ggplot

J. Wall

2020-04-11

We will look at ways to make your graphs more readable when you produce a report with them. Not necessary when doing data exploration, but when it is time to share the plots, need to make them understandable. We will look at:

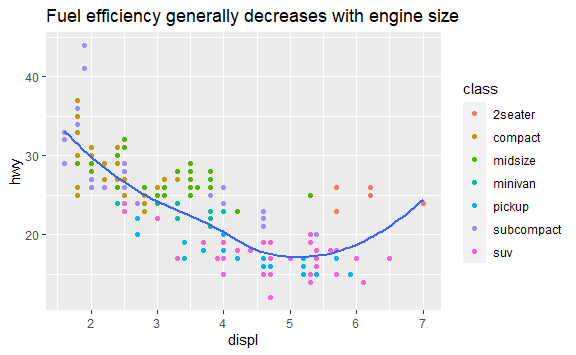
* labels
* annotations
* scales
* themes
* zooming
* saving plots

## Labels

Read 28.1 - 28.2 in book (<https://r4ds.had.co.nz/graphics-for-communication.html#label>)

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 labs(title = "Fuel efficiency generally decreases with engine size")

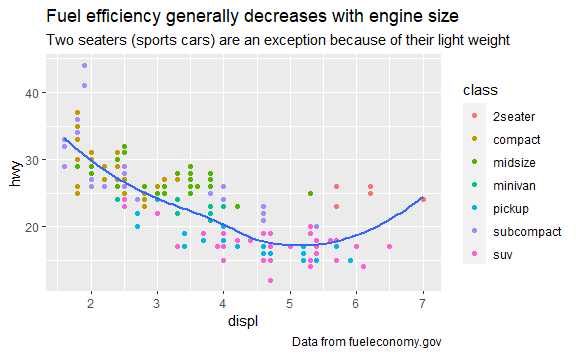
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 Note that we use a title that summarizes the finding, not just says what the graph plots.

* subtitle adds additional detail in a smaller font beneath the title.
* caption adds text at the bottom right of the plot, often used to describe the source of the data.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 labs(  
 title = "Fuel efficiency generally decreases with engine size",  
 subtitle = "Two seaters (sports cars) are an exception because of their light weight",  
 caption = "Data from fueleconomy.gov"  
 )

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

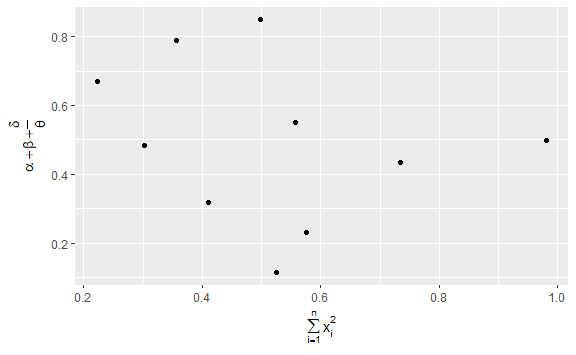


Replace x and y axis labels with longer names and include units

g <- ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_smooth(se = FALSE) +  
 labs(  
 x = "Engine displacement (L)",  
 y = "Highway fuel economy (mpg)",  
 colour = "Car type"  
 )

Use math expressions by using quote function instead of ""

df <- tibble(  
 x = runif(10),  
 y = runif(10)  
)  
ggplot(df, aes(x, y)) +  
 geom\_point() +  
 labs(  
 x = quote(sum(x[i] ^ 2, i == 1, n)),  
 y = quote(alpha + beta + frac(delta, theta))  
 )



-**Exercise 1:**  (1/2 point) Take any graph for your project work and add a meaningful title, subtitle and axis labels.

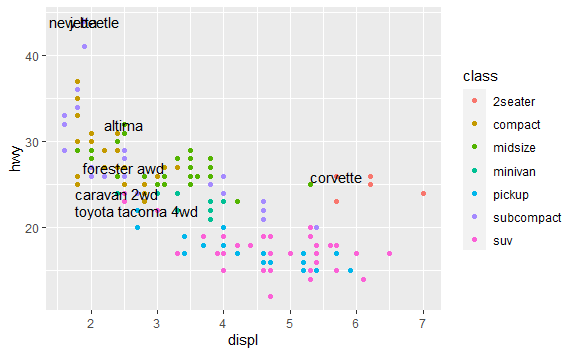
## Annotations

Read 28.3 in book (<https://r4ds.had.co.nz/graphics-for-communication.html#annotations>) Can use geom\_text() which is like geom\_point but can take a label as an argument. Other alternatives are geom\_label and ggrepel:: geom\_label\_repel.

