

Visualizing uncertainty

Daniel Anderson

Week 7, Class 2

Reviewing

Lab 3

Data viz in the wild

Ann-Marie

Murat

Sarah Donaldson & Hyeonjin on deck

Agenda

- 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

Thinking about uncertainty

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

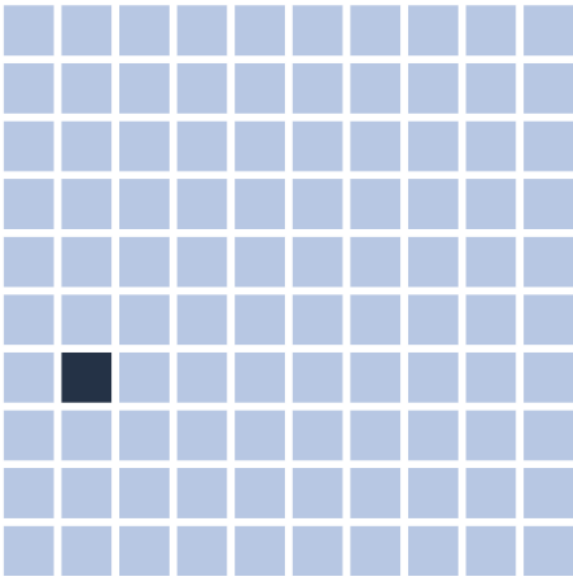
- We are nearly always uncertain of future events (forecasting)
- We can also be uncertain about past events
 - I saw a parked car at 8 AM, but the next time I looked at 2PM it was gone. What time did it leave?

Quantifying uncertainty

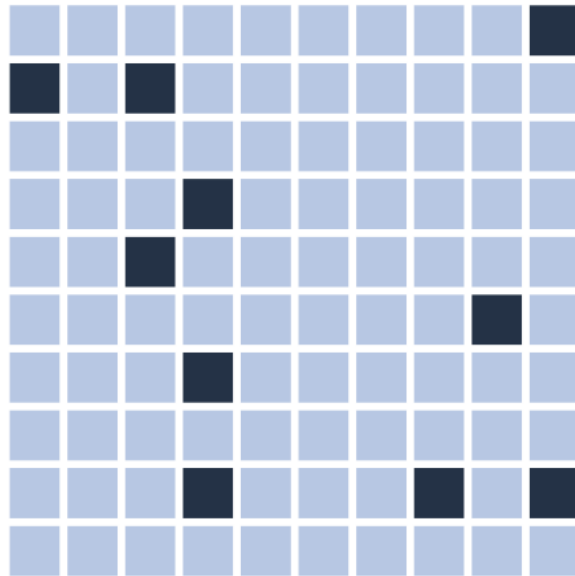
- We quantify our uncertainty mathematically using probability
- Framing probabilities as frequencies is generally more intuitive

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 occurrence 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
```

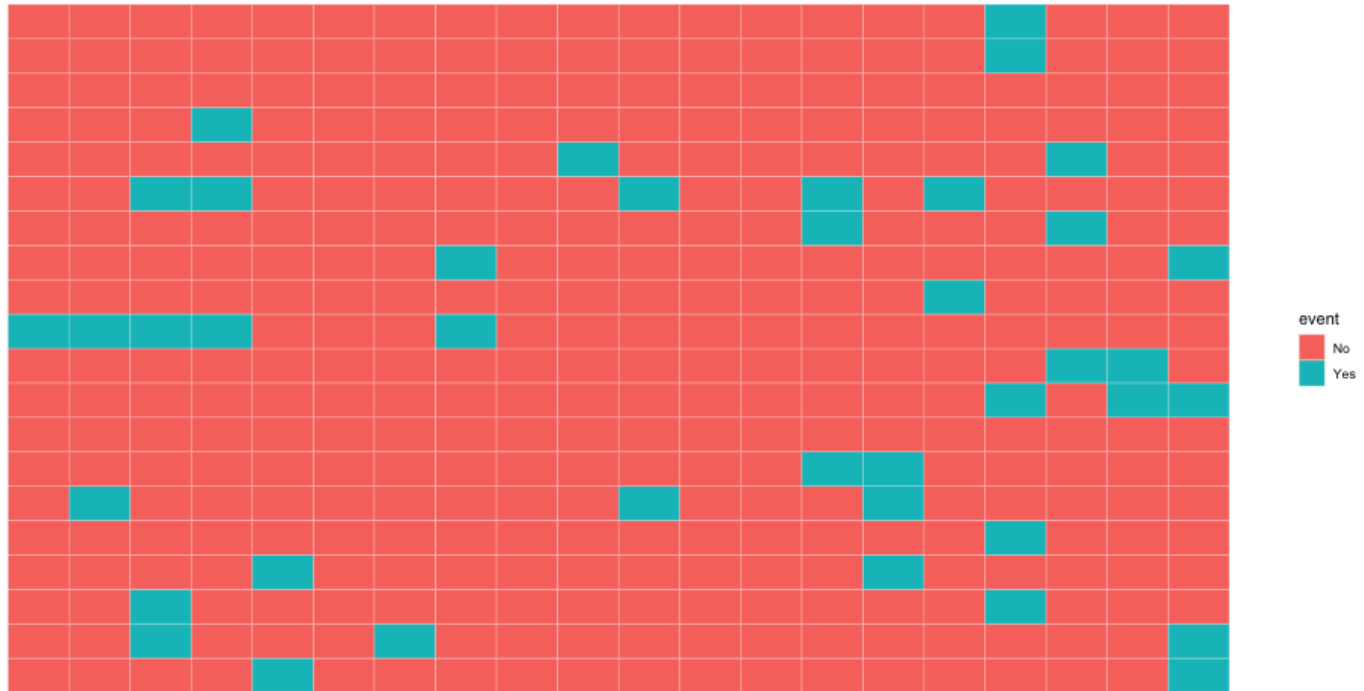
Create the variable

```
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
```

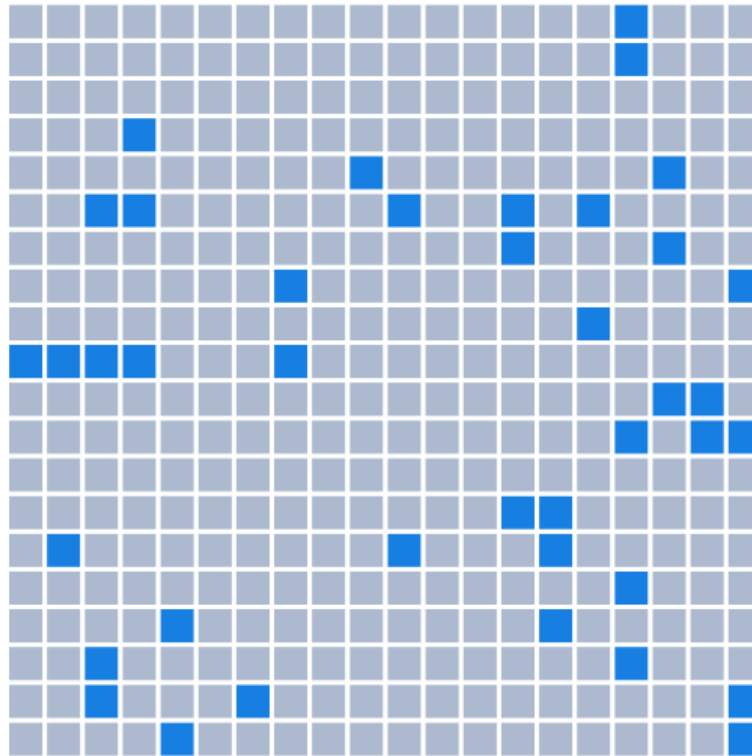
Fill in

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



Customize

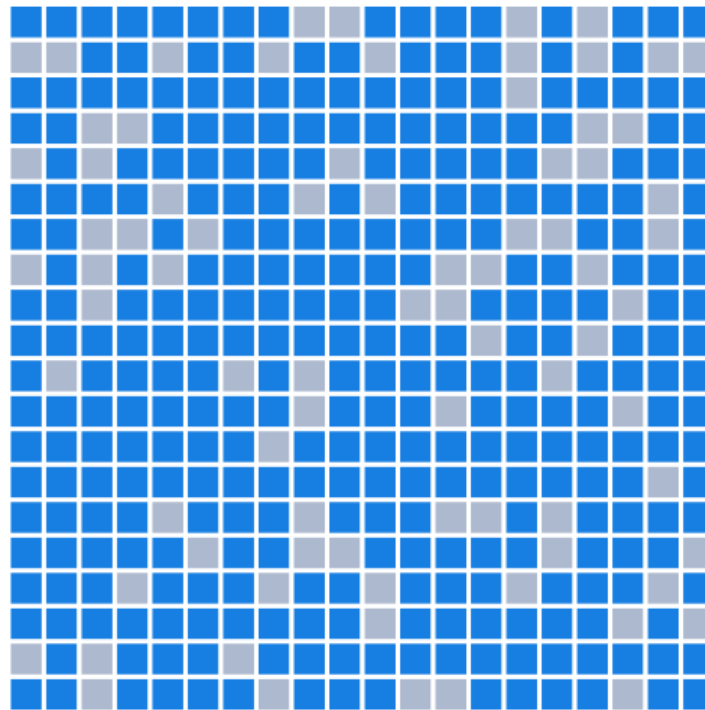
```
library(colorspace)
ggplot(grid, aes(x, y)) +
  geom_tile(aes(fill = event), color = "white", size = 1.4) +
  scale_fill_manual(
    name = "Event Occurred",
    values = c(
      desaturate(
        lighten("#1694E8", 0.5),
        0.7),
      "#1694E8"
    )
  ) +
  coord_fixed() +
  theme_void()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



