Engaging and Effective ggplot (2)

VCS, Rice 2025

Kieran Healy
Duke University

March 2025

Polishing your plots and Presenting them

Load our packages

```
library(here)       # manage file paths
library(tidyverse)  # your friend and mine
library(socviz)     # data and some useful functions
library(ggrepel)    # Text and labels
library(colorspace) # luminance-balanced palettes
library(scales)      # scale adjustments and enhancements
library(ggforce)    # useful enhancements to ggplot
```

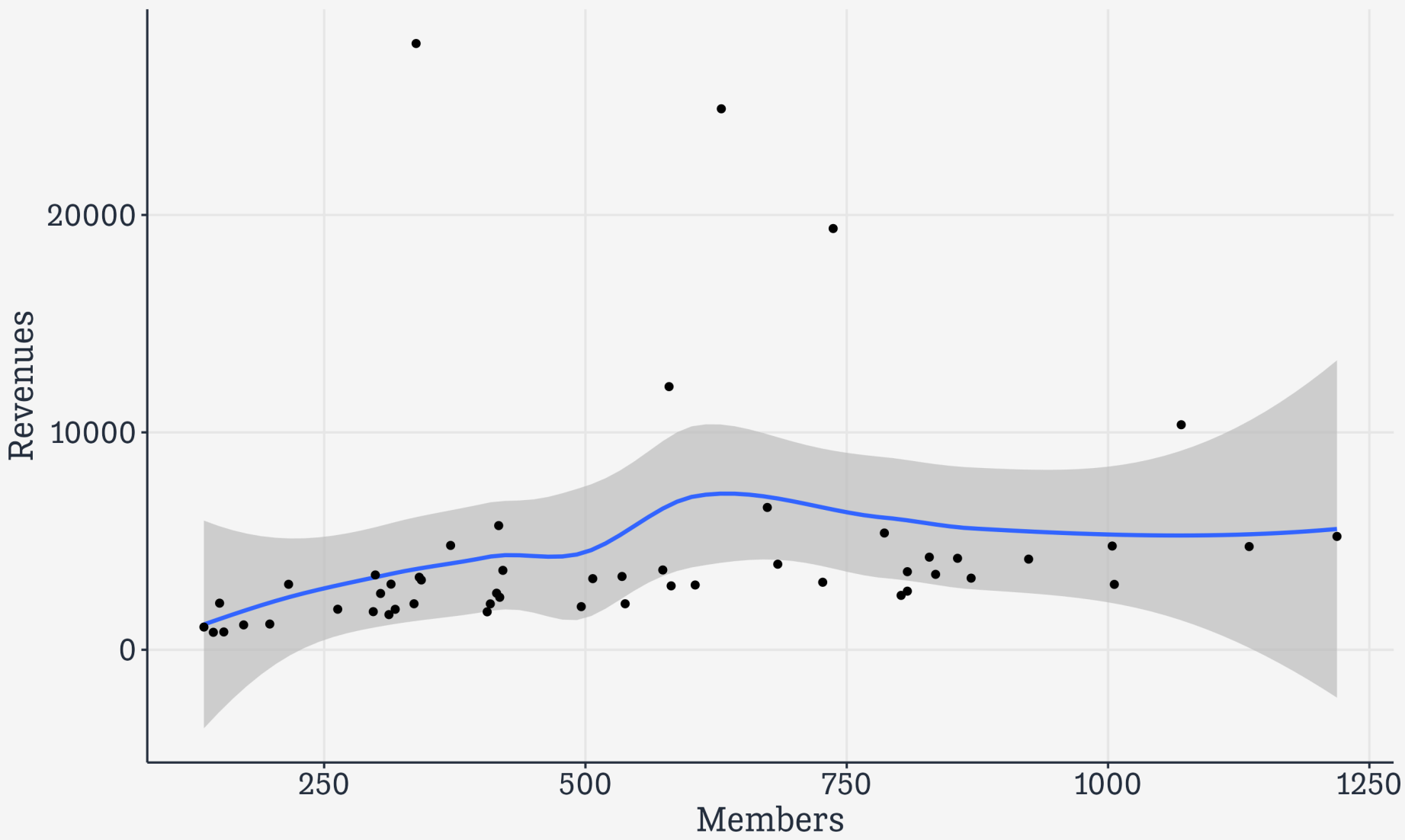
Piece by piece,
Layer by layer

Build your plots a piece at a time

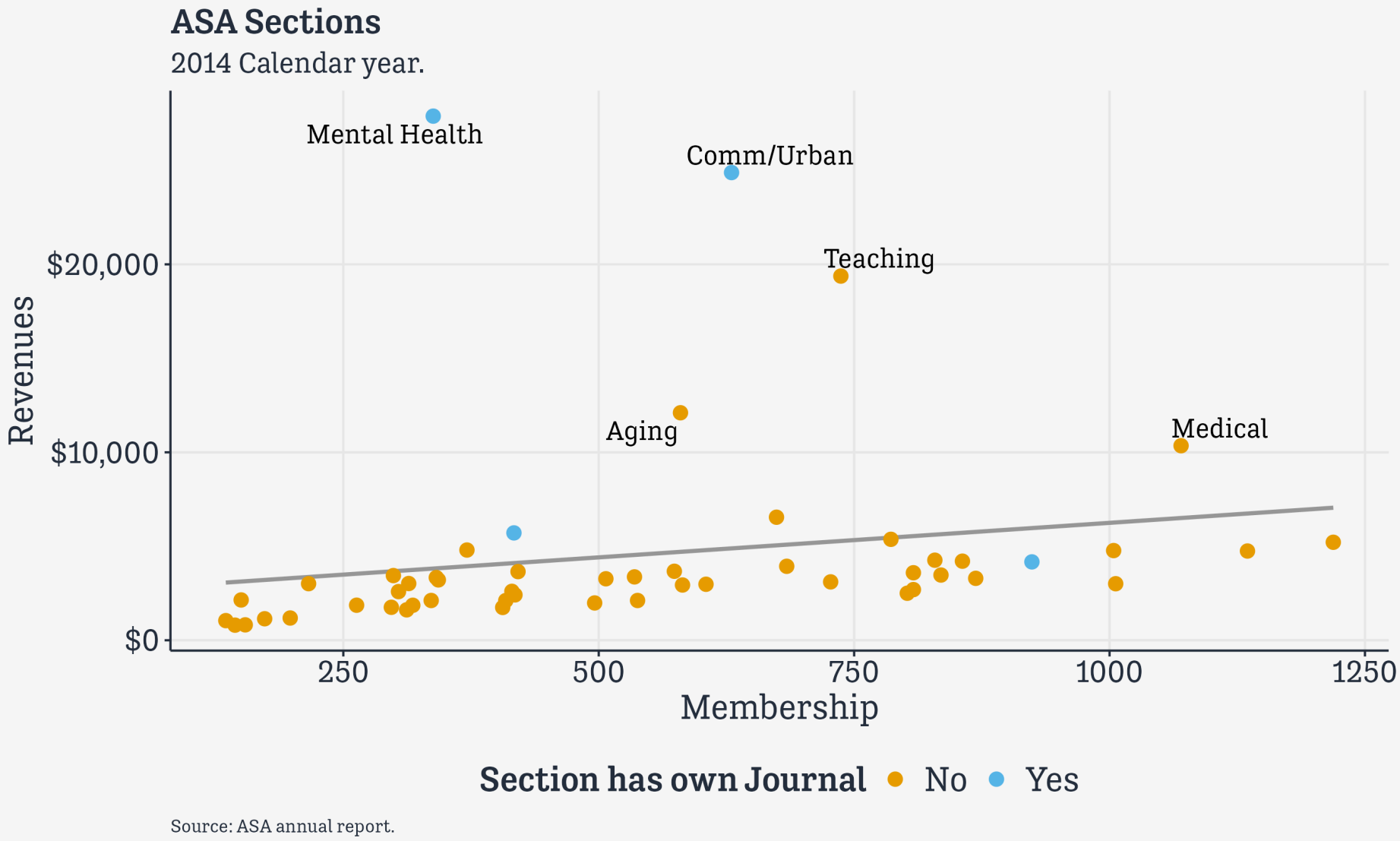
```
asasec ← as_tibble(asasec)
asasec
```

```
# A tibble: 572 × 9
   Section        Sname Beginning Revenues Expenses Ending Journal  Year Members
   <fct>          <fct>     <int>    <int>    <int>  <int> <fct>   <int>   <int>
 1 Aging and the… Aging     12752    12104    12007  12849 No       2005     598
 2 Alcohol, Drug… Alco…     11933     1144      400  12677 No       2005     301
 3 Altruism and … Altr…      1139     1862     1875   1126 No       2005      NA
 4 Animals and S… Anim…       473      820     1116    177 No       2005     209
 5 Asia/Asian Am… Asia       9056     2116     1710   9462 No       2005     365
 6 Body and Embo… Body       3408     1618     1920   3106 No       2005      NA
 7 Children and … Chil…      3692     3653     3713   3632 No       2005     418
 8 Coll Behavior… CBSM       8127     3470     2704   8893 No       2005     708
 9 Communication… CITA…     17093     4800     4804  17089 No       2005     301
10 Community and… Comm…     26598    24883    23379  28102 Yes      2005     721
# ℹ 562 more rows
```

Build your plots a piece at a time



Build your plots a piece at a time



Source: ASA annual report.

Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)
```

Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec
```

```
# A tibble: 572 × 9
   Section        Sname Beginning Revenues Expenses Ending Journal  Year Members
   <fct>          <fct>     <int>    <int>    <int>  <int> <fct>   <int>   <int>
 1 Aging and the… Aging     12752    12104    12007  12849 No       2005     598
 2 Alcohol, Drug… Alco…     11933     1144      400  12677 No       2005     301
 3 Altruism and … Altr…      1139     1862     1875   1126 No       2005      NA
 4 Animals and S… Anim…       473      820     1116    177 No       2005     209
 5 Asia/Asian Am… Asia       9056     2116     1710   9462 No       2005     365
 6 Body and Embo… Body       3408     1618     1920   3106 No       2005      NA
 7 Children and … Chil…      3692     3653     3713   3632 No       2005     418
 8 Coll Behavior… CBSM       8127     3470     2704   8893 No       2005     708
 9 Communication… CITA…     17093     4800     4804  17089 No       2005     301
10 Community and… Comm…     26598    24883    23379  28102 Yes      2005     721
# ℹ 562 more rows
```

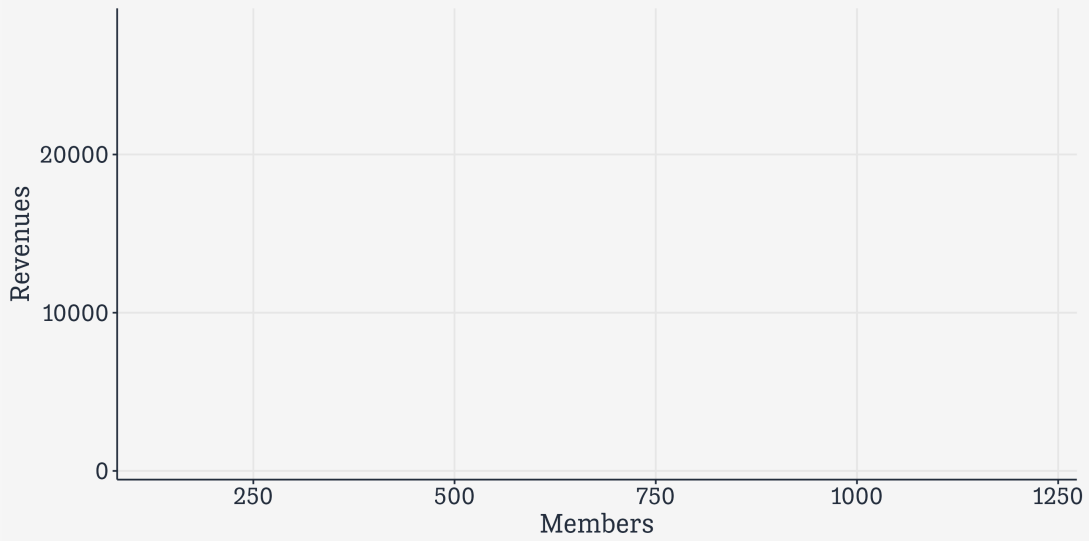
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014)
```

