Homework #3

Author

L. Harvey

Homework #3

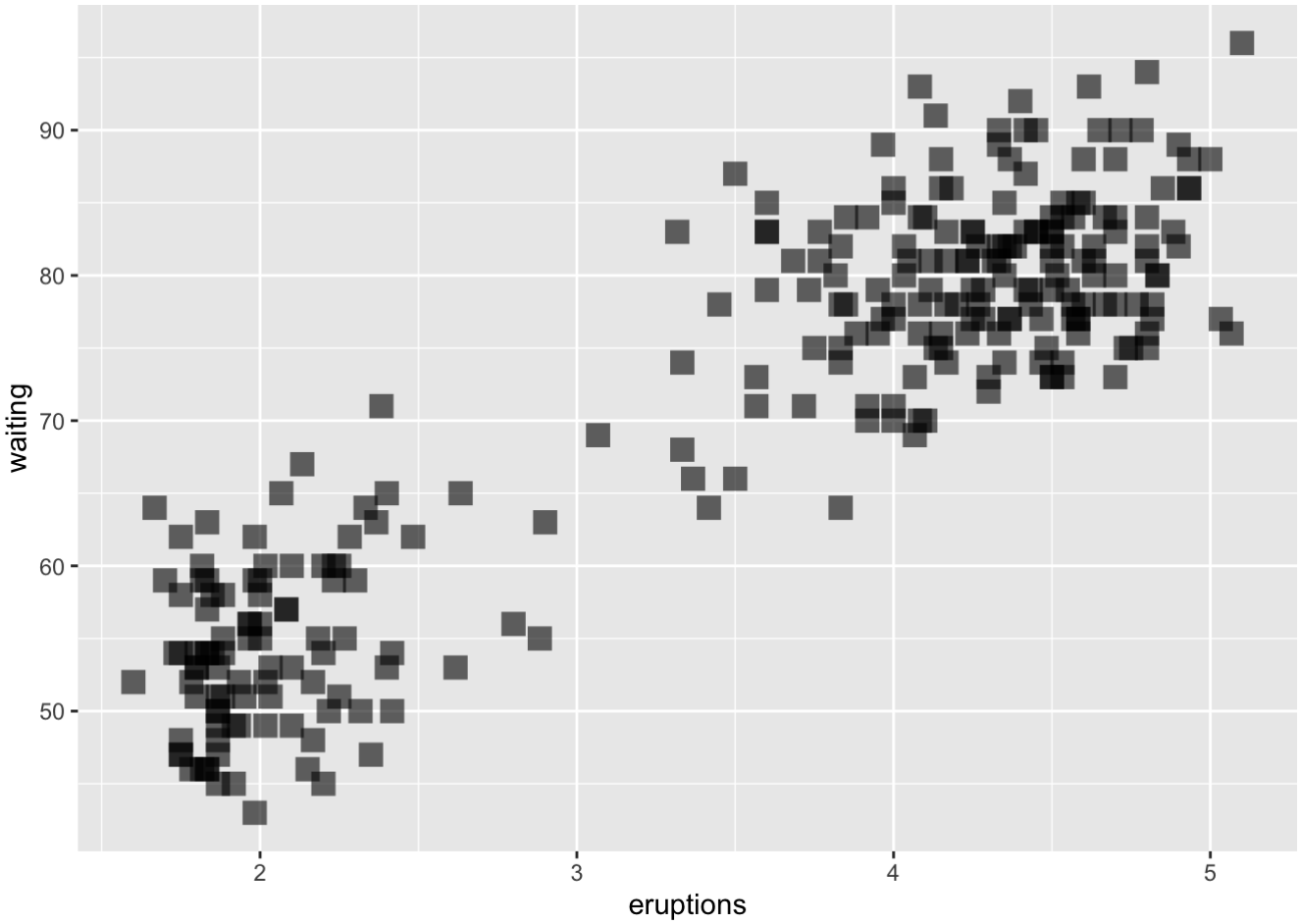
Exercise 3.1.1.1: Introduction

Modify the code given on the homework to make the points larger, slightly-transparent squares.

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.2     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.3     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

```
ggplot(faithful) +
  geom_point(aes(x = eruptions, y = waiting),
    alpha = .6,
    shape = 15,
    size = 4)
```



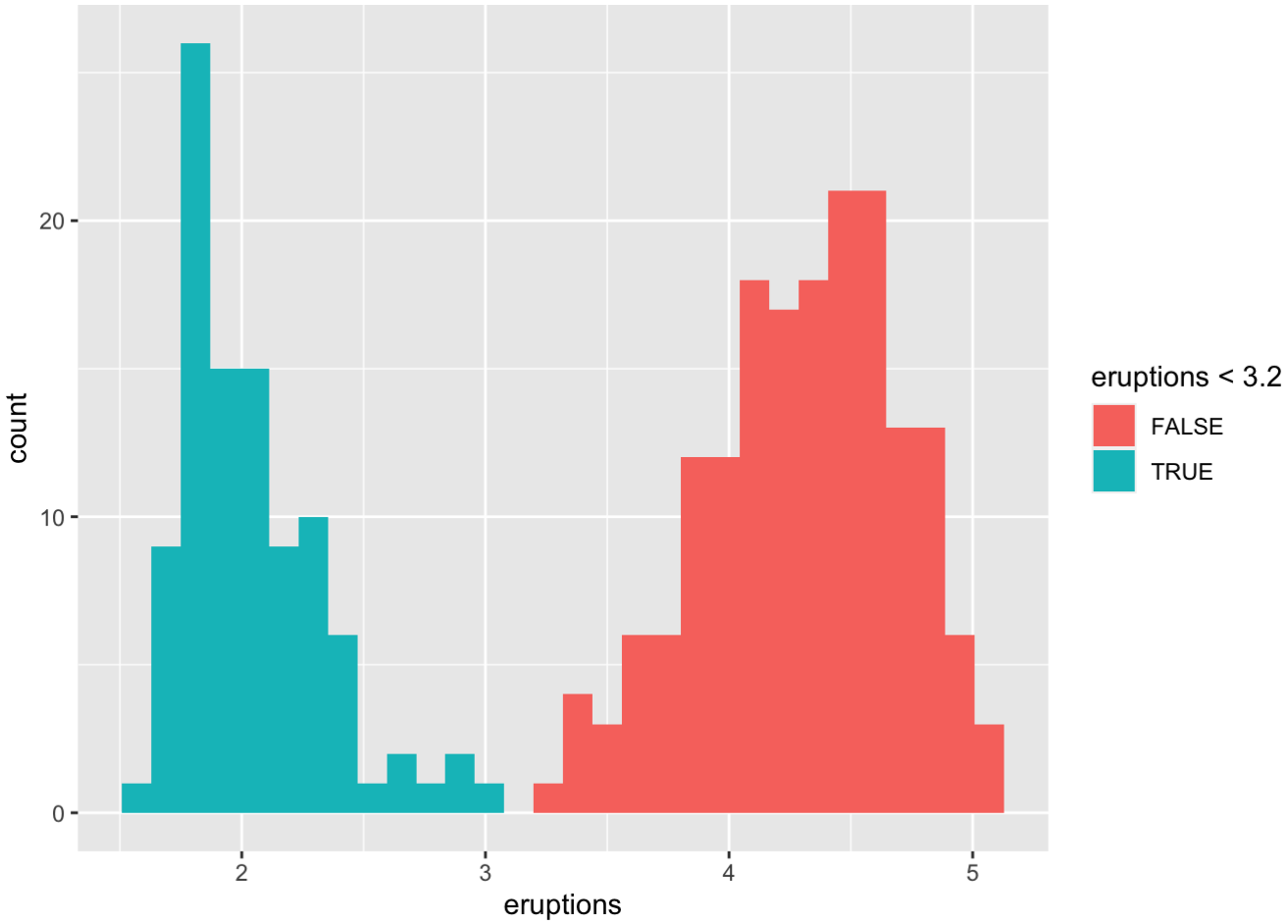
(Look at those handsome little squares!!!)