Event Occurred No Yes

Chance of rain

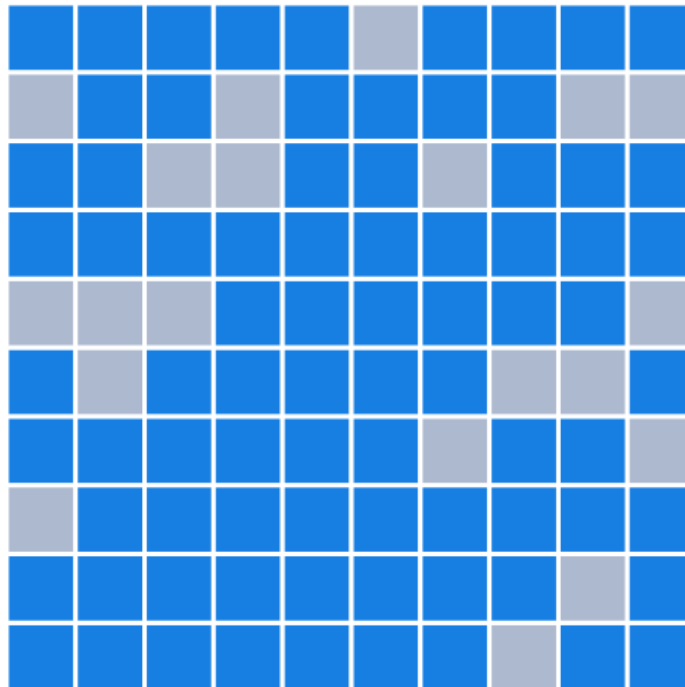
80%



Event Occurred ■ No ■ Yes

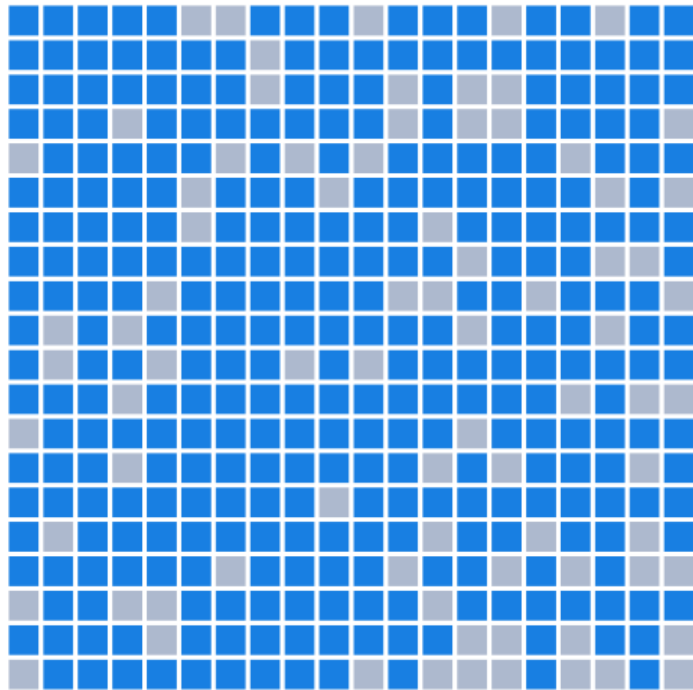
Vary grid size

10 x 10



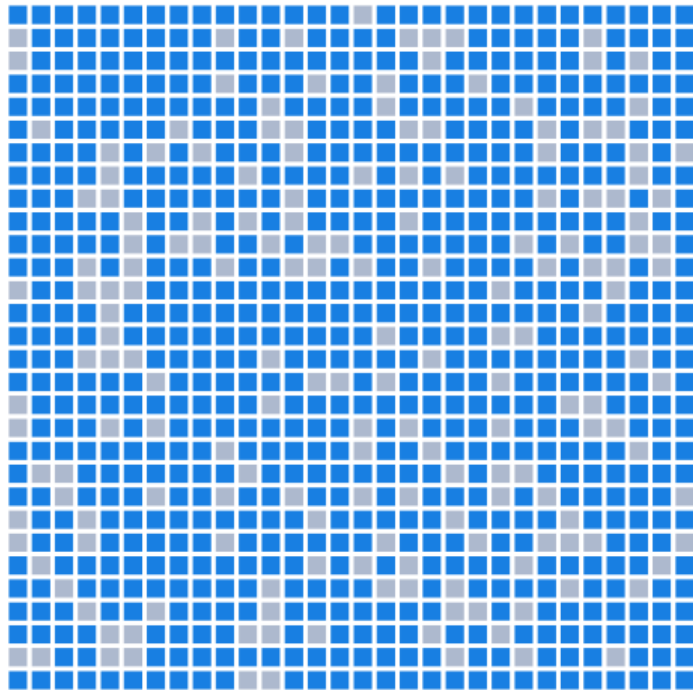
Vary grid size

20 x 20



Vary grid size

30 x 30



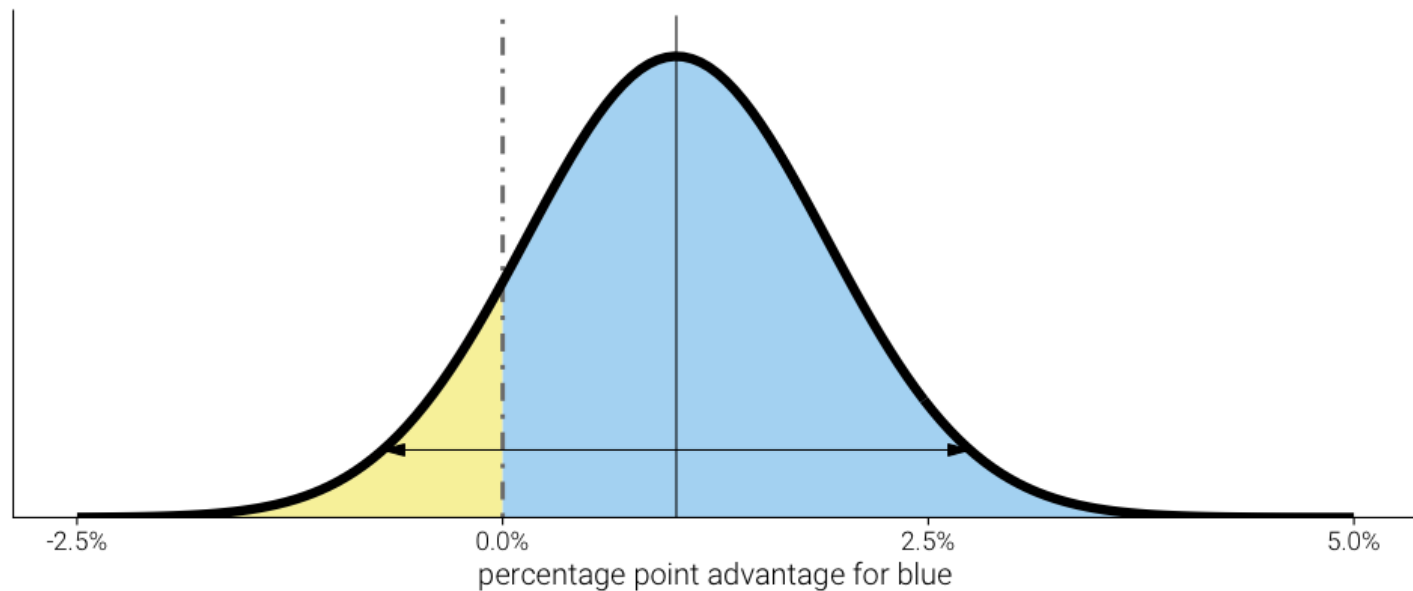
(probs too many)

Non-discrete
probabilities

Hypothetical

Blue party has 1% advantage (technically 1.02%) w/ margin of error of 1.76 points

Who will win?



A bit of math

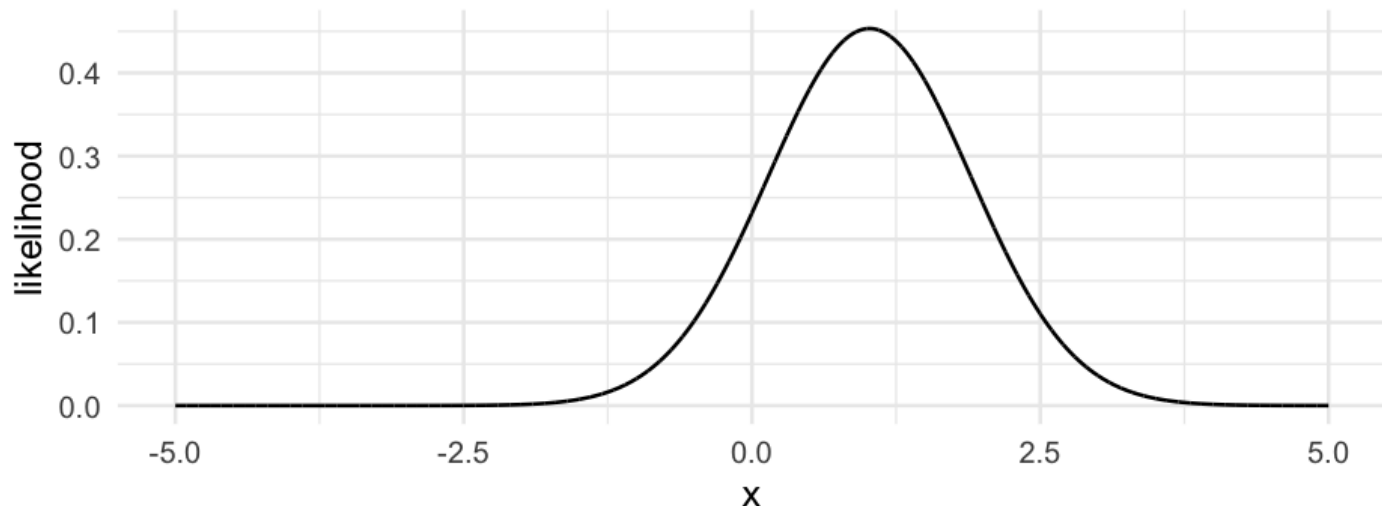
Our prior distribution was defined by $\mu = 1.02$ and $sd = 0.88$ (margin of error divided by two).

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

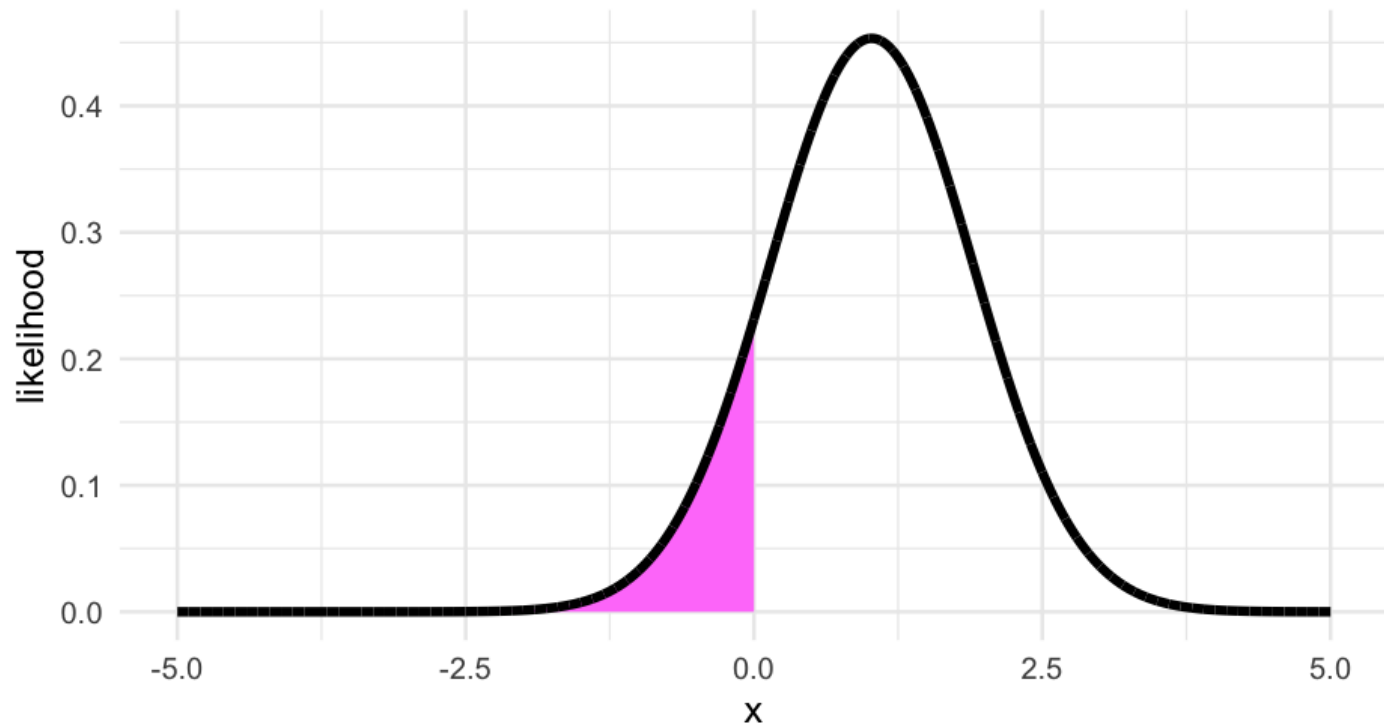
The hard way

Calculate the exact probability of data below zero under this distribution

```
x <- seq(-5, 5, 0.001)
likelihood <- dnorm(x, 1.02, 0.88)
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] 0.1232096
```

Easier: Simulate

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

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

Discretized plot

```
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 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 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 0.91 0.93 0.95 0.97 0.99
```

