Visualizing uncertainty

Daniel Anderson

Week 7

Data viz in the wild

Cassie

Amy

Diana

Errol and Mandi on deck

Agenda

- Finish up plot refinement slides
- Common ways of visualizing uncertainty
  - And how to implement them with {ggplot2}
- Framing uncertainty as relative frequencies
  - Discrete probabilities
  - Non-discrete probabilities
- Understanding standard errors
  - Non-standard ways of visualizing SEs
- HOPs (pretty quick)

Agenda (continued)

Assuming we still have time (I think we will), we will also at least introduce **tables**. We probably will not have time for fonts, but I have slides for them as well in case we do.

- Be comfortable with the basics of `gt`
  - create a table
  - format columns
  - create spanner heads
  - etc.
- Understand how to use additional fonts (if you so choose)

Expectations for today

Similar to last class, we will have times for you to practice, and times where I will ask you to follow along.

Please make sure R is up and running.

A quick warning

This will be a little more stats focused than basically any lecture we have through the first three courses of this sequence

Finishing plot refinement

Uncertainty

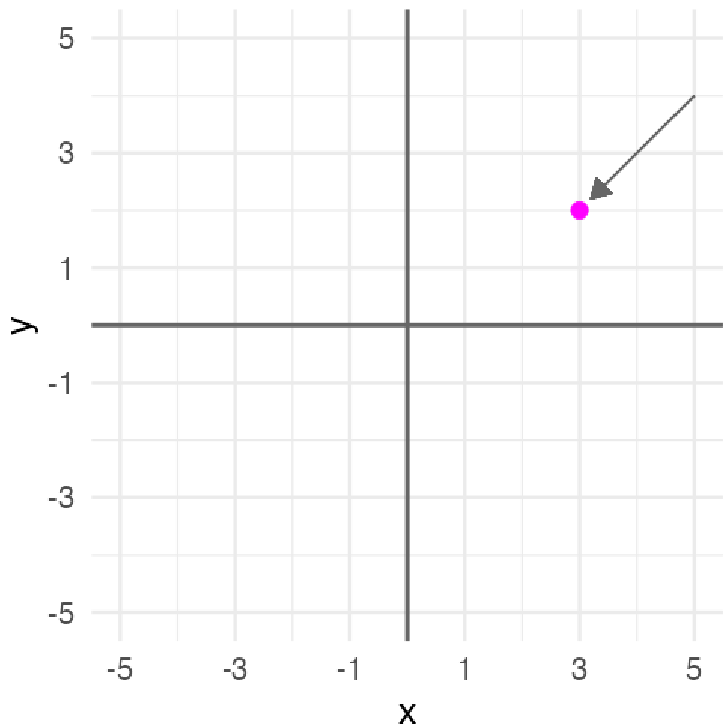
Learning objectives

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

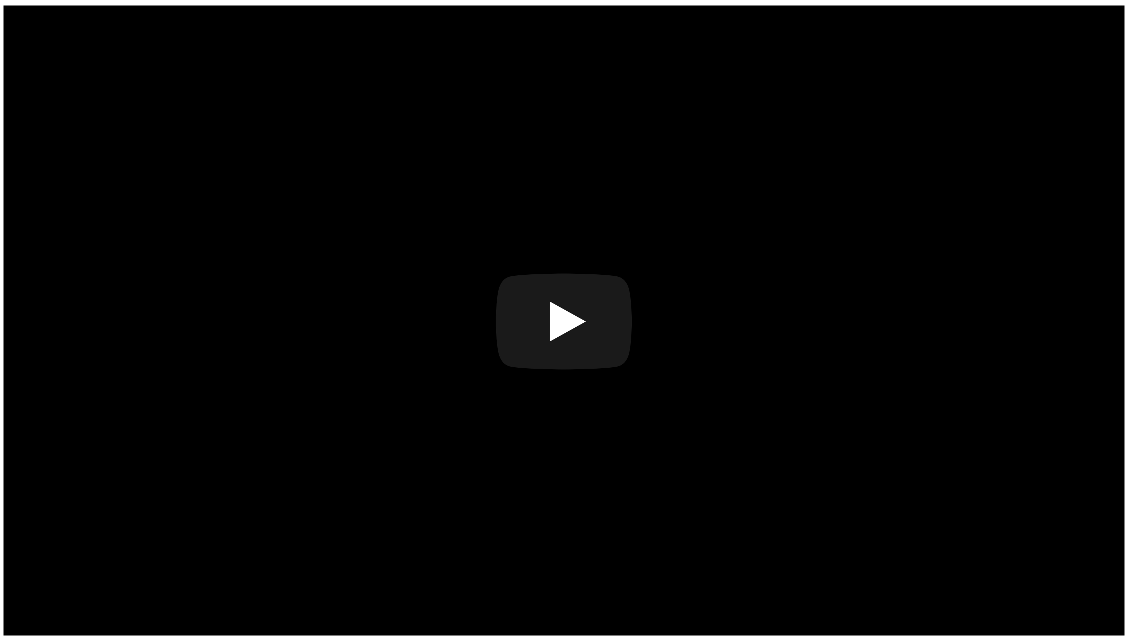
2. Understand how to implement basic methods, and the resources available to you to implement more advanced methods

The primary problem

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



Let's have Dr. Kay explain

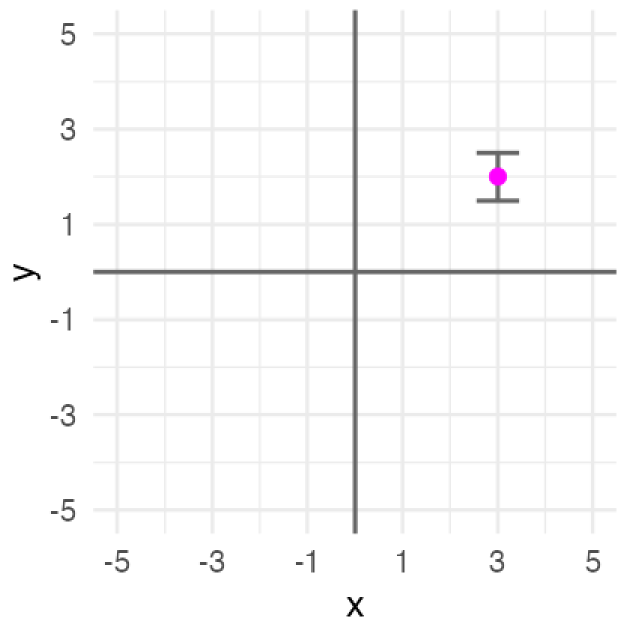


Some secondary problems

- 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`

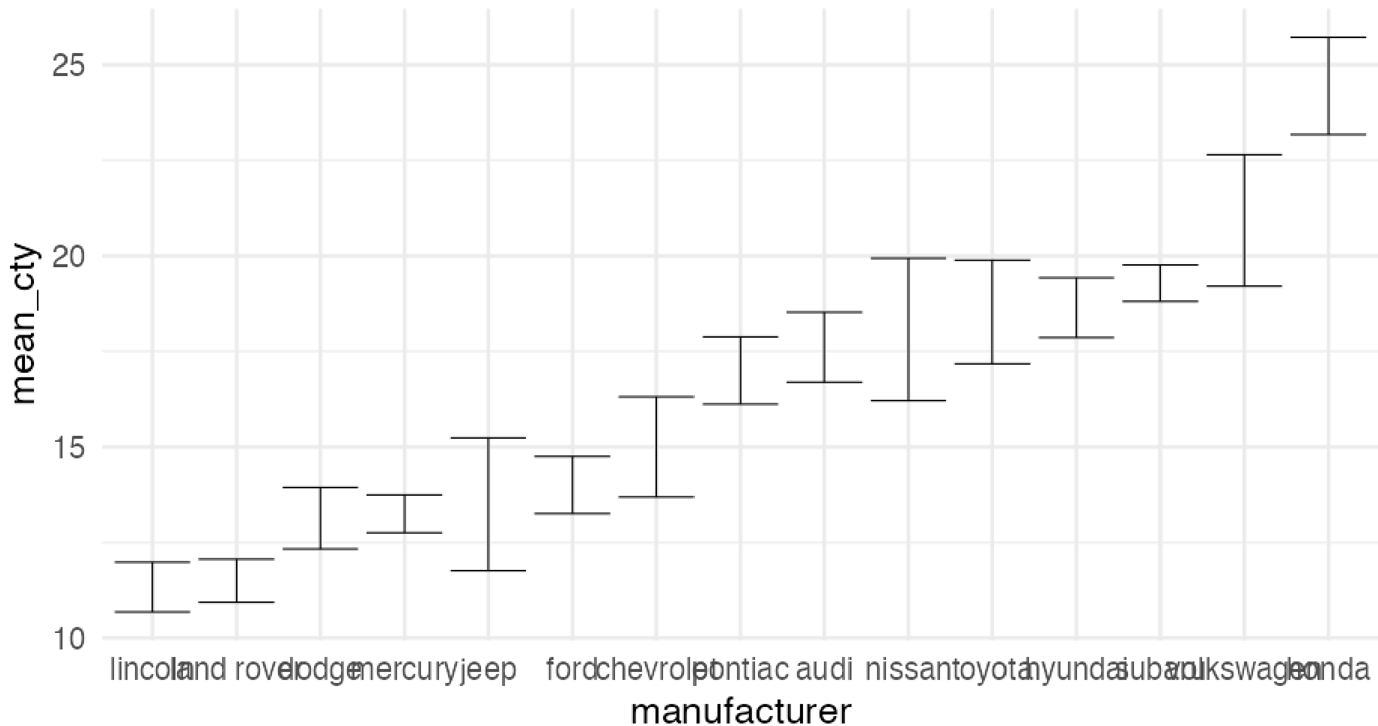
- Requires `xmin` and `xmax`

Example

```
mpg_by_man <- mpg %>%
  group_by(manufacturer) %>%
  summarize(mean_cty = mean(cty),
            se_cty = sd(cty) / sqrt(n()))

head(mpg_by_man)
```

```
## # A tibble: 6 × 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
```

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

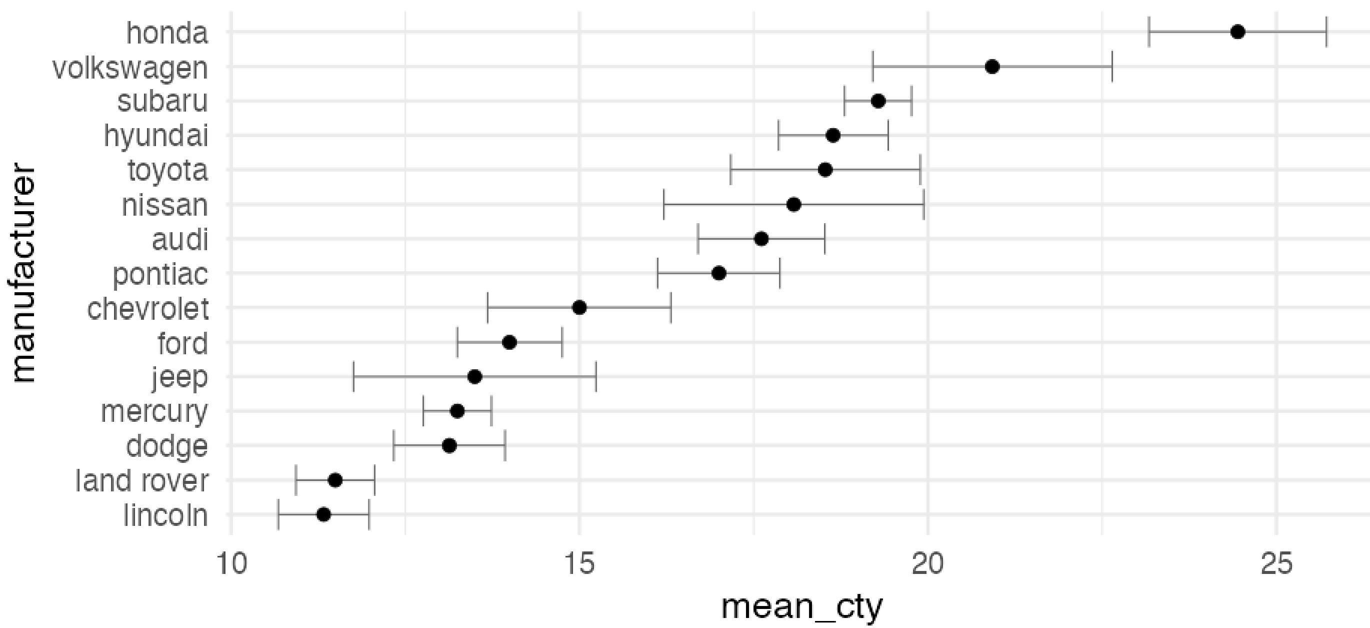


Put points on top

Not under

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

```r
mpg_by_man %>%
  mutate(manufacturer = fct_reorder(manufacturer, mean_cty)) %>%
  ggplot(aes(mean_cty, manufacturer)) +
    geom_errorbarh(
      aes(xmin = mean_cty - 1.96 * se_cty,
          xmax = mean_cty + 1.96 * se_cty),
      color = "gray40"
    ) +
   geom_point()
```



Practice

- Use the Palmer penguins dataset
- Plot the mean **`bill_length_mm`** for each species with 95% confidence intervals

```r
library(palmerpenguins)
penguins
```

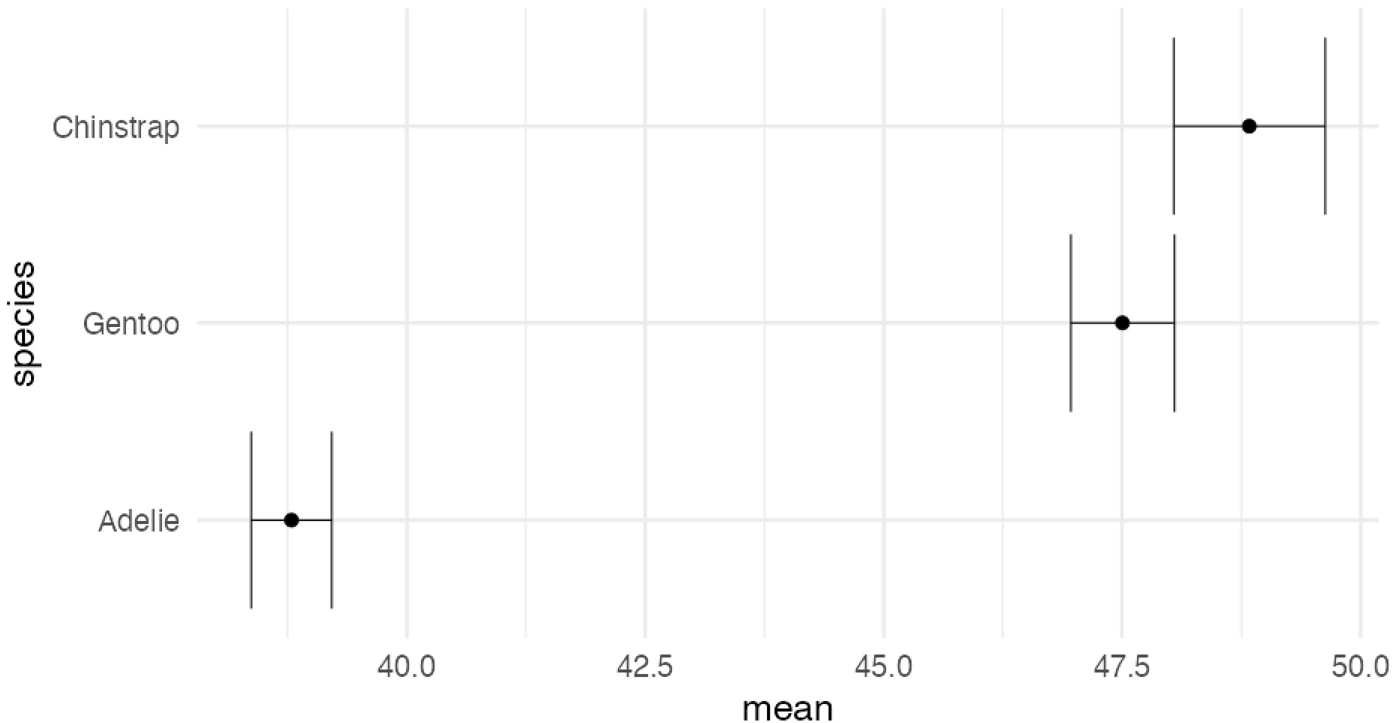
```
## # A tibble: 344 × 8
##   species island    bill_length_mm
##   <fct>   <fct>              <dbl>
## 1 Adelie  Torgersen           39.1
## 2 Adelie  Torgersen           39.5
## 3 Adelie  Torgersen           40.3
## 4 Adelie  Torgersen           NA  
## 5 Adelie  Torgersen           36.7
## 6 Adelie  Torgersen           39.3
## # … with 338 more rows, and 5 more variable
## #   bill_depth_mm <dbl>,
## #   flipper_length_mm <int>,
## #   body_mass_g <int>, sex <fct>, year <int
```

