

Visualizing uncertainty

Daniel Anderson

Week 6, Class 2

Reviewing

Homework 1

Data viz in the wild

Raleigh

Maggie

Ann-Marie and Murat on Deck

Agenda

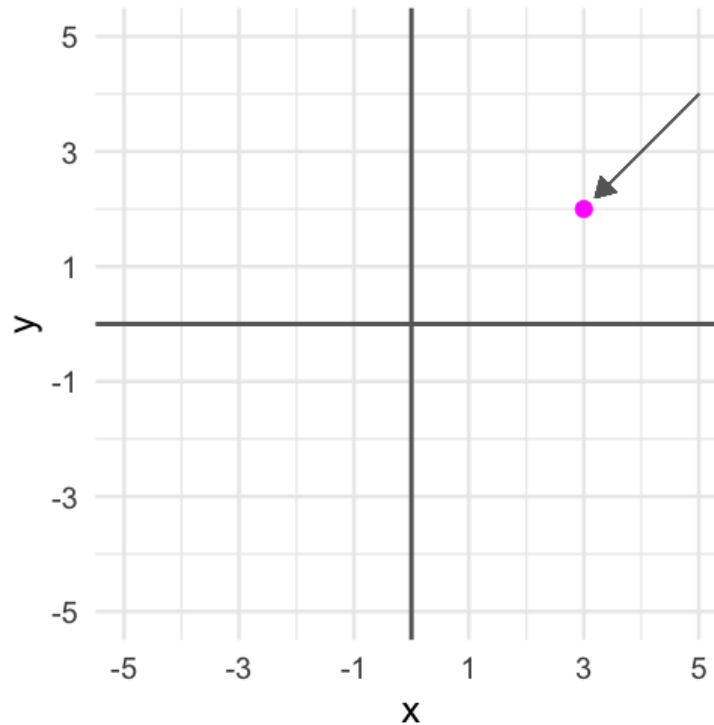
- Common ways of visualizing uncertainty
 - And how to implement them with {ggplot2}
- Framing uncertainty as relative frequencies
 - Discrete probabilities
 - Non-discrete probabilities
 - Understanding AUC calculations
- Understanding standard errors
 - Non-standard ways of visualizing SEs
- HOPs (briefly)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

Learning objectives

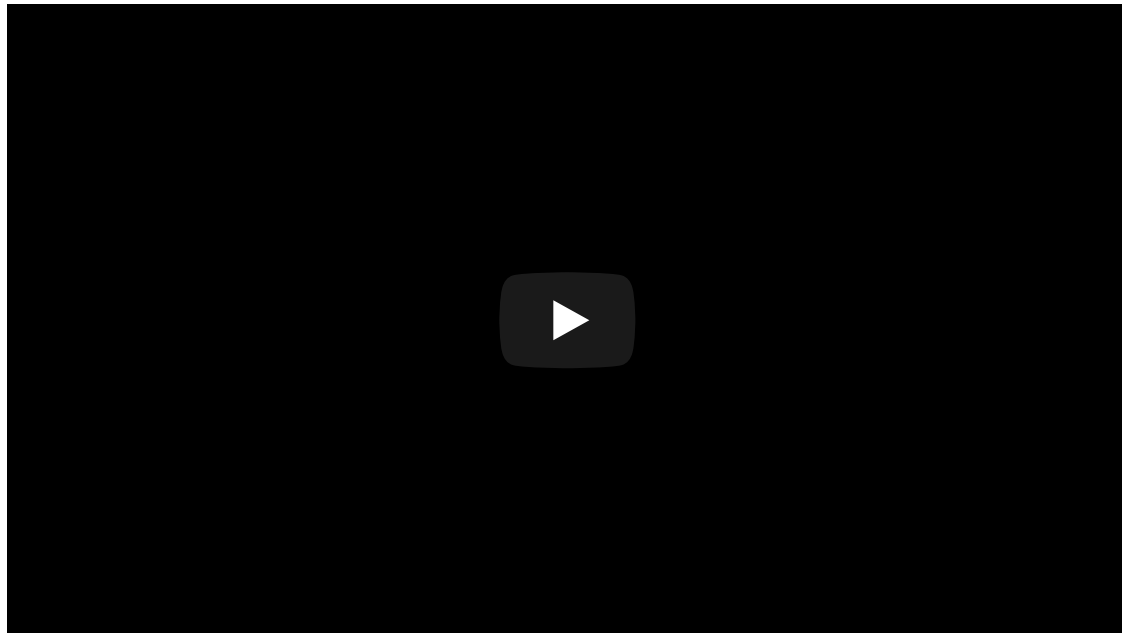
1. Understand there are lots of different ways to visualize uncertainty, and the best method may often be non-standard.
2. Understand how to implement basic methods, and the resources available to you to implement more advanced methods

The primary problem

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



Let's have Dr. Kay explain

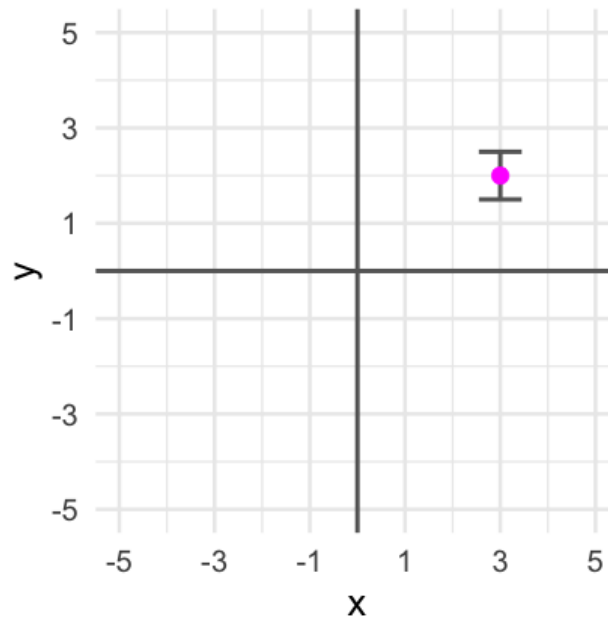


Some secondary problem

- 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`

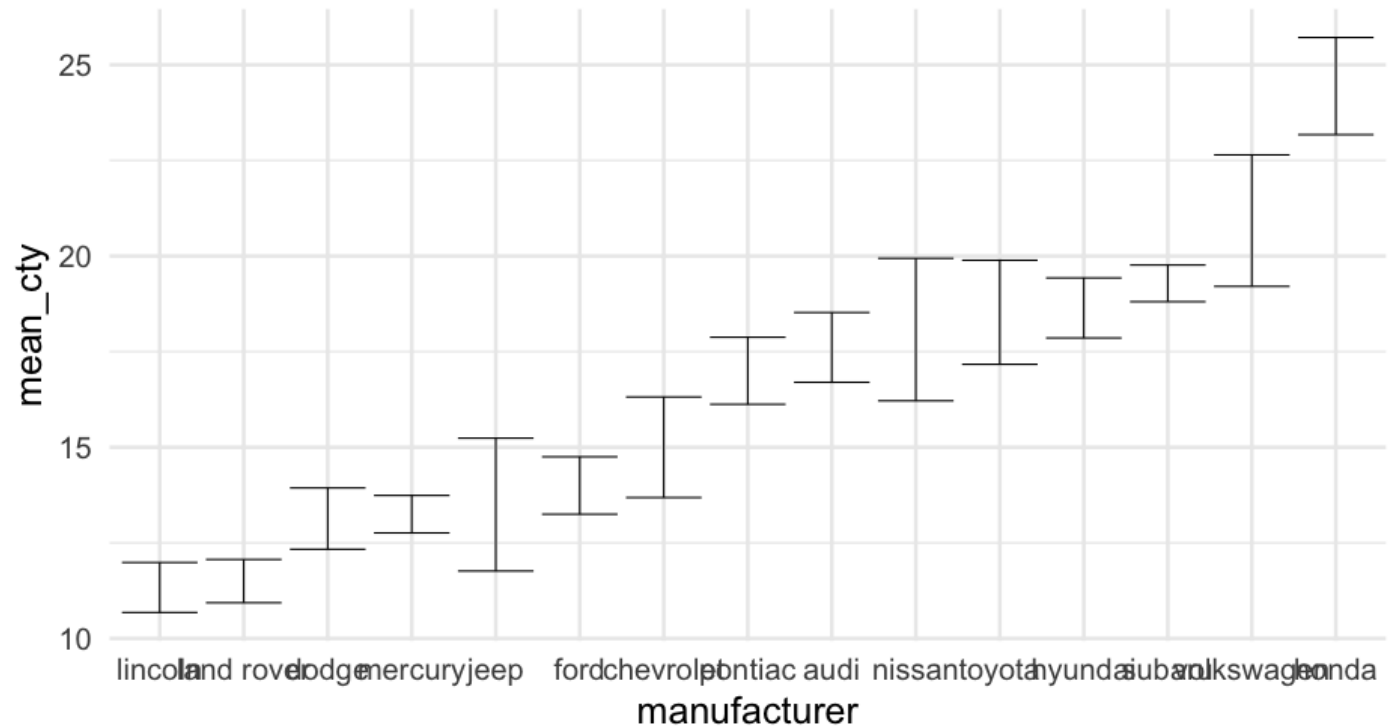
- Requires `xmin` and `xmax`

Example

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

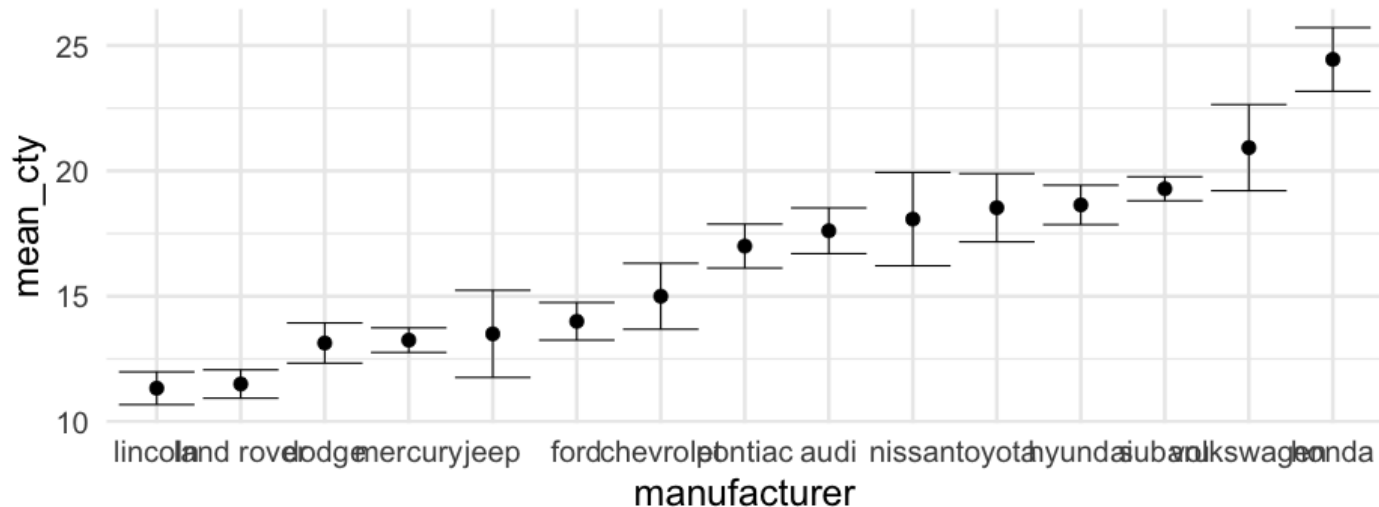
```
mpg_by_man %>%
  mutate(manufacturer = fct_reorder(manufacturer, mean_cty)) %>%
  ggplot(aes(manufacturer, mean_cty)) +
  geom_errorbar(aes(ymin = mean_cty + qnorm(0.025)*se_cty,
                    ymax = mean_cty + qnorm(0.975)*se_cty))
```