Exercise 3.1.1.2

Putting some color on this graph!

```
ggplot(faithful) +
  geom_histogram(aes(x = eruptions, fill = eruptions < 3.2))
```

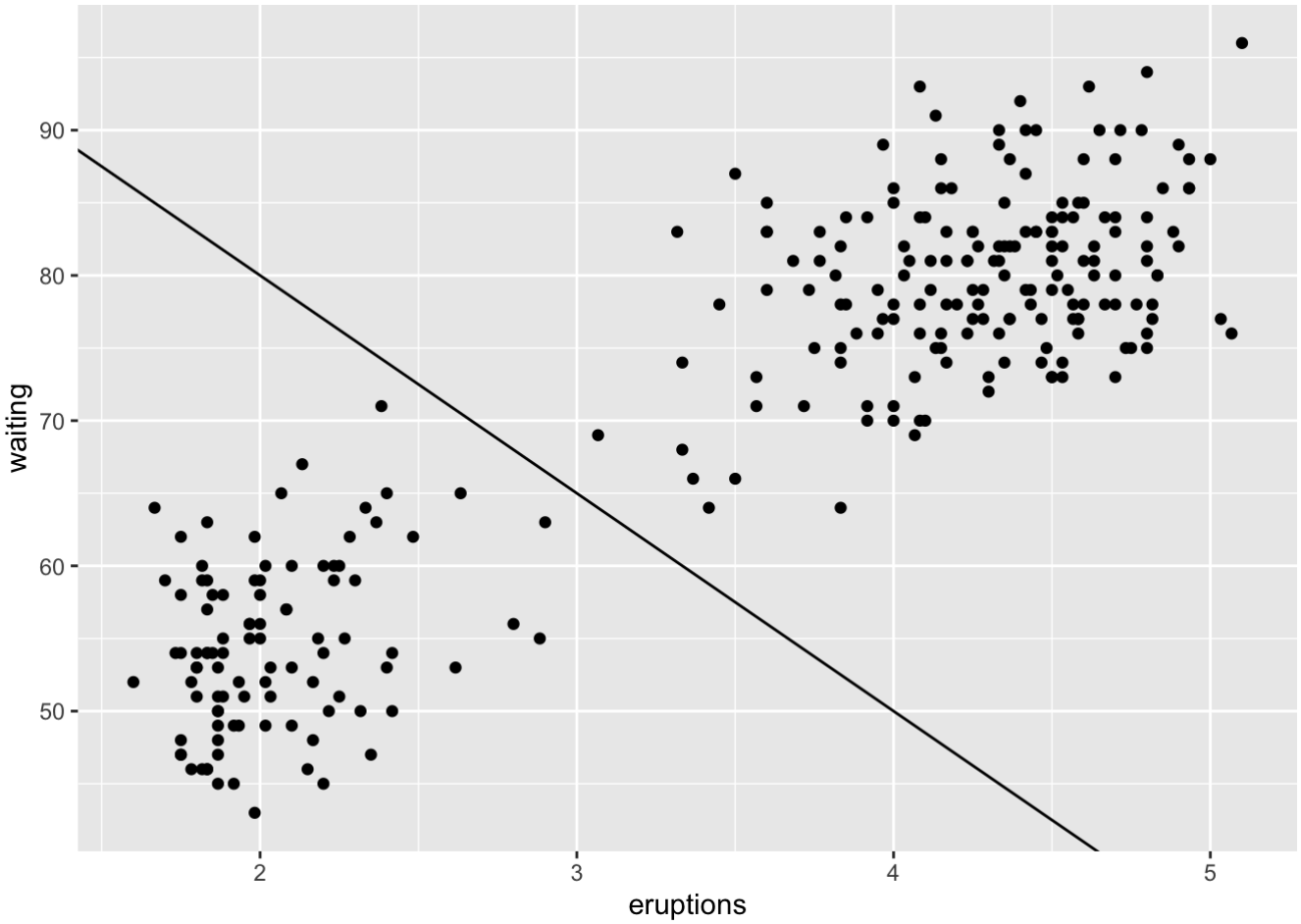
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Exercise 3.1.1.3

Adding a line to separate the distributions.

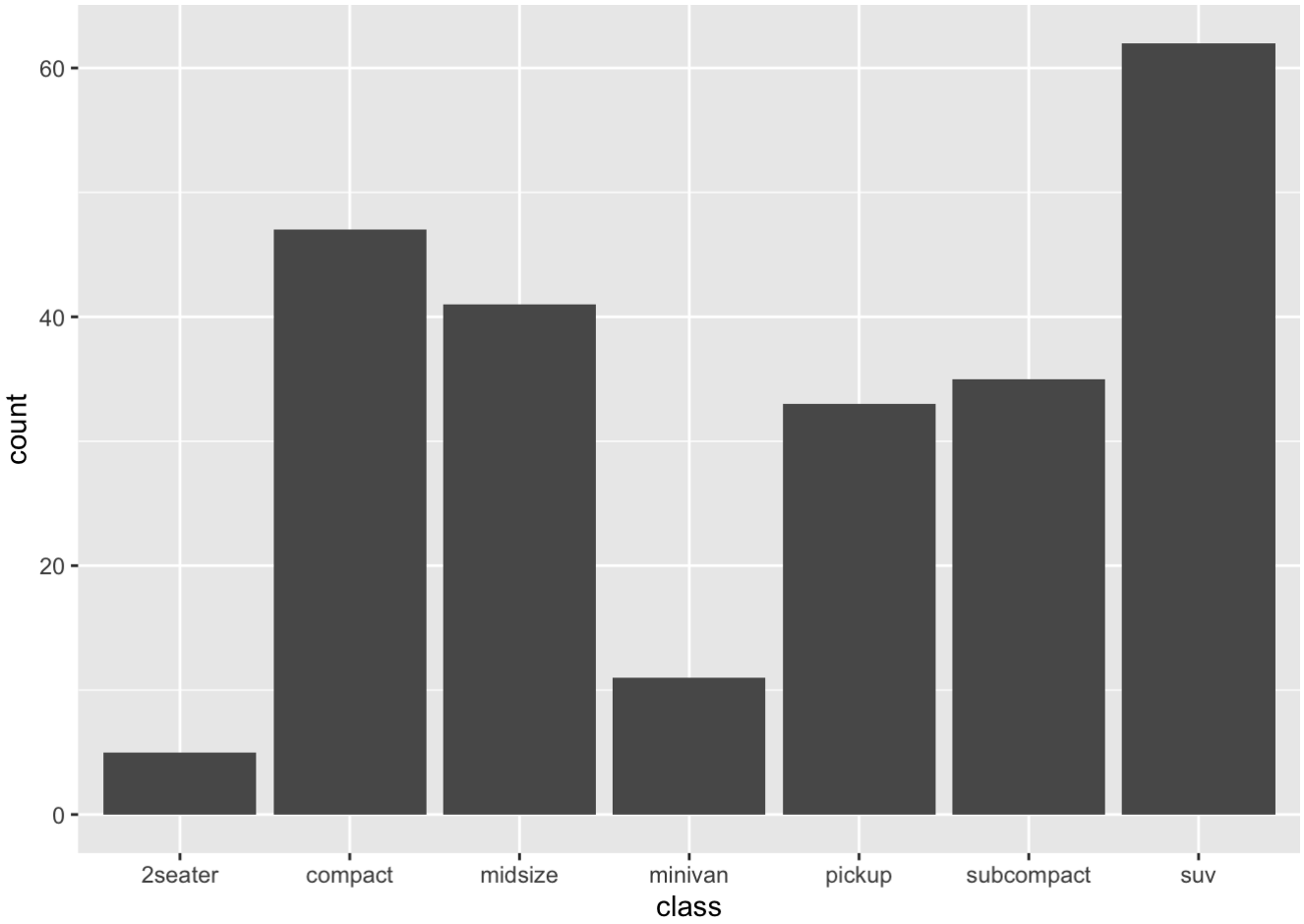
```
ggplot(faithful) +
  geom_point(aes(x = eruptions, y = waiting)) +
  geom_abline(slope = -15, intercept = 110)
```



Exercise 3.1.2: The Statistics Layer

Testing out the “Statistics” layer on R.

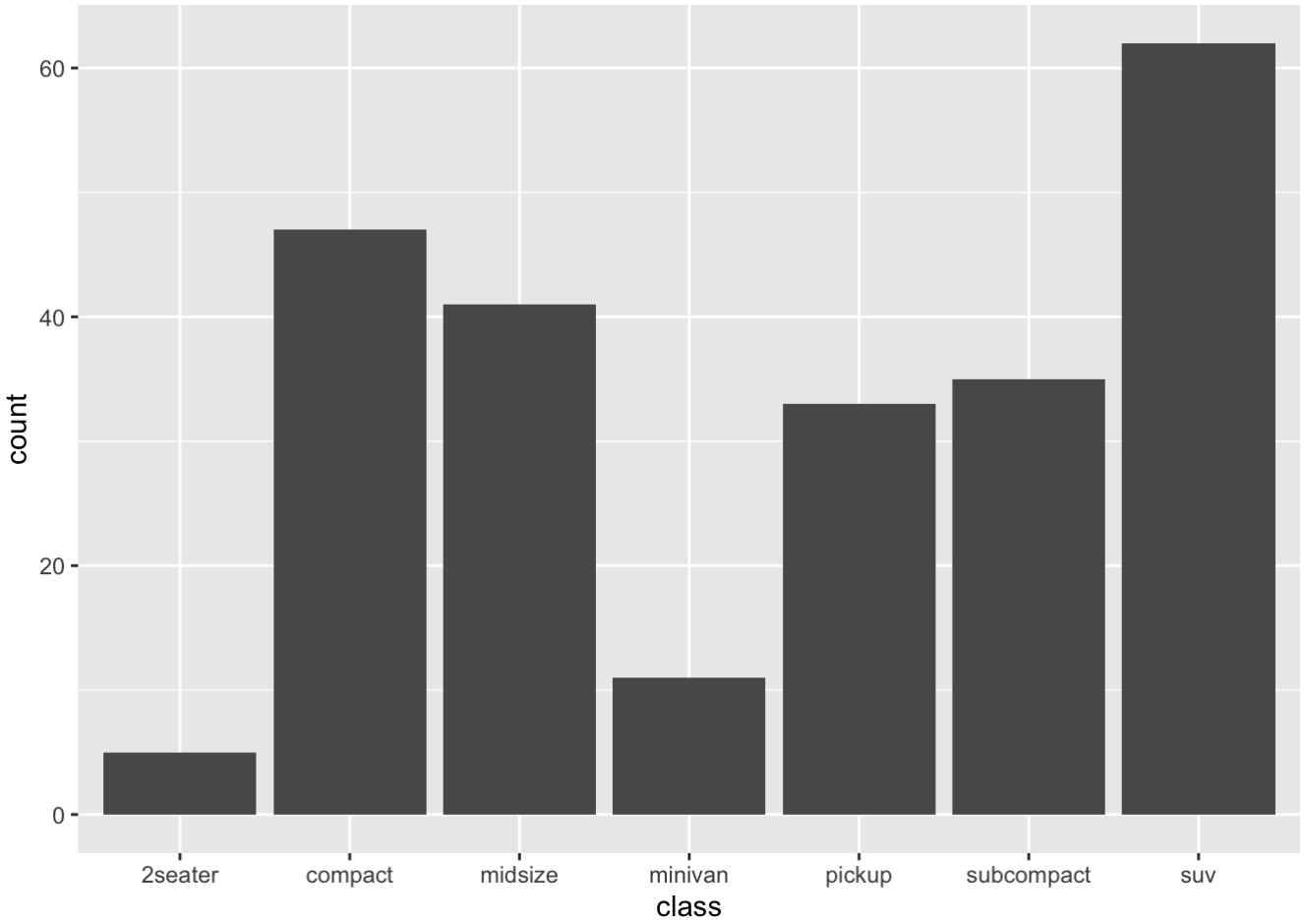
```
data("mpg")
mpg |>
  ggplot(aes(x = class)) +
  geom_bar()
```



