Visualizing uncertainty

Daniel Anderson
Week 7, Class 1

UNIVERSITY OF OREGON

Agenda

- Common ways of visualizing uncertainty
  - And how to implement them with {ggplot2}
- Framing uncertainty as relative frequencies
  - Discrete probabilities
  - Non-discrete probabilities
  - Understanding AUC calculations
- Understanding standard errors
  - Non-standard ways of visualizing SEs
- HOPs (briefly)
  - Also bootstrapping

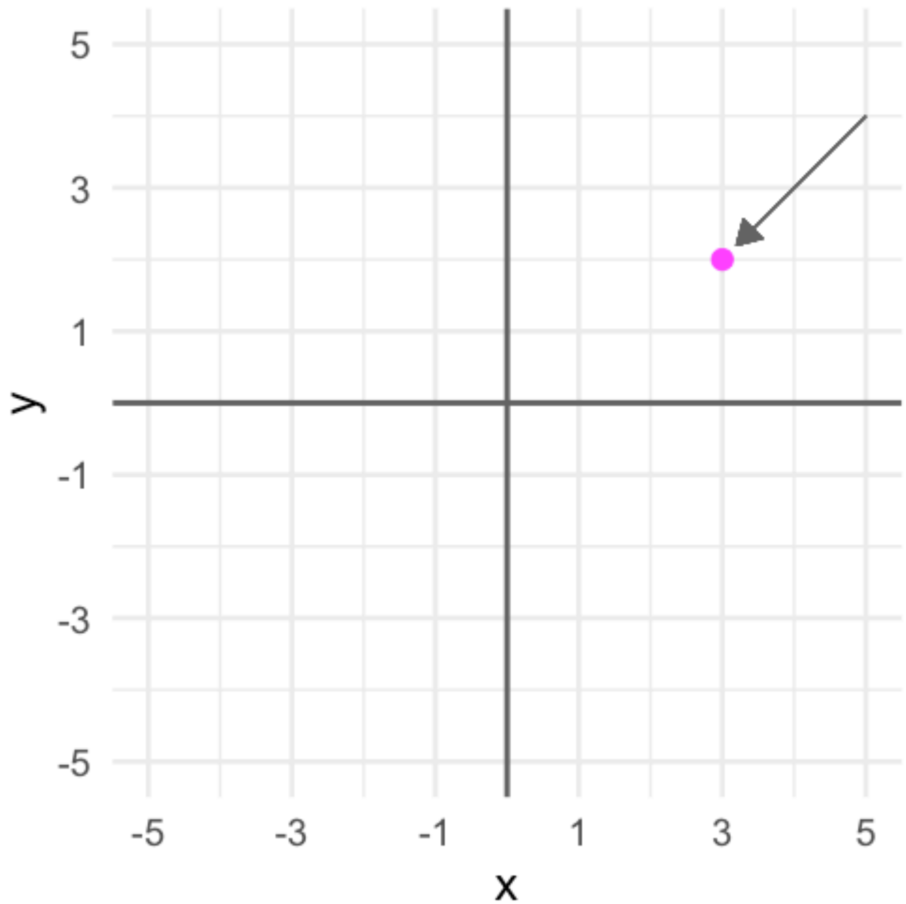
Learning objectives

1. Understand there are lots of different ways to visualize uncertainty, and the best method may often be non-standard.

2. Understand how to implement basic methods, and the resources available to you to implement more advanced methods

The primary problem

- When we see a point on a plot, we interpret it as **THE** value.



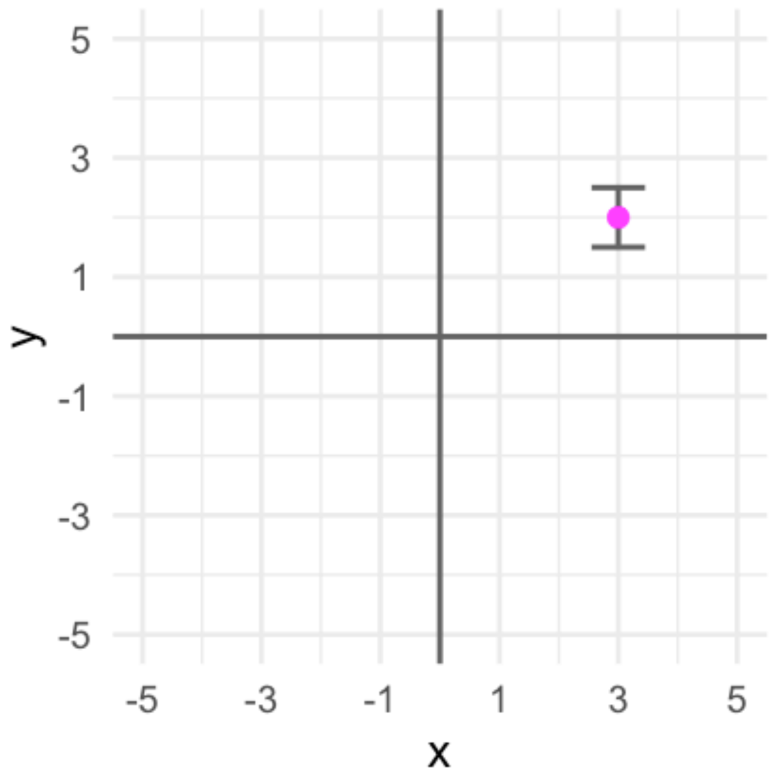
Let's have Dr. Kay explain

Some secondary problem

- We're not great at understanding probabilities
- We regularly round probabilities to 100% or 0%
- As probabilities move to the tails, we're generally worse

How do we typically communicate uncertainty?

- Error bars



How?

Vertical error bars

`geom_errorbar`

- Requires `ymin` and `ymax` aesthetics
- You have to supply these - no calculation for you

Horizontal error bars

`geom_errorbarh`

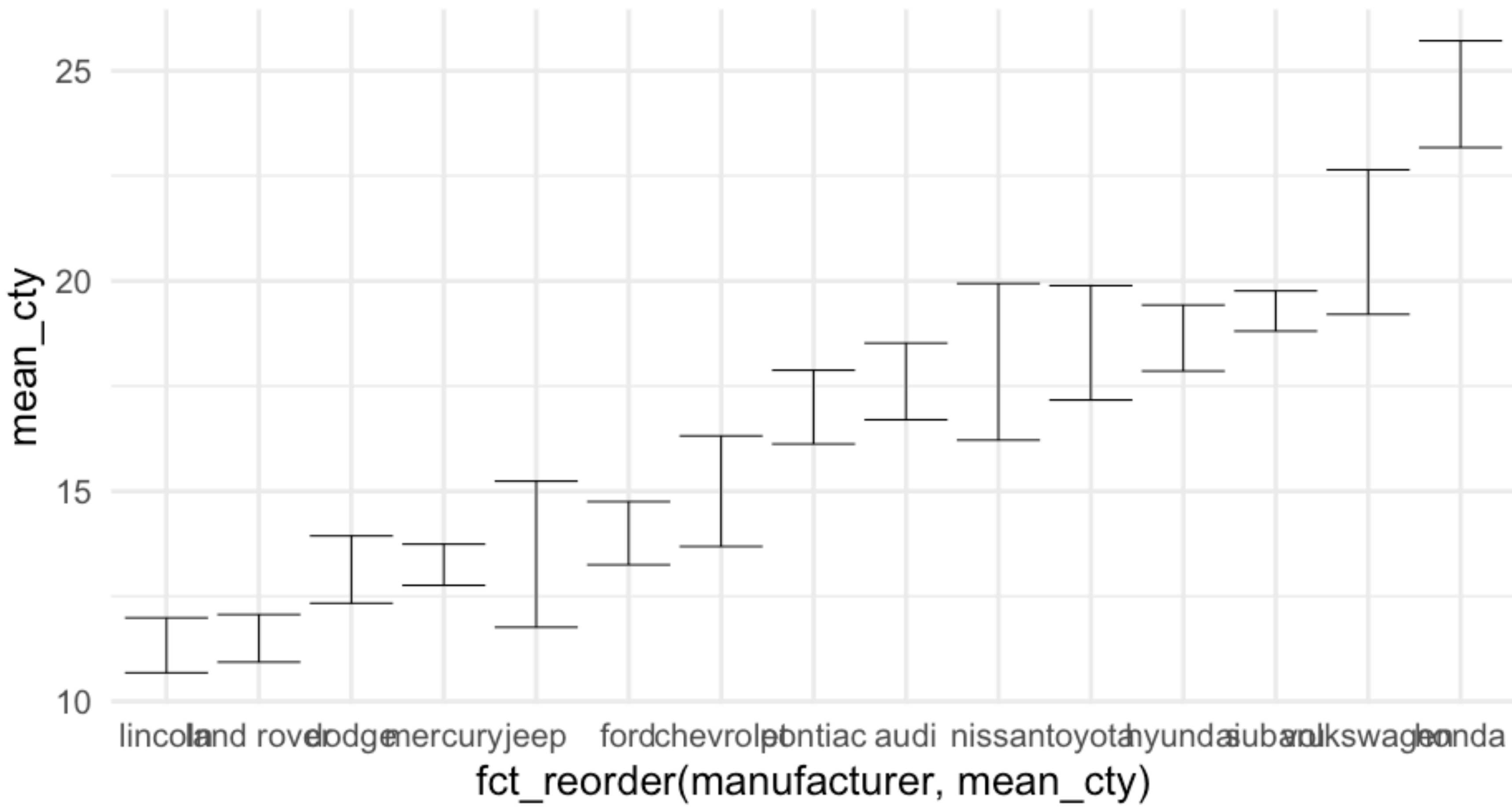
- Requires `xmin` and `xmax`

Example

```
mpg_by_man <- mpg %>%
  group_by(manufacturer) %>%
  summarize(mean_cty = mean(cty),
            se_cty = sundry::se(cty))

head(mpg_by_man)
```

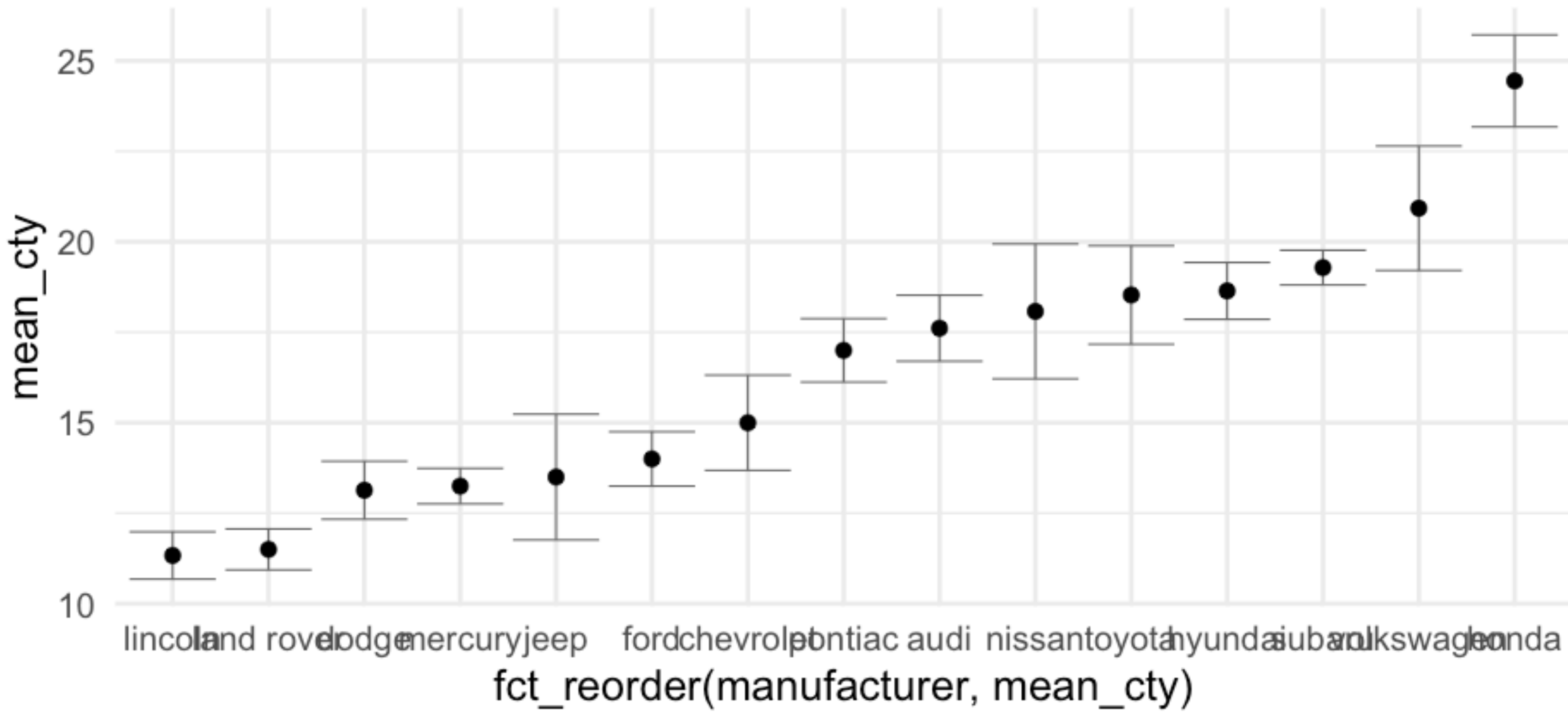
```
## # A tibble: 6 x 3
##   manufacturer mean_cty    se_cty
##   <chr>           <dbl>     <dbl>
## 1 audi         17.61111 0.4653967
## 2 chevrolet    15       0.6710383
## 3 dodge        13.13514 0.4085464
## 4 ford         14       0.3829708
## 5 honda        24.44444 0.6478835
## 6 hyundai      18.64286 0.4006470
```

```
ggplot(mpg_by_man, aes(fct_reorder(manufacturer, mean_cty), mean_cty)) +
  geom_errorbar(aes(ymin = mean_cty + qnorm(0.025)*se_cty,
                    ymax = mean_cty + qnorm(0.975)*se_cty))
```

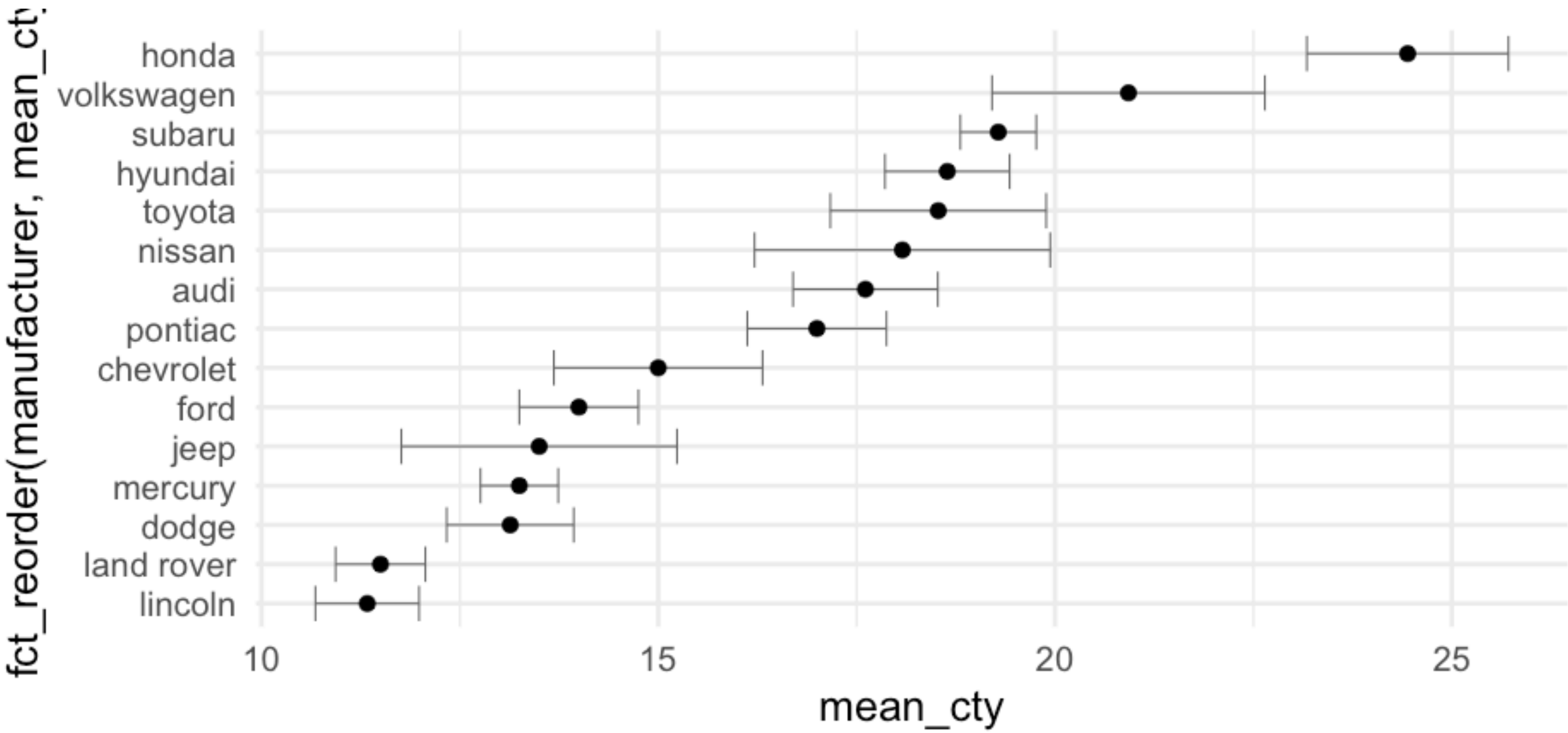


Put points on top (not under)

```
ggplot(mpg_by_man, aes(fct_reorder(manufacturer, mean_cty), mean_cty)) +
  geom_errorbar(aes(ymin = mean_cty + qnorm(0.025)*se_cty,
                    ymax = mean_cty + qnorm(0.975)*se_cty),
                color = "gray40") +
   geom_point()
```



```
ggplot(mpg_by_man, aes(fct_reorder(manufacturer, mean_cty), mean_cty)) +
  geom_errorbar(aes(ymin = mean_cty - 1.96*se_cty,
                    ymax = mean_cty + 1.96*se_cty),
                color = "gray40") +
   geom_point() +
   coord_flip()
```

