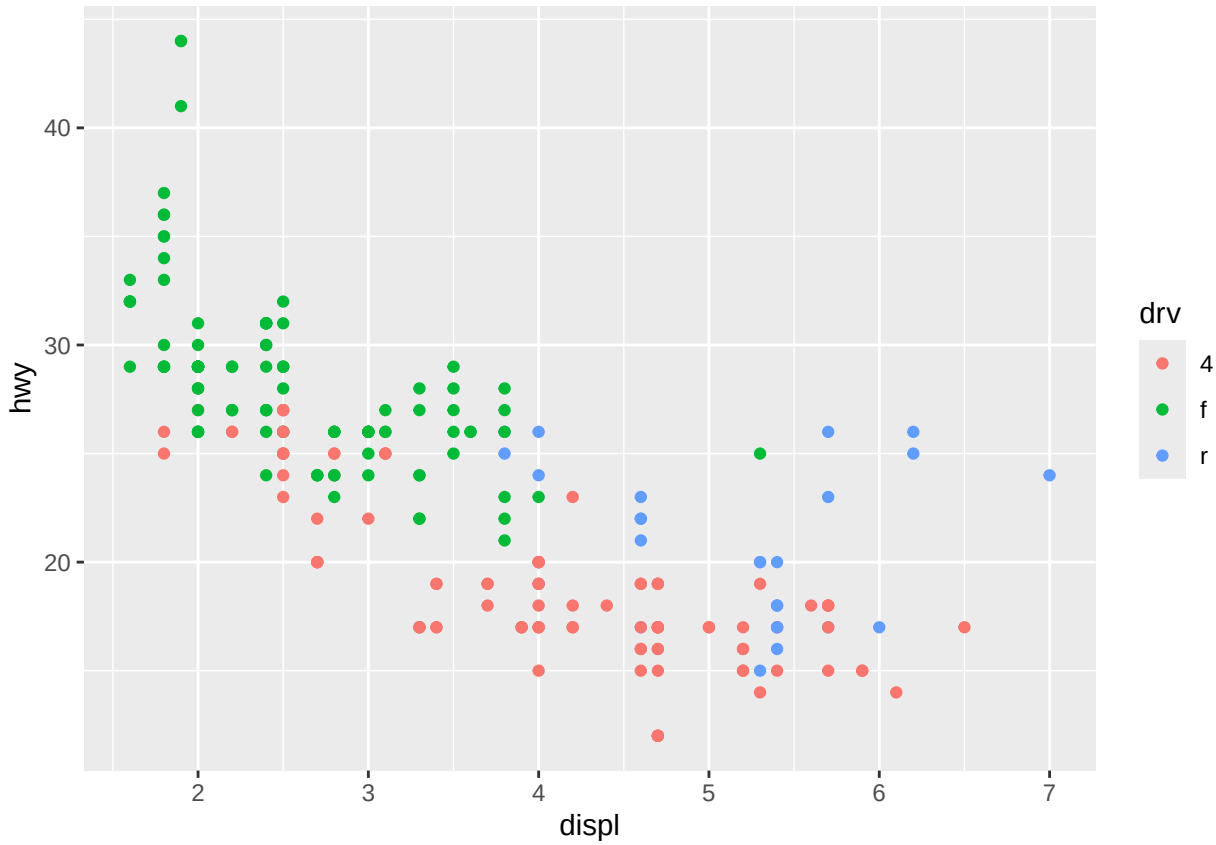



```
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
geom_point()
```

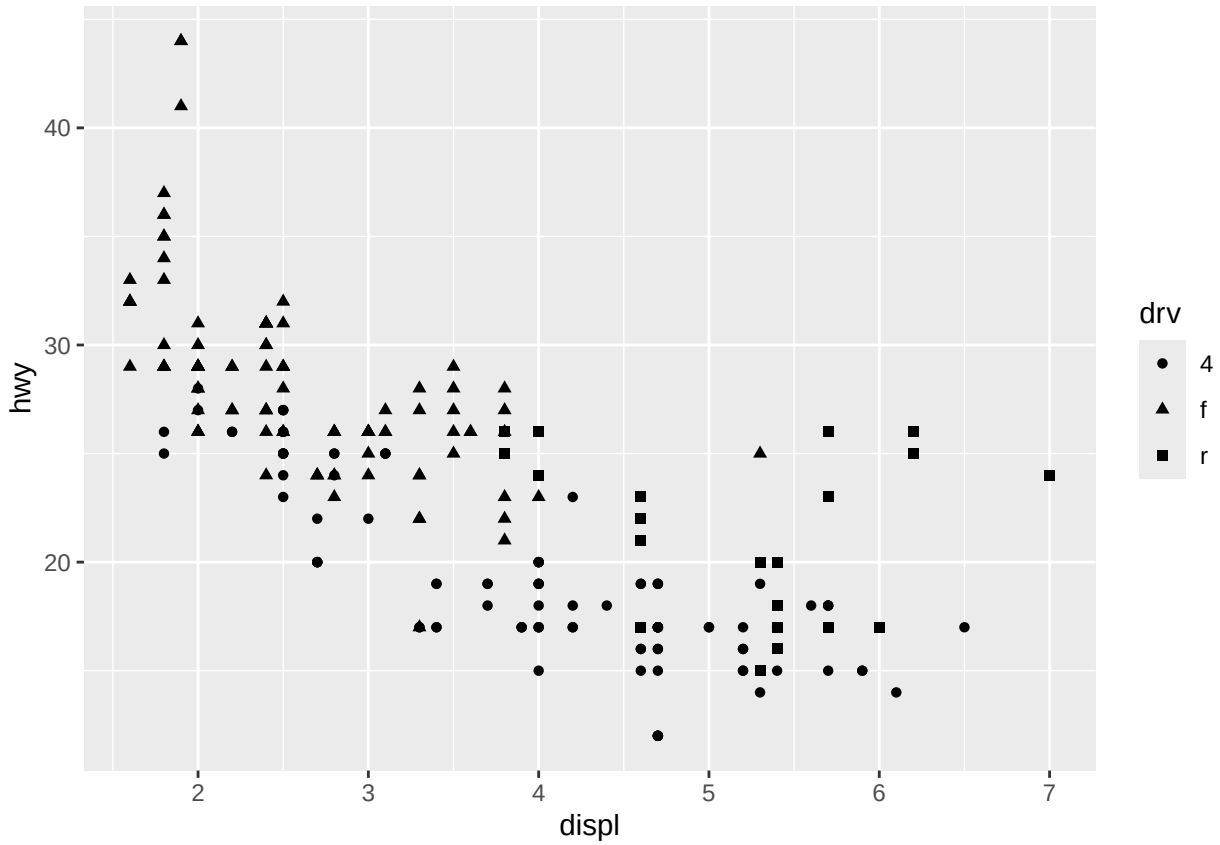



**Annotating by Categorical Variable**

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
geom_point()
```




```
#color=drv, indicates vcolor of points on scatterplot depends on value for drv
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, shape = drv)) +
geom_point()
```




```
#shape=drv indicates shape of points on scatterplot depends on value for drv
```

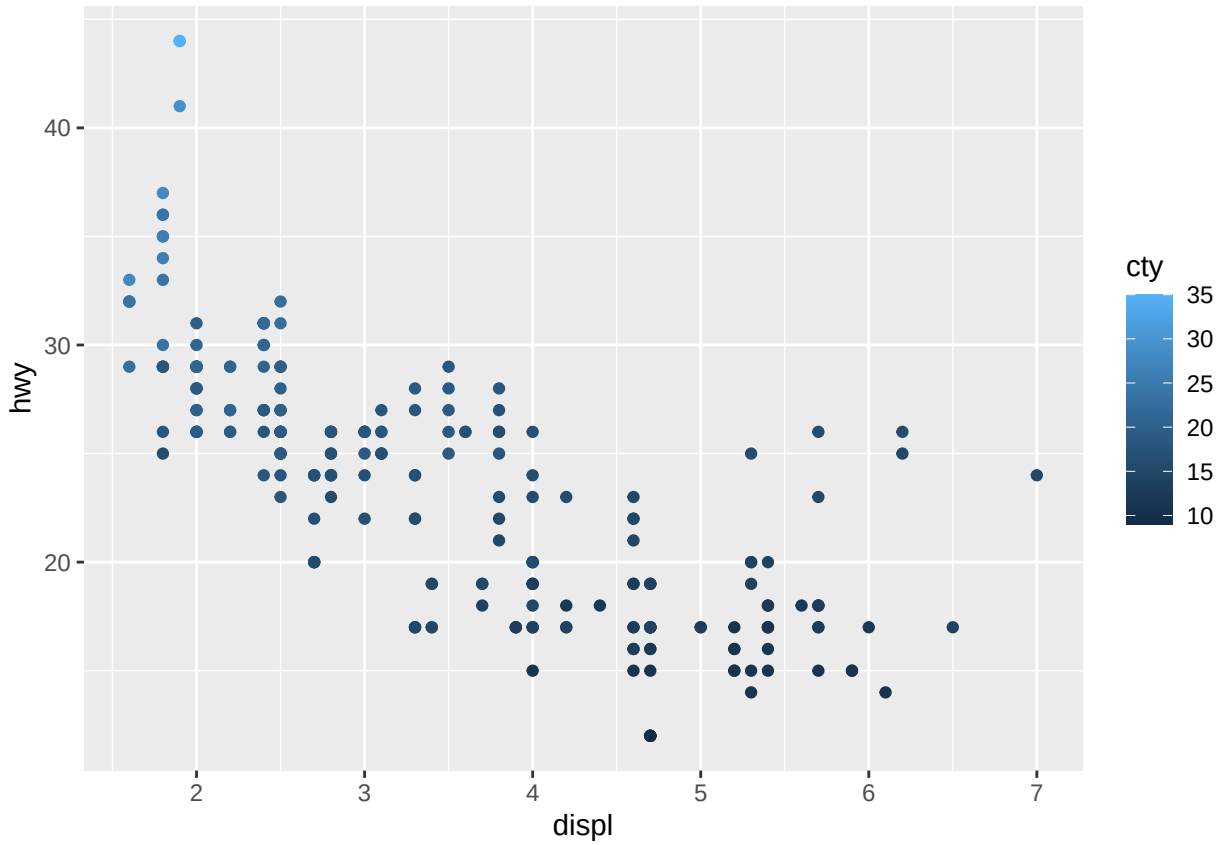
Exercise: What happens when you map a categorical variable to the size or alpha aesthetics?
Exercise: Consider the penguins dataset that comes from the palmer penguins package:

```
#install.packages("palmerpenguins")
library(palmerpenguins)
data("penguins")
```

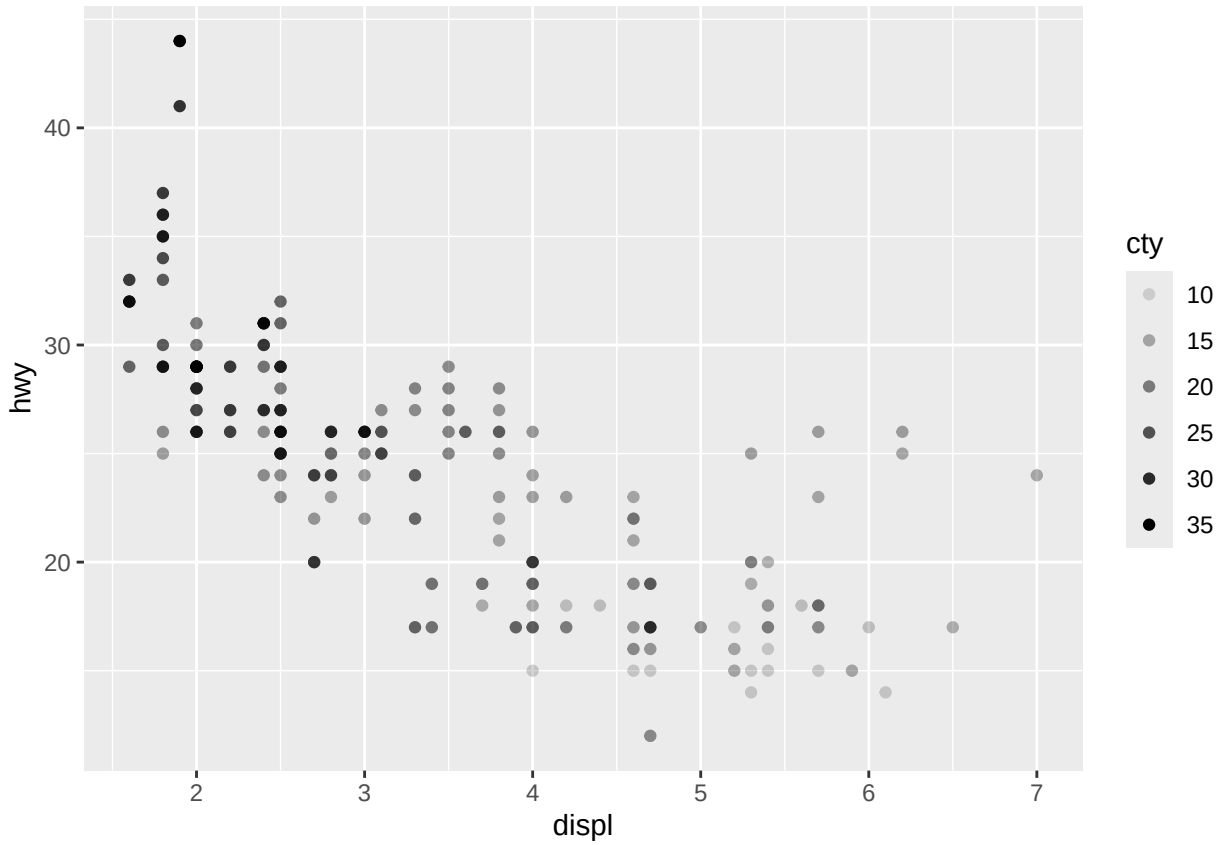
Make a plot of bill length vs bill depth, color coding by the species of penguin. What do you notice?

### Annotating by Quantitative Variable

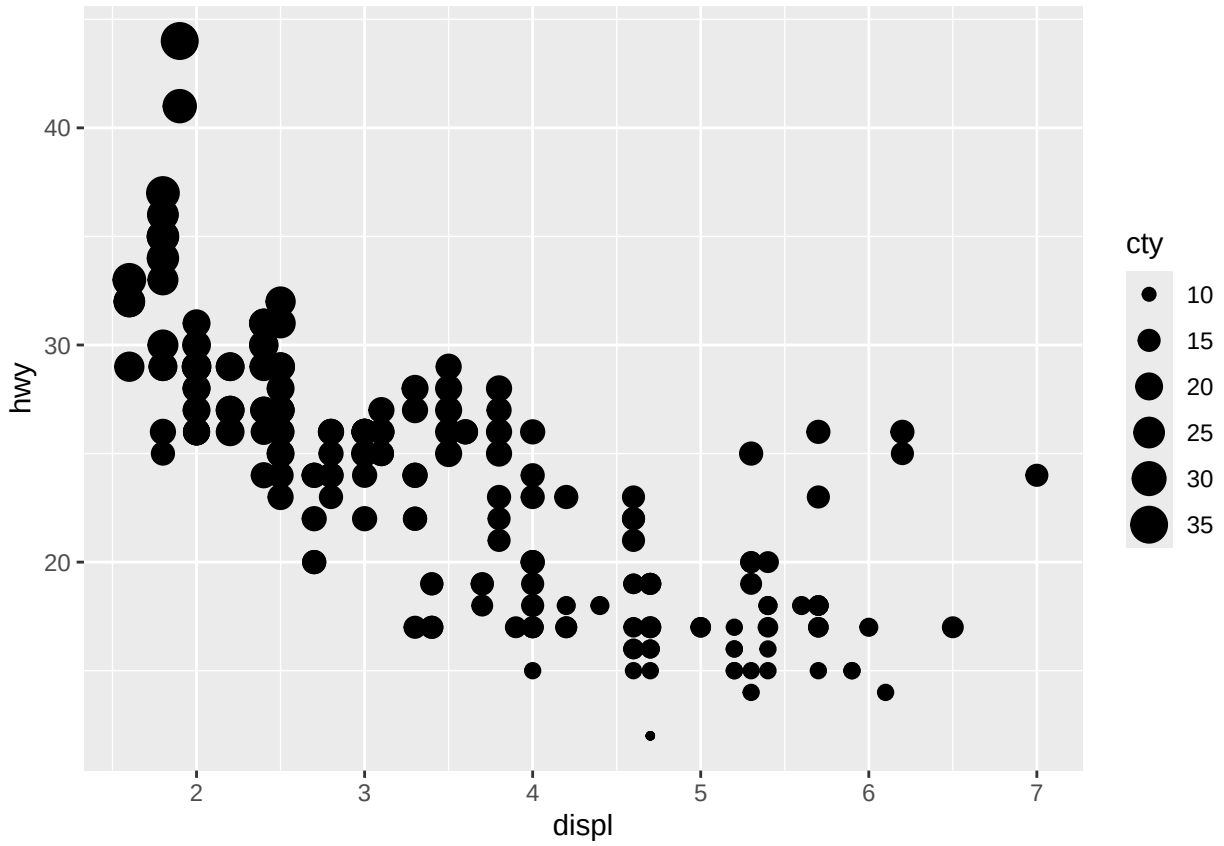
```
# Specifying quantitative  variable to determine color of point
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = cty)) +
geom_point()
```