05:00

Solution

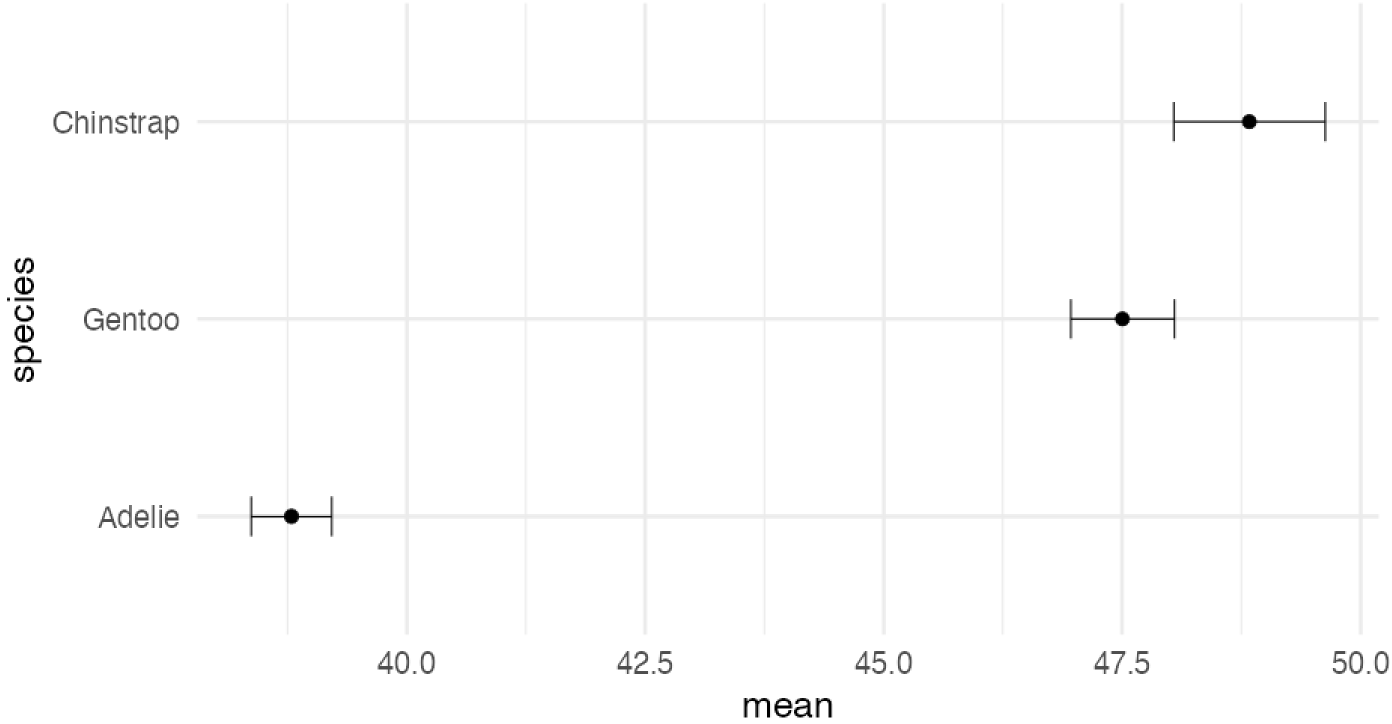
Or one possible solution

```
mn_se <- penguins %>%
  group_by(species) %>%
  summarize(
    mean = mean(bill_length_mm, na.rm = TRUE),
    se = sd(bill_length_mm, na.rm = TRUE) / sqrt(n()),
    lower = mean + qnorm(0.025) * se,
    upper = mean + qnorm(0.975) * se,
  ) %>%
  mutate(species = fct_reorder(species, mean))

ggplot(mn_se, aes(mean, species)) +
  geom_errorbarh(aes(xmin = lower, xmax = upper)) +
  geom_point()
```


Chinstrap
Gentoo
Adelie
species
40.0
42.5
45.0
47.5
50.0
mean

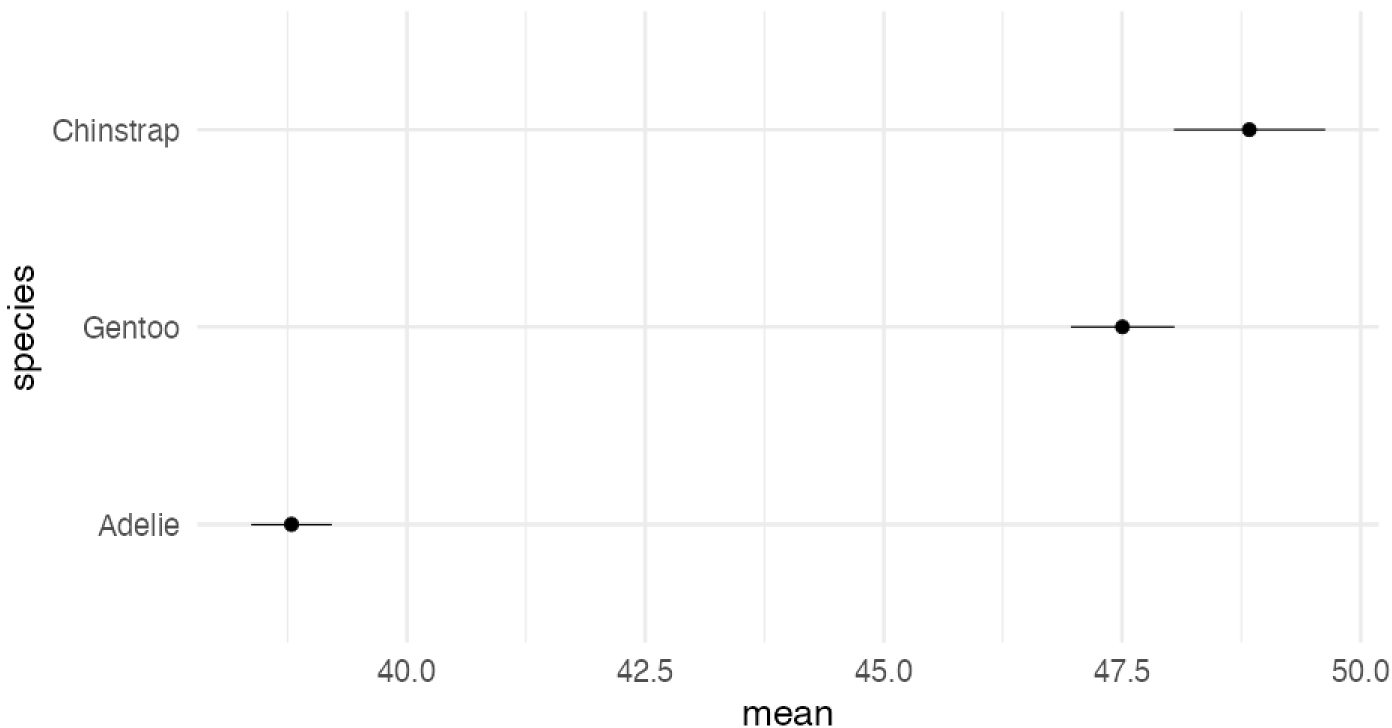
Change the height

```
ggplot(mn_se, aes(mean, species)) +
  geom_errorbarh(aes(xmin = lower, xmax = upper), height = 0.2) +
  geom_point()
```



Slight variant

```
ggplot(mn_se, aes(mean, species)) +
  geom_linerange(aes(xmin = lower, xmax = upper)) +
  geom_point()
```



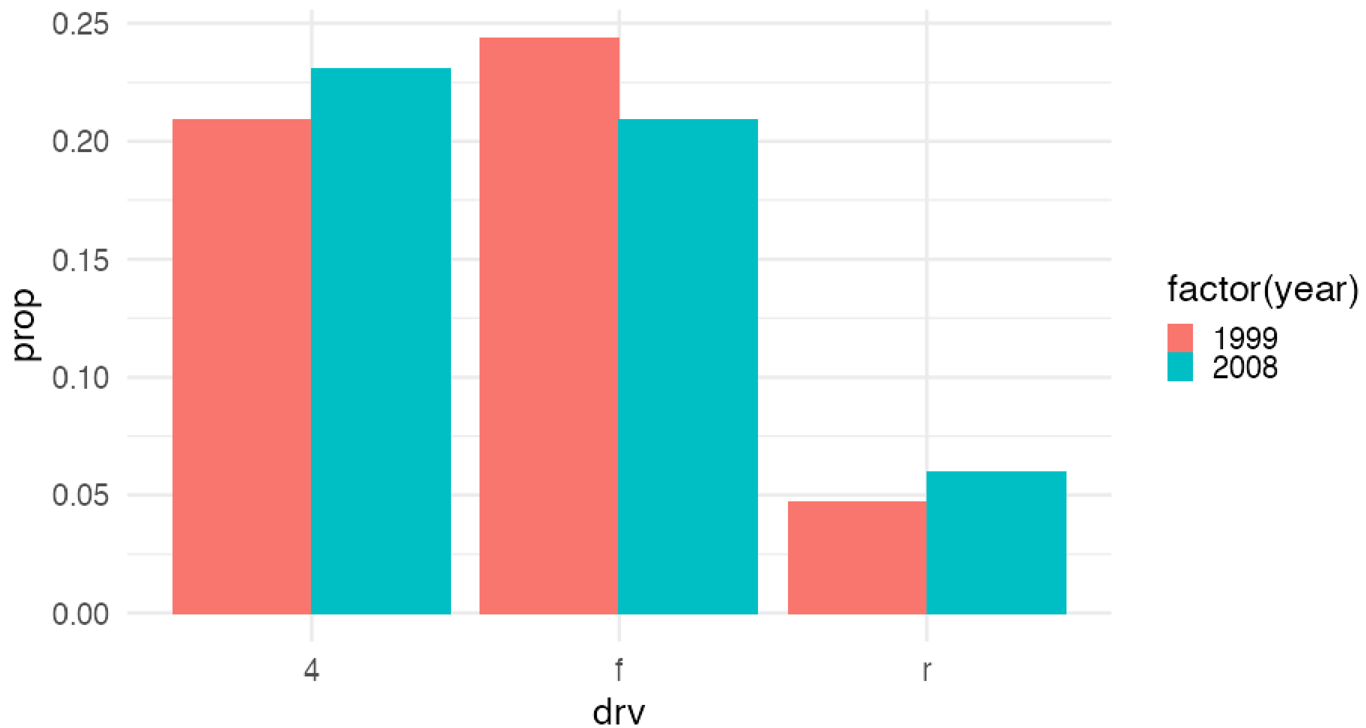
Dodging

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

head(props)
```

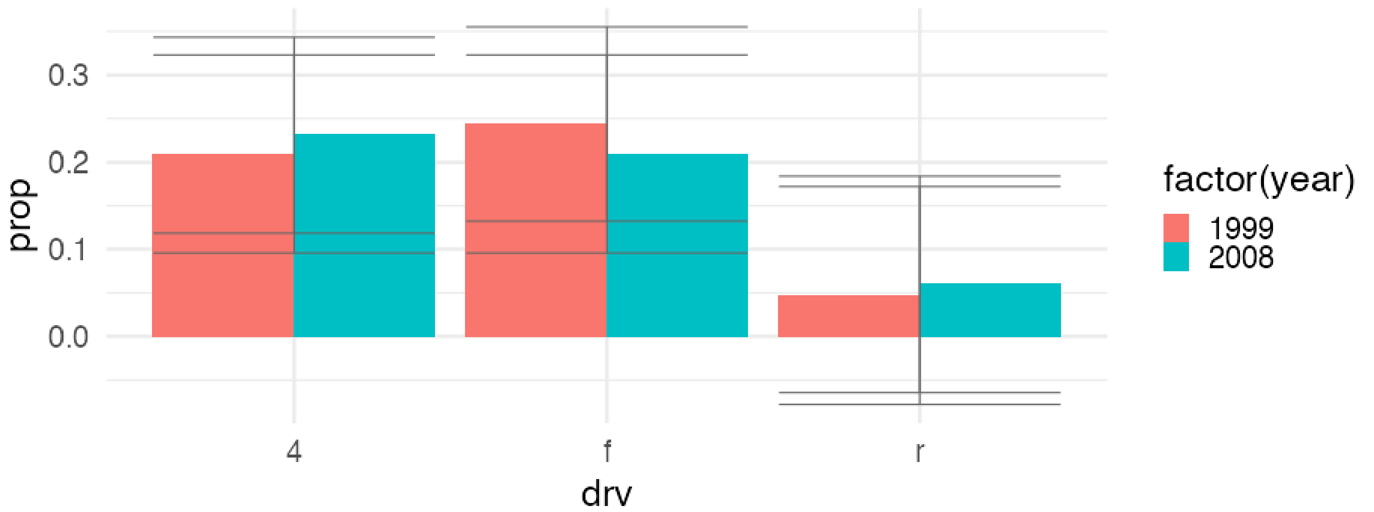
```
## # A tibble: 6 × 5
##   drv    year     n       prop    prop_se
##   <chr> <int> <int>      <dbl>      <dbl>
## 1 4      1999    49 0.2094017  0.05812594
## 2 4      2008    54 0.2307692  0.05733508
## 3 f      1999    57 0.2435897  0.05685528
## 4 f      2008    49 0.2094017  0.05812594
## 5 r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

Bar plot

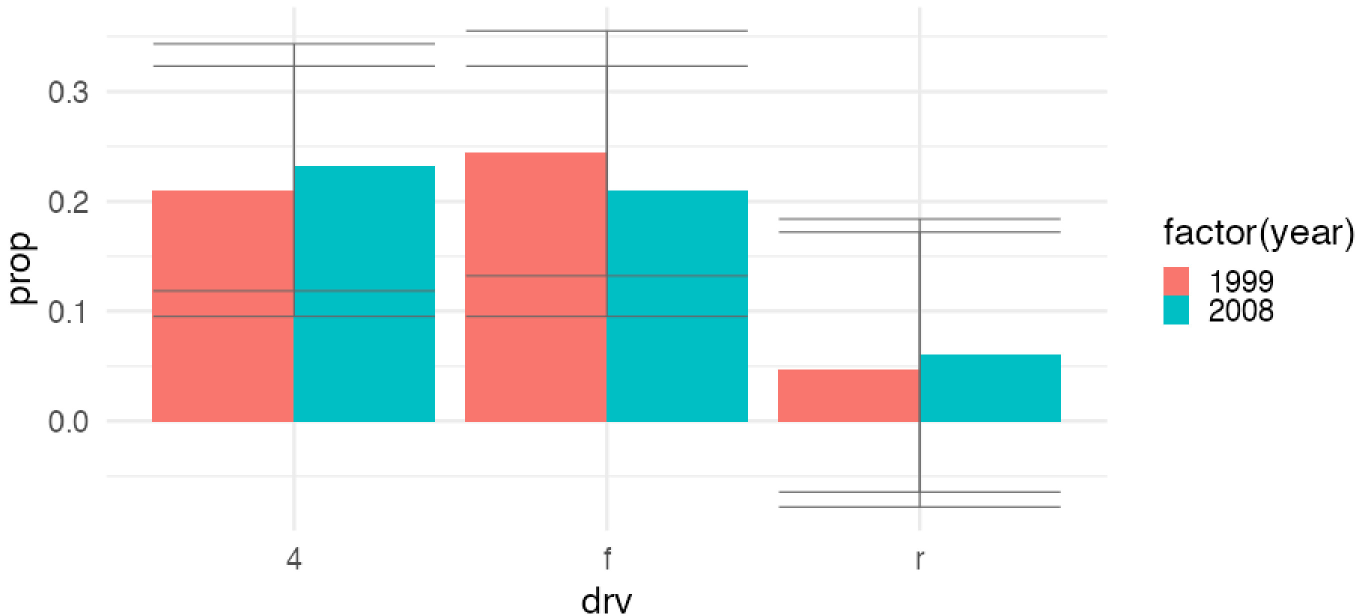
```
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = "dodge")
```