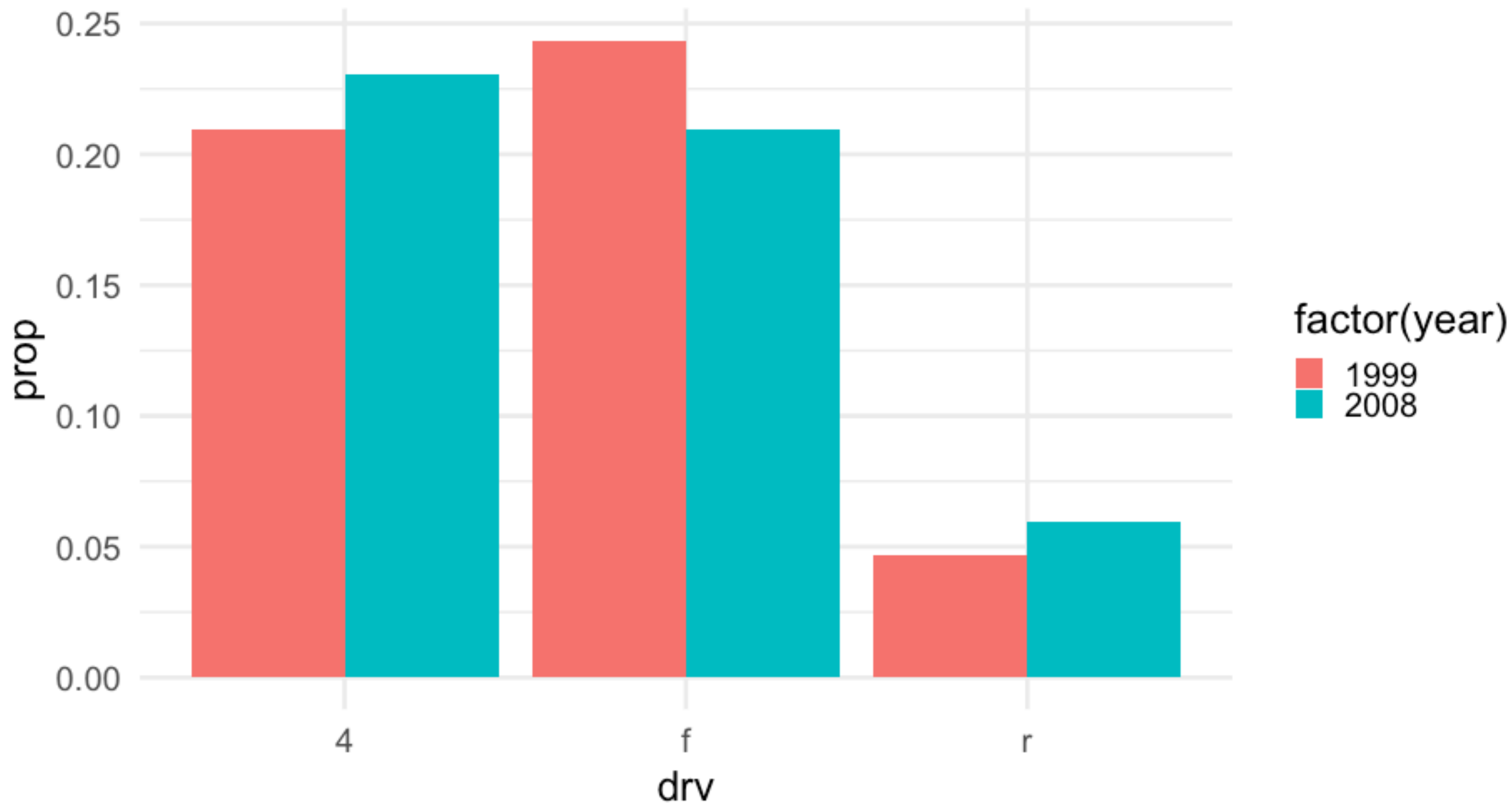


Dodging

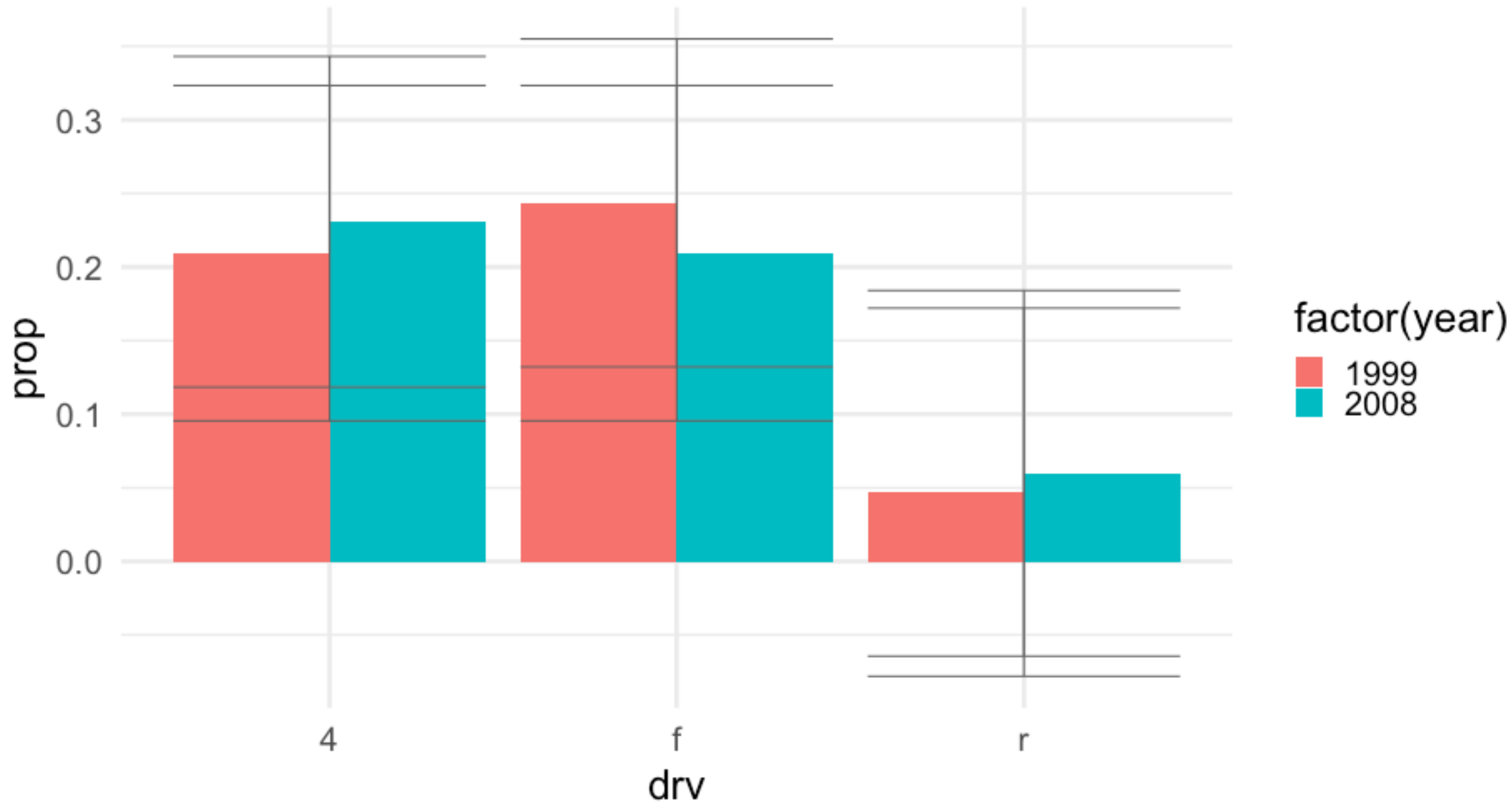
```
props <- mpg %>%
  count(drv, year) %>%
  mutate(prop = n/sum(n),
         prop_se = sqrt((prop*(1-prop)) / n))

head(props)
```

```
## # A tibble: 6 x 5
##   drv    year     n       prop    prop_se
##   <chr> <int> <int>      <dbl>      <dbl>
## 1 4      1999    49 0.2094017  0.05812594
## 2 4      2008    54 0.2307692  0.05733508
## 3 f      1999    57 0.2435897  0.05685528
## 4 f      2008    49 0.2094017  0.05812594
## 5 r      1999    11 0.04700855 0.06381703
## 6 r      2008    14 0.05982906 0.06338631
```

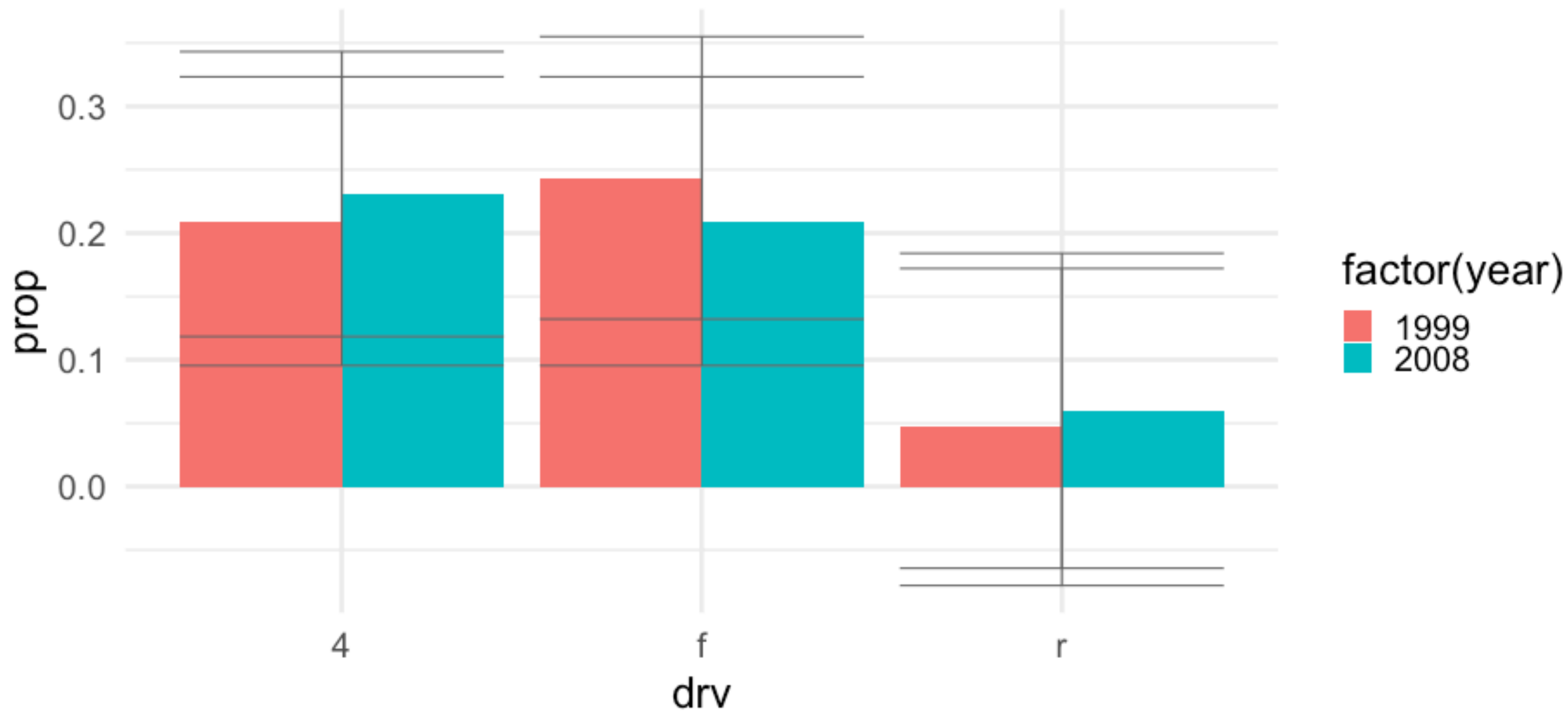
```r
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = "dodge")
```



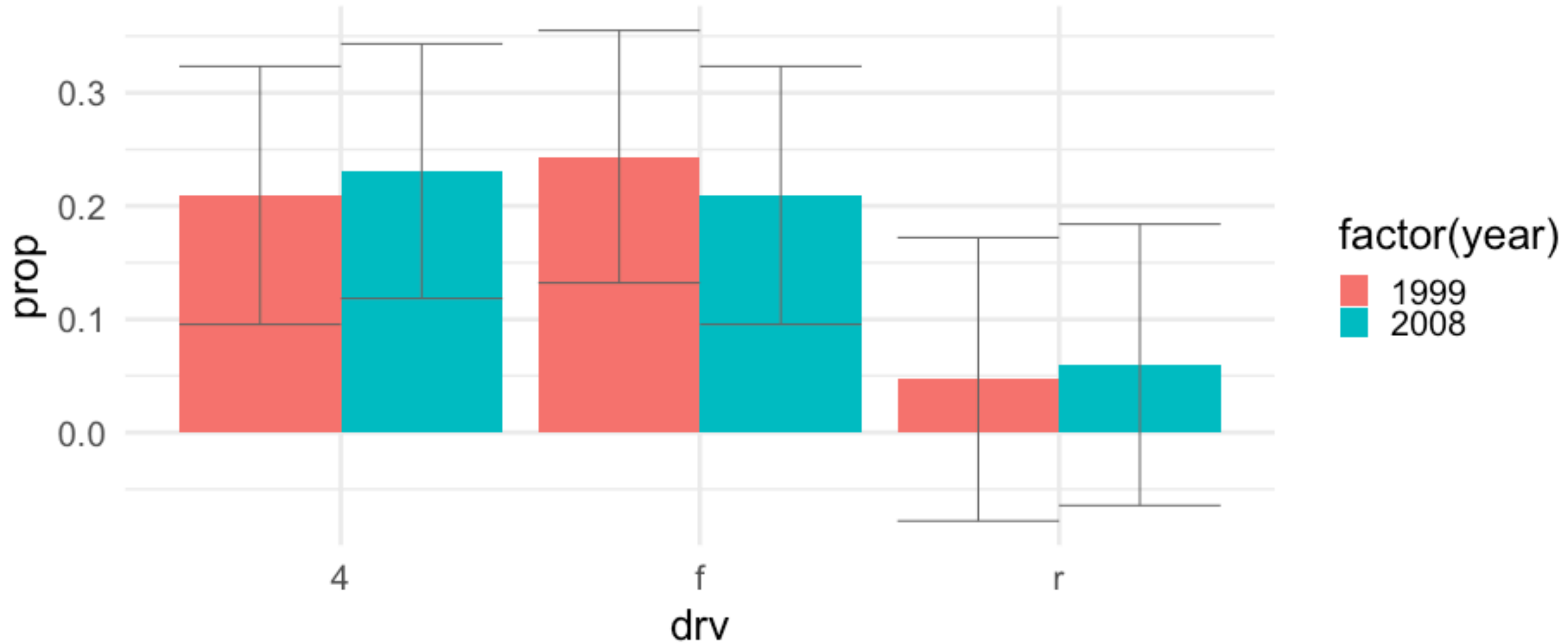
```r
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = "dodge") +
  geom_errorbar(aes(ymin = prop - 1.96*prop_se,
                    ymax = prop + 1.96*prop_se),
                color = "gray40")
```



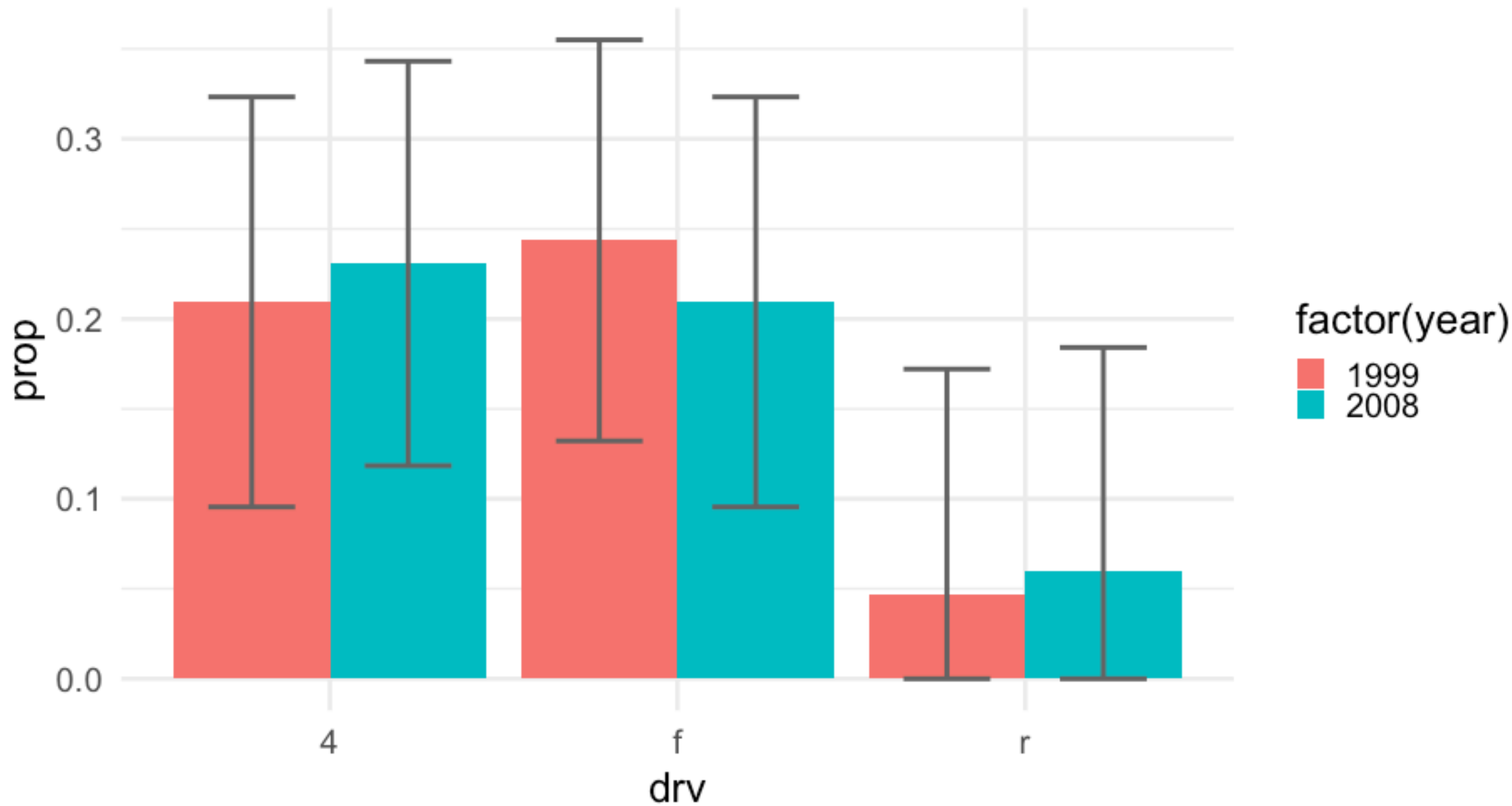
```
pd <- position_dodge(.9)
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = pd) +
  geom_errorbar(aes(ymin = prop - 1.96*prop_se,
                    ymax = prop + 1.96*prop_se),
                color = "gray40",
                position = pd)
```



```r
pd <- position_dodge(.9)
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = pd) +
  geom_errorbar(aes(ymin = prop - 1.96*prop_se,
                    ymax = prop + 1.96*prop_se,
                    group = year),
                color = "gray40",
                position = pd)
```



```r
pd <- position_dodge(.9)
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = pd) +
  geom_errorbar(aes(ymin = ifelse(prop - 1.96*prop_se < 0,
                                  0,
                                  prop - 1.96*prop_se),
                    ymax = prop + 1.96*prop_se,
                    group = year),
                color = "gray40",
                position = pd,
                width = 0.5,
                size = 1.4)
```


0.3
0.2
0.1
0.0
prop
4
f
r
drv
factor(year)
1999
2008

Thinking about uncertainty

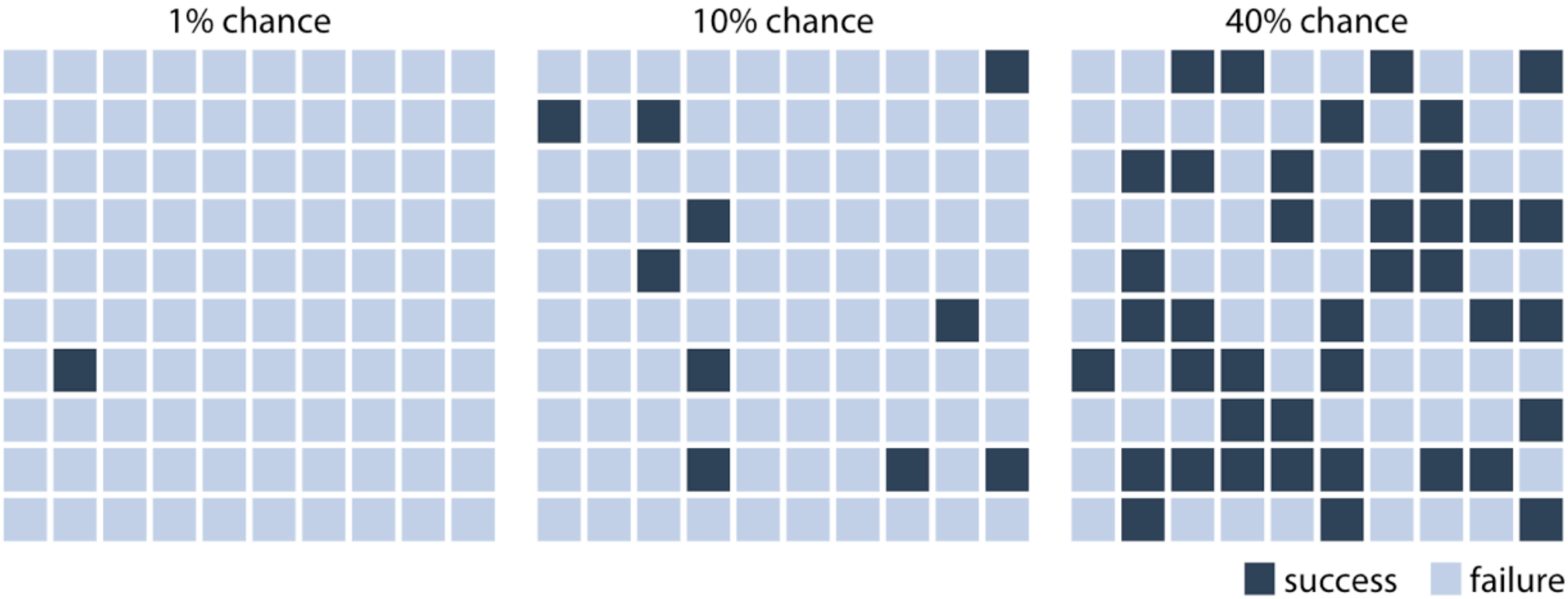
Uncertainty means exactly what it sounds like - we are not 100% sure.