best\_in\_class <- mpg %>%  
 group\_by(class) %>%  
 filter(row\_number(desc(hwy)) == 1)

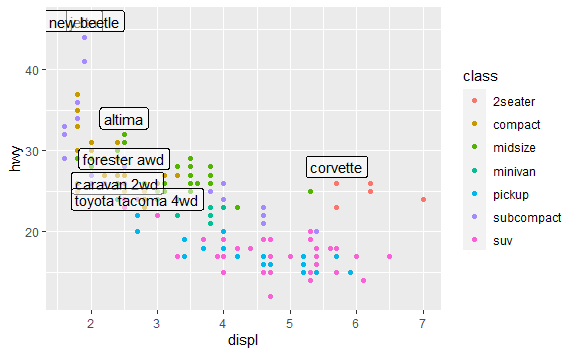
## Warning in rep.default(list\_of(integer()), length = nrow(groups)): partial  
## argument match of 'length' to 'length.out'

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_text(aes(label = model), data = best\_in\_class)



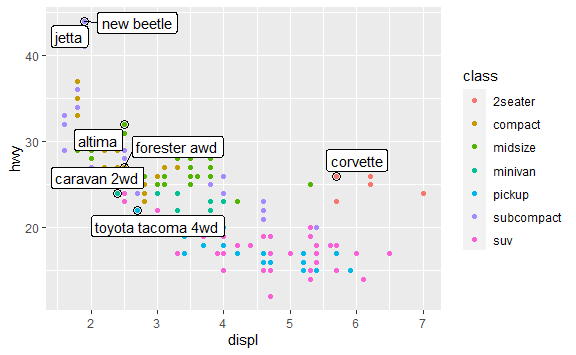
geom\_label draws a rectangle around the label and can use nudge\_y to move the label off the point, but does not keep labels from being place on top of each other.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_label(aes(label = model), data = best\_in\_class,   
 nudge\_y = 2, alpha = 0.5)

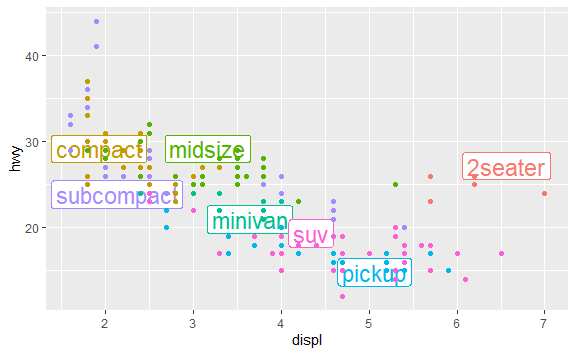


Use ggrepel::geom\_label\_repel() instead when needed to keep labels from printing on top of each other. Or use it to put legends on the graph instead of beside or below graph.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_point(size = 3, shape = 1, data = best\_in\_class) +  
 ggrepel::geom\_label\_repel(aes(label = model), data = best\_in\_class)



# putting legend labels on the graph in the middle of their data  
class\_avg <- mpg %>%  
 group\_by(class) %>%  
 summarise(  
 displ = median(displ),  
 hwy = median(hwy)  
 )  
  
ggplot(mpg, aes(displ, hwy, colour = class)) +  
 ggrepel::geom\_label\_repel(aes(label = class),  
 data = class\_avg,  
 size = 6,  
 label.size = 0,  
 segment.color = NA  
 ) +  
 geom\_point() +  
 theme(legend.position = "none")

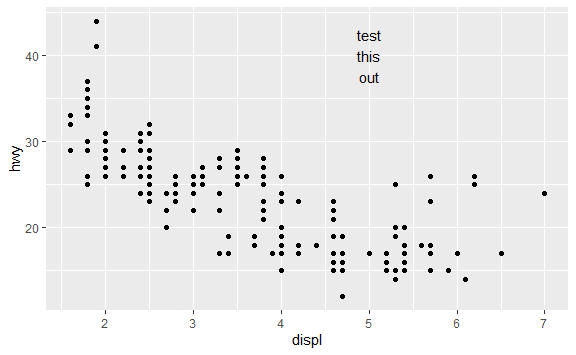


-**Exercise 2:**  (1/2 point) Add labels to a graph of your choice (preferable the one used for the previous exercise).