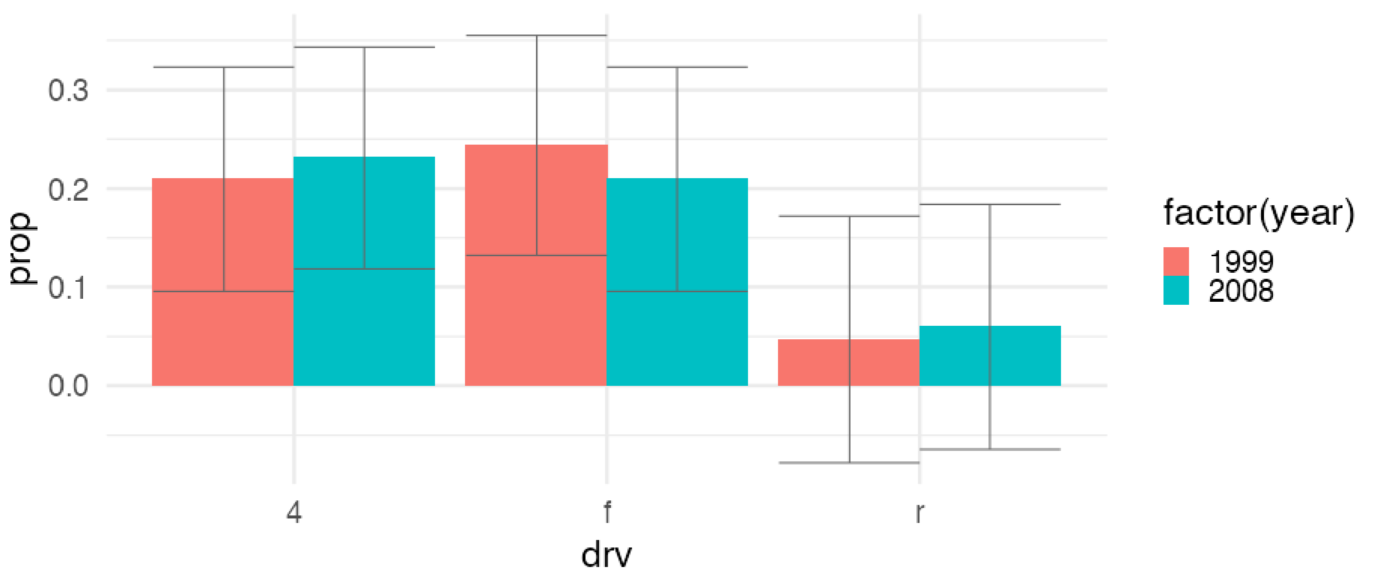
```
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = "dodge") +
  geom_errorbar(
    aes(ymin = prop - 1.96 * prop_se,
        ymax = prop + 1.96 * prop_se),
    color = "gray40"
  )
```



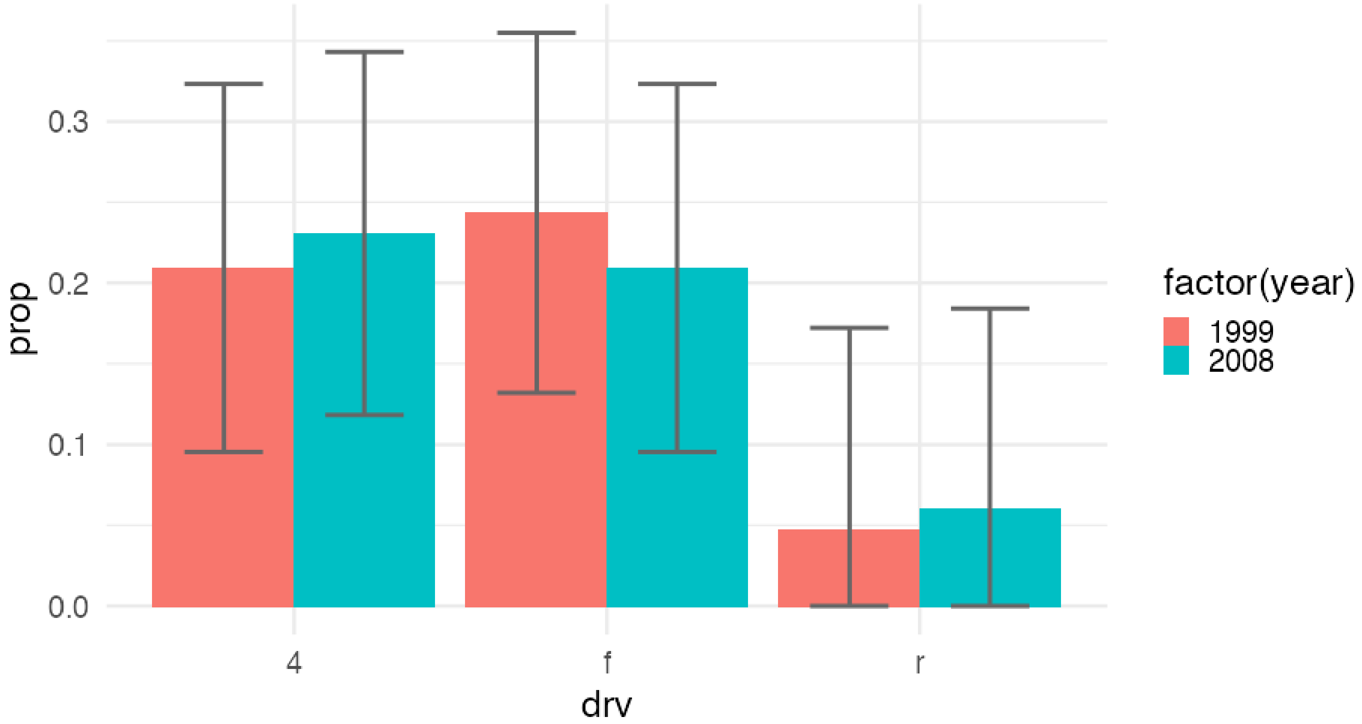
```r
pd <- position_dodge(.9)
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = pd) +
  geom_errorbar(
    aes(ymin = prop - 1.96 * prop_se,
        ymax = prop + 1.96 * prop_se),
    color = "gray40",
    position = pd
  )
```



```
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
  )
```



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

Explain error bars

Error bars could represent any of the following

- Standard deviation (of the data)
- Standard error (of the estimate)
- Confidence interval (of the estimate)

Possibly through a caption, make sure your audience knows what your represent.

If Confidence intervals, state the interval size (e.g., 68%, 90%, 95%)

Thinking about uncertainty

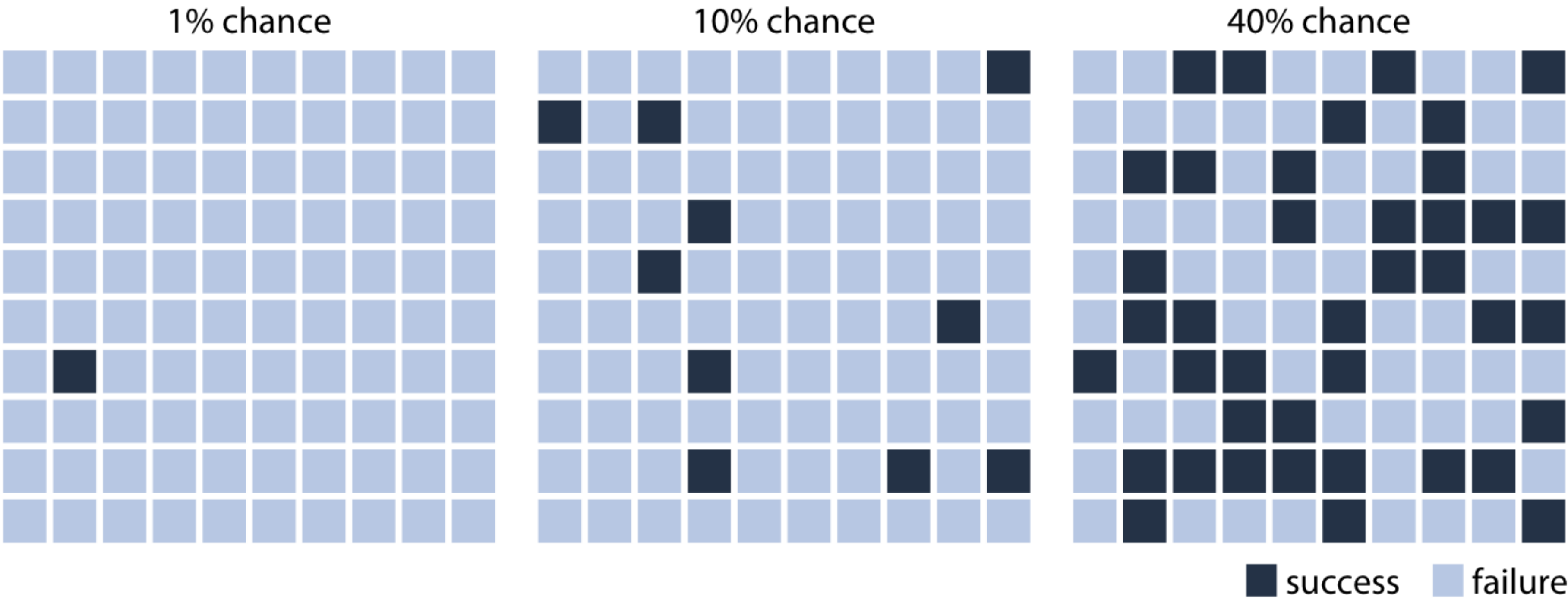
Uncertainty means exactly what it sounds like – we are not 100% sure.

- 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

```
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
```

```
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") +
  theme_void()
```

Create occurrence rate

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

How?

- Plenty of options, here's one

Please follow along

Consider 10%

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

# n to sample
n_sample <- nrow(grid) * .10

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

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

```r
length(samp)
```

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

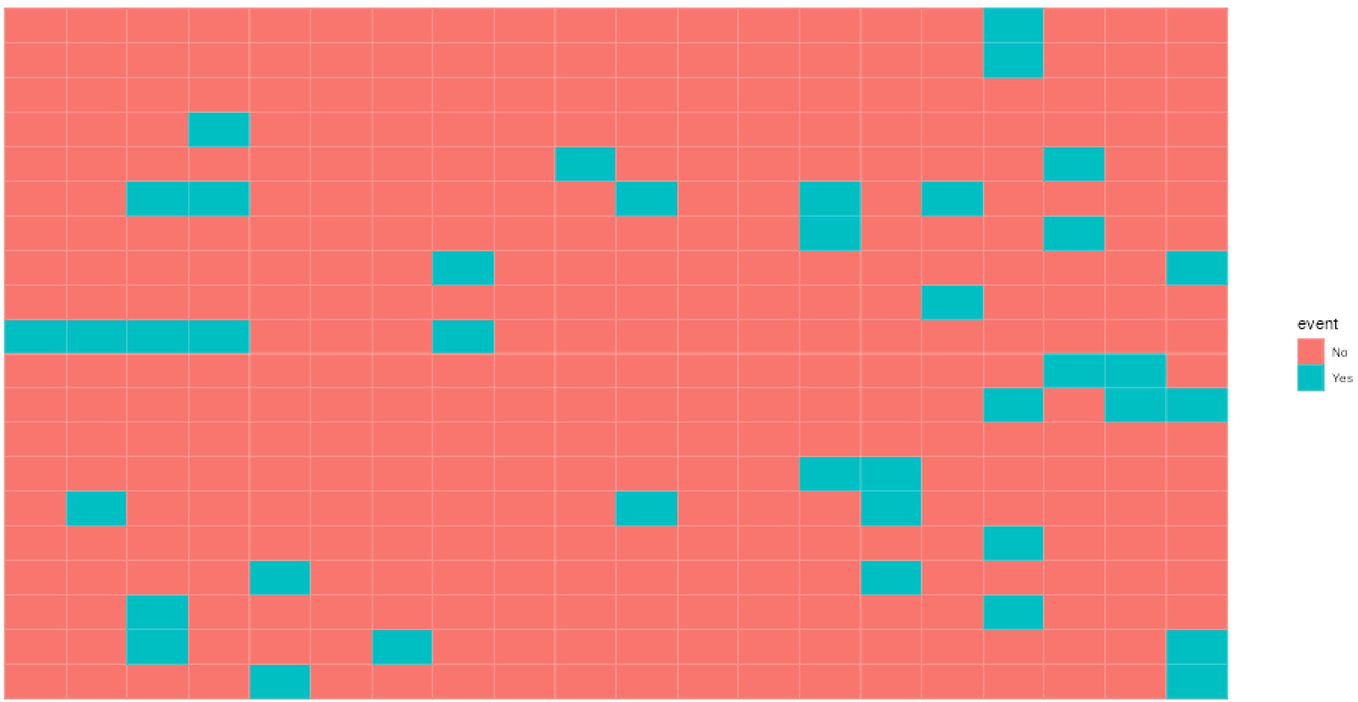
Create the variable

```
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
```

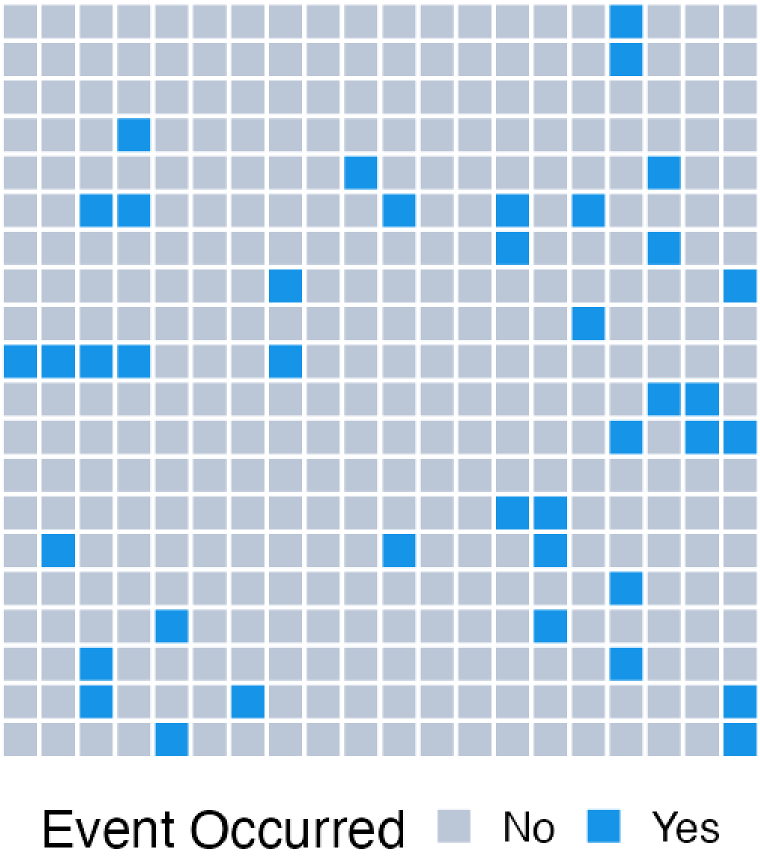
Fill in

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

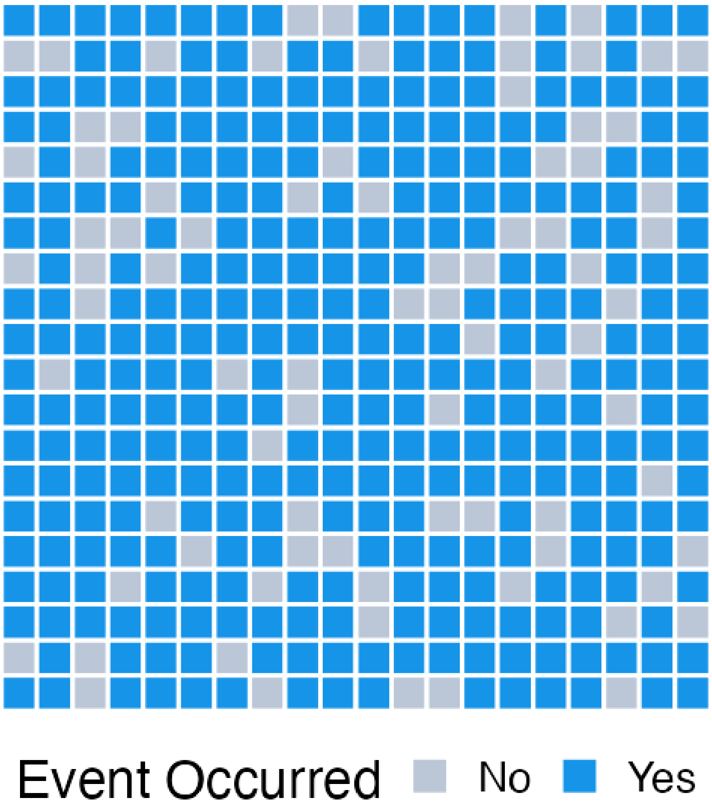


Customize

```r
library(colorspace)
ggplot(grid, aes(x, y)) +
  geom_tile(aes(fill = event), color = "white", size = 1.4) +
  scale_fill_manual(
    name = "Event Occurred",
    values = c(
      desaturate(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"))
```