- We are nearly always uncertain of future events (forecasting)
- We can also be uncertain about past events
  - I saw a parked car at 8 AM, but the next time I looked at 2PM it was gone. What time did it leave?

Quantifying uncertainty

- We quantify our uncertainty mathematically using probability
- Framing probabilities as frequencies is generally more intuitive

Framing a single uncertainty


1% chance
10% chance
40% chance
success
failure

How do we make these?

- Start out by making a grid

```r
grid <- expand.grid(x = 1:20, y = 1:20)
head(grid)
```

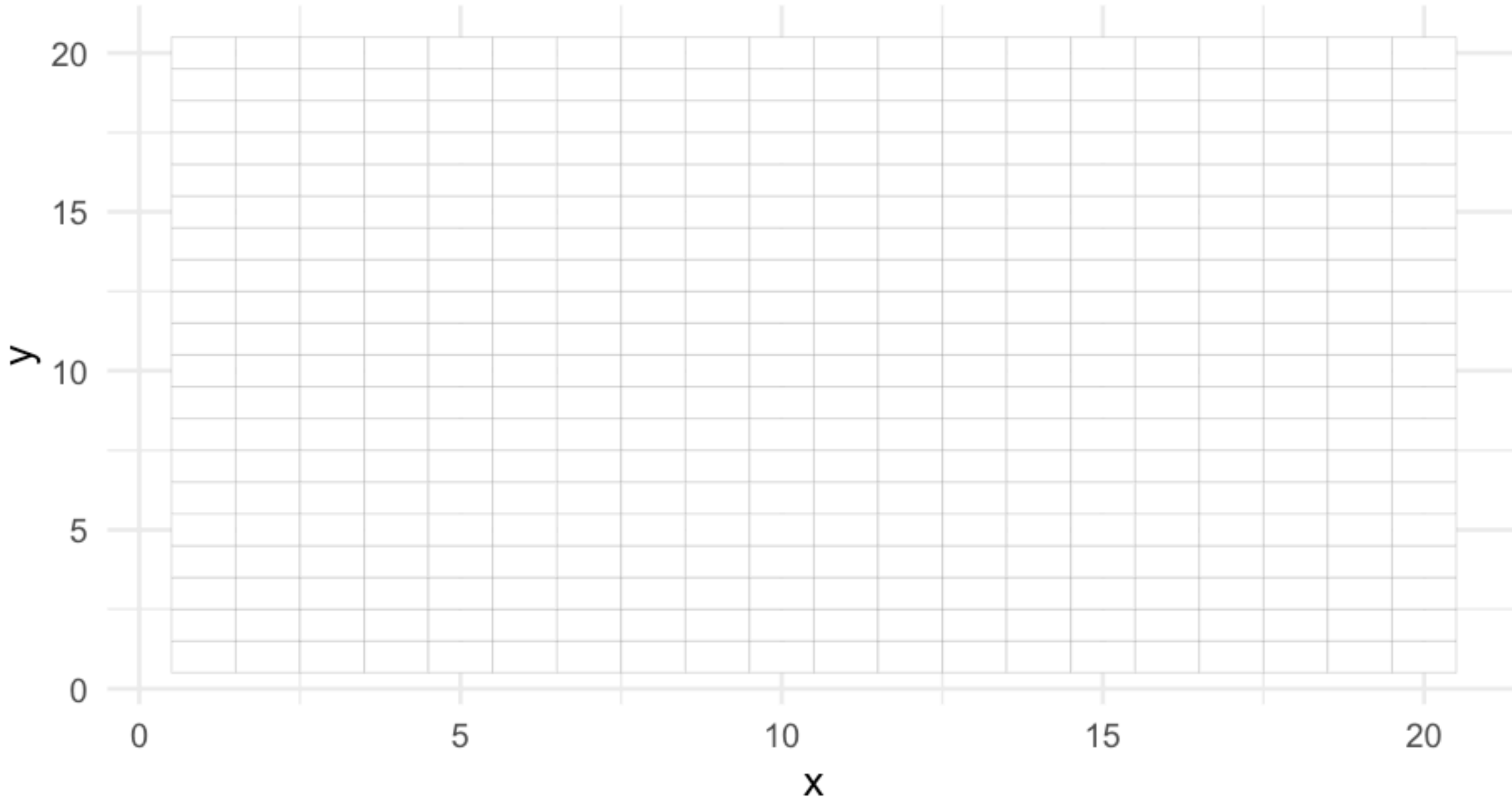
```
##   x y
## 1 1 1
## 2 2 1
## 3 3 1
## 4 4 1
## 5 5 1
## 6 6 1
```

```r
tail(grid)
```

```
##      x  y
## 395 15 20
## 396 16 20
## 397 17 20
## 398 18 20
## 399 19 20
## 400 20 20
```

Look at the grid

```
ggplot(grid, aes(x, y)) +
  geom_tile(color = "gray40",
            fill = "white")
```



Create occurrence rate

- For each sequence of x, create a variable that has the given occurence rate

How?

- Plenty of options, here's one

Consider 10%

```
nrow(grid)*.10 # n to sample
```

```
## [1] 40
```

```
set.seed(86753098)
samp <- sample(seq_len(nrow(grid)), nrow(grid)*.10)
head(samp)
```

```
## [1] 318 134 180 283 177 248
```

```
length(samp)
```

```
## [1] 40
```

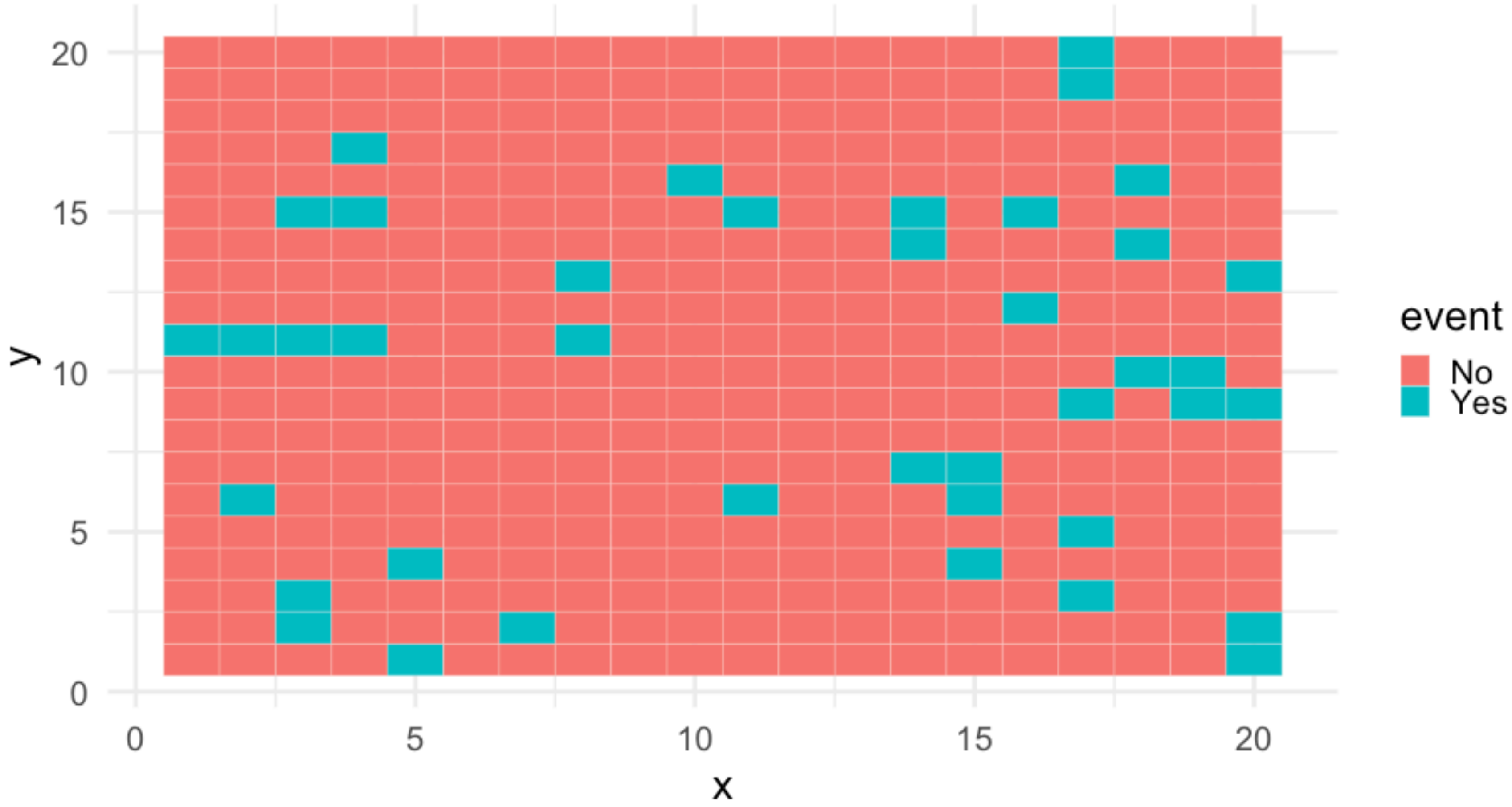
Create the variable

```
grid <- grid %>%
  rownames_to_column("row_id") %>%
  mutate(event = ifelse(row_id %in% samp, "Yes", "No"))
head(grid)
```

```
##   row_id x y event
## 1      1 1 1    No
## 2      2 2 1    No
## 3      3 3 1    No
## 4      4 4 1    No
## 5      5 5 1   Yes
## 6      6 6 1    No
```

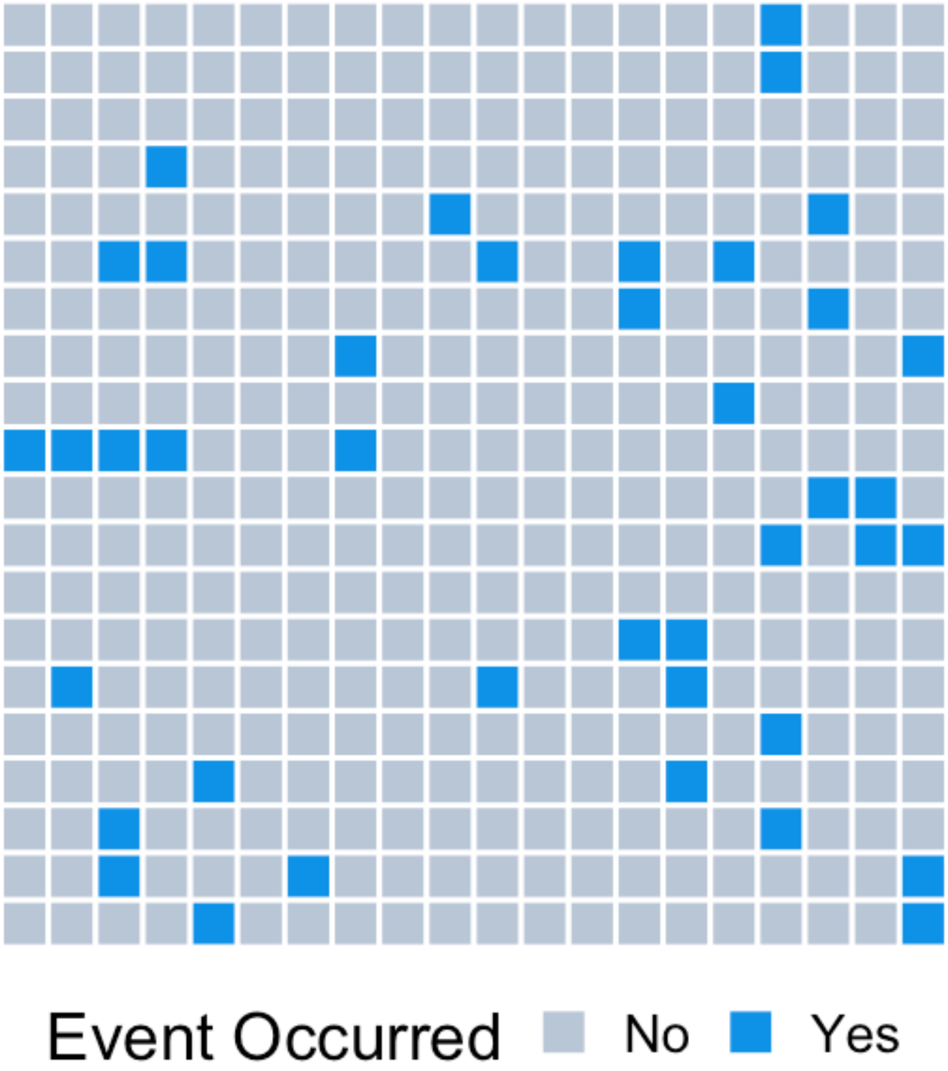
Fill in

```
ggplot(grid, aes(x, y)) +
  geom_tile(aes(fill = event), color = "white")
```

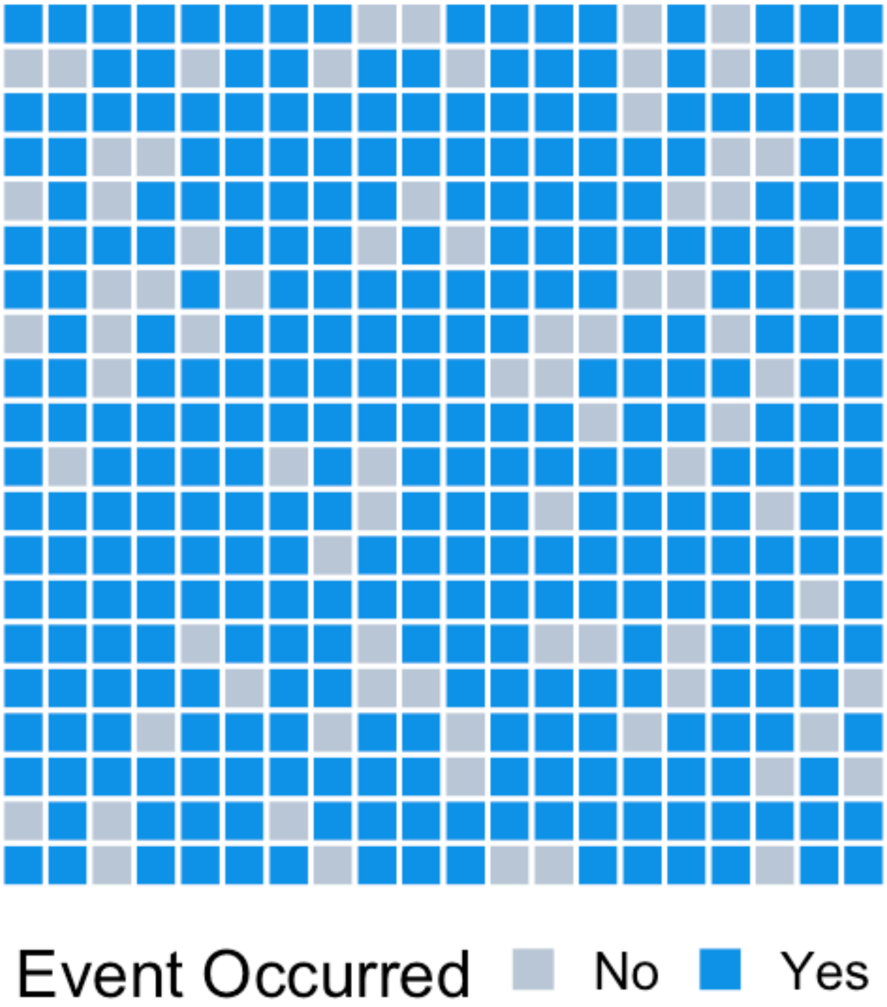


Customize

```
ggplot(grid, aes(x, y)) +
  geom_tile(aes(fill = event), color = "white", size = 1.4) +
  scale_fill_manual("Event Occurred",
                    values = c(
                      colorspace::desaturate(
                        colorspace::lighten("#1694E8", 0.5),
                      0.7),
                      "#1694E8")
                    ) +
  coord_fixed() +
  theme_void() +
  theme(legend.position = c(0.75, 0),
        legend.direction = "horizontal",
        plot.margin = margin(b = 1, unit = "cm"))
```

