R4: Visualising Data Relations

Data visualisation with {ggplot2} in {tidyverse}

Andreas Reschreiter

2023-12-14

Table of contents

1 Libraries

```
suppressPackageStartupMessages({
  library(tidyverse) # includes ggplot2
})
```

2 Data

We will visualize the `datasets::mtcars` data.

```
glimpse(mtcars)
```

```
Rows: 32
Columns: 11
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,~
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,~
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16~
$ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180~
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,~
$ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.~
$ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18~
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,~
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,~
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,~
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,~
```

3 Date manipultation

Fix vs/am labels first:

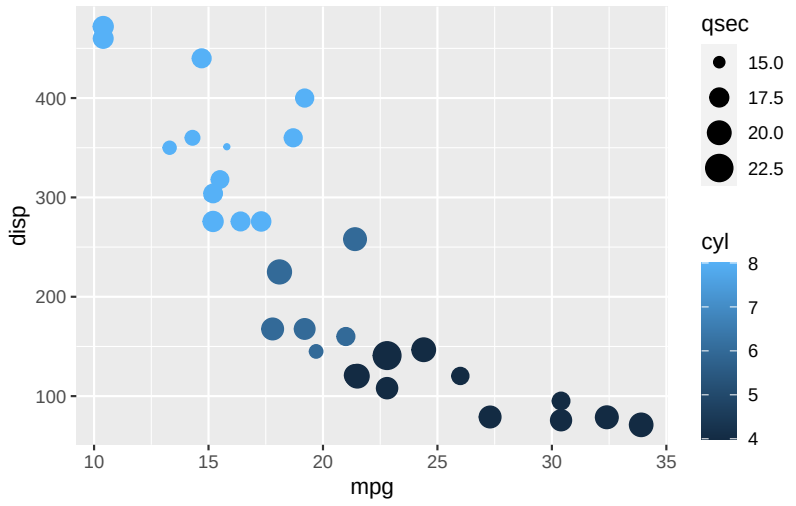
```
cars <- as_tibble(mtcars, rownames = "names") %>%
  mutate(am = ifelse(am == 1, "manual", "automatic"),
                     vs = ifelse(vs == 1, "straight", "v-shaped"))
```

4 Make {ggplot2} plots

1. To create a plot, create a `ggplot()` object (and link it to a data set)
2. Define the (default) mapping of variables to "aestethics".
3. Choose a "geom". This can define specific aesthetics
4. Add context (title, legends, axis)

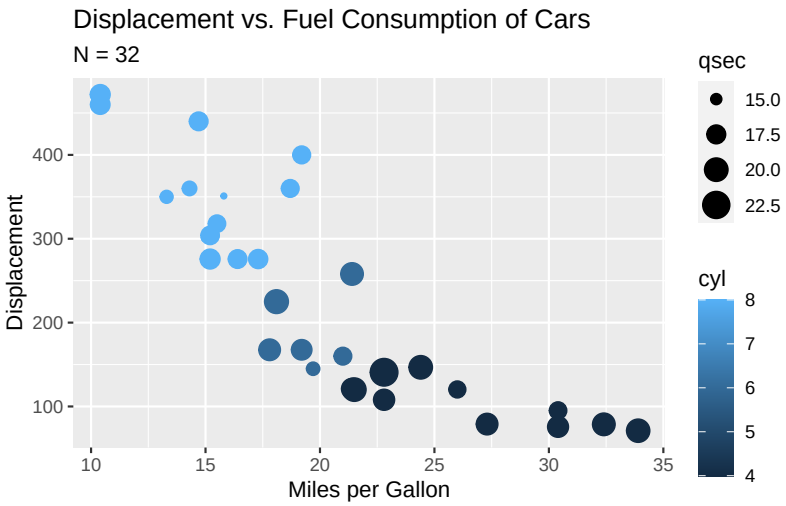
5 A {ggplot2} plot

```
g <- ggplot(cars) +
    aes(x = mpg, y = disp) + ## apply to all geoms
    geom_point(aes(col = cyl, size = qsec))
g
```