```
# A tibble: 52 × 9
   Section        Sname Beginning Revenues Expenses Ending Journal  Year Members
   <fct>          <fct>     <int>    <int>    <int>  <int> <fct>   <int>   <int>
 1 Aging and the… Aging     12752    12104    12007  12849 No       2014     580
 2 Alcohol, Drug… Alco…     11933     1144      400  12677 No       2014     173
 3 Altruism and … Altr…      1139     1862     1875   1126 No       2014     318
 4 Animals and S… Anim…       473      820     1116    177 No       2014     154
 5 Asia/Asian Am… Asia       9056     2116     1710   9462 No       2014     336
 6 Body and Embo… Body       3408     1618     1920   3106 No       2014     312
 7 Children and … Chil…      3692     3653     3713   3632 No       2014     421
 8 Coll Behavior… CBSM       8127     3470     2704   8893 No       2014     835
 9 Communication… CITA…     17093     4800     4804  17089 No       2014     371
10 Community and… Comm…     26598    24883    23379  28102 Yes      2014     630
# ℹ 42 more rows
```

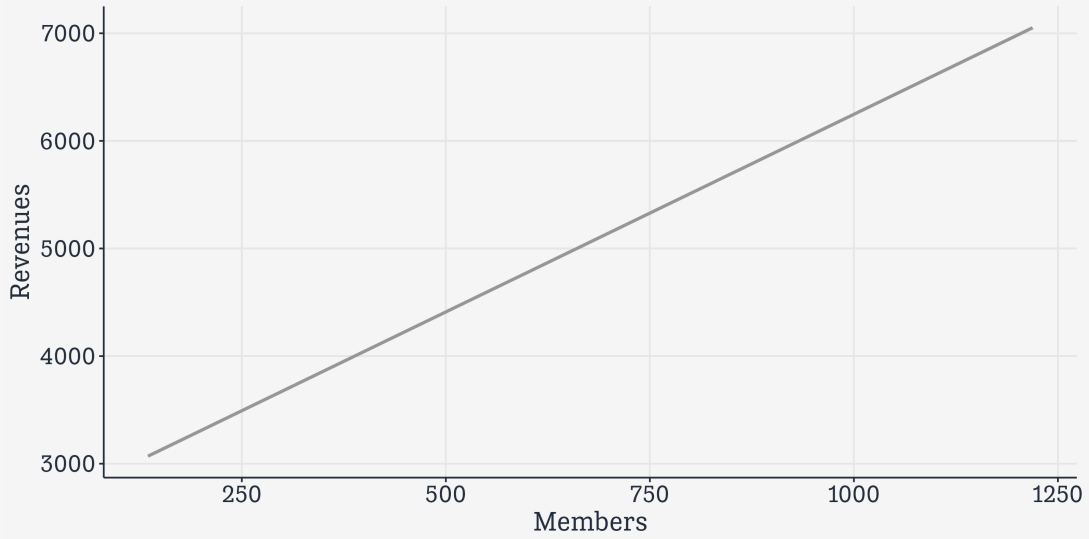
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname))
```



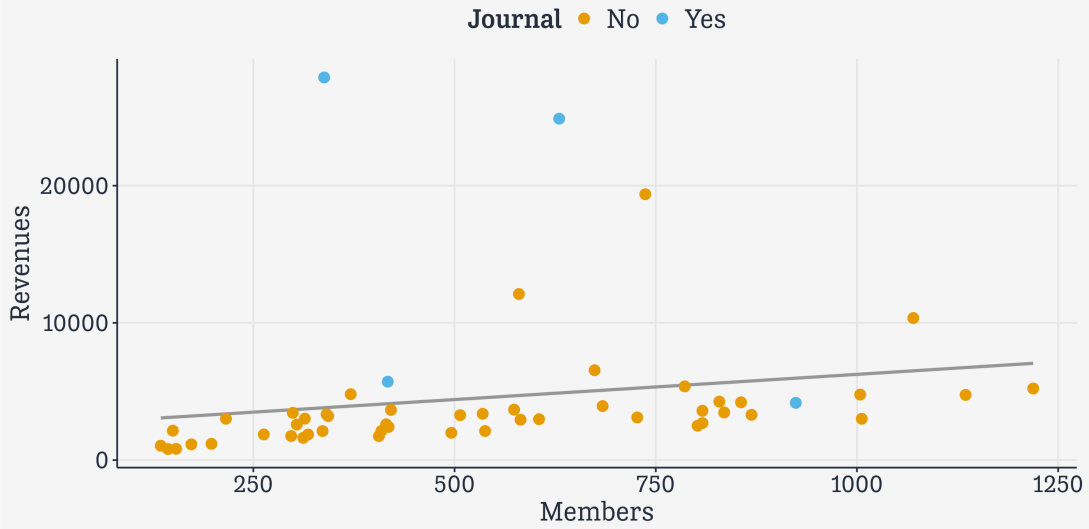
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60")
```



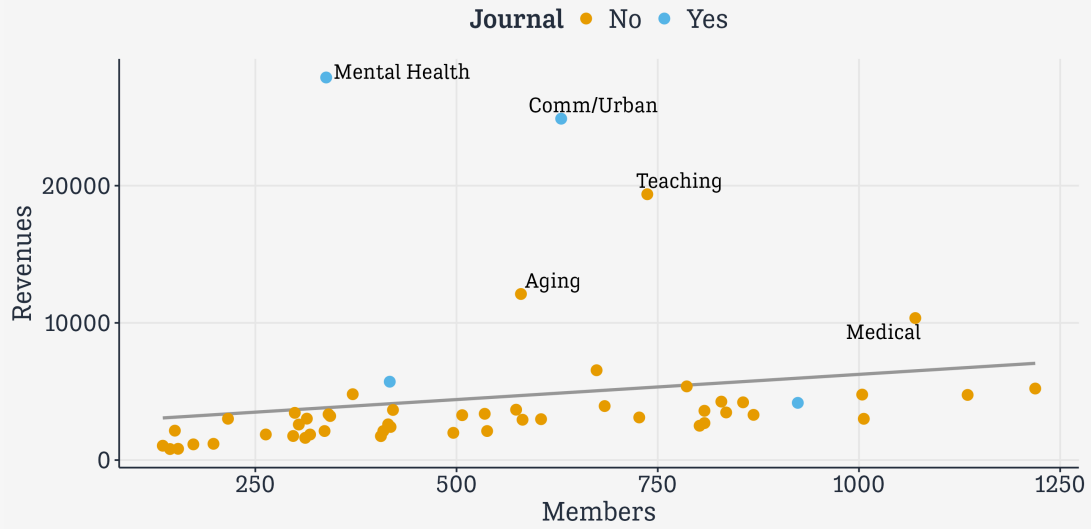
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3))
```