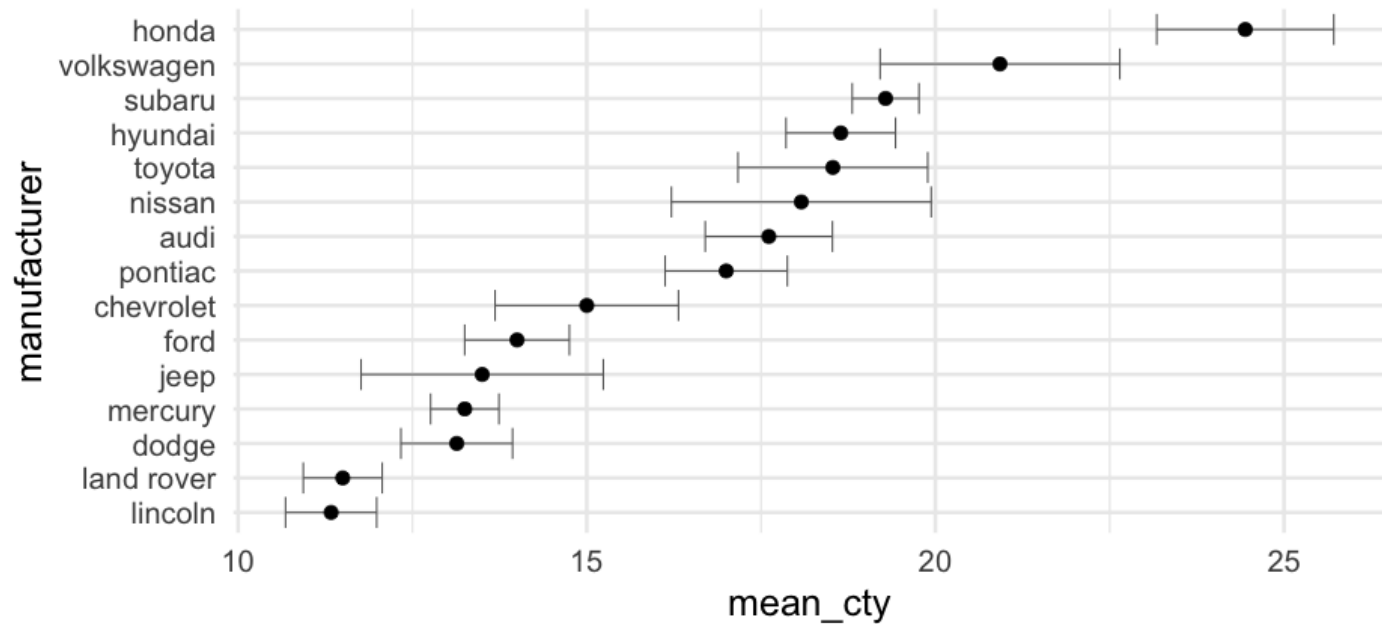
Put points on top

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()
```



```
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()
```

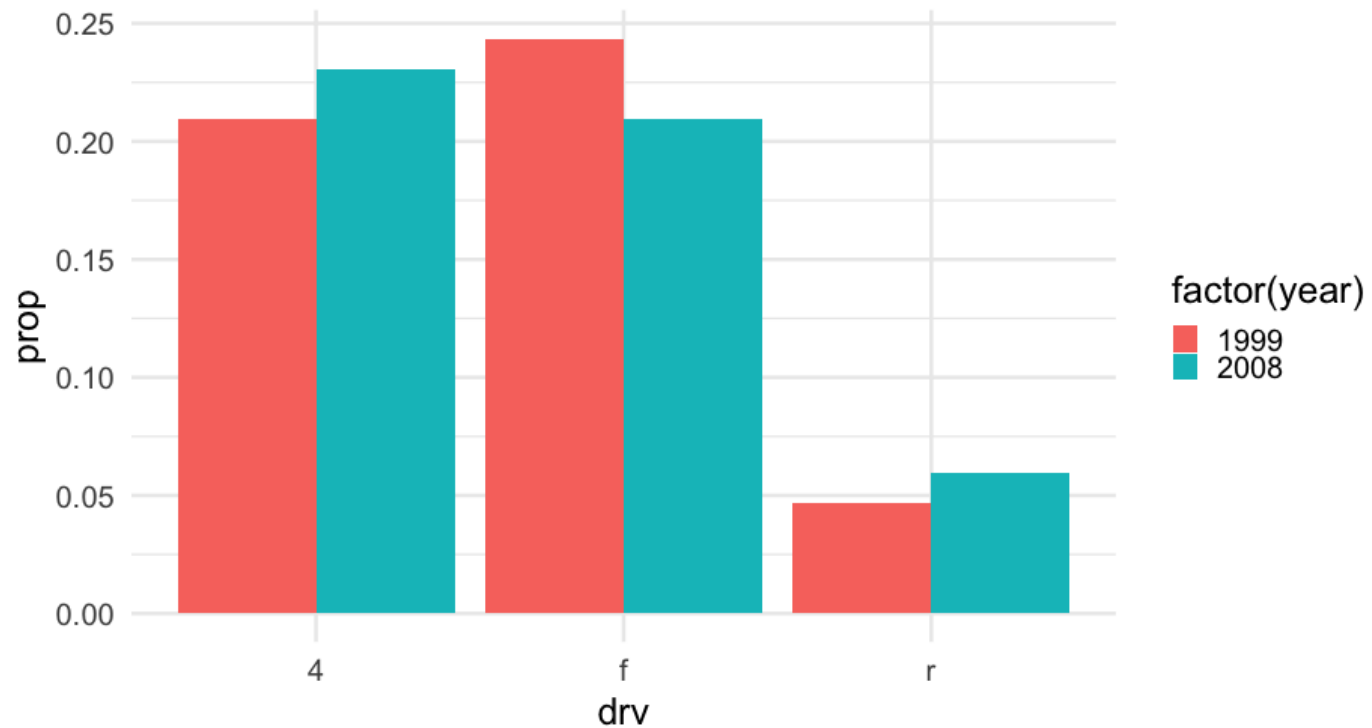


Dodging

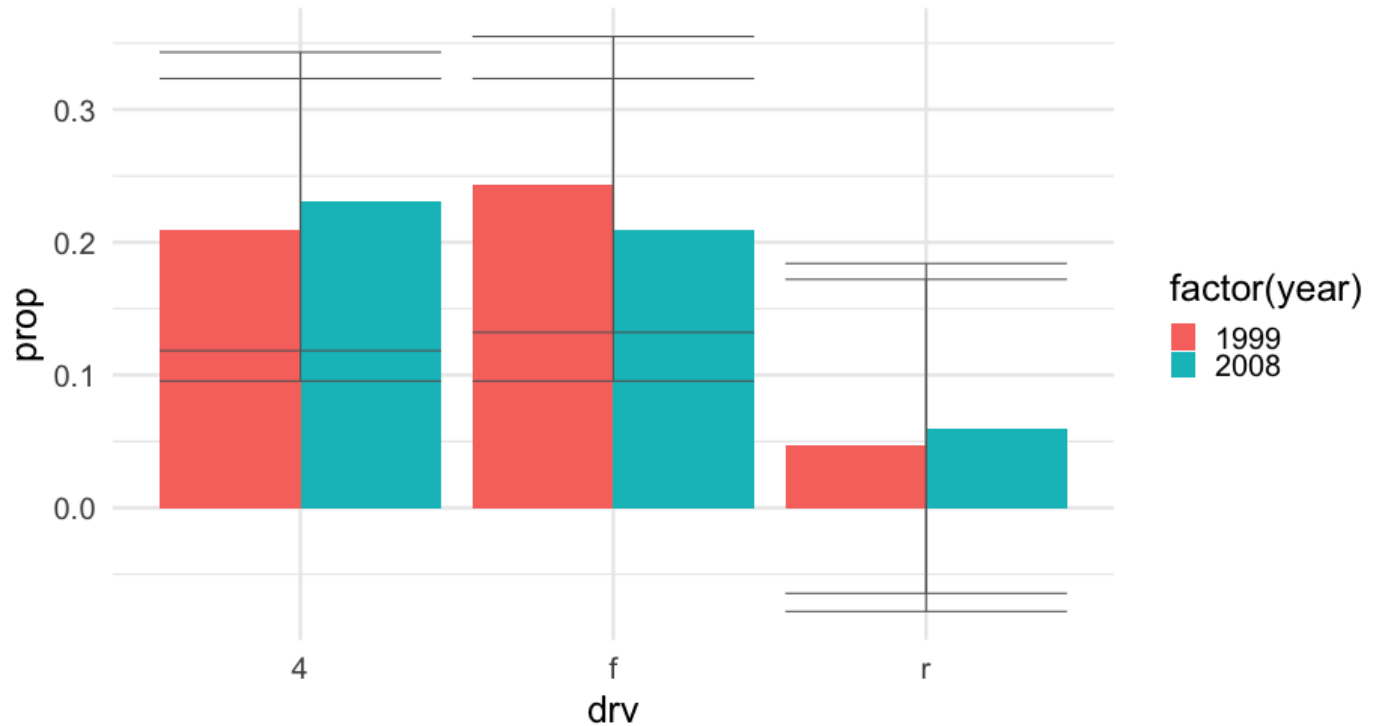
```
props <- mpg %>%  
  count(drv, year) %>%  
  mutate(prop = n/sum(n),  
          prop_se = sqrt((prop*(1-prop)) / n))  
  
head(props)
```

```
## # 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
```

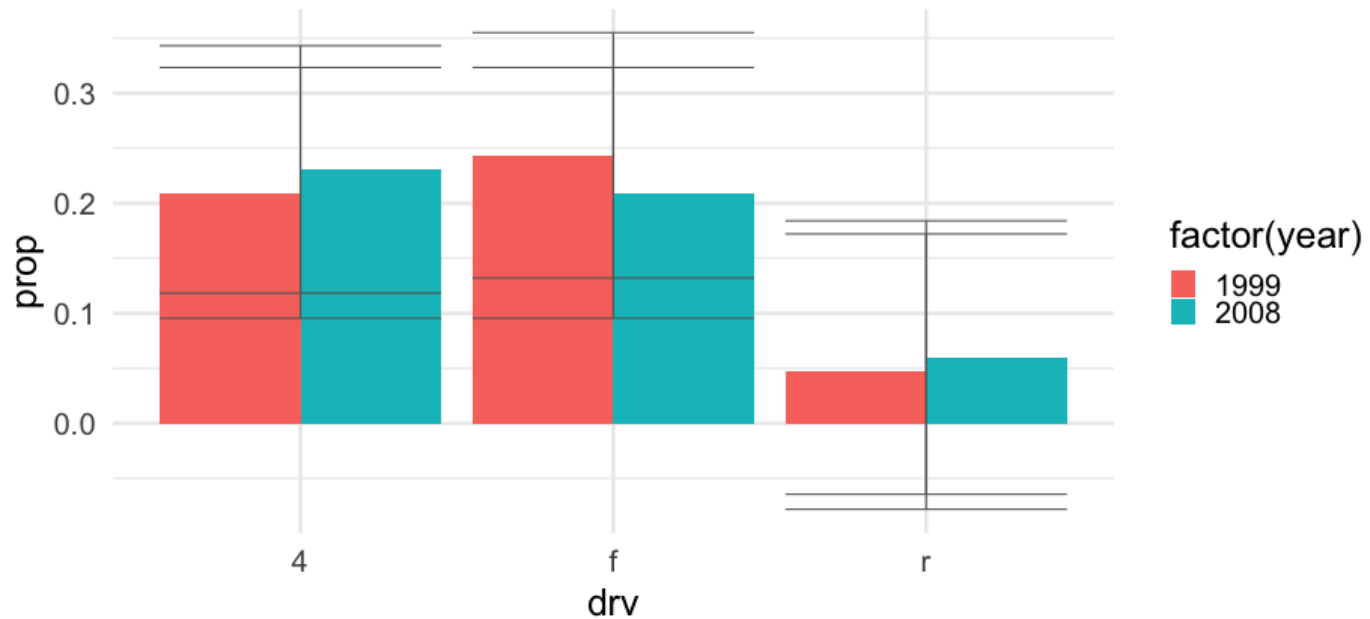
```
ggplot(props, aes(drv, prop)) +  
  geom_col(aes(fill = factor(year)), position = "dodge")
```



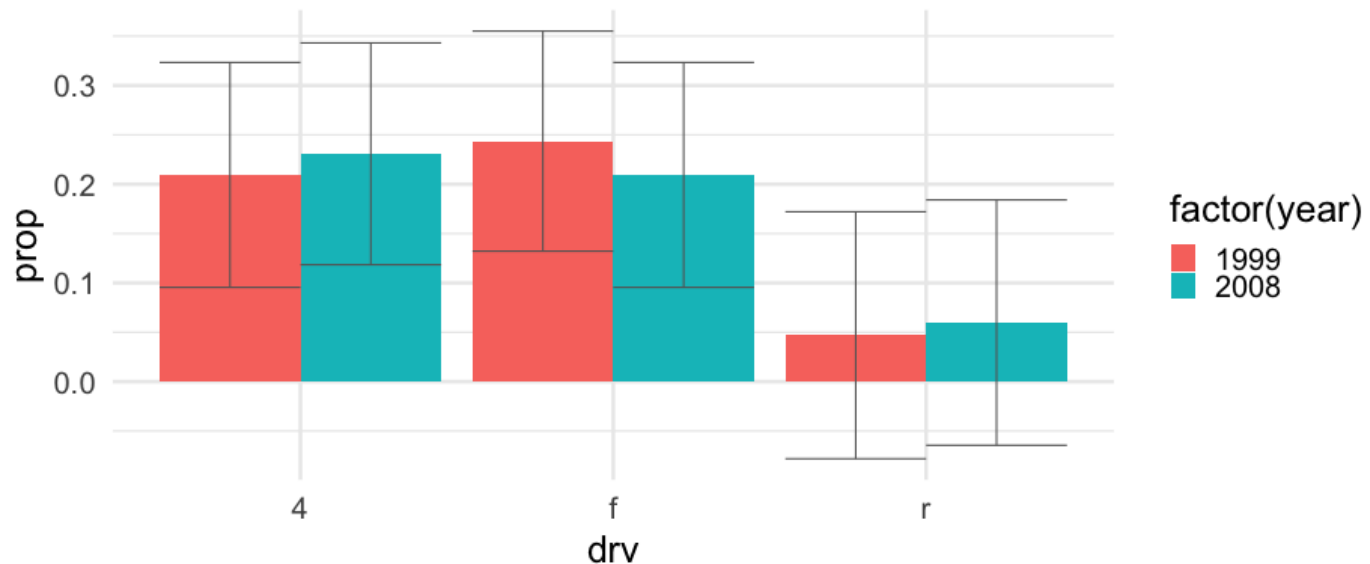

```
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = "dodge") +
  geom_errorbar(aes(ymin = prop - 1.96*prop_se,
                    ymax = prop + 1.96*prop_se),
               color = "gray40")
```



```
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)
```



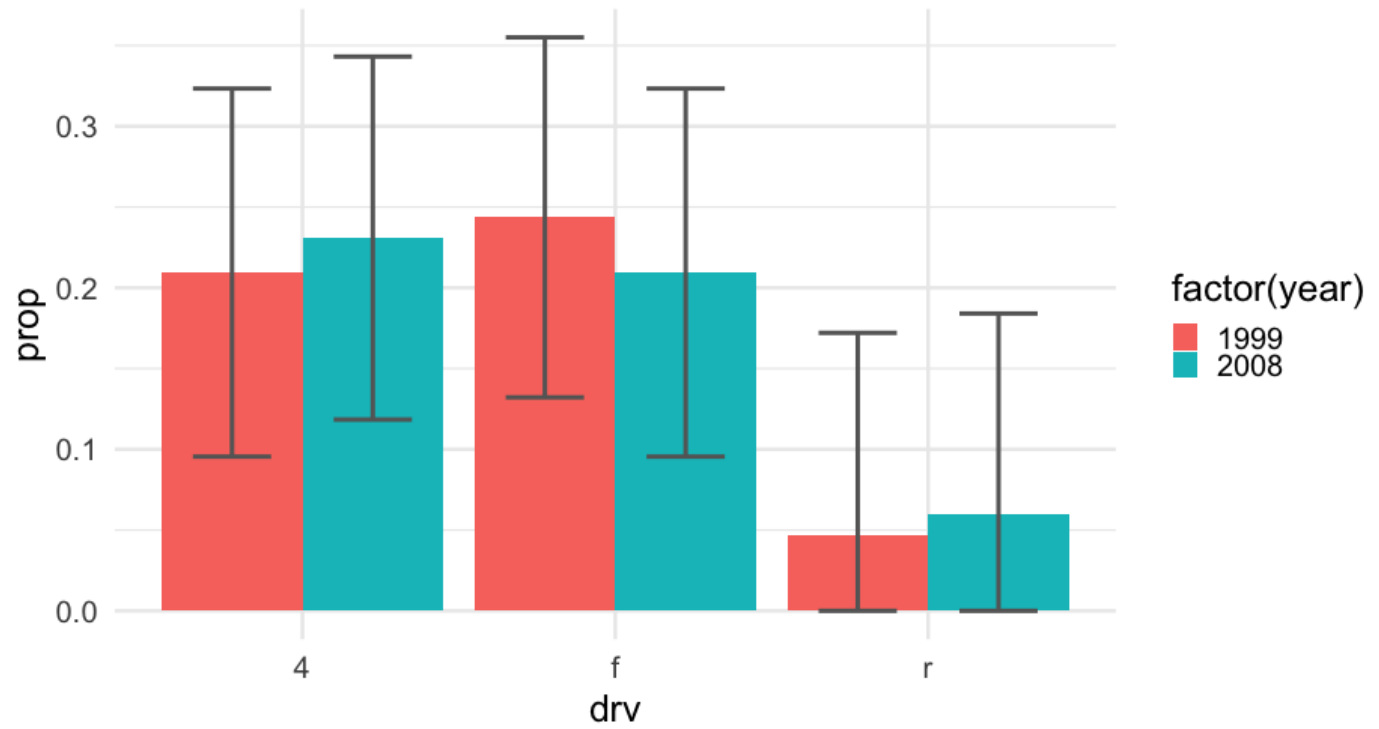
```
pd <- position_dodge(.9)
ggplot(props, aes(drv, prop)) +
  geom_col(aes(fill = factor(year)), position = pd) +
  geom_errorbar(aes(ymin = prop - 1.96*prop_se,
                    ymax = prop + 1.96*prop_se,
                    group = year),
               color = "gray40",
               position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



```

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)

```



Thinking about uncertainty

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

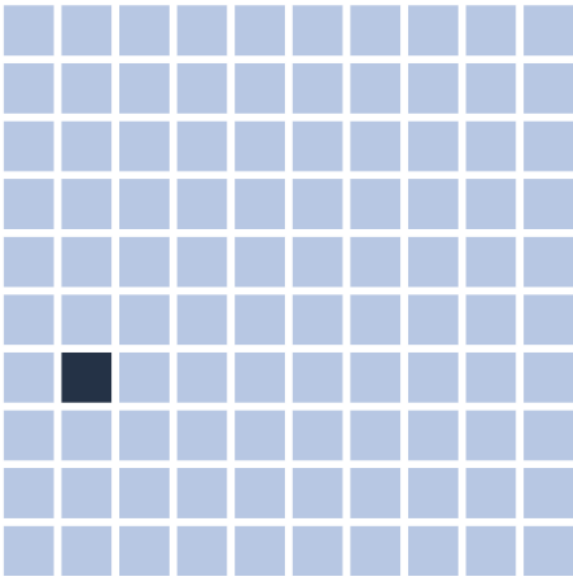
- We are nearly always uncertain of future events (forecasting)
- We can also be uncertain about past events
 - I saw a parked car at 8 AM, but the next time I looked at 2PM it was gone. What time did it leave?