```
# Specifying quantitative variable to determine shading of point
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, alpha = cty)) +
geom_point()
```



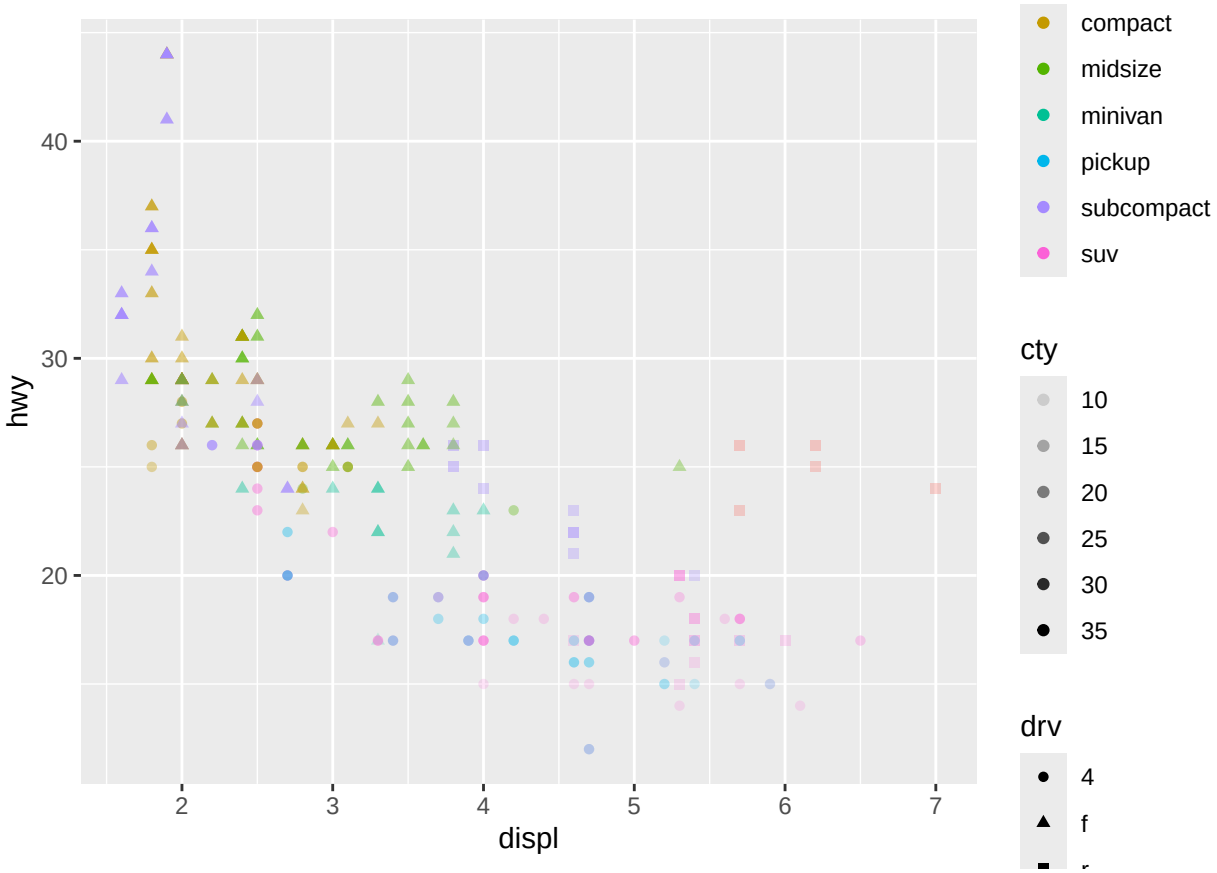

```
# Specifying quantitative variable to determine size of point
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, size = cty)) +
geom_point()
```




- Exercise:

What happens when we map a quantitative variable to the shape aesthetic?

```
# Shading dtermined by cty, shape by drv, color by class
ggplot(data = mpg, mapping = aes(x = displ,
y = hwy,
alpha = cty,
shape = drv,
color = class)) +
geom_point()
```

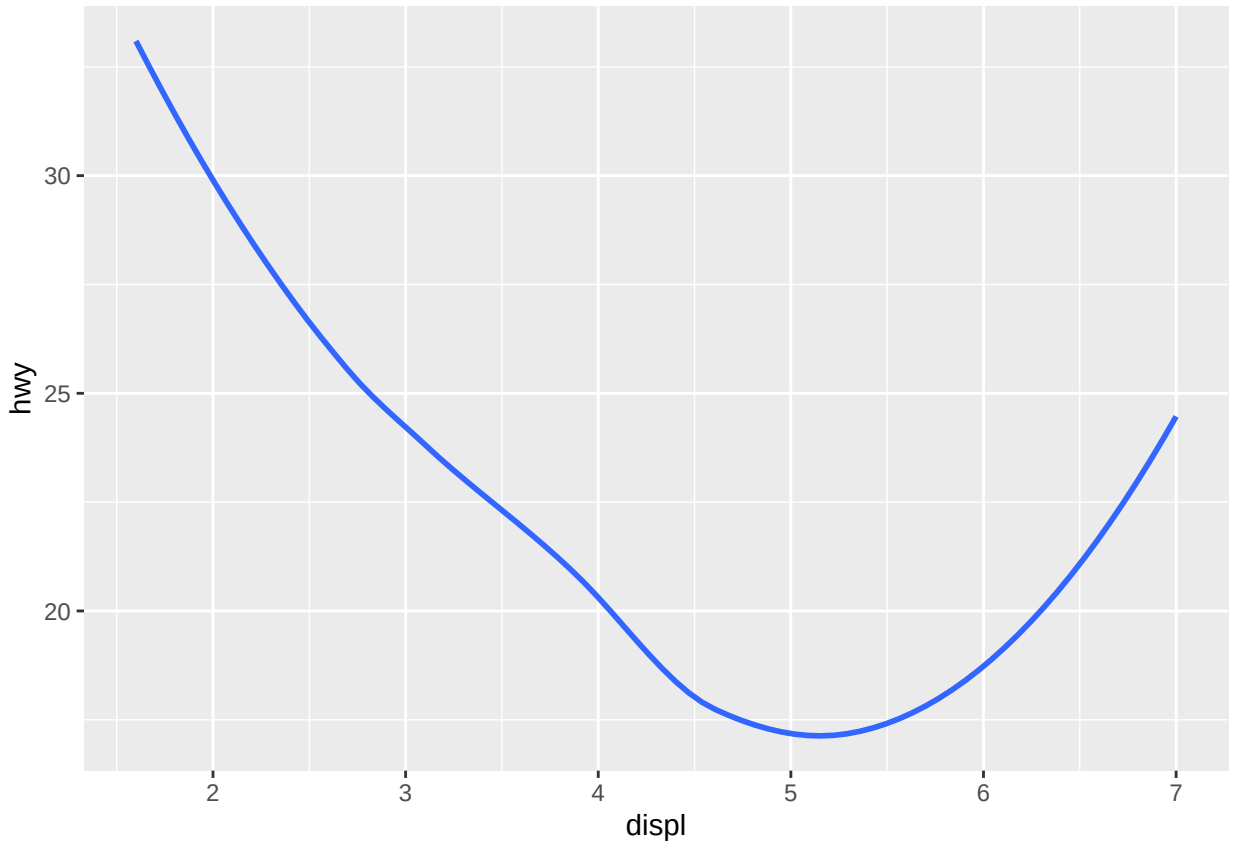



Exercise: Using the penguins dataset, plot bill length vs bill depth, annotating by flipper length and body mass and species. Do you notice anything?

### Smoothing

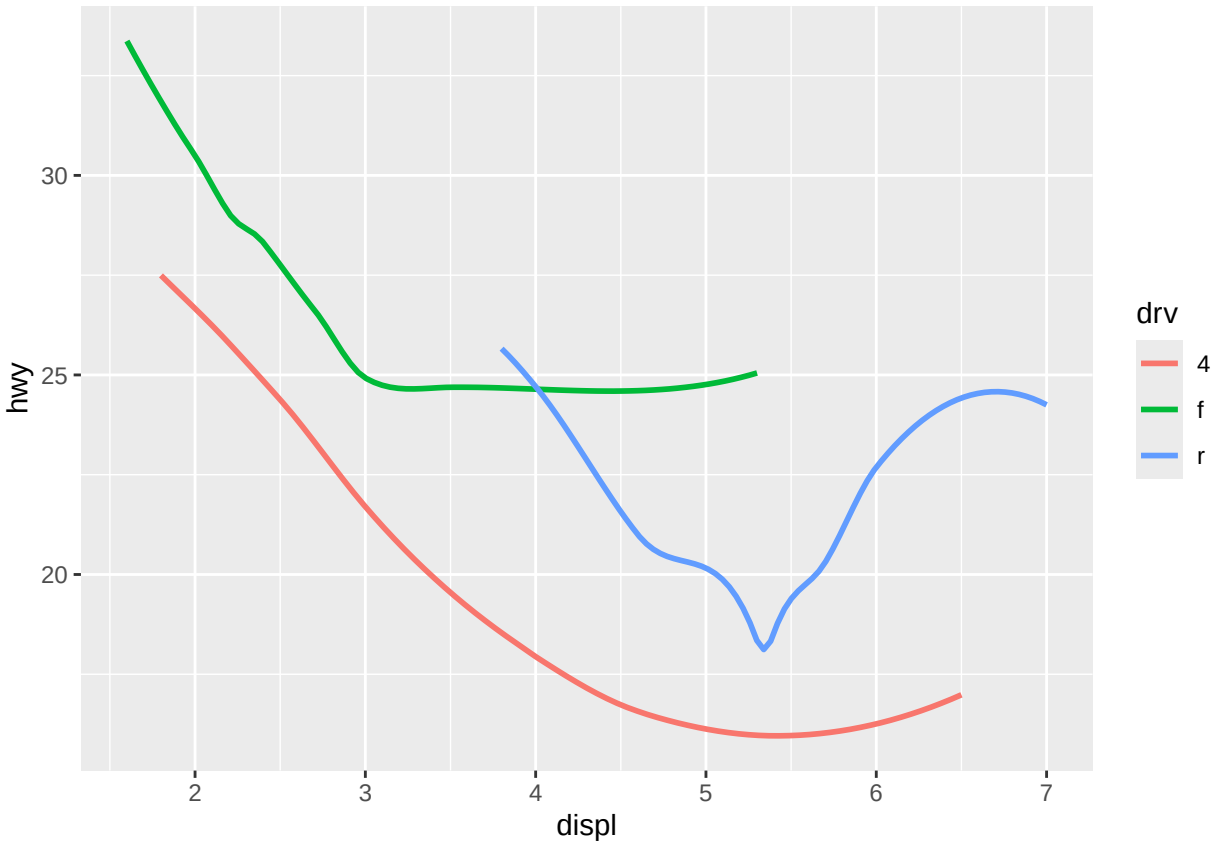
```
### Plotting non-parametric trend line
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_smooth(se = FALSE)
```

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



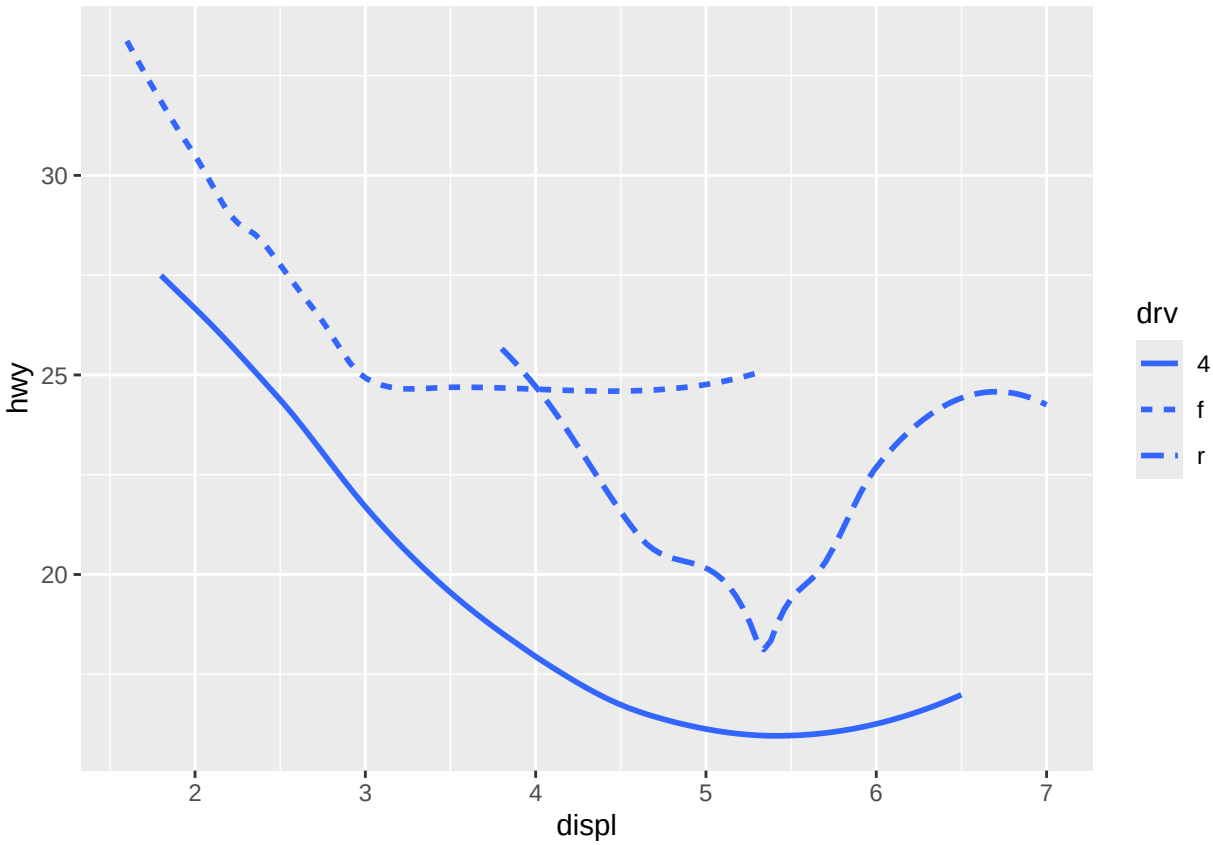

```
### Coloring different lines for different groups
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
geom_smooth(se = FALSE)
```

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



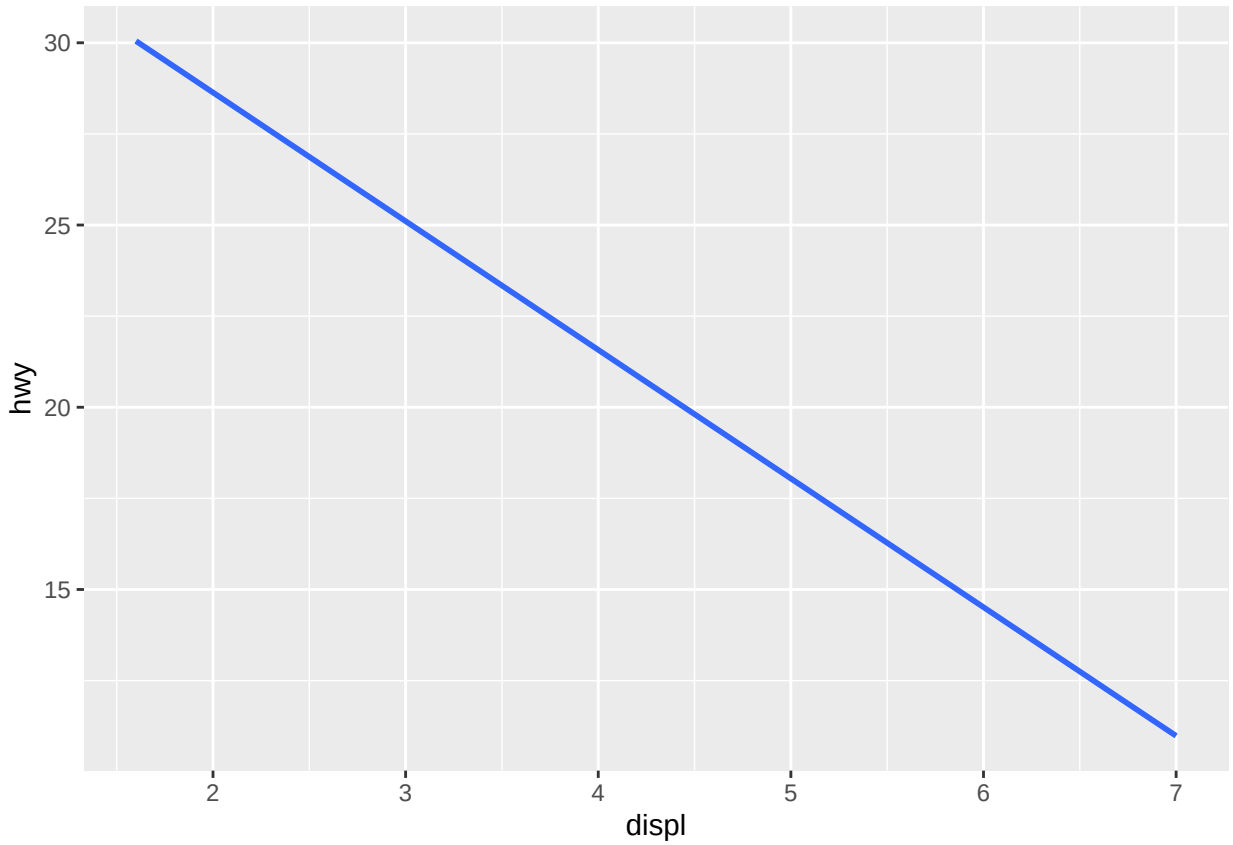