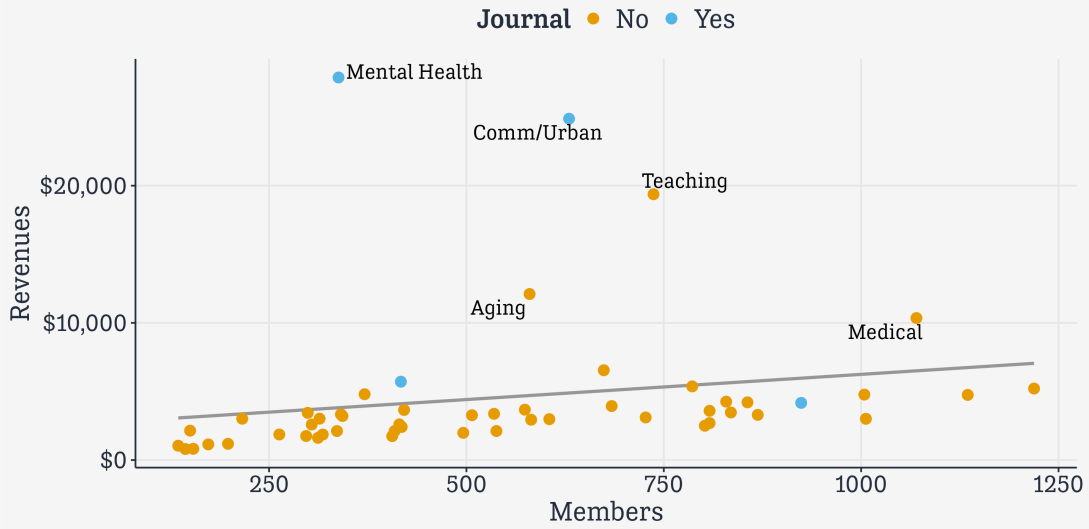
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3)) +
  geom_text_repel(data = df_rich,
                  size = rel(5),
                  mapping =
                  aes(family = "Tenso Slide"))
```



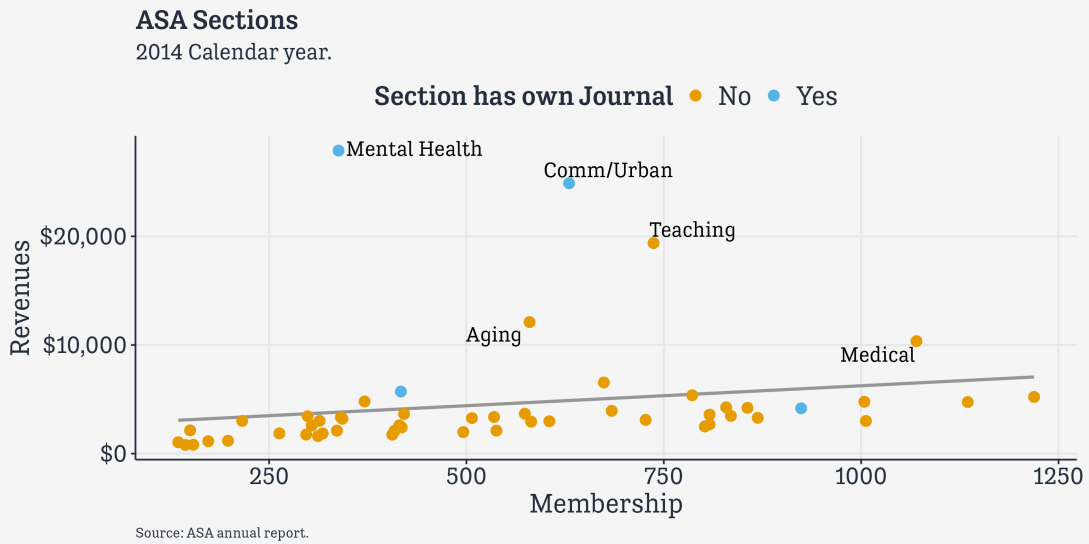
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3)) +
  geom_text_repel(data = df_rich,
                  size = rel(5),
                  mapping =
                  aes(family = "Tenso Slide")) +
  scale_y_continuous(labels =
                       scales::label_dollar())
```