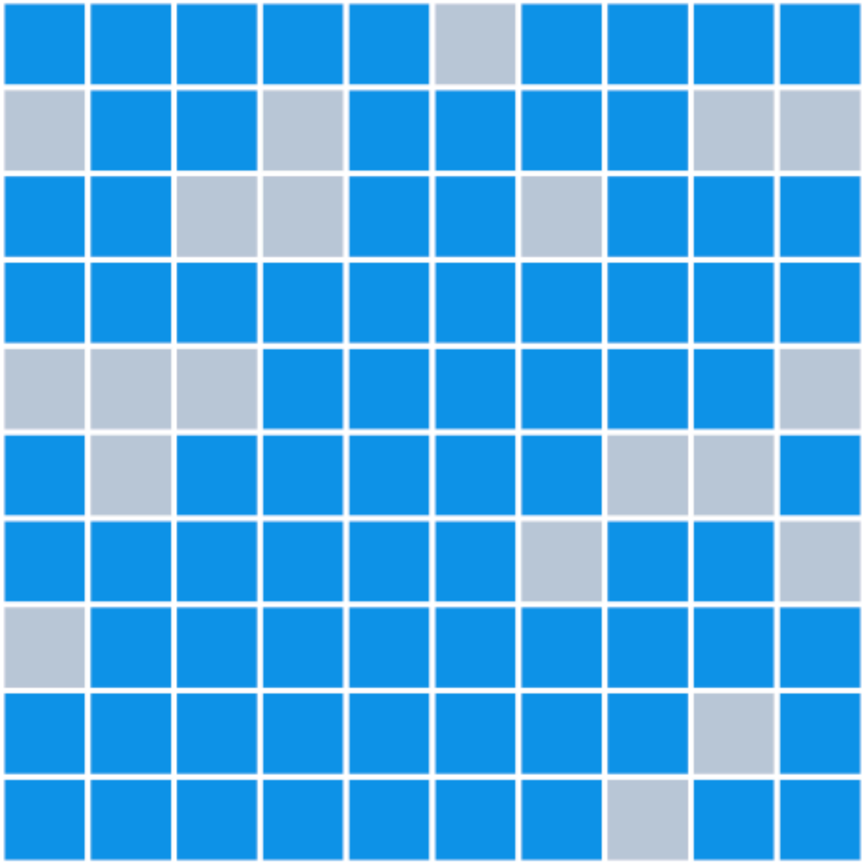

Event Occurred
No
Yes

Chance of rain



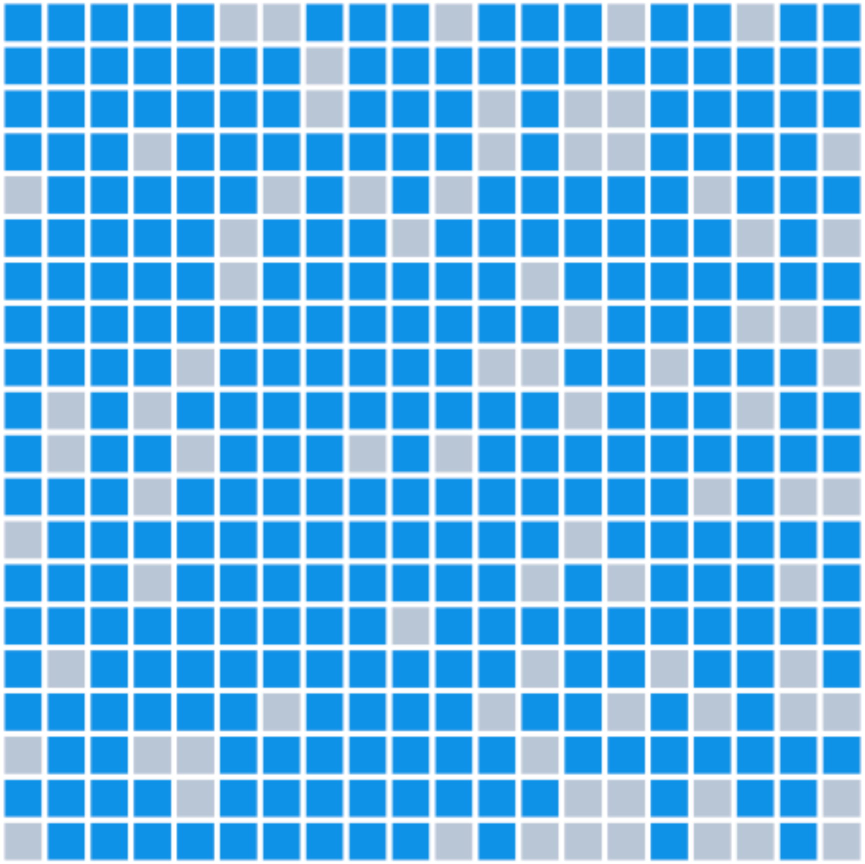
Vary grid size

10 x 10



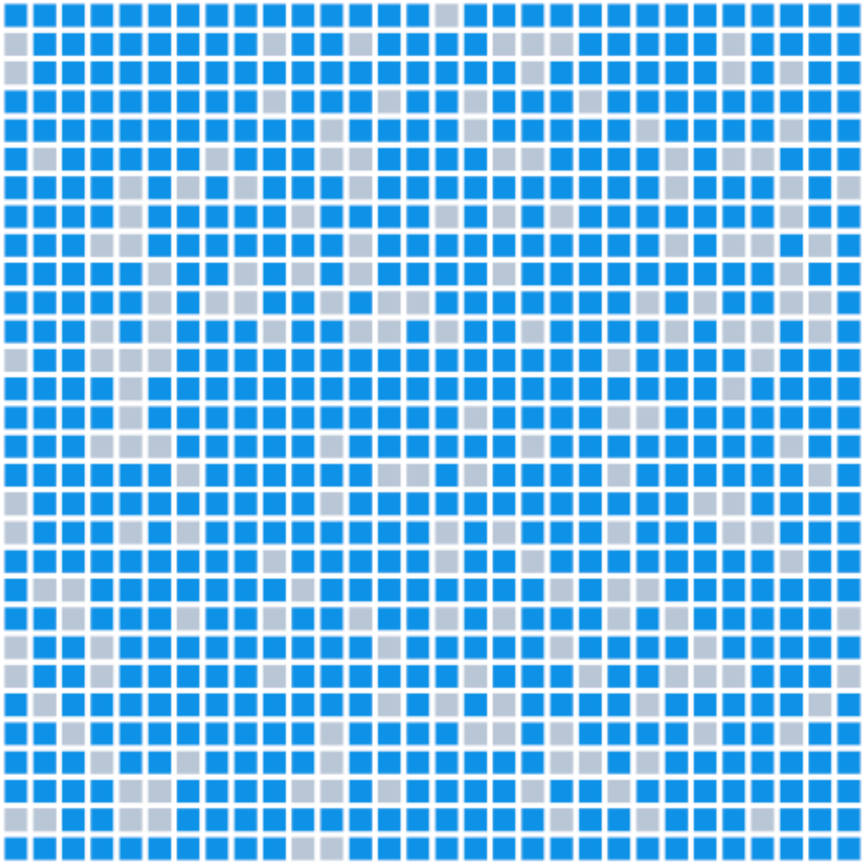
Vary grid size

20 x 20



Vary grid size

30 x 30



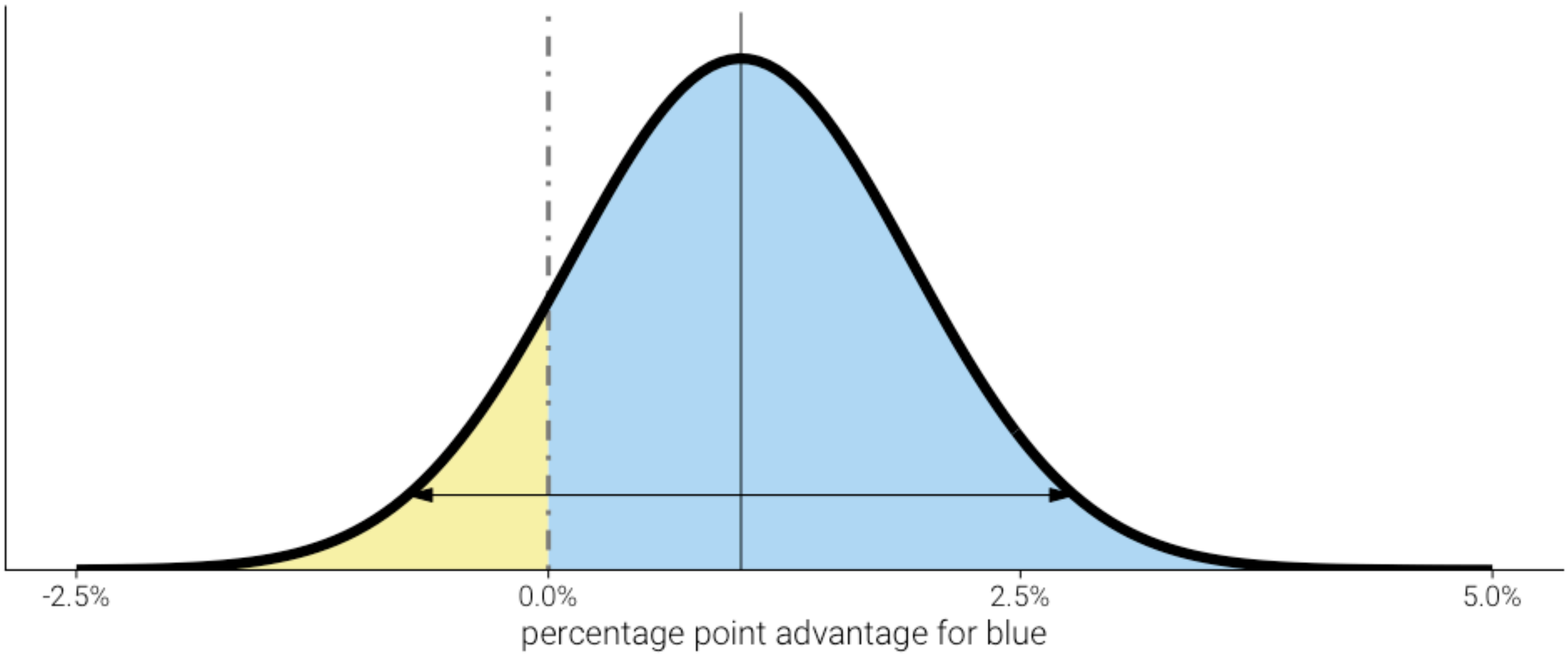
(probs too many)

Non-discrete probabilities

Hypothetical

Blue party has 1% advantage w/ margin of error of 1.76 points

Who will win?



A bit of math

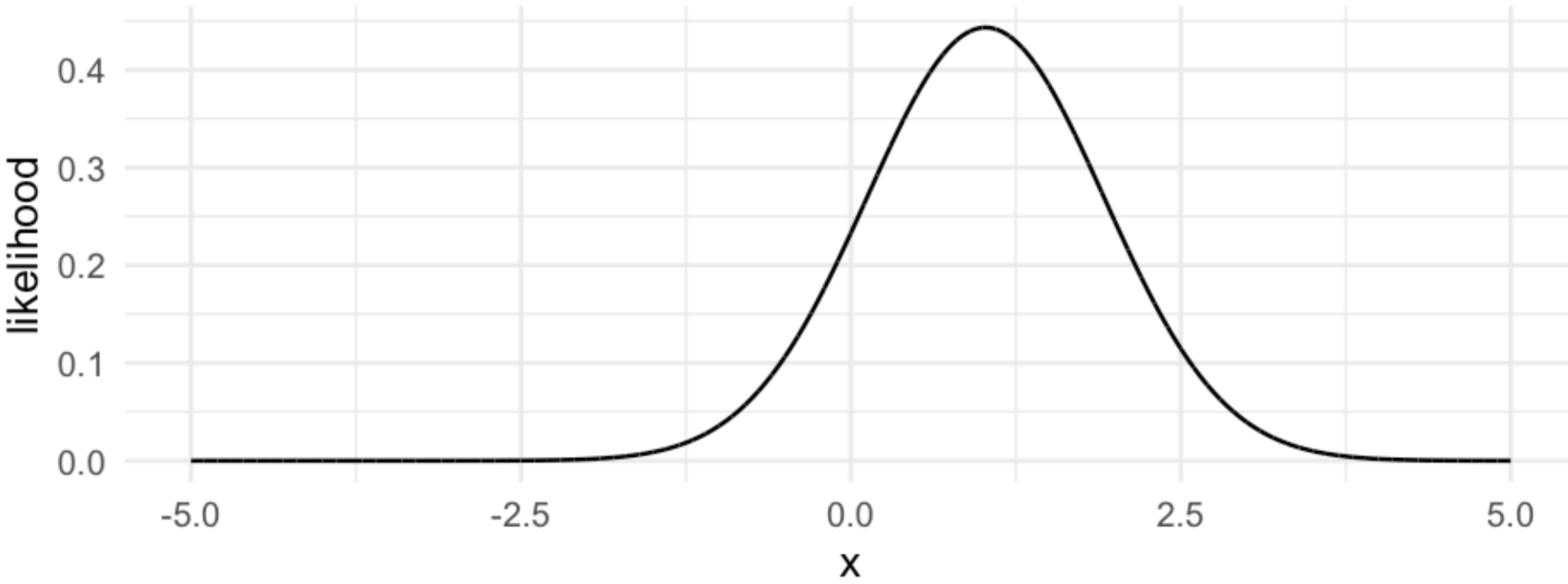
Our prior distribution was defined by $\mu = 1.02$ and $sd = 0.9$.

- What's the chance the end result is below zero?

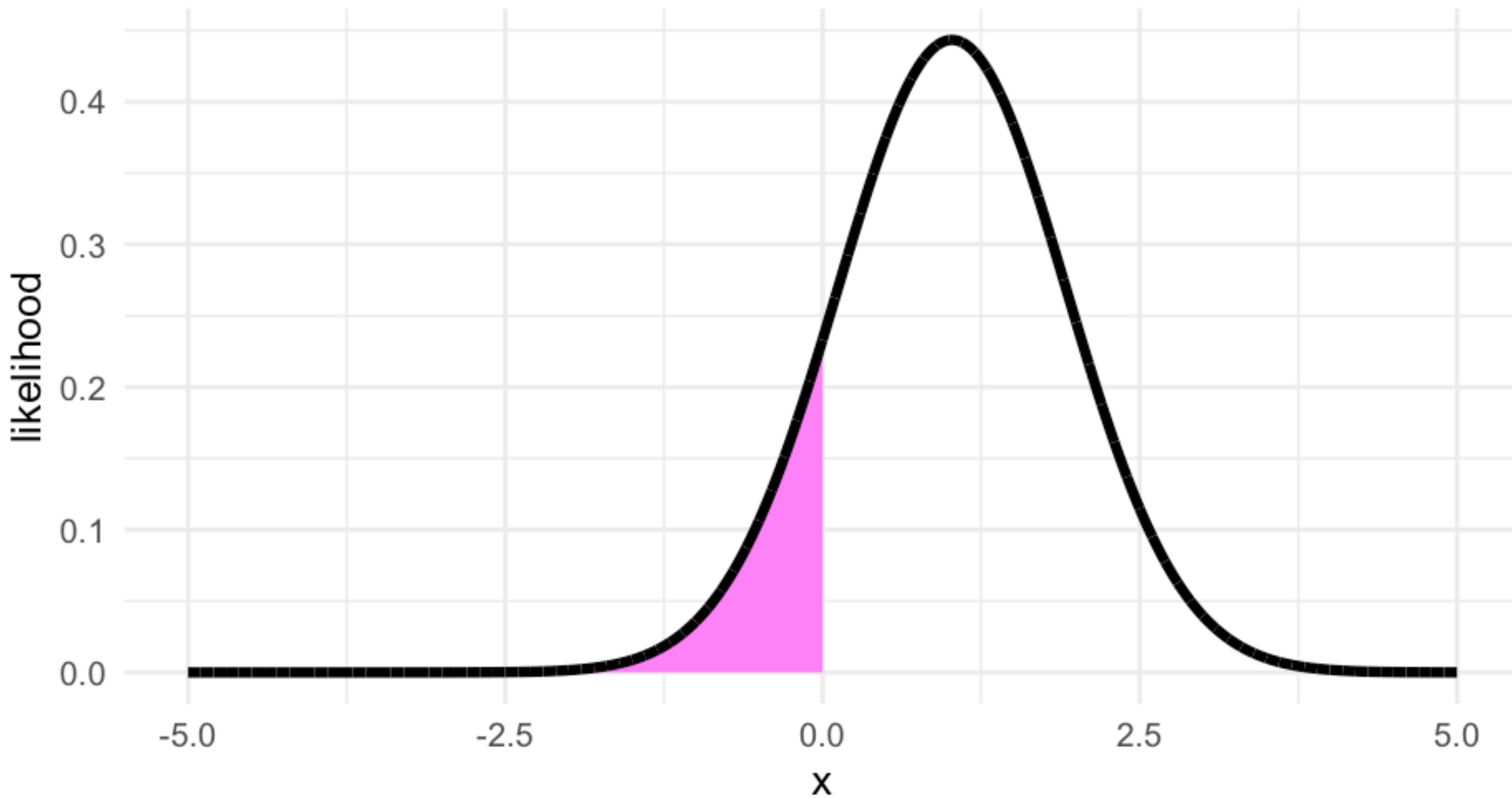
The hard way

Calculate the exact probability of data below zero under this distribution

```
x <- seq(-5, 5, 0.001)
likelihood <- dnorm(x, 1.02, 0.9)
sim <- data.frame(x, likelihood)
ggplot(sim, aes(x, likelihood)) +
  geom_line(size = 1.2)
```



How do we calculate this portion?