More futzing with the data:

```
mpg_counted <- mpg |>
  group_by(class) |>
  summarize(count = n())

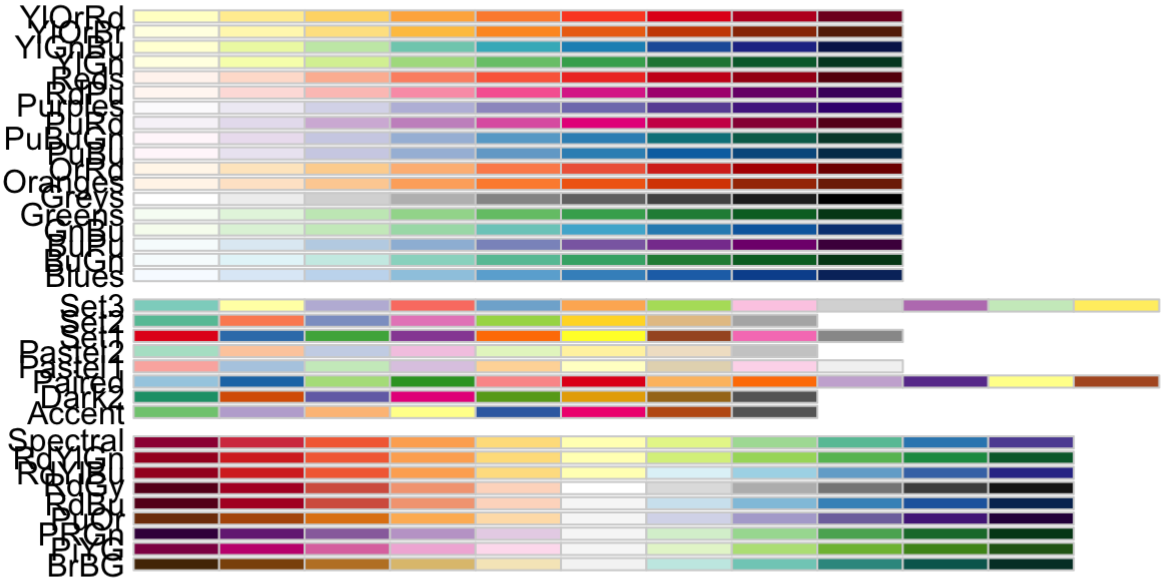
mpg_counted |>
  ggplot() +
  geom_bar(aes(x = class, y = count), stat = "identity")
```



Exercise 3.1.3.1: The Scales

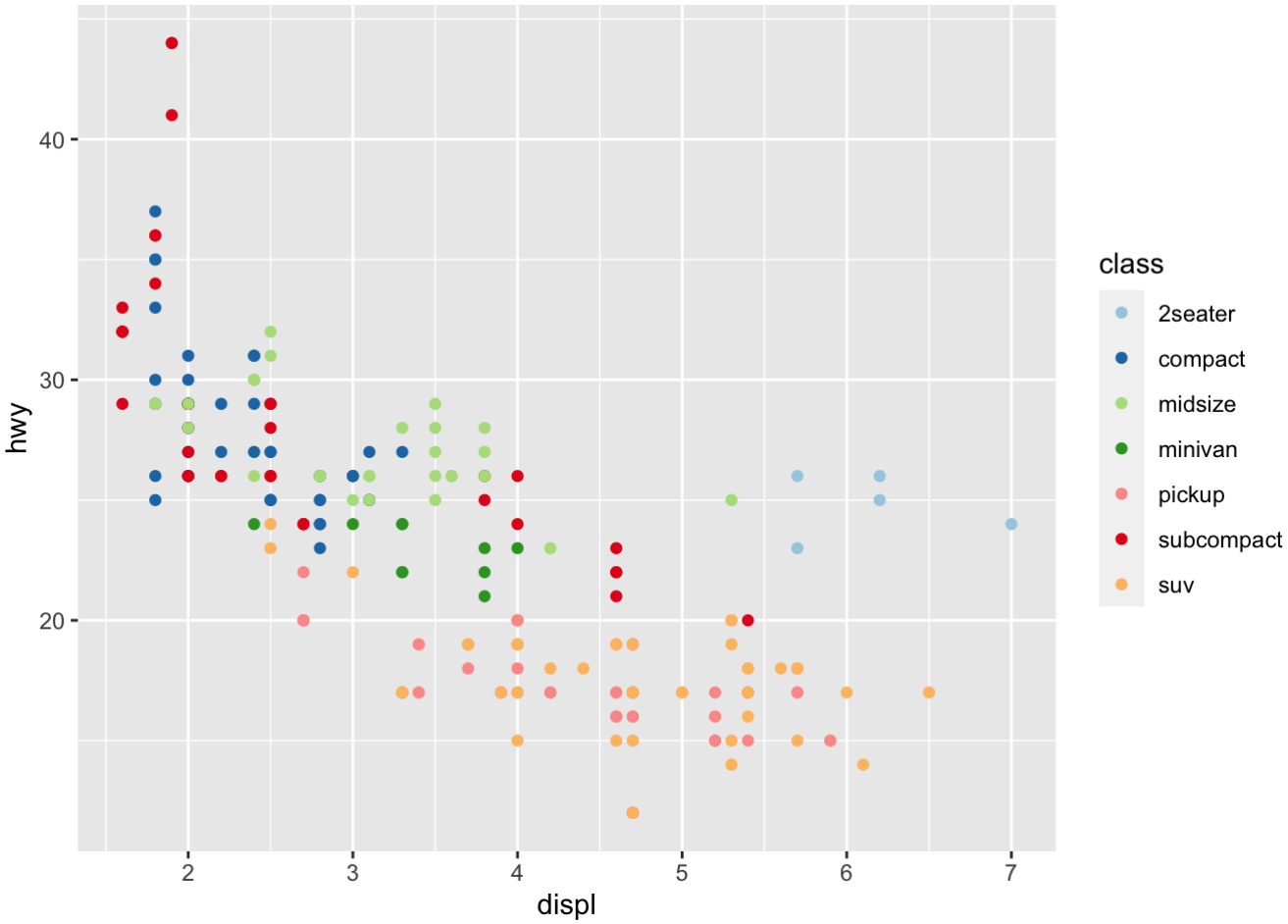
Utilizing ColorBrewer.

```
RColorBrewer::display.brewer.all()
```



My modified code, including custom color:

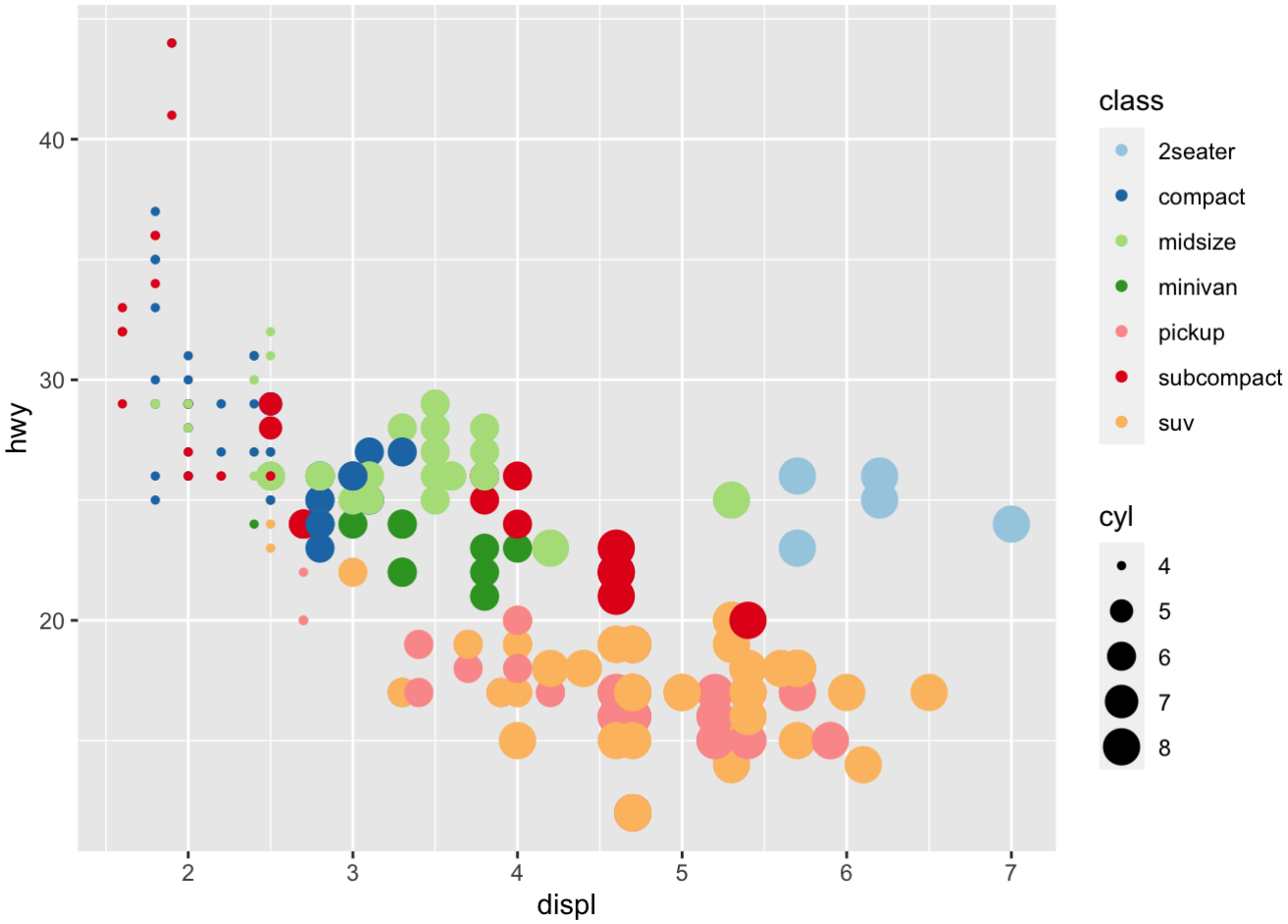
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, color = class)) +
  scale_color_brewer(type = "div", palette = "Paired")
```



Exercise 3.1.3.2

Creating a bubble chart.

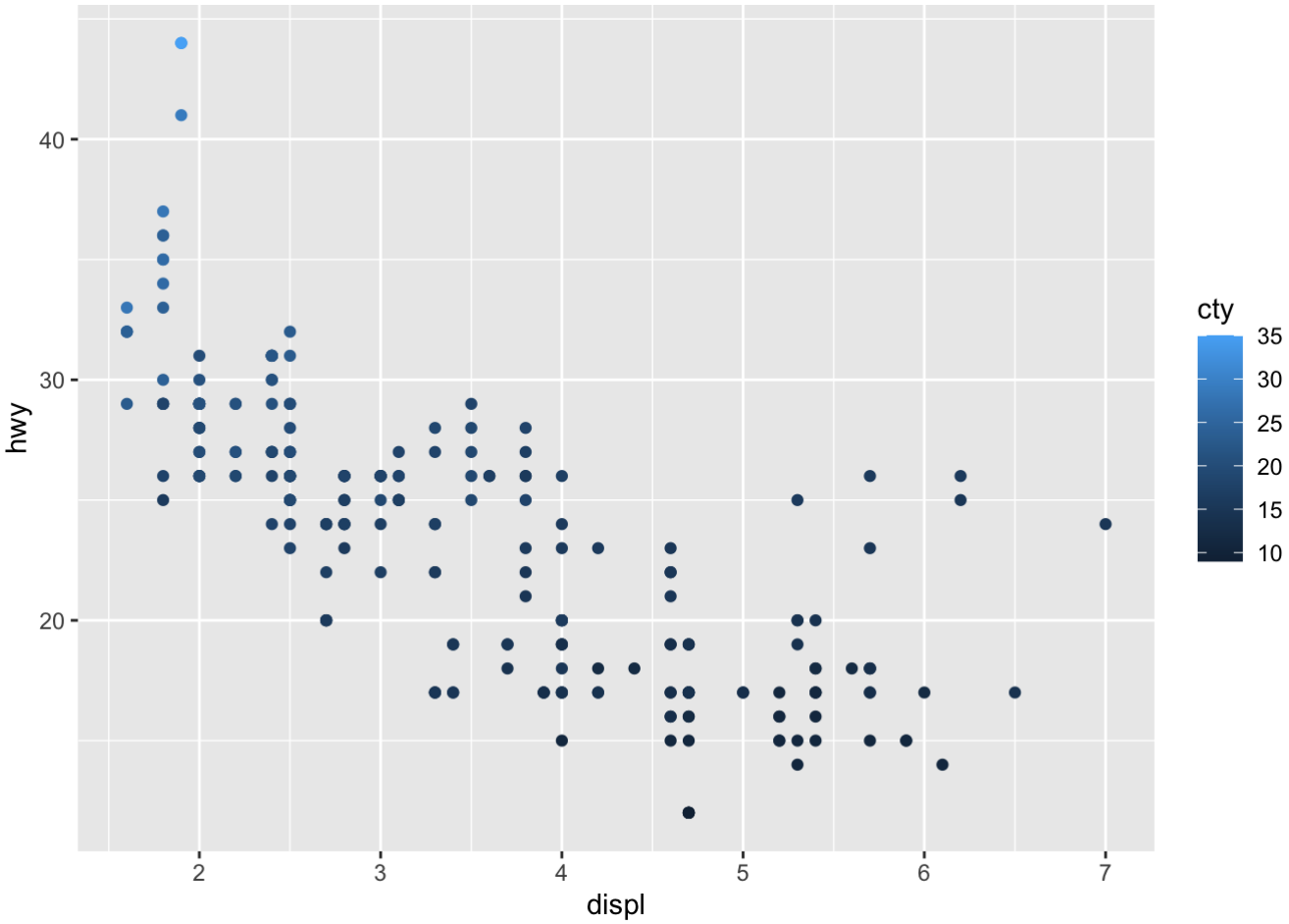
```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, color = class, size = cyl)) +
  scale_color_brewer(type = "div", palette = "Paired")
```



Exercise 3.1.3.3

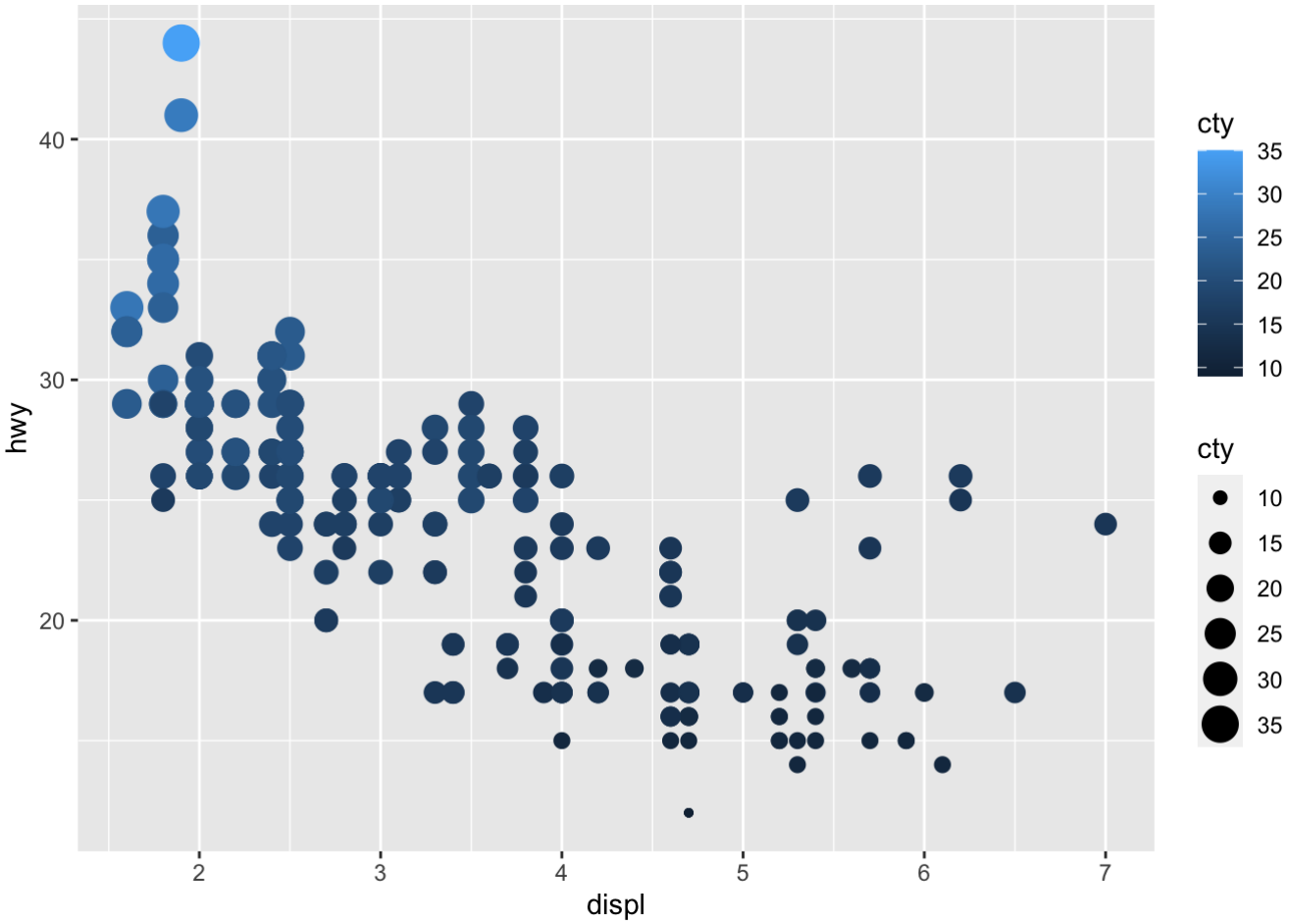
Modify the code so that color is mapped to the “cty” variable.

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, color = cty))
```



If we change the command so that “cty” is mapped onto the color variable, the guide changes from showing two different guides for “class” and “cty” to only showing one guide for “cty.” Furthermore, the aesthetic of the guide changes—now, it becomes a single-size scale for the “cty” key, whereas in the previous call, the “cyl” guide showed different sizes to indicate data on the graph.