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

```
## [1] -1.073713087 -0.672714247 -0.460368264 -0.308211925 -0.186679530 -0.06247984 0.087209949 0.161251272 0.229893334 0.294220878 0.358548419 0.422876460 0.487204501 0.551532542 0.615860583 0.680188625 0.744516666 0.808844708 0.873172750 0.937500791 1.001828833 1.066156875 1.130484917 1.194812959 1.259141000 1.323469042 1.387797084 1.452125125 1.516453167 1.580781209 1.645109251 1.709437293 1.773765335 1.838093377 1.902421419 1.966749461 2.031077503 2.095405545 2.159733587 2.224061629 2.288389671 2.352717713 2.417045755 2.481373797 2.545701839 2.610029881 2.674357923 2.738685965 2.803014007 2.867342049 2.931670091 2.995998133 3.060326175 3.124654217 3.188982259 3.253310301 3.317638343 3.381966385 3.446294427 3.510622469 3.574950511 3.639278553 3.703606595 3.767934637 3.832262679 3.896590721 3.960918763 4.025246805 4.089574847 4.153902889 4.218230931 4.282558973 4.346887015 4.411215057 4.475543099 4.539871141 4.604199183 4.668527225 4.732855267 4.797183309 4.861511351 4.925839393 4.990167435 5.054495477 5.118823519 5.183151561 5.247479603 5.311807645 5.376135687 5.440463729 5.504791771 5.569119813 5.633447855 5.697775897 5.762103939 5.826431981 5.890760023 5.955088065 6.019416107 6.083744149 6.148072191 6.212400233 6.276728275 6.341056317 6.405384359 6.469712401 6.534040443 6.598368485 6.662696527 6.727024569 6.791352611 6.855680653 6.919908695 6.984236737 7.048564779 7.112892821 7.177220863 7.241548905 7.305876947 7.370204989 7.434533031 7.498861073 7.563189115 7.627517157 7.691845199 7.756173241 7.820501283 7.884829325 7.949157367 8.013485409 8.077813451 8.142141493 8.206469535 8.270797577 8.335125619 8.399453661 8.463781703 8.528109745 8.592437787 8.656765829 8.721093871 8.785421913 8.849750000 8.914078042 8.978406084 9.042734126 9.107062168 9.171390210 9.235718252 9.300046294 9.364374336 9.428702378 9.493030420 9.557358462 9.621686504 9.686014546 9.750342588 9.814670630 9.878998672 9.943326714 10.007654756 10.071982798 10.136310840 10.200638882 10.264966924 10.329294966 10.393623008 10.457951050 10.522279092 10.586607134 10.650935176 10.715263218 10.779591260 10.843919302 10.908247344 10.972575386 11.036903428 11.101231470 11.165559512 11.229887554 11.294215596 11.358543638 11.422871680 11.487199722 11.551527764 11.615855806 11.680183848 11.744511890 11.808839932 11.873167974 11.937496016 12.001824058 12.066152100 12.130480142 12.194808184 12.259136226 12.323464268 12.387792310 12.452120352 12.516448394 12.580776436 12.645104478 12.709432520 12.773760562 12.838088604 12.902416646 12.966744688 13.031072730 13.095400772 13.159728814 13.224056856 13.288384898 13.352712940 13.417040982 13.481369024 13.545697066 13.610025108 13.674353150 13.738681192 13.803009234 13.867337276 13.931665318 13.995993360 14.060321402 14.124649444 14.188977486 14.253305528 14.317633570 14.381961612 14.446289654 14.510617696 14.574945738 14.639273780 14.703601822 14.767929864 14.832257906 14.896585948 14.960913990 15.025242032 15.089570074 15.153898116 15.218226158 15.282554200 15.346882242 15.411210284 15.475538326 15.539866368 15.604194410 15.668522452 15.732850494 15.797178536 15.861506578 15.925834620 15.990162662 16.054490704 16.118818746 16.183146788 16.247474830 16.311802872 16.376130914 16.440458956 16.504787000 16.569115042 16.633443084 16.697771126 16.762099168 16.826427210 16.890755252 16.955083294 17.019411336 17.083739378 17.148067420 17.212395462 17.276723504 17.341051546 17.405379588 17.469707630 17.534035672 17.598363714 17.662691756 17.727019798 17.791347840 17.855675882 17.919903924 17.984231966 18.048560008 18.112888050 18.177216092 18.241544134 18.305872176 18.370200218 18.434528260 18.498856302 18.563184344 18.627512386 18.691840428 18.756168470 18.820496512 18.884824554 18.949152596 19.013480638 19.077808680 19.142136722 19.206464764 19.270792806 19.335120848 19.399448890 19.463776932 19.528104974 19.592433016 19.656761058 19.721089100 19.785417142 19.849745184 19.914073226 19.978401268 20.042729310 20.107057352 20.171385394 20.235713436 20.300041478 20.364369520 20.428697562 20.493025604 20.557353646 20.621681688 20.686009730 20.750337772 20.814665814 20.878993856 20.943321898 21.007649940 21.071977982 21.136306024 21.200634066 21.264962108 21.329290150 21.393618192 21.457946234 21.522274276 21.586602318 21.650930360 21.715258402 21.779586444 21.843914486 21.908242528 21.972570570 22.036898612 22.101226654 22.165554696 22.229882738 22.294210780 22.358538822 22.422866864 22.487194906 22.551522948 22.615850990 22.680179032 22.744507074 22.808835116 22.873163158 22.937491200 23.001819242 23.066147284 23.130475326 23.194803368 23.259131410 23.323459452 23.387787494 23.452115536 23.516443578 23.580771620 23.645099662 23.709427704 23.773755746 23.838083788 23.902411830 23.966739872 24.031067914 24.095395956 24.159724000 24.224052042 24.288380084 24.352708126 24.417036168 24.481364210 24.545692252 24.610020294 24.674348336 24.738676378 24.803004420 24.867332462 24.931660504 24.995988546 25.060316588 25.124644630 25.188972672 25.253300714 25.317628756 25.381956798 25.446284840 25.510612882 25.574940924 25.639268966 25.703597008 25.767925050 25.832253092 25.896581134 25.960909176 26.025237218 26.089565260 26.153893302 26.218221344 26.282549386 26.346877428 26.411205470 26.475533512 26.539861554 26.604189596 26.668517638 26.732845680 26.797173722 26.861501764 26.925829806 26.990157848 27.054485890 27.118813932 27.183141974 27.247470016 27.311798058 27.376126100 27.440454142 27.504782184 27.569110226 27.633438268 27.697766310 27.762094352 27.826422394 27.890750436 27.955078478 28.019406520 28.083734562 28.148062604 28.212390646 28.276718688 28.341046730 28.405374772 28.469702814 28.534030856 28.598358898 28.662686940 28.727014982 28.791343024 28.855671066 28.919999108 28.984327150 29.048655192 29.112983234 29.177311276 29.241639318 29.305967360 29.370295402 29.434623444 29.498951486 29.563279528 29.627607570 29.691935612 29.756263654 29.820591696 29.884919738 29.949247780 30.013575822 30.077903864 30.142231906 30.206559948 30.270887990 30.335216032 30.399544074 30.463872116 30.528200158 30.592528200 30.656856242 30.721184284 30.785512326 30.849840368 30.914168410 30.978496452 31.042824494 31.107152536 31.171480578 31.235808620 31.300136662 31.364464704 31.428792746 31.493120788 31.557448830 31.621776872 31.686104914 31.750432956 31.814761000 31.879089042 31.943417084 32.007745126 32.072073168 32.136401210 32.200729252 32.265057294 32.329385336 32.393713378 32.458041420 32.522369462 32.586697504 32.651025546 32.715353588 32.779681630 32.844009672 32.908337714 32.972665756 33.036993798 33.101321840 33.165649882 33.230000000 33.294350000 33.358700000 33.423050000 33.487400000 33.551750000 33.616100000 33.680450000 33.744800000 33.809150000 33.873500000 33.937850000 34.002200000 34.066550000 34.130900000 34.195250000 34.259600000 34.323950000 34.388300000 34.452650000 34.517000000 34.581350000 34.645700000 34.710050000 34.774400000 34.838750000 34.903100000 34.967450000 35.031800000 35.096150000 35.160500000 35.224850000 35.289200000 35.353550000 35.417900000 35.482250000 35.546600000 35.610950000 35.675300000 35.739650000 35.804000000 35.868350000 35.932700000 35.997050000 36.061400000 36.125750000 36.190100000 36.254450000 36.318800000 36.383150000 36.447500000 36.511850000 36.576200000 36.640550000 36.704900000 36.769250000 36.833600000 36.897950000 36.962300000 37.026650000 37.091000000 37.155350000 37.219700000 37.284050000 37.348400000 37.412750000 37.477100000 37.541450000 37.605800000 37.670150000 37.734500000 37.798850000 37.863200000 37.927550000 37.991900000 38.056250000 38.120600000 38.184950000 38.249300000 38.313650000 38.378000000 38.442350000 38.506700000 38.571050000 38.635400000 38.699750000 38.764100000 38.828450000 38.892800000 38.957150000 39.021500000 39.085850000 39.150200000 39.214550000 39.278900000 39.343250000 39.407600000 39.471950000 39.536300000 39.600650000 39.665000000 39.729350000 39.793700000 39.858050000 39.922400000 39.986750000 40.051100000 40.115450000 40.179800000 40.244150000 40.308500000 40.372850000 40.437200000 40.501550000 40.565900000 40.630250000 40.694600000 40.758950000 40.823300000 40.887650000 40.952000000 41.016350000 41.080700000 41.145050000 41.209400000 41.273750000 41.338100000 41.402450000 41.466800000 41.531150000 41.595500000 41.659850000 41.724200000 41.788550000 41.852900000 41.917250000 41.981600000 42.045950000 42.110300000 42.174650000 42.239000000 42.303350000 42.367700000 42.432050000 42.496400000 42.560750000 42.625100000 42.689450000 42.753800000 42.818150000 42.882500000 42.946850000 43.011200000 43.075550000 43.139900000 43.204250000 43.268600000 43.332950000 43.397300000 43.461650000 43.526000000 43.590350000 43.654700000 43.719050000 43.783400000 43.847750000 43.912100000 43.976450000 44.040800000 44.105150000 44.169500000 44.233850000 44.298200000 44.362550000 44.426900000 44.491250000 44.555600000 44.620000000 44.684350000 44.748700000 44.813050000 44.877400000 44.941750000 45.006100000 45.070450000 45.134800000 45.199150000 45.263500000 45.327850000 45.392200000 45.456550000 45.520900000 45.585250000 45.649600000 45.713950000 45.778300000 45.842650000 45.907000000 45.971350000 46.035700000 46.100050000 46.164400000 46.228750000 46.293100000 46.357450000 46.421800000 46.486150000 46.550500000 46.614850000 46.679200000 46.743550000 46.807900000 46.872250000 46.936600000 47.000950000 47.065300000 47.129650000 47.194000000 47.258350000 47.322700000 47.387050000 47.451400000 47.515750000 47.580100000 47.644450000 47.708800000 47.773150000 47.837500000 47.901850000 47.966200000 48.030550000 48.094900000 48.159250000 48.223600000 48.287950000 48.352300000 48.416650000 48.481000000 48.545350000 48.609700000 48.674050000 48.738400000 48.802750000 48.867100000 48.931450000 48.995800000 49.060150000 49.124500000 49.188850000 49.253200000 49.317550000 49.381900000 49.446250000 49.510600000 49.574950000 49.639300000 49.703650000 49.768000000 49.832350000 49.896700000 49.961050000 50.025400000 50.089750000 50.154100000 50.218450000 50.282800000 50.347150000 50.411500000 50.475850000 50.540200000 50.604550000 50.668900000 50.733250000 50.797600000 50.861950000 50.926300000 50.990650000 51.055000000 51.119350000 51.183700000 51.248050000 51.312400000 51.376750000 51.441100000 51.505450000 51.569800000 51.634150000 51.698500000 51.762850000 51.827200000 51.891550000 51.955900000 52.020250000 52.084600000 52.148950000 52.213300000 52.277650000 52.342000000 52.406350000 52.470700000 52.535050000 52.599400000 52.663750000 52.728100000 52.792450000 52.856800000 52.921150000 52.985500000 53.049850000 53.114200000 53.178550000 53.242900000 53.307250000 53.371600000 53.435950000 53.500300000 53.564650000 53.629000000 53.693350000 53.757700000 53.822050000 53.886400000 53.950750000 54.015100000 54.079450000 54.143800000 54.208150000 54.272500000 54.336850000 54.401200000 54.465550000 54.529900000 54.594250000 54.658600000 54.722950000 54.787300000 54.851650000 54.916000000 54.980350000 55.044700000 55.109050000 55.173400000 55.237750000 55.302100000 55.366450000 55.430800000 55.495150000 55.559500000 55.623850000 55.688200000 55.752550000 55.816900000 55.881250000 55.945600000 56.010000000 56.074350000 56.138700000 56.203050000 56.267400000 56.331750000 56.396100000 56.460450000 56.524800000 56.589150000 56.653500000 56.717850000 56.782200000 56.846550000 56.910900000 56.975250000 57.039600000 57.103950000 57.168300000 57.232650000 57.297000000 57.361350000 57.425700000 57.490050000 57.554400000 57.618750000 57.683100000 57.747450000 57.811800000 57.876150000 57.940500000 58.004850000 58.069200000 58.133550000 58.197900000 58.262250000 58.326600000 58.390950000 58.455300000 58.519650000 58.584000000 58.648350000 58.712700000 58.777050000 58.841400000 58.905750000 58.970100000 59.034450000 59.098800000 59.163150000 59.227500000 59.291850000 59.356200000 59.420550000 59.484900000 59
```

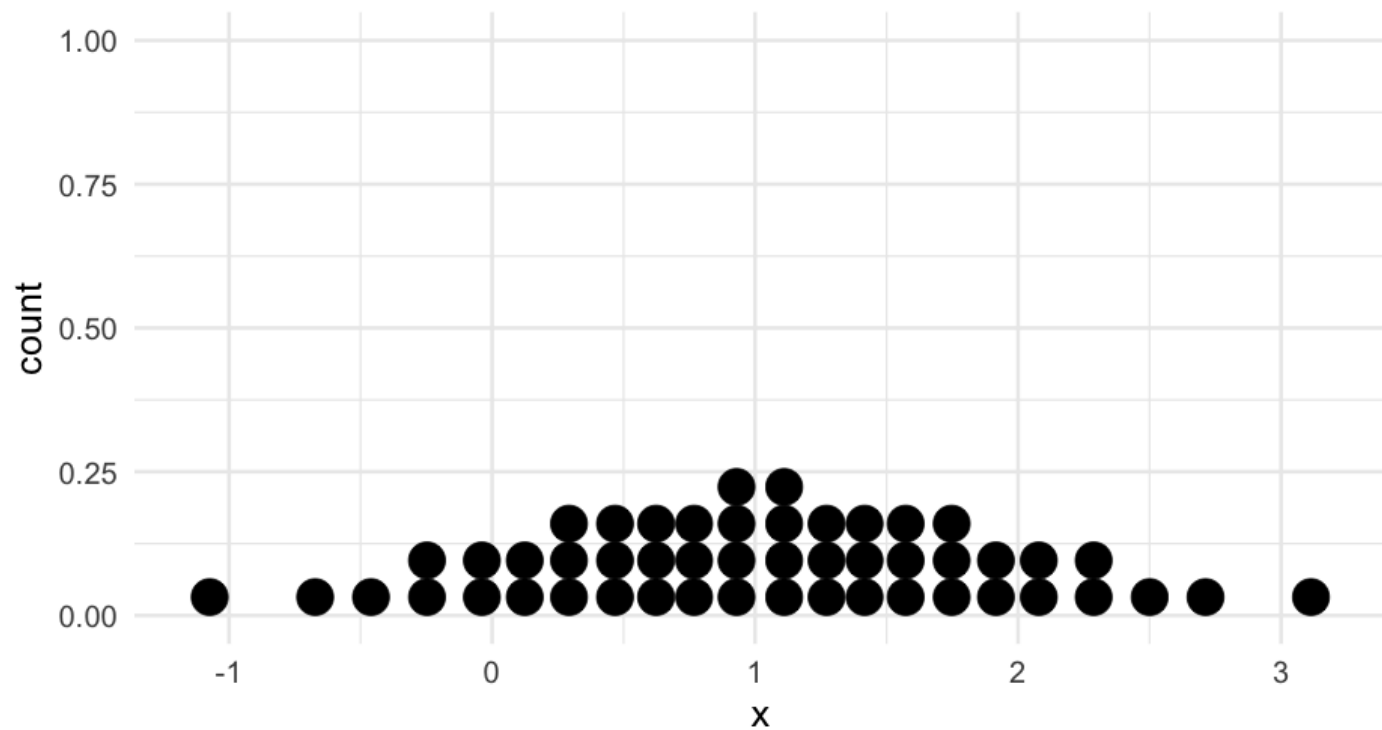
```
discretized <- data.frame(  
  x = qnorm(ppoints(50), 1.02, 0.9)  
  ) %>%  
  mutate(winner = ifelse(x <= 0, "#b1daf4", "#f8f1a9"))  
  
head(discretized)
```

```
##           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
```