Quantifying uncertainty

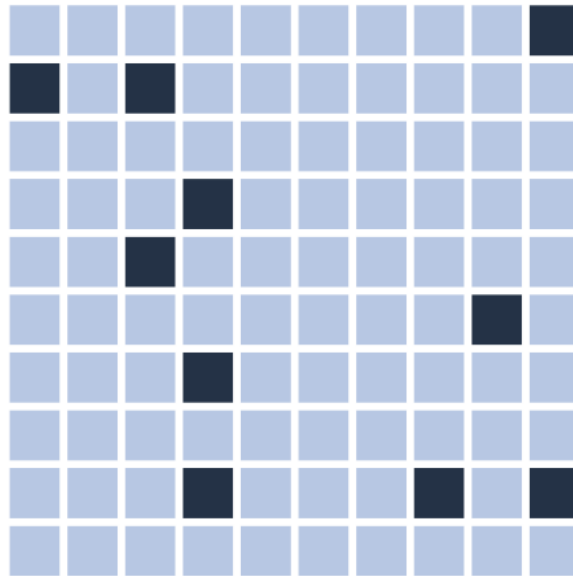
- We quantify our uncertainty mathematically using probability
- Framing probabilities as frequencies is generally more intuitive

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
```

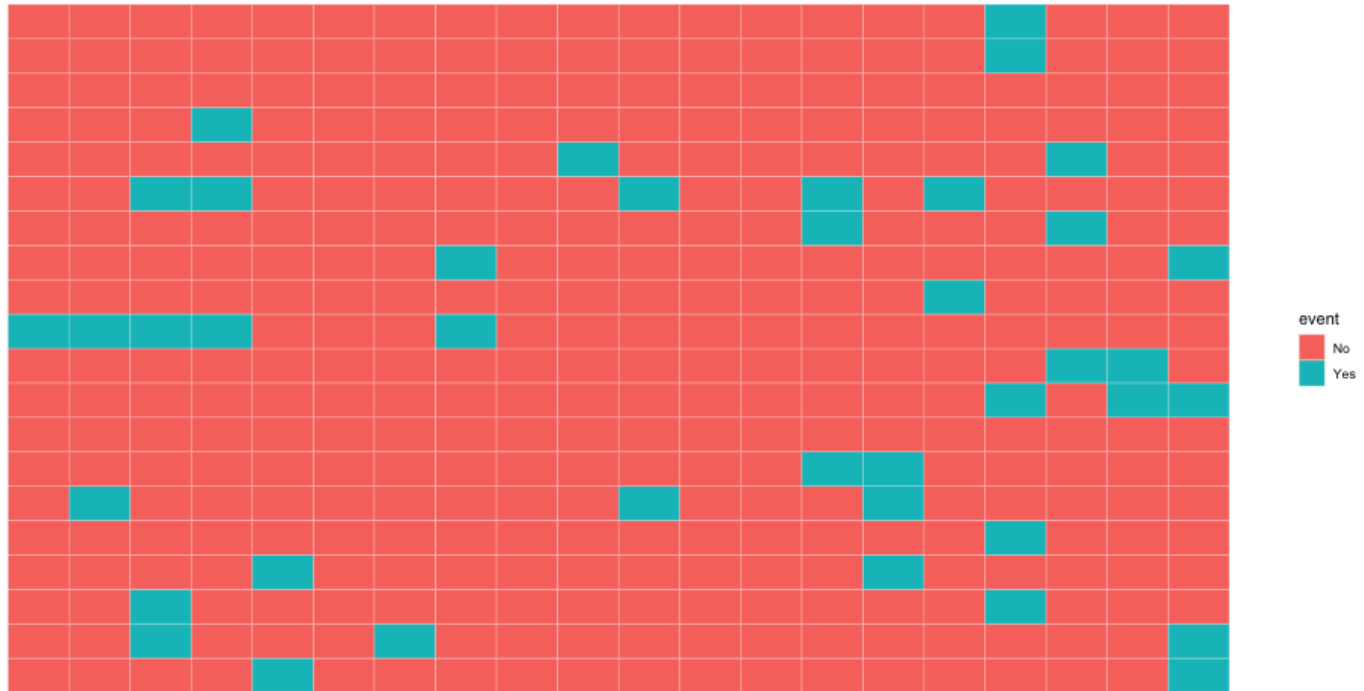
Create the variable

```
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
```

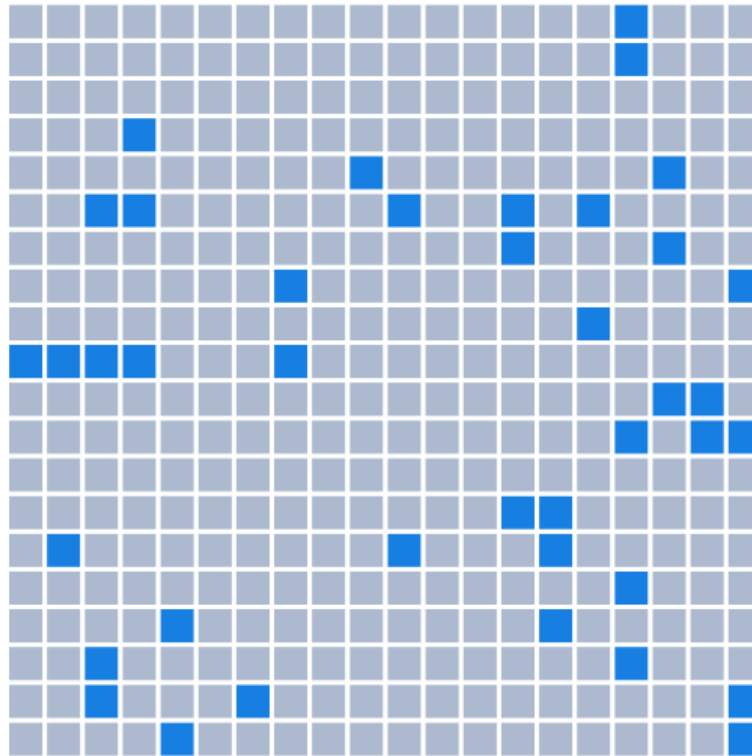
Fill in

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



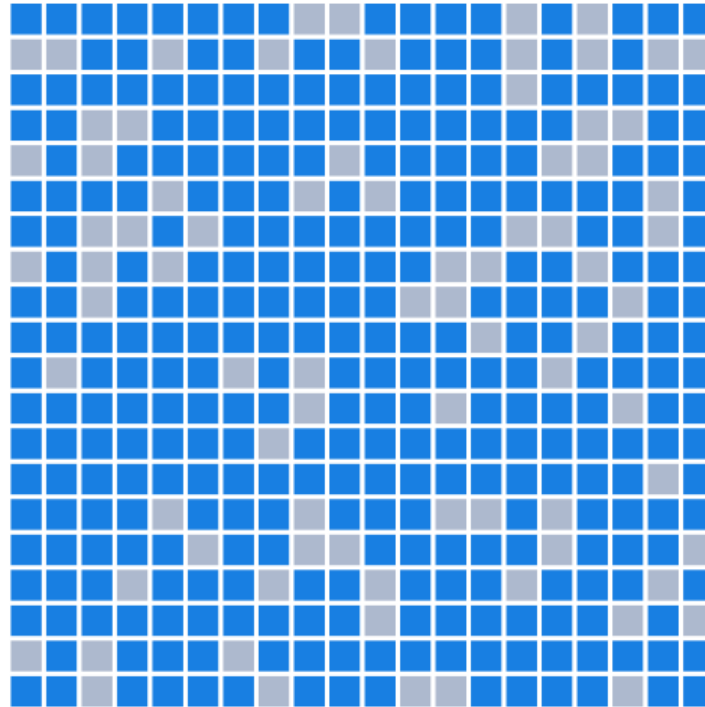
Customize

```
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

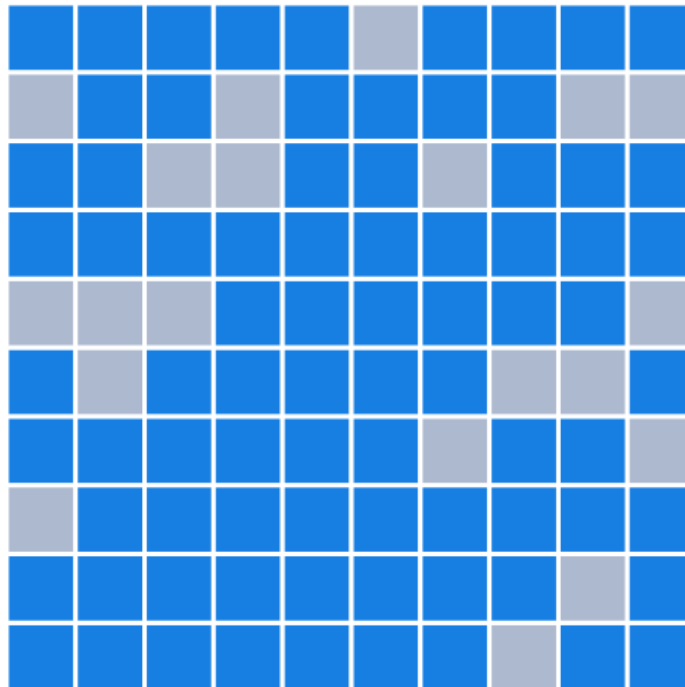
Chance of rain



Event Occurred ■ No ■ Yes

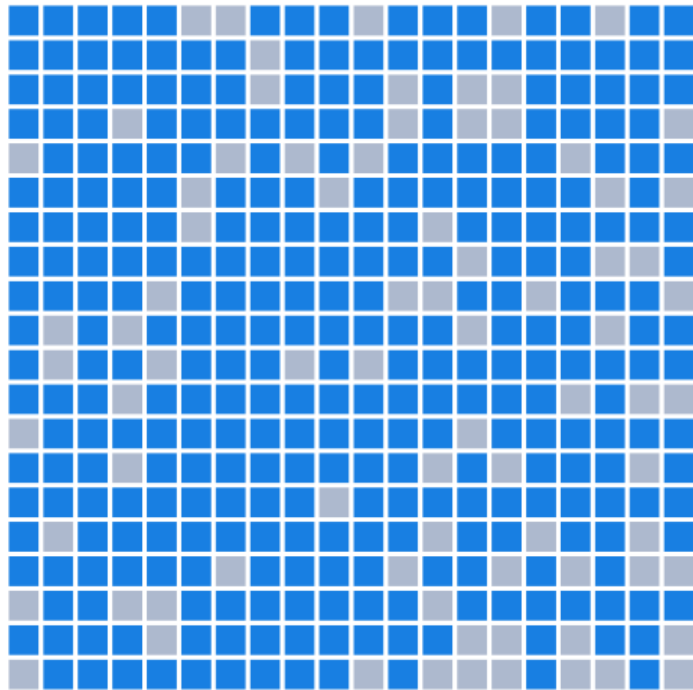
Vary grid size

10 x 10



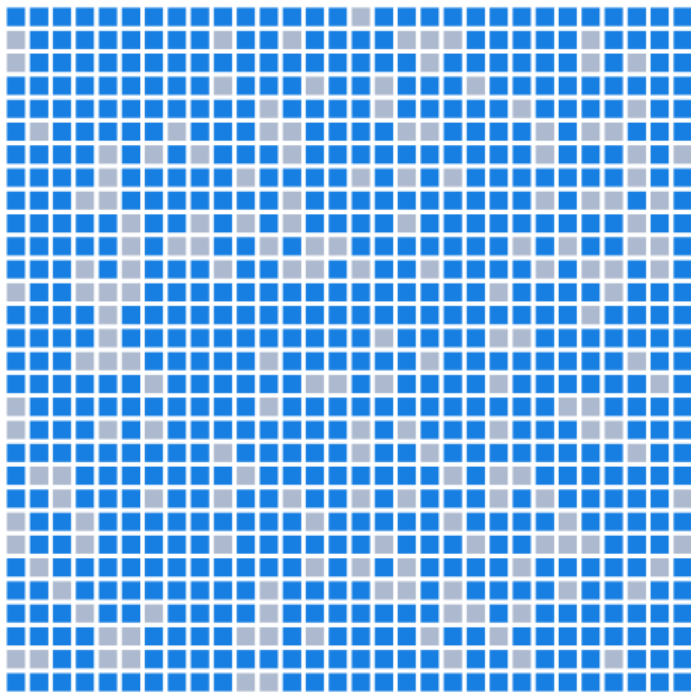
Vary grid size

20 x 20



Vary grid size

30 x 30



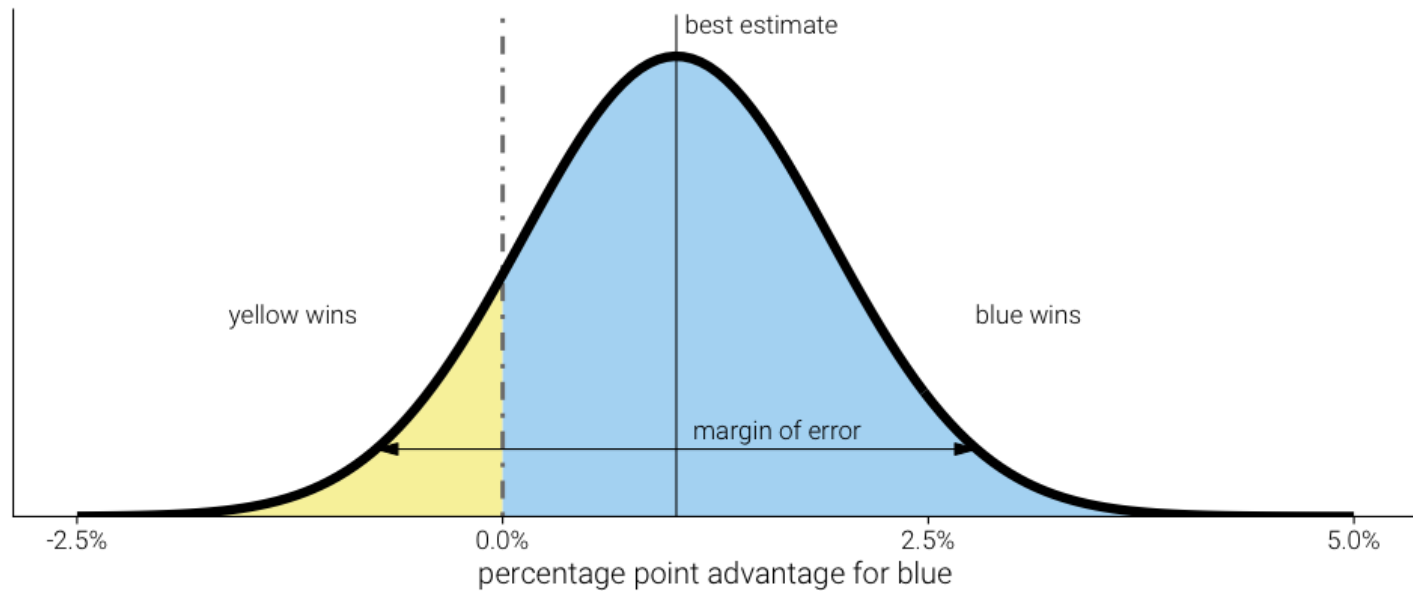
(probs too many)

Non-discrete
probabilities

Hypothetical

Blue party has 1% advantage 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 $sd = 0.9$

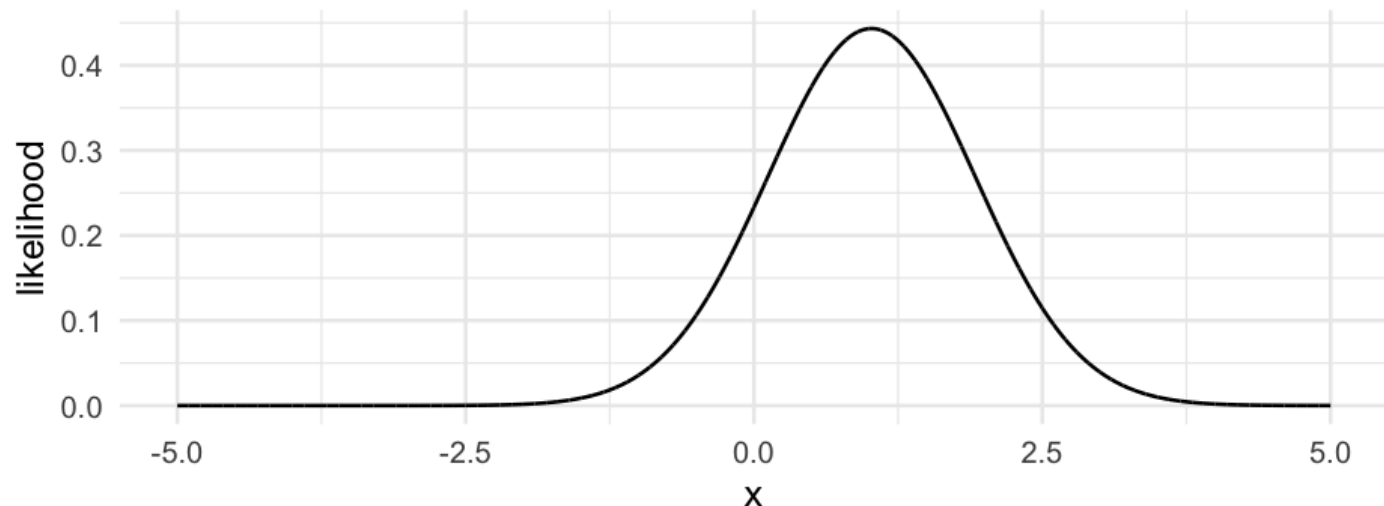
.