Integrate

```
zab <- filter(sim, x <= 0)
pracma::trapz(zab$x, zab$likelihood)
```

```
## [1] 0.1285372
```

Easier: Simulate

```
random_draws <- rnorm(1e5, 1.02, 0.9)
table(random_draws > 0) / 1e5
```

```
##
##   FALSE    TRUE
## 0.12968 0.87032
```

Discretized plot

```r
ppoints(50)
```

```
##  [1] 0.01 0.03 0.05 0.07 0.09 0.11 0.13 0.15 0.17 0.19 0.21 0.23 0.25 0.27 0.29
## [16] 0.31 0.33 0.35 0.37 0.39 0.41 0.43 0.45 0.47 0.49 0.51 0.53 0.55 0.57 0.59
## [31] 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81 0.83 0.85 0.87 0.89
## [46] 0.91 0.93 0.95 0.97 0.99
```

```r
qnorm(ppoints(50), 1.02, 0.9)
```

```
##  [1] -1.073713087 -0.672714247 -0.460368264 -0.308211925 -0.186679530
##  [6] -0.083875308  0.006247984  0.087209949  0.161251272  0.229893334
## [11]  0.294220878  0.355037836  0.412959225  0.468468308  0.521953752
## [16]  0.573734687  0.624078151  0.673211580  0.721331988  0.768612869
## [21]  0.815209521  0.861263252  0.906904788  0.952257124  0.997437983
## [26]  1.042562017  1.087742876  1.133095212  1.178736748  1.224790479
## [31]  1.271387131  1.318668012  1.366788420  1.415921849  1.466265313
## [36]  1.518046248  1.571531692  1.627040775  1.684962164  1.745779122
## [41]  1.810106666  1.878748728  1.952790051  2.033752016  2.123875308
## [46]  2.226679530  2.348211925  2.500368264  2.712714247  3.113713087
```

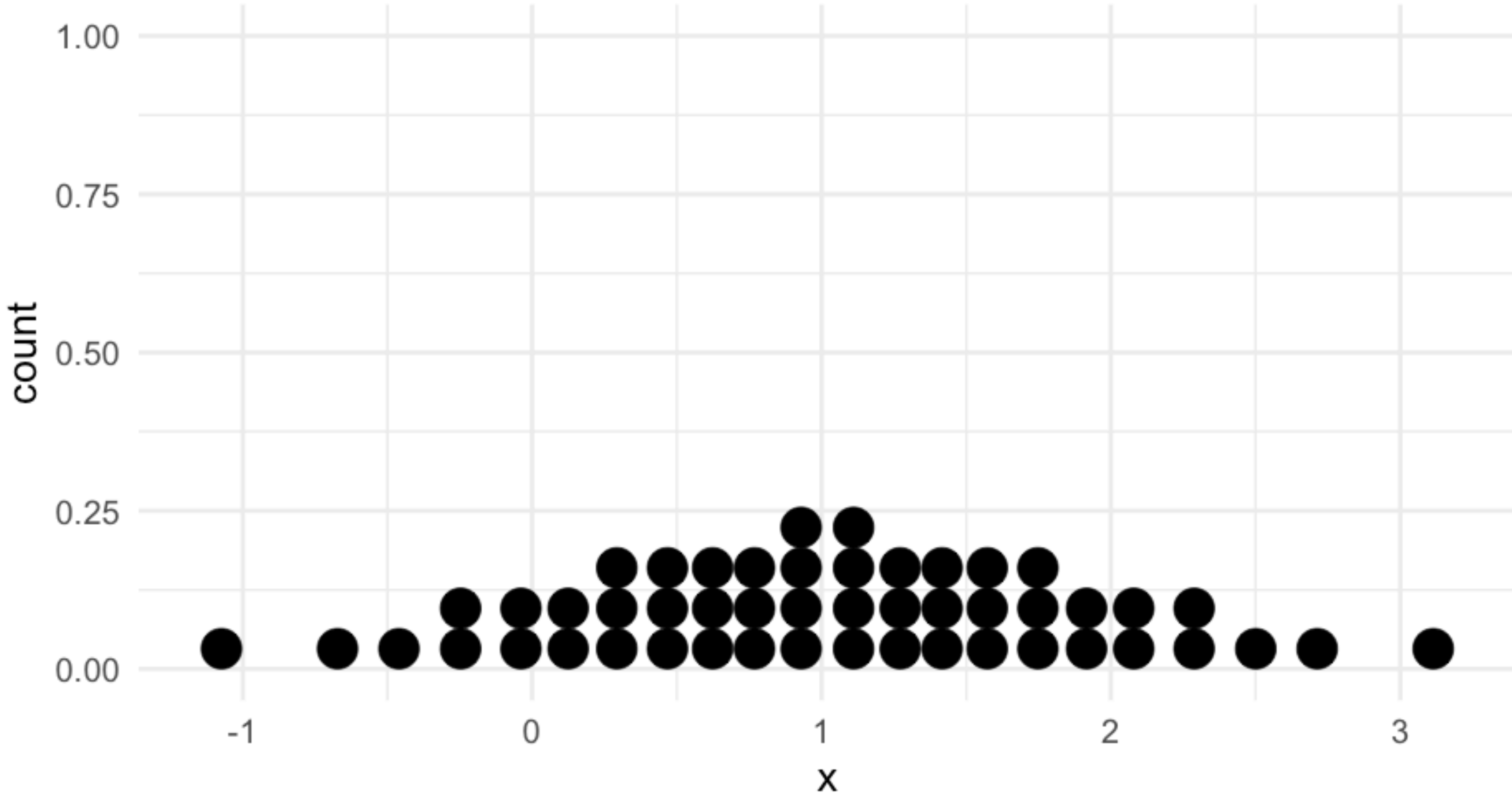
```r
discretized <- data.frame(x = qnorm(ppoints(50), 1.02, 0.9)) %>%
  mutate(winner = ifelse(x <= 0, "#b1daf4", "#f8f1a9"))

head(discretized)
```

```
##             x  winner
## 1 -1.07371309 #b1daf4
## 2 -0.67271425 #b1daf4
## 3 -0.46036826 #b1daf4
## 4 -0.30821193 #b1daf4
## 5 -0.18667953 #b1daf4
## 6 -0.08387531 #b1daf4
```

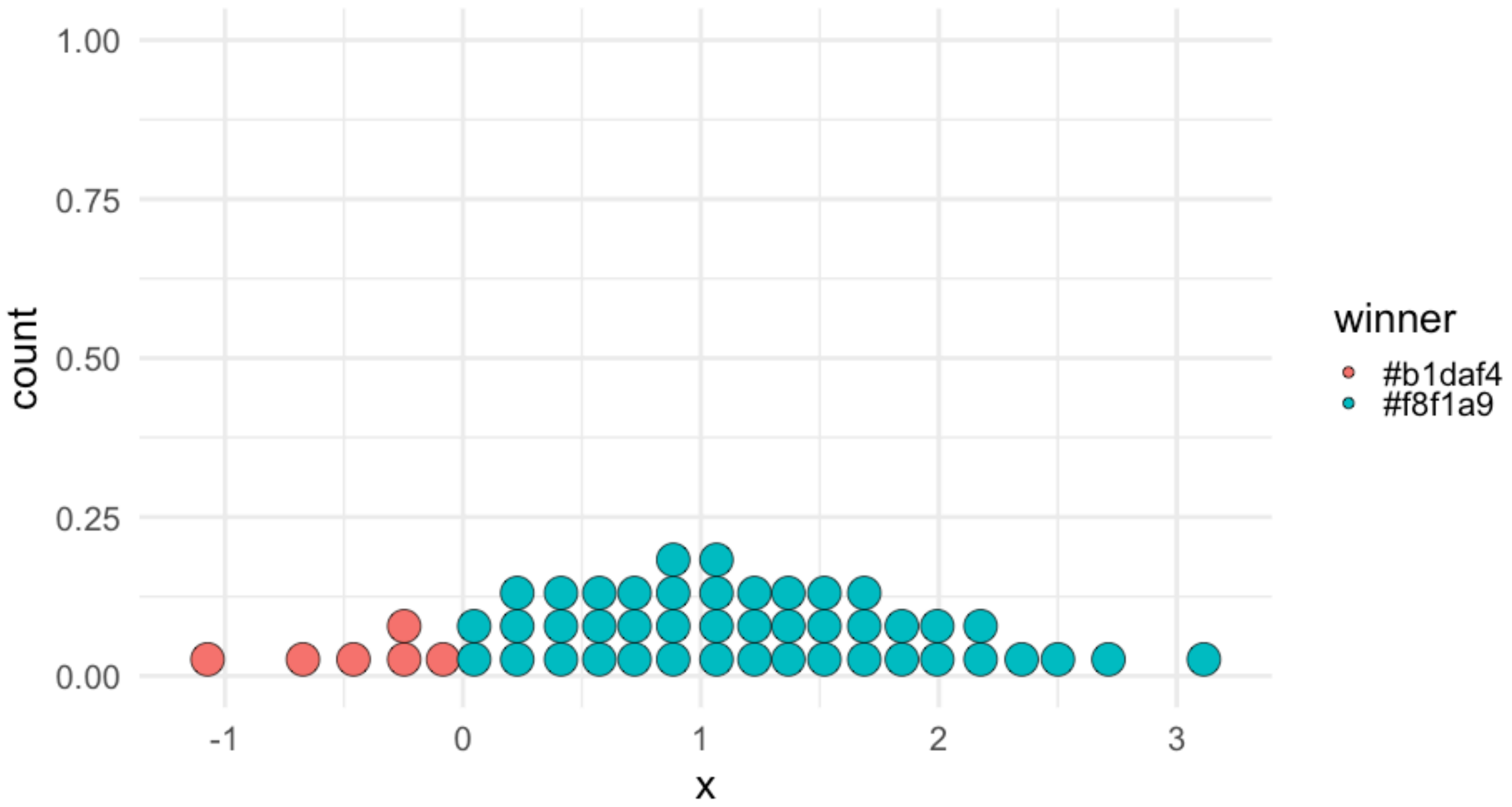
```r
tail(discretized)
```

```
##           x  winner
## 45 2.123875 #f8f1a9
## 46 2.226680 #f8f1a9
## 47 2.348212 #f8f1a9
## 48 2.500368 #f8f1a9
## 49 2.712714 #f8f1a9
## 50 3.113713 #f8f1a9
```

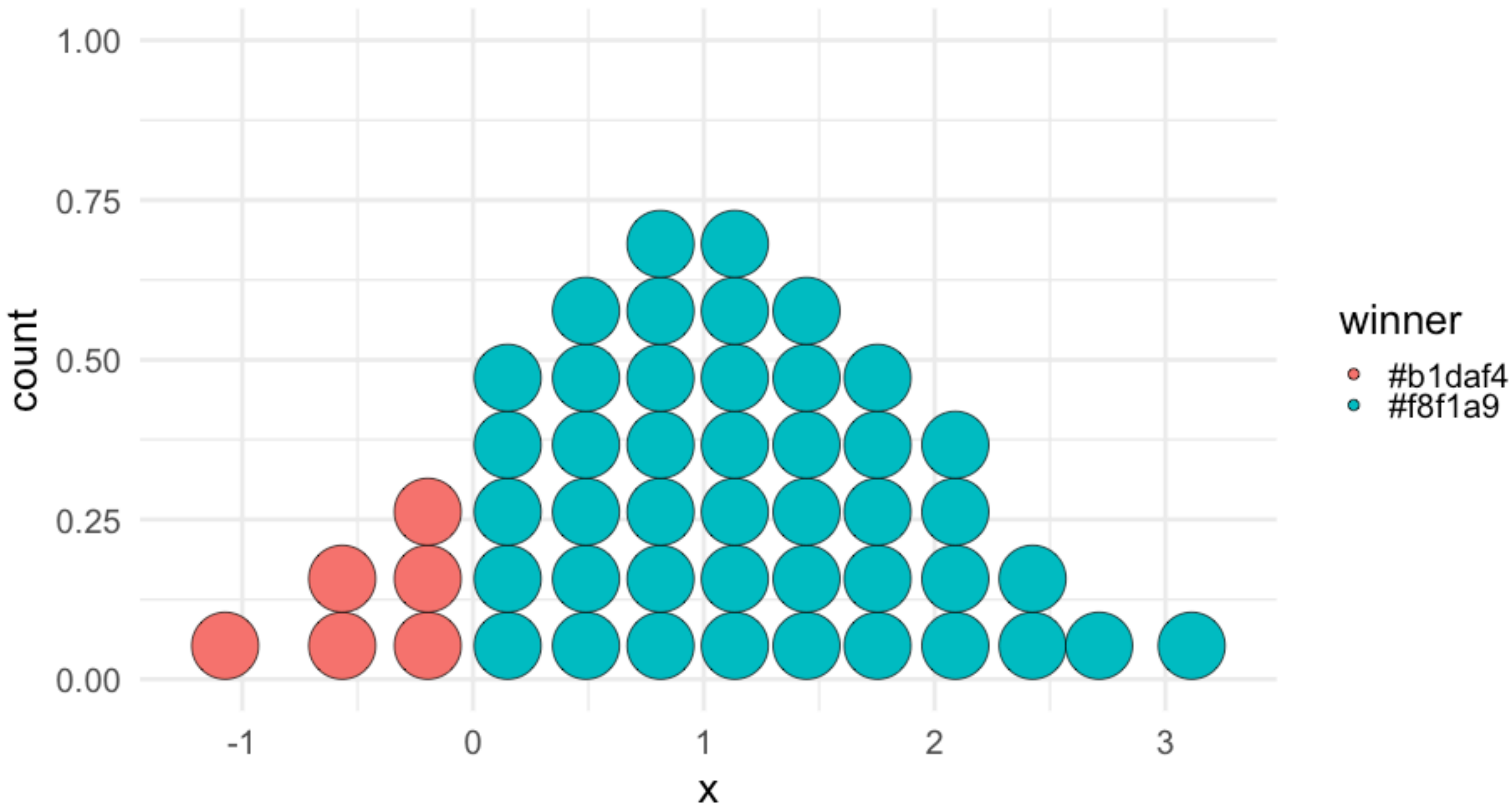
```r
ggplot(discretized, aes(x)) +
  geom_dotplot()
```



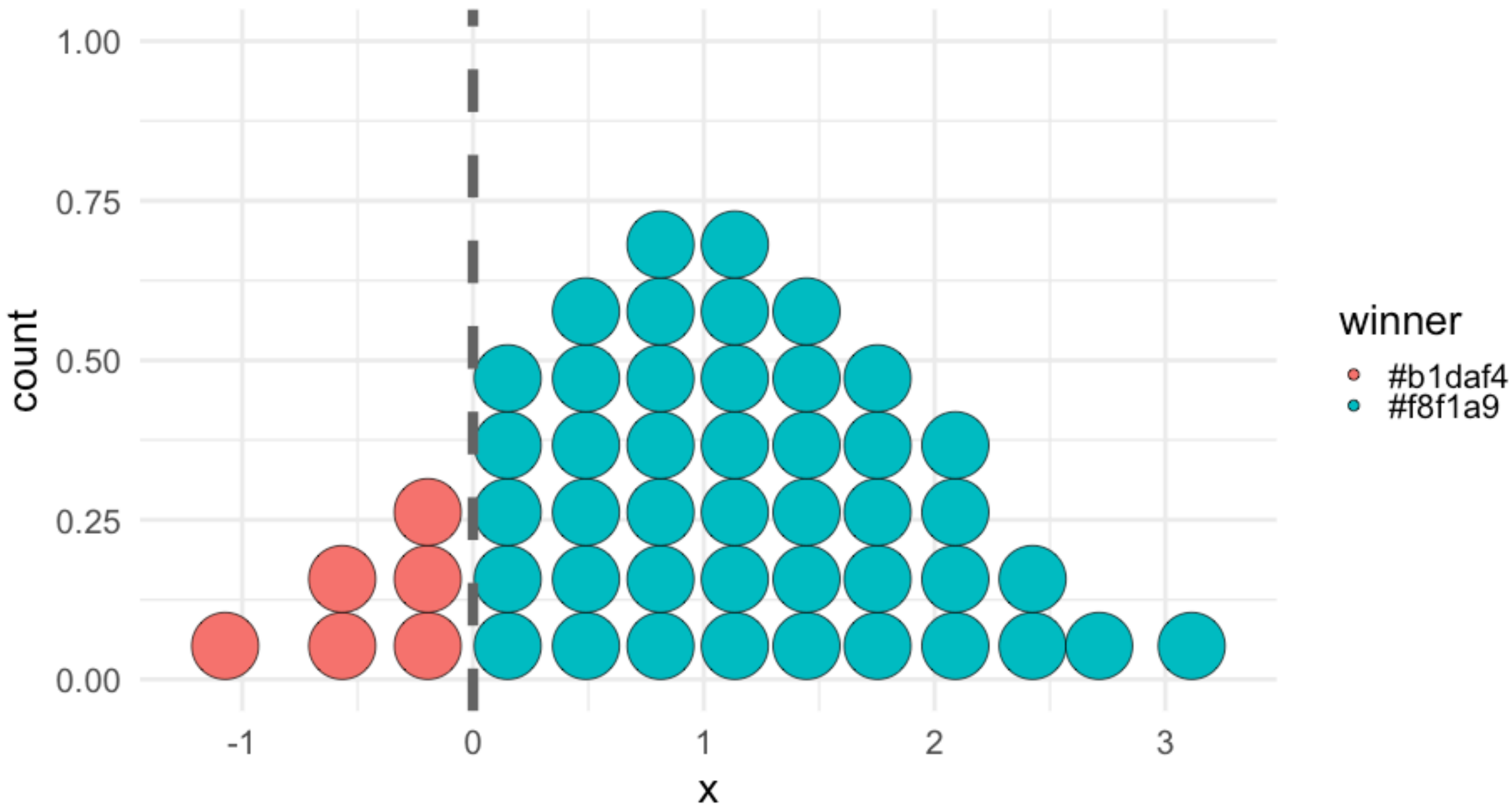
```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner))
```



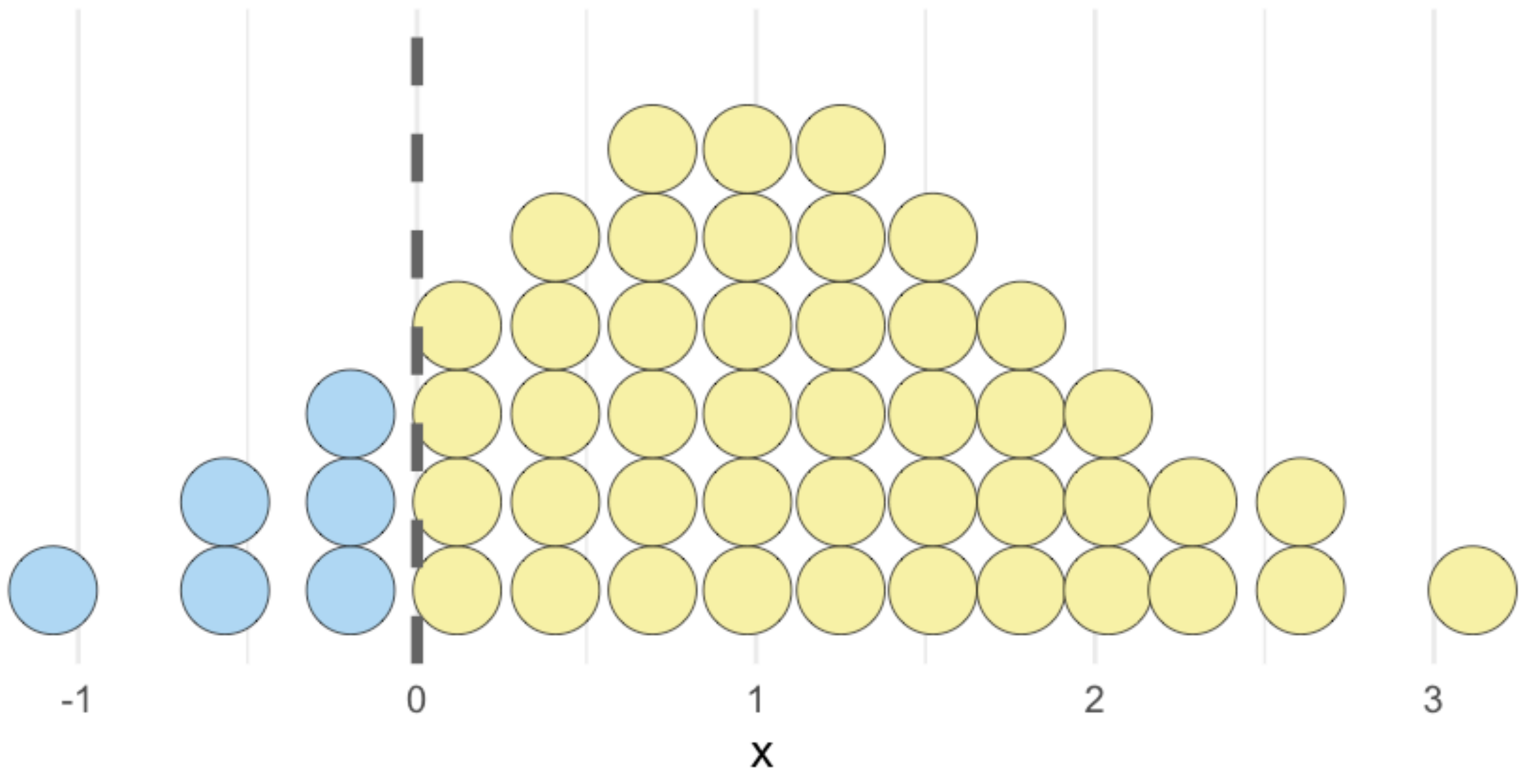
```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.29)
```



```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.29) +
  geom_vline(xintercept = 0, color = "gray40", linetype = 2, size = 3)
```