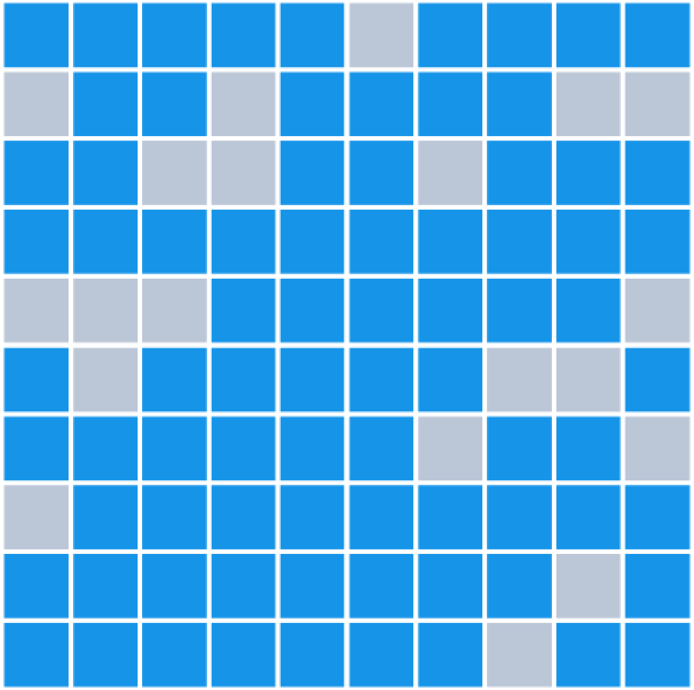

Event Occurred
No
Yes

Chance of rain



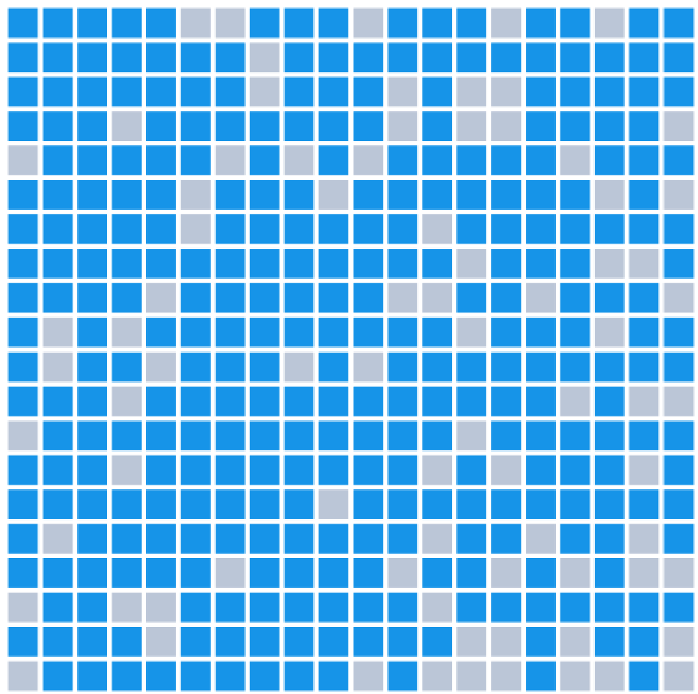
Vary grid size

10 x 10



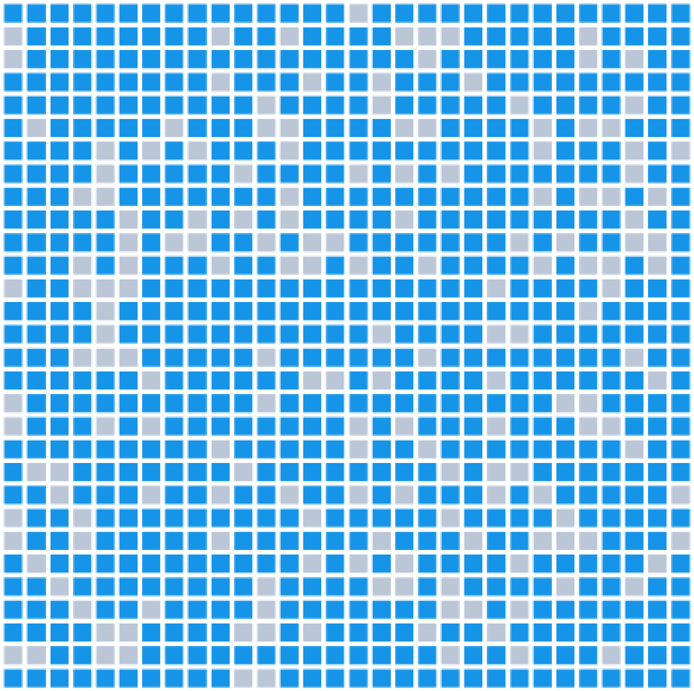
Vary grid size

20 x 20



Vary grid size

30 x 30



Optimal size might depend on your venue

Practice

- Create a new grid showing 41% likelihood of something happening
- Vary the fill colors and grid size
- Compare with a neighbor – what do you like about yours vs. theirs?

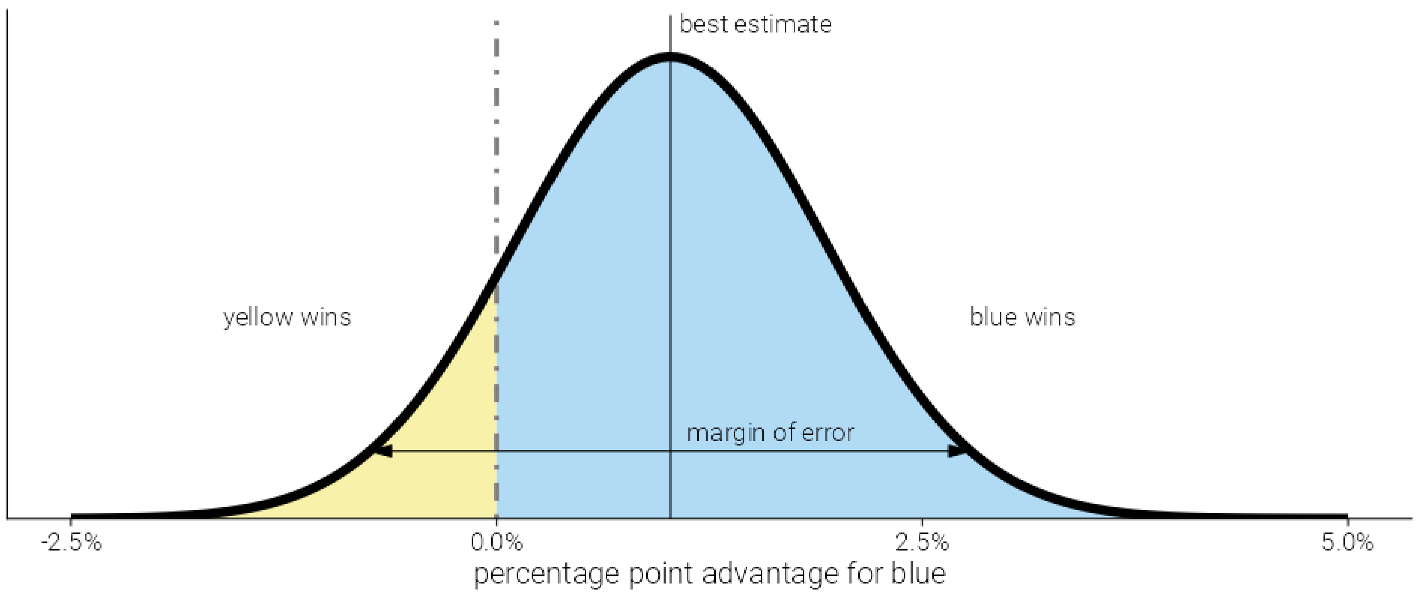
05:00

Non–discrete probabilities

Hypothetical

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

Who will win and **by how much**?



A bit of math

Our distribution was defined by $\boldsymbol{\mu = 1.02}$ and $\boldsymbol{sd = 0.9}$.

- What's the chance the end result is **below zero**? (yellow wins)

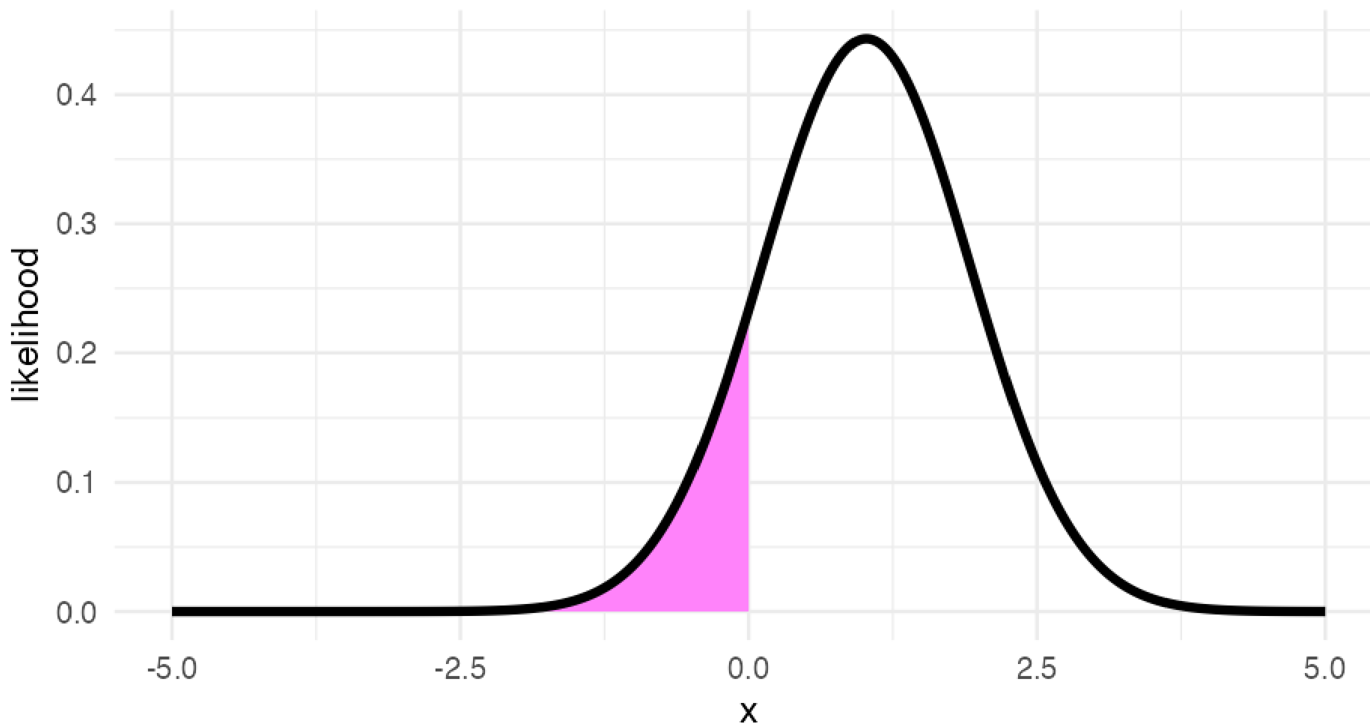
The hard way

Calculate the exact probability

The distribution

```
x <- seq(-5, 5, 0.001) # some sample data
likelihood <- dnorm(x, 1.02, 0.9) # probability of occuring
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] 0.129
```

Easier: Simulate

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

```
##
## FALSE  TRUE
## 0.128 0.872
```

Problem

This approach works okay, but...

- We're not great an interpreting probabilities
- Not great at infering a probability from a density

Discretized plot

```r
ppoints(50)
```

```
##  [1] 0.01 0.03 0.05 0.07 0.09 0.11 0.13 0.15
##  [9] 0.17 0.19 0.21 0.23 0.25 0.27 0.29 0.31
## [17] 0.33 0.35 0.37 0.39 0.41 0.43 0.45 0.47
## [25] 0.49 0.51 0.53 0.55 0.57 0.59 0.61 0.63
## [33] 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79
## [41] 0.81 0.83 0.85 0.87 0.89 0.91 0.93 0.95
## [49] 0.97 0.99
```

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

```
##  [1] -1.07371 -0.67271 -0.46037 -0.30821
##  [5] -0.18668 -0.08388  0.00625  0.08721
##  [9]  0.16125  0.22989  0.29422  0.35504
## [13]  0.41296  0.46847  0.52195  0.57373
## [17]  0.62408  0.67321  0.72133  0.76861
## [21]  0.81521  0.86126  0.90690  0.95226
## [25]  0.99744  1.04256  1.08774  1.13310
## [29]  1.17874  1.22479  1.27139  1.31867
## [33]  1.36679  1.41592  1.46627  1.51805
## [37]  1.57153  1.62704  1.68496  1.74578
## [41]  1.81011  1.87875  1.95279  2.03375
## [45]  2.12388  2.22668  2.34821  2.50037
```

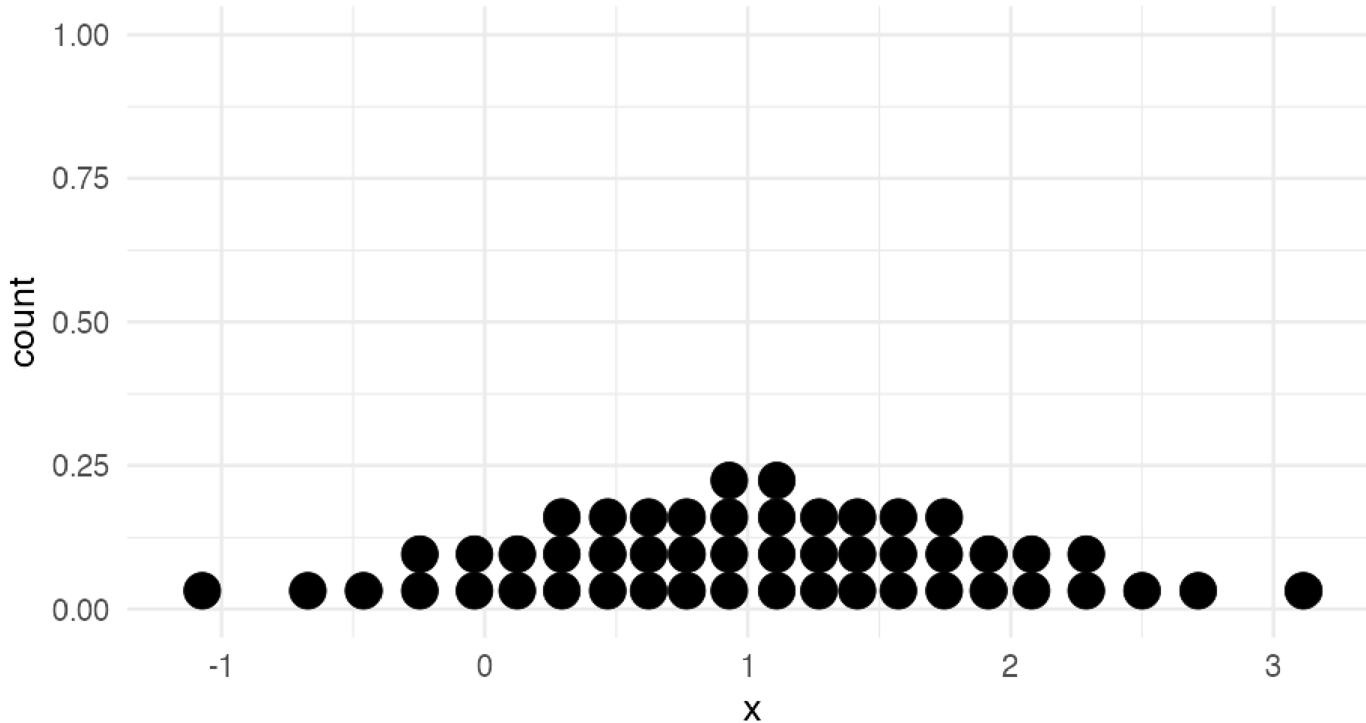
```r
discretized <- data.frame(
  x = qnorm(ppoints(50), 1.02, 0.9)
  ) %>%
  mutate(winner = ifelse(x <= 0, "#b1daf4", "#f8f1a9"))

head(discretized)
```

```
##         x  winner
## 1 -1.0737 #b1daf4
## 2 -0.6727 #b1daf4
## 3 -0.4604 #b1daf4
## 4 -0.3082 #b1daf4
## 5 -0.1867 #b1daf4
## 6 -0.0839 #b1daf4
```