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

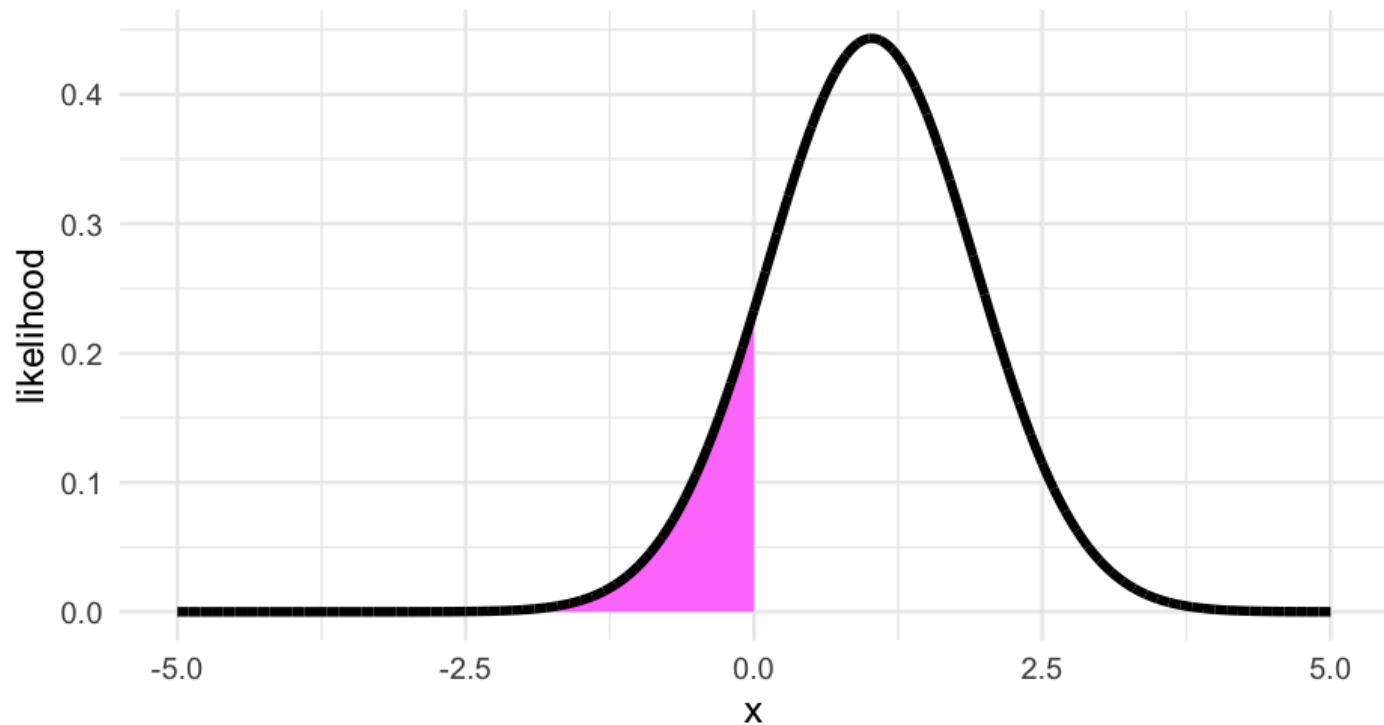
The hard way

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] 0.129
```

Easier: Simulate

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

```
##
## FALSE  TRUE
##  0.13  0.87
```

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] -1.07371 -0.67271 -0.46037 -0.30821 -0.18668 -0.08388 0.00625 0.09775 0.19129 0.28699 0.38494 0.48514 0.58779 0.69294 0.79959 0.90774 1.01739 1.12854 1.24119 1.35534 1.47099 1.58814 1.70679 1.82694 1.94859 2.07174 2.19639 2.32254 2.45019 2.57934 2.71000 2.84214 2.97579 3.11094 3.24759 3.38574 3.52539 3.66654 3.80919 3.95334 4.09899 4.24614 4.39479 4.54494 4.69659 4.84974 4.99439 5.14054 5.28819 5.43734 5.58799 5.73914 5.89179 6.04594 6.19259 6.34174 6.49339 6.64654 6.79119 6.93734 7.08499 7.23414 7.38479 7.53694 7.69059 7.84574 7.99239 8.14054 8.28919 8.43934 8.59099 8.74414 8.89879 9.05494 9.21259 9.37174 9.53239 9.69454 9.85819 10.02334 10.18999 10.35814 10.52779 10.69894 10.87159 11.04574 11.22139 11.39854 11.57719 11.75734 11.93899 12.12214 12.30679 12.49294 12.68059 12.86974 13.06039 13.25254 13.44619 13.64134 13.83799 14.03614 14.23579 14.43694 14.63959 14.84374 15.04939 15.25654 15.46519 15.67534 15.88699 16.09914 16.31279 16.52794 16.74459 16.96274 17.18239 17.40354 17.62619 17.85034 18.07599 18.30314 18.53179 18.76194 18.99359 19.22674 19.46139 19.69754 19.93519 20.17434 20.41499 20.65714 20.90079 21.14594 21.39259 21.64074 21.89039 22.14154 22.39419 22.64834 22.90399 23.16114 23.41979 23.67994 23.94159 24.20474 24.46939 24.73554 25.00319 25.27234 25.54299 25.81514 26.08879 26.36394 26.64059 26.91874 27.19839 27.47954 27.76219 28.04634 28.33199 28.61914 28.90779 29.19794 29.48959 29.78274 30.07739 30.37354 30.67119 30.97034 31.27099 31.57314 31.87679 32.18194 32.48859 32.79674 33.10639 33.41754 33.73019 34.04434 34.35999 34.67714 34.99579 35.31594 35.63759 35.96074 36.28539 36.61154 36.93919 37.26834 37.59899 37.93114 38.26479 38.59994 38.93659 39.27474 39.61439 39.95554 40.29819 40.64234 40.98799 41.33514 41.68379 42.03394 42.38559 42.73874 43.09339 43.44954 43.80719 44.16634 44.52699 44.88914 45.25279 45.61794 45.98459 46.35274 46.72239 47.09354 47.46619 47.84034 48.21599 48.59314 48.97179 49.35194 49.73359 50.11674 50.50139 50.88754 51.27519 51.66434 52.05499 52.44714 52.84079 53.23594 53.63259 54.03074 54.43039 54.83154 55.23419 55.63834 56.04399 56.45114 56.85979 57.26994 57.68159 58.09474 58.50939 58.92554 59.34319 59.76234 60.18299 60.60514 61.02879 61.45394 61.88059 62.30874 62.73839 63.16954 63.60219 64.03634 64.47199 64.90914 65.34779 65.78794 66.22959 66.67274 67.11739 67.56354 68.01119 68.46034 68.91099 69.36314 69.81679 70.27194 70.72859 71.18674 71.64639 72.10754 72.57019 73.03434 73.49999 73.96714 74.43579 74.90594 75.37859 75.85374 76.33039 76.80854 77.28819 77.76934 78.25199 78.73614 79.22179 79.70894 80.19759 80.68774 81.17939 81.67254 82.16719 82.66334 83.16099 83.66014 84.16079 84.66294 85.16659 85.67174 86.17839 86.68654 87.19619 87.70734 88.21999 88.73414 89.24979 89.76694 90.28559 90.80574 91.32739 91.85054 92.37519 92.90134 93.42899 93.95814 94.48879 95.02094 95.55459 96.08974 96.62639 97.16454 97.70419 98.24534 98.78799 99.33214 99.87879 100.42694 100.97659 101.52774 102.08039 102.63454 103.19019 103.74734 104.30599 104.86614 105.42779 105.99094 106.55559 107.12174 107.68939 108.25854 108.82919 109.40134 109.97499 110.55014 111.12679 111.70494 112.28459 112.86574 113.44839 114.03254 114.61819 115.20534 115.79399 116.38414 116.97579 117.56894 118.16359 118.75974 119.35739 119.95654 120.55719 121.15934 121.76299 122.36814 122.97479 123.58294 124.19259 124.80374 125.41639 126.03054 126.64619 127.26334 127.88199 128.50214 129.12379 129.74694 130.37159 130.99774 131.62539 132.25454 132.88519 133.51734 134.15099 134.78614 135.42279 136.06094 136.70059 137.34174 137.98439 138.62854 139.27419 139.92134 140.57000 141.22014 141.87179 142.52494 143.17959 143.83574 144.49339 145.15254 145.81319 146.47534 147.13899 147.80414 148.47079 149.13894 149.80859 150.47974 151.15239 151.82654 152.50219 153.17934 153.85799 154.53814 155.21979 155.90294 156.58759 157.27374 157.96139 158.65054 159.34119 160.03334 160.72699 161.42214 162.11879 162.81694 163.51659 164.21774 164.92039 165.62454 166.33019 167.03734 167.74599 168.45614 169.16779 169.88094 170.59559 171.31174 172.02939 172.74854 173.46919 174.19134 174.91499 175.64014 176.36679 177.09494 177.82459 178.55574 179.28839 180.02254 180.75819 181.49534 182.23399 182.97414 183.71579 184.45894 185.20359 185.94974 186.69739 187.44654 188.19719 188.94934 189.70299 190.45814 191.21479 191.97294 192.73259 193.49374 194.25639 195.02054 195.78619 196.55334 197.32199 198.09214 198.86379 199.63694 200.41159 201.18774 201.96539 202.74454 203.52519 204.30734 205.09099 205.87614 206.66279 207.45094 208.24059 209.03174 209.82439 210.61854 211.41419 212.21134 213.01000 213.81014 214.61179 215.41494 216.21959 217.02574 217.83339 218.64254 219.45319 220.26534 221.07899 221.89414 222.71079 223.52894 224.34859 225.16974 225.99239 226.81654 227.64219 228.46934 229.29799 230.12814 230.95979 231.79294 232.62759 233.46374 234.30139 235.14054 235.98119 236.82334 237.66699 238.51214 239.35879 240.20694 241.05659 241.90774 242.76039 243.61454 244.47019 245.32734 246.18599 247.04614 247.90779 248.77094 249.63559 250.50174 251.36939 252.23854 253.10919 253.98134 254.85499 255.72914 256.60479 257.48194 258.36059 259.24074 260.12239 261.00554 261.89019 262.77634 263.66399 264.55314 265.44379 266.33594 267.22959 268.12474 269.02139 269.91954 270.81919 271.72034 272.62299 273.52714 274.43279 275.33994 276.24859 277.15874 278.07039 278.98354 279.89819 280.81434 281.73199 282.65114 283.57179 284.49394 285.41759 286.34274 287.26939 288.19754 289.12719 290.05834 290.99099 291.92514 292.86079 293.79794 294.73659 295.67674 296.61839 297.56154 298.50619 299.45234 300.40000 301.34914 302.30079 303.25394 304.20859 305.16474 306.12239 307.08154 308.04219 309.00434 310.06799 311.03314 312.00079 312.96994 313.94159 314.91474 315.88939 316.86554 317.84319 318.82234 319.80299 320.78514 321.76879 322.75394 323.74059 324.72874 325.71839 326.70954 327.70219 328.69634 329.69199 330.68914 331.68779 332.68794 333.68959 334.69274 335.69739 336.70354 337.71119 338.72034 339.73099 340.74314 341.75679 342.77194 343.78859 344.80674 345.82639 346.84754 347.86919 348.89234 349.91699 350.94314 351.97079 352.99994 354.03059 355.06274 356.09639 357.13154 358.16819 359.20634 360.24599 361.28714 362.32979 363.37394 364.41959 365.46674 366.51539 367.56554 368.61719 369.67034 370.72499 371.78114 372.83879 373.89794 374.95859 376.02074 377.08439 378.14954 379.21619 380.28434 381.35399 382.42514 383.49779 384.57194 385.64759 386.72474 387.80339 388.88354 389.96519 391.04834 392.13299 393.21914 394.30679 395.39594 396.48659 397.57874 398.67239 399.76754 400.86419 401.96234 403.06199 404.16314 405.26579 406.36994 407.47559 408.58274 409.69139 410.80154 411.91319 413.02634 414.14099 415.25714 416.37479 417.49394 418.61459 419.73674 420.86039 421.98554 423.11219 424.24034 425.36999 426.50114 427.63379 428.76794 429.90359 431.04074 432.17939 433.31954 434.46119 435.60434 436.74899 437.89514 439.04279 440.19194 441.34259 442.49474 443.64839 444.80354 445.96019 447.11834 448.27799 449.43914 450.60179 451.76594 452.93159 454.09874 455.26739 456.43754 457.60919 458.78234 459.95749 461.13459 462.31274 463.49189 464.67204 465.85319 467.03534 468.21949 469.40564 470.59379 471.78394 472.97609 474.17019 475.36534 476.56149 477.75864 478.95679 480.15594 481.35609 482.55724 483.75939 484.96254 486.16669 487.37184 488.57799 489.78514 490.99329 492.20239 493.41254 494.62369 495.83579 497.04894 498.26309 499.47824 500.69439 501.91154 503.12969 504.34879 505.56894 506.79009 508.01224 509.23539 510.45954 511.68469 512.91079 514.13794 515.36609 516.59519 517.82534 519.05649 520.28859 521.52174 522.75589 523.99099 525.22714 526.46429 527.70239 528.94149 530.18164 531.42279 532.66494 533.90809 535.15219 536.39734 537.64349 538.89064 540.13879 541.38794 542.63809 543.88919 545.14134 546.39449 547.64864 548.90379 550.15994 551.41709 552.67519 553.93429 555.19439 556.45549 557.71759 558.98069 560.24479 561.50989 562.77599 564.04309 565.31119 566.58029 567.85039 569.12149 570.39359 571.66669 572.94079 574.21589 575.49199 576.76909 578.04719 579.32629 580.60639 581.88749 583.16959 584.45269 585.73679 587.02189 588.30799 589.59509 590.88319 592.17229 593.46239 594.75349 596.04559 597.33869 598.63279 599.92789 601.22399 602.52109 603.81919 605.11829 606.41839 607.71949 609.02159 610.32469 611.62879 612.93389 614.23999 615.54709 616.85519 618.16429 619.47439 620.78549 622.09759 623.41069 624.72479 626.03989 627.35599 628.67309 629.99119 631.31029 632.63039 633.95149 635.27359 636.59669 637.92079 639.24589 640.57199 641.89909 643.22719 644.55629 645.88639 647.21749 648.54959 649.88269 651.21679 652.55189 653.88799 655.22509 656.56319 657.90229 659.24239 660.58349 661.92559 663.26869 664.61279 665.95789 667.30399 668.65109 669.99919 671.34829 672.69839 674.04949 675.40159 676.75469 678.10879 679.46389 680.81999 682.17709 683.53519 684.89429 686.25439 687.61549 688.97759 690.34069 691.70479 693.06989 694.43599 695.80309 697.17119 698.54029 699.91039 701.28149 702.65359 704.02669 705.40079 706.77589 708.15199 709.52909 710.90719 712.28629 713.66639 715.04749 716.42959 717.81269 719.19679 720.58189 721.96799 723.35509 724.74319 726.13229 727.52239 728.91349 730.30559 731.69869 733.09279 734.48789 735.88399 737.28109 738.67919 740.07829 741.47839 742.87949 744.28159 745.68469 747.08879 748.49389 749.89999 751.30709 752.71519 754.12429 755.53439 756.94549 758.35759 759.77069 761.18479 762.59989 764.01599 765.43309 766.85119 768.27029 769.69039 771.11149 772.53359 773.95669 775.38079 776.80589 778.23199 779.65909 781.08719 782.51629 783.94639 785.37749 786.80959 788.24269 789.67679 791.11189 792.54799 793.98509 795.42319 796.86229 798.30239 799.74349 801.18559 802.62869 804.07279 805.51789 806.96399 808.41109 809.85919 811.30829 812.75839 814.20949 815.66159 817.11469 818.56879 819.92389 821.37999 822.83709 824.29519 825.75429 827.21439 828.67549 830.13759 831.60069 833.06479 834.52989 835.99599 837.46309 838.93119 840.39929 841.86839 843.33849 844.80959 846.28169 847.75479 849.22889 850.70399 852.17909 853.65519 855.13129 856.60839 858.08649 859.56559 861.04569 862.52679 864.00889 865.49199 866.97609 868.46119 869.94729 871.43439 872.92249 874.41159 875.90169 877.39279 878.88489 880.37799 881.87209 883.36719 884.86329 886.36039 887.85849 889.35759 890.85769 892.35879 893.86089 895.36399 896.86809 898.37319 899.87929 901.38639 902.89449 904.40359 905.91369 907.42479 908.93689 910.44999 911.96409 913.47919 914.99529 916.51239 918.03049 919.54959 921.06969 922.59079 924.11289 925.63599 927.15909 928.68319 930.20829 931.73439 933.26149 934.78959 936.31869 937.84879 939.37989 940.91199 942.44509 943.97919 945.51429 947.05039 948.58749 950.12559 951.66469 953.20479 954.74589 956.28799 957.83109 959.37519 960.92029 962.46639 964.01349 965.56159 967.11069 968.66079 970.21189 971.76399 973.31709 974.87219 976.42829 977.98539 979.54349 981.10259 982.66269 984.22379 985.78589 987.34899 988.91309 990.47819 992.04429 993.61139 995.17949 996.74859 998.31869 999.88979 1000.46189 1001.93499 1003.40909 1004.88419 1006.36029 1007.83739 1009.31549 1010.79459 1012.27469 1013.75579 1015.23689 1016.71899 1018.20209 1019.68619 1021.17129 1022.65739 1024.14449 1025.63259 1027.12169 1028.61179 1030.10289 1031.59499 1033.08809 1034.58219 1036.07729 1037.57339 1039.07049 1040.56859 1042.06769 1043.56779 1045.06889 1046.57099 1048.07409 1049.57819 1051.08329 1052.58939 1054.09649 1055.60459 1057.11369 1058.62379 1060.13489 1061.64699 1063.15909 1064.67219 1066.18629 1067.70139 1069.21749 1070.73459 1072.25269 1073.77179 1075.29189 1076.81299 1078.33509 1079.85819 1081.38229 1082.90739 1084.43349 1085.96059 1087.48869 1089.01779 1090.54789 1092.07899 1093.61009 1095.14219 1096.67529 1098.20939 1099.74449 1101.28059 1102.81769 1104.35579 1105.89489 1107.43499 1108.97609 1110.51819 1112.06229 1113.60739 1115.15349 1116.70059 1118.24869 1119.79779 1121.34789 1122.89899 1124.45009 
```