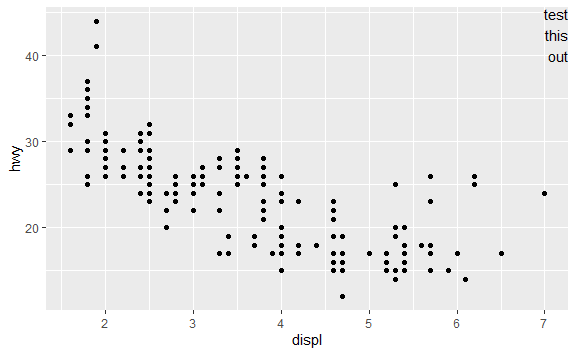
## Add text as a single label

you can add a single label to a graph by creating a data frame containing the text

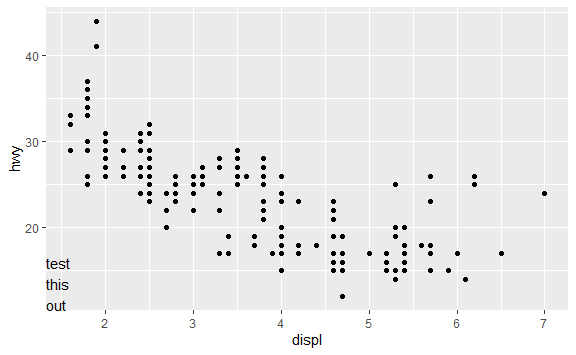
text\_df <- tibble(text = "test\nthis\nout", x = 5, y = 40)  
g1 <- ggplot(mpg, aes(displ, hwy)) +  
 geom\_point()  
g1 + geom\_text(aes(x, y, label = text), data = text\_df)



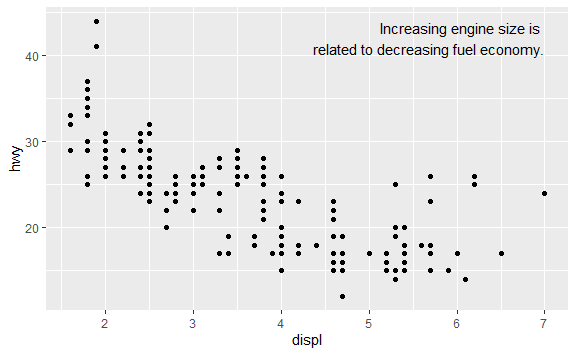
# to place on the border of the plot, use +Inf for the values  
text\_df <- tibble(text = "test\nthis\nout", x = Inf, y = Inf)  
g1 + geom\_text(aes(x, y, label = text), data = text\_df,  
 vjust = "top", hjust = "right")



# do the same, but put the text in the bottom left corner  
text\_df <- tibble(text = "test\nthis\nout", x = -Inf, y = -Inf)  
g1 + geom\_text(aes(x, y, label = text), data = text\_df,  
 vjust = "bottom", hjust = "left")



# use summarize to find max sizes  
label <- mpg %>%  
 summarise(  
 displ = max(displ),  
 hwy = max(hwy),  
 label = "Increasing engine size is \nrelated to decreasing fuel economy."  
 )  
  
g1 + geom\_text(aes(label = label), data = label,   
 vjust = "top", hjust = "right")

 \* geom\_hline() and geom\_vline() to add reference lines.

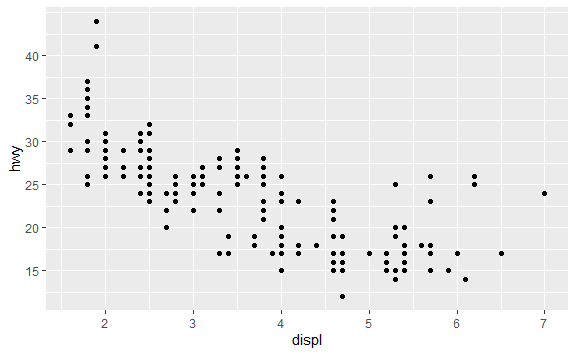
* geom\_rect() to draw a rectangle around points of interest. The boundaries of the rectangle are defined by aesthetics xmin, xmax, ymin, ymax.
* geom\_segment() with the arrow argument to draw attention to a point with an arrow. Use aesthetics x and y to define the starting location, and xend and yend to define the end location.

-**Exercise 3:**  (1/2 point) Add a text box to one corner of your graph and put either a rectangle or an arrow on your graph.