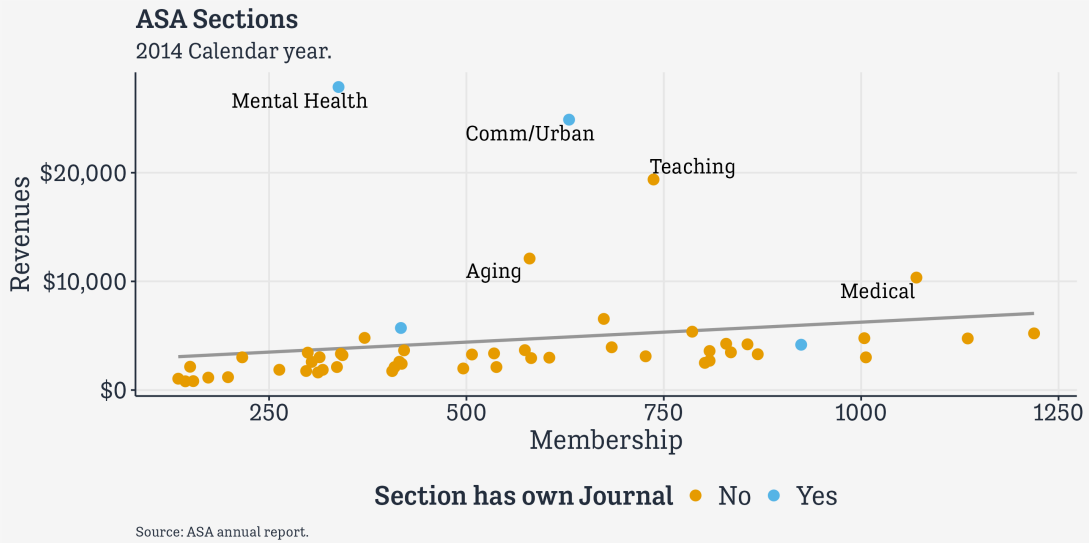
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3)) +
  geom_text_repel(data = df_rich,
                  size = rel(5),
                  mapping =
                  aes(family = "Tenso Slide")) +
  scale_y_continuous(labels =
                       scales::label_dollar()) +
  labs(x="Membership", y="Revenues",
       color = "Section has own Journal",
       title = "ASA Sections",
       subtitle = "2014 Calendar year.",
       caption = "Source: ASA annual report.")
```



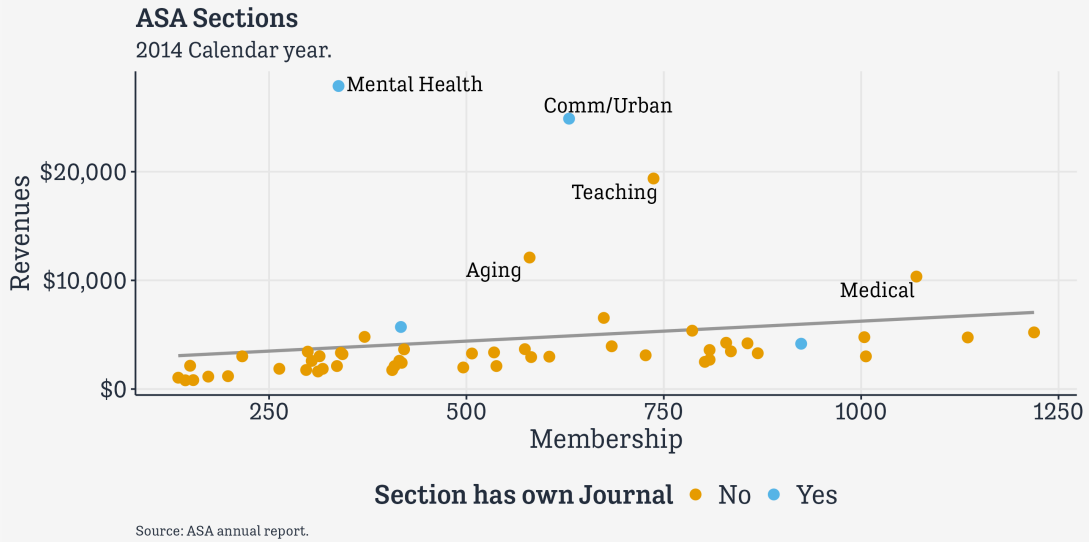
Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3)) +
  geom_text_repel(data = df_rich,
                  size = rel(5),
                  mapping =
                  aes(family = "Tenso Slide")) +
  scale_y_continuous(labels =
                       scales::label_dollar()) +
  labs(x="Membership", y="Revenues",
       color = "Section has own Journal",
       title = "ASA Sections",
       subtitle = "2014 Calendar year.",
       caption = "Source: ASA annual report.") +
  theme(legend.position = "bottom")
```