6 Labels

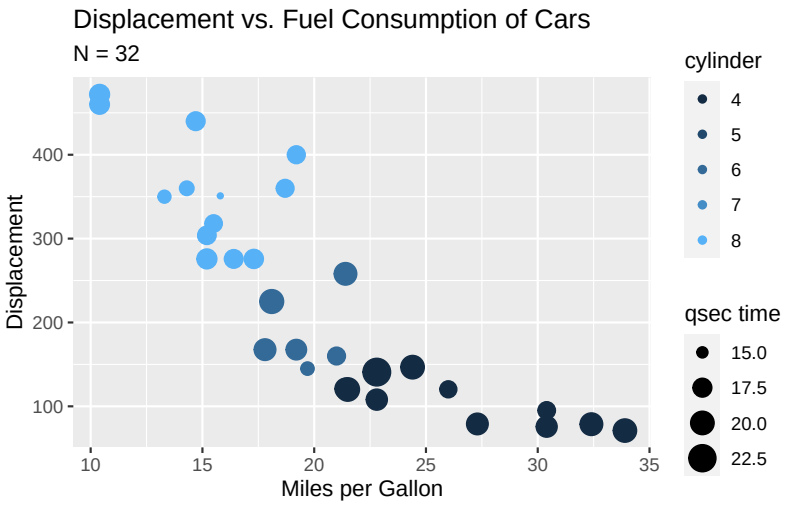
```
g <- g + ggtitle("Displacement vs. Fuel Consumption of Cars",
            subtitle = paste0("N = ", nrow(cars))) +
  xlab("Miles per Gallon") + ylab("Displacement")
g
```



7 Legends

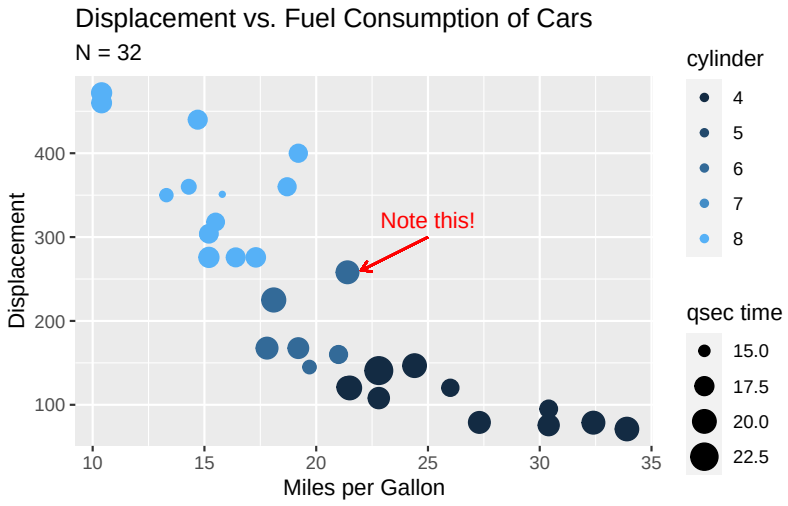
```
## use discrete values instead of color bar
g <- g + guides(col = "legend") +

  ## modify guide names
  scale_color_continuous(name = "cylinder") +
  scale_size_continuous(name = "qsec time")
g
```



8 Annotations

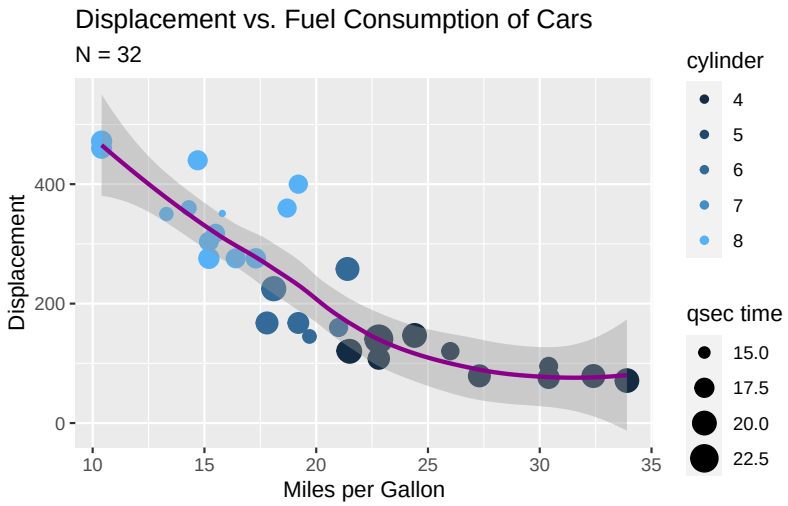
```
g +
  geom_segment(aes(x = 25, y = 300, xend = 22, yend = 260),
               arrow = arrow(length = unit(0.2, "cm")),
               col = "red") +
  annotate("text", 25, 320, label = "Note this!", col = "red")
```