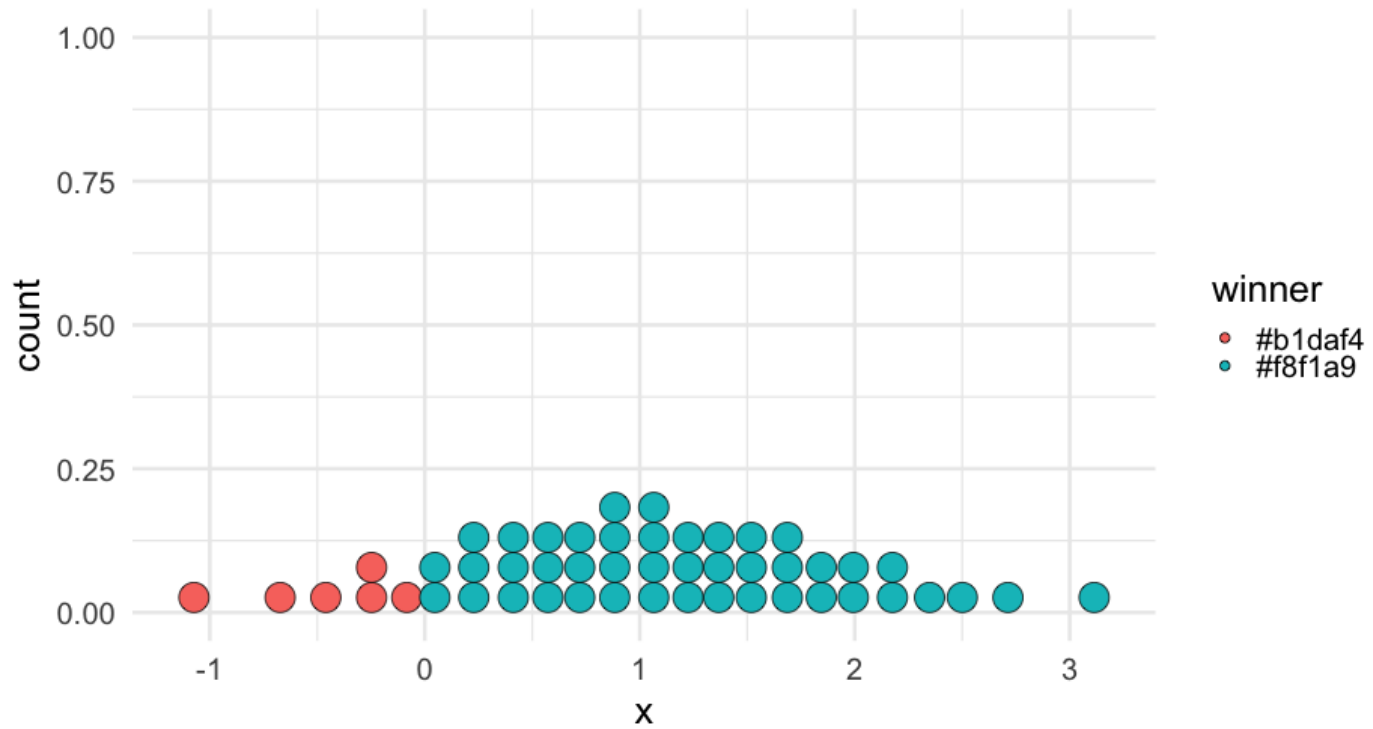
```
tail(discretized)
```

```
##           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
```

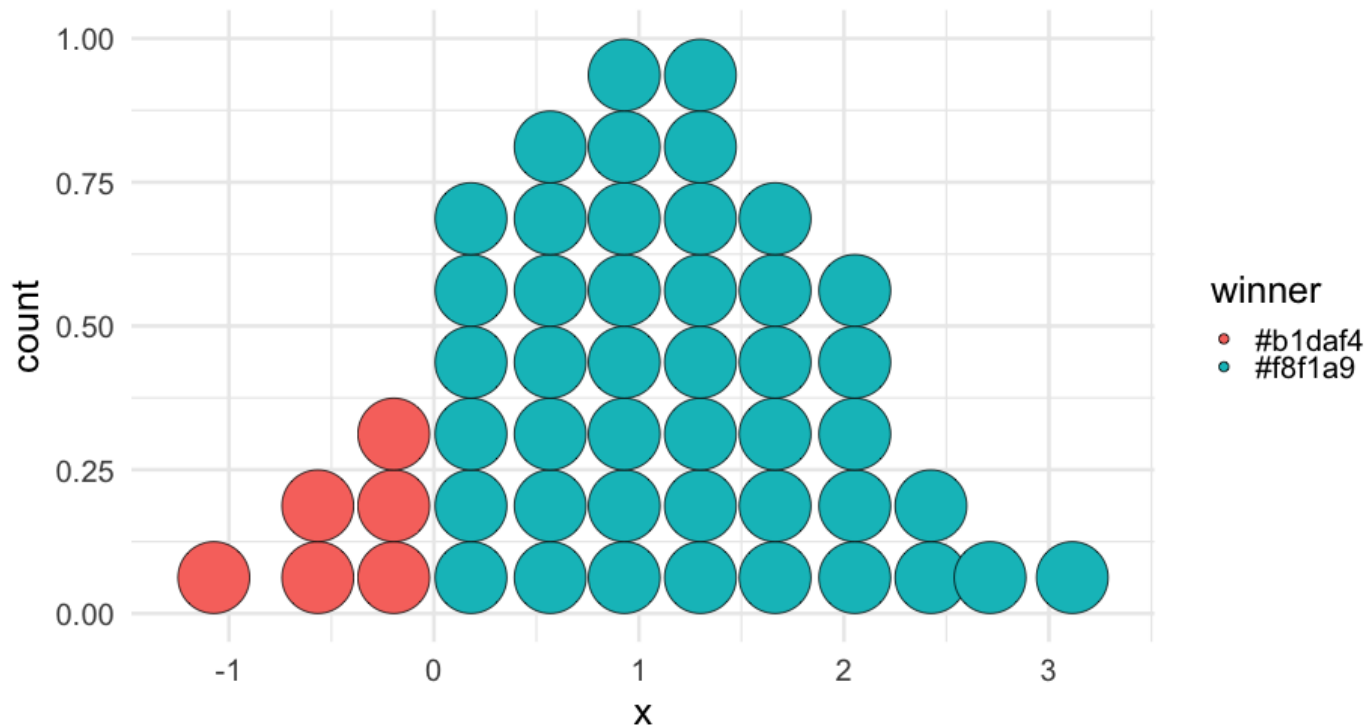
```
ggplot(discretized, aes(x)) +  
  geom_dotplot()
```



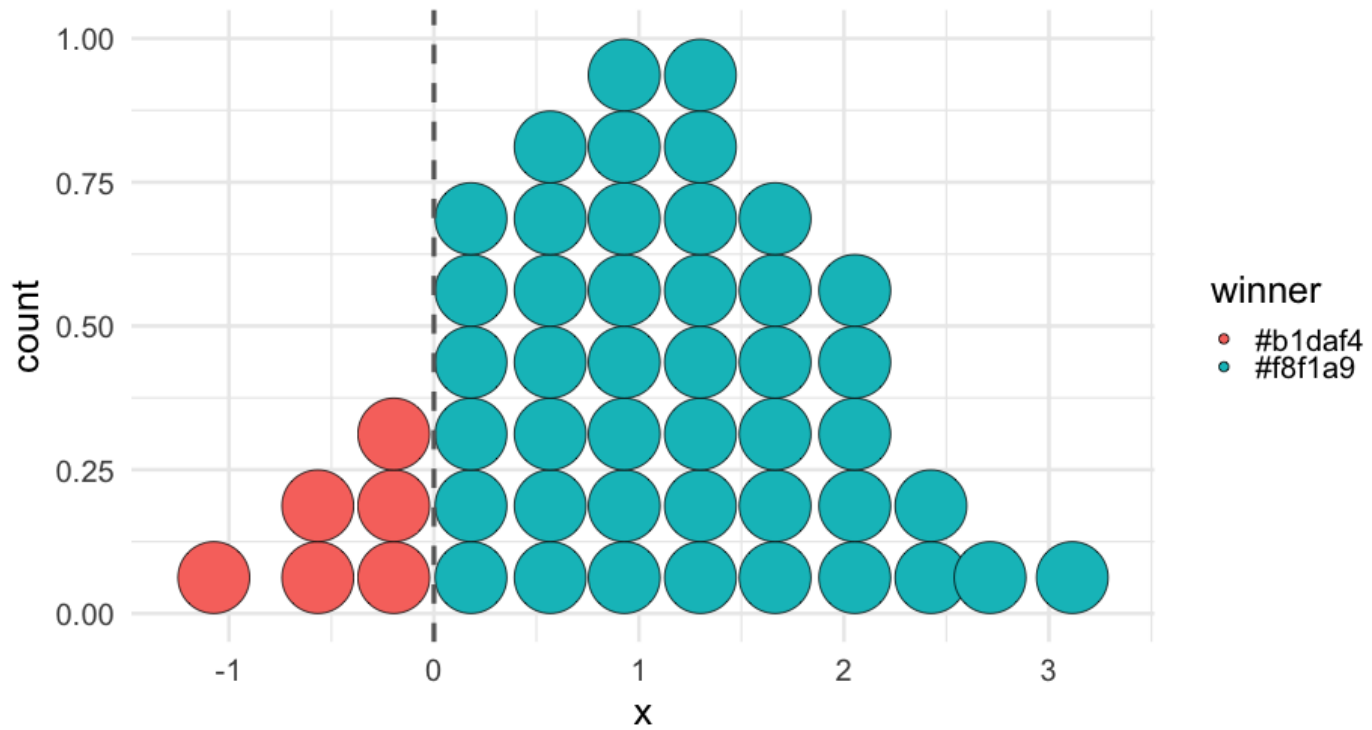
```
ggplot(discretized, aes(x)) +  
  geom_dotplot(aes(fill = winner))
```



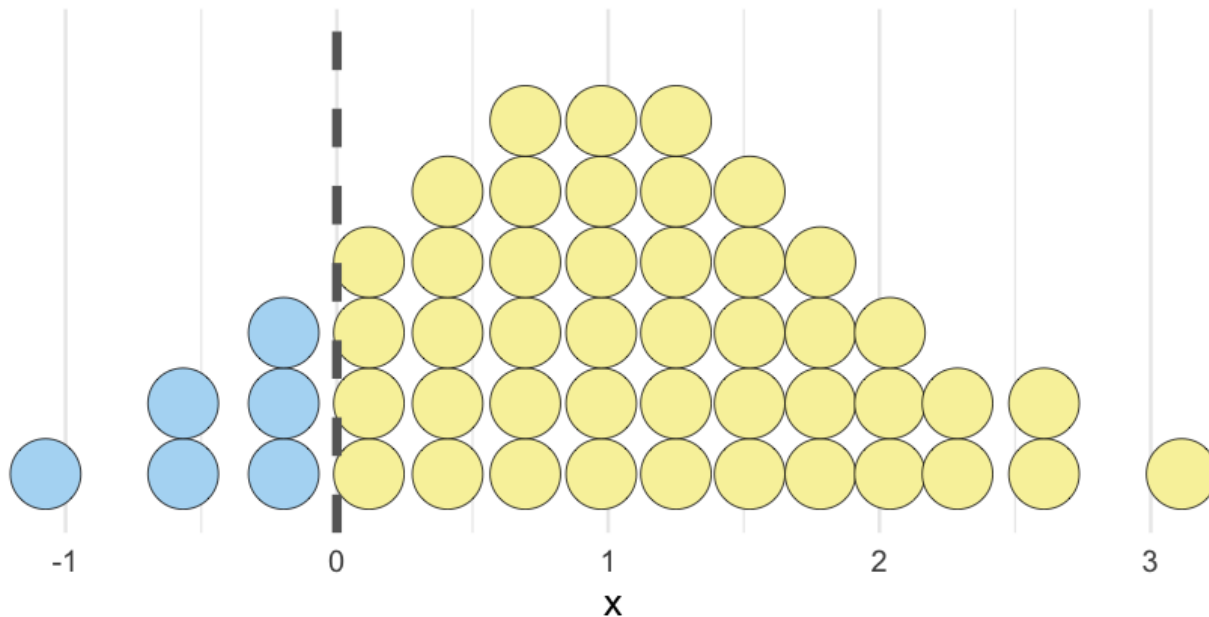
```
ggplot(discretized, aes(x)) +  
  geom_dotplot(aes(fill = winner), binwidth = 0.35)
```