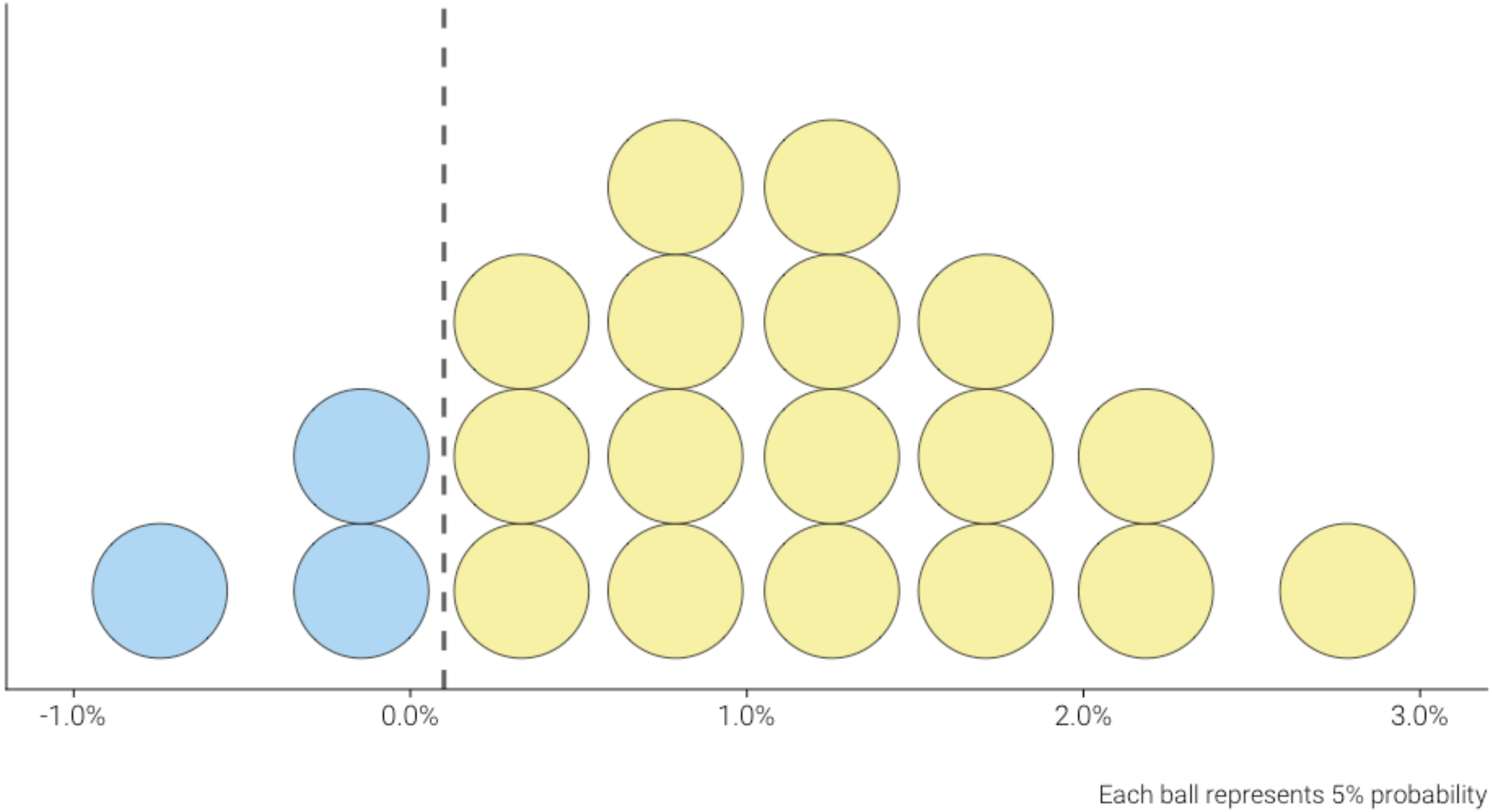


```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.26) +
  geom_vline(xintercept = 0, color = "gray40", linetype = 2, size = 3) +
  scale_fill_identity(guide = "none") +
  scale_y_continuous(name = "",
                     breaks = NULL)
```



Probs too many though

```r
discretized2 <- data.frame(x = qnorm(ppoints(20), 1.02, 0.9)) %>%
  mutate(winner = ifelse(x <= 0, "#b1daf4", "#f8f1a9"))

ggplot(discretized2, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.4) +
  geom_vline(xintercept = 0.1, color = "gray40", linetype = 2, size = 1.4) +
  scale_fill_identity(guide = "none") +
  scale_x_continuous(name = "",
                     limits = c(-1, 3),
                     labels = scales::percent_format(scale = 1)) +
  theme_dviz_open(20, font_family = "Roboto Light") +
  scale_y_continuous(breaks = NULL,
                     name = "") +
  labs(caption = "Each ball represents 5% probability")
```


-1.0%
0.0%
1.0%
2.0%
3.0%
Each ball represents 5% probability

Uncertainty of point estimates

Quick review (hopefully a review)

- What is a standard error?
- Standard deviation of the sampling distribution
- What is the sampling distribution?
- Samples from the underlying, population-based, generative distribution
- What does this mean, exactly?
- Let's simulate to explore

Simulation

- Imagine the "real" distribution has $\mu = 100$ and $\sigma = 10$.
- Let's draw a sample of 10 from this distribution

```
set.seed(123)
samp10a <- rnorm(n = 10, mean = 100, sd = 10)
samp10a
```

```
##  [1]  94.39524  97.69823 115.58708 100.70508 101.29288 117.15065 104.60916
##  [8]  87.34939  93.13147  95.54338
```

- Calculate the mean

```
mean(samp10a)
```

```
## [1] 100.7463
```

Do it a second time

```r
samp10b <- rnorm(n = 10, mean = 100, sd = 10)
samp10b
```

```
##  [1] 112.24082 103.59814 104.00771 101.10683  94.44159 117.86913 104.97850
##  [8]  80.33383 107.01356  95.27209
```

```r
mean(samp10b)
```

```
## [1] 102.0862
```

Do it a bunch of times

```r
# from purrr (base methods work basically just as well in this case)
samples <- rerun(1000, rnorm(10, mean = 100, sd = 10))
samples
```

```
## [[1]]
##  [1]  89.32176  97.82025  89.73996  92.71109  93.74961  83.13307 108.37787
##  [8] 101.53373  88.61863 112.53815
##
## [[2]]
##  [1] 104.26464  97.04929 108.95126 108.78133 108.21581 106.88640 105.53918
##  [8]  99.38088  96.94037  96.19529
##
## [[3]]
##  [1]  93.05293  97.92083  87.34604 121.68956 112.07962  88.76891  95.97115
##  [8]  95.33345 107.79965  99.16631
##
## [[4]]
##  [1] 102.53319  99.71453  99.57130 113.68602  97.74229 115.16471  84.51247
##  [8] 105.84614 101.23854 102.15942
##
## [[5]]
##  [1] 103.79639  94.97677  96.66793  89.81425  89.28209 103.03529 104.48210
##  [8] 100.53004 109.22267 120.50085
##
```

Calculate all means

```
head(
     map_dbl(samples, mean)
     )
```

```
## [1]  95.75441 103.22045  99.91284 102.21686 101.23084  96.37082
```

- What's the sd of these means? That's the standard error.

```
sd(map_dbl(samples, mean))
```

```
## [1] 3.144175
```

- Note that it depends on sample size. Let's re-do this, pulling a sample of 100 each time.

```
samples2 <- rerun(1000, rnorm(100, mean = 100, sd = 10))
sd(map_dbl(samples2, mean))
```

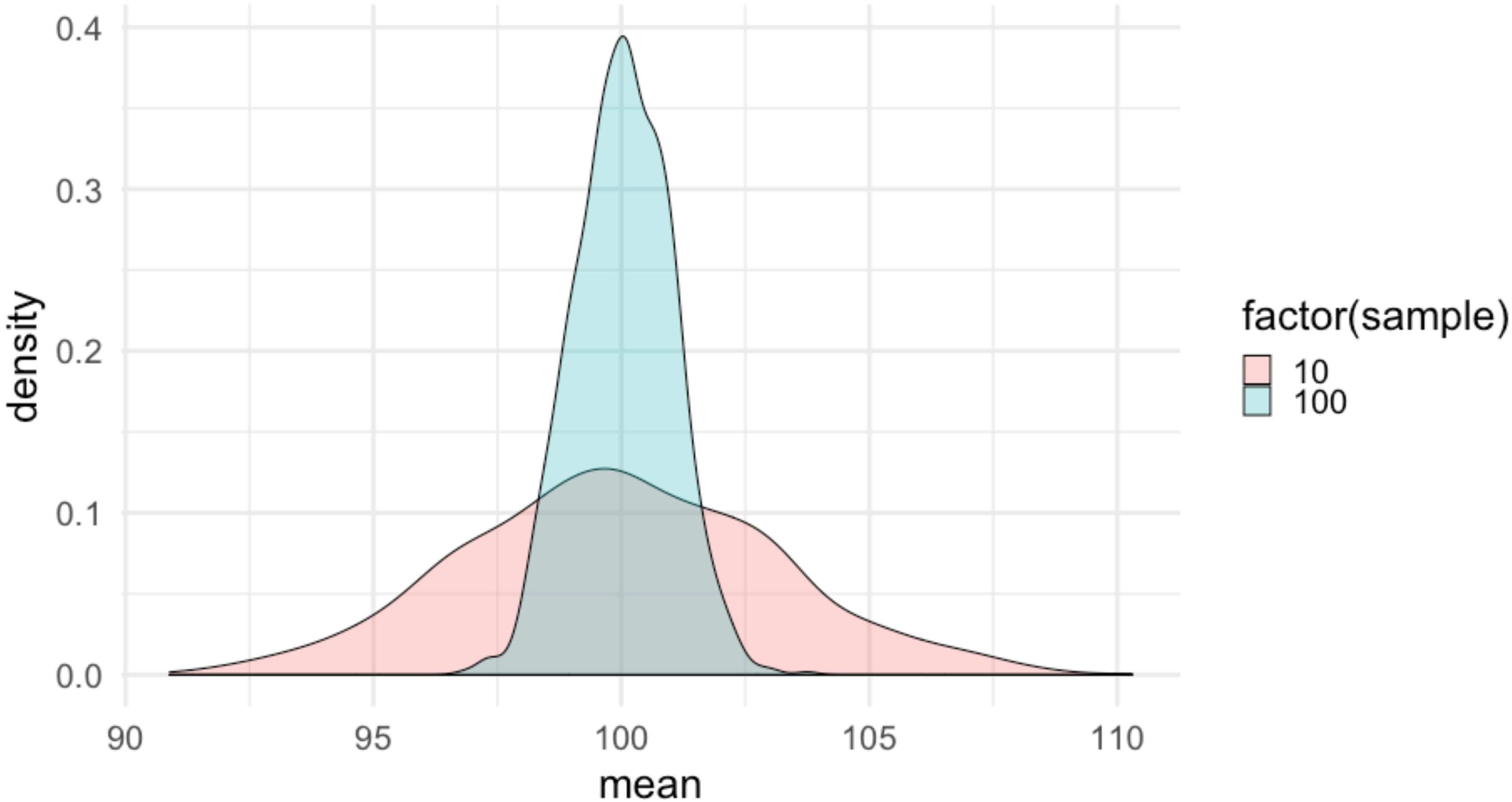
```
## [1] 0.9728883
```

Visualize the sampling distributions

```
sample_means <- tibble(iter = rep(1:1000, 2),
                       sample = rep(c(10, 100), each = 1000),
                       mean = c(map_dbl(samples, mean),
                                map_dbl(samples2, mean))
                       )
sample_means
```

```
## # A tibble: 2,000 x 3
##     iter sample      mean
##    <int>  <dbl>     <dbl>
##  1     1     10  95.75441
##  2     2     10 103.2204
##  3     3     10  99.91284
##  4     4     10 102.2169
##  5     5     10 101.2308
##  6     6     10  96.37082
##  7     7     10 103.1310
##  8     8     10 104.3709
##  9     9     10  96.04152
## 10    10     10  96.77087
## # … with 1,990 more rows
```

```r
ggplot(sample_means, aes(mean)) +
  geom_density(aes(fill = factor(sample)), alpha = 0.3)
```



Fit a model

```
m <- lm(cty ~ displ + class, mpg)
summary(m)
```

```
##
## Call:
## lm(formula = cty ~ displ + class, data = mpg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2689 -1.1503 -0.0156  1.0341 12.9782
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      28.7768     1.4729  19.538  < 2e-16 ***
## displ            -2.1716     0.1747 -12.433  < 2e-16 ***
## classcompact     -3.5991     1.2522  -2.874  0.00444 **
## classmidsize     -3.6755     1.2063  -3.047  0.00259 **
## classminivan     -5.5951     1.3060  -4.284 2.71e-05 ***
## classpickup      -6.1825     1.1214  -5.513 9.60e-08 ***
## classsubcompact  -2.6290     1.2369  -2.125  0.03464 *
## classsuv         -5.5994     1.0872  -5.150 5.65e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Visualize with standard errors

```r
library(broom)
tidied_m <- tidy(m, conf.int = TRUE)

tidied_m
```

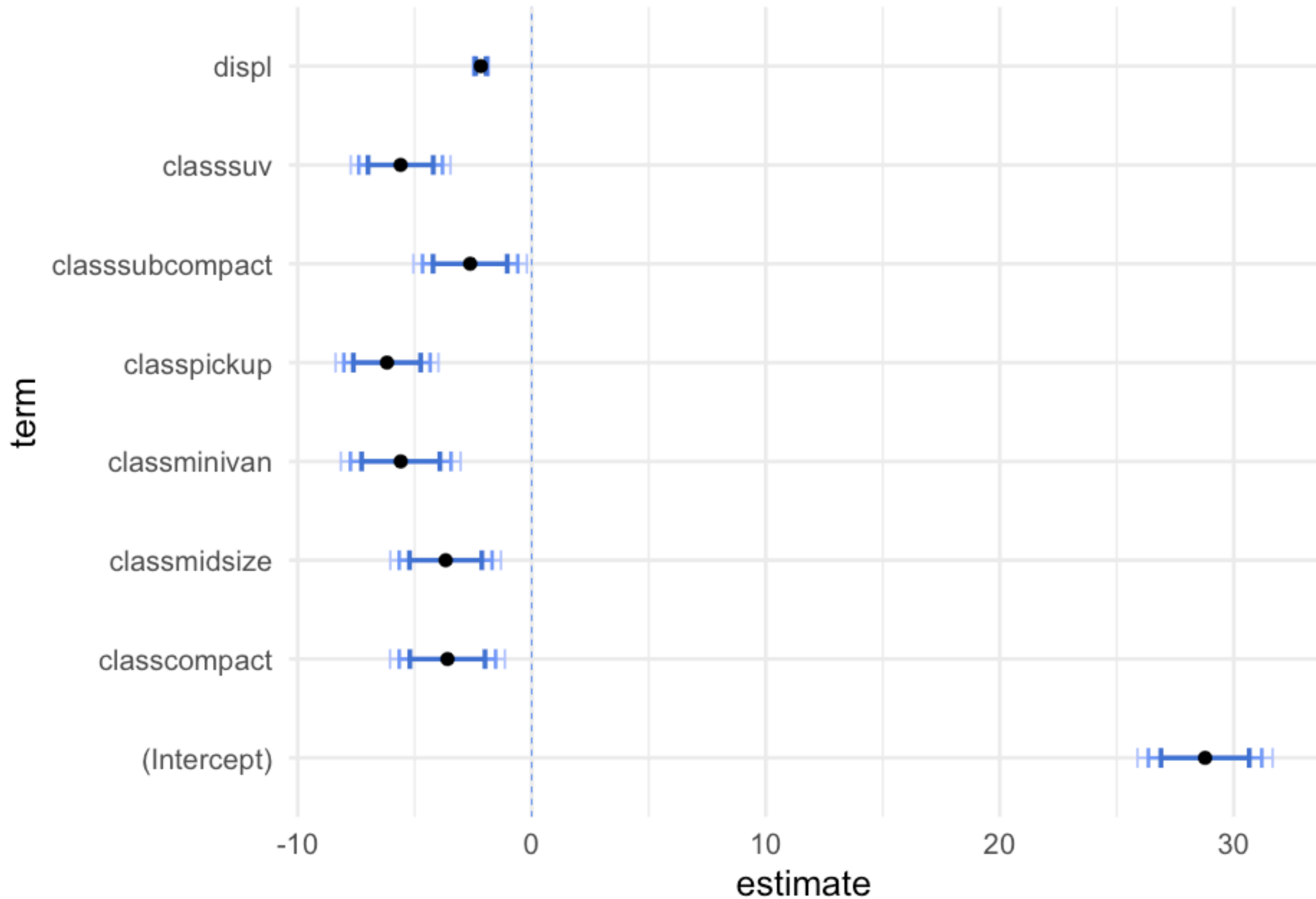
```
## # A tibble: 8 x 7
##   term           estimate std.error  statistic      p.value  conf.low  conf.hig
##   <chr>             <dbl>     <dbl>      <dbl>        <dbl>     <dbl>     <dbl
## 1 (Intercept)   28.77682  1.472892   19.53763  1.905873e-50 25.87446  31.67918
## 2 displ         -2.171562 0.1746638 -12.43281  2.197130e-27 -2.515740 -1.827384
## 3 classcompact  -3.599125 1.252190   -2.874265 4.436052e- 3 -6.066585 -1.131664
## 4 classmidsize  -3.675526 1.206253   -3.047061 2.585762e- 3 -6.052466 -1.298585
## 5 classminivan  -5.595070 1.305993   -4.284151 2.714490e- 5 -8.168550 -3.021590
## 6 classpickup   -6.182466 1.121448   -5.512931 9.600087e- 8 -8.392297 -3.972634
## 7 classsubcomp… -2.629038 1.236950   -2.125420 3.463687e- 2 -5.066467 -0.191608
## 8 classsuv      -5.599361 1.087160   -5.150446 5.652249e- 7 -7.741628 -3.457093
```