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

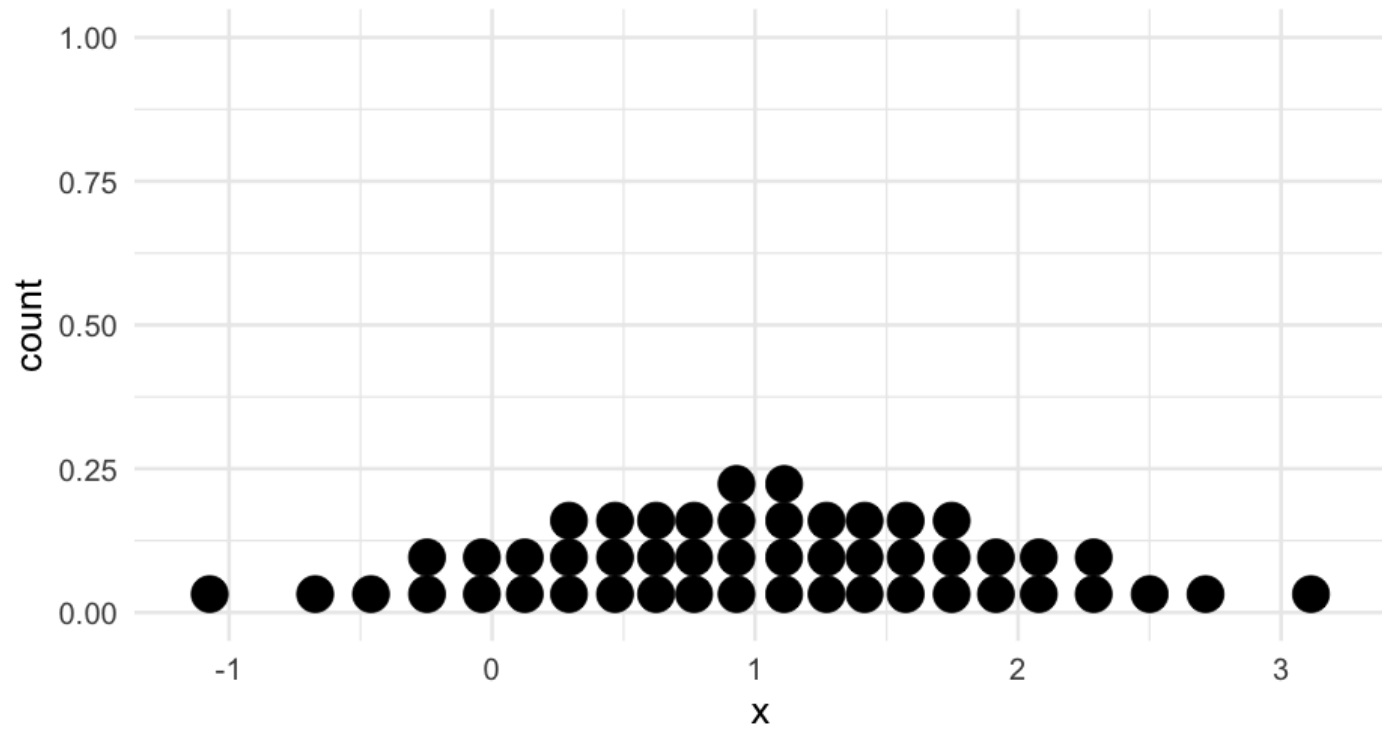
```
head(discretized)
```

```
##           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
```

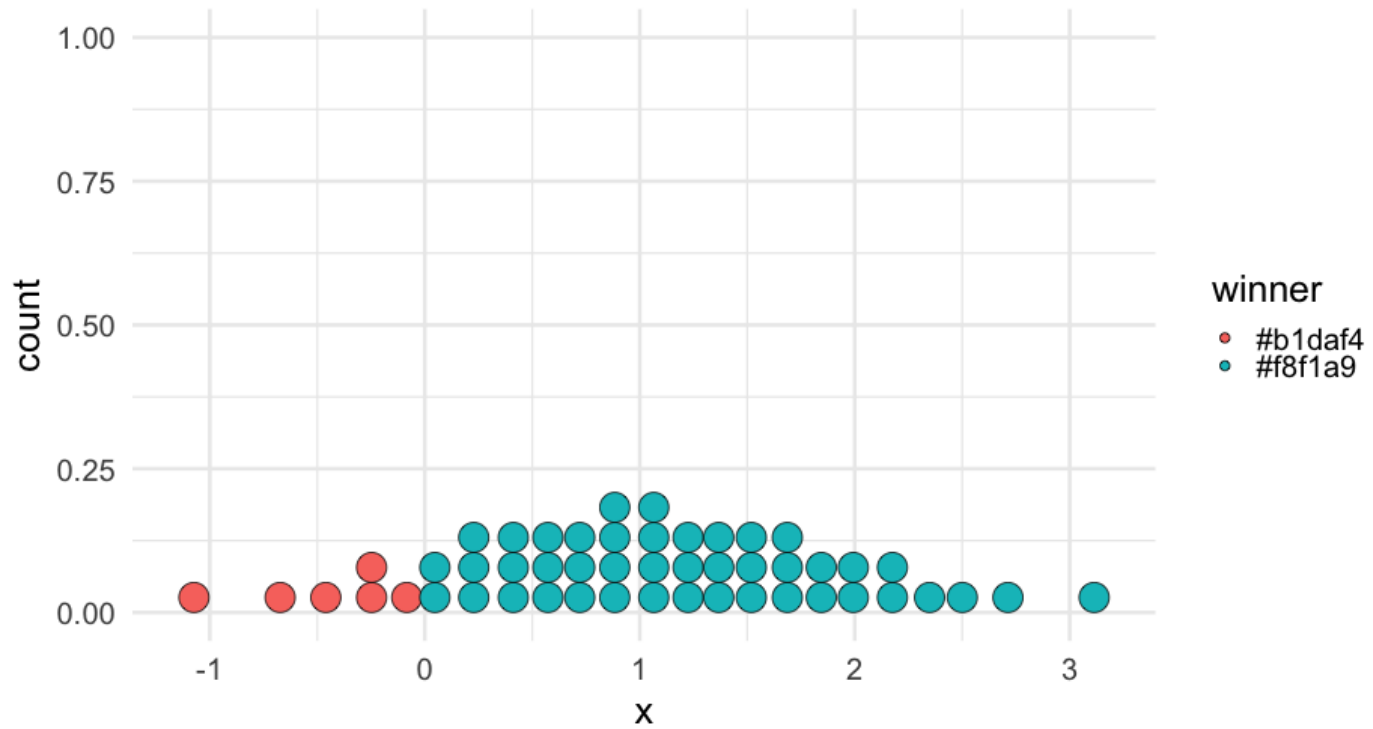
```
tail(discretized)
```

```
##           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
```

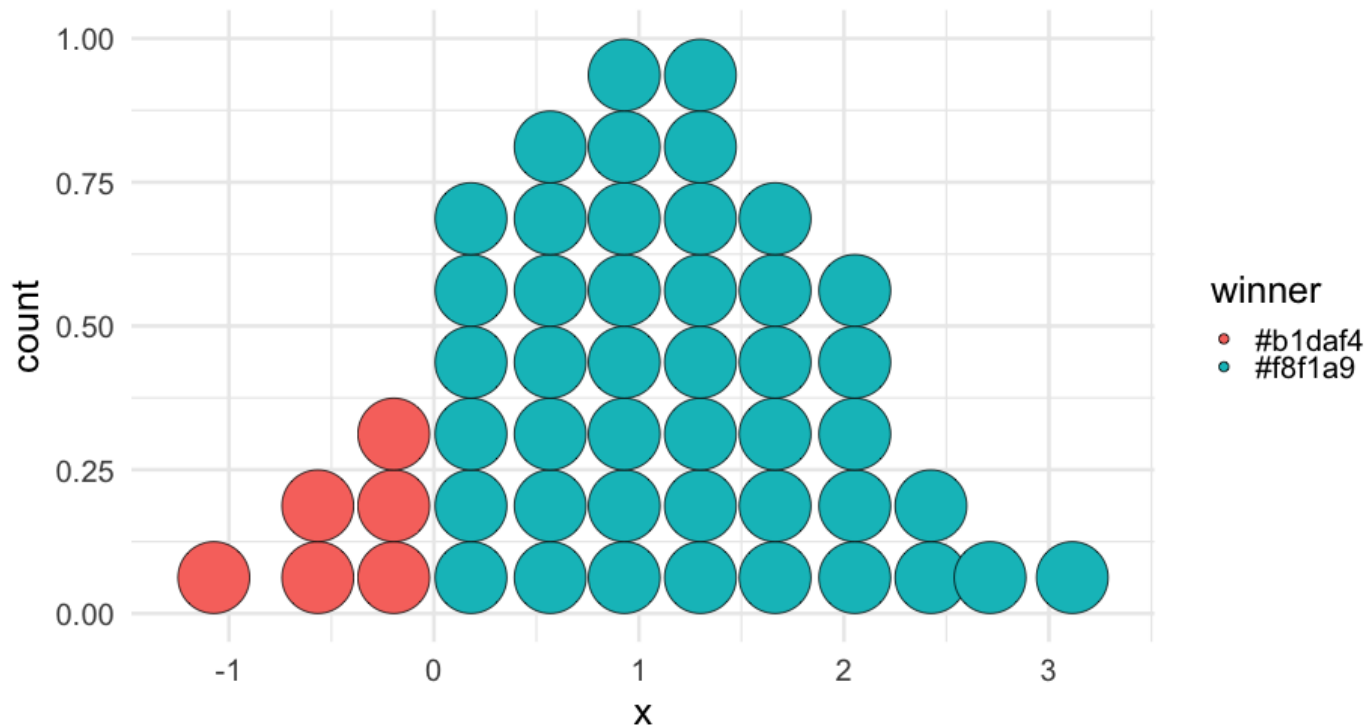
```
ggplot(discretized, aes(x)) +  
  geom_dotplot()
```



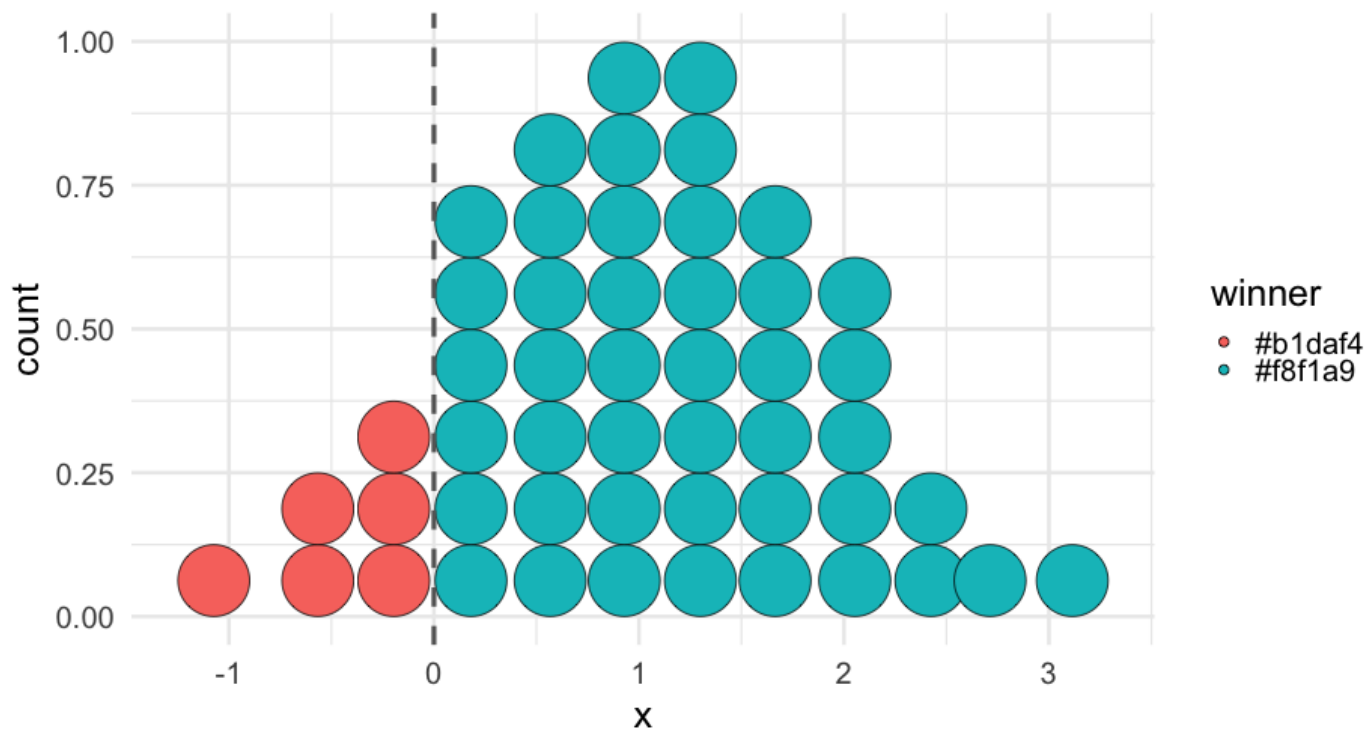
```
ggplot(discretized, aes(x)) +  
  geom_dotplot(aes(fill = winner))
```



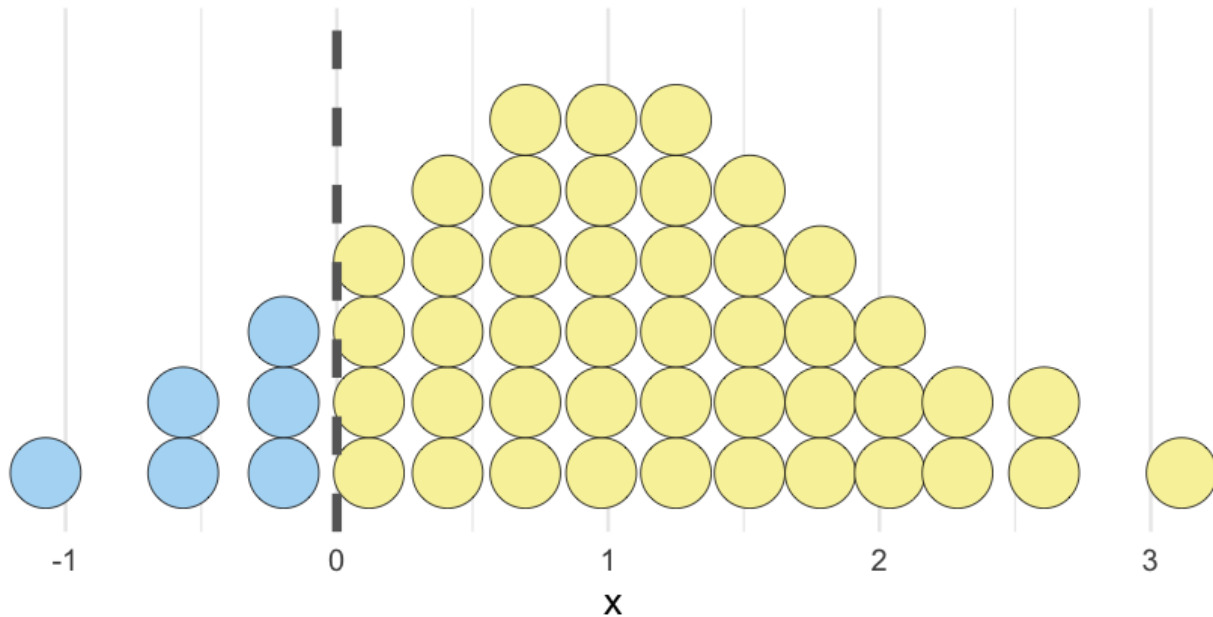
```
ggplot(discretized, aes(x)) +  
  geom_dotplot(aes(fill = winner), binwidth = 0.35)
```




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

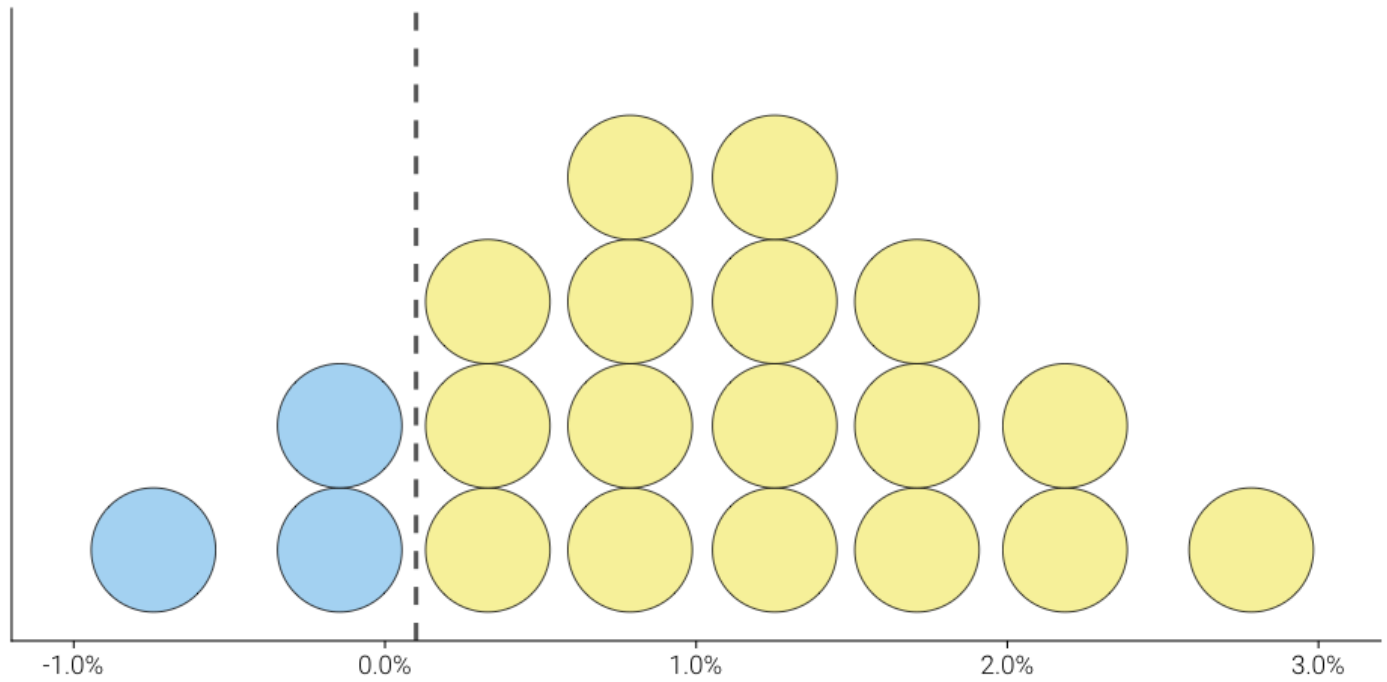


```
ggplot(discretized, aes(x)) +
  geom_dotplot(aes(fill = winner), binwidth = 0.26) +
  geom_vline(xintercept = 0, color = "gray40", linetype = 2, size = 1) +
  scale_fill_identity(guide = "none") +
  scale_y_continuous(name = "",
                     breaks = NULL)
```