9 Several layers (geoms) in one graph

The graphs are "layered", so that several geoms can be combined:

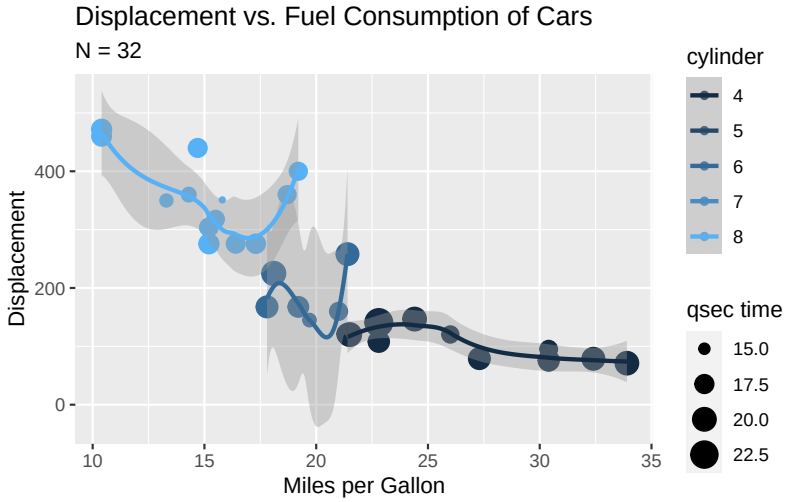
```
g + geom_smooth(col = "dark magenta")
```



10 Groupings

If categorical variables are mapped in an aestethic, they automatically define visual groups:

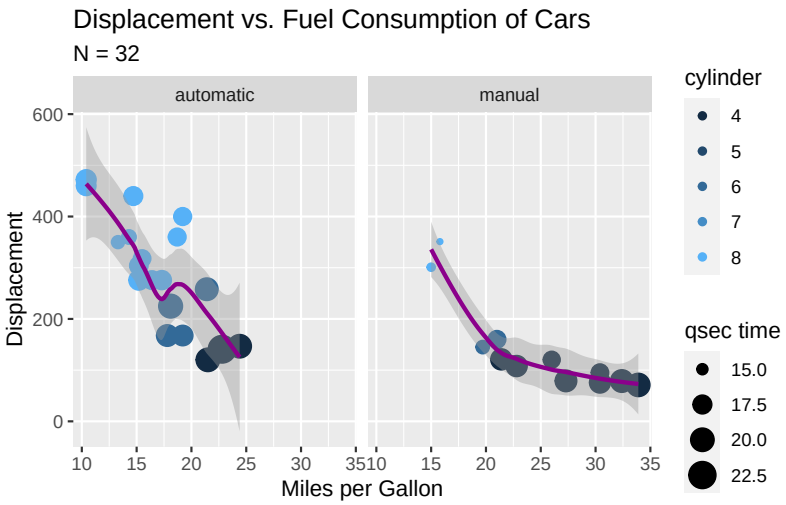
```
g + geom_smooth(aes(group = cyl, col = cyl))
```



11 Facets

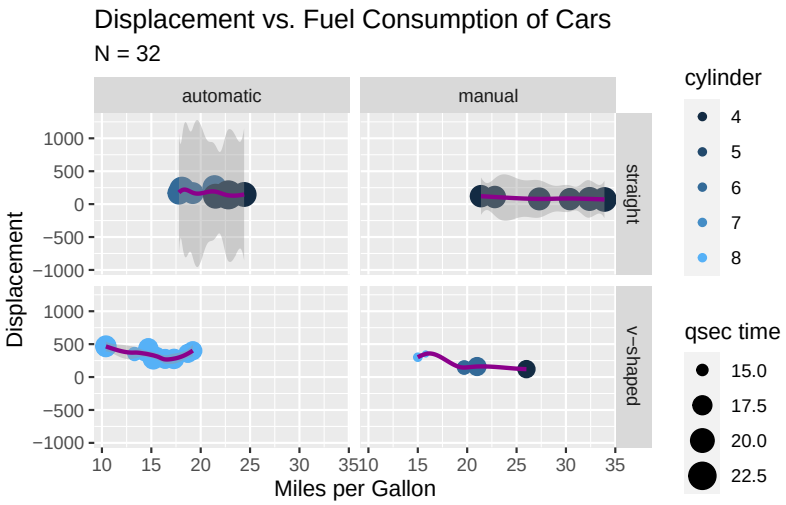
Another kind of layer are "facets", used to create plots conditioned by factors:

```
g +
  geom_smooth(col = "dark magenta") +
  facet_wrap(~ am)
```