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy, color = cty, size = cty))
```

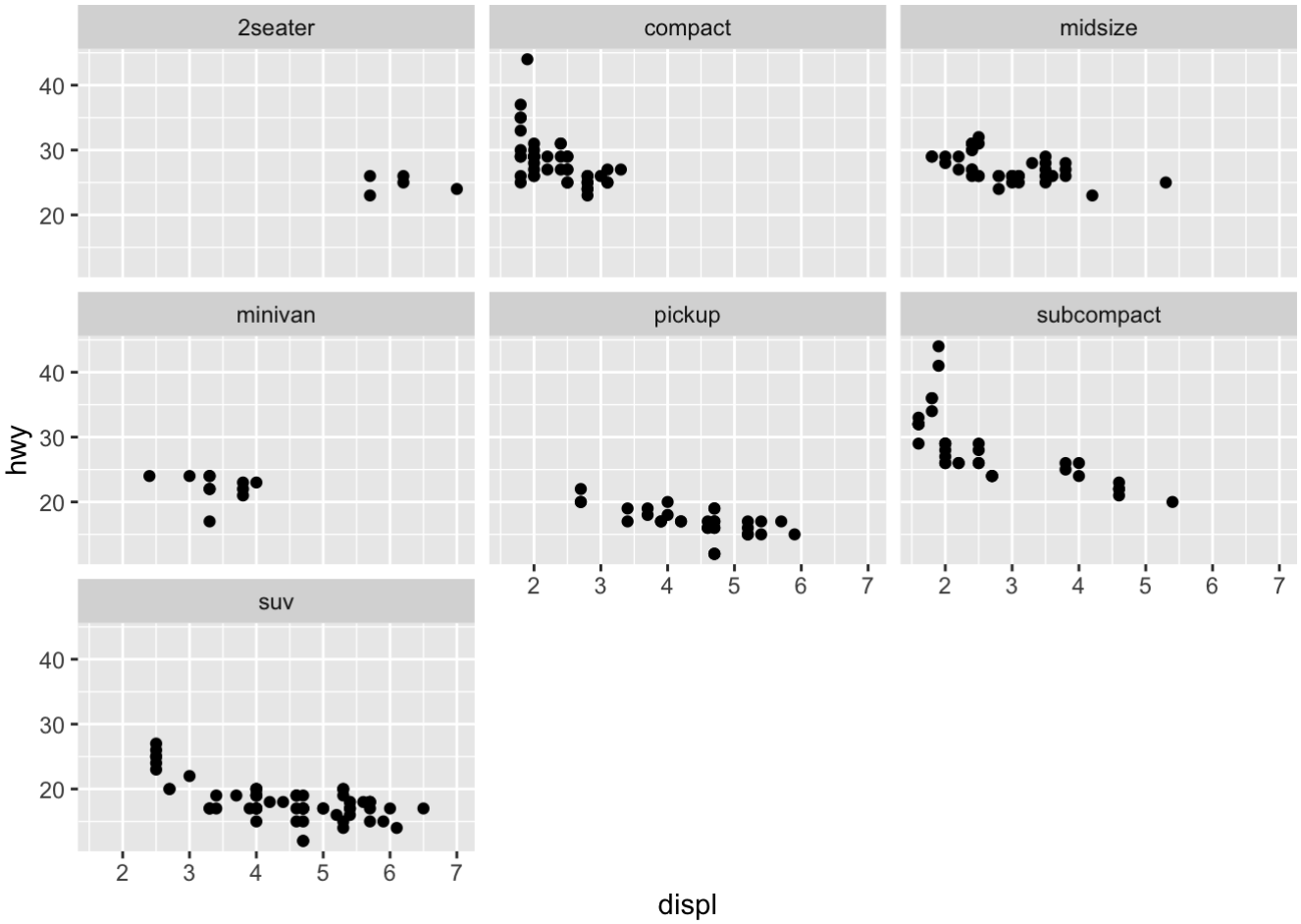


When multiple aesthetics are mapped onto the same variable, this splits the guide type into two different kinds: one, organized on a single-sized scale based on a light-to-dark color gradient, and the other, organized on a single-color scale based on increasing size gradients for plot points.

Exercise 3.1.4: Facets

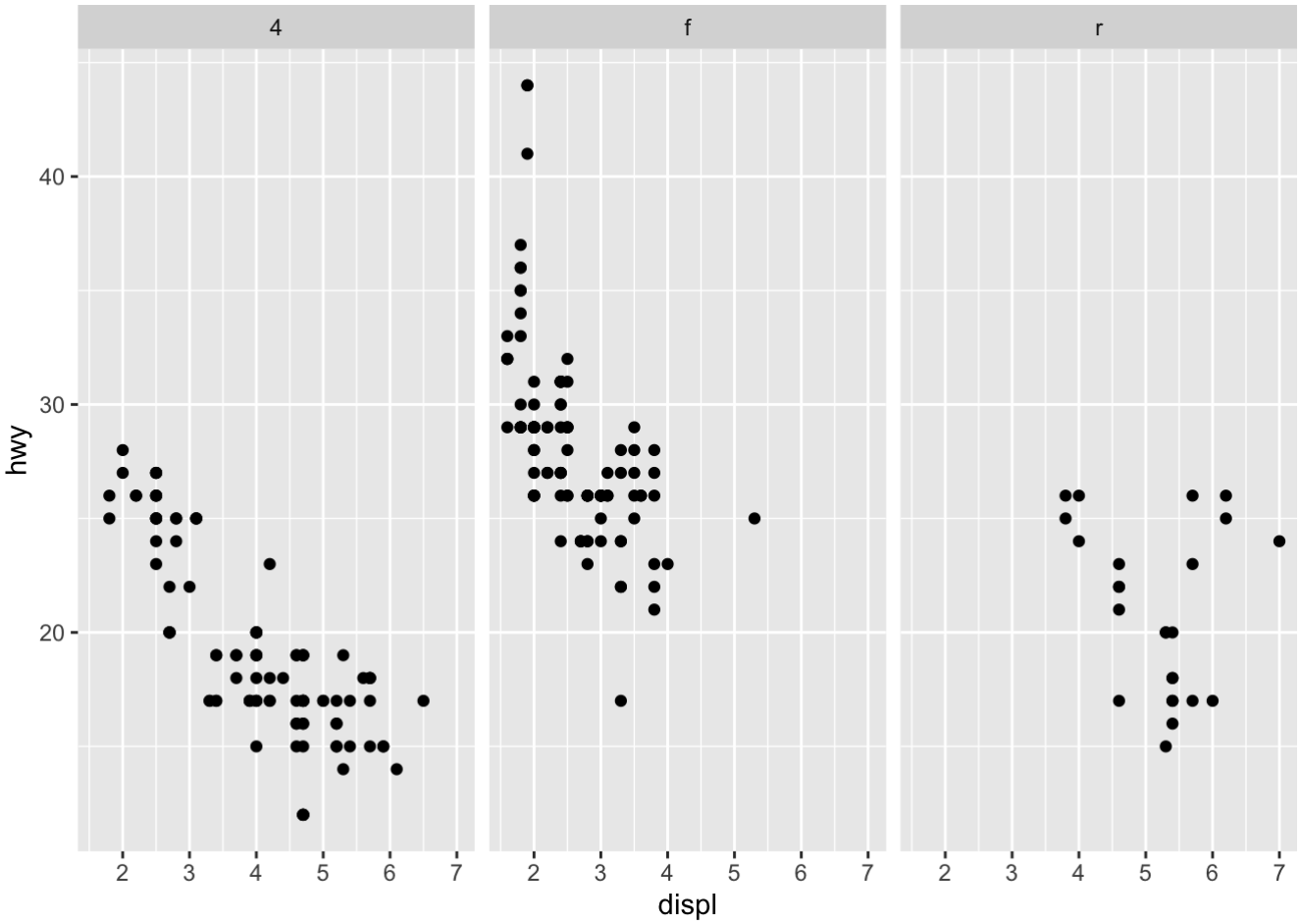
Facets!

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ class)
```