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) +  
  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] 94.4 97.7 115.6 100.7 101.3 117.2 104.6 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 80.3 107.0 95.3
```

```
mean(samp10b)
```

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


Do it a bunch of times

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

```
samples
```

```
## [[1]]  
## [1]  89.3  97.8  89.7  92.7  93.7  83.1 108.4 101.5  88.6 112.5  
##  
## [[2]]  
## [1] 104.3  97.0 109.0 108.8 108.2 106.9 105.5  99.4  96.9  96.2  
##  
## [[3]]  
## [1]  93.1  97.9  87.3 121.7 112.1  88.8  96.0  95.3 107.8  99.2  
##  
## [[4]]  
## [1] 102.5  99.7  99.6 113.7  97.7 115.2  84.5 105.8 101.2 102.2  
##  
## [[5]]  
## [1] 103.8  95.0  96.7  89.8  89.3 103.0 104.5 100.5 109.2 120.5  
##  
## [[6]]  
## [1]  95.1  76.9 110.1  92.9  93.1 110.3  97.2  87.8 101.8  98.6  
##  
## [[7]]  
## [1] 100.1 103.9  96.3 106.4  97.8 103.3 111.0 104.4  96.7 111.5  
##  
## [[8]]
```

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
```

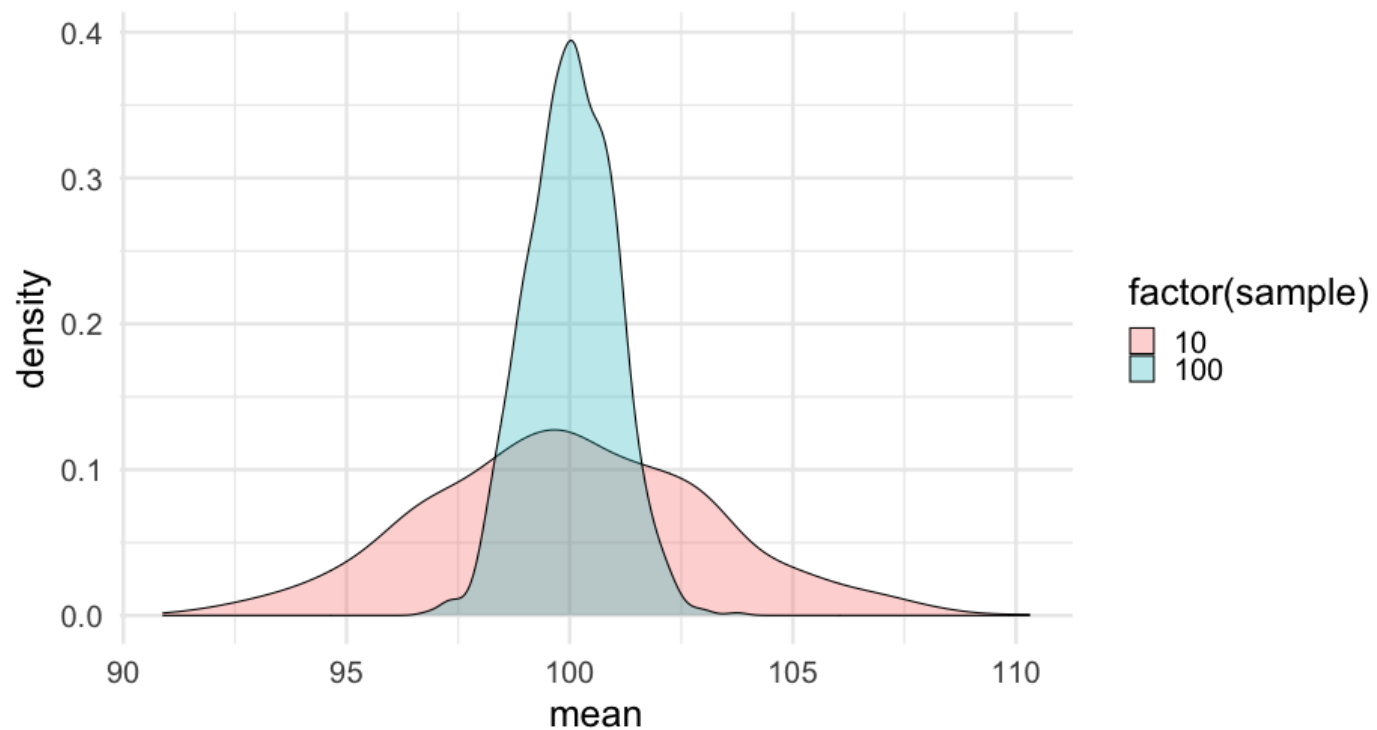
Visualize the sampling distributions

```
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 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
## # ... 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 Pr(>|t|)
## (Intercept)    28.777      1.473   19.54 < 2e-16 ***
## displ         -2.172      0.175  -12.43 < 2e-16 ***
## classcompact   -3.599      1.252   -2.87  0.0044 **
## classmidsize   -3.676      1.206   -3.05  0.0026 **
## classminivan   -5.595      1.306   -4.28  2.7e-05 ***
## classpickup    -6.182      1.121   -5.51  9.6e-08 ***
## classsubcompact -2.629      1.237   -2.13  0.0346 *
## classsuvs      -5.599      1.087   -5.15  5.7e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.25 on 226 degrees of freedom
## Multiple R-squared:  0.729,    Adjusted R-squared:  0.721
```

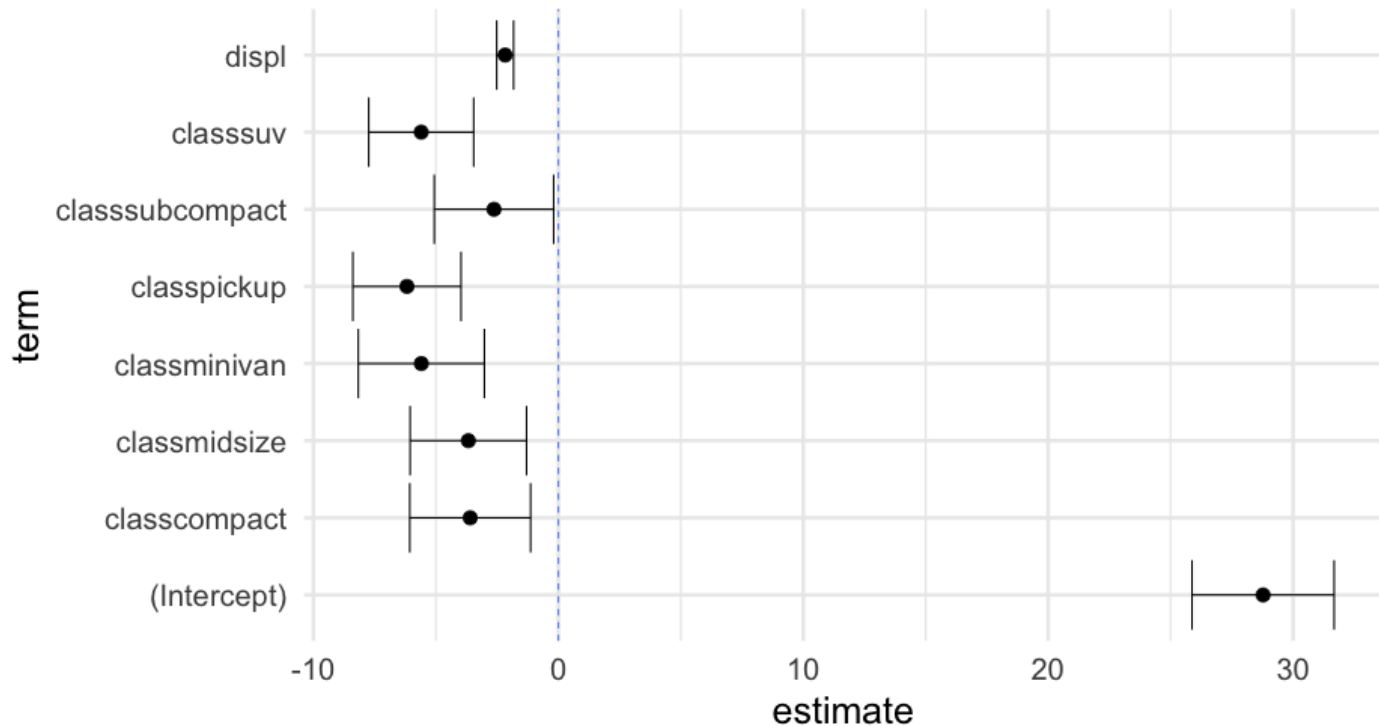
Visualize with standard errors

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

```
tidied_m
```

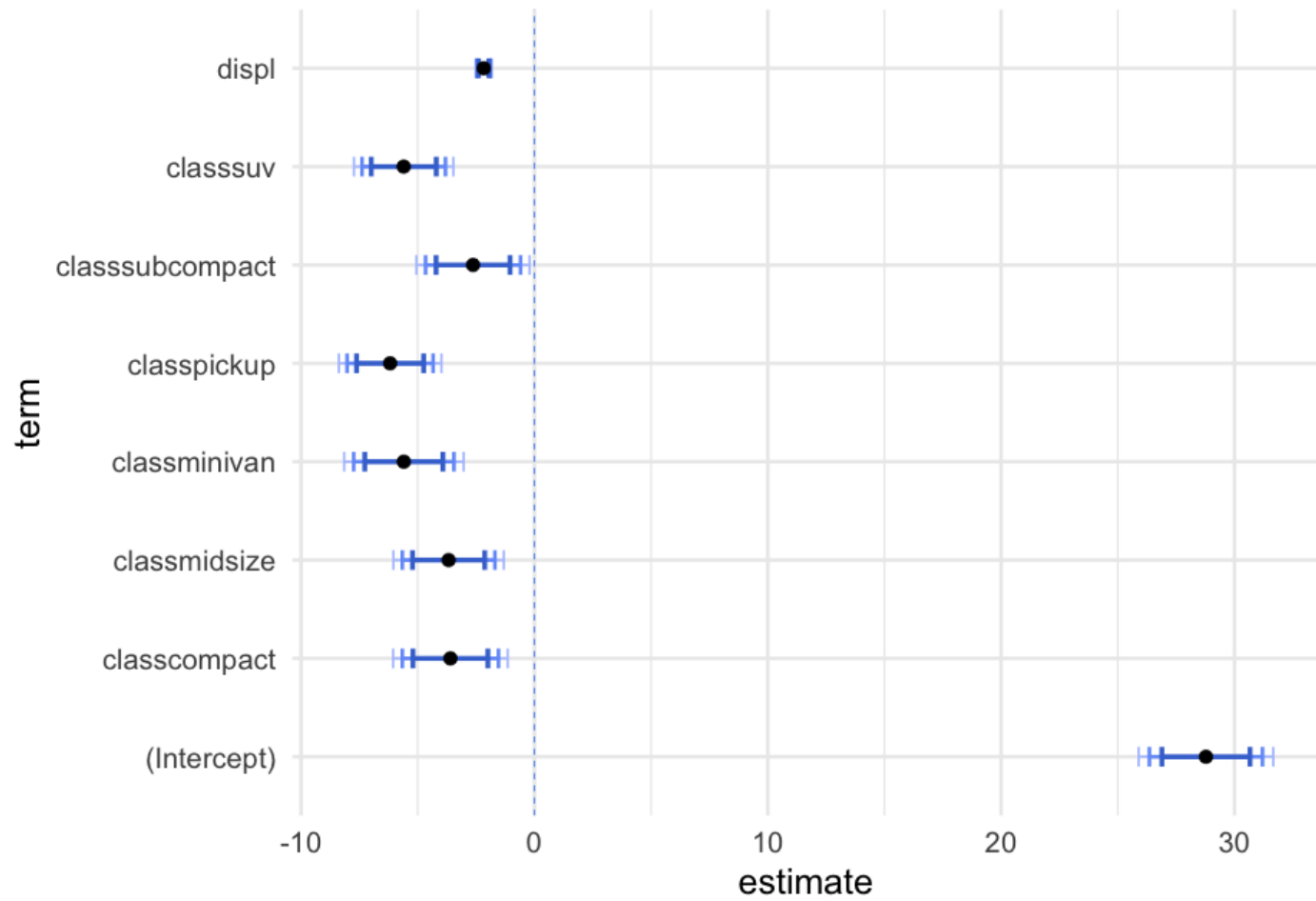
```
## # A tibble: 8 x 7
##   term          estimate std.error statistic    p.value  conf.low conf.high
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)  28.77682  1.472892  19.53763 1.905873e-50 25.87446 31.67919
## 2 displ       -2.171562  0.1746638 -12.43281 2.197130e-27 -2.515740 -1.827384
## 3 classcompact -3.599125  1.252190   -2.874265 4.436052e- 3 -6.066585 -1.131665
## 4 classmidsize -3.675526  1.206253   -3.047061 2.585762e- 3 -6.052466 -1.298586
## 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 -3.973035
## # ... with 2 more rows
```

```
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

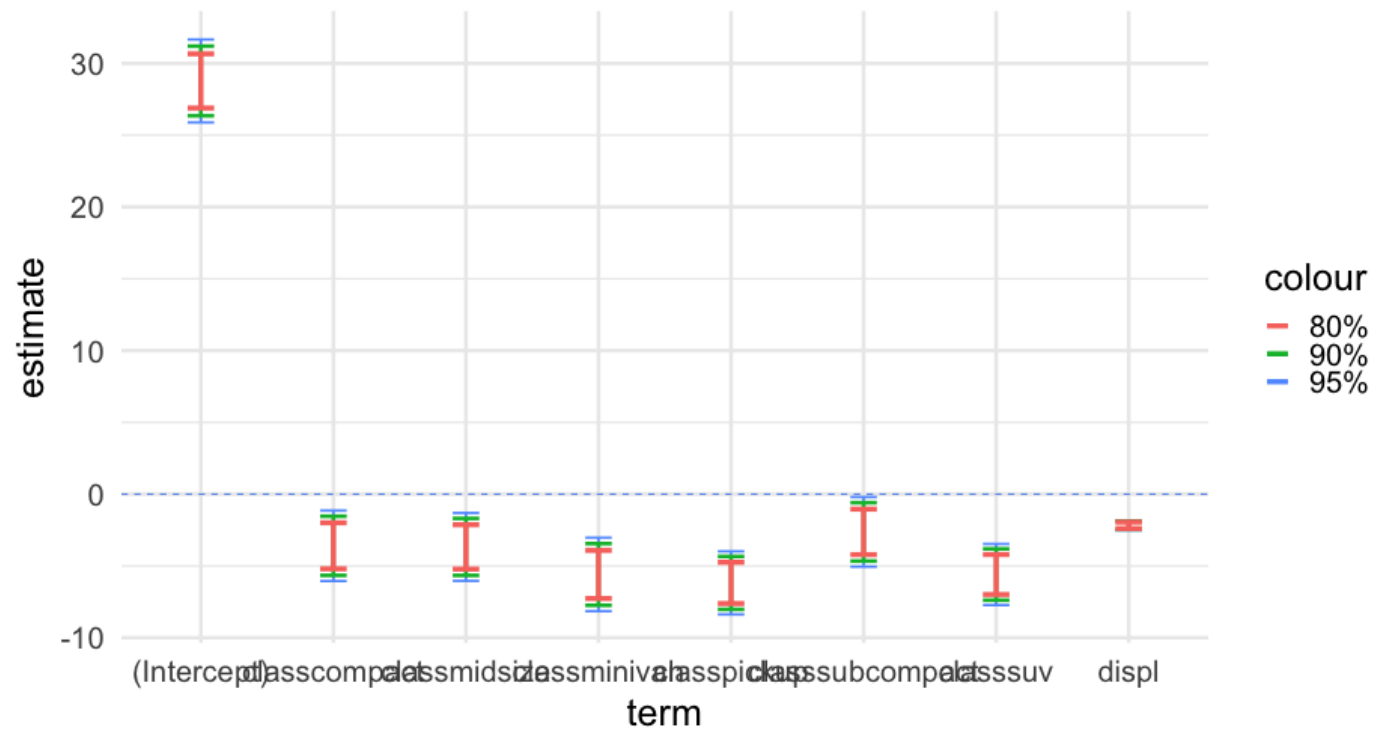
```
library(colorspace)
ggplot(tidied_m, aes(term, estimate)) +
  geom_hline(yintercept = 0,
             color = "cornflowerblue",
             linetype = 2) +
  geom_errorbar(aes(ymin = estimate + qnorm(.025)*std.error,
                    ymax = estimate + qnorm(.975)*std.error),
               color = 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()
```



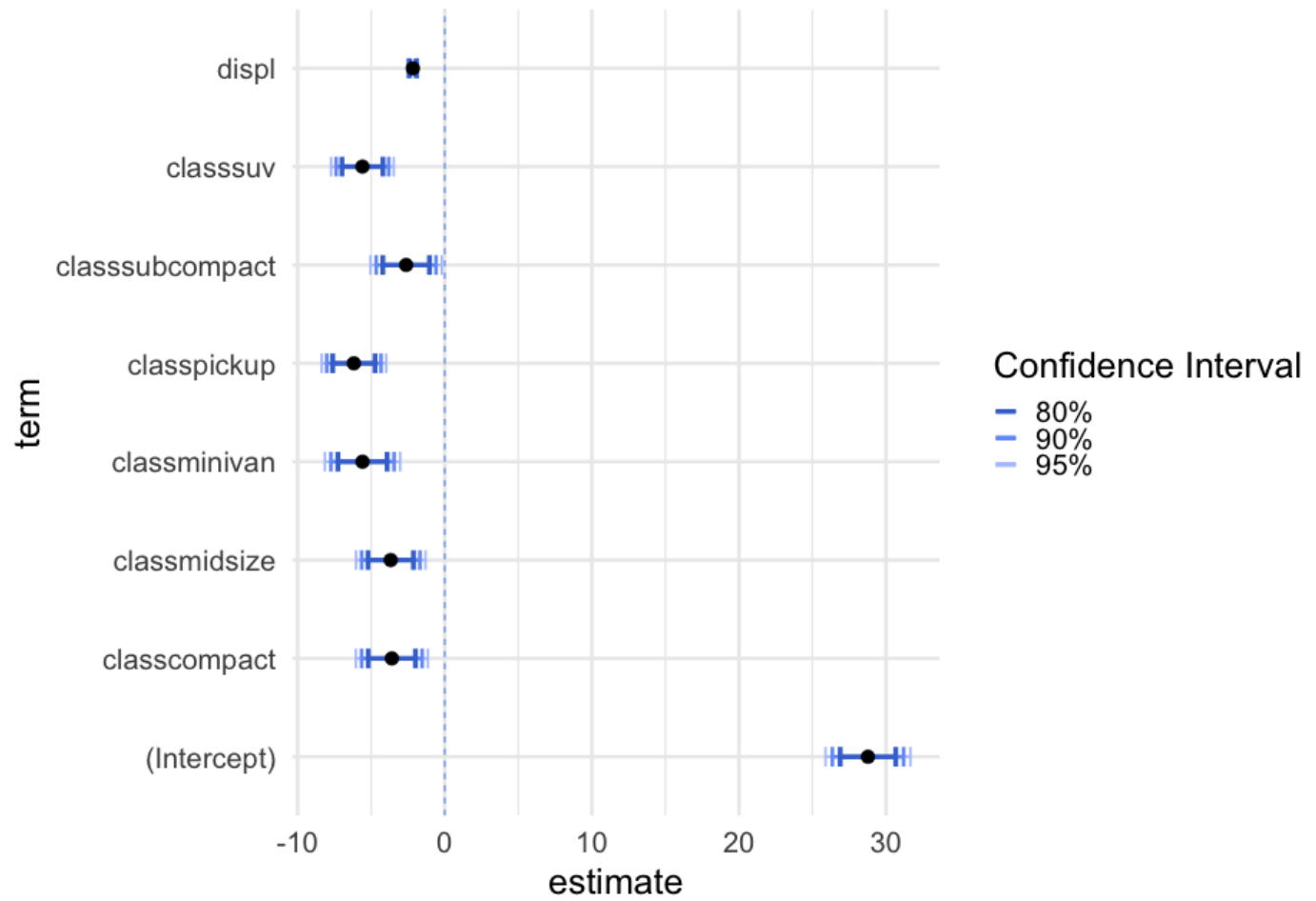
Add levels to legend