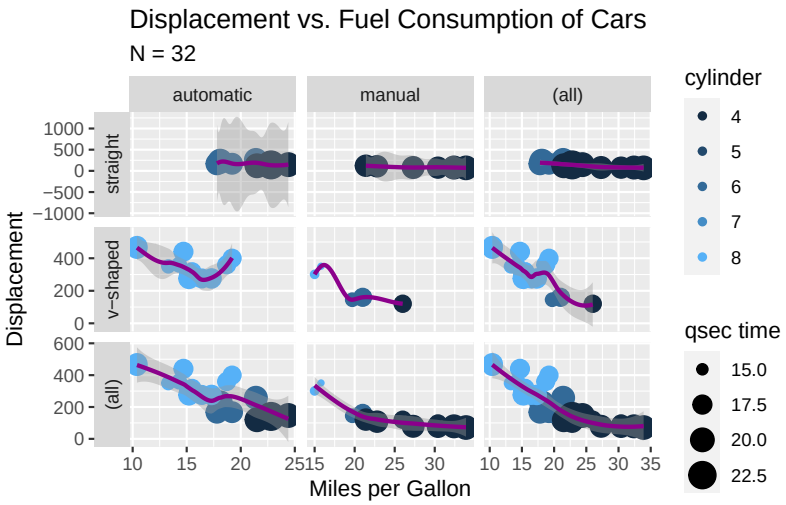
12 Two conditioning variables

```
g +
  geom_smooth(col = "dark magenta") +
  facet_grid(vs ~ am)
```



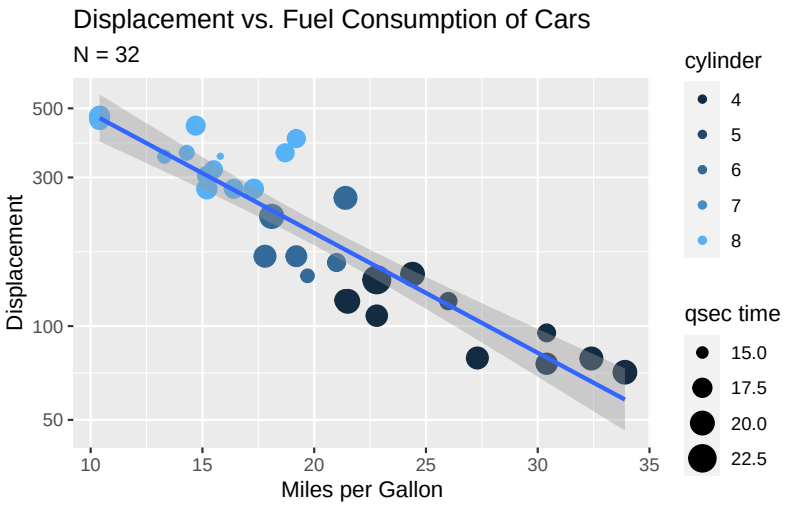
13 Some more tweaks:

```
g +
  geom_smooth(col = "dark magenta") +
  facet_grid(vs ~ am,
             margins = TRUE,
             scales = "free",
#             space = "free",
             switch = "y")
```