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

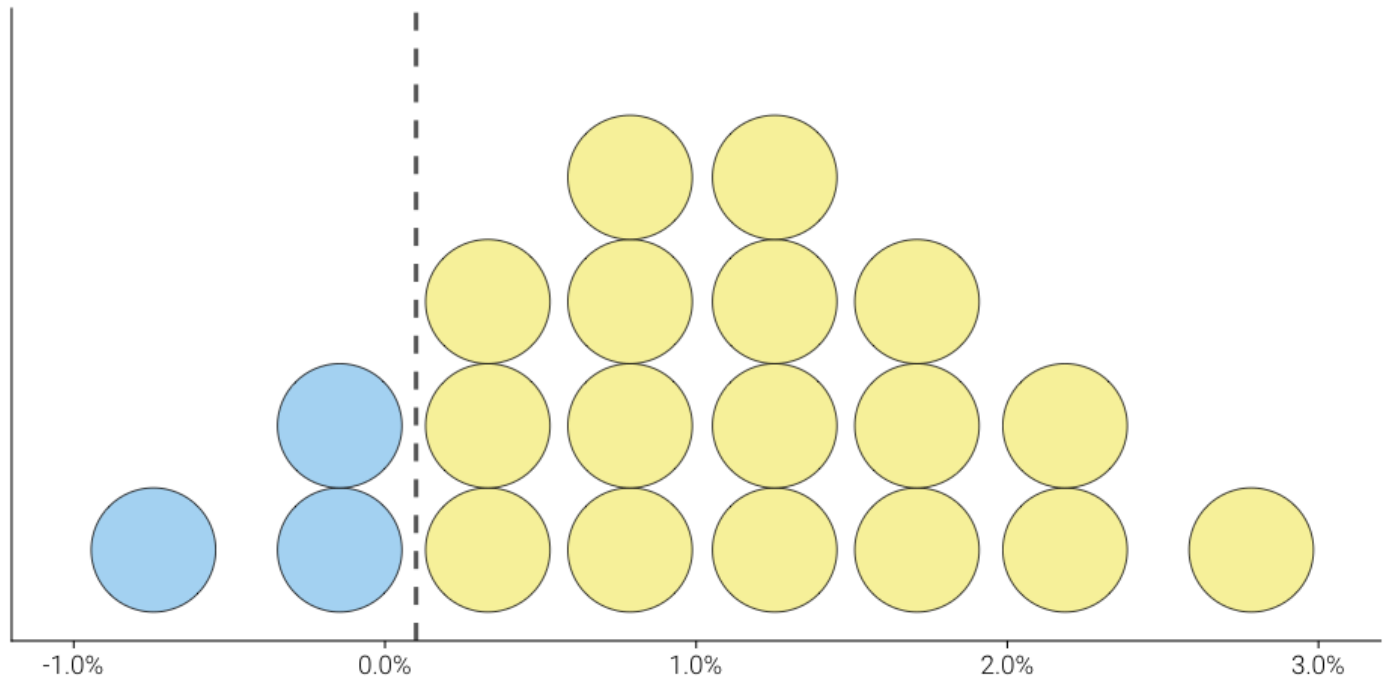


```
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.26) +
  geom_vline(xintercept = 0,
             color = "gray40",
             linetype = 2,
             size = 3) +
  scale_fill_identity(guide = "none") +
  scale_y_continuous(name = "",
                     breaks =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



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 104.609
## [9] 93.13147 95.54338
```

- Calculate the mean

```
mean(samp10a)
```

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

Do it a second time

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

```
## [1] 112.24082 103.59814 104.00771 101.10683 94.44159 117.86913 104.978
## [9] 107.01356 95.27209
```

```
mean(samp10b)
```

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


Do it a bunch of times

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

```
samples
```

```
## [[1]]  
## [1] 89.32176 97.82025 89.73996 92.71109 93.74961 83.13307 108.377  
## [9] 88.61863 112.53815  
##  
## [[2]]  
## [1] 104.26464 97.04929 108.95126 108.78133 108.21581 106.88640 105.539  
## [9] 96.94037 96.19529  
##  
## [[3]]  
## [1] 93.05293 97.92083 87.34604 121.68956 112.07962 88.76891 95.971  
## [9] 107.79965 99.16631  
##  
## [[4]]  
## [1] 102.53319 99.71453 99.57130 113.68602 97.74229 115.16471 84.512  
## [9] 101.23854 102.15942  
##  
## [[5]]  
## [1] 103.79639 94.97677 96.66793 89.81425 89.28209 103.03529 104.482  
## [9] 109.22267 120.50085  
##  
## [[6]]  
## [1] 95.08969 76.90831 110.05739 92.90799 93.11991 110.25571 497.152
```

Calculate all means

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

```
## [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.

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

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

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()
```

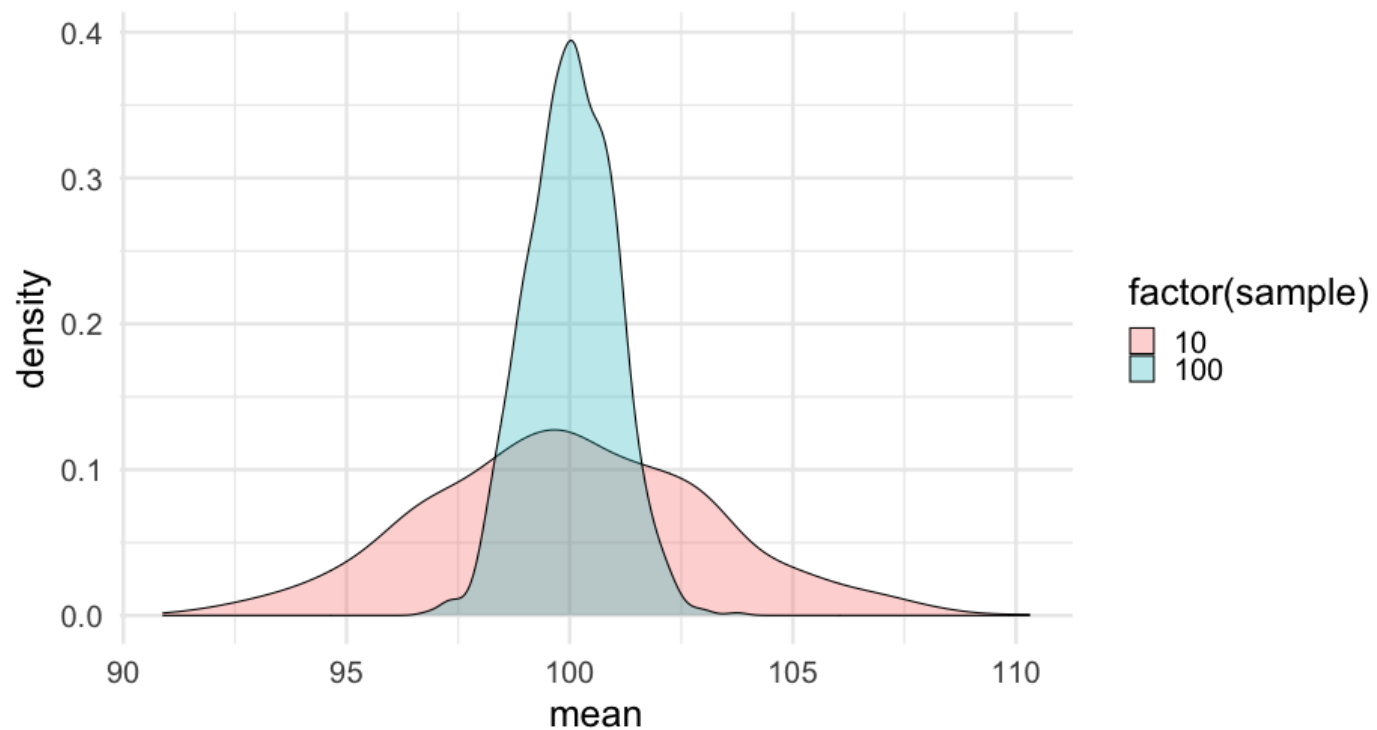
```
## [1] 0.9728883
```

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
```

```
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 *
## classsuvs     -5.5994     1.0872  -5.150 5.65e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.249 on 226 degrees of freedom
## Multiple R-squared:  0.7291,    Adjusted R-squared:  0.7207
```

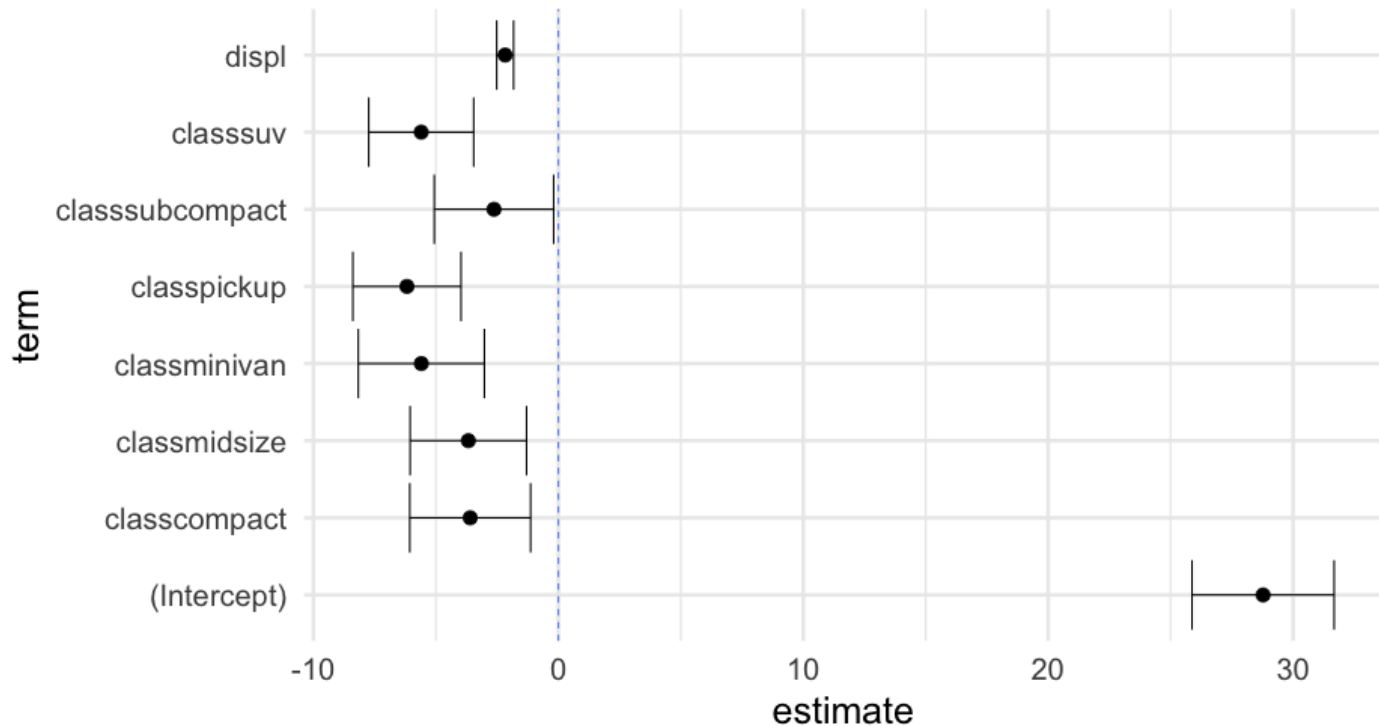
Visualize with standard errors

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