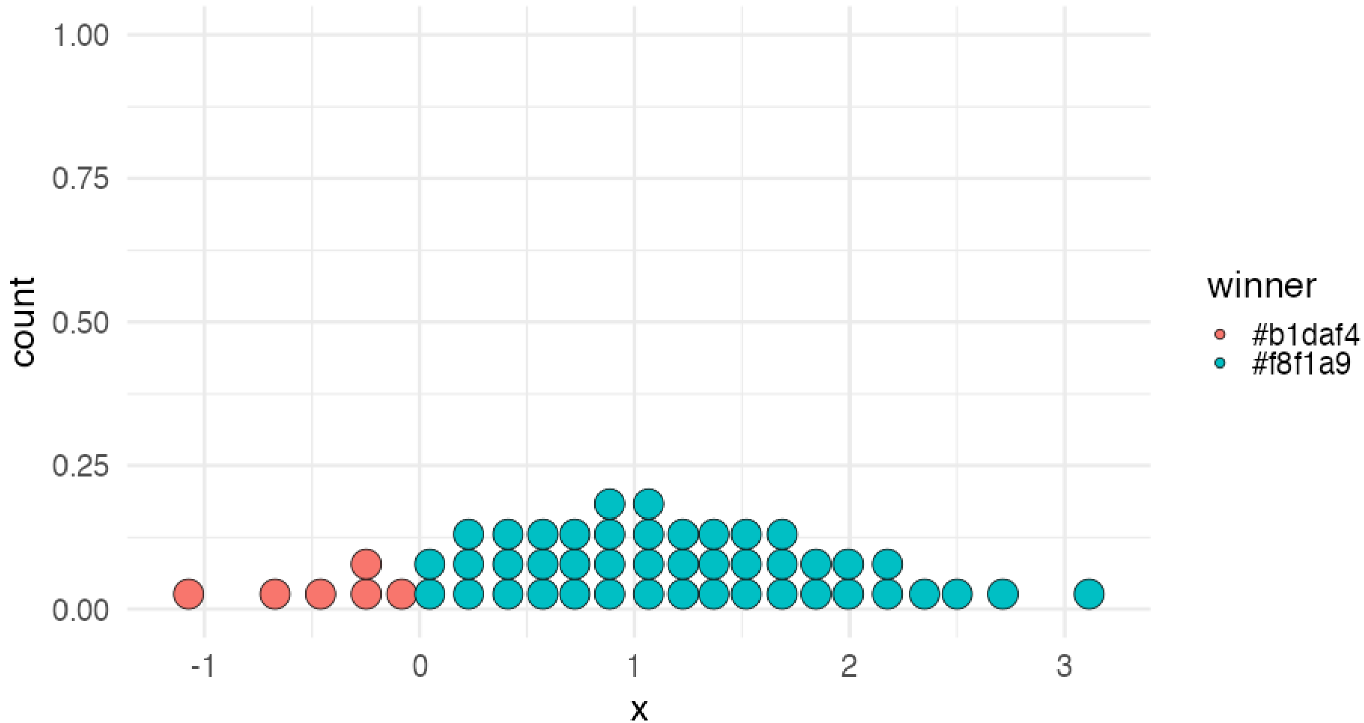
```r
tail(discretized)
```

```
##       x  winner
## 45 2.12 #f8f1a9
## 46 2.23 #f8f1a9
## 47 2.35 #f8f1a9
## 48 2.50 #f8f1a9
## 49 2.71 #f8f1a9
## 50 3.11 #f8f1a9
```

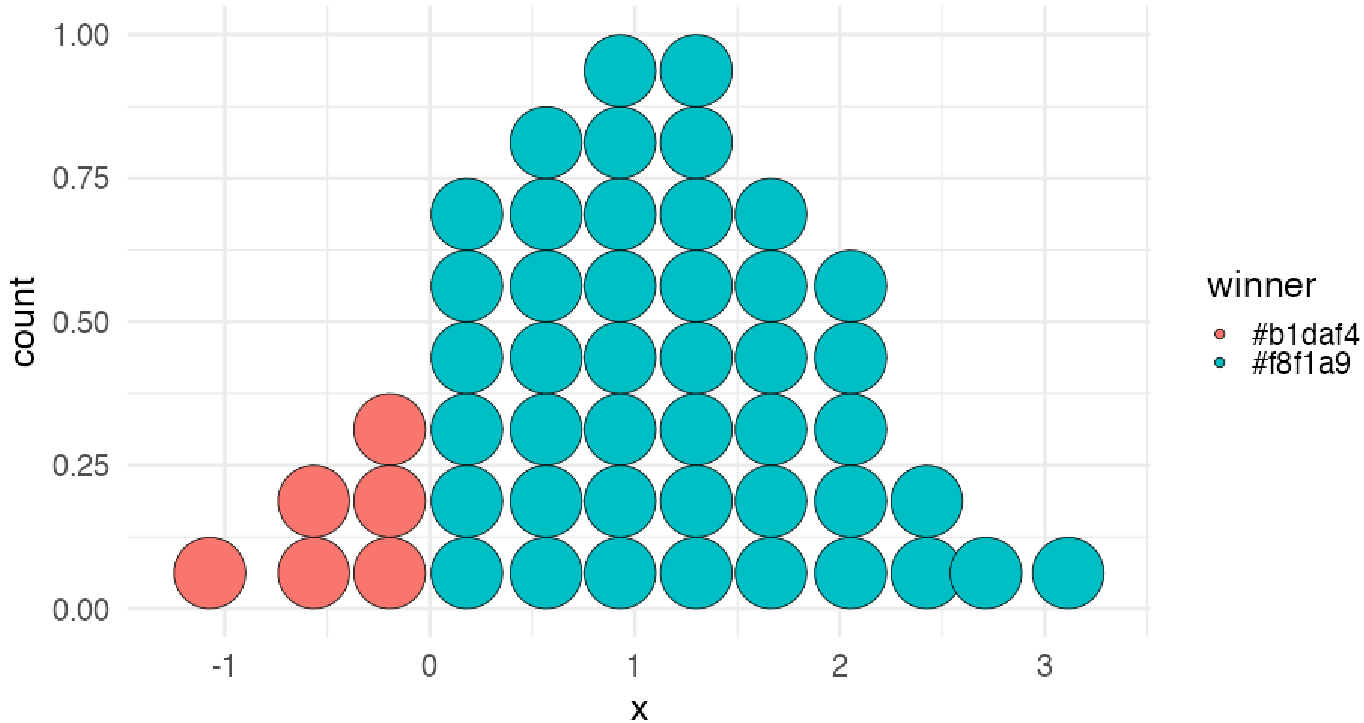
```r
ggplot(discretized, aes(x)) +
  geom_dotplot()
```



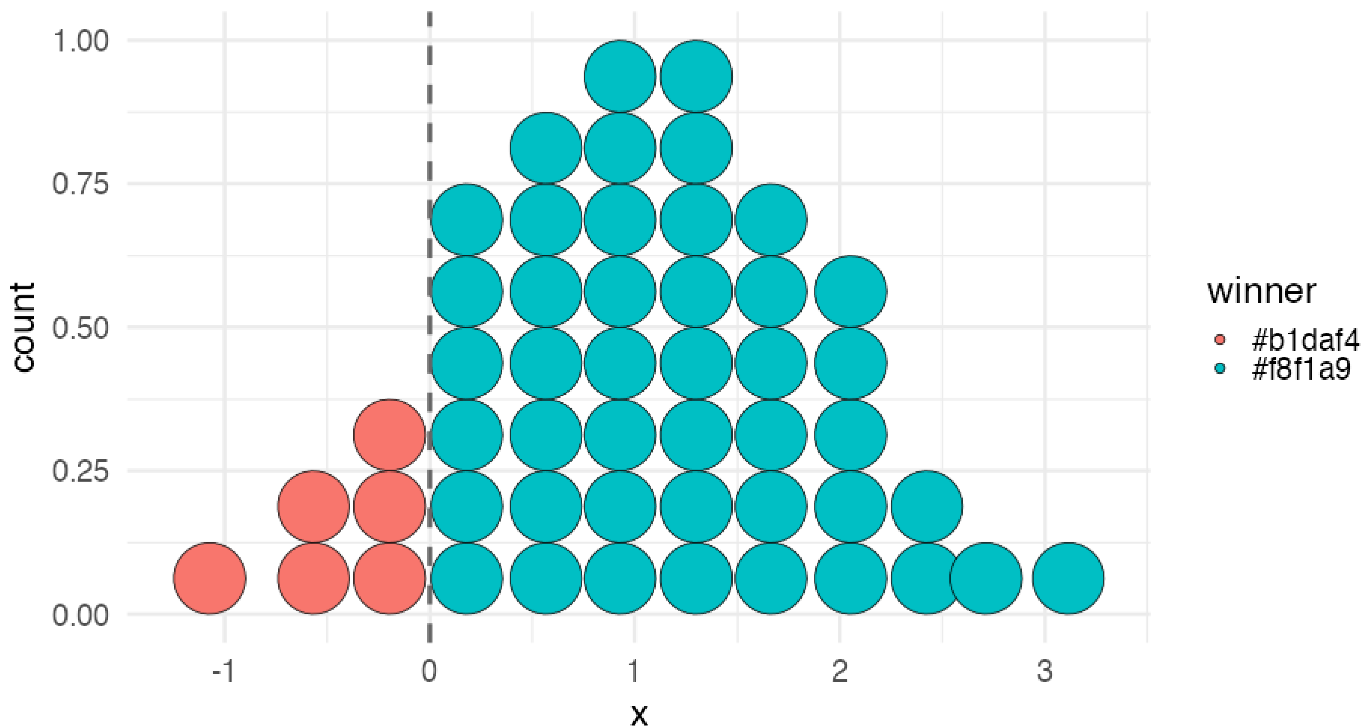
```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner))
```



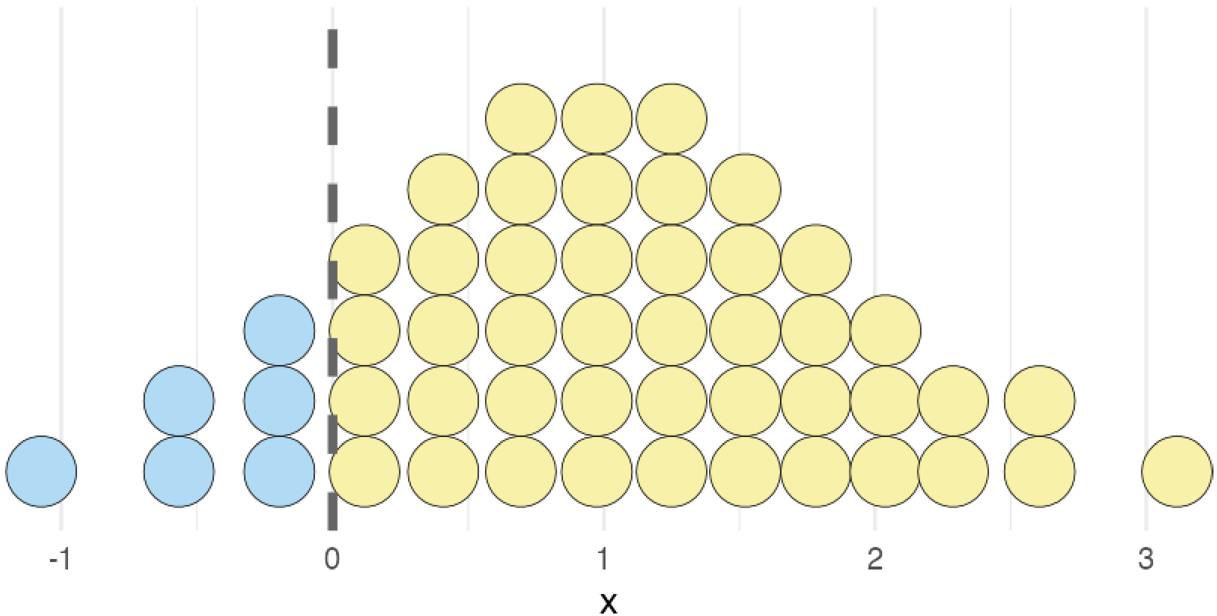
```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.35)
```



```r
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.35) +
  geom_vline(xintercept = 0,
             color = "gray40",
             linetype = "dashed",
             size = 1.5)
```

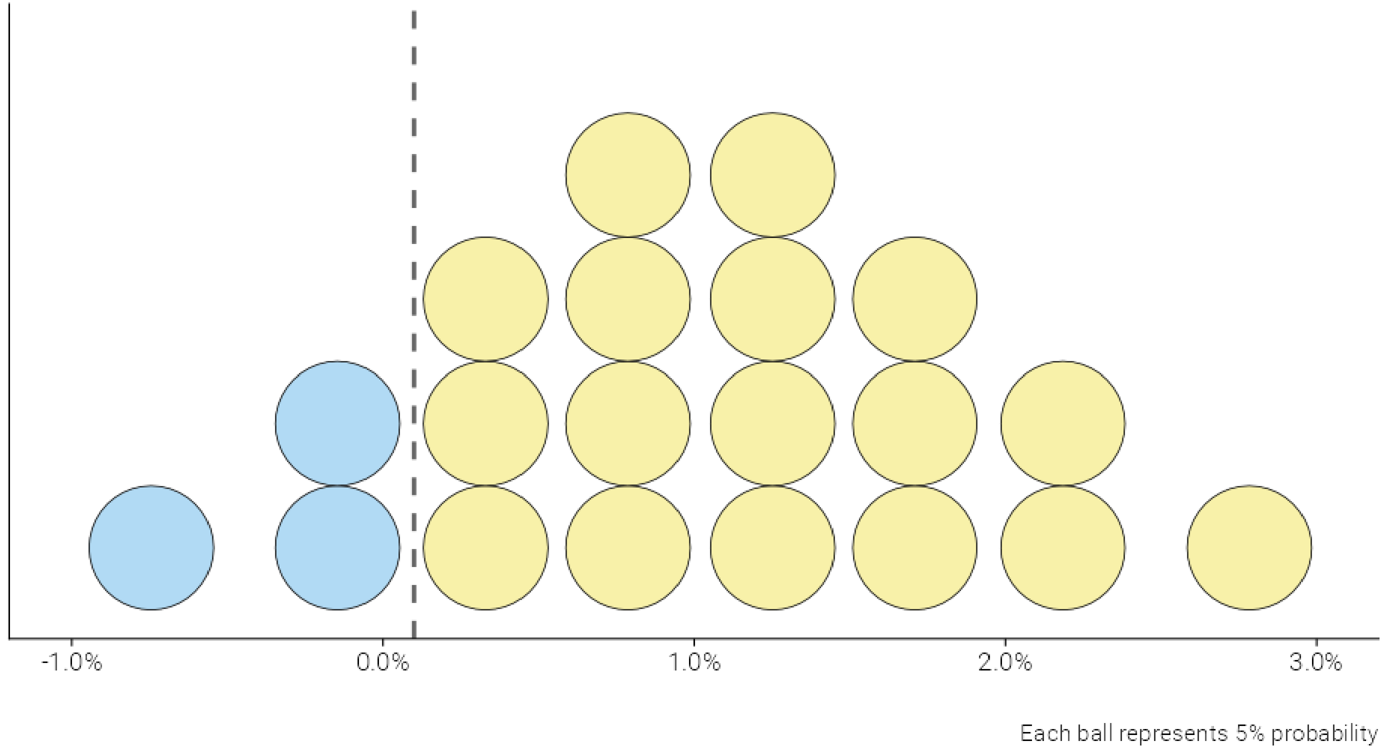


```
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.26) +
  geom_vline(xintercept = 0,
             color = "gray40",
             linetype = 2,
             size = 3) +
  scale_fill_identity(guide = "none") +
  scale_y_continuous(name = "",
                     breaks =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



Probs too many though

```
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 (hopefully) 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]  94.4  97.7 115.6 100.7 101.3 117.2 104.6
##   [8]  87.3  93.1  95.5
```

- Calculate the mean

```
mean(samp10a)
```

```
## [1] 101
```

Do it a second time

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

```
##  [1] 112.2 103.6 104.0 101.1  94.4 117.9 105.0
##  [8]  80.3 107.0  95.3
```

```
mean(samp10b)
```

```
## [1] 102
```

Do it a bunch of times

```r
samples <- replicate(1000, rnorm(10, mean = 100, sd = 10),
                     simplify = FALSE)
samples
```

```
## [[1]]
##  [1]  89.3  97.8  89.7  92.7  93.7  83.1 108.4
##  [8] 101.5  88.6 112.5
##
## [[2]]
##  [1] 104.3  97.0 109.0 108.8 108.2 106.9 105.5
##  [8]  99.4  96.9  96.2
##
## [[3]]
##  [1]  93.1  97.9  87.3 121.7 112.1  88.8  96.0
##  [8]  95.3 107.8  99.2
##
## [[4]]
##  [1] 102.5  99.7  99.6 113.7  97.7 115.2  84.5
##  [8] 105.8 101.2 102.2
##
## [[5]]
##  [1] 103.8  95.0  96.7  89.8  89.3 103.0 104.5
##  [8] 100.5 109.2 120.5
##
## [[6]]
```

Calculate all means

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

```
## [1]  95.8 103.2  99.9 102.2 101.2  96.4
```

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

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

```
## [1] 3.14
```

Sample size

Let's re–do this, pulling a sample of 100 each time.