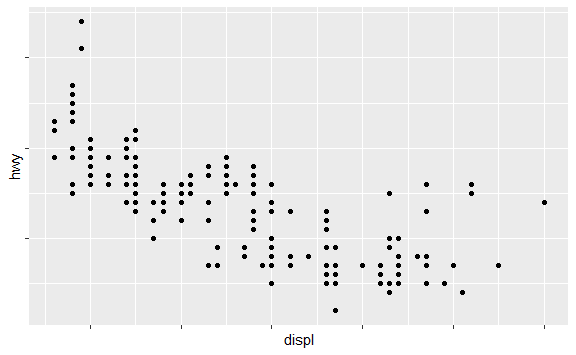
## Scales

Read 28.4 in book (<https://r4ds.had.co.nz/graphics-for-communication.html#scales>) Override the default tick marks.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point() +  
 scale\_y\_continuous(breaks = seq(15, 40, by = 5))

 To take the tick labels off, set the scale to NULL.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point() +  
 scale\_x\_continuous(labels = NULL) +  
 scale\_y\_continuous(labels = NULL)

 Now let’s set the ticks to be the years when presidents changed.

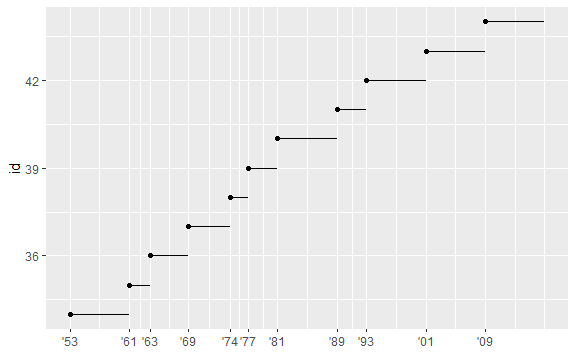
presidential

## # A tibble: 11 x 4  
## name start end party   
## <chr> <date> <date> <chr>   
## 1 Eisenhower 1953-01-20 1961-01-20 Republican  
## 2 Kennedy 1961-01-20 1963-11-22 Democratic  
## 3 Johnson 1963-11-22 1969-01-20 Democratic  
## 4 Nixon 1969-01-20 1974-08-09 Republican  
## 5 Ford 1974-08-09 1977-01-20 Republican  
## 6 Carter 1977-01-20 1981-01-20 Democratic  
## 7 Reagan 1981-01-20 1989-01-20 Republican  
## 8 Bush 1989-01-20 1993-01-20 Republican  
## 9 Clinton 1993-01-20 2001-01-20 Democratic  
## 10 Bush 2001-01-20 2009-01-20 Republican  
## 11 Obama 2009-01-20 2017-01-20 Democratic

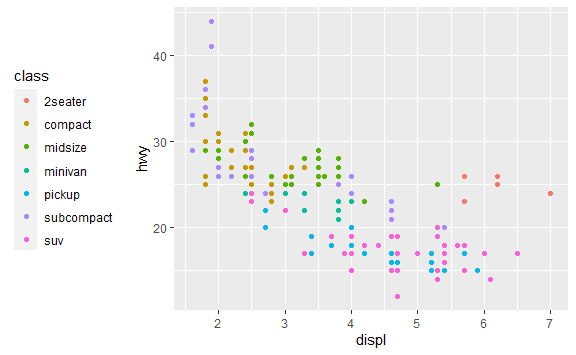
presidential %>%  
 mutate(id = 33 + row\_number())

## # A tibble: 11 x 5  
## name start end party id  
## <chr> <date> <date> <chr> <dbl>  
## 1 Eisenhower 1953-01-20 1961-01-20 Republican 34  
## 2 Kennedy 1961-01-20 1963-11-22 Democratic 35  
## 3 Johnson 1963-11-22 1969-01-20 Democratic 36  
## 4 Nixon 1969-01-20 1974-08-09 Republican 37  
## 5 Ford 1974-08-09 1977-01-20 Republican 38  
## 6 Carter 1977-01-20 1981-01-20 Democratic 39  
## 7 Reagan 1981-01-20 1989-01-20 Republican 40  
## 8 Bush 1989-01-20 1993-01-20 Republican 41  
## 9 Clinton 1993-01-20 2001-01-20 Democratic 42  
## 10 Bush 2001-01-20 2009-01-20 Republican 43  
## 11 Obama 2009-01-20 2017-01-20 Democratic 44