```
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

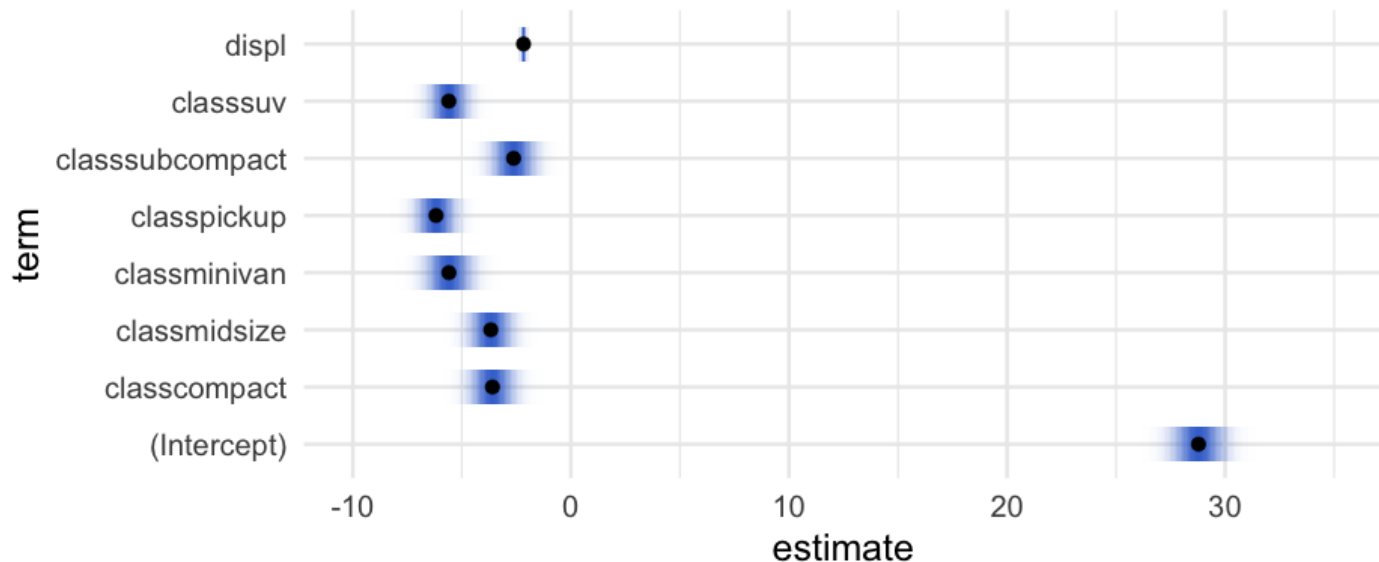


```
p +  
  scale_color_manual("Confidence Interval",  
                     values = c("#4375D3",  
                                lighten("#4375D3", .3),  
                                lighten("#4375D3", .6))) +  
  geom_point() +  
  coord_flip()
```



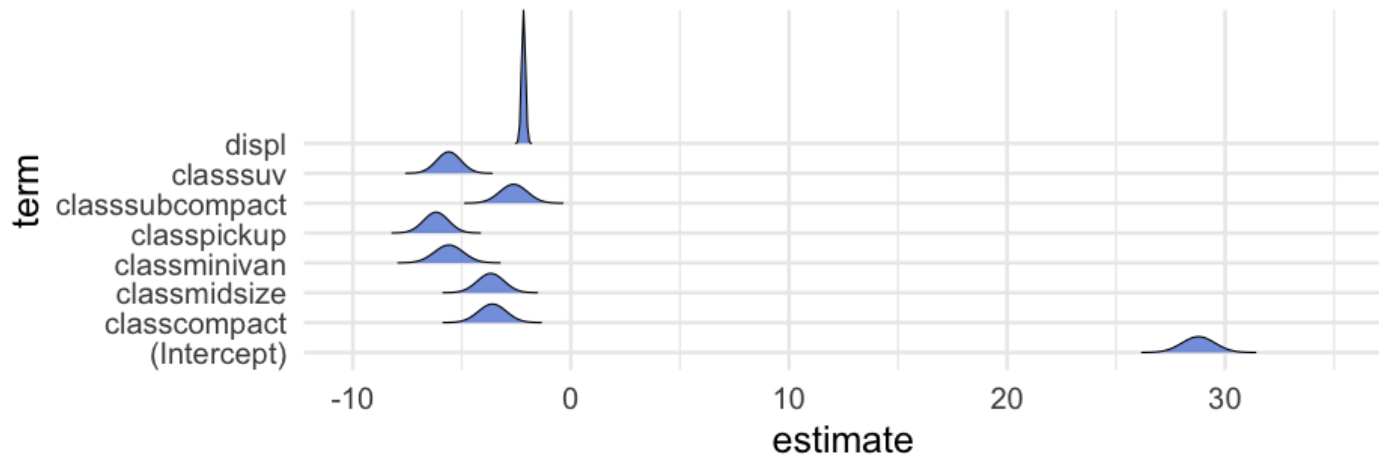
Density stripes

```
#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)
```

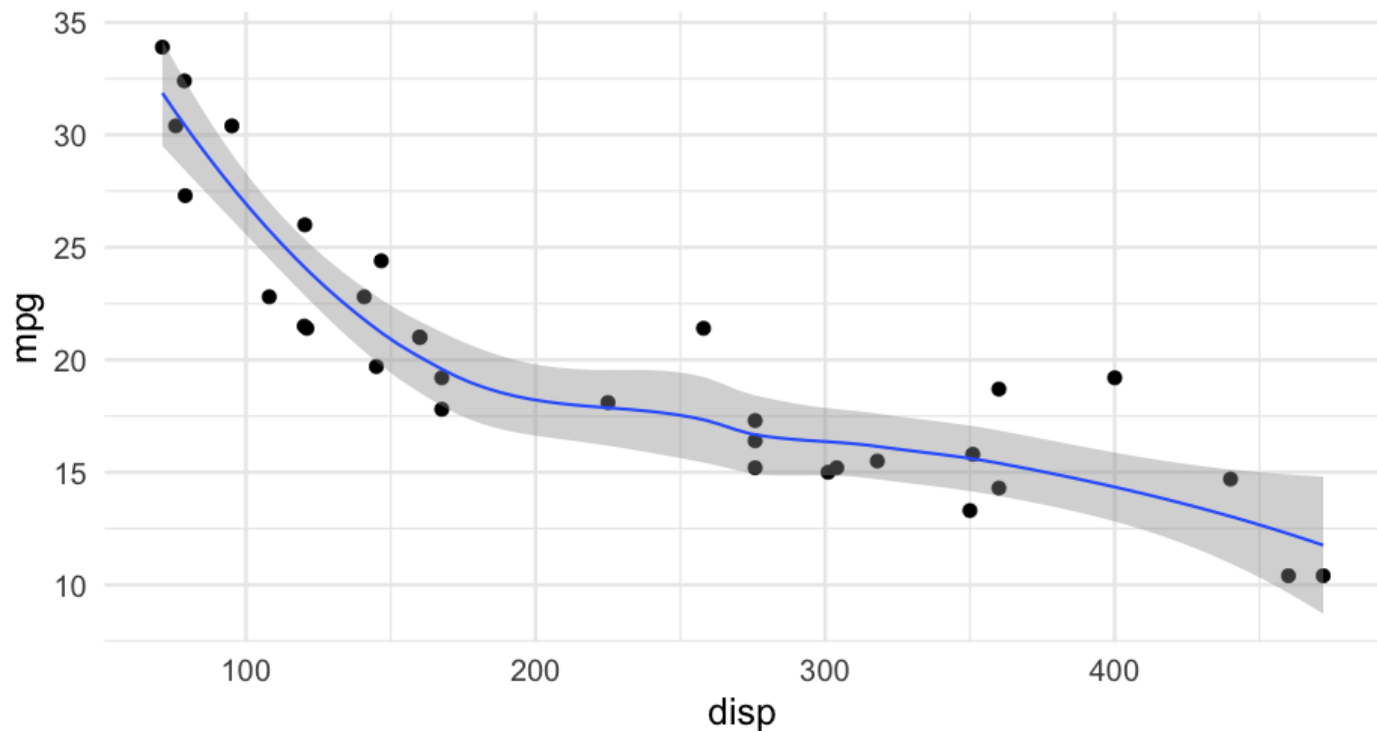


HOPs

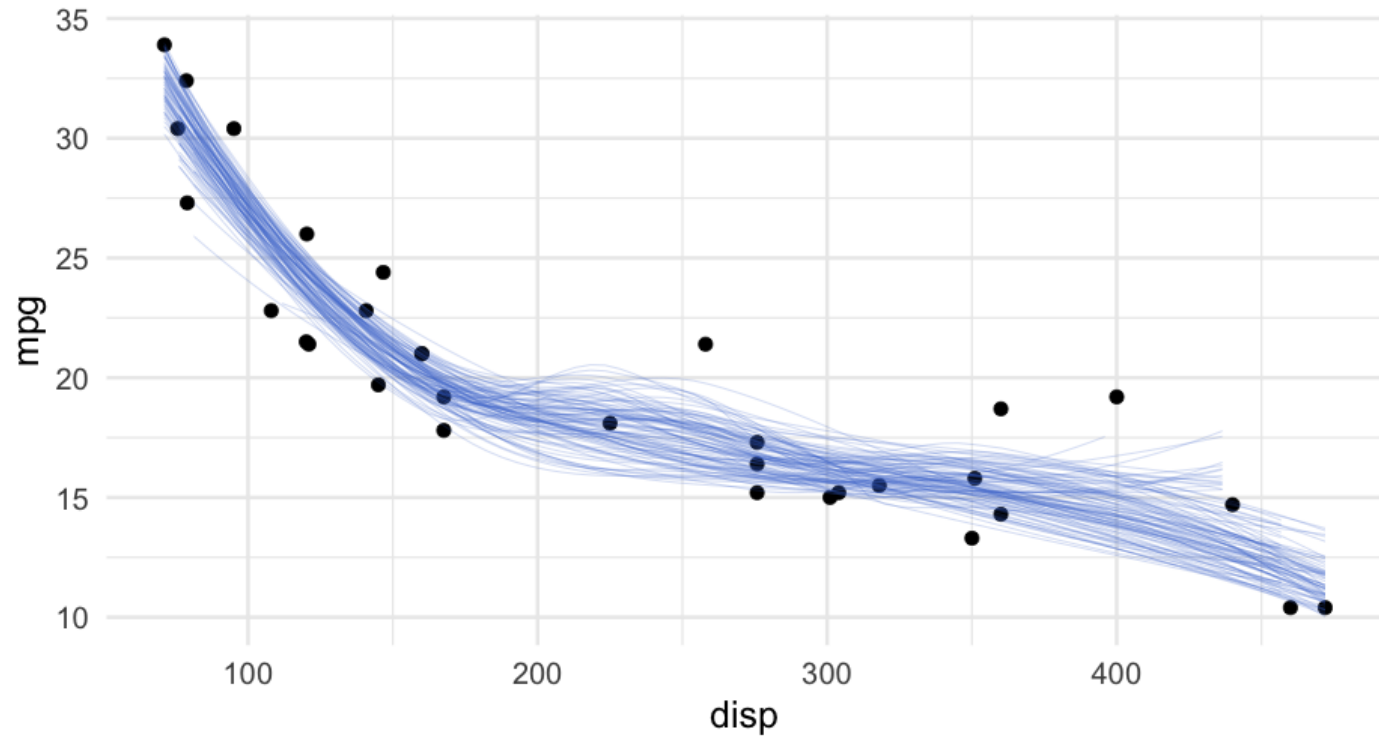
Hypothetical Outcome Plots (and related plots)