```
samples2 <- replicate(1000, rnorm(100, mean = 100, sd = 10),
                      simplify = FALSE)
map_dbl(samples2, mean) %>%
  sd()
```

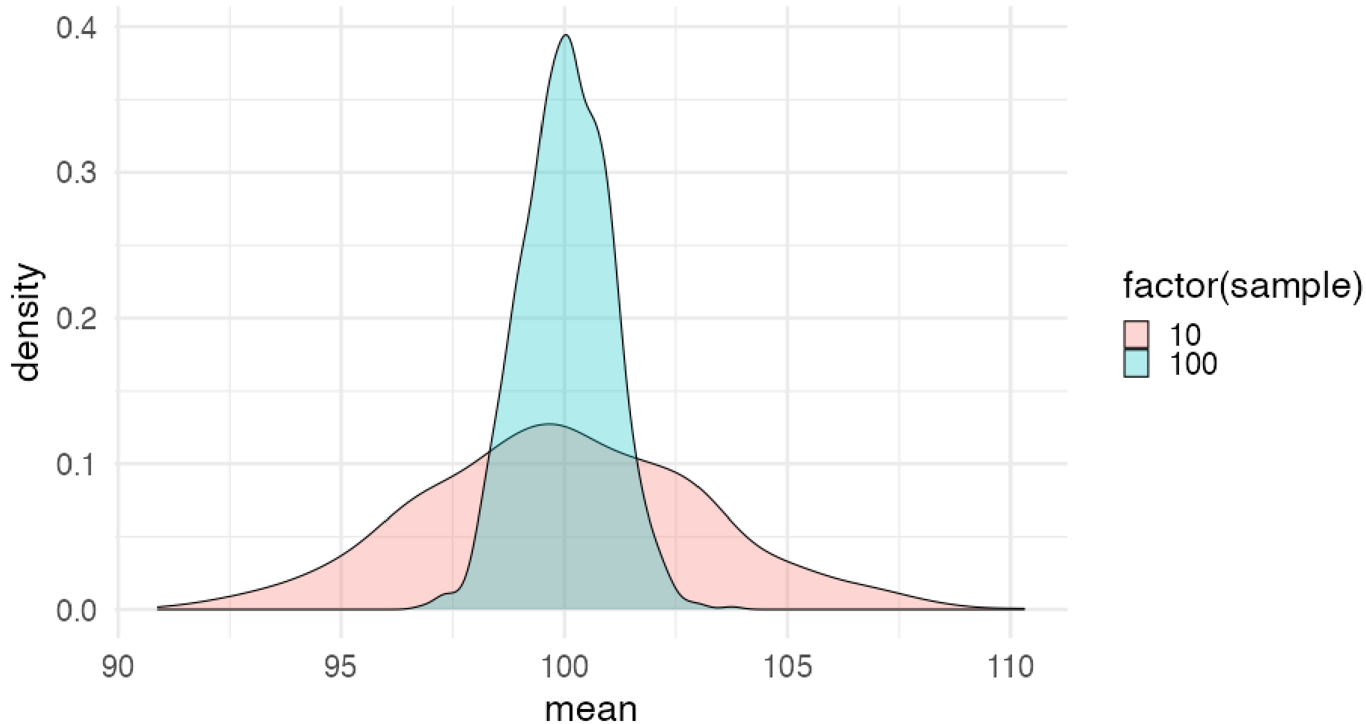
```
## [1] 0.973
```

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 × 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
## # … with 1,994 more rows
```

```
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)
```

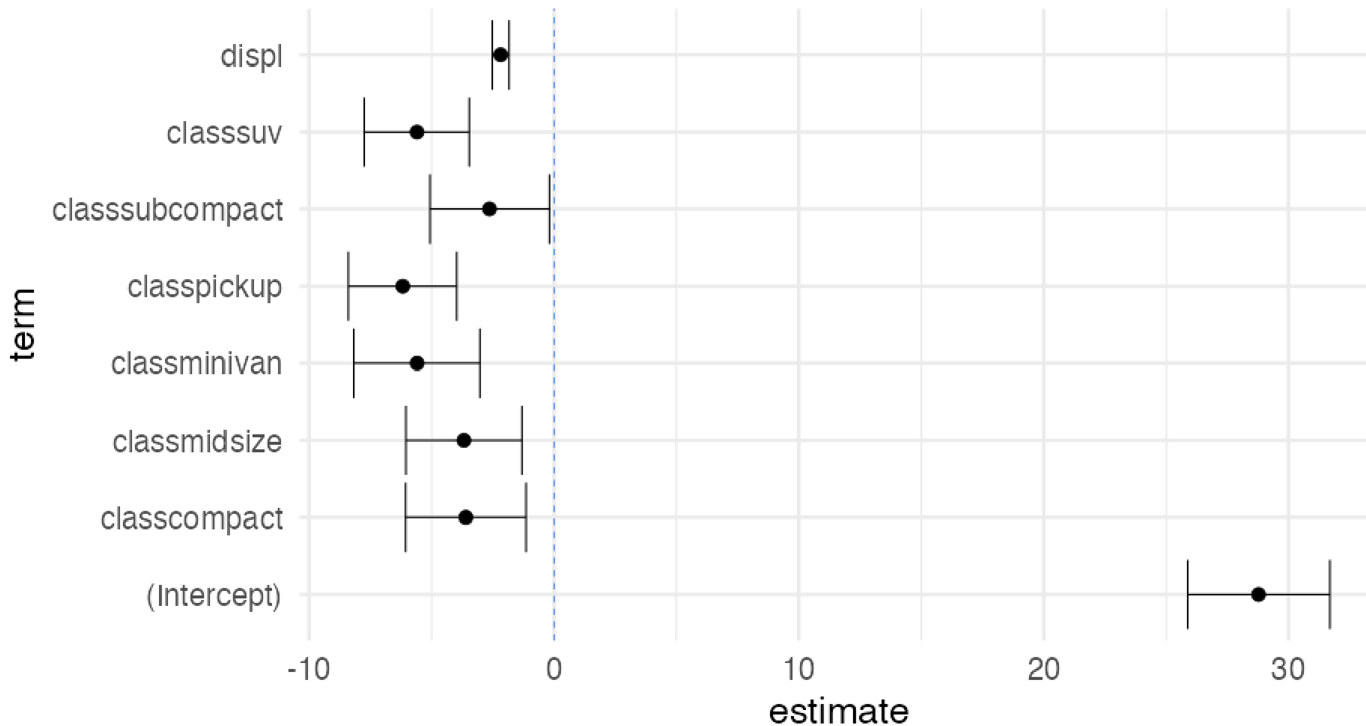
```
##
## Call:
## lm(formula = cty ~ displ + class, data = mpg)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.269 -1.150 -0.016  1.034 12.978
##
## Coefficients:
##                  Estimate Std. Error t value
## (Intercept)        28.777      1.473   19.54
## displ              -2.172      0.175  -12.43
## classcompact       -3.599      1.252   -2.87
## classmidsize       -3.676      1.206   -3.05
## classminivan       -5.595      1.306   -4.28
## classpickup        -6.182      1.121   -5.51
## classsubcompact    -2.629      1.237   -2.13
## classsuv           -5.599      1.087   -5.15
##                  Pr(>|t|)
## (Intercept)       < 2e-16 ***
## displ             < 2e-16 ***
## classcompact       0.0044 **
```

Visualize with standard errors

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

tidied_m
```

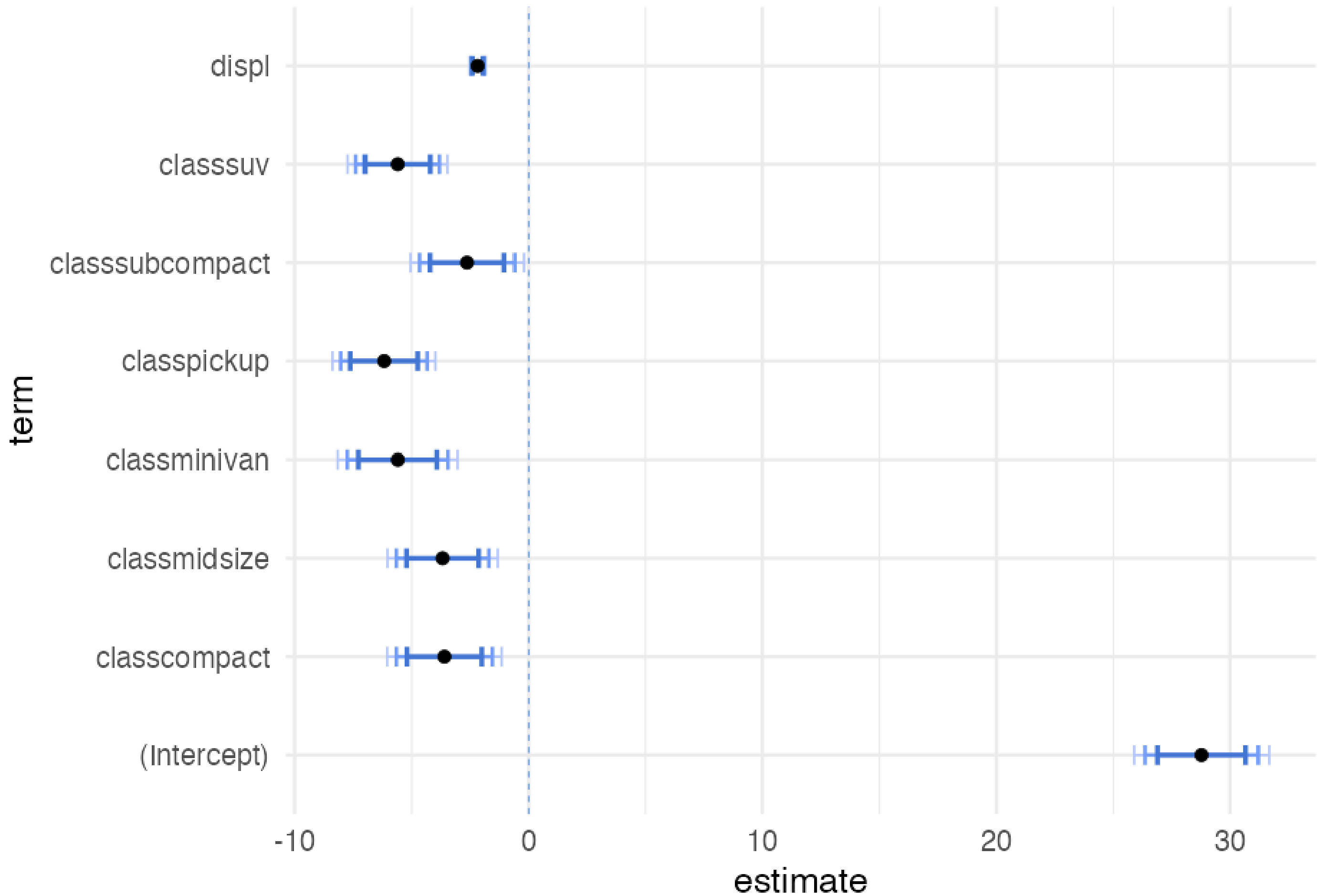
```
## # A tibble: 8 × 7
##   term          estimate std.error  statistic
##   <chr>            <dbl>     <dbl>      <dbl>
## 1 (Intercept)  28.77682  1.472892   19.53763
## 2 displ        -2.171562 0.1746638 -12.43281
## 3 classcompact -3.599125 1.252190   -2.874265
## 4 classmidsize -3.675526 1.206253   -3.047061
## 5 classminivan -5.595070 1.305993   -4.284151
## 6 classpickup  -6.182466 1.121448   -5.512931
## # … with 2 more rows, and 3 more variables:
## #   p.value <dbl>, conf.low <dbl>,
## #   conf.high <dbl>
```

```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()
```



Multiple error bars