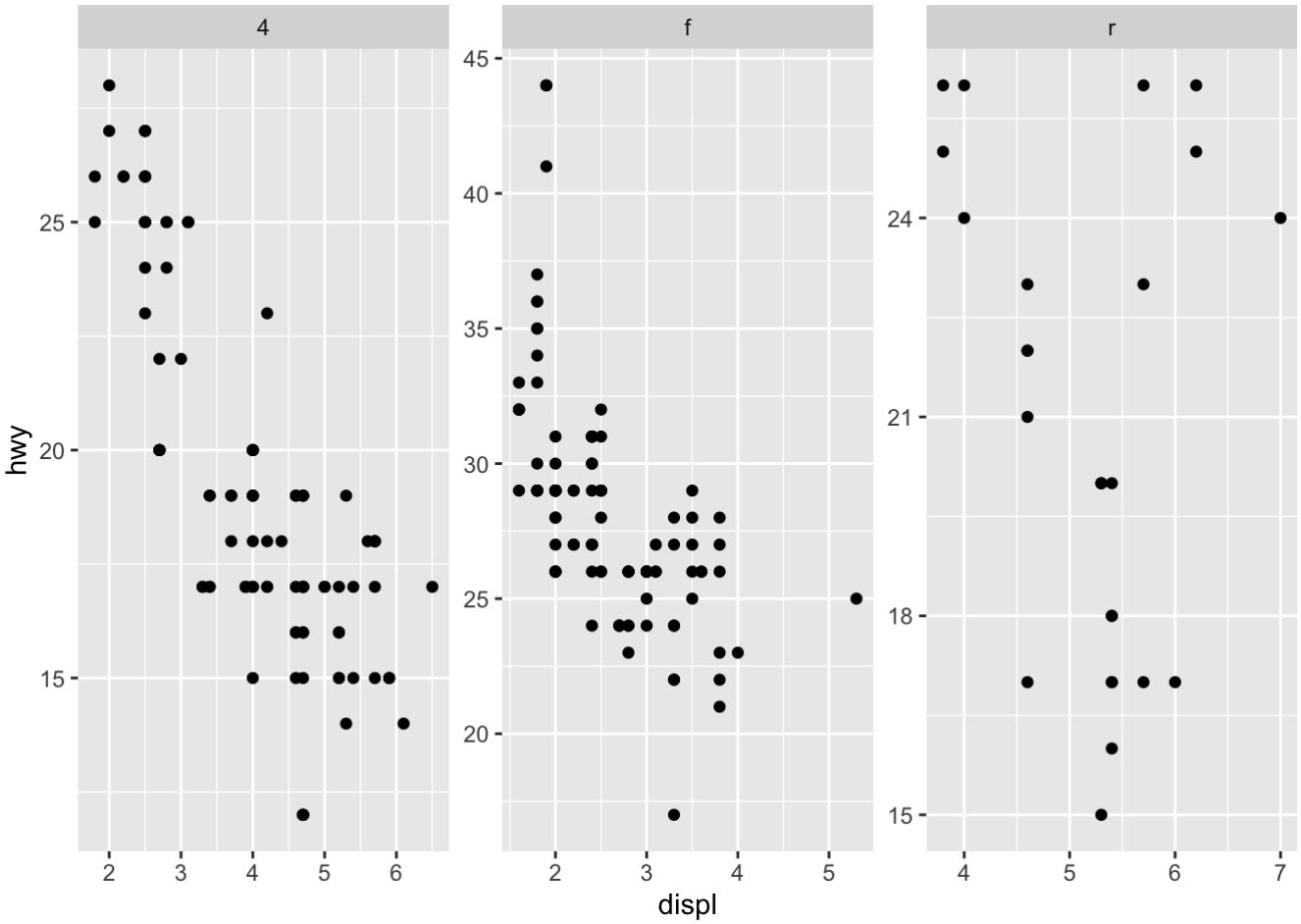
Experimenting with the facet feature (original code below):

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ drv)
```

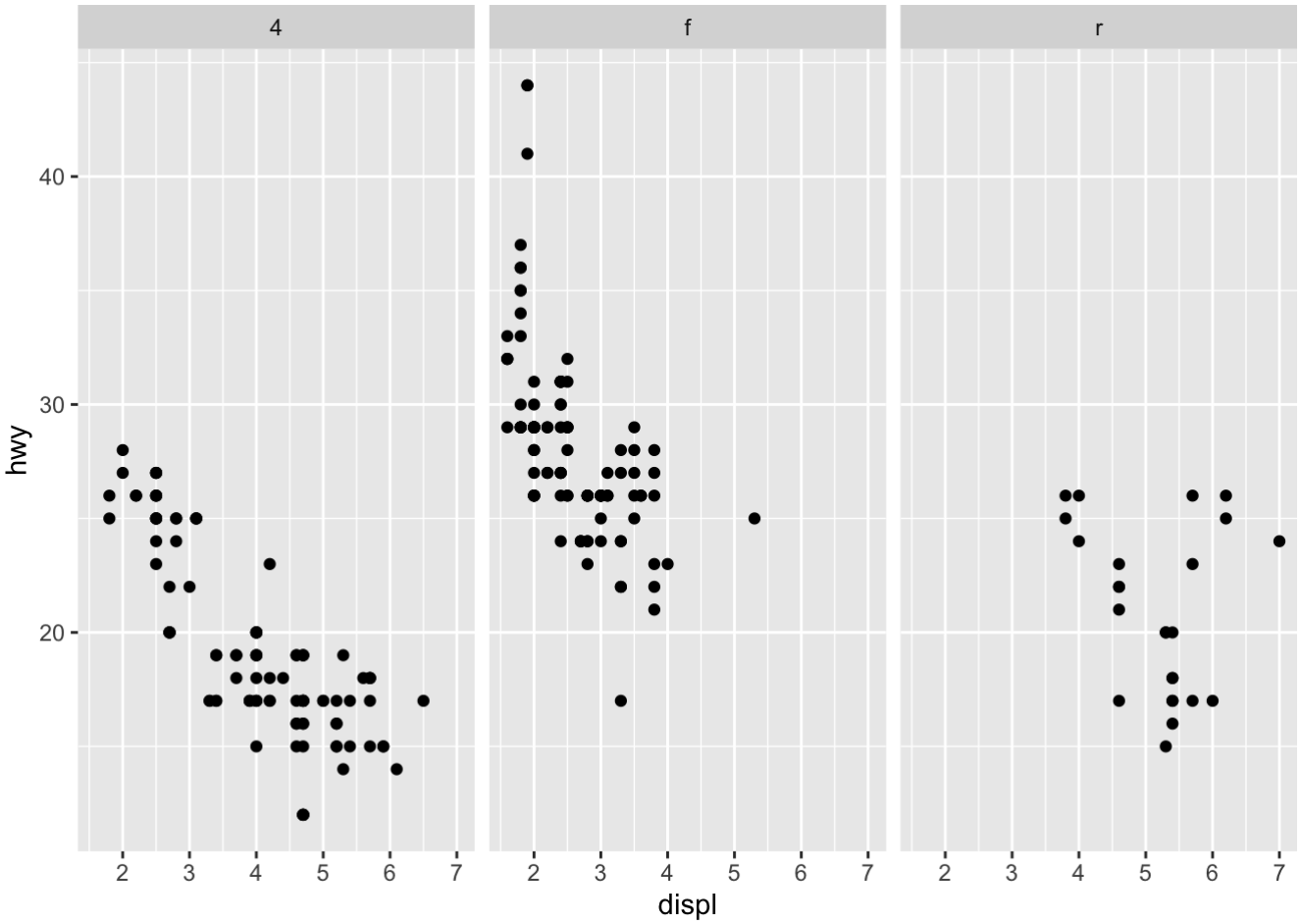


Experimenting with the facet feature:

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ drv, scales = "free")
```



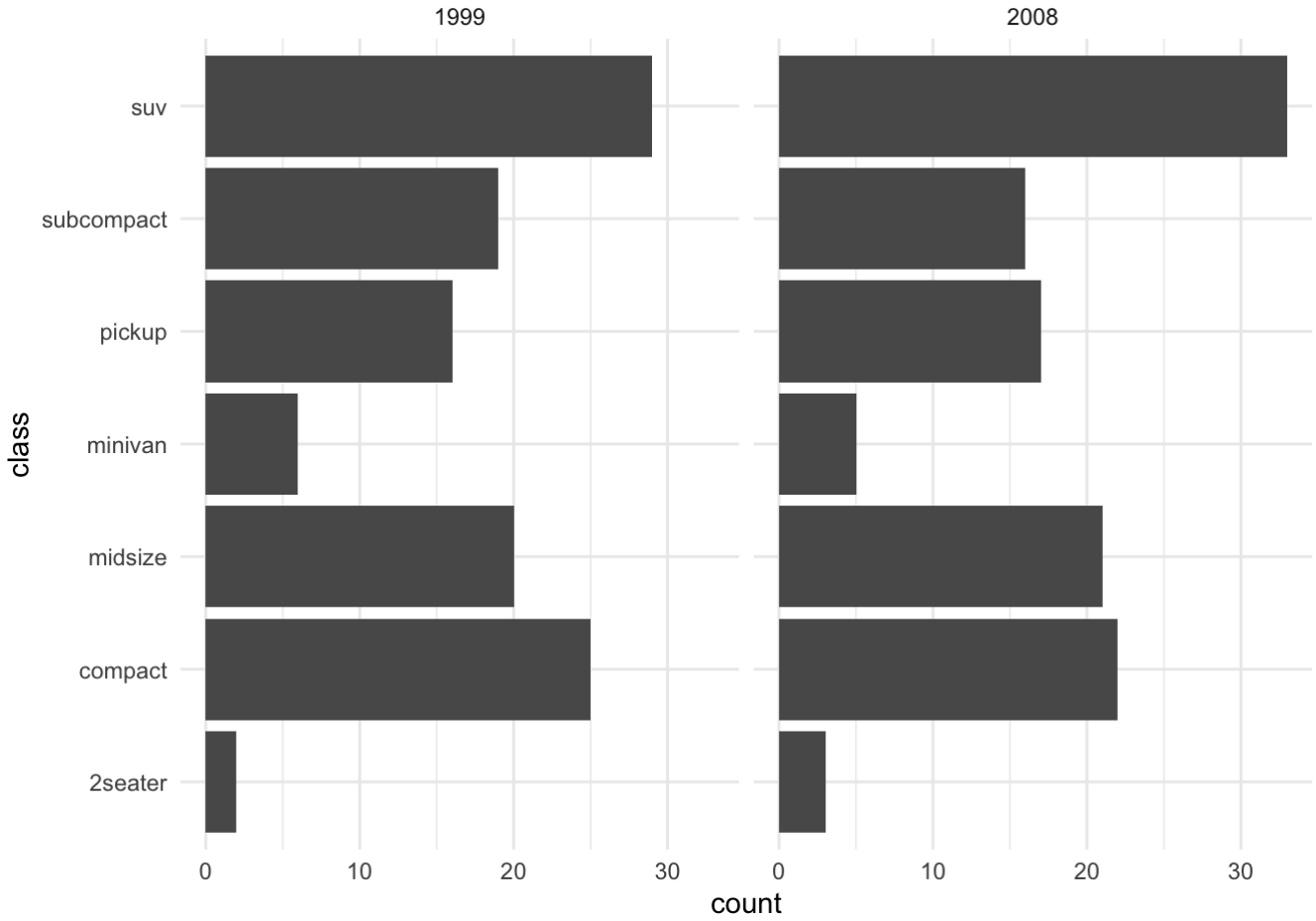

```
ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy)) +
  facet_wrap(~ drv, scales = "fixed")
```