```
### Different line types for different groups
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, linetype = drv)) +
geom_smooth(se = FALSE)
```

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



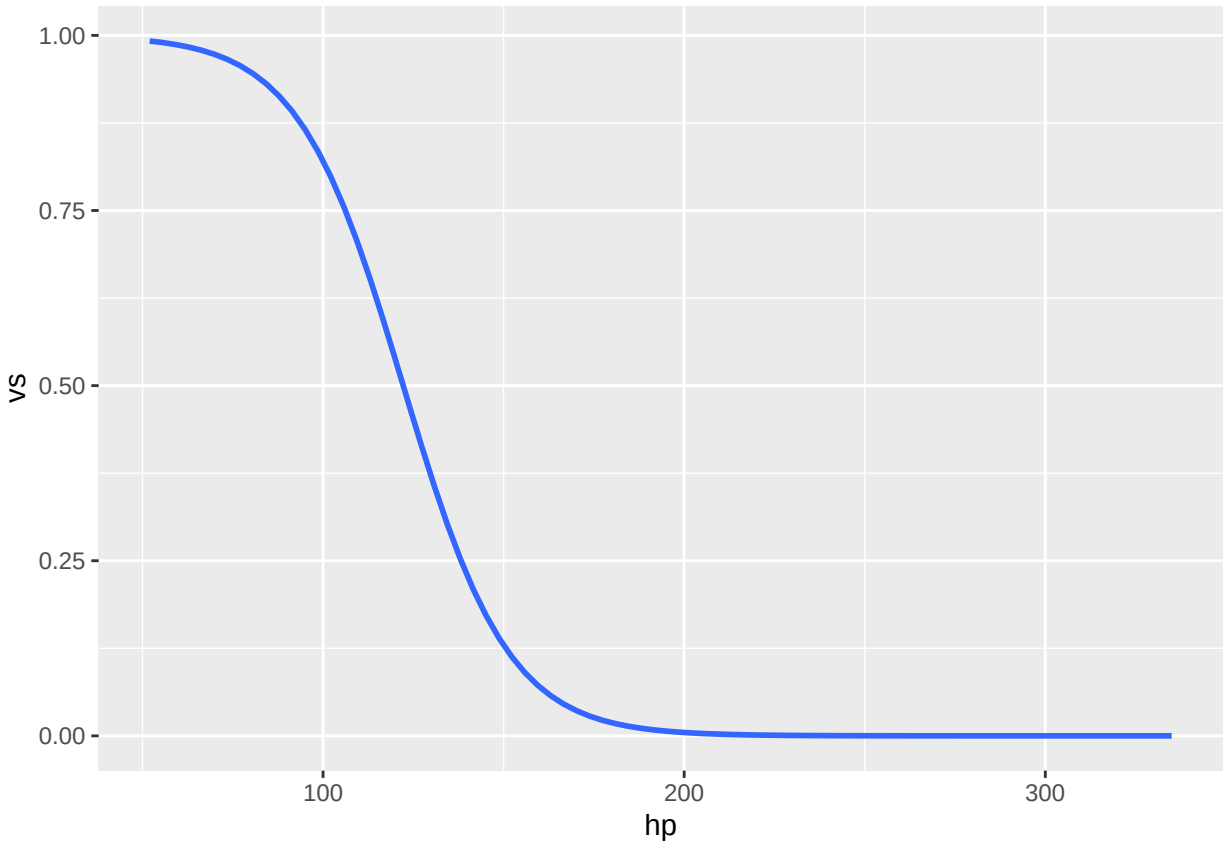

```
### Default loess, plotting OLS ft by specifying method=lm
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_smooth(se = FALSE, method = lm)
```

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




```
### Plotting logistic curves for binary data
ggplot(data = mtcars, mapping = aes(x = hp, y = vs)) +
geom_smooth(method = glm, method.args = list(family = "binomial"), se = FALSE)
```

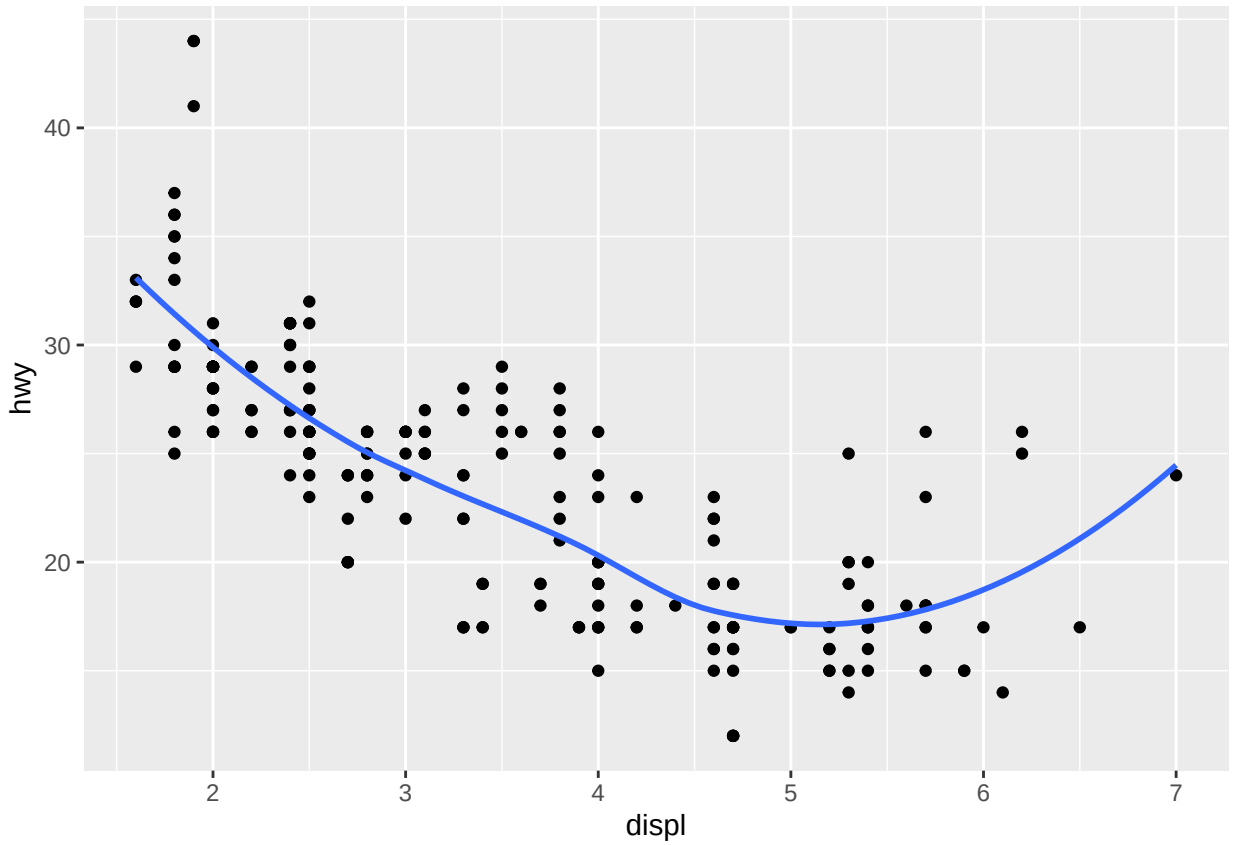
```
## `geom_smooth()` using formula = 'y ~ x'
```



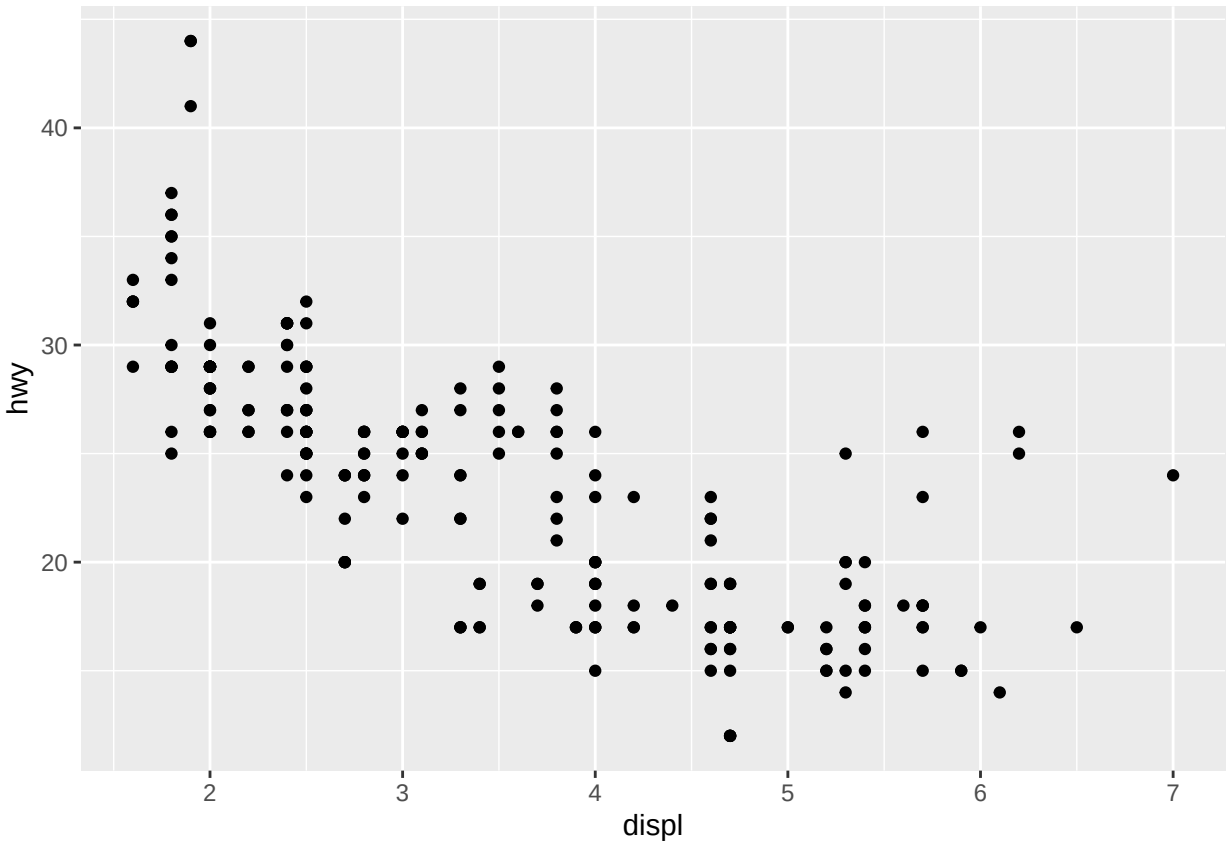

### Layering Geoms

```
### Including both scatterplot and trendline
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth(se = FALSE)
```

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

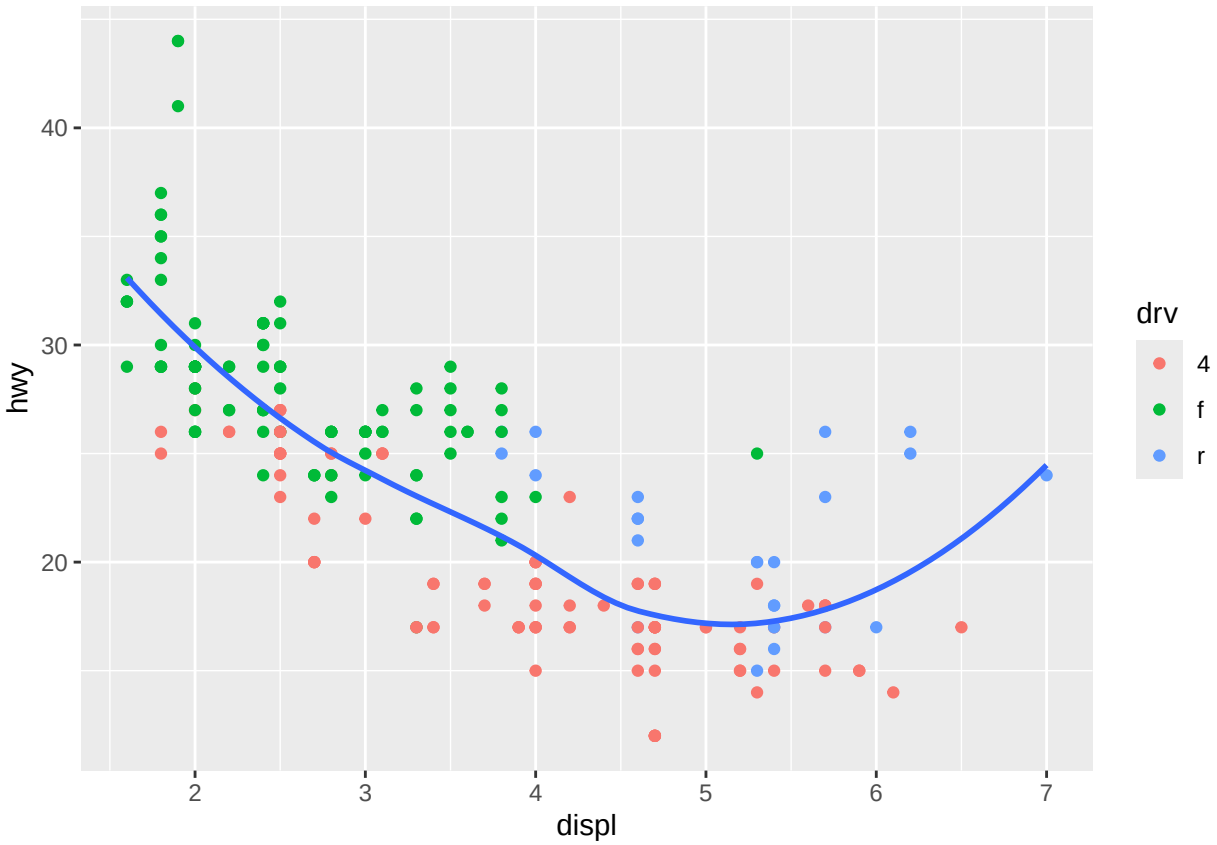



```
### Can specify aesthetic within particular geom
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```




```
## Should produce an error, if aesthetic not defined in ggplot, need to respecify for each geometric ob
# ggplot(data = mpg) +
#   geom_point(mapping = aes(x = displ, y = hwy)) +
#   geom_smooth()

### Specify aesthetics needed for all geom objects in ggplot (x,y), place object specific aesthetics wi
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = drv)) +
  geom_smooth(se = FALSE)
```

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

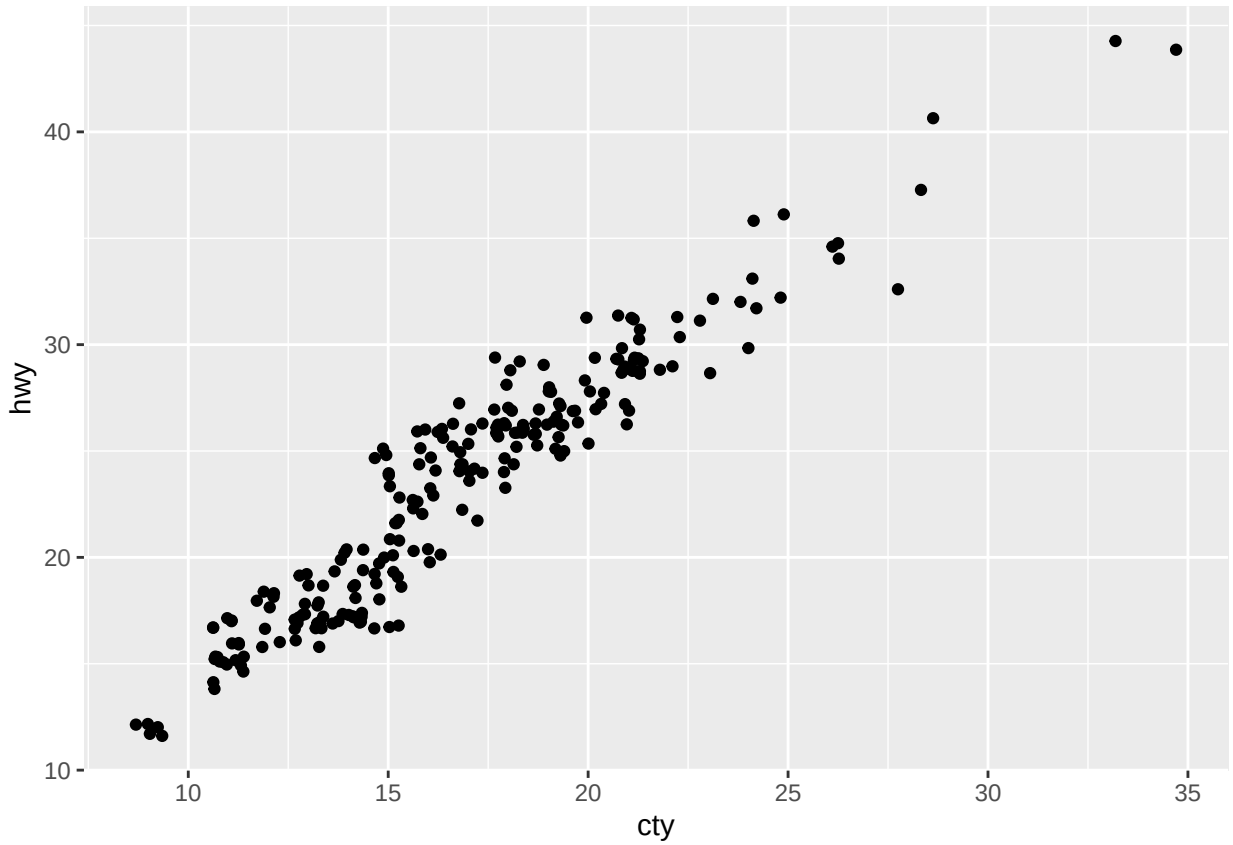



- Exercise:

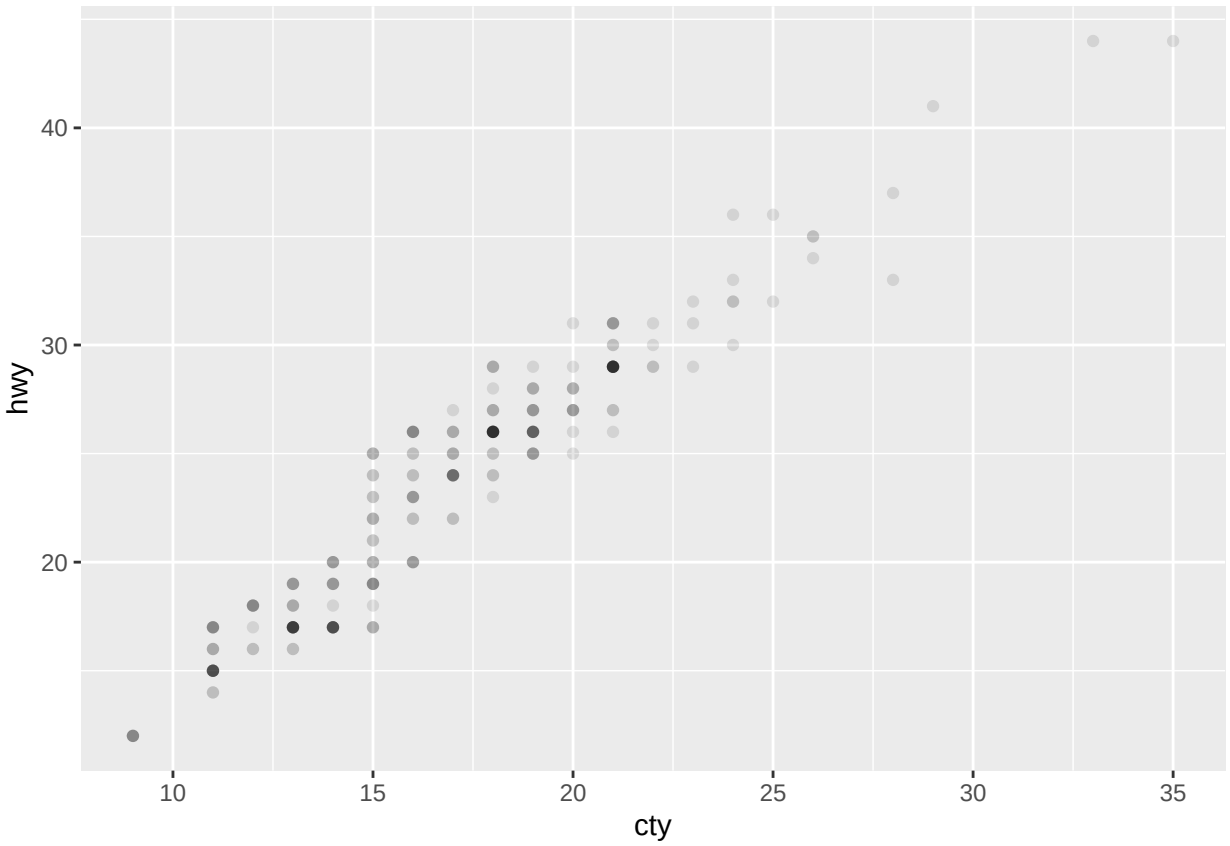
From the penguins dataset, try adding the OLS line to the scatterplot of bill length vs bill depth. Color code by species, but the OLS line should not depend on species.

### Overplotting

```
## Accounting for overlapping in points by slightly nudging them around, best for small data
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
geom_jitter()
```




```
##  Accounting for overlapping by setting all points to have low transparency, best for large data
ggplot(data = mpg, mapping = aes(x = cty, y = hwy)) +
geom_point(alpha = 0.1)
```




- Exercise:

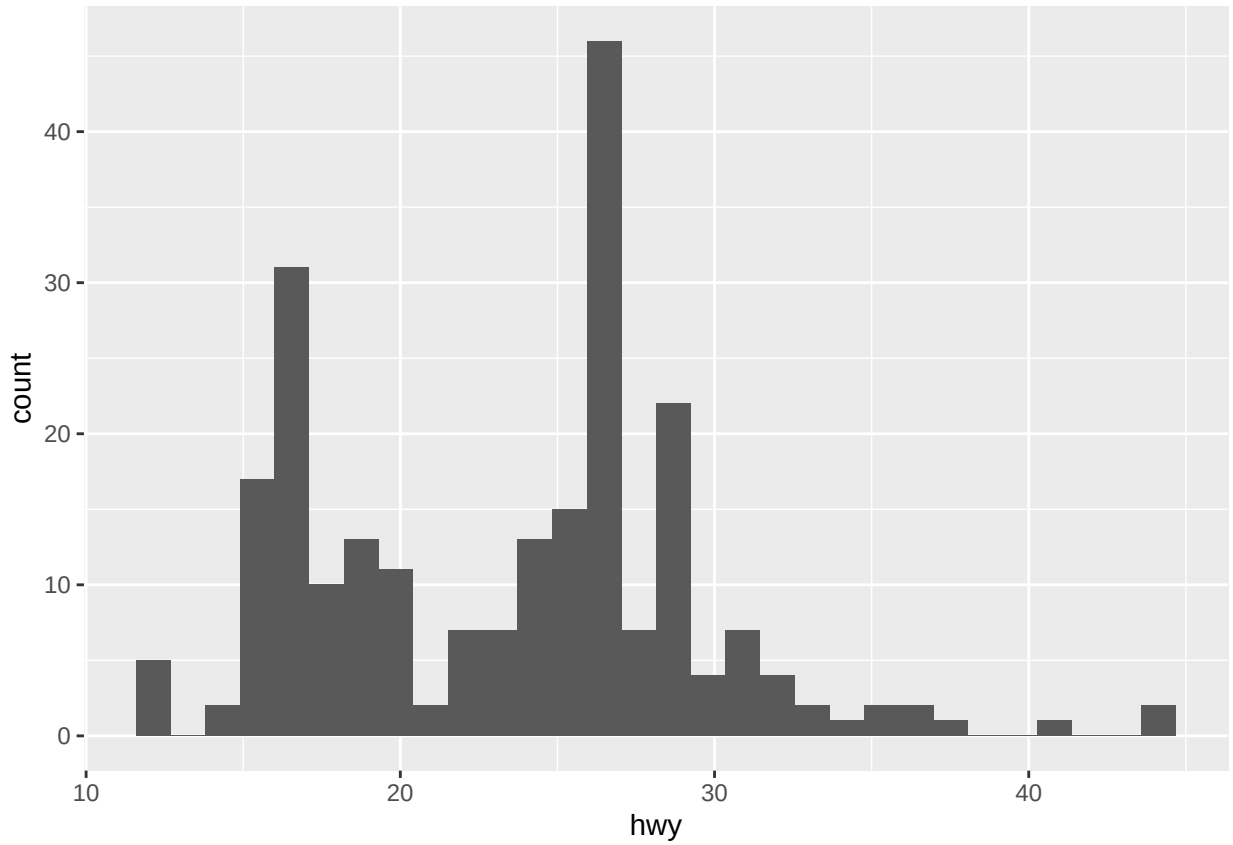
Consider the diamonds dataset from ggplot2. Load it and then plot carat vs price. Account for overplotting if possible

```
data("diamonds")
```

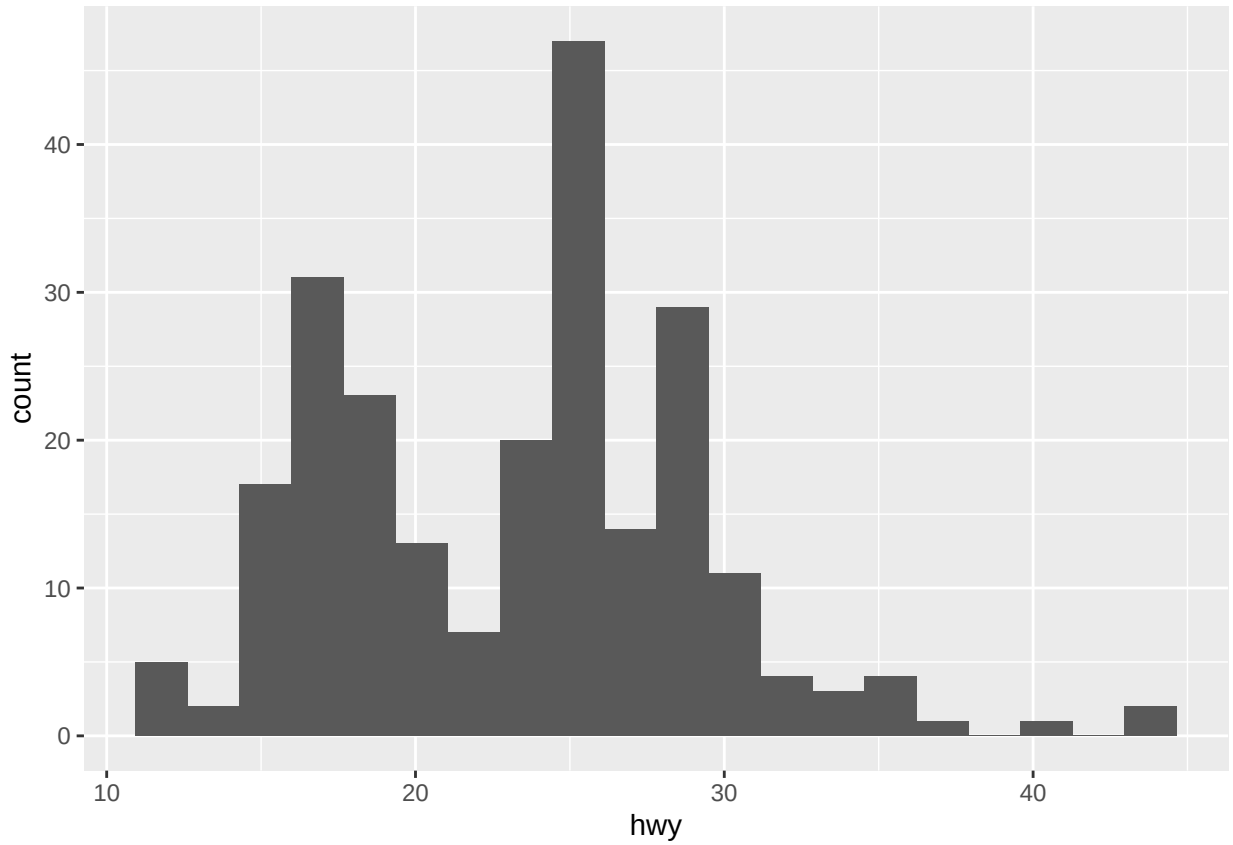
##One Quantitative Variable

```
# Histogram of hwy
ggplot(data = mpg, mapping = aes(x = hwy)) +
geom_histogram()
```

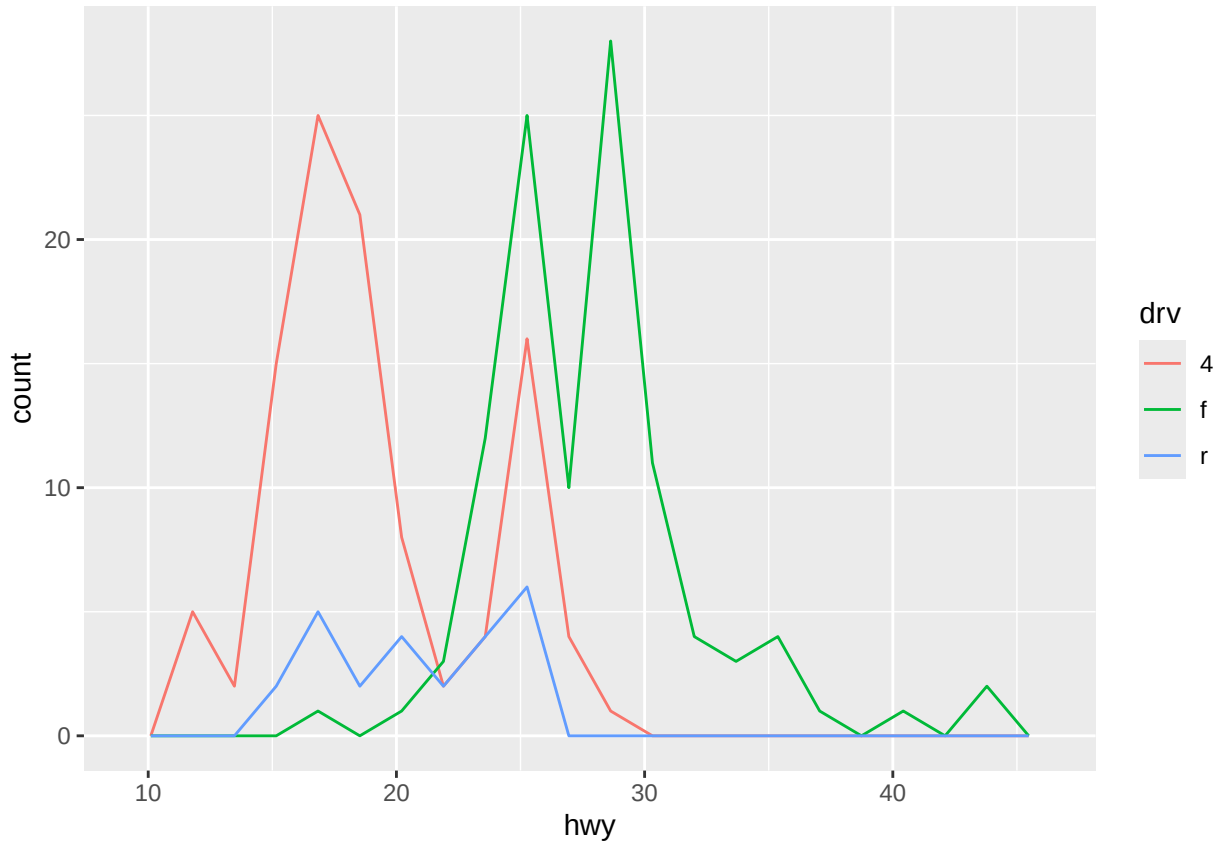
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Specifying number of bins for histogram
ggplot(data = mpg, mapping = aes(x = hwy)) +
geom_histogram(bins = 20)
```

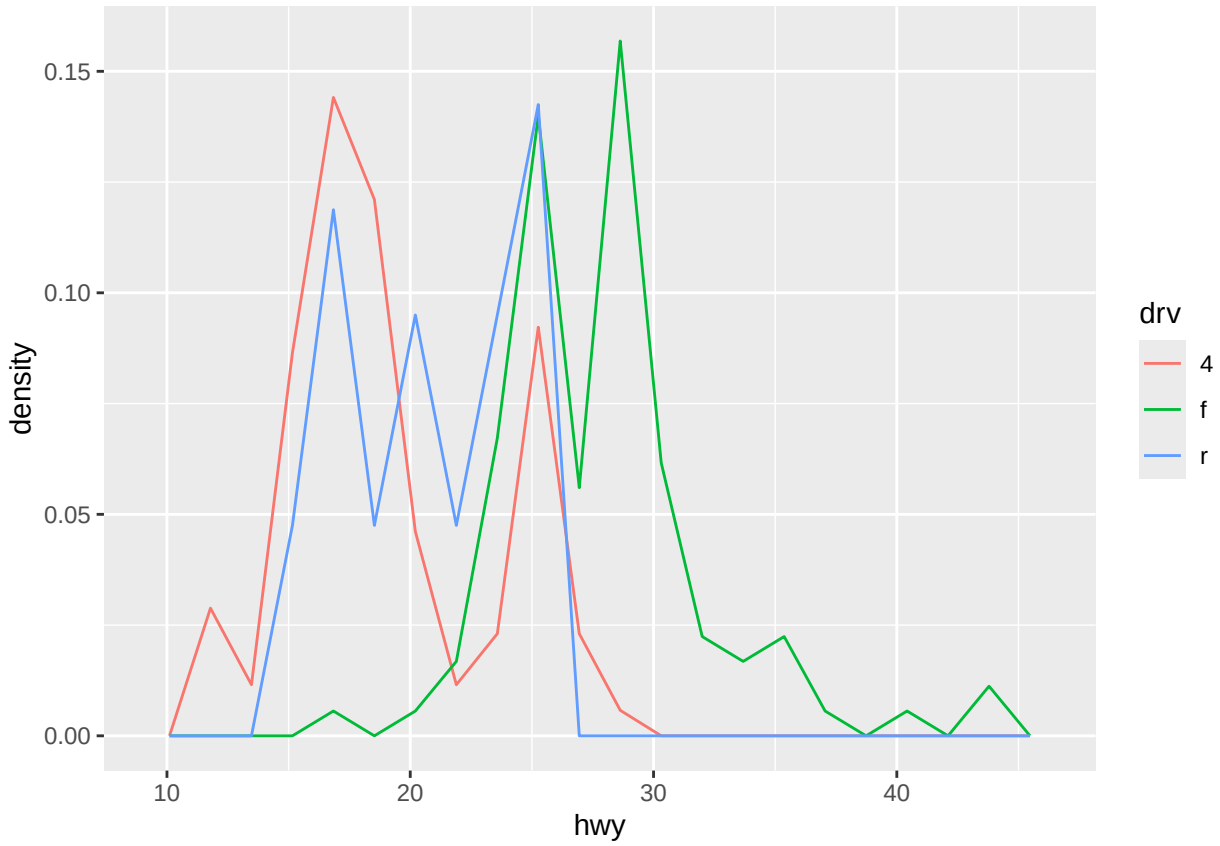



```
#fregpoly- like histogram except lines specify counts
# color=drv specifies different frequency polygons for different groups, differentiated by color
ggplot(data = mpg, mapping = aes(x = hwy, color = drv)) +
geom_freqpoly(bins = 20)
```