```
tidied_m
```

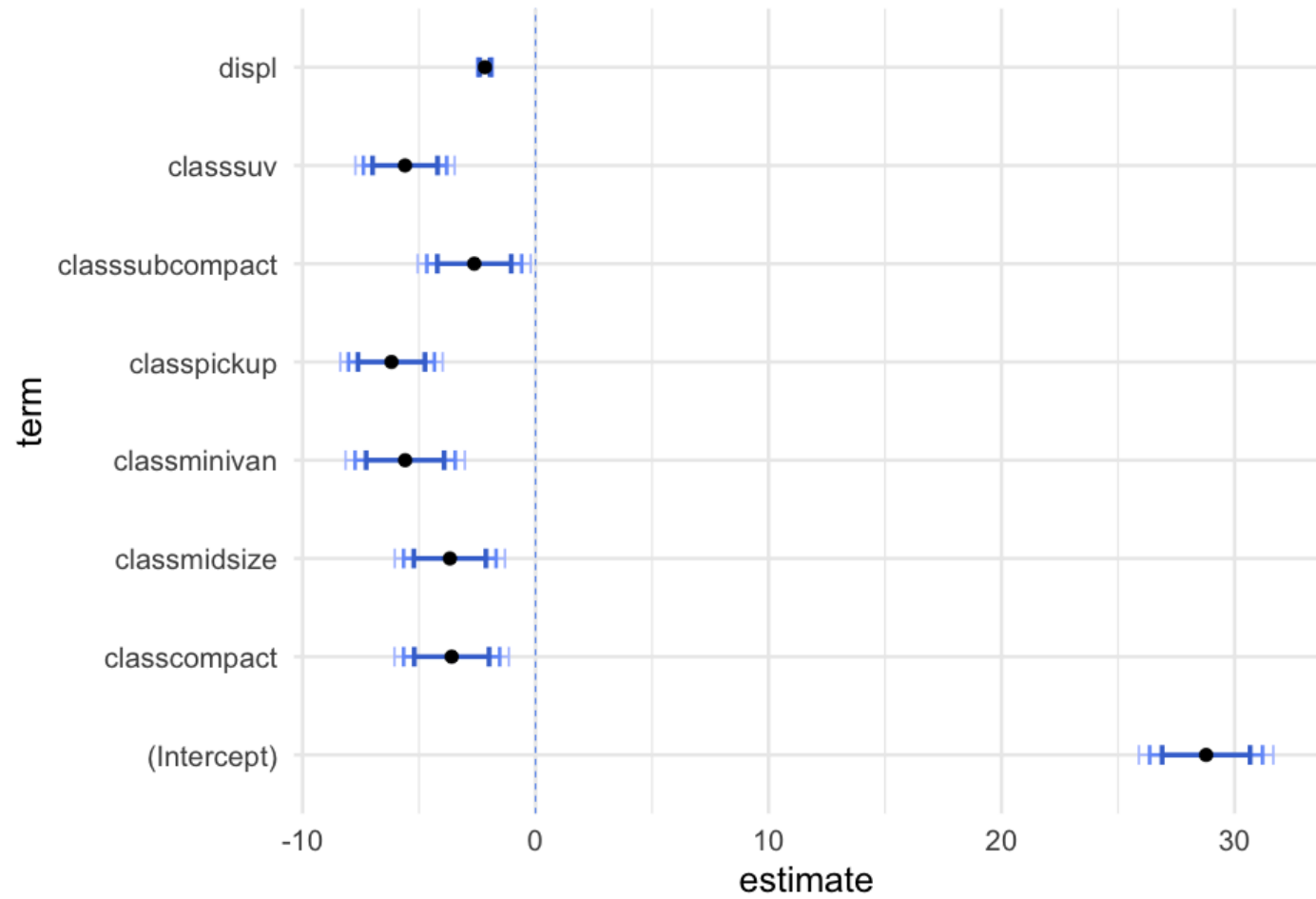
```
## # A tibble: 8 x 7
##   term                estimate std.error statistic      p.value  conf.low
##   <chr>              <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)      28.77682  1.472892  19.53763 1.905873e-50 25.87446
## 2 displ           -2.171562  0.1746638 -12.43281 2.197130e-27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

```
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

```
library(colospace)
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()
```

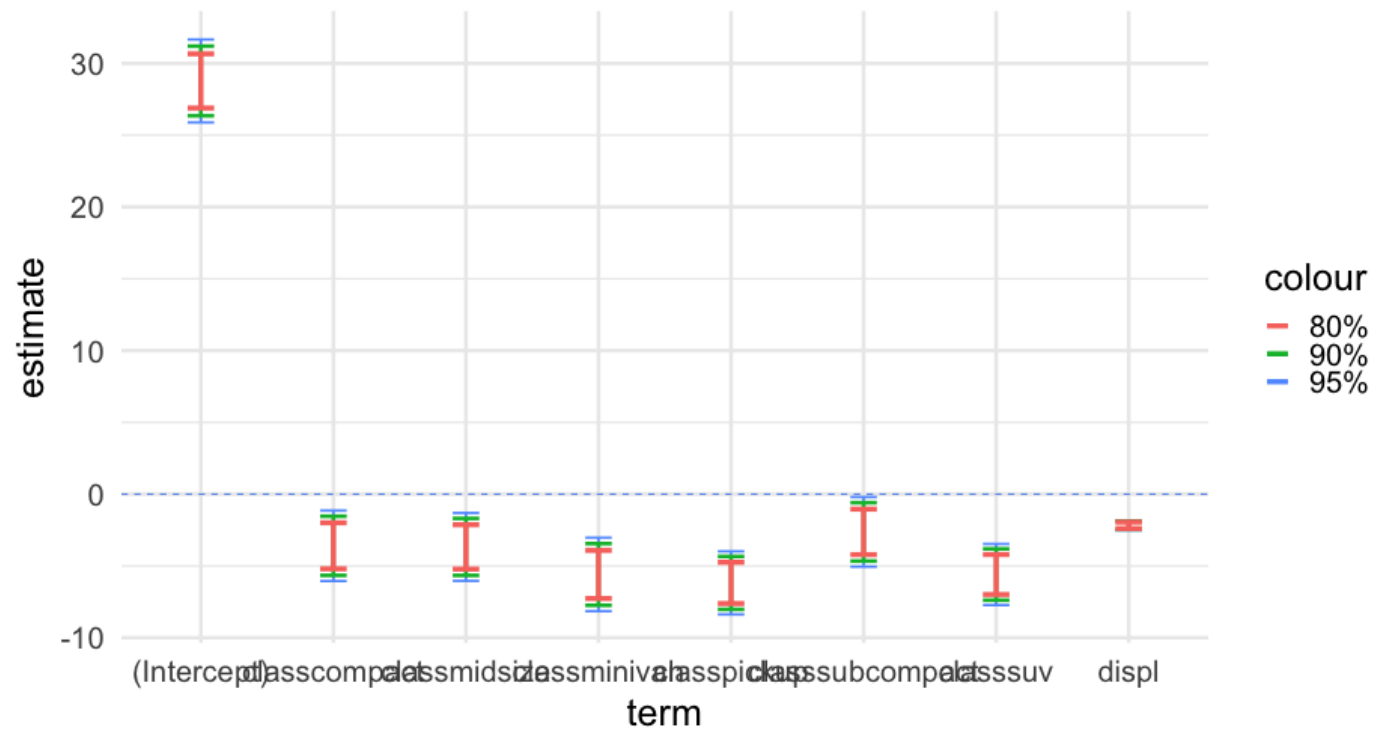


Add levels to legend

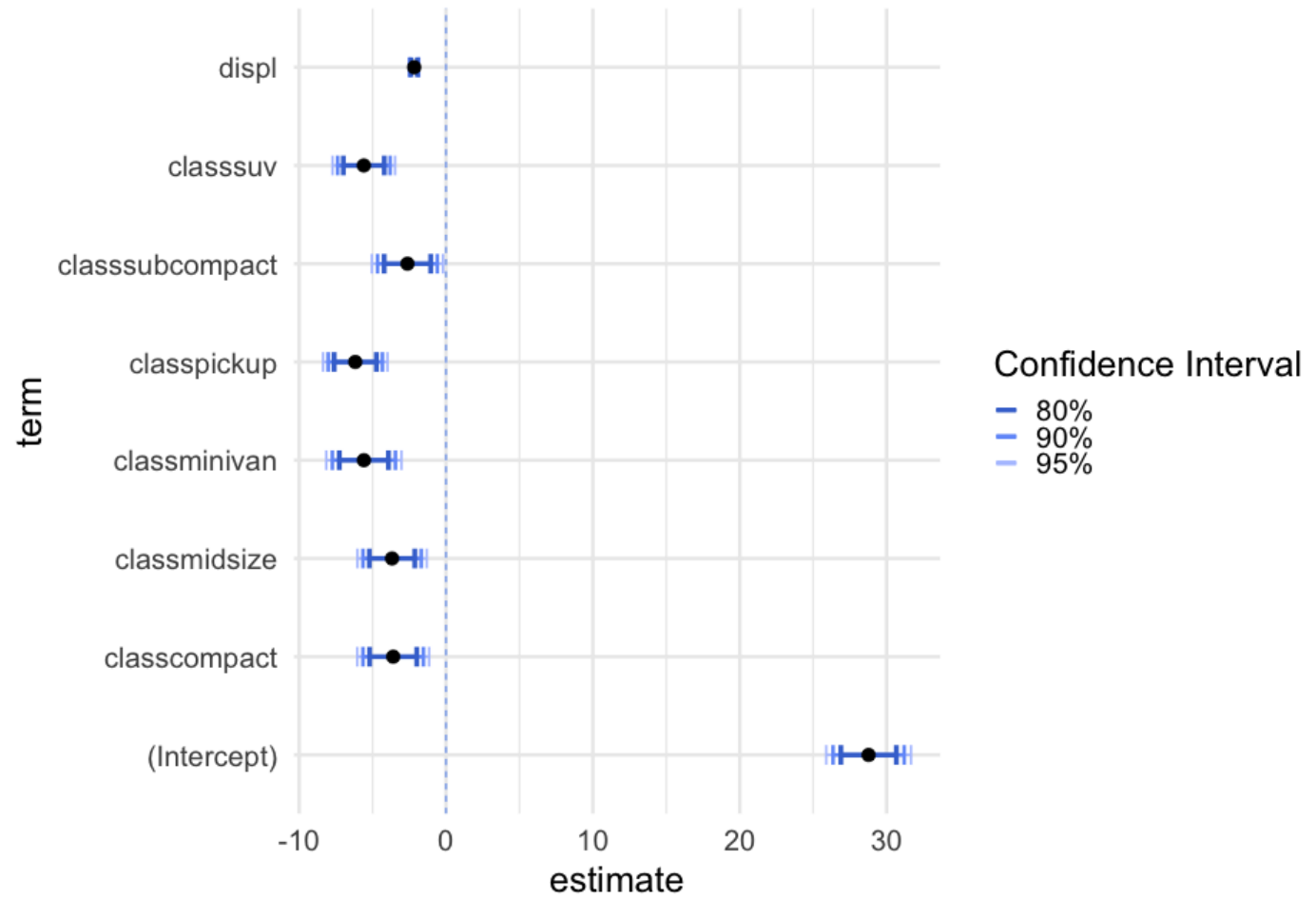
Include the color spec in `aes()`

```
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

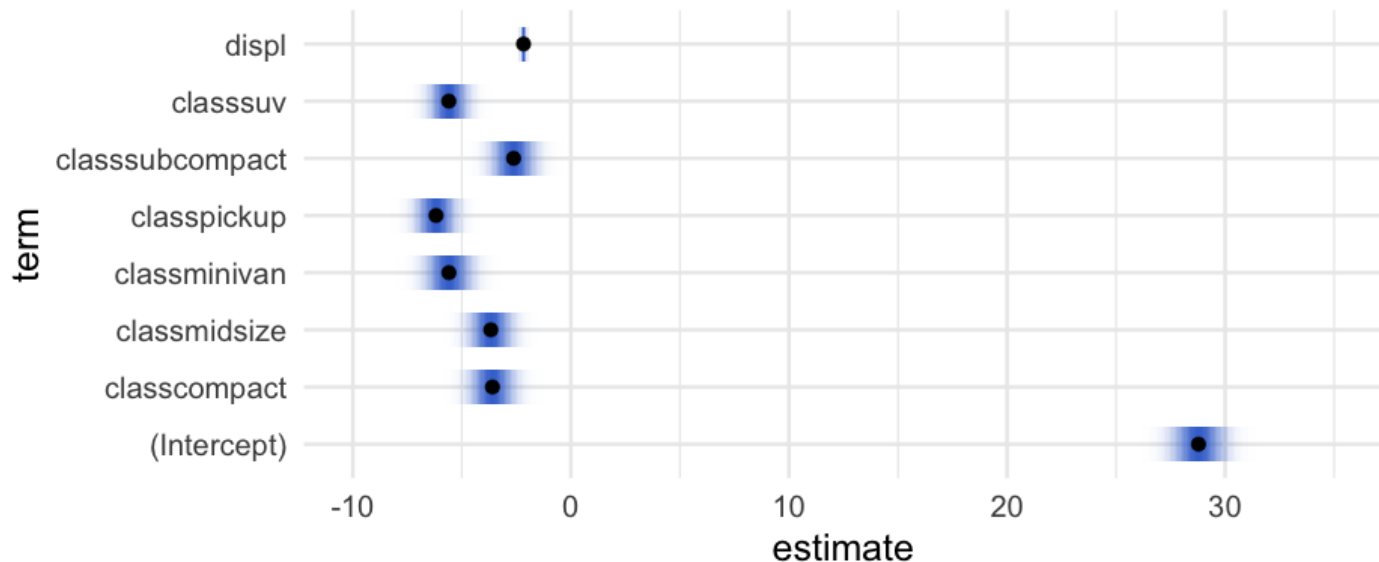


```
p +  
  scale_color_manual("Confidence Interval",  
                     values = c("#4375D3",  
                                lighten("#4375D3", .3),  
                                lighten("#4375D3", .6))) +  
  geom_point() +  
  coord_flip()
```