Exercise 3.1.5: Theme

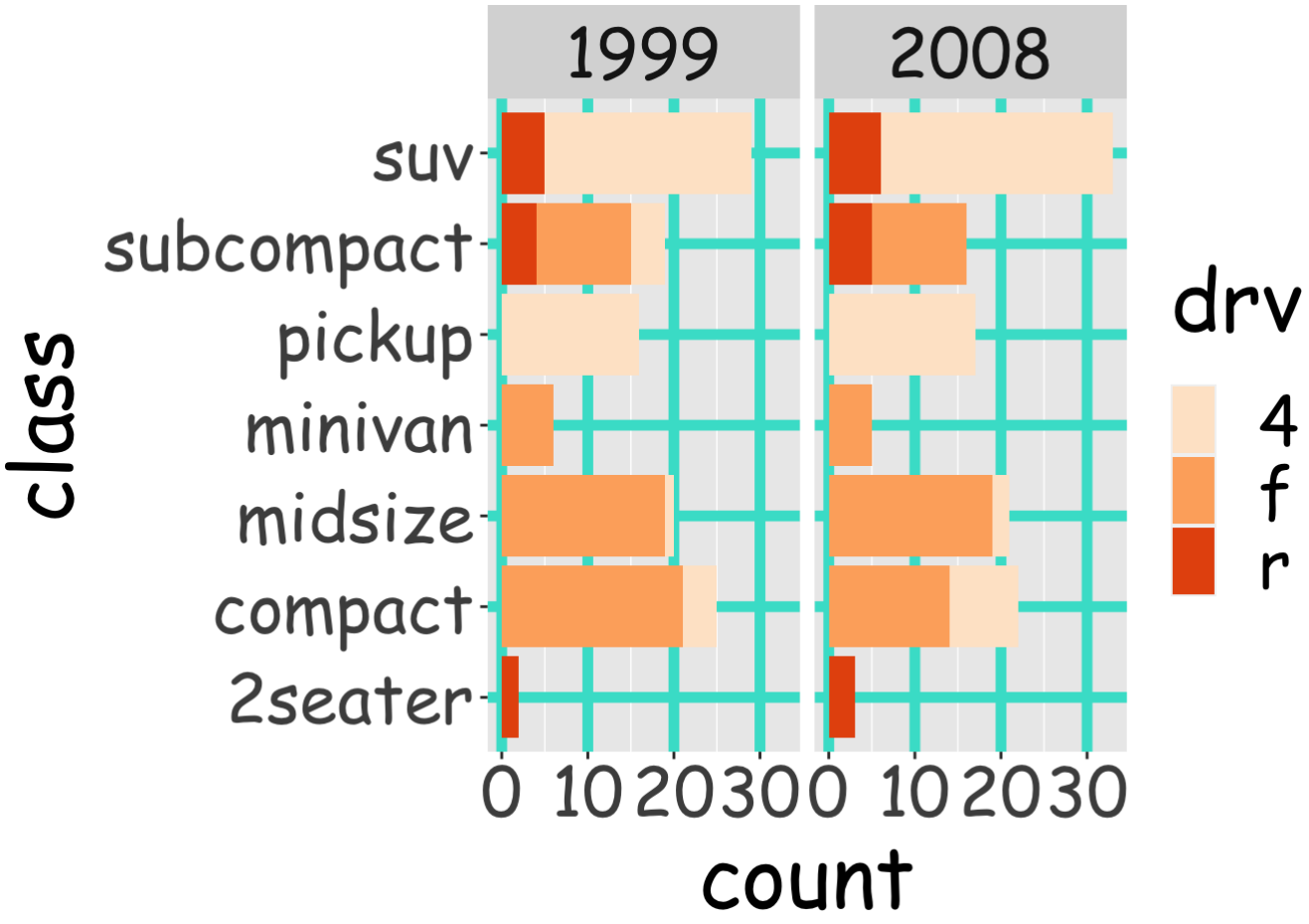
Testing out different themes. Starting with the given code:

```
ggplot(mpg) +
  geom_bar(aes(y = class)) +
  facet_wrap(~year) +
  theme_minimal()
```



Now it's time to make this shit hideous (my time to shine)!

```
mpg |>
  ggplot(aes(y = class, fill = drv)) +
  geom_bar() +
  facet_wrap(~year) +
  scale_fill_brewer(type = "seq", palette = 7) +
theme(
    text = element_text(family = "Comic Sans MS", size = 32),
    panel.grid.major = element_line(color = "turquoise", linewidth = 1.9)
  )
```




```
  labs(
    title = "Number of (Hideous) Car Models Per Class",
    caption = "source: http://fueleconomy.gov",
    x = 'Number of (Hideous) Cars',
    y = NULL
  )
```

```
$x
[1] "Number of (Hideous) Cars"

$y
NULL

$title
[1] "Number of (Hideous) Car Models Per Class"

$caption
[1] "source: http://fueleconomy.gov"

attr(,"class")
[1] "labels"
```

Exercise 3.2: Simulation

Exercise 3.2.1.1

```
::: {.cell}
```

```
:::


::: {.cell}

```{.r .cell-code}
sims <- simulation_votes(dem_prob_pop = 0.52, sample_size = 300, num_sims = 500)

results <- sims |>
  group_by(id) |>
  summarize(dem_prop = mean(vote == "Dem"))

:::

mean(results$dem_prop)

[1] 0.5201133

sd(results$dem_prop)

[1] 0.02871183
```

In the simulation above, the average dem_prop is 0.5187. The standard deviation of dem_prop is 0.0281. Different values of sample_size will lead to different levels of predictive accuracy in the results. For example, a sample size of 100 will have an accurate, but less specific average results than a sample size of 500.

Exercise 3.2.1.2

I created 5 individual simulations: sim1 (sample_size = 50), sim2 (sample_size = 200), sim3 (sample_size = 500), kevin (sample_size = 1000), and sim5 (sample_size = 2000).

```
sim1 <- simulation_votes(dem_prob_pop = 0.52, sample_size = 50, num_sims = 500)

sim2 <- simulation_votes(dem_prob_pop = 0.52, sample_size = 200, num_sims = 500)

sim3 <- simulation_votes(dem_prob_pop = 0.52, sample_size = 500, num_sims = 500)

kevin <- simulation_votes(dem_prob_pop = 0.52, sample_size = 1000, num_sims = 500)

sim4 <- simulation_votes(dem_prob_pop = 0.52, sample_size = 2000, num_sims = 500)
```

In order to put them together into one data set:

```
results1 <- sim1 |>
  group_by(id, sample_size) |>
  summarize(dem_prop = mean(vote == "Dem"))

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

results2 <- sim2 |>
  group_by(id, sample_size) |>
  summarize(dem_prop = mean(vote == "Dem"))

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

results3 <- sim3 |>
  group_by(id, sample_size) |>
```
```

```
  summarize (dem_prop = mean(vote == "Dem"))
```

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

```
resultskevin <- kevin |>
  group_by(id, sample_size) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

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

```
results4 <- sim4 |>
  group_by(id, sample_size) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