Build your plots a piece at a time

```
df_rich ← asasec ▷ filter(Year == 2014 &
                          Revenues > 7000)

asasec ▷
  filter(Year == 2014) ▷
  ggplot(mapping = aes(x = Members,
                       y = Revenues,
                       label = Sname)) +
  geom_smooth(method = "lm",
              se = FALSE,
              color = "gray60") +
  geom_point(mapping = aes(color = Journal),
             size = rel(3)) +
  geom_text_repel(data = df_rich,
                  size = rel(5),
                  mapping =
                  aes(family = "Tenso Slide")) +
  scale_y_continuous(labels =
                       scales::label_dollar()) +
  labs(x="Membership", y="Revenues",
       color = "Section has own Journal",
       title = "ASA Sections",
       subtitle = "2014 Calendar year.",
       caption = "Source: ASA annual report.") +
  theme(legend.position = "bottom")
```



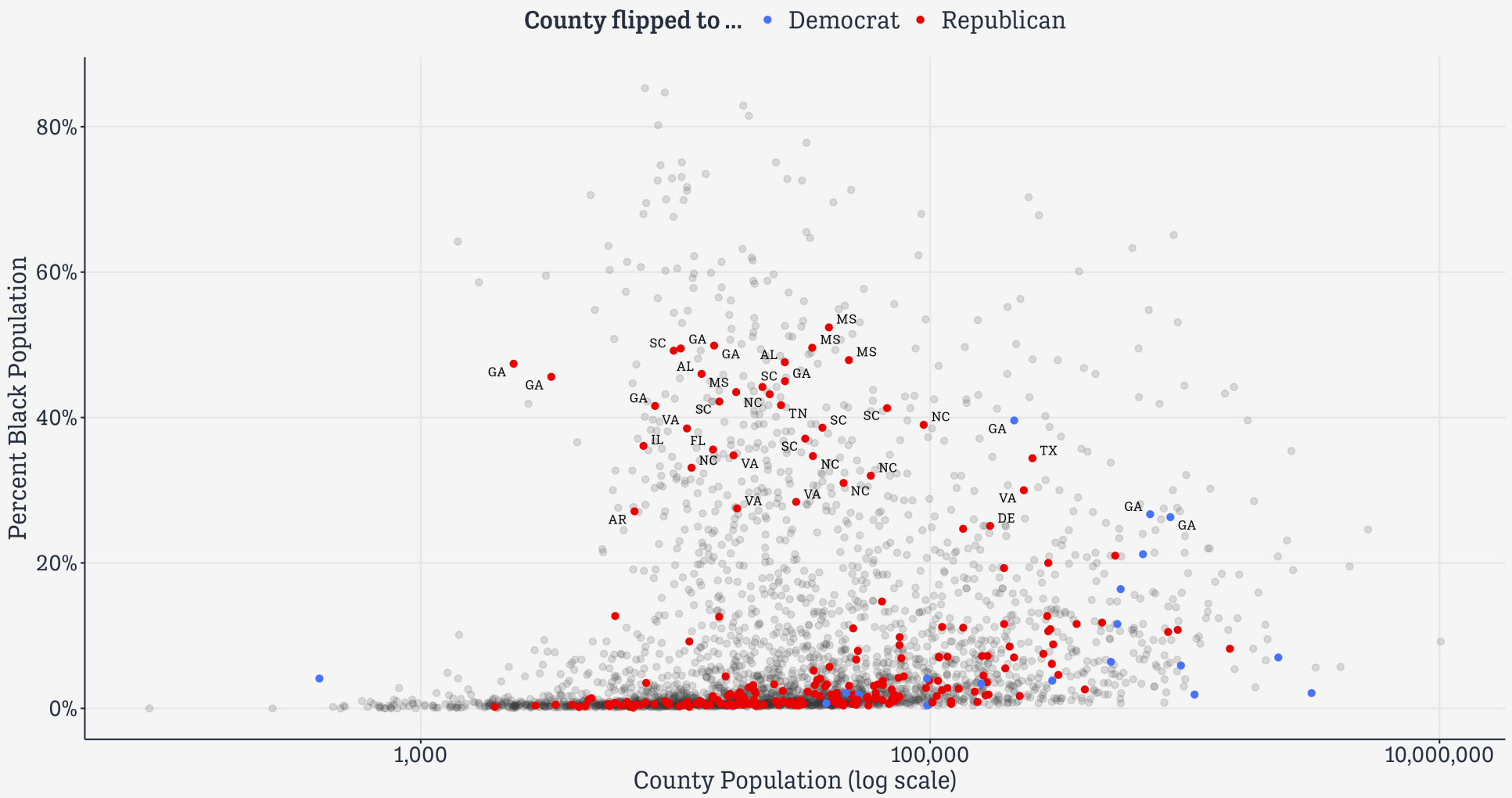
Layer color and text to your advantage

Some 2016 Election Data

```
county_data ▷
  select(name, state, pop, black, partywinner16, flipped) ▷
  sample_n(10)