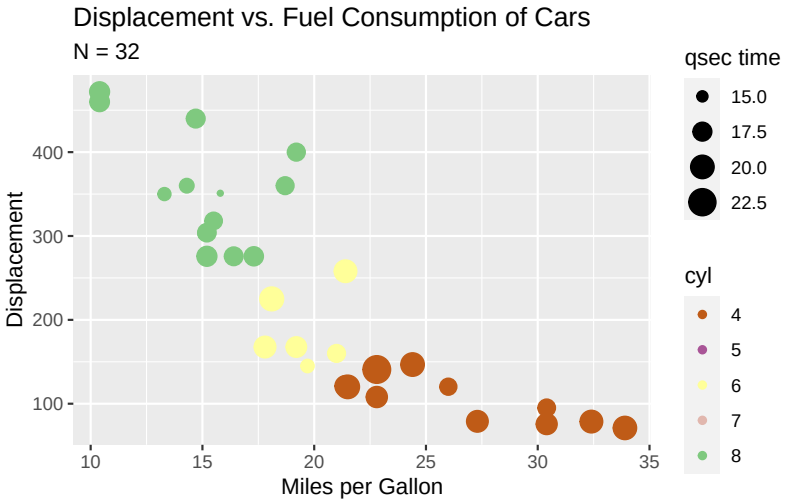


14 Scales

```
g + scale_y_log10() + geom_smooth(method = "lm")
```

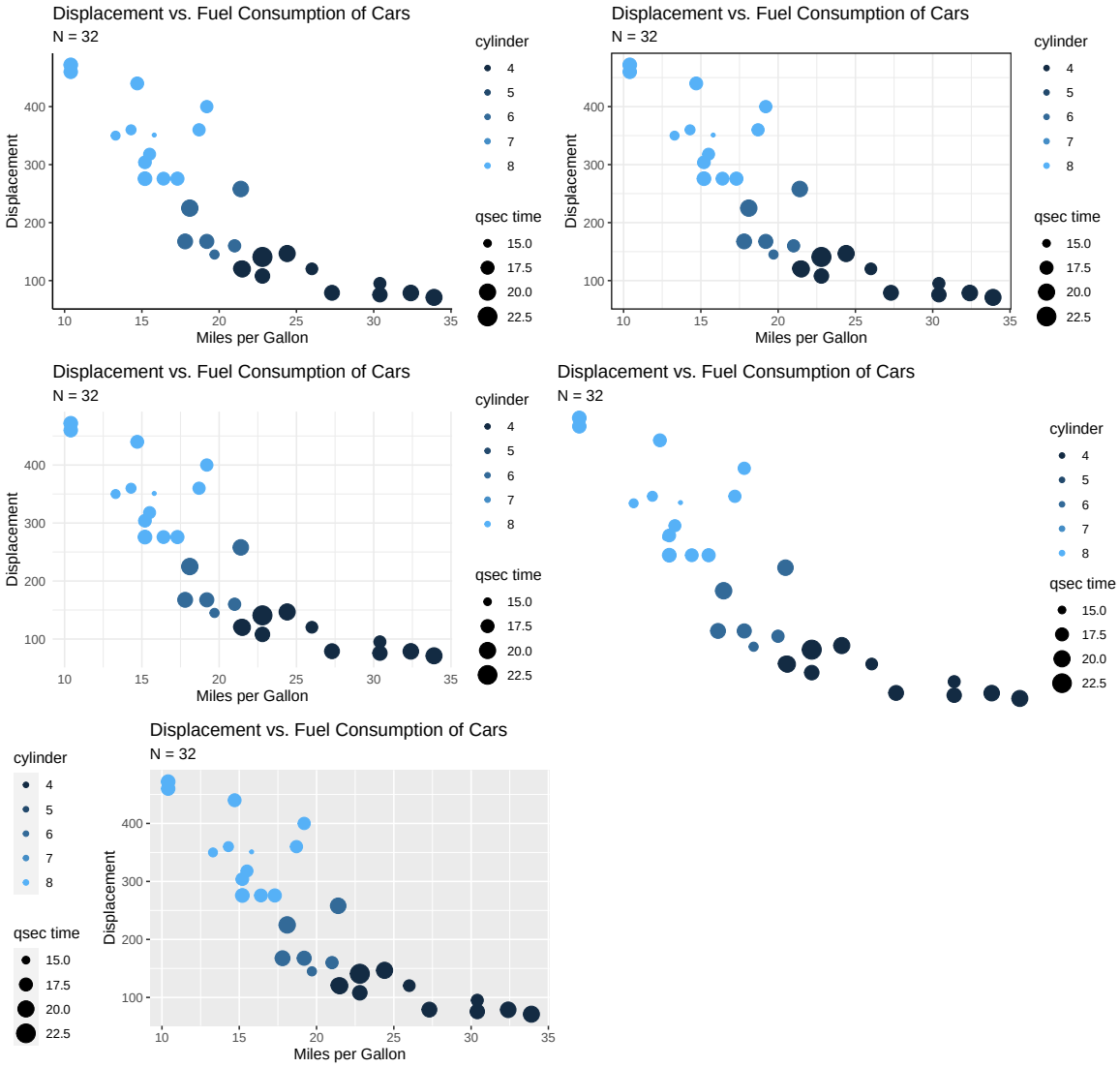



```
g + scale_color_distiller(type = "qual") ## formally, cylinder is quantitative!
```



15 Themes

```
g + theme_classic()
g + theme_bw()
g + theme_minimal()
g + theme_void()
g + theme(legend.position = "left")
```