```r
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()
```

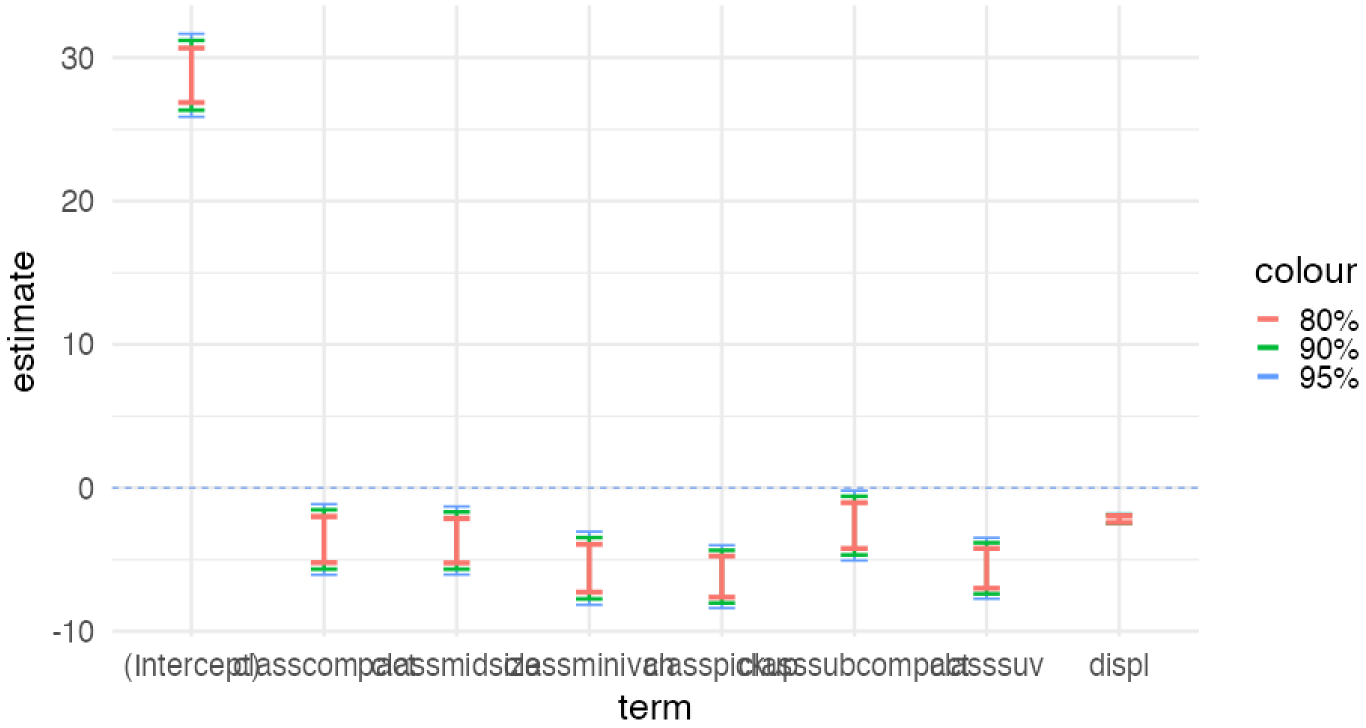


displ
classsuv
classsubcompact
classpickup
classminivan
classmidsize
classcompact
(Intercept)
term
-10
0
10
20
30
estimate

Add levels to legend

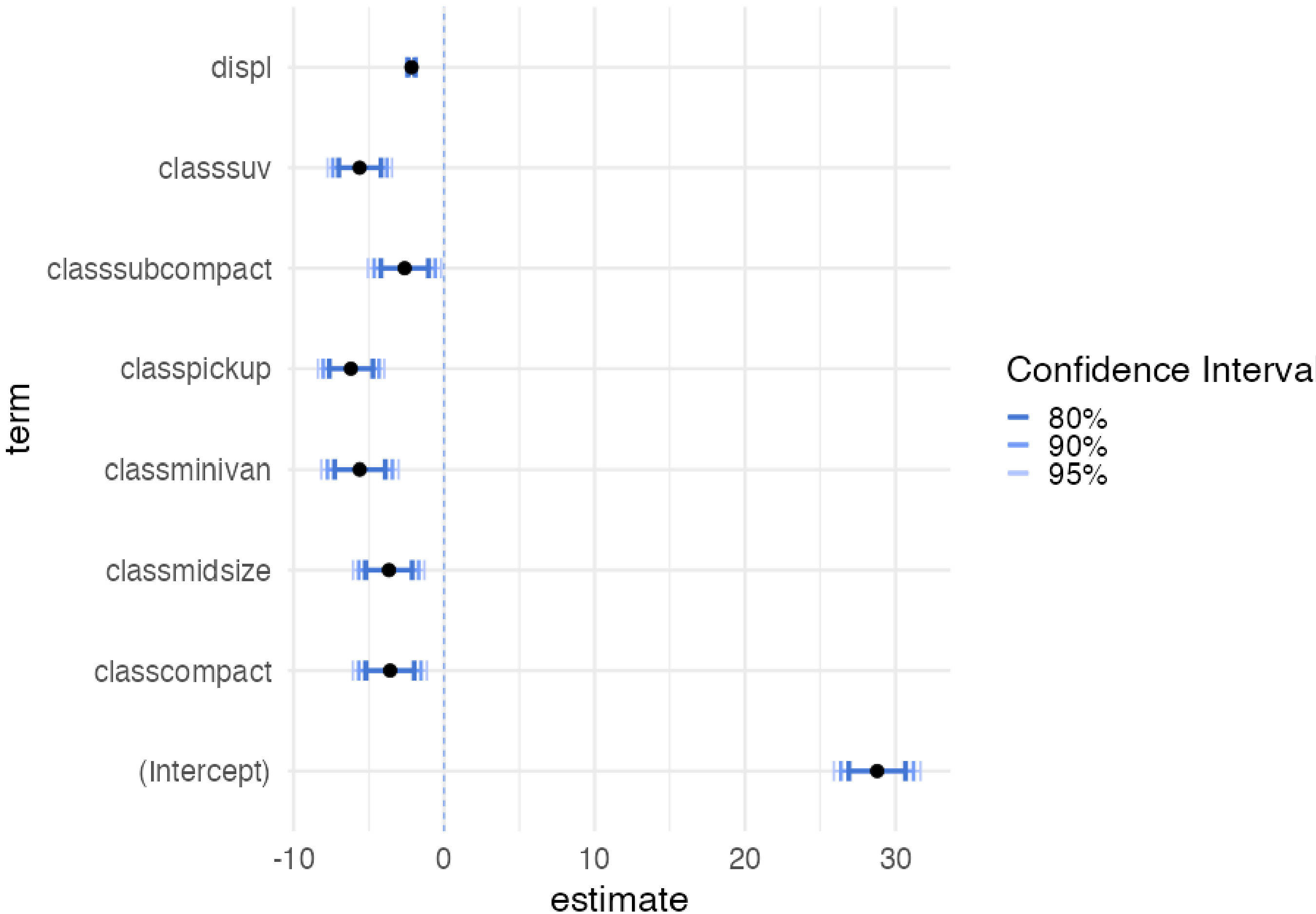
```
p <- ggplot(tidied_m, aes(term, estimate)) +
  geom_hline(yintercept = 0,
             color = "cornflowerblue",
             linetype = 2) +
  geom_errorbar(aes(ymin = estimate + qnorm(.025)*std.error,
                    ymax = estimate + qnorm(.975)*std.error,
                    color = "95%"),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

```
p
```



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

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

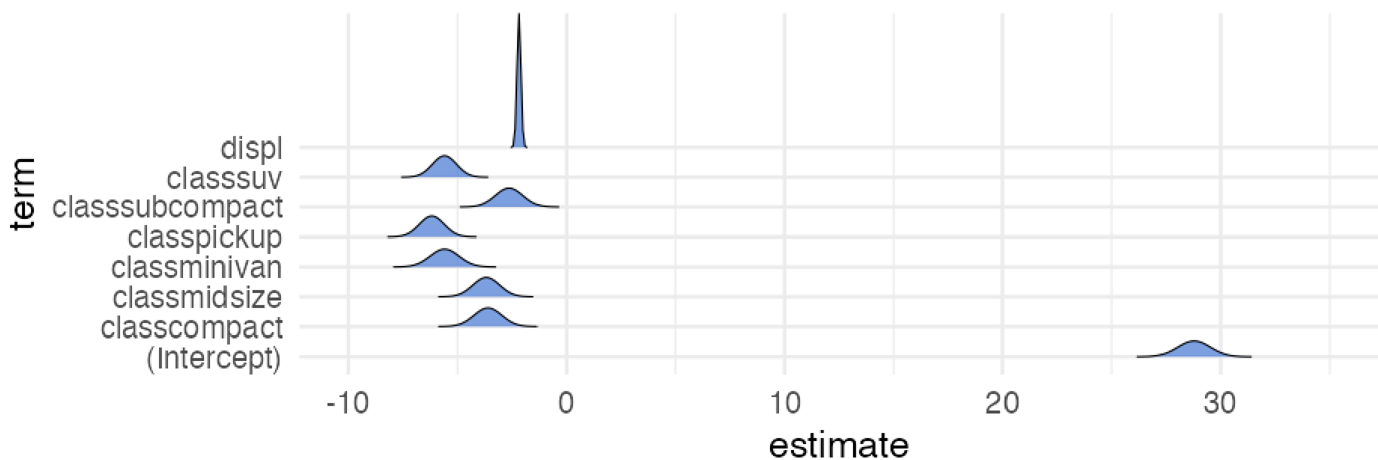

displ
classsuv
classsubcompact
classpickup
classminivan
classmidsize
classcompact
(Intercept)
term
-10
0
10
20
30
estimate
Confidence Interval
80%
90%
95%

Density stripes

```
#remotes::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)
```



Practice

- Go back to your species means from Palmer Penguins
- Reproduce the plot using one of the three methods we just saw
    - Multiple error bars
    - Density stripes
    - Densities

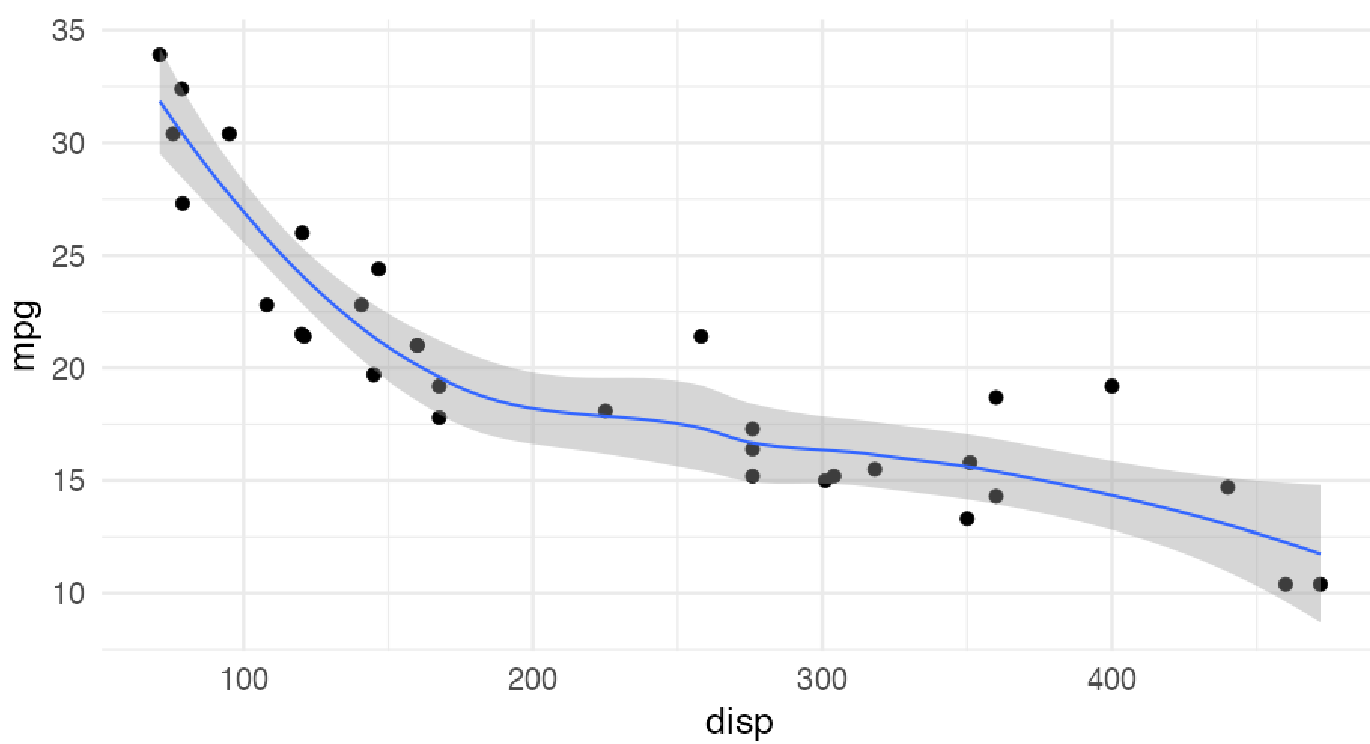
05:00

HOPs

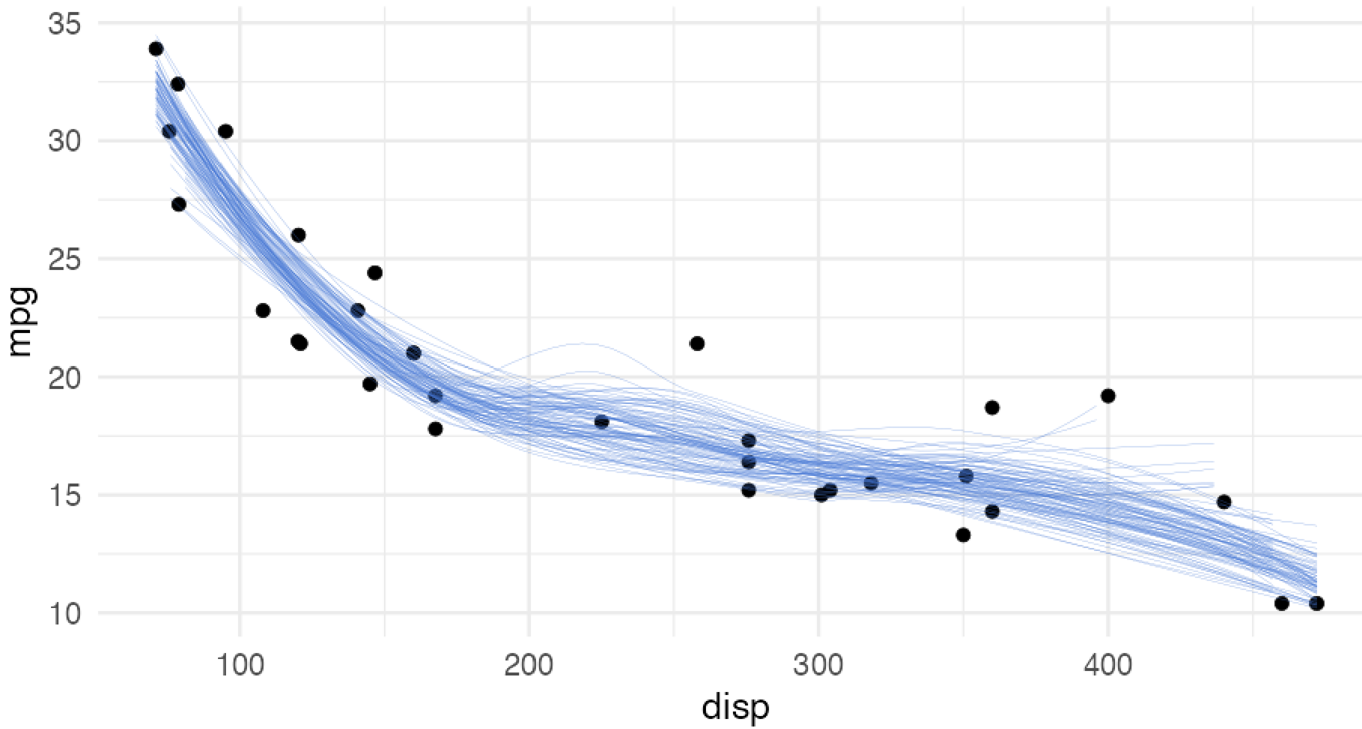
Hypothetical Outcome Plots (and related plots)

Standard regression plot

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



Alternative



How?