```
## y= ..density .., plotting density instead of counts on y-axis
ggplot(data = mpg, mapping = aes(x = hwy, y = ..density.., color = drv)) +
geom_freqpoly(bins = 20)
```

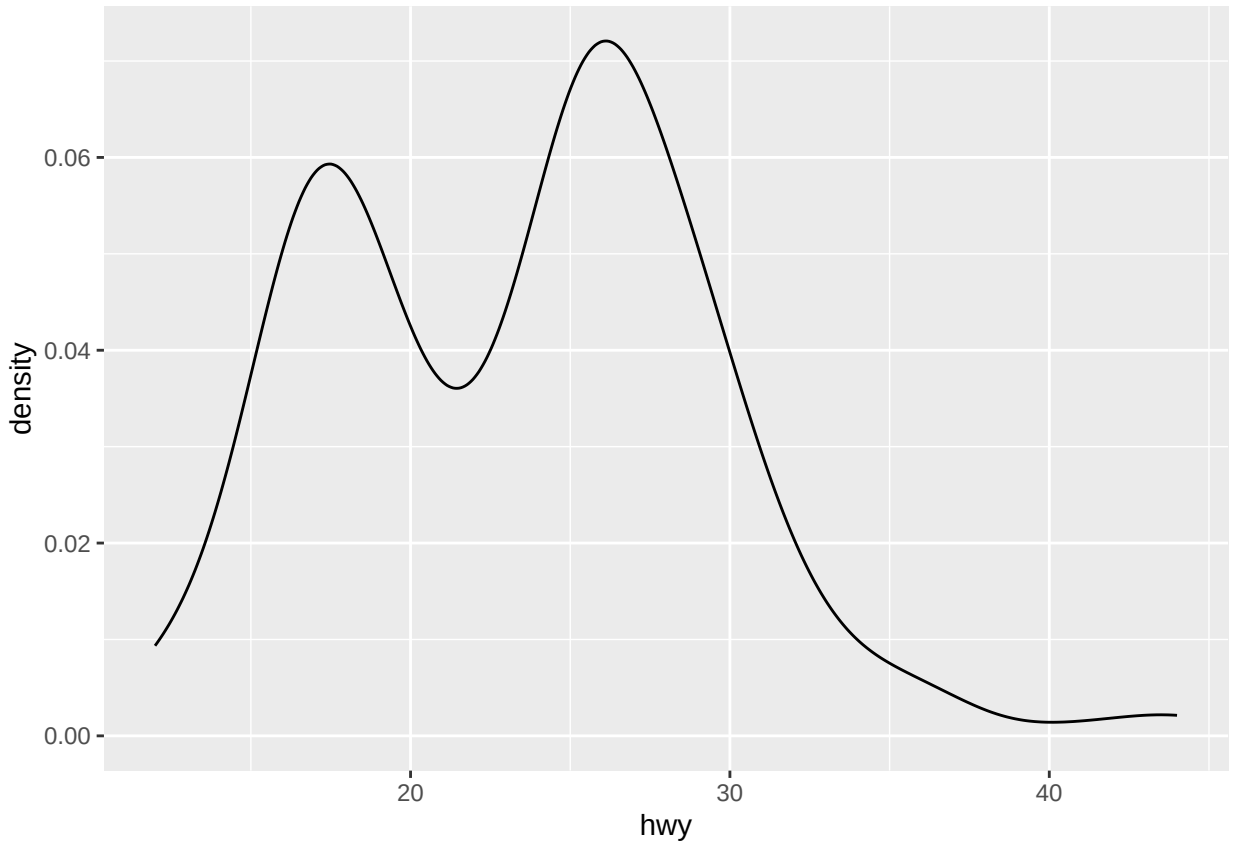
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



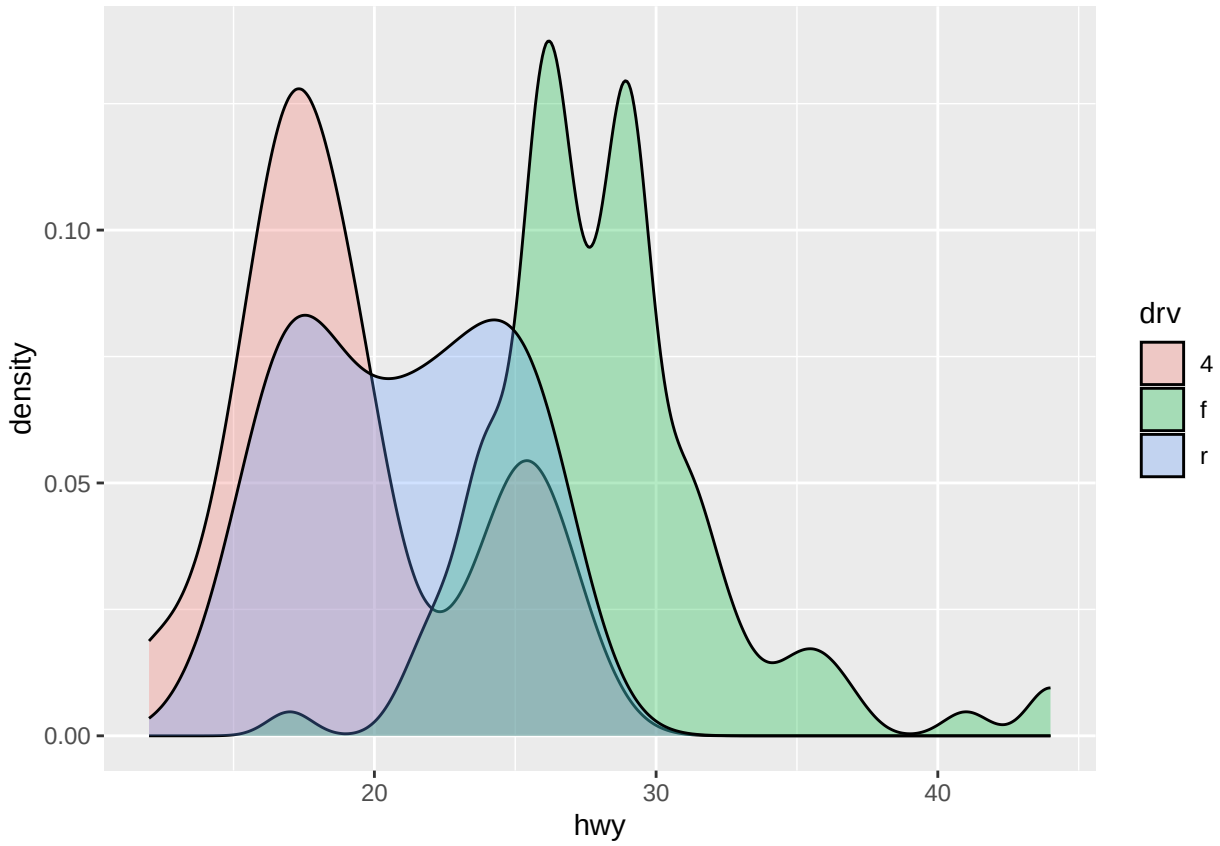

Exercise:
Create a frequency polygon of the price of a diamond, color coding by cut.

**Density Plot**

```
###Plotting density (smoothed histogram)
ggplot(data = mpg, mapping = aes(x = hwy)) +
geom_density()
```




```
### fill=drv, different densities for different values of drv, each filkled with different color
### alpha=0.3, specifying transparency of fill
ggplot(data = mpg, mapping = aes(x = hwy, fill = drv)) +
geom_density(alpha = 0.3)
```

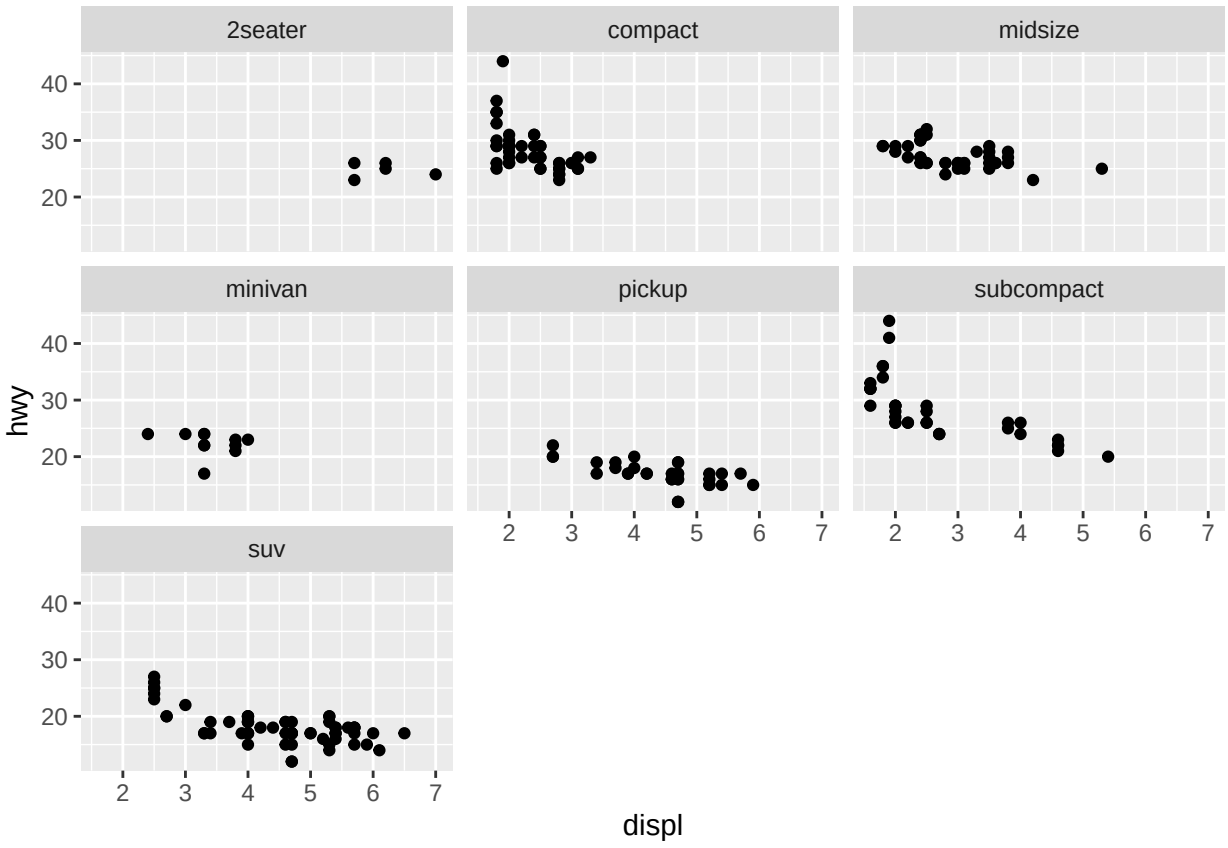



### One categorical variable, one quantitative variable

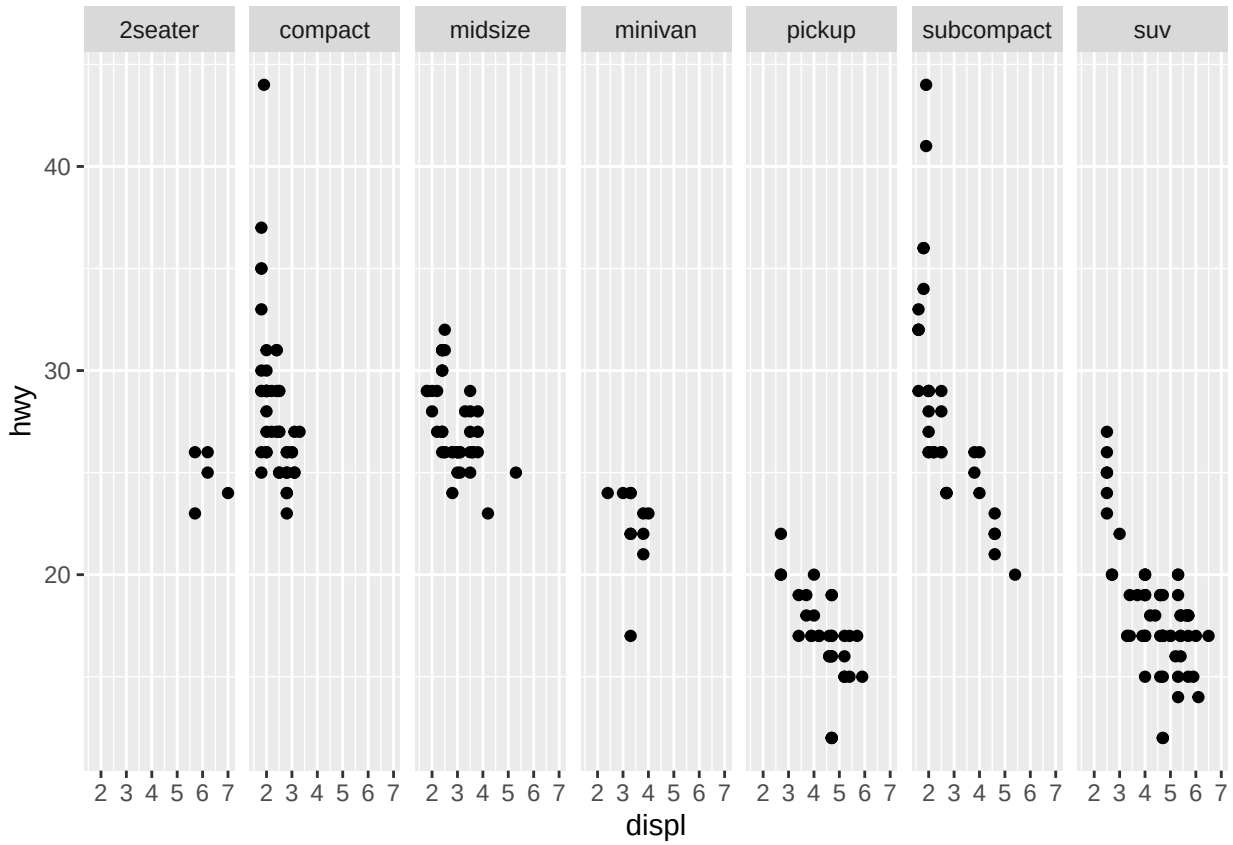
Exercise:
Create a boxplot for cut vs log-price. Does anything surprise you? Can you think of another plot that could explain your surprise?

### Facets

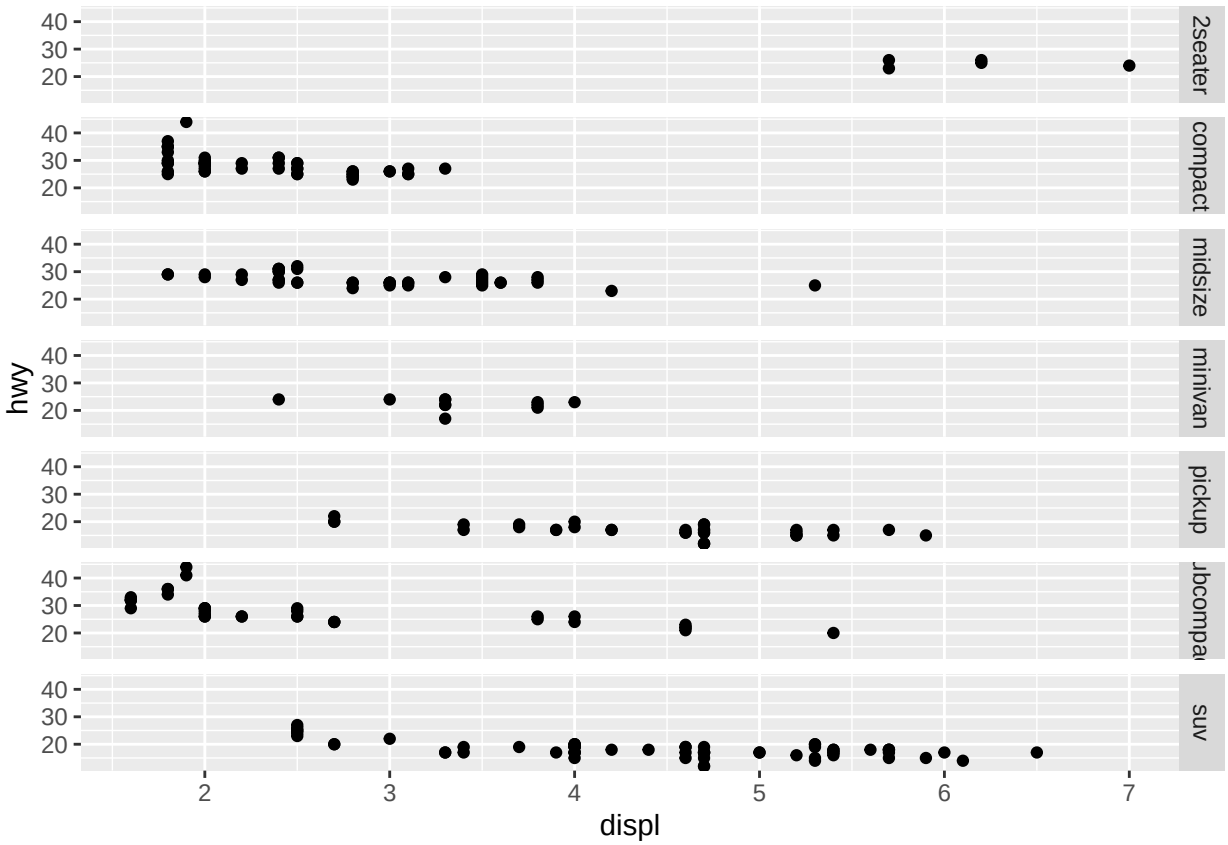
```
## Creates table of scatterplots, one for each level of class
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
facet_wrap( ~ class)
```



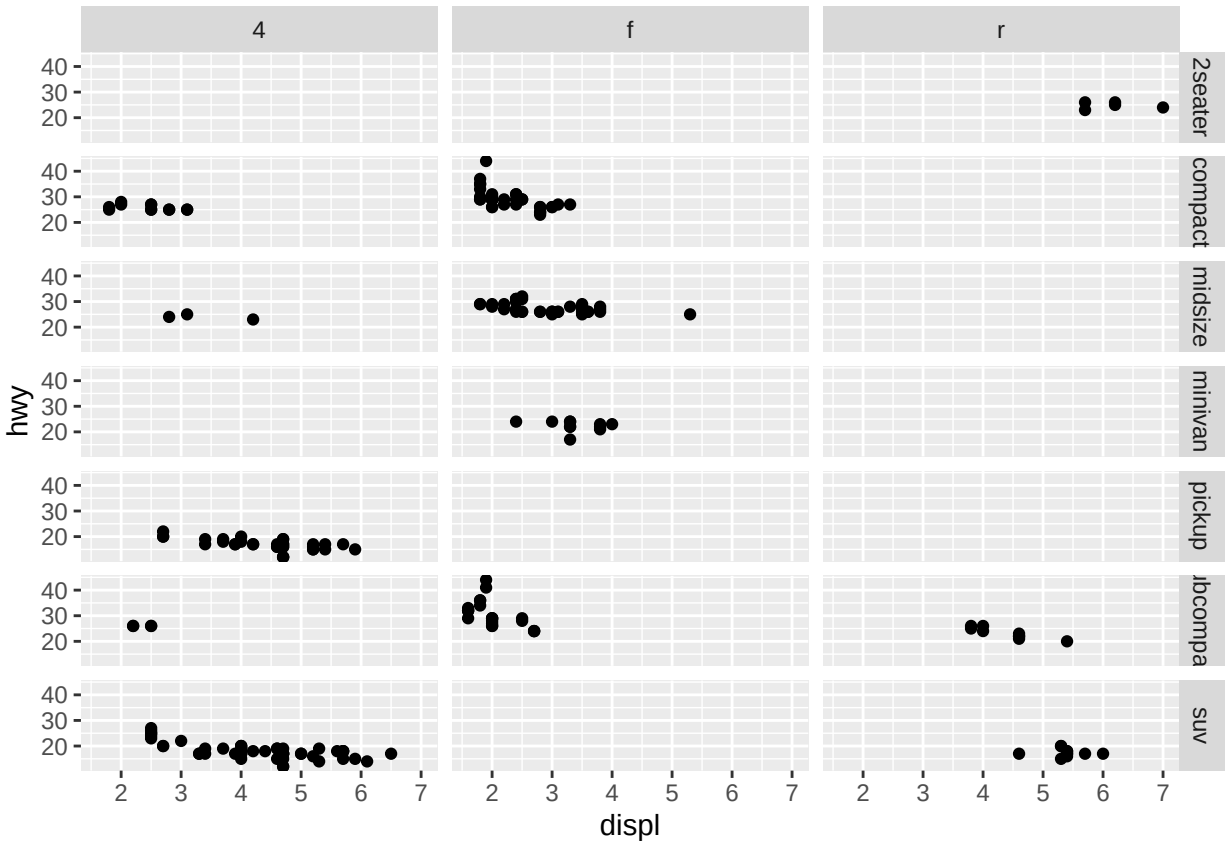