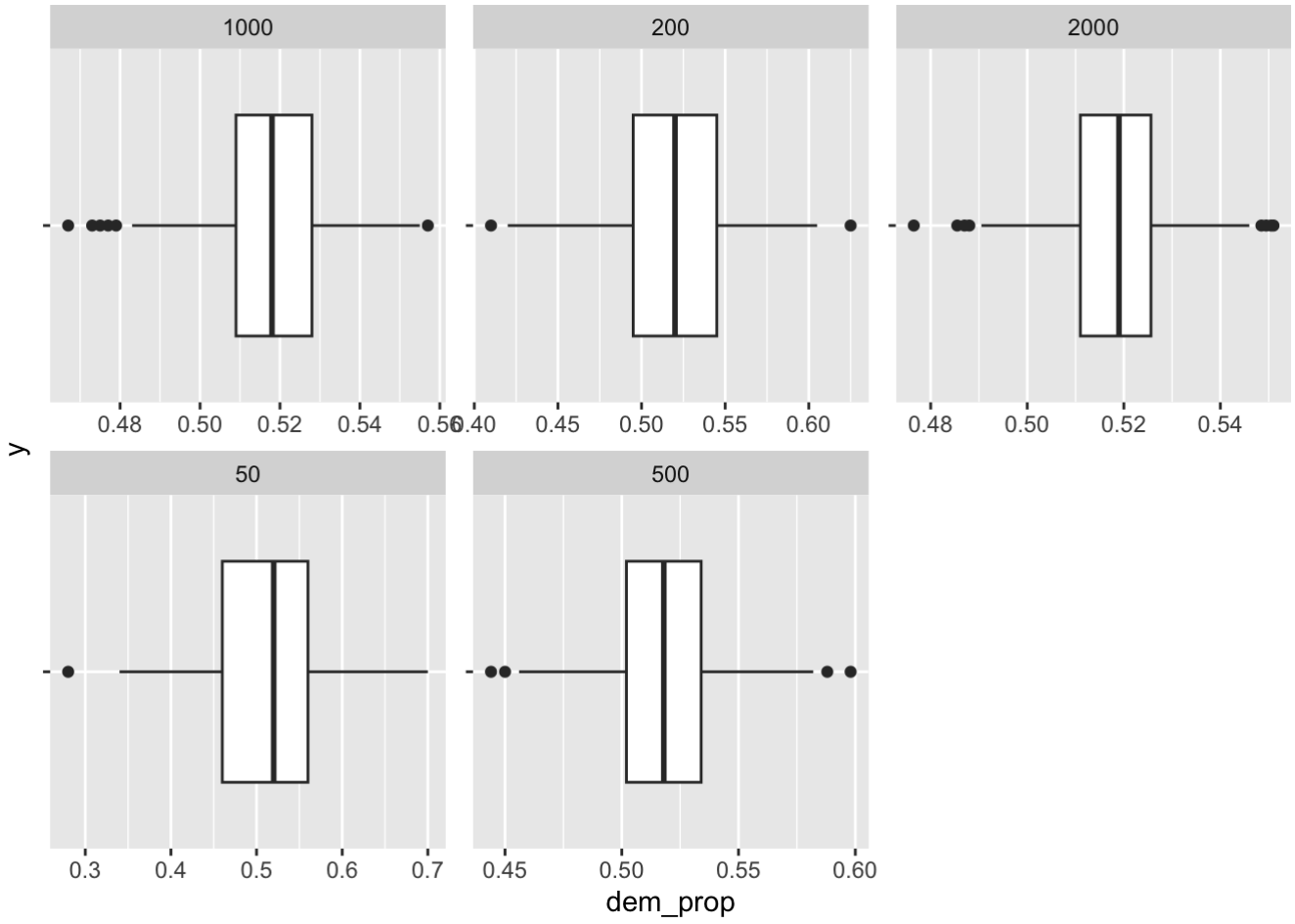
```
`summarise()` has grouped output by 'id'. You can override using the `.groups`
argument.
```

NOW put them all together:

```
resultsall <- bind_rows(results1, results2, results3, resultskevin, results4)
```

Running the boxplots:

```
ggplot(resultsall, aes(dem_prop, "")) +
  geom_boxplot() +
  facet_wrap(~ sample_size, scales = "free")
```



What we're seeing here is that, as the sample size gets larger, the interquartile range of the data gets smaller, meaning that the data is more precise in its findings.

Exercise 3.2.1.3

Putting together five different simulations with different values of dem_prob_pop:

```
pop49 <- simulation_votes(dem_prob_pop = 0.49, sample_size = 50, num_sims = 500)
```

```
pop52 <- simulation_votes(dem_prob_pop = 0.52, sample_size = 50, num_sims = 500)
```

```
pop55 <- simulation_votes(dem_prob_pop = 0.55, sample_size = 50, num_sims = 500)
```

```
pop58 <- simulation_votes(dem_prob_pop = 0.58, sample_size = 50, num_sims = 500)
```

```
pop60 <- simulation_votes(dem_prob_pop = 0.60, sample_size = 50, num_sims = 500)
```

Now code them for results:

```
results49 <- pop49 |>
  group_by(id, dem_prob_pop) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

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

```
results52 <- pop52 |>
  group_by(id, dem_prob_pop) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

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

```
results55 <- pop55 |>
  group_by(id, dem_prob_pop) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

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

```
results58 <- pop58 |>
  group_by(id, dem_prob_pop) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

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

```
results60 <- pop60 |>
  group_by(id, dem_prob_pop) |>
  summarize(dem_prop = mean(vote == "Dem"))
```

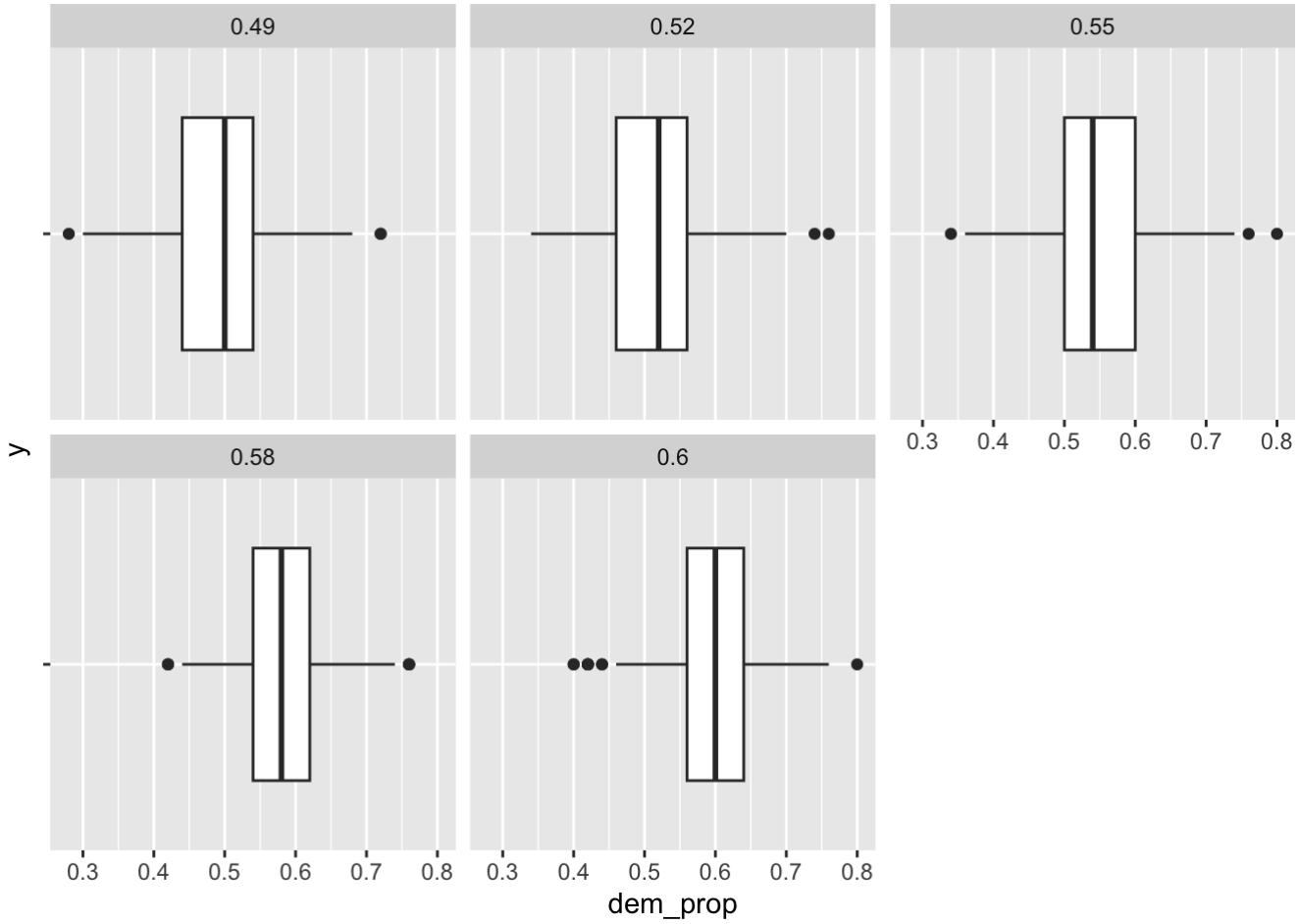
```
`summarise()` has grouped output by 'id'. You can override using the `.groups`
argument.
```

Putting them all together into one big frame:

```
results_all_pop <- bind_rows(results49, results52, results55, results58, results60)
```

Running the boxplots:

```
ggplot(results_all_pop, aes(dem_prop, "")) +
  geom_boxplot() +
  facet_wrap(~ dem_prob_pop, scales = "fixed")
```



These box plots all have means close to one another and close comparable distance between their interquartile ranges.