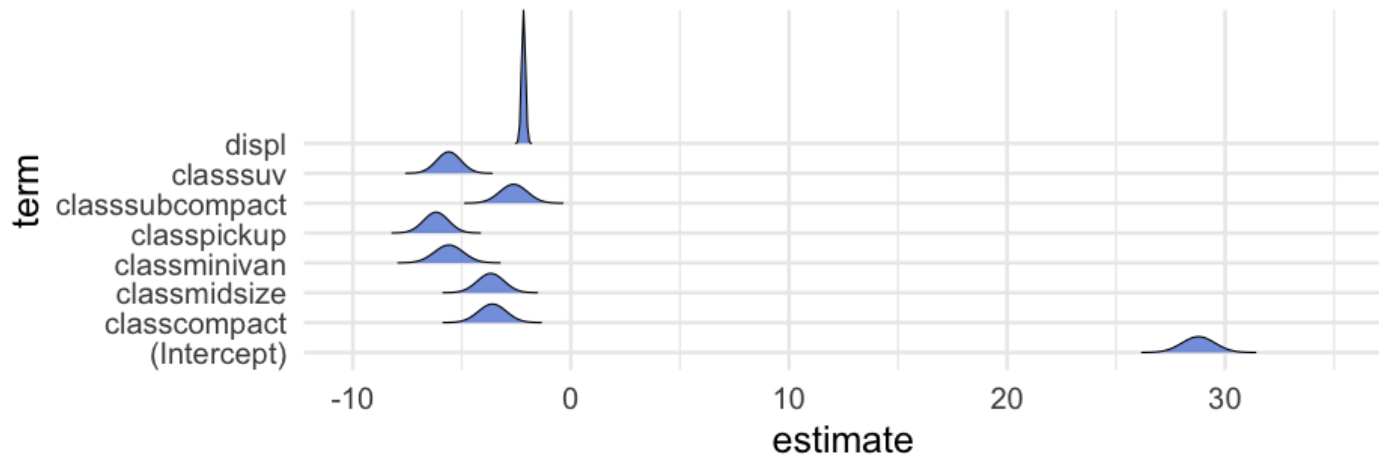
Density stripes

```
#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(ggribes)
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)
```

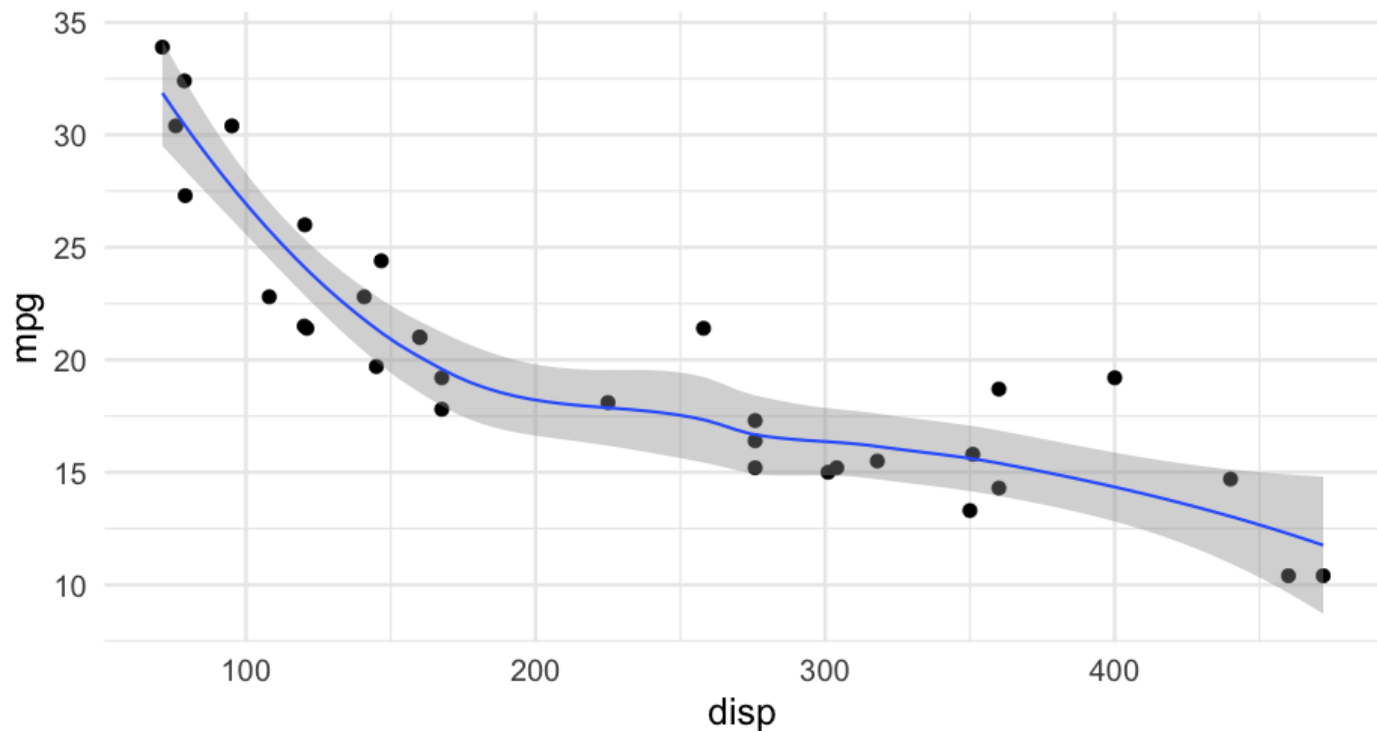


HOPs

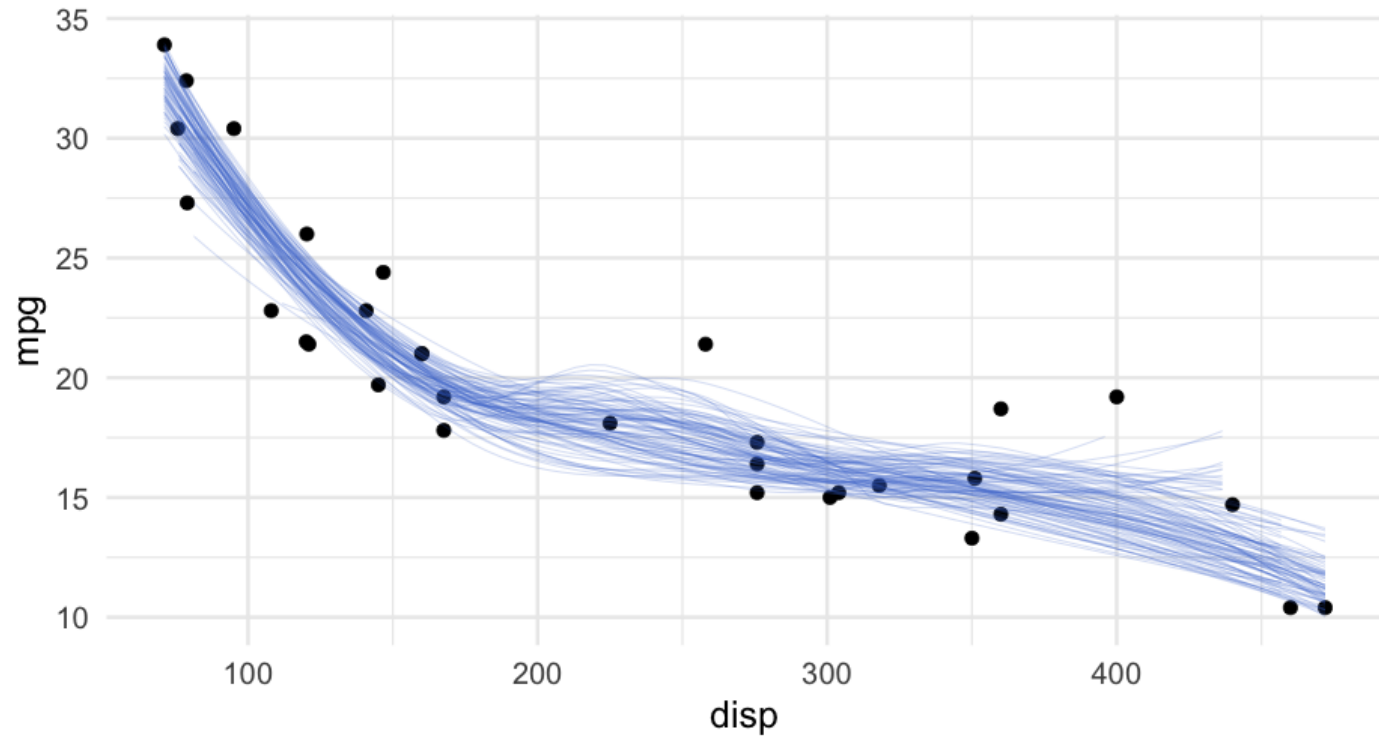
Hypothetical Outcome Plots (and related plots)

Standard regression plot

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



Alternative



How?

Bootstrapping

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

```
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 1  
## [29] 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 3  
## [29] 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  
## [29] 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 1  
## [29] 19 21  1 16
```

Extract samples

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

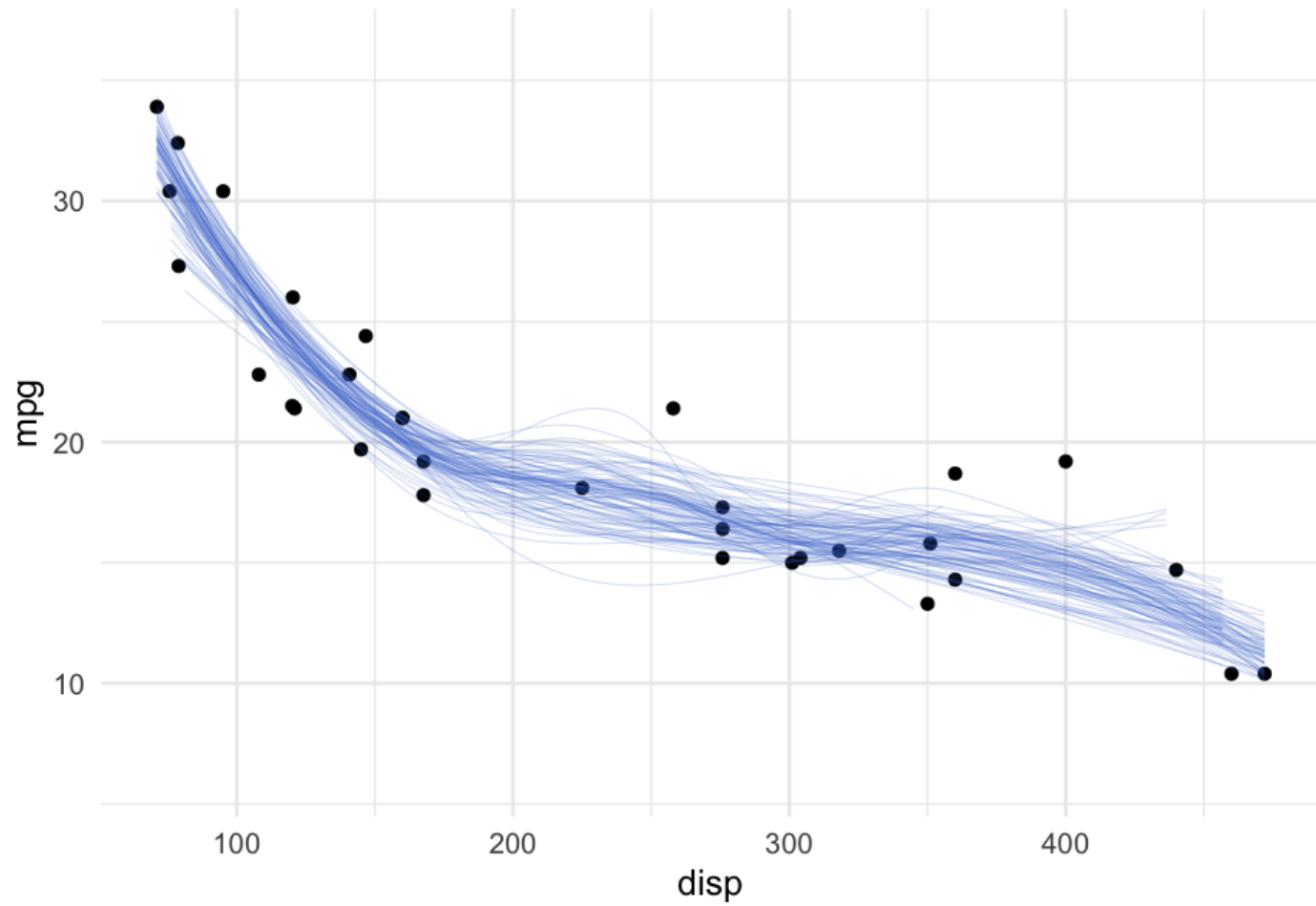
```
##           sample  mpg  cyl  disp  hp drat   wt   qsec vs  am gear c
## Maserati Bora...1      1 15.0   8 301.0 335 3.54 3.57 14.60 0   1    5
## Valiant...2          1 18.1   6 225.0 105 2.76 3.46 20.22 1   0    3
## Volvo 142E...3        1 21.4   4 121.0 109 4.11 2.78 18.60 1   1    4
## Merc 450SE...4        1 16.4   8 275.8 180 3.07 4.07 17.40 0   0    3
## Mazda RX4...5         1 21.0   6 160.0 110 3.90 2.62 16.46 0   1    4
## Merc 450SLC...6       1 15.2   8 275.8 180 3.07 3.78 18.00 0   0    3
```

```
tail(d_samps)
```

```
##           sample  mpg  cyl  disp  hp drat   wt   qsec
## Lincoln Continental.1...3195    100 10.4   8 460.0 215 3.00 5.424 17.82
## Pontiac Firebird...3196         100 19.2   8 400.0 175 3.08 3.845 17.05
## Fiat X1-9.1...3197             100 27.3   4  79.0  66 4.08 1.935 18.90
## Mazda RX4 Wag.3...3198          100 21.0   6 160.0 110 3.90 2.875 17.02
## Hornet Sportabout...3199        100 18.7   8 360.0 175 3.15 3.440 17.02
## Lotus Europa...3200            100 30.4   4  95.1 113 3.77 1.513 16.90
```