```
## Creates table of scatterplots, one for each level of class, organized in one row
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
facet_grid(. ~ class)
```




```
## Creates table of scatterplots, one for each level of class, organized in one column
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
facet_grid(class ~ .)
```



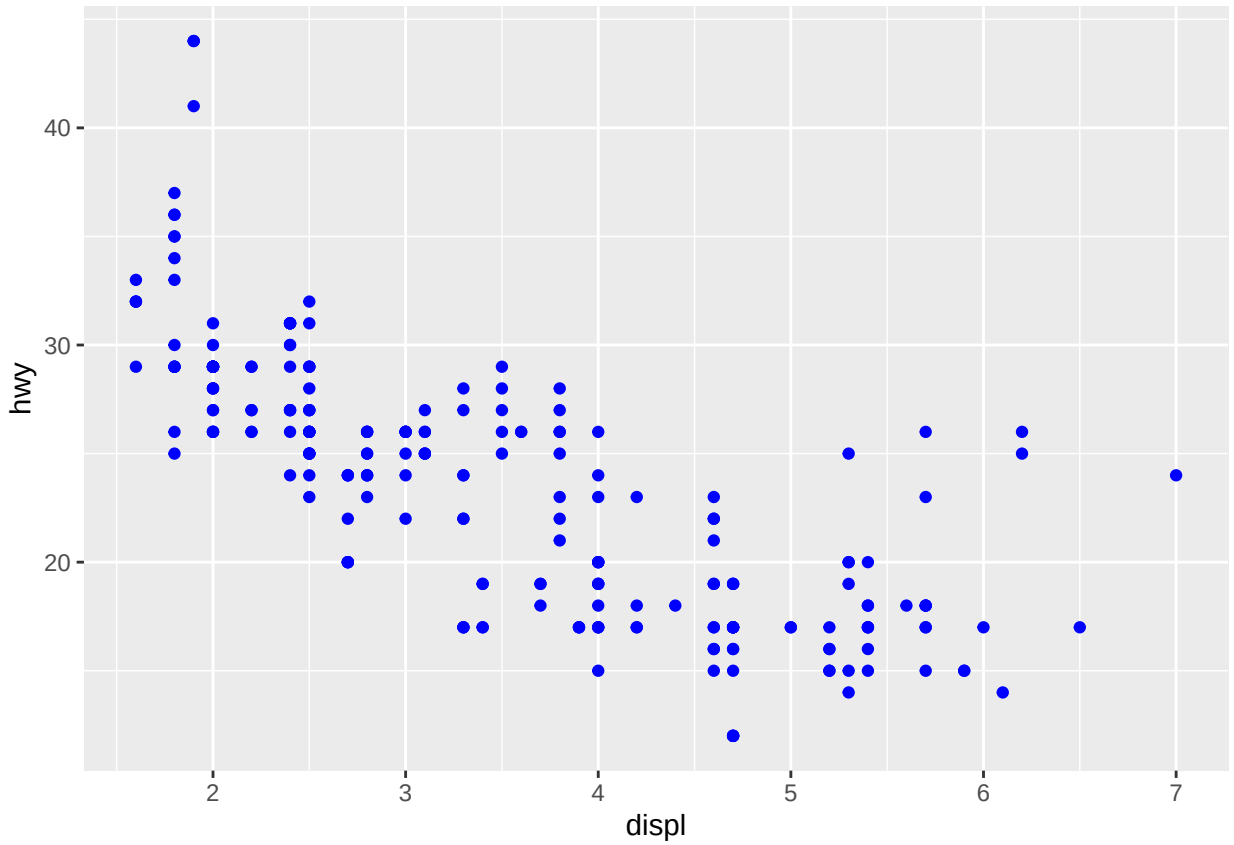

```
## Creates table of scatterplots, one for each combination of class and drv, each row= class, column=dr
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point() +
facet_grid(class ~ drv)
```



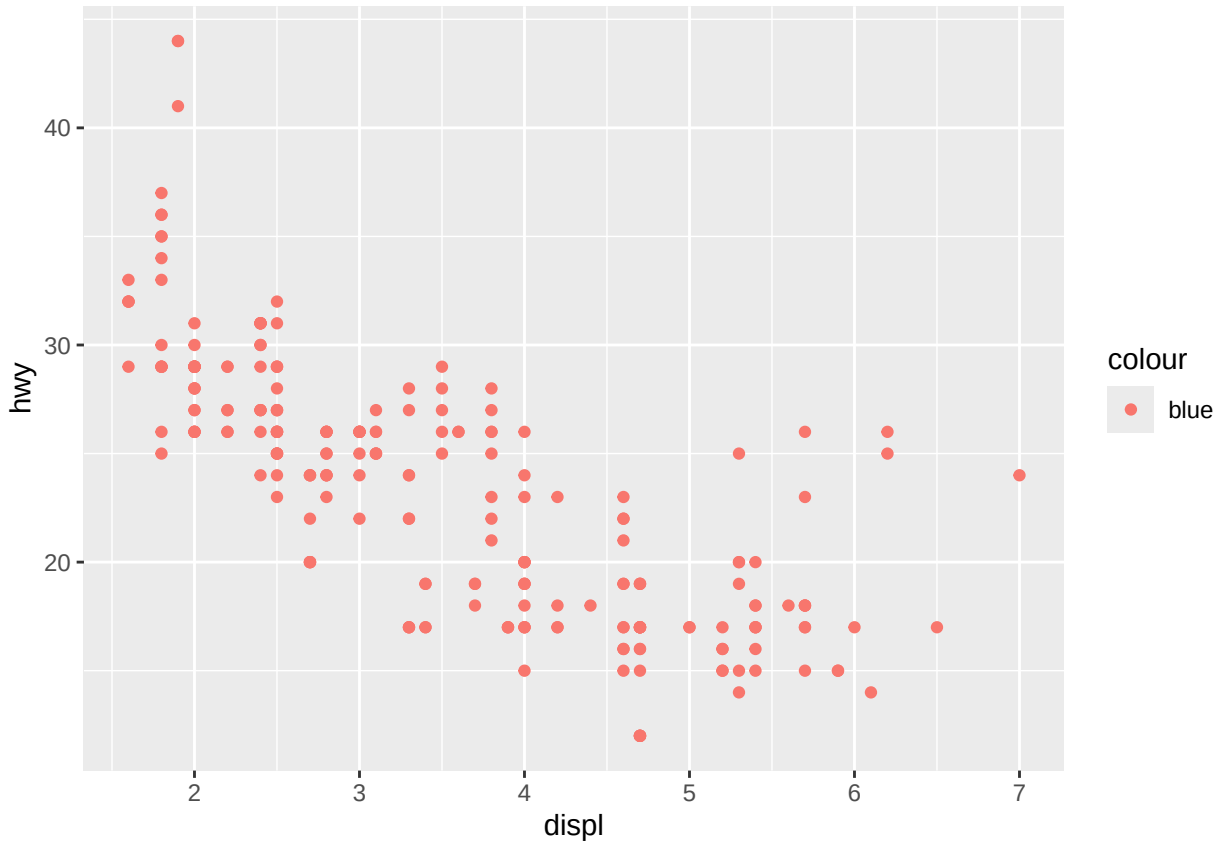

Exercise:
Create a boxplot of cut vs price for each level of color.

## Aesthetics for all objects

```
## If you want to specify that all points have color blue, must specify color outside of aes
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
geom_point(color = "blue")
```

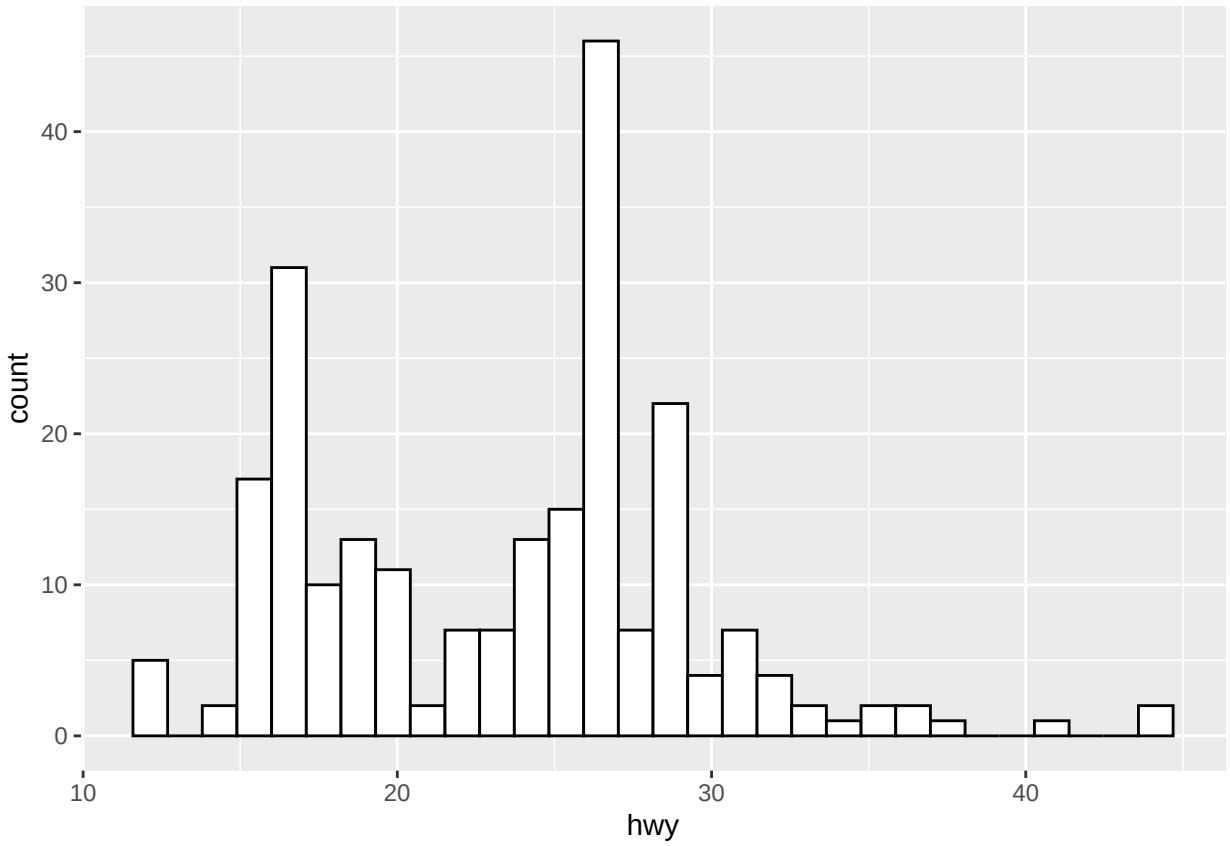



```
#Otherwise R assumes "blue" is categorical variable for which all observations have same value
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = "blue")) +
geom_point()
```



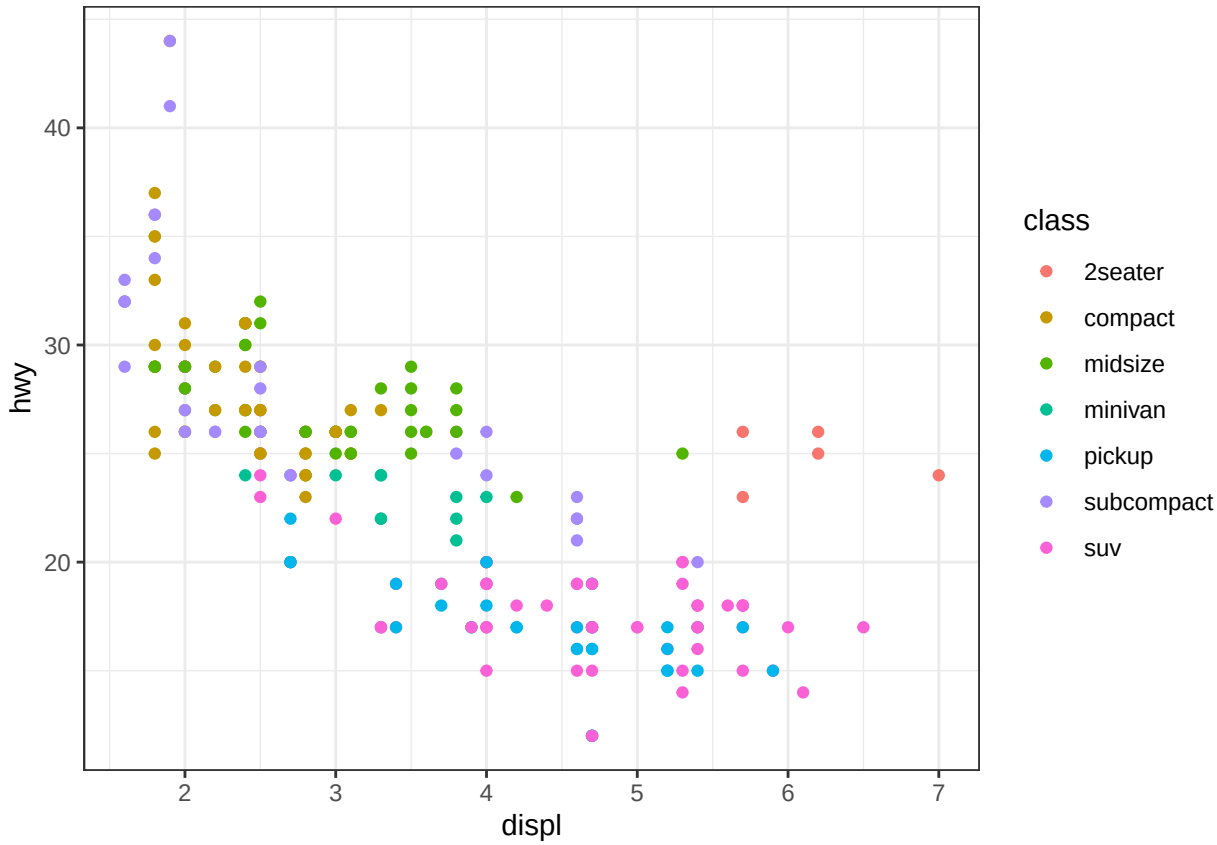

```
# Specifying specific properties for histogram, regardless of variable
ggplot(data = mpg, mapping = aes(x = hwy)) +
geom_histogram(fill = "white", color = "black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