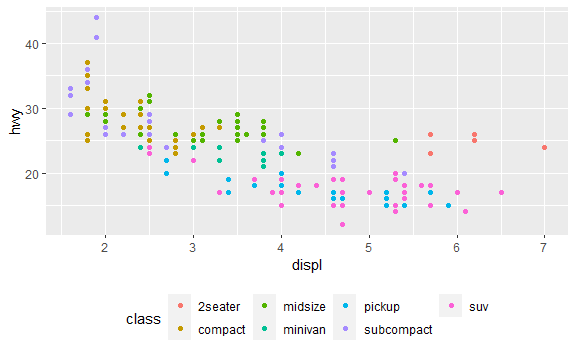
presidential %>%  
 mutate(id = 33 + row\_number()) %>%  
 ggplot(aes(start, id)) +  
 geom\_point() +  
 geom\_segment(aes(xend = end, yend = id)) +  
 scale\_x\_date(NULL,   
 breaks = presidential$start,  
 date\_labels = "'%y")

 Control where the legend is placed.

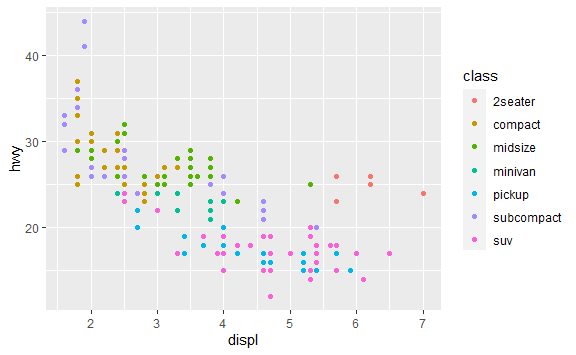
base <- ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class))  
  
base + theme(legend.position = "left")



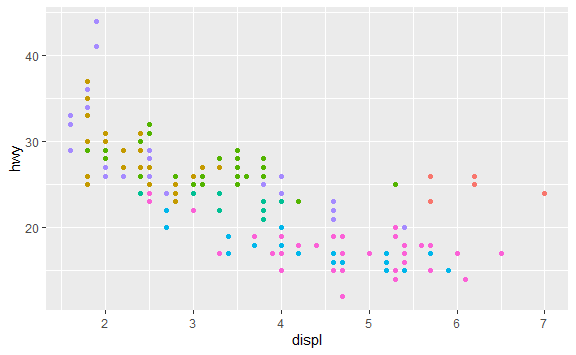
base + theme(legend.position = "bottom")



base + theme(legend.position = "right")



base + theme(legend.position = "none")



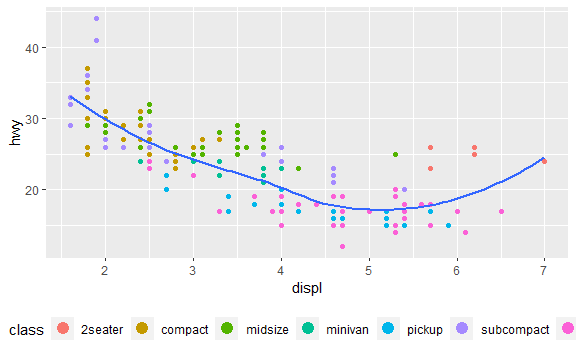
-**Exercise 4:**  (1/2 point) Modify the breaks on either the x or y axis of your graph.

## guides()

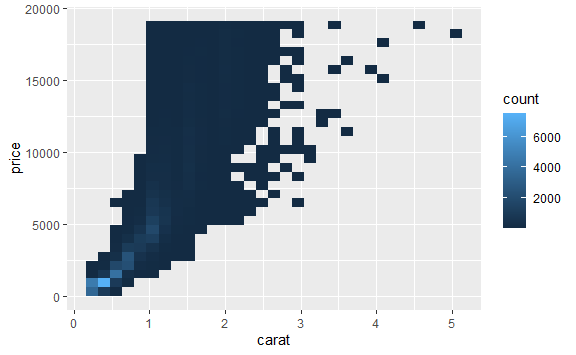
To control the display of individual legends, use guides(). The following example shows two important settings: controlling the number of rows the legend uses with nrow, and overriding one of the aesthetics to make the points bigger.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_smooth(se = FALSE) +  
 theme(legend.position = "bottom") +  
 guides(colour = guide\_legend(nrow = 1,   
 override.aes = list(size = 4)))