```r
ggplot(tidied_m, aes(term, estimate)) +
  geom_hline(yintercept = 0,
             color = "cornflowerblue",
             linetype = 2) +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high)) +
  geom_point() +
  coord_flip()
```

Alternative methods

Multiple error bars

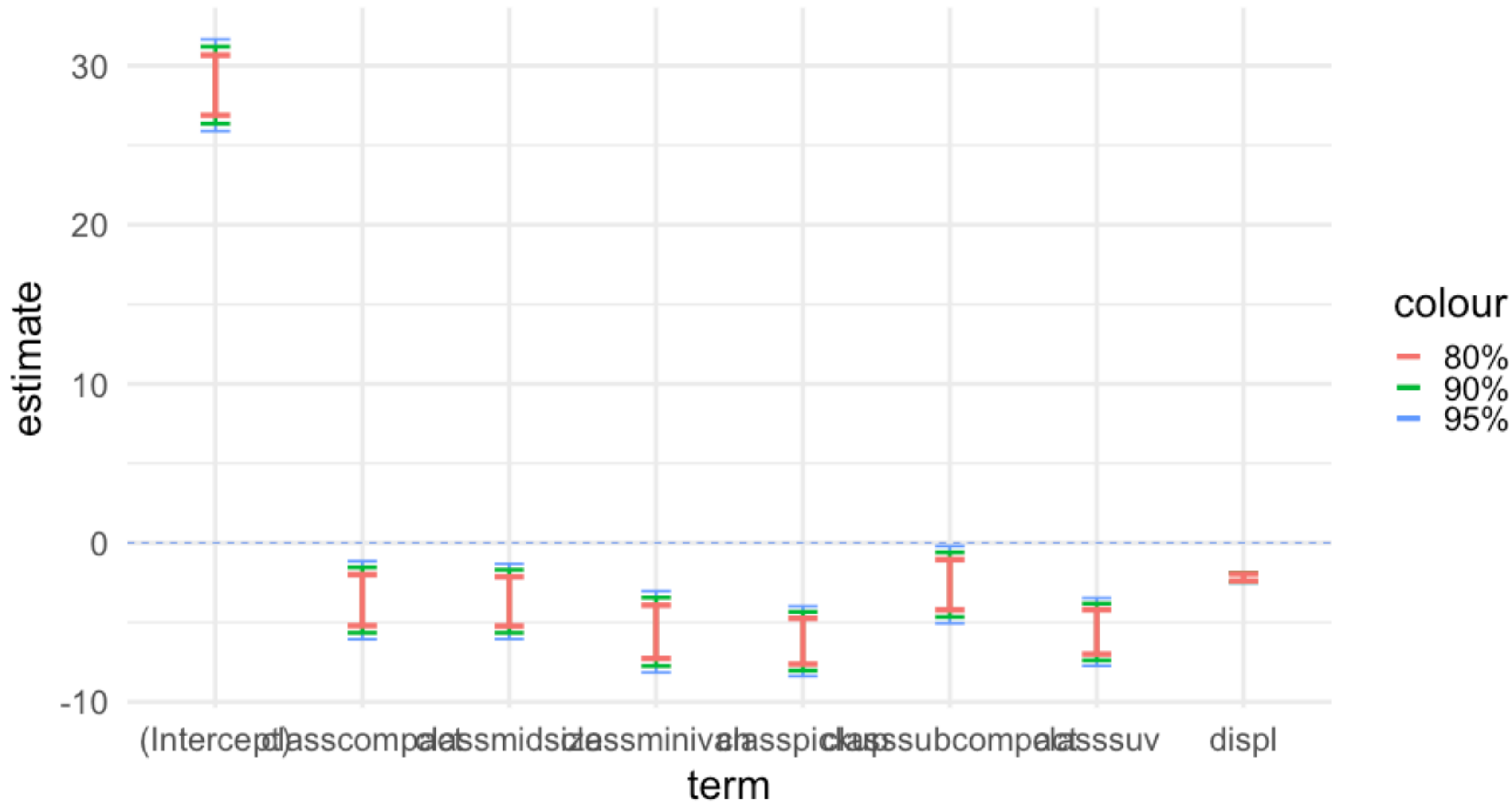
```
library(colorspace)
ggplot(tidied_m, aes(term, estimate)) +
  geom_hline(yintercept = 0,
             color = "cornflowerblue",
             linetype = 2) +
  geom_errorbar(aes(ymin = estimate + qnorm(.025)*std.error,
                    ymax = estimate + qnorm(.975)*std.error),
                color = lighten("#4375D3", .6),
                width = 0.2,
                size = 0.8) + # 95% CI
  geom_errorbar(aes(ymin = estimate + qnorm(.05)*std.error,
                    ymax = estimate + qnorm(.95)*std.error),
                color = lighten("#4375D3", .3),
                width = 0.2,
                size = 1.2) + # 90% CI
  geom_errorbar(aes(ymin = estimate + qnorm(.1)*std.error,
                    ymax = estimate + qnorm(.9)*std.error),
                color = "#4375D3",
                width = 0.2,
                size = 1.6) + # 80% CI
  geom_point() +
  coord_flip()
```


displ
classsuv
classsubcompact
classpickup
classminivan
classmidsize
classcompact
(Intercept)
term
-10
0
10
20
30
estimate

Add levels to legend

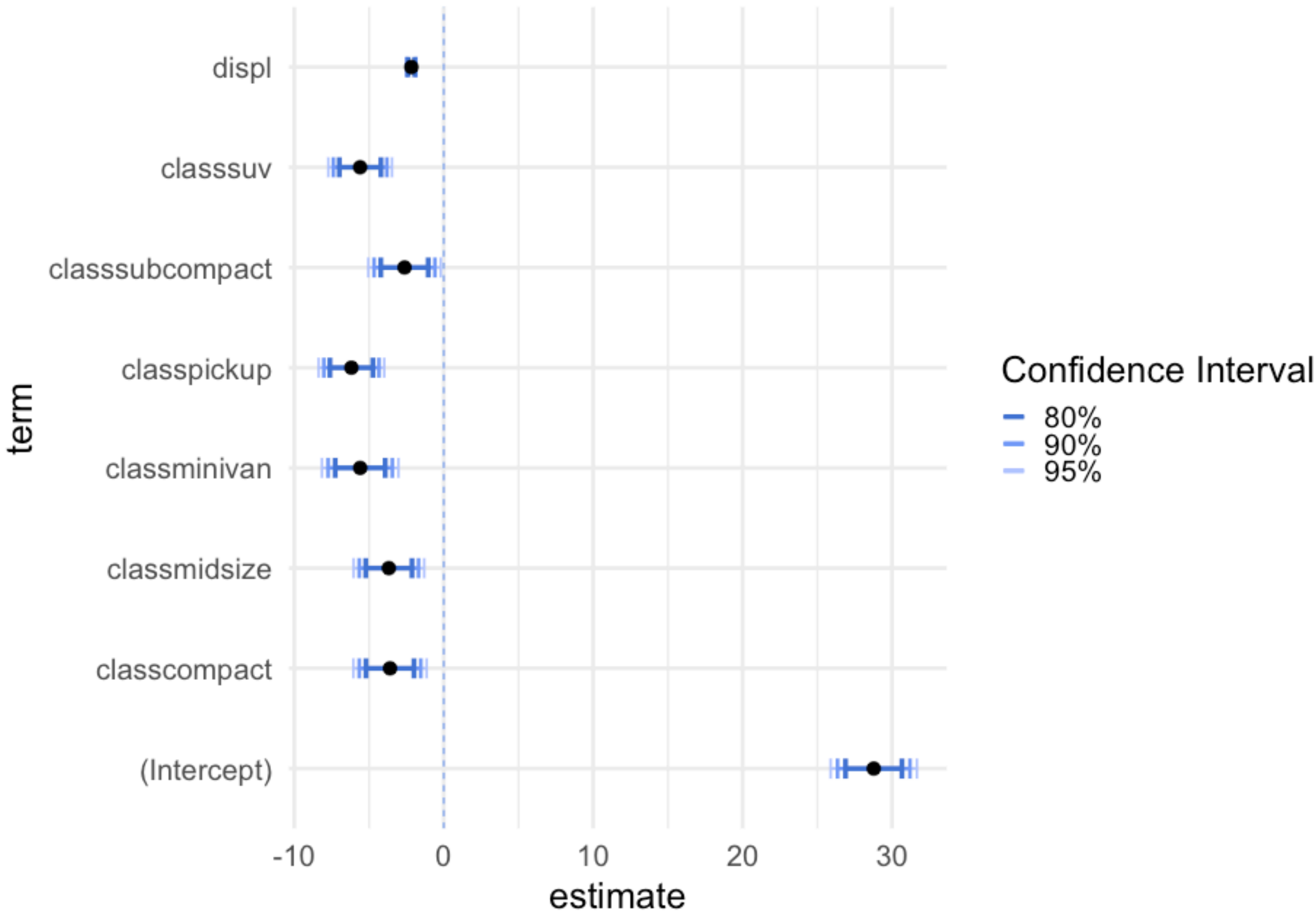
```
p <- ggplot(tidied_m, aes(term, estimate)) +
  geom_hline(yintercept = 0,
             color = "cornflowerblue",
             linetype = 2) +
  geom_errorbar(aes(ymin = estimate + qnorm(.025)*std.error,
                    ymax = estimate + qnorm(.975)*std.error,
                    color = "95%"),
                width = 0.2,
                size = 0.8) +
  geom_errorbar(aes(ymin = estimate + qnorm(.05)*std.error,
                    ymax = estimate + qnorm(.95)*std.error,
                    color = "90%"),
                width = 0.2,
                size = 1.2) + # 90% CI
  geom_errorbar(aes(ymin = estimate + qnorm(.1)*std.error,
                    ymax = estimate + qnorm(.9)*std.error,
                    color = "80%"),
                width = 0.2,
                size = 1.6) # 80% CI
```

```
p
```



30
20
10
0
-10
estimate
(Intercept)
classcompact
classmidsize
classminivan
classpickup
classsubcompact
classsuv
displ
term
colour
80%
90%
95%

```r
p +
  scale_color_manual("Confidence Interval",
                     values = c("#4375D3",
                                lighten("#4375D3", .3),
                                lighten("#4375D3", .6))) +
  geom_point() +
  coord_flip()
```



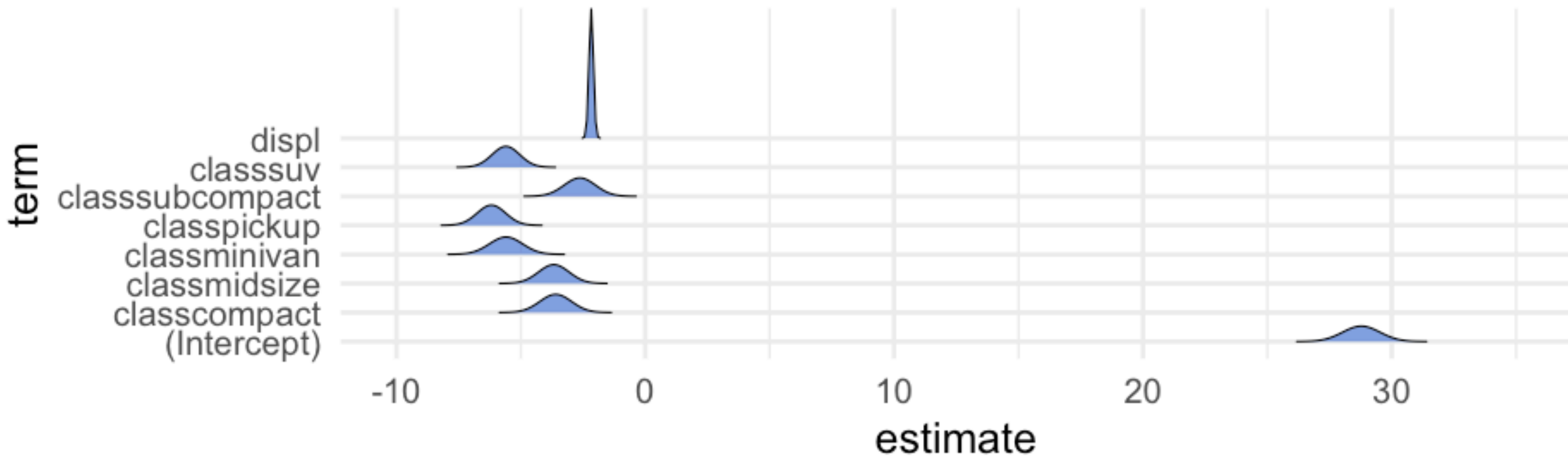
displ
classsuv
classsubcompact
classpickup
classminivan
classmidsize
classcompact
(Intercept)
term
-10
0
10
20
30
estimate
Confidence Interval
80%
90%
95%

Density stripes

```
#devtools::install_github("wilkelab/ungeviz")
library(ungeviz)
ggplot(tidied_m, aes(estimate, term)) +
  stat_confidence_density(aes(moe = std.error),
                          fill = "#4375D3",
                          height = 0.6) +
  xlim(-10, 35) +
  geom_point()
```

Actual densities

```
library(ggridges)
ggplot(tidied_m, aes(estimate, term)) +
  stat_confidence_density(aes(moe = std.error,
                              height = stat(density)),
                          geom = "ridgeline",
                          confidence = 0.95,
                          min_height = 0.001,
                          alpha = 0.7,
                          fill = "#4375D3") +
  xlim(-10, 35)
```

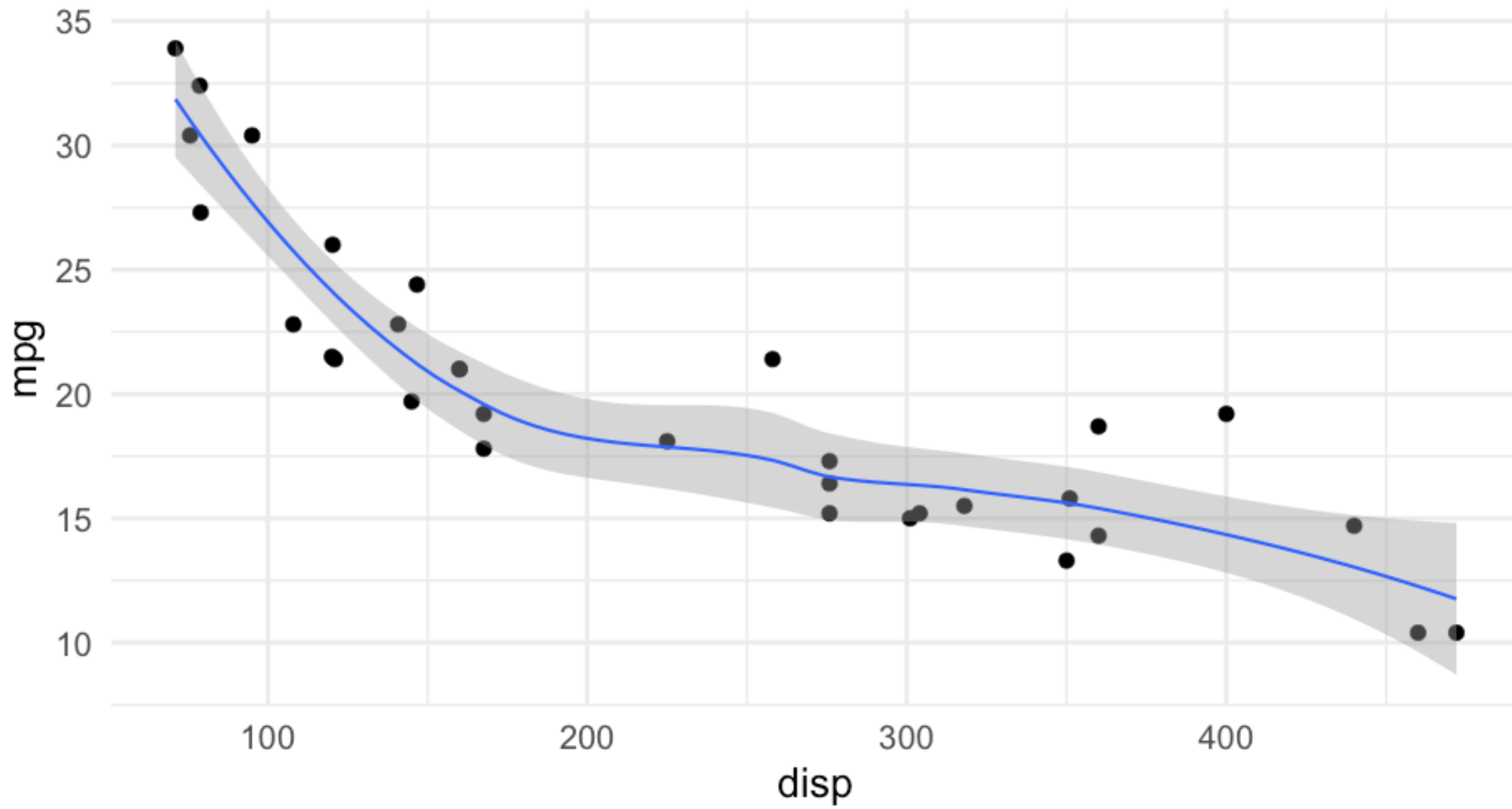


HOPs

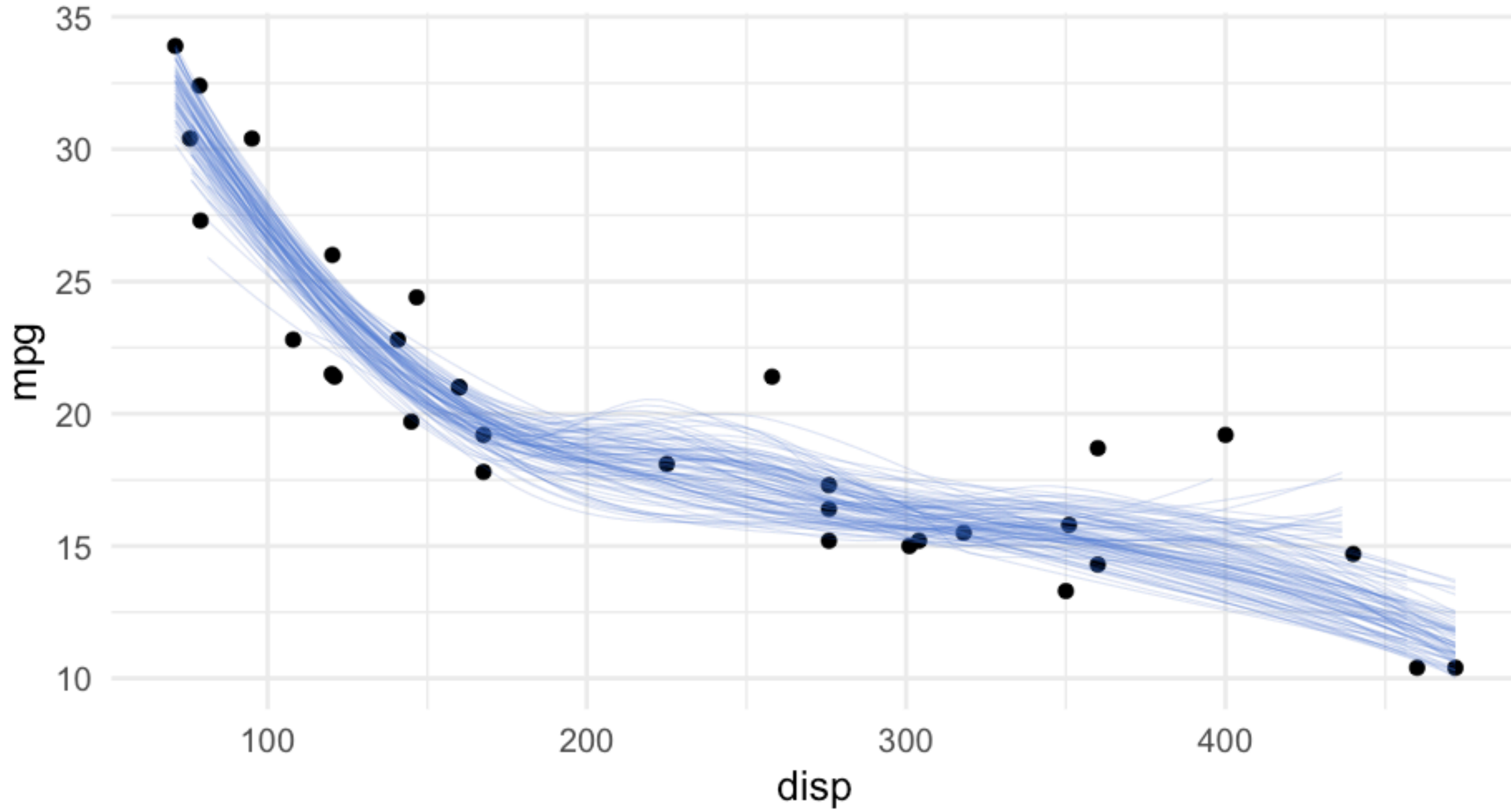
Hypothetical Outcome Plots (and related plots)

Standard regression plot