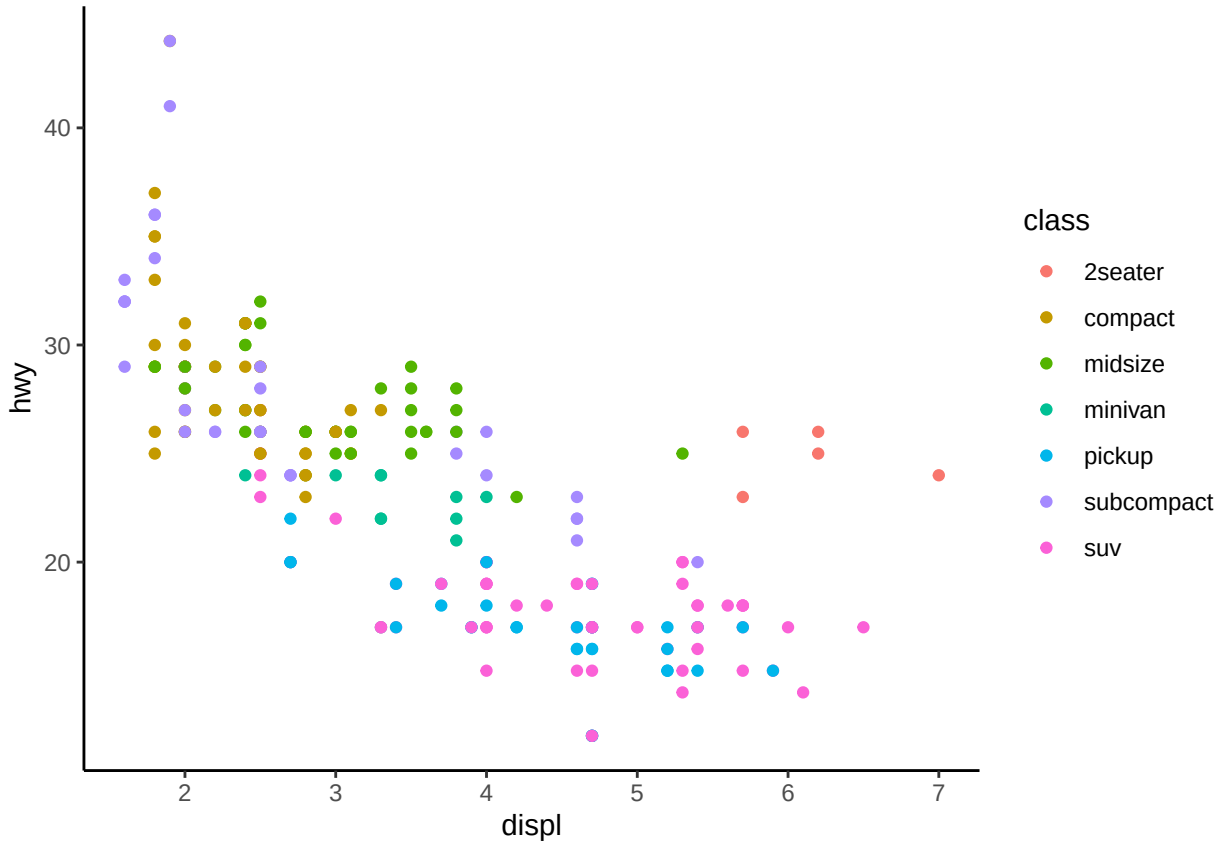



##Themes

```
## Each them specifies a different way to have background
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_bw()
```



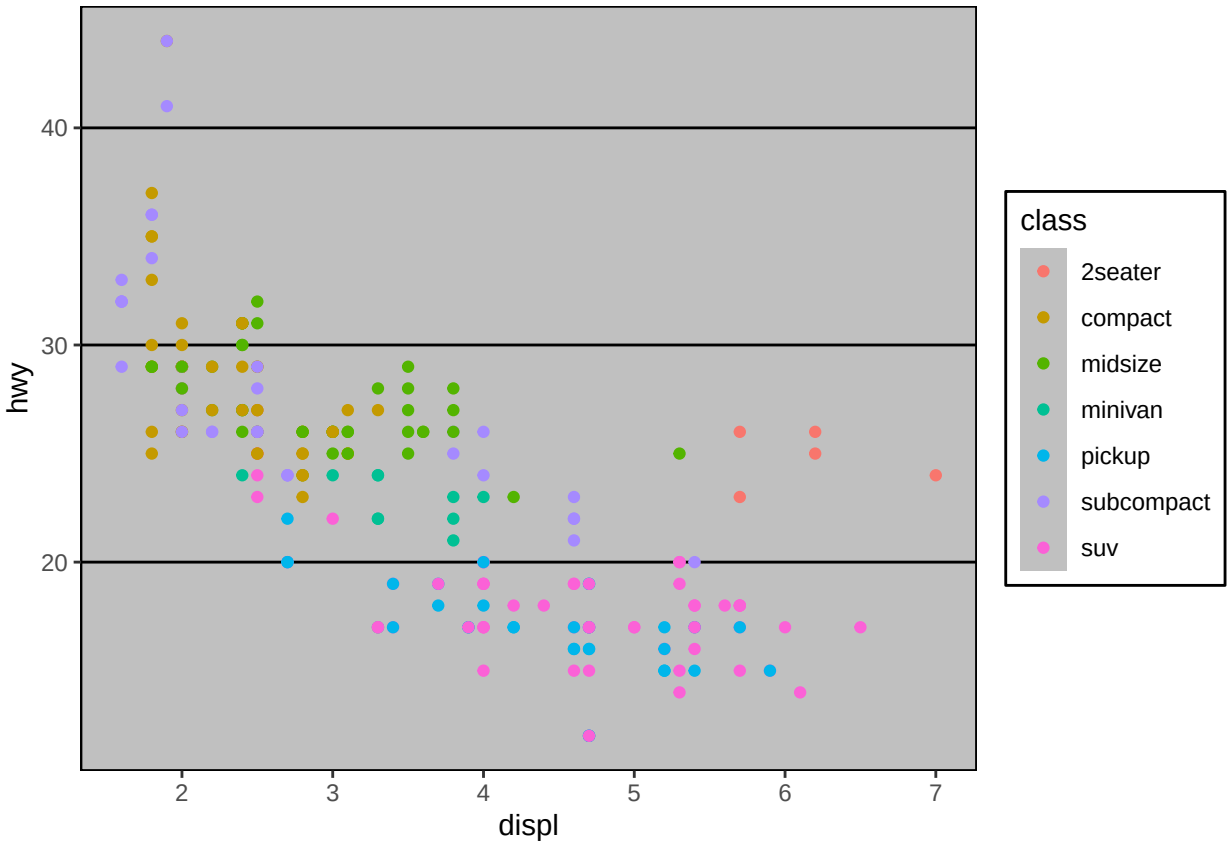

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_classic()
```



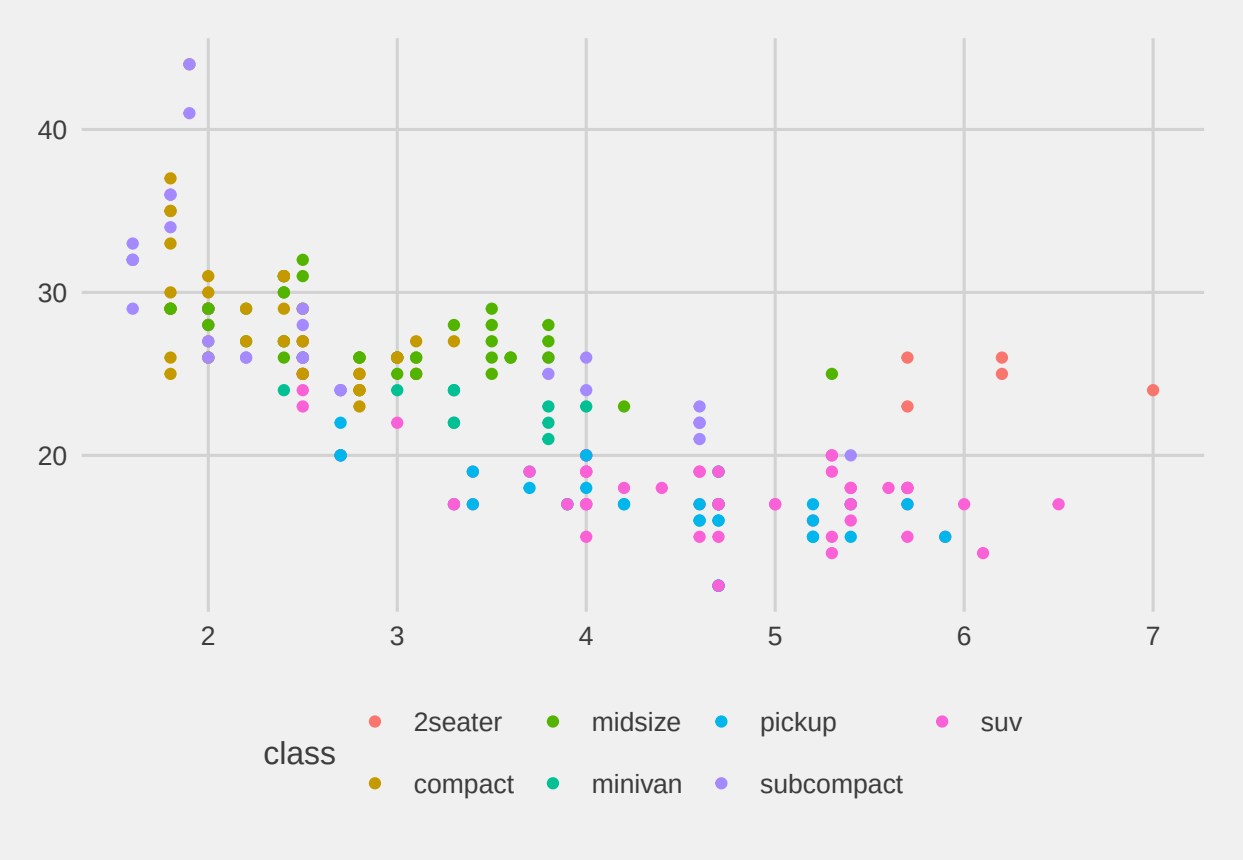

```
install.packages("ggthemes")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(ggthemes)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_excel()
```




```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_fivethirtyeight()
```

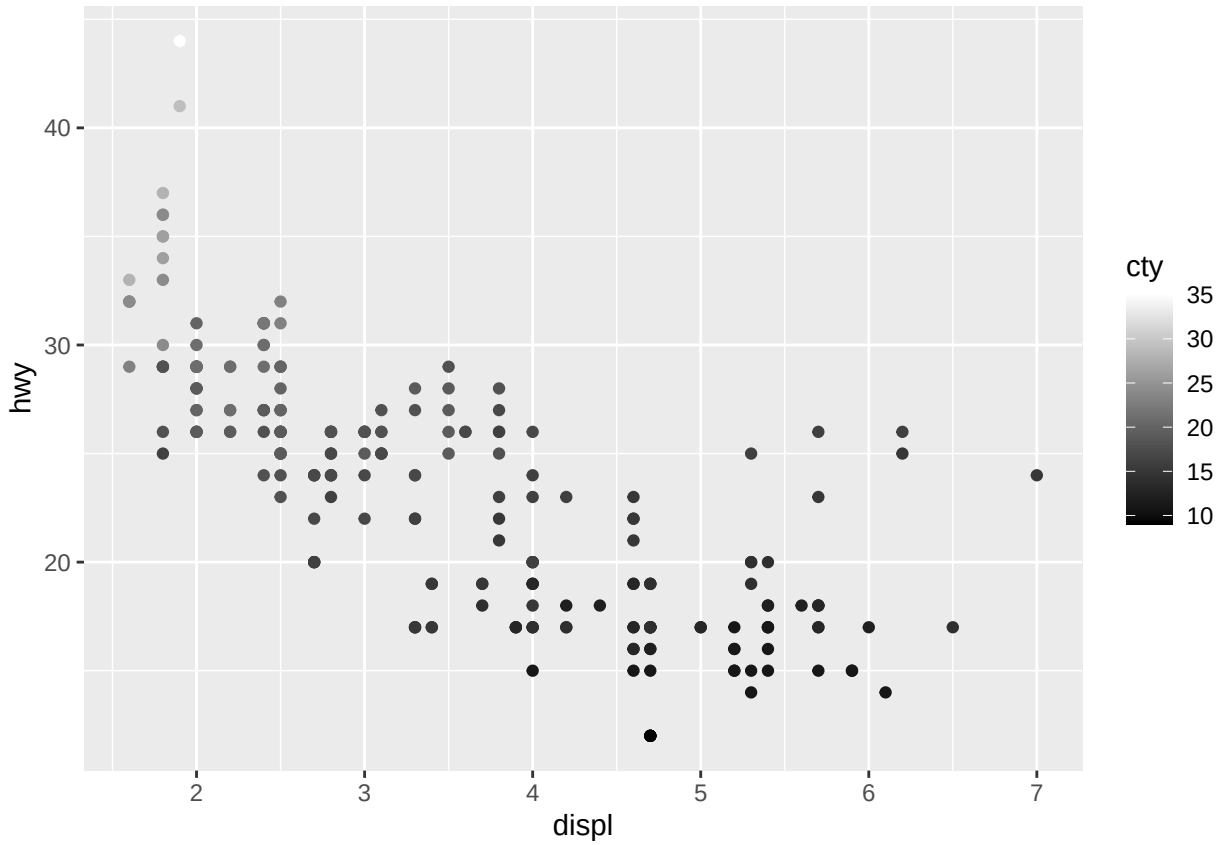



- Exercise:

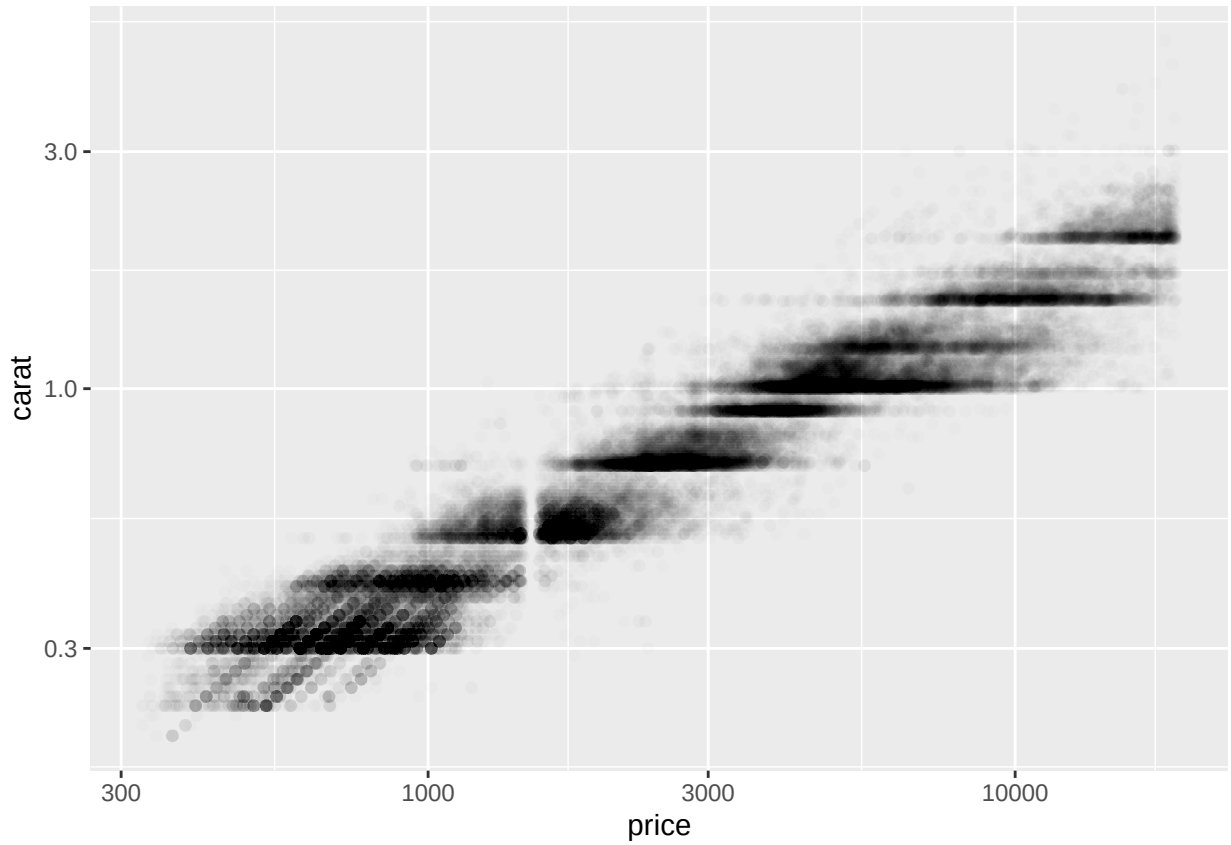
Create a boxplot of cut vs price for each level of color. Make the boxplots all orange and use the black and white theme.

##Scales