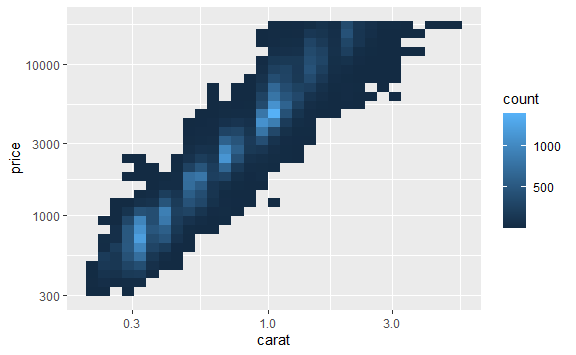
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 We can overwrite the default scale.

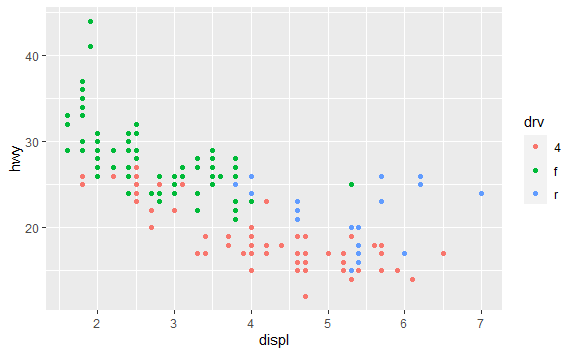
ggplot(diamonds, aes(carat, price)) +  
 geom\_bin2d()



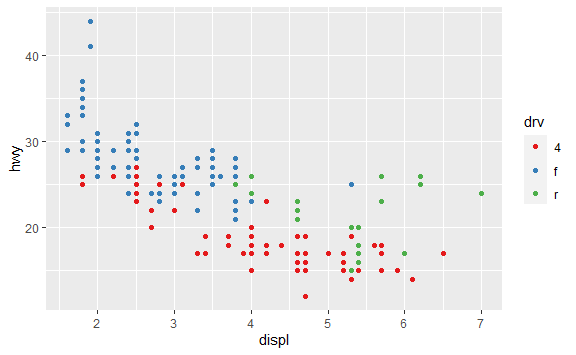
ggplot(diamonds, aes(carat, price)) +  
 geom\_bin2d() +   
 scale\_x\_log10() +   
 scale\_y\_log10()



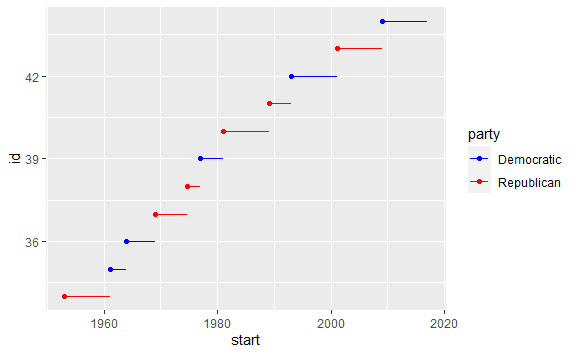
# or change the color palette  
  
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = drv))



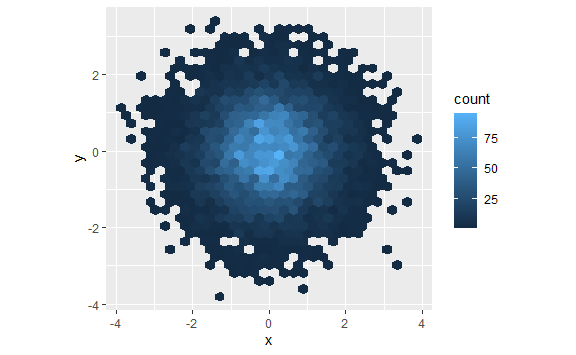
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = drv)) +  
 scale\_colour\_brewer(palette = "Set1")



presidential %>%  
 mutate(id = 33 + row\_number()) %>%  
 ggplot(aes(start, id, colour = party)) +  
 geom\_point() +  
 geom\_segment(aes(xend = end, yend = id)) +  
 scale\_colour\_manual(values = c(Republican = "red", Democratic = "blue"))