Displacement vs. Fuel Consumption of Cars
N = 32
Displacement
Miles per Gallon
cylinder
4
5
6
7
8
qsec time
15.0
17.5
20.0
22.5
100
200
300
400
10
15
20
25
30
35

16 Saving plots

```
ggsave("test.png")
ggsave("test.pdf")
ggsave("test.jpeg", g) ## save plot from a ggplot object
```

17 Include saved plots

Figure 1 includes the saved plot via a code chunk.

```
knitr::include_graphics("test.png")
# use ouside of junks  `![](test.png)` to include graphs
```

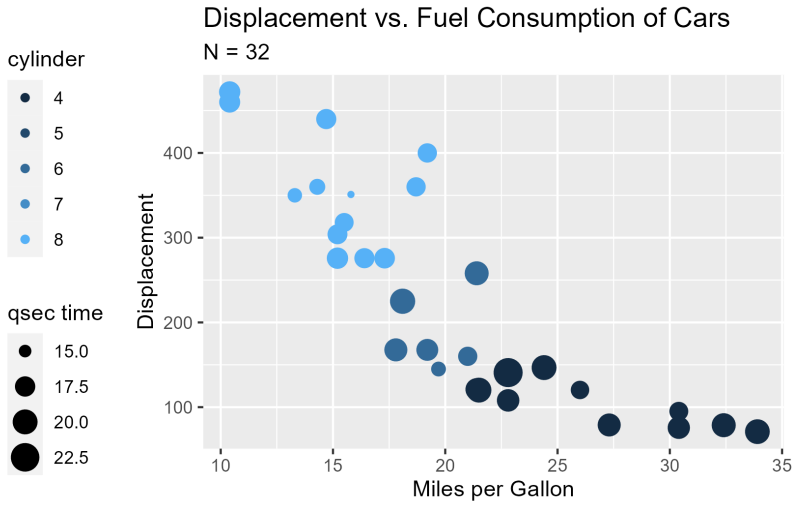


Figure 1: A saved plot included via Code chunk

Can include saved plot via Markdown code:

```
![Plot included from outside of code chunk](test.png)
```

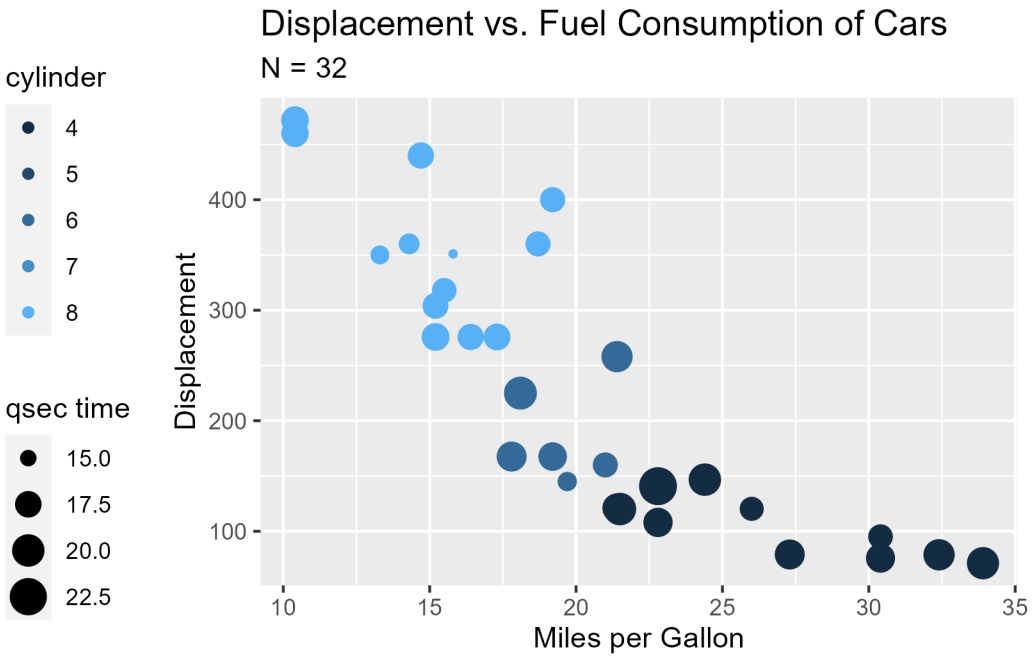


Figure 2: Plot included from outside of code chunk