```
## scale_color_continuous, low= which color associates to smaller values cty, high=which color associat
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = cty)) +
geom_point() +
scale_color_continuous(low = "black", high = "white")
```

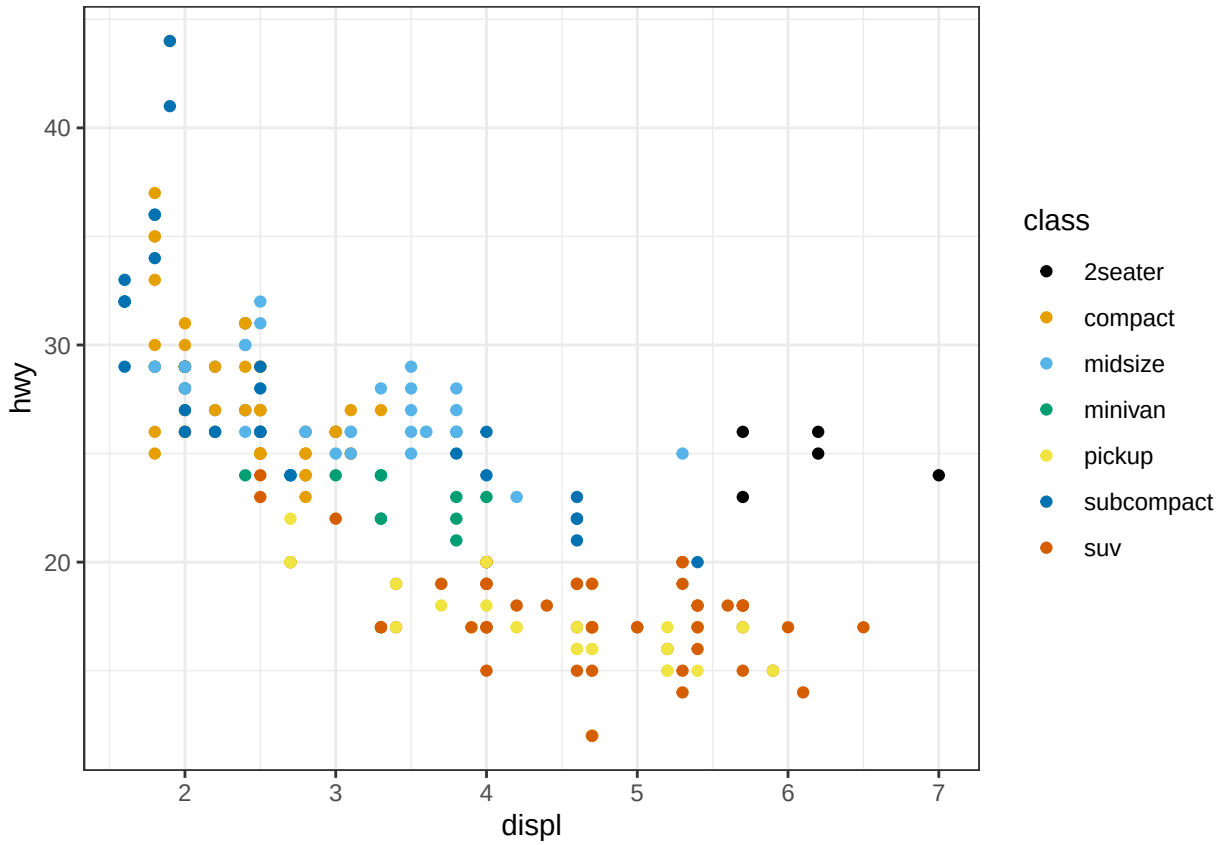



```
##Scaling x and y axis by log10
ggplot(data = diamonds, mapping = aes(x = price, y = carat)) +
scale_y_log10() +
scale_x_log10() +
geom_point(alpha = 0.01)
```

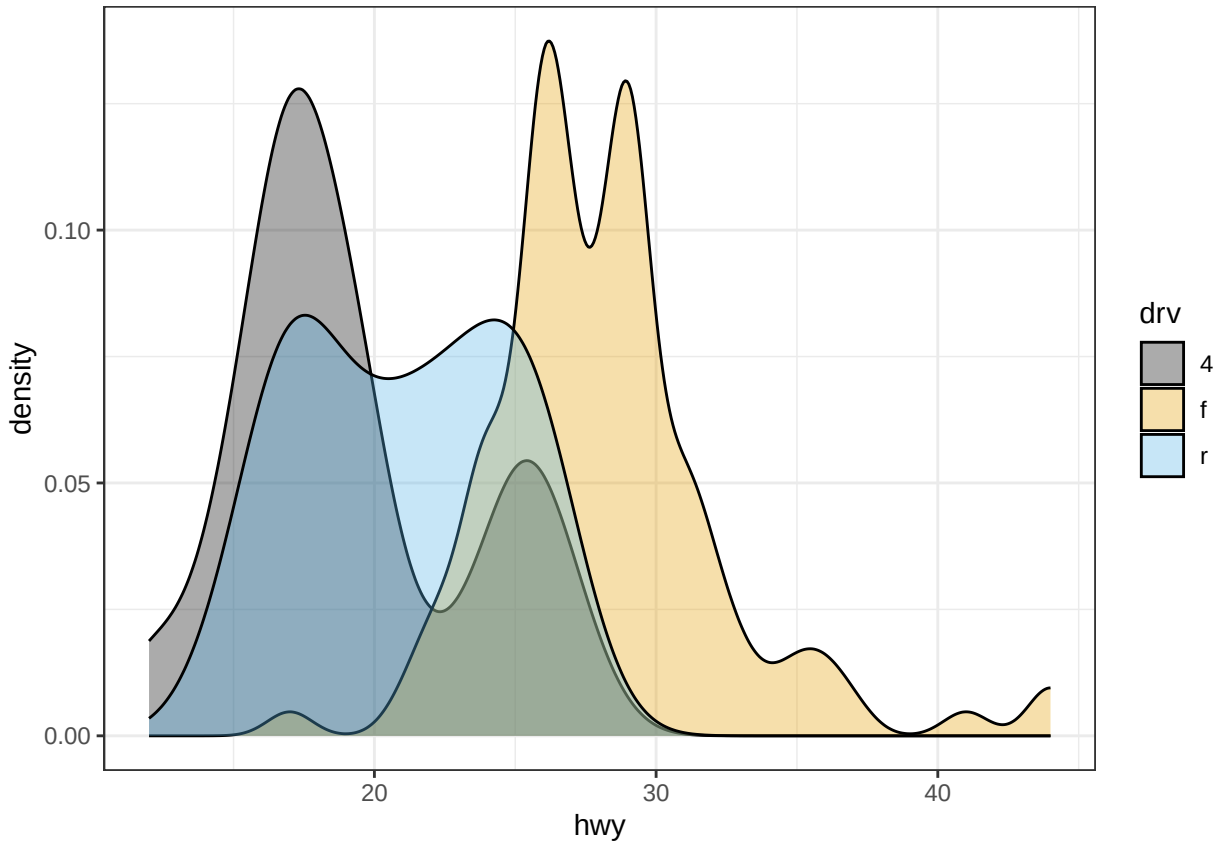



## Colorblind Safe Palettes

```
### scale_color_colorblind- gives points colorblind safe colors
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_bw() +
scale_color_colorblind()
```

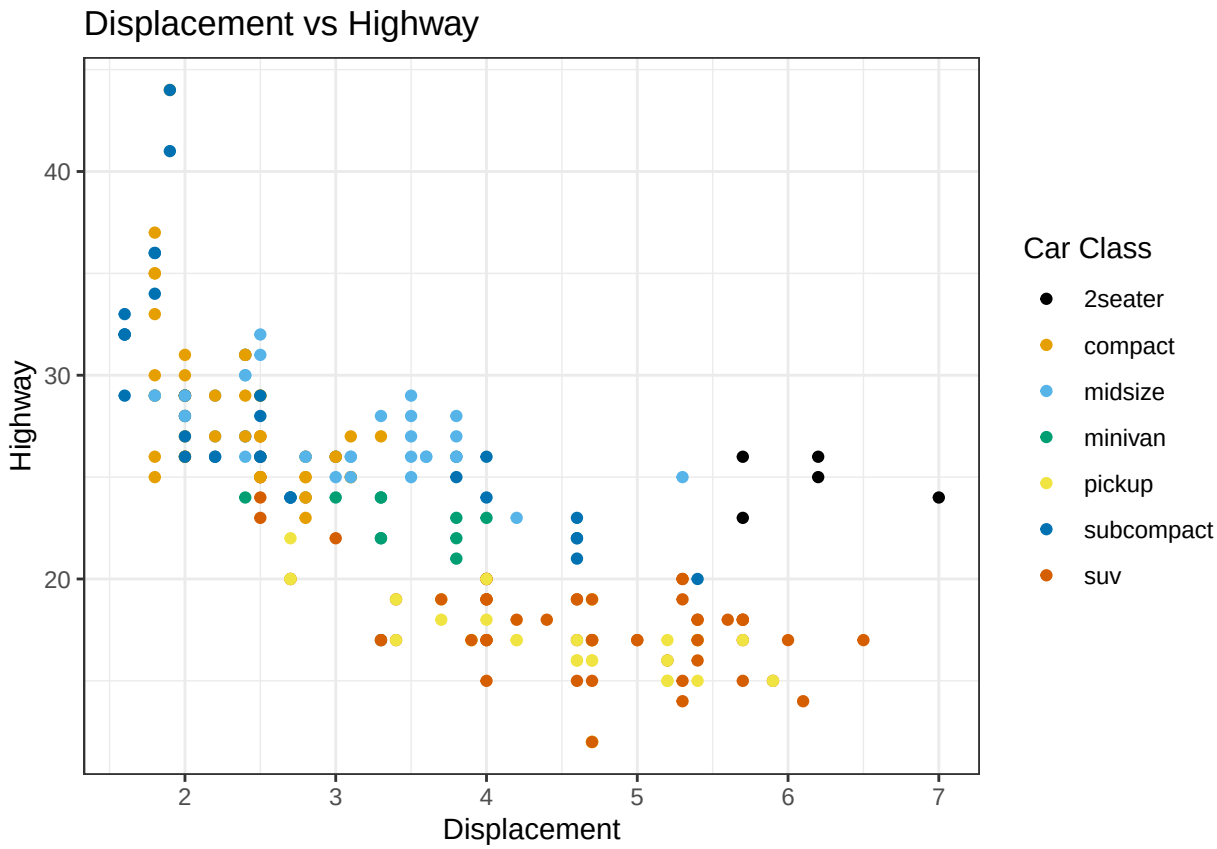



```
### scale_fill_colorblind- fills density with colorblind safe colors
ggplot(data = mpg, mapping = aes(x = hwy, fill = drv)) +
geom_density(alpha = 1/3) +
theme_bw() +
scale_fill_colorblind()
```

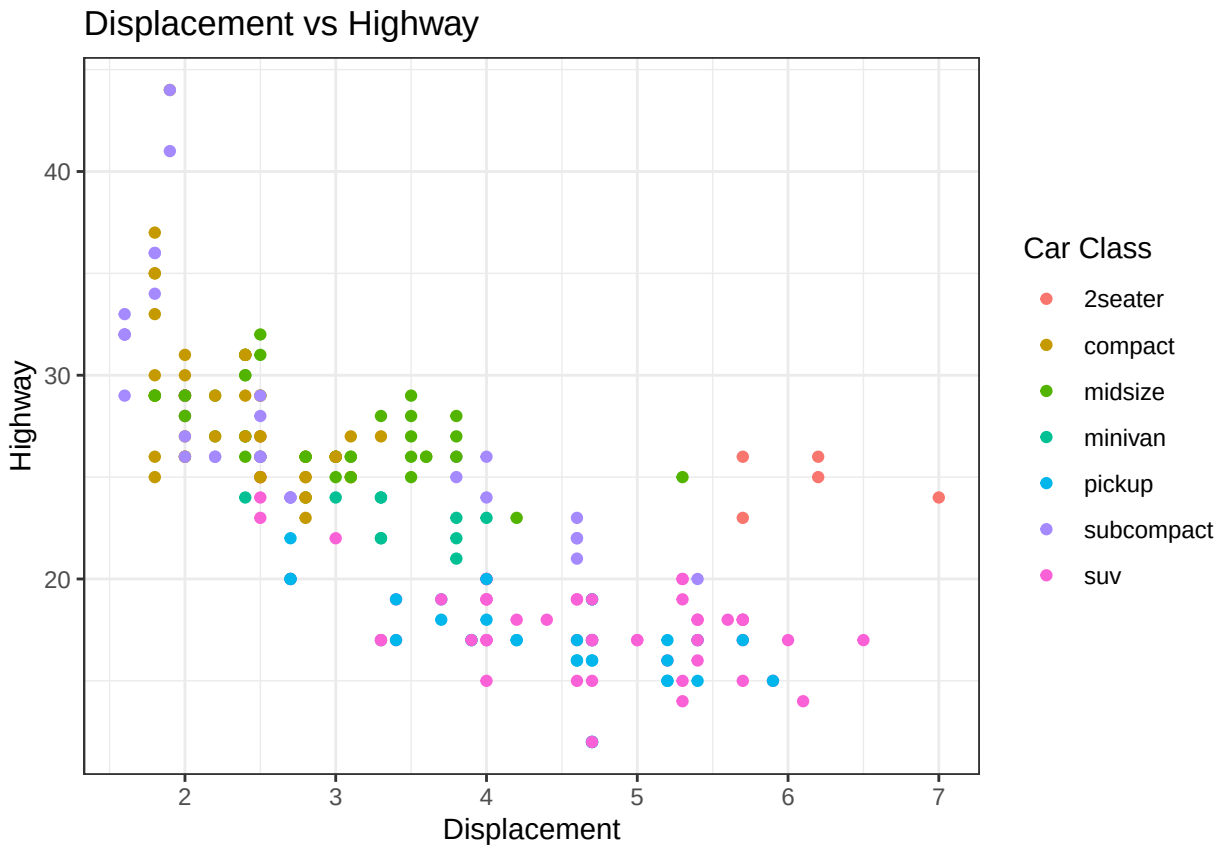



##Labels

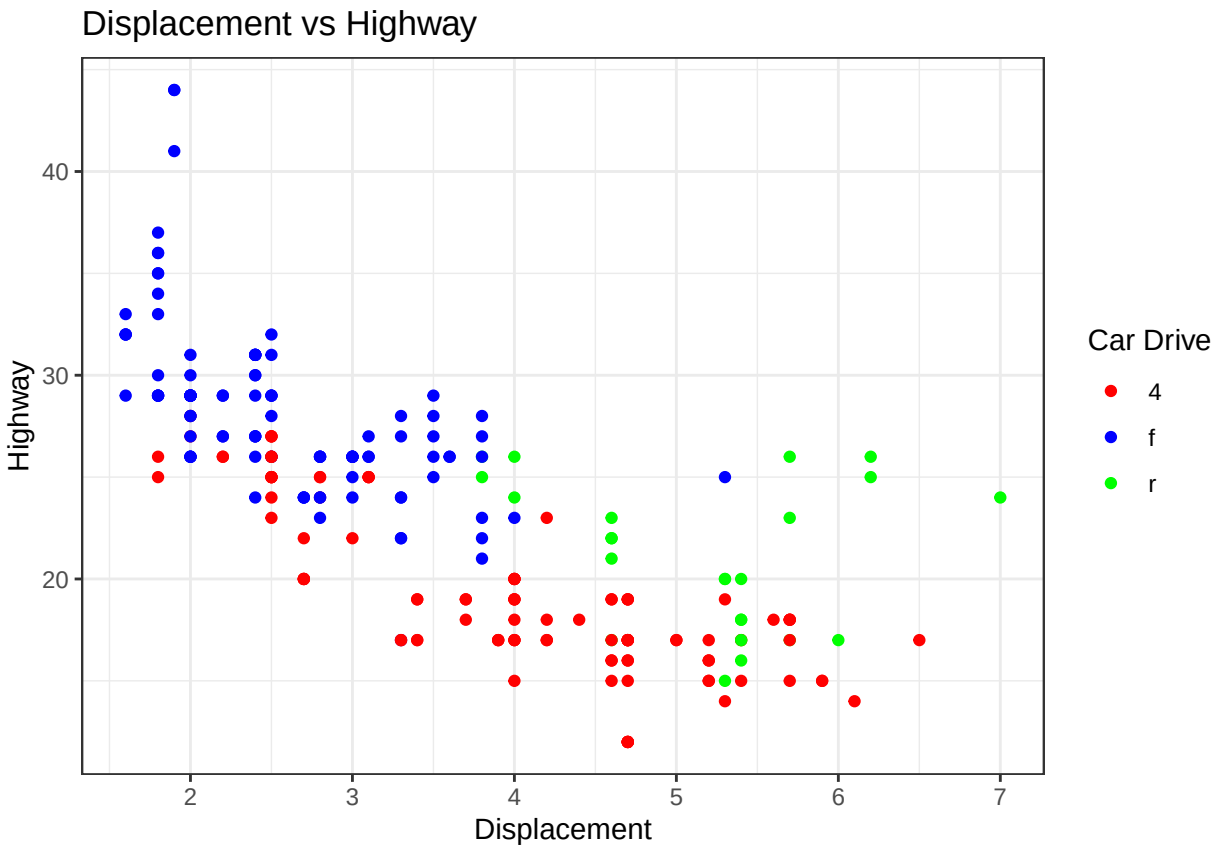
```
## xlab= x-axis label
## ylab= y-axis label
## ggtitle= plot title
## scale_color_colorblind(name = "Car Class")= title for for key
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_bw() +
scale_color_colorblind(name = "Car Class") +
xlab("Displacement") +
ylab("Highway") +
ggtitle("Displacement vs Highway")
```




```
## If you dan't want to use colorblind scale, can use scale_color_discrete, scale_fill_discrete()
## scale_color_continuous, scale_fill_continuous() (depending on situation)
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_bw() +
scale_color_discrete(name = "Car Class") +
xlab("Displacement") +
ylab("Highway") +
ggtitle("Displacement vs Highway")
```




```
## Set own colors manually
scale.colors=c(`4`="red", f="blue", r="green")
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) +
geom_point() +
theme_bw() +
scale_color_manual(name = "Car Drive", values=scale.colors) +
xlab("Displacement") +
ylab("Highway") +
ggtitle("Displacement vs Highway")
```




##Saving Plots

```
## Storing ggplotobject in variable pl
ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = class)) +
geom_point() +
theme_bw() +
scale_color_colorblind() ->
pl

## Saving plot using ggsave
## filename= name of file ypu want to save to
## plot= ggPlot object that you want to put in pdf
## width and height of plot
## family= font for labels
ggsave(filename = "my_first_plot.pdf",
plot = pl,
width = 6,
height = 4,
family = "Times")

## pdf-opens connection to pdf file
pdf(file = "my_second_plot.pdf", width = 6, height = 4, family = "Times")
## prints saved ggPlot object  to pdf file
print(pl)
##closes connection
dev.off()
```

```
## pdf
```

```
##   2
```

Exercise:
Create a boxplot of cut vs price for each level of color. Make the boxplots all orange and use the black and white theme. Save this plot to the output folder using ggsave().