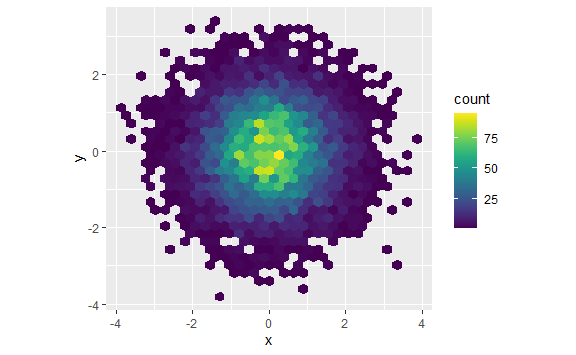
df <- tibble(  
 x = rnorm(10000),  
 y = rnorm(10000)  
)  
ggplot(df, aes(x, y)) +  
 geom\_hex() +  
 coord\_fixed()



#> Loading required package: methods  
library(viridis)

## Loading required package: viridisLite

ggplot(df, aes(x, y)) +  
 geom\_hex() +  
 viridis::scale\_fill\_viridis() +  
 coord\_fixed()



-**Exercise 5:**  Choose 2 different color palettes (besides the default) for your graph. Put both in your knitted document.

## Zooming

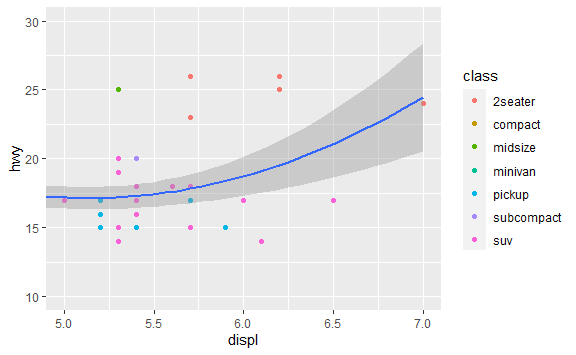
Read 28.5 in book(<https://r4ds.had.co.nz/graphics-for-communication.html#zooming>)

3 ways to zoom:

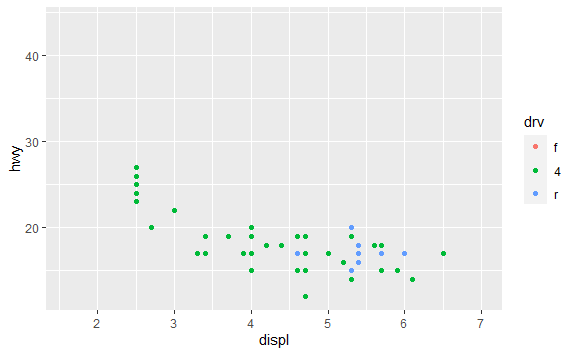
* Adjust what data are plotted ( but this will effect any smoothing you are doing)
* Set the limits in each scale
* Set xlim and ylim in coord\_cartesian()

ggplot(mpg, mapping = aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth() +  
 coord\_cartesian(xlim = c(5, 7), ylim = c(10, 30))

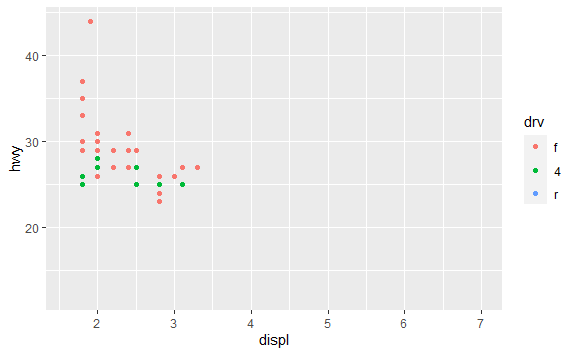
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



# set limits for the entire data set  
x\_scale <- scale\_x\_continuous(limits = range(mpg$displ))  
y\_scale <- scale\_y\_continuous(limits = range(mpg$hwy))  
col\_scale <- scale\_colour\_discrete(limits = unique(mpg$drv))  
  
# use those limits for graphs of different data subsets  
suv <- mpg %>% filter(class == "suv")  
compact <- mpg %>% filter(class == "compact")  
  
ggplot(suv, aes(displ, hwy, colour = drv)) +  
 geom\_point() +  
 x\_scale +  
 y\_scale +  
 col\_scale



ggplot(compact, aes(displ, hwy, colour = drv)) +  
 geom\_point() +  
 x\_scale +  
 y\_scale +  
 col\_scale