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)
```

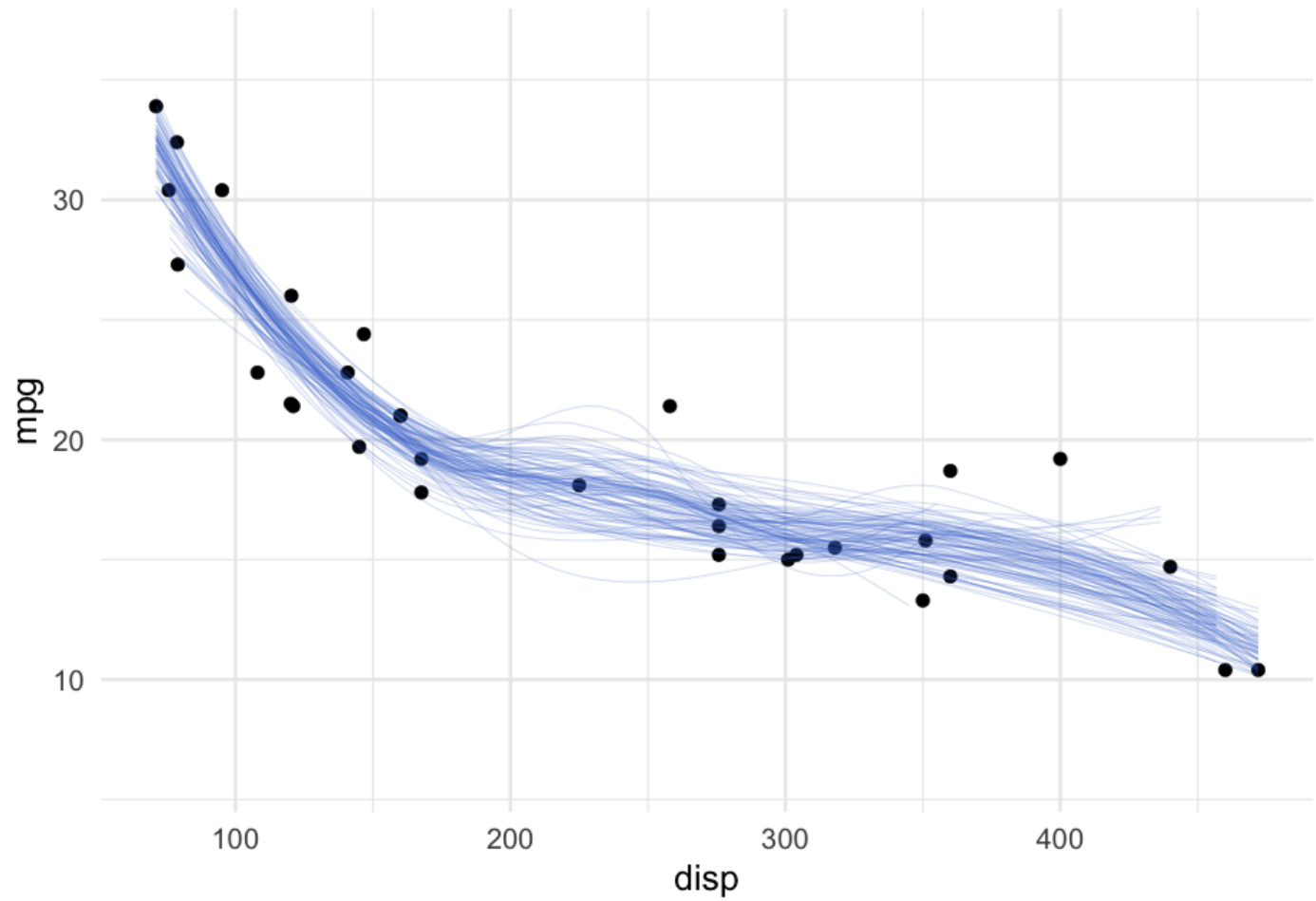
```
##           sample  mpg  cyl  disp  hp  drat    wt  qsec  vs  am  gear  car
## Maserati Bora...1      1 15.0   8   301 335  3.54  3.57 14.6   0   1     5
## Valiant...2          1 18.1   6   225 105  2.76  3.46 20.2   1   0     3
## Volvo 142E...3        1 21.4   4   121 109  4.11  2.78 18.6   1   1     4
## Merc 450SE...4        1 16.4   8   276 180  3.07  4.07 17.4   0   0     3
## Mazda RX4...5         1 21.0   6   160 110  3.90  2.62 16.5   0   1     4
## Merc 450SLC...6       1 15.2   8   276 180  3.07  3.78 18.0   0   0     3
```

```
tail(d_samps)
```

```
##           sample  mpg  cyl  disp  hp  drat    wt  qsec  vs
## Lincoln Continental.1...3195    100 10.4   8  460.0 215  3.00  5.42 17.8
## Pontiac Firebird...3196         100 19.2   8  400.0 175  3.08  3.85 17.1
## Fiat X1-9.1...3197              100 27.3   4   79.0  66  4.08  1.94 18.9
## Mazda RX4 Wag.3...3198           100 21.0   6  160.0 110  3.90  2.88 17.0
## Hornet Sportabout...3199         100 18.7   8  360.0 175  3.15  3.44 17.0
## Lotus Europa...3200              100 30.4   4   95.1 113  3.77  1.51 16.9
```

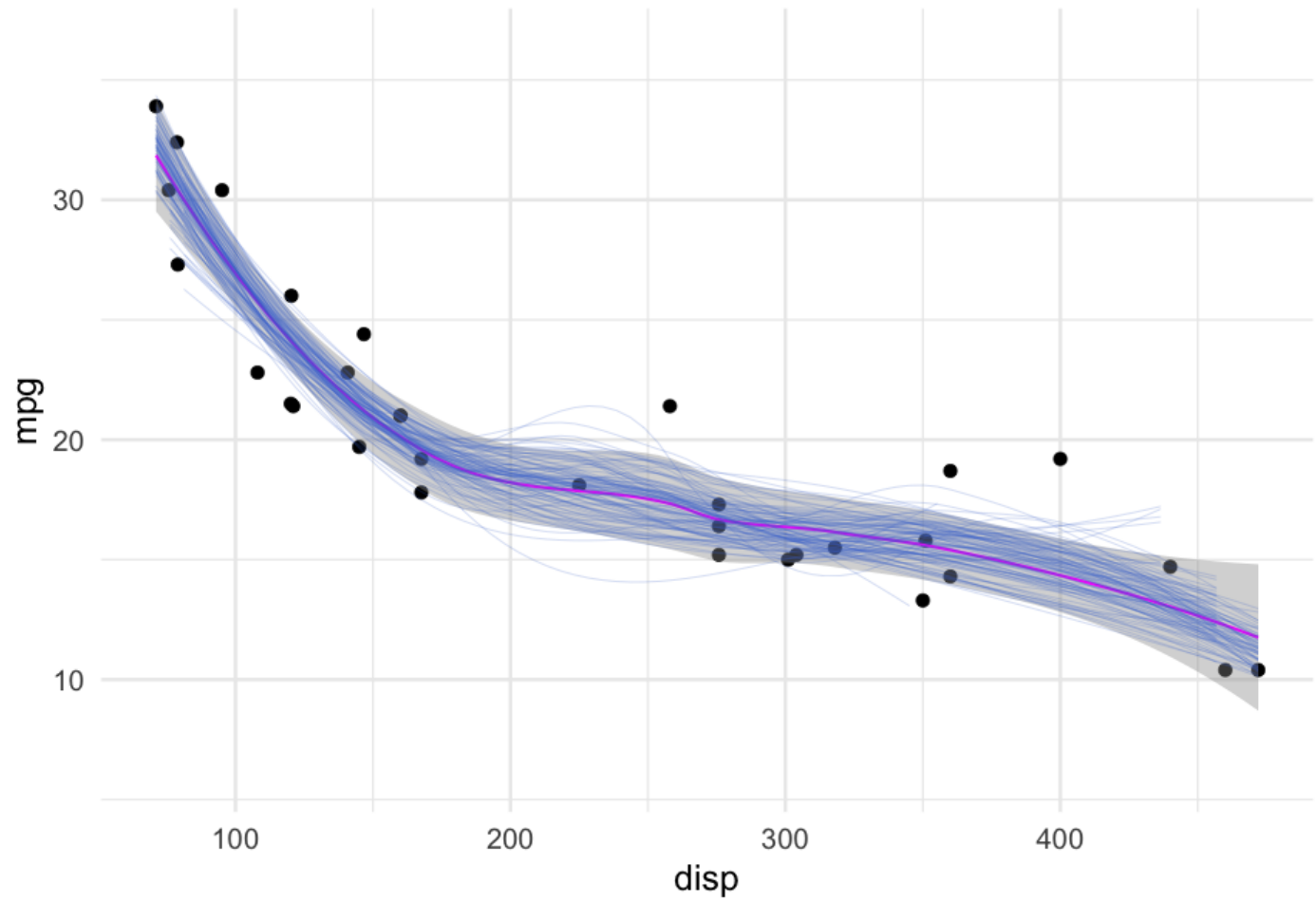
Plot both data sources

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



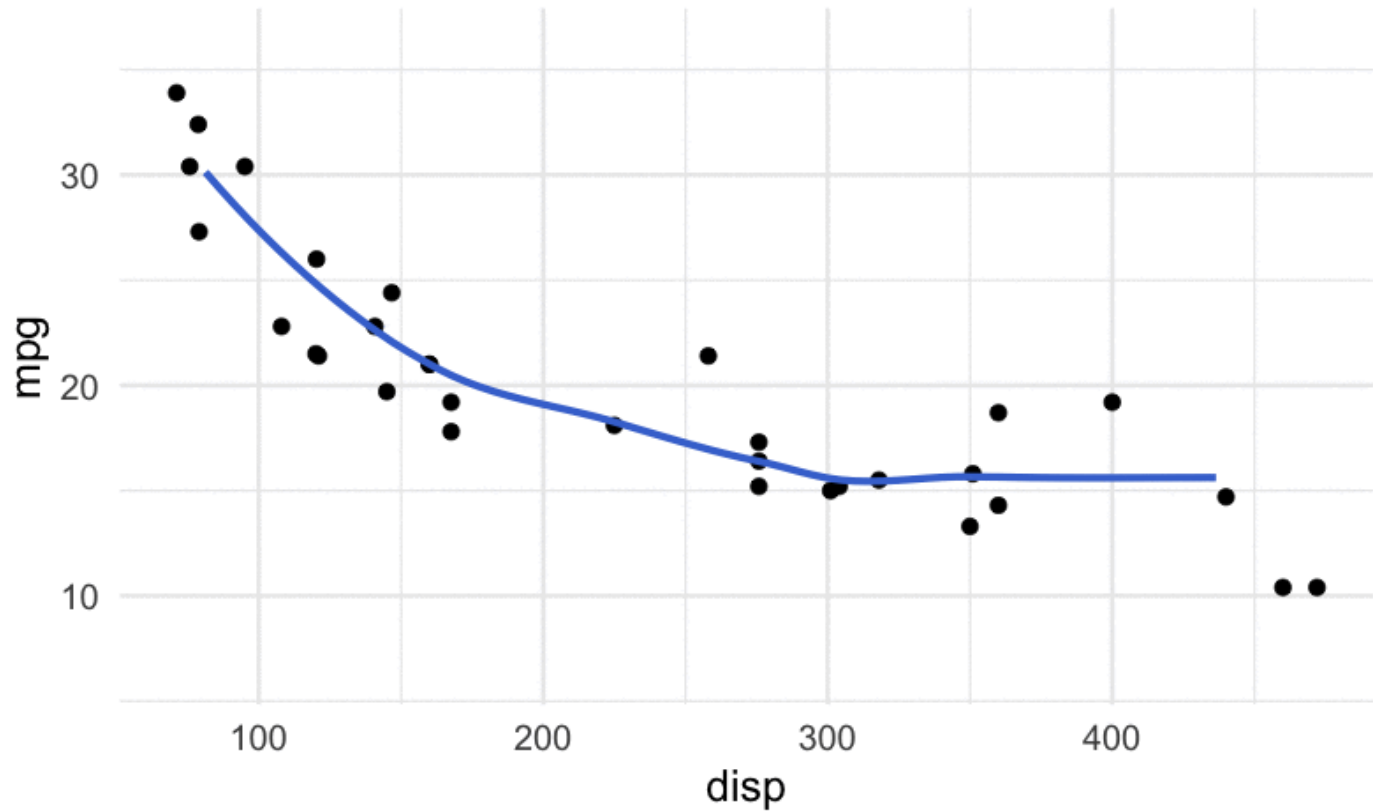
Note, they match up

```
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"

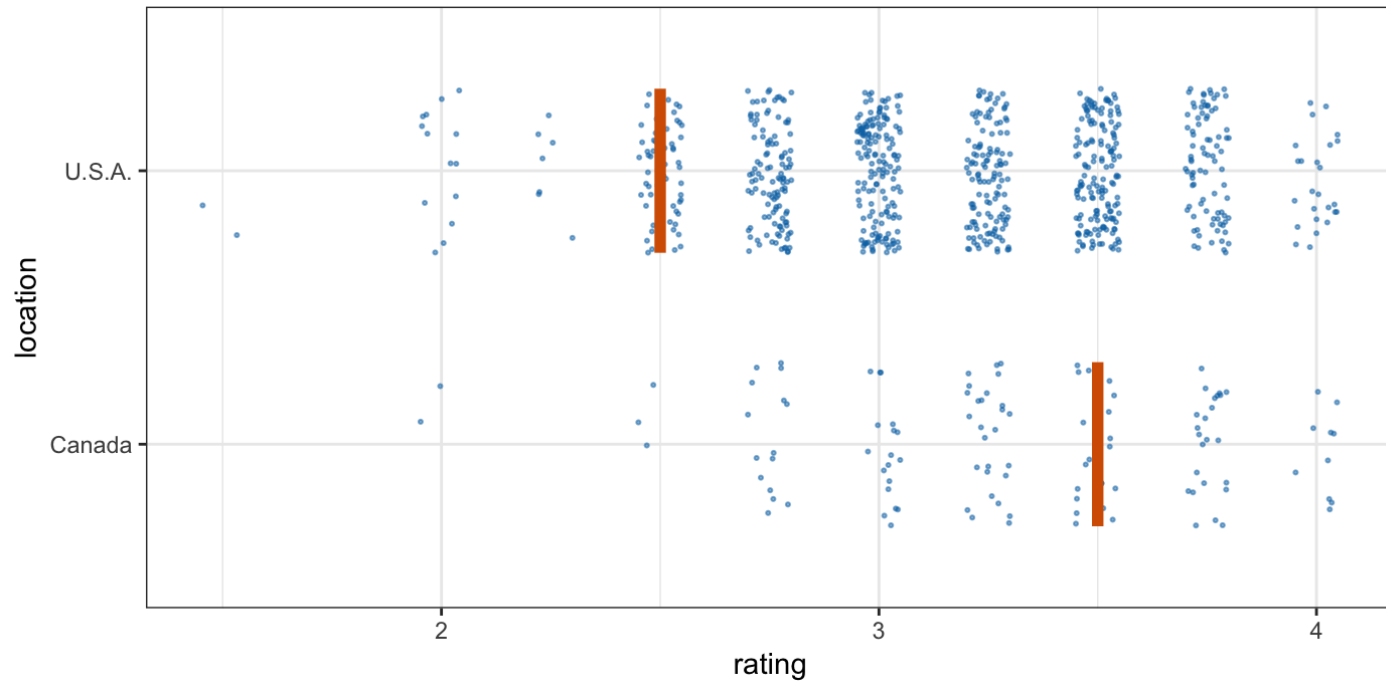


How?

gganimate::transition_states

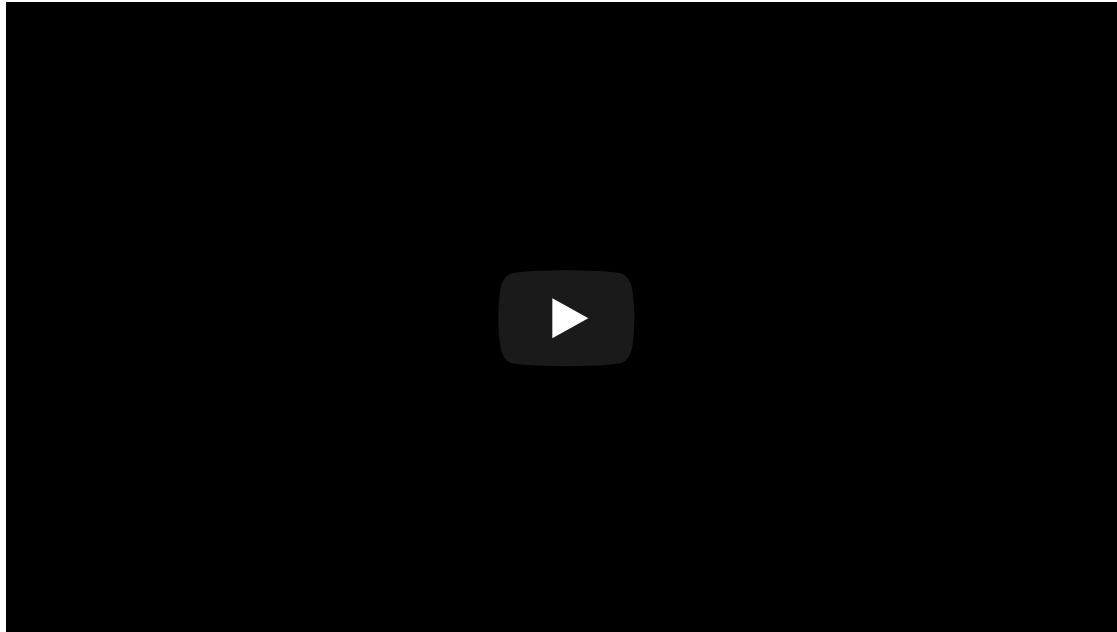
```
library(gganimate)
ggplot(mtcars, aes(displacement, mpg)) +
  geom_point() +
  stat_smooth(data = d_samps,
              geom = "line",
              size = 2,
              color = "#4375D3",
              fullrange = TRUE) +
  transition_states(sample,
                    transition_length = 0.5,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

Another example



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

Guest Lecturer: Akhila Nekkanti