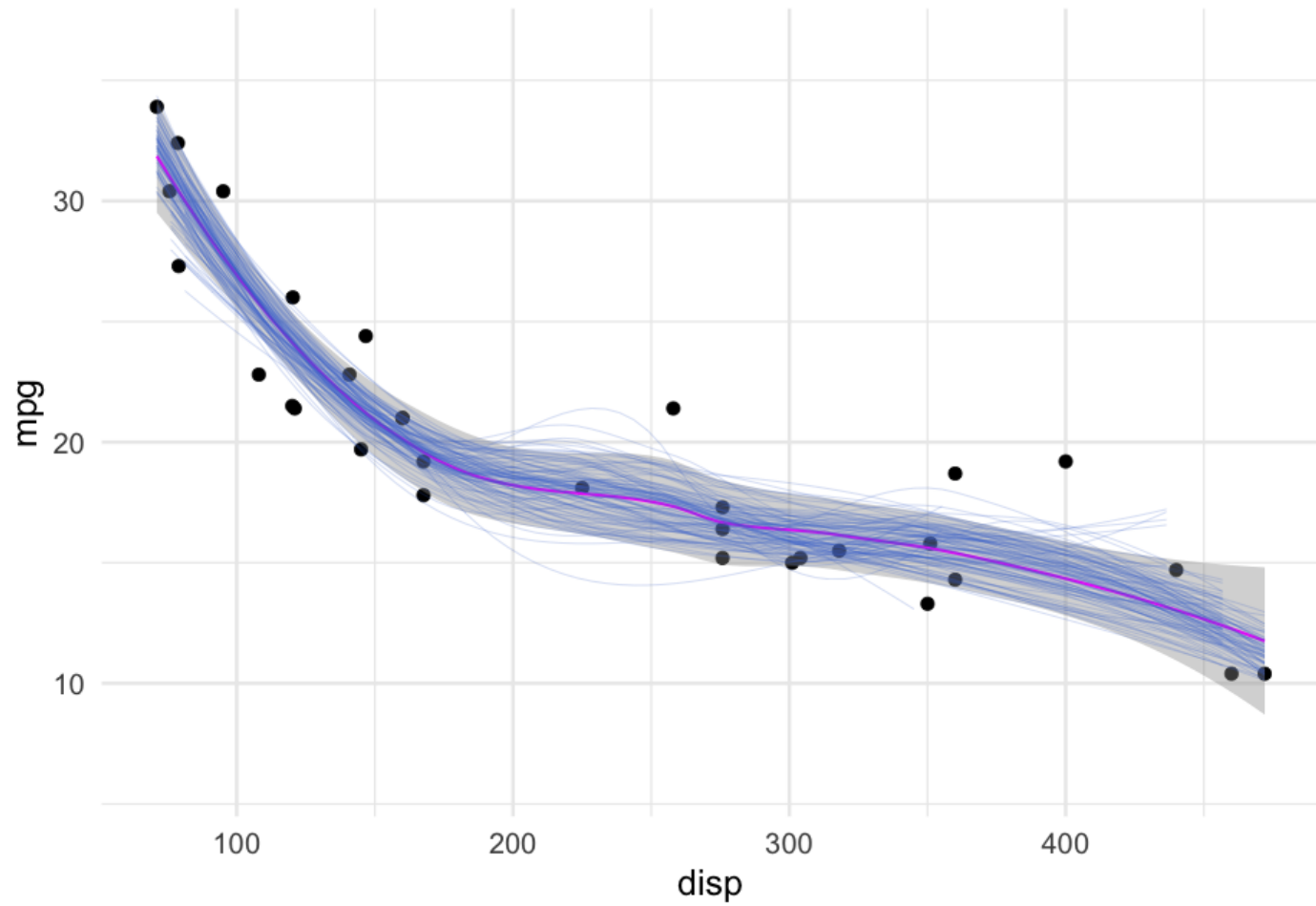
Plot both data sources

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



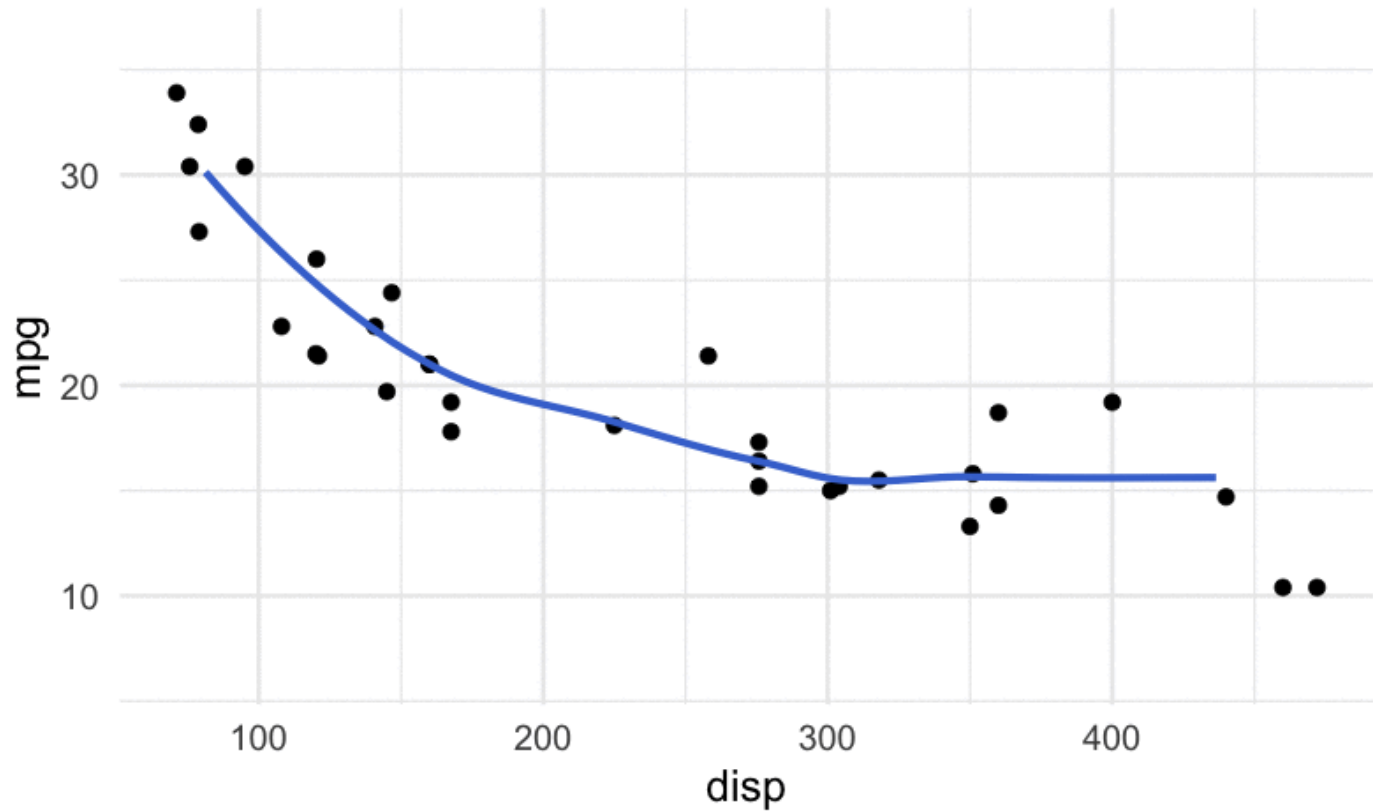
Note, they match up

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

HOPs

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



How?

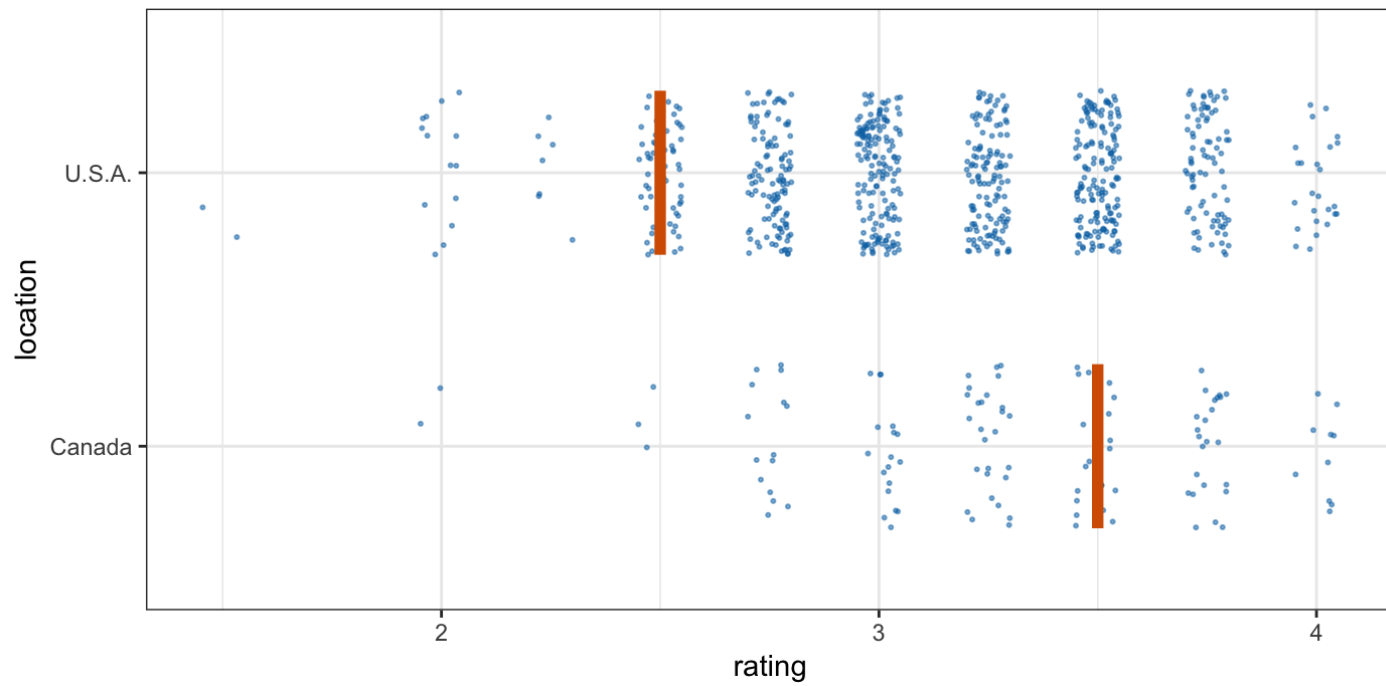
gganimate::transition_states

```
library(gganimate)
ggplot(mtcars, aes(displacement, 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

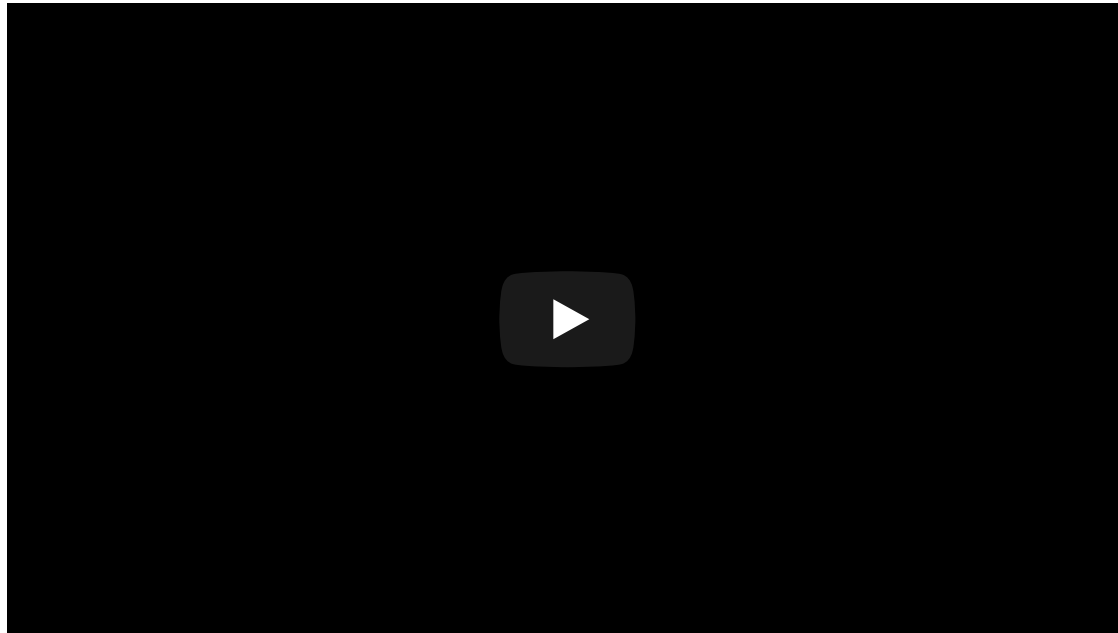
Canada has the highest average rating for their chocolate bars in the world, according to a recent study. But the sample size is also smaller than other countries, like the US.

How certain are we that Canada's chocolate is better?
Compare the ratings of a randomly chosen Canadian chocolate bar to a randomly chosen US Chocolate bar.



Another example

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

Tables and Fonts