Bootstrapping

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

row_samps
```

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

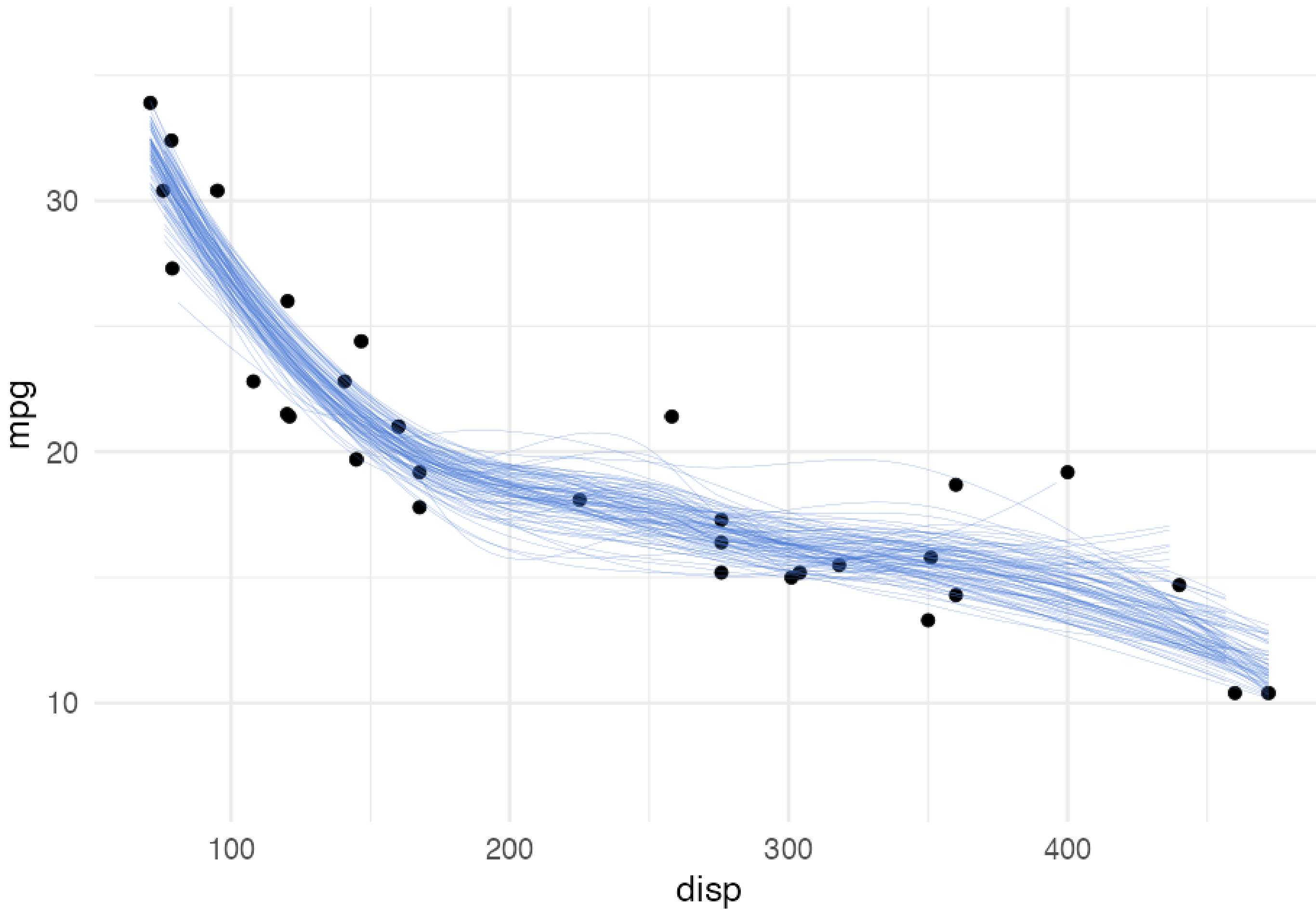
Extract samples

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

```
##                          sample  mpg cyl disp
## Dodge Challenger...1          1 15.5   8  318
## Ferrari Dino...2              1 19.7   6  145
## Pontiac Firebird...3          1 19.2   8  400
## Pontiac Firebird.1...4        1 19.2   8  400
## Hornet Sportabout...5         1 18.7   8  360
## Maserati Bora...6             1 15.0   8  301
##                           hp drat   wt qsec vs
## Dodge Challenger...1     150 2.76 3.52 16.9  0
## Ferrari Dino...2         175 3.62 2.77 15.5  0
## Pontiac Firebird...3     175 3.08 3.85 17.1  0
## Pontiac Firebird.1...4 175 3.08 3.85 17.1  0
## Hornet Sportabout...5    175 3.15 3.44 17.0  0
## Maserati Bora...6        335 3.54 3.57 14.6  0
##                          am gear carb
## Dodge Challenger...1      0    3    2
## Ferrari Dino...2          1    5    6
## Pontiac Firebird...3      0    3    2
## Pontiac Firebird.1...4    0    3    2
## Hornet Sportabout...5     0    3    2
## Maserati Bora...6         1    5    8
```

Plot both data sources

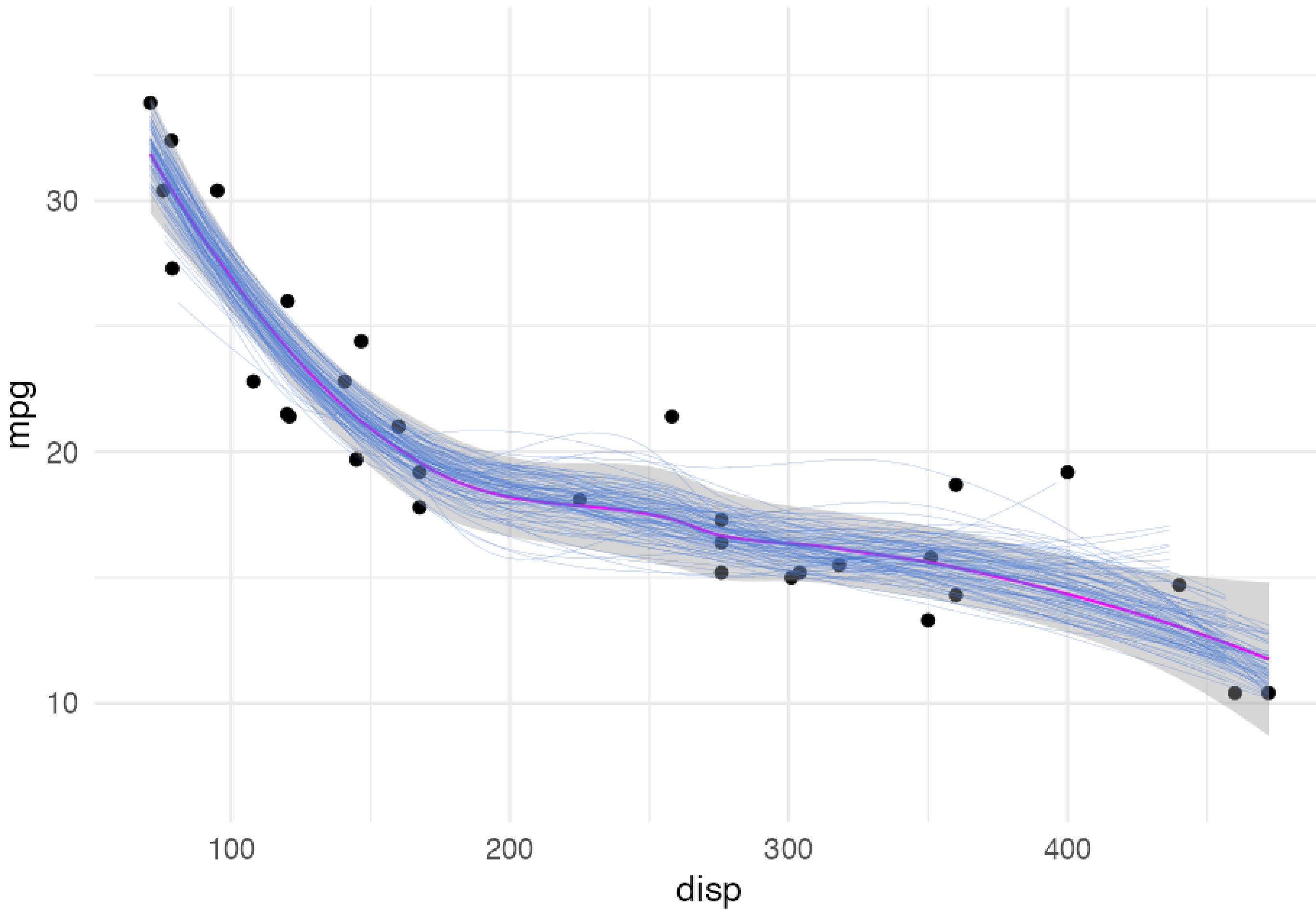
```
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  stat_smooth(
    aes(group = sample),
    data = d_samps,
    geom = "line",
    color = "#4375D3",
    fullrange = TRUE,
    size = 0.1
  )
```



mpg
30
20
10
100
200
300
400
disp

Note, they match up

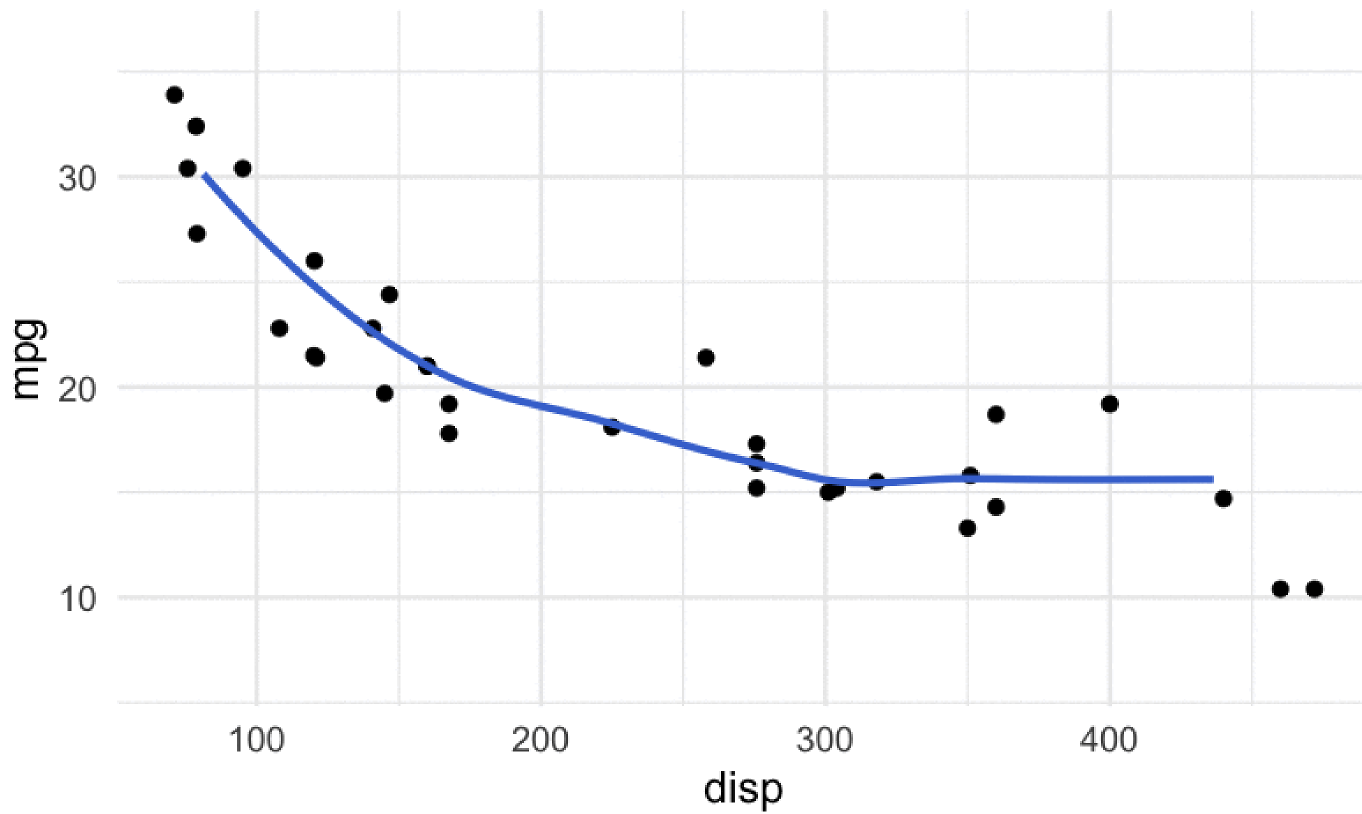
```
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  geom_smooth(color = "magenta") +
  stat_smooth(
    aes(group = sample),
    data = d_samps,
    geom = "line",
    color = "#4375D3",
    fullrange = TRUE,
    size = 0.1
  )
```



30
20
10
mpg
100
200
300
400
disp

HOPs

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

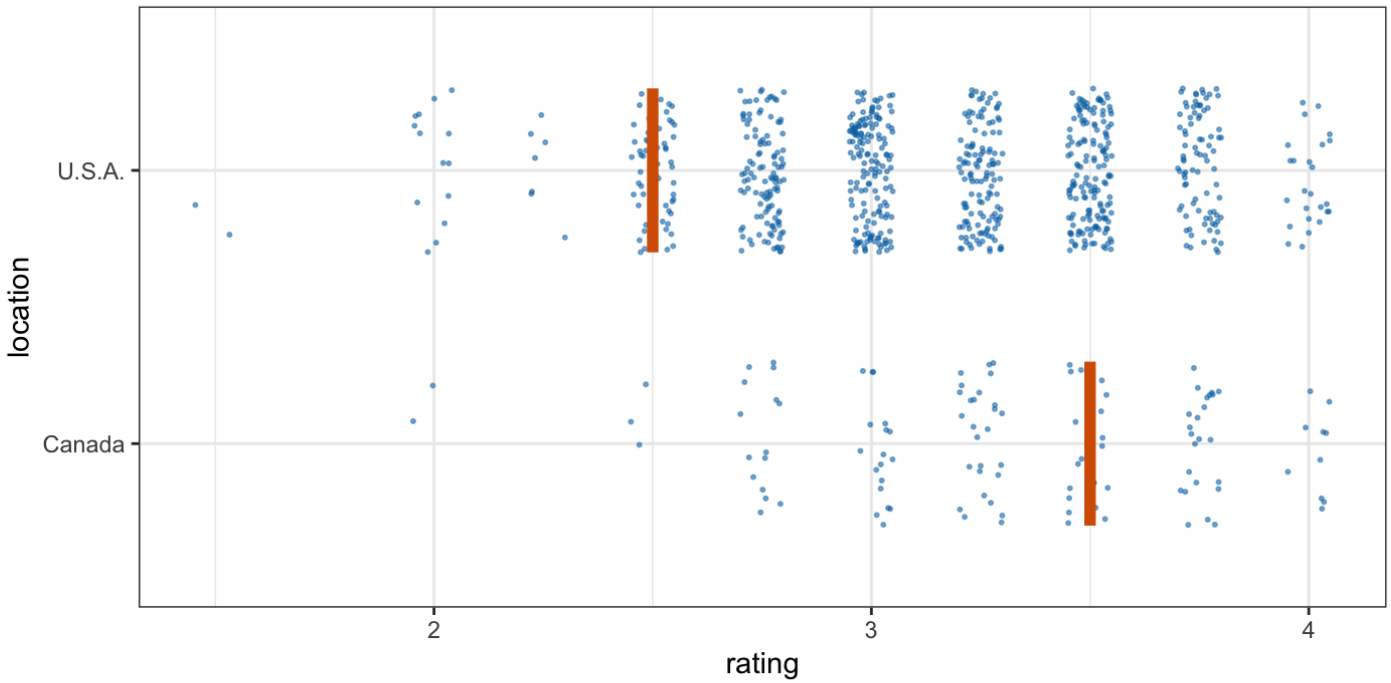


How?

gganimate::transition_states

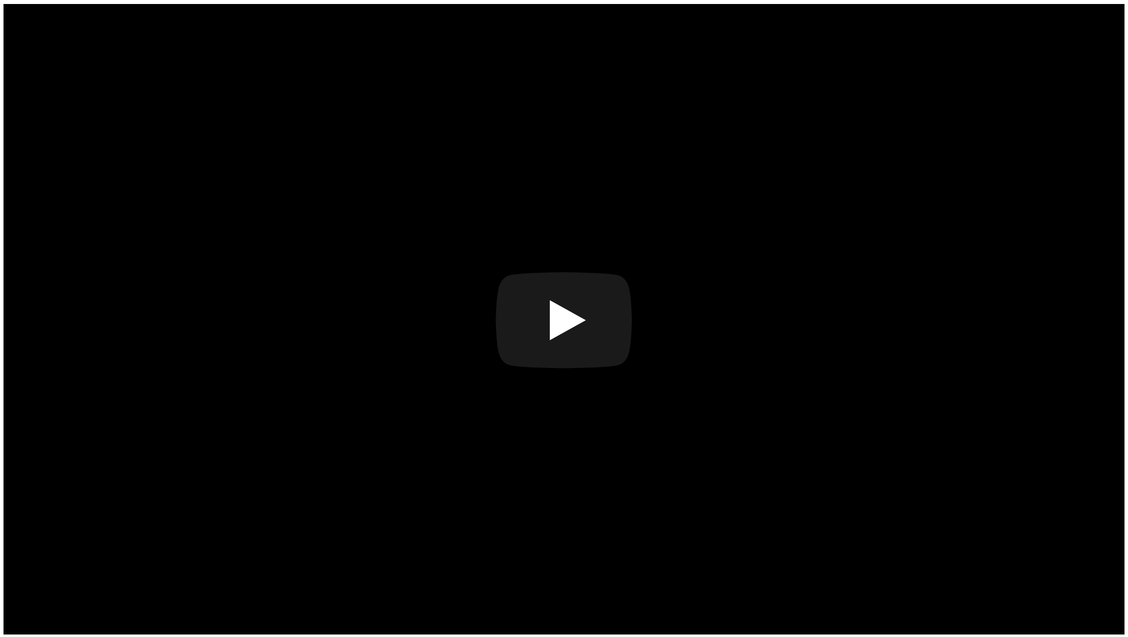
```r
library(gganimate)
ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  stat_smooth(
    data = d_samps,
    geom = "line",
    size = 2,
    color = "#4375D3",
    fullrange = TRUE
  ) +
  transition_states(
    sample,
    transition_length = 0.5,
    state_length = 0.5
  ) +
  ease_aes('linear') # 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)
- 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.

Next time: Dashboards, websites, etc.