```
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_smooth()
```



Alternative



How?

Bootstrapping

```r
row_samps <- rerun(100,
      sample(seq_len(nrow(mtcars)),
             nrow(mtcars),
             replace = TRUE))
row_samps
```

```
## [[1]]
##  [1] 31  6 32 12  1 14  2 11 20 10 26 10 22 30 25 25  5 31 19 13 19 20 17  1  4
## [26] 12 12 25 20 21 16 23
##
## [[2]]
##  [1] 11 23 14  1 24 20 10 30 27 24 22 23 25  1 18 18 25  8  8 16 25 19 31 13 11
## [26] 10 21  6 14 14 12 24
##
## [[3]]
##  [1] 27 29 22  5  6  8 14 16  7 13 17 13 21 10  7 21  7 20 30 30  5 10  9  8  4
## [26] 15 16 21 27 23 19  7
##
## [[4]]
##  [1] 16  7  8 28  3 17 13 26  8 30  3 32 20 10  2  6 19 21 11  6 16  9 17  4 25
## [26]  4 27 29 19 21  1 16
##
```

Extract samples

```
d_samps <- map_df(row_samps, ~mtcars[., ], .id = "sample")
head(d_samps)
```

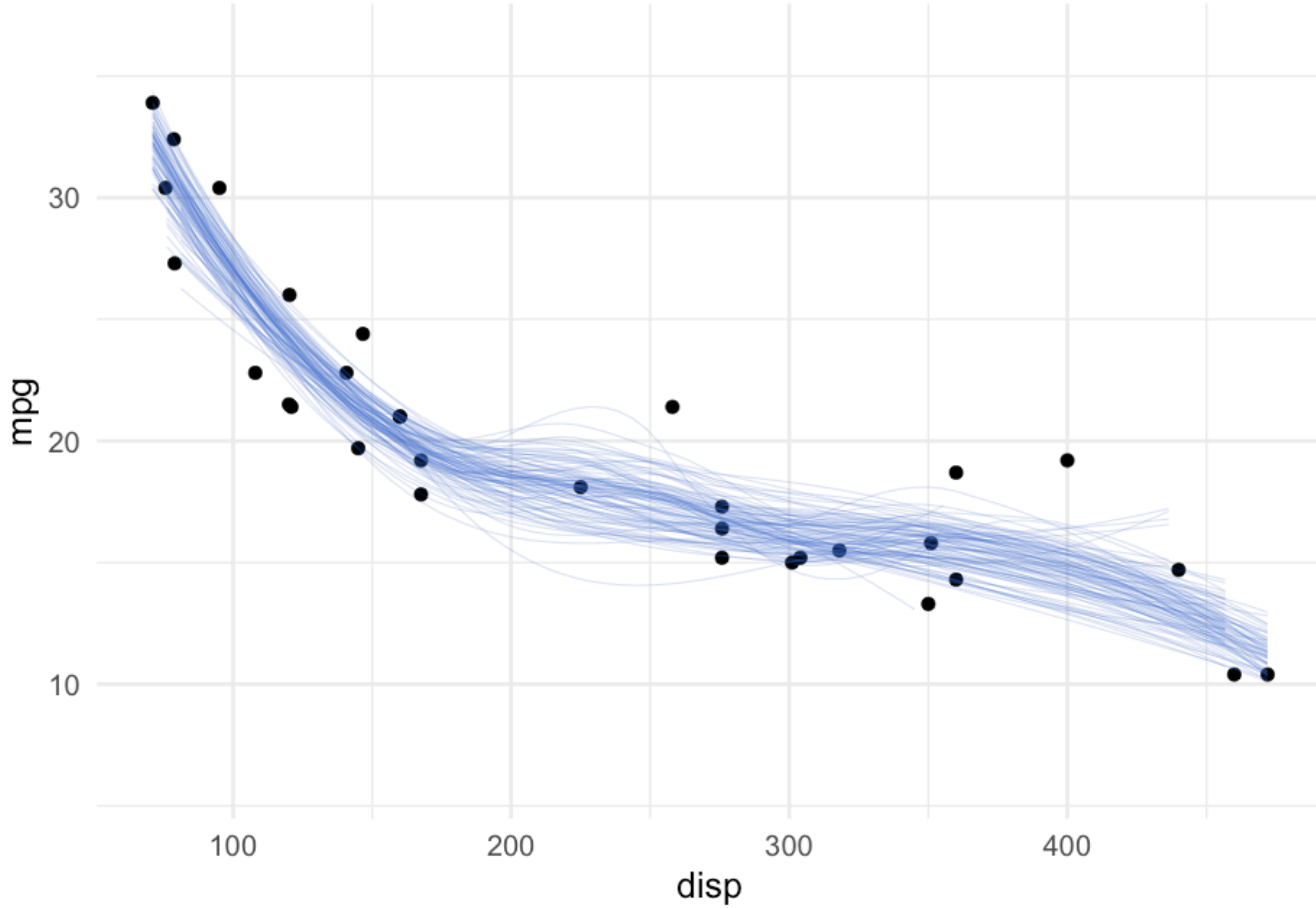
```
##   sample  mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## 1      1 15.0   8 301.0 335 3.54 3.57 14.60  0  1    5    8
## 2      1 18.1   6 225.0 105 2.76 3.46 20.22  1  0    3    1
## 3      1 21.4   4 121.0 109 4.11 2.78 18.60  1  1    4    2
## 4      1 16.4   8 275.8 180 3.07 4.07 17.40  0  0    3    3
## 5      1 21.0   6 160.0 110 3.90 2.62 16.46  0  1    4    4
## 6      1 15.2   8 275.8 180 3.07 3.78 18.00  0  0    3    3
```

```
tail(d_samps)
```

```
##      sample  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 3195    100 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## 3196    100 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## 3197    100 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## 3198    100 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## 3199    100 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 3200    100 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

Plot both data sources

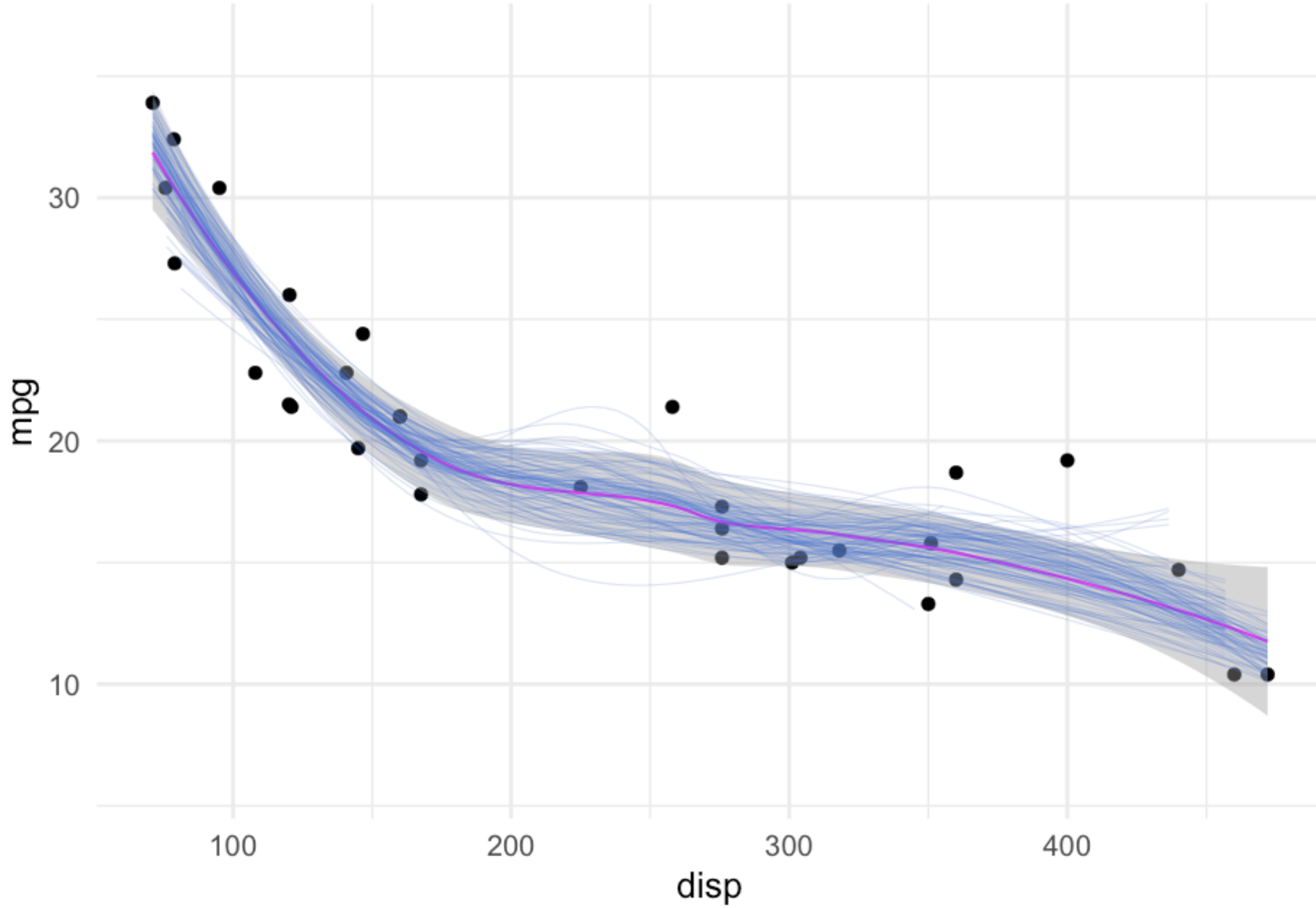
```
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  stat_smooth(aes(group = sample),
              data = d_samps,
              geom = "line",
              color = "#4375D3",
              fullrange = TRUE,
              size = 0.1)
```



mpg
30
20
10
100
200
300
400
disp

Note, they match up

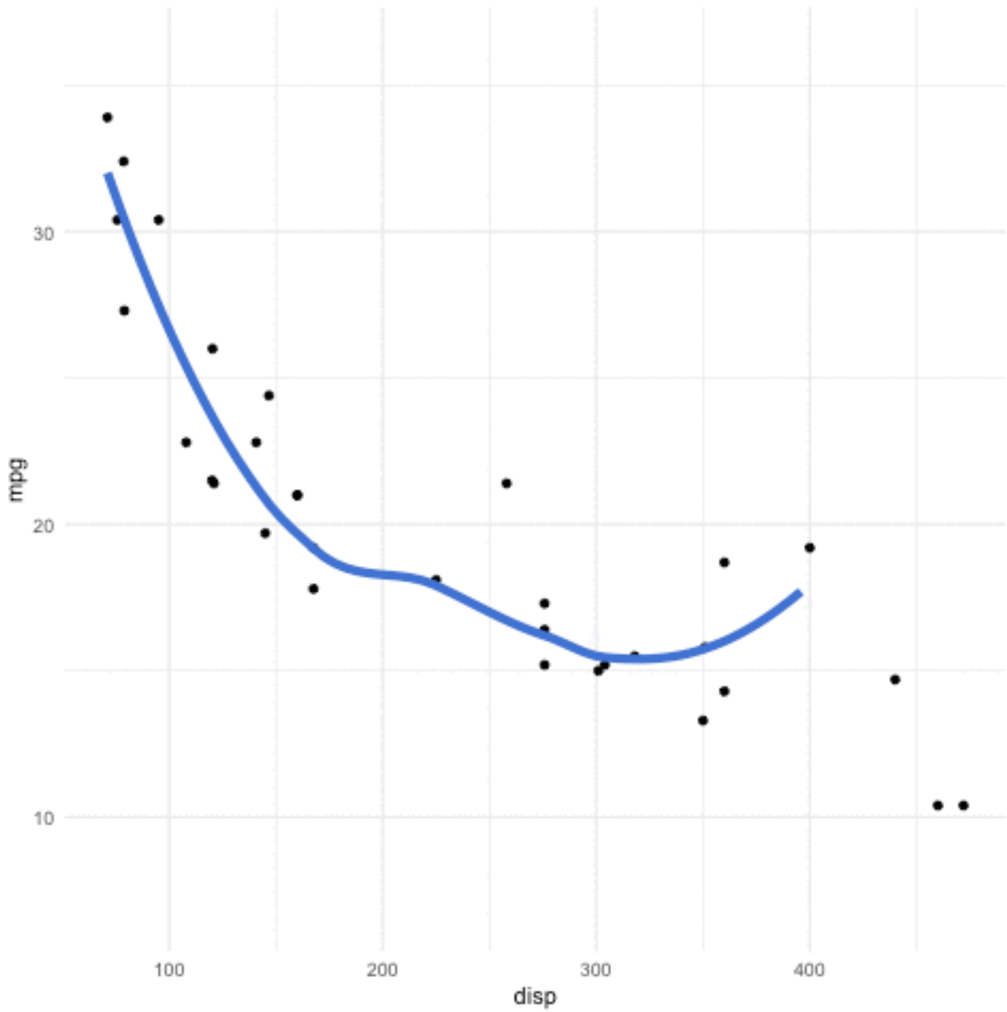
```
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_smooth(color = "magenta") +
  stat_smooth(aes(group = sample),
              data = d_samps,
              geom = "line",
              color = "#4375D3",
              fullrange = TRUE,
              size = 0.1)
```



mpg
30
20
10
100
200
300
400
disp

HOPs

Hops animate the process, so you can't ever really settle on one "truth"

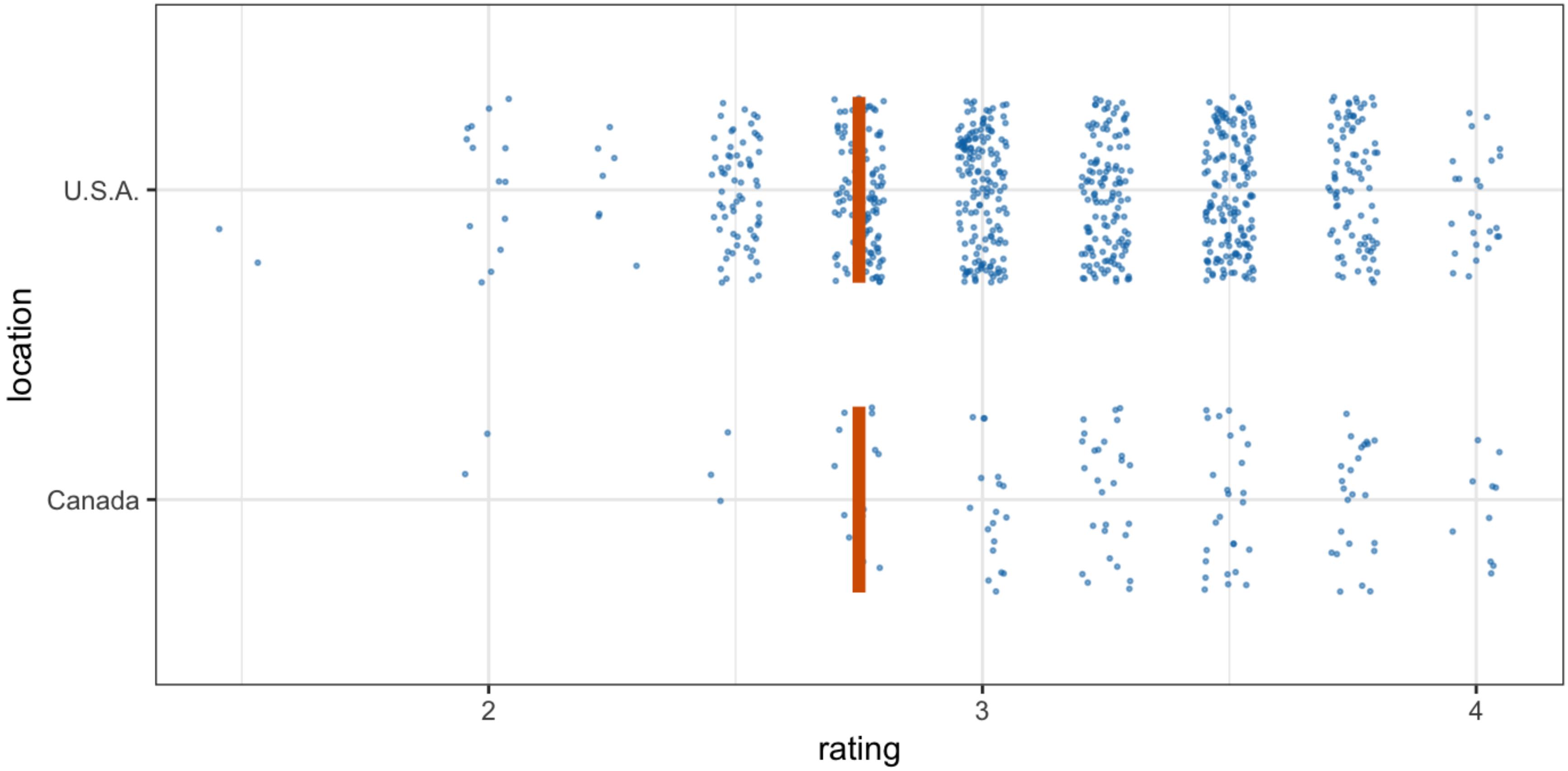


How?

`gganimate::transition_states`

```
library(gganimate)
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  stat_smooth(data = d_samps,
              geom = "line",
              color = "#4375D3",
              fullrange = TRUE) +
  transition_states(sample,
                    transition_length = 0.5,
                    state_length = 0.5) +
   ease_aes('linear') # Smoother transitions
```

Another example



Another examples

From Dr. Kay again

Conclusions

- Lots of tools at your disposal (perhaps so many it can be difficult to choose)
- Consider animations if it fits the medium

- Do try to communicate uncertainty whenever possible

- I'd recommend checking out Clause Wilke's talk from `rstudio::conf(2019L)`, where he talks about the ungeviz package (which looks really cool and promising and I hope to play around with more in the future).

Next time

- Tables with the gt package and a few others
- Fonts with `showtext` and/or `extrafont`