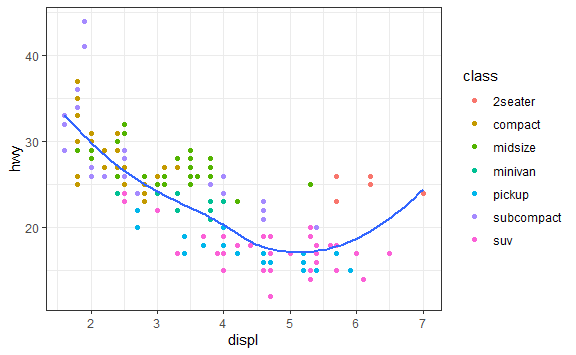
## Themes

Read 28.6 in book (<https://r4ds.had.co.nz/graphics-for-communication.html#themes>) Themes built in to ggplot: <http://r4ds.had.co.nz/images/visualization-themes.png>

Explore a couple of these by adding for instance + theme\_bw() to your plot.

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 theme\_bw()

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



# you can install ggthemes and get many more themes

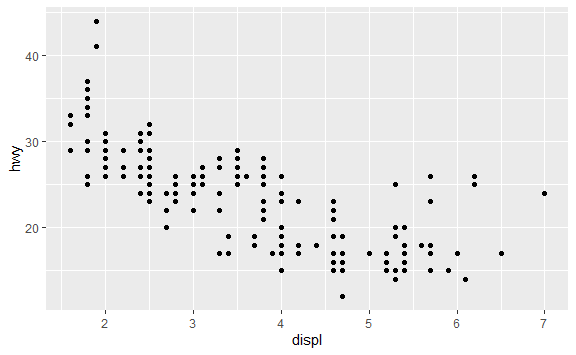
-**Exercise 6:**  Use two different themes (besides the default) for your graph.

## Saving your plots

Read 28.7 - 28.8 in book (<https://r4ds.had.co.nz/graphics-for-communication.html#saving-your-plots>)

Use ggsave() to save your plot to disk.

ggplot(mpg, aes(displ, hwy)) + geom\_point()



ggsave("plot\_displ\_hwy.pdf")

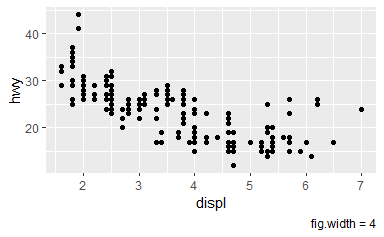
## Saving 6 x 3.7 in image

Note that this saves a different size image when I knit the document than if I entered the commands from the console.

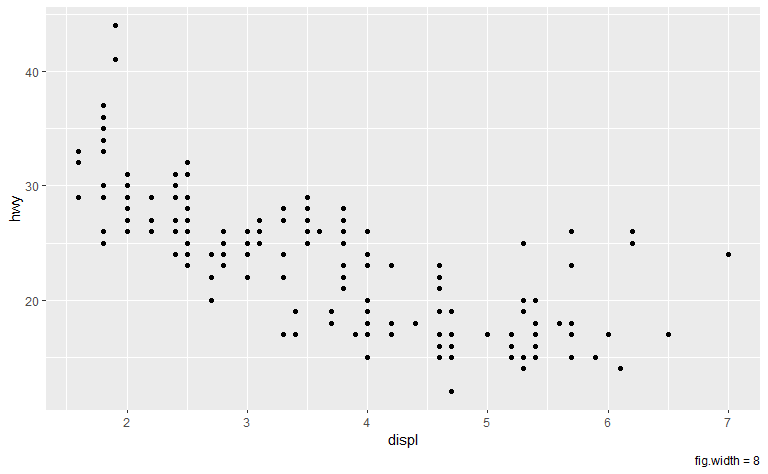
There are five main options that control figure sizing: fig.width, fig.height, fig.asp, out.width and out.height. We can specify these in the chunk options. Let’s look at default options of out.width = “70%” with fig.align = “center”. This says the figure will take 70 % of the available line space and be centered on the page. We can set the defaults of fig.width = 6 (6") and fig.asp = 0.618 (golden ratio), then we can overwrite them in individual chunks if needed.

What happens if we change fig.width in an individual code chunk?

ggplot(mpg, aes(displ, hwy)) + geom\_point() +   
 labs(caption = "fig.width = 4")



ggplot(mpg, aes(displ, hwy)) + geom\_point() +  
 labs(caption = "fig.width = 8")



-**Exercise 7:**  Use three different size options for your graph. Include all three in your knitted document.  
Save the one you like the best as a .jpg file.