```

```
                  name state    pop black partywinner16 flipped
1        Racine County    WI 195163  11.6    Republican     Yes
2        Monroe County    AL  21947  41.3    Republican      No
3        Owyhee County    ID  11353   0.9    Republican      No
4          Norton city    VA   4031   6.7    Republican      No
5     Worcester County    MD  51675  13.9    Republican      No
6    New London County    CT 273676   6.8      Democrat      No
7      Clermont County    OH 201560   1.4    Republican      No
8     Chattooga County    GA  24939  10.4    Republican      No
9       Madison County    KY  87340   4.5    Republican      No
10 Westmoreland County    PA 359320   2.5    Republican      No
```

Flipped counties, 2016



Counties in gray did not flip.

We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")
```

We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
```

We build it a piece at a time

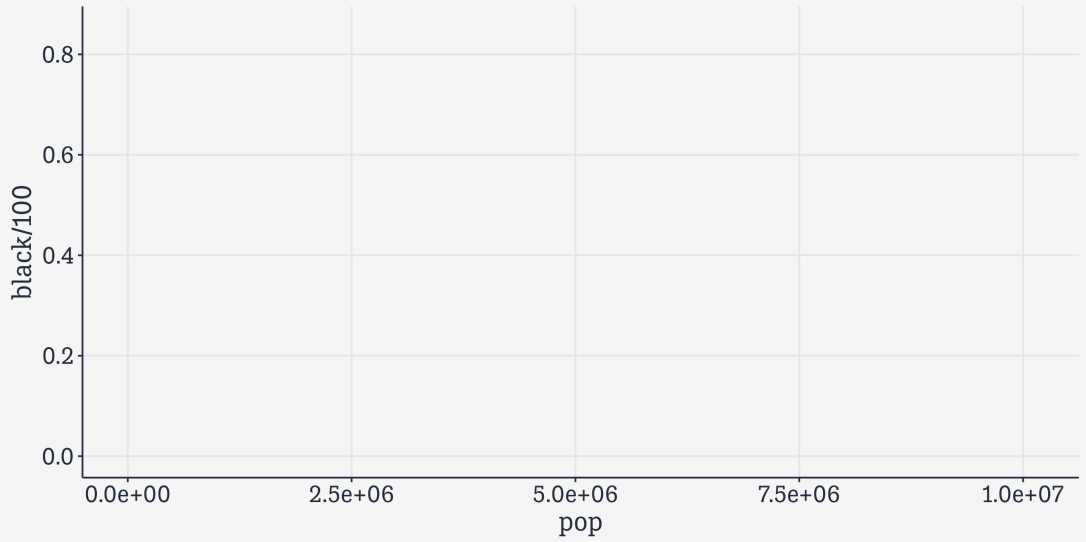
```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
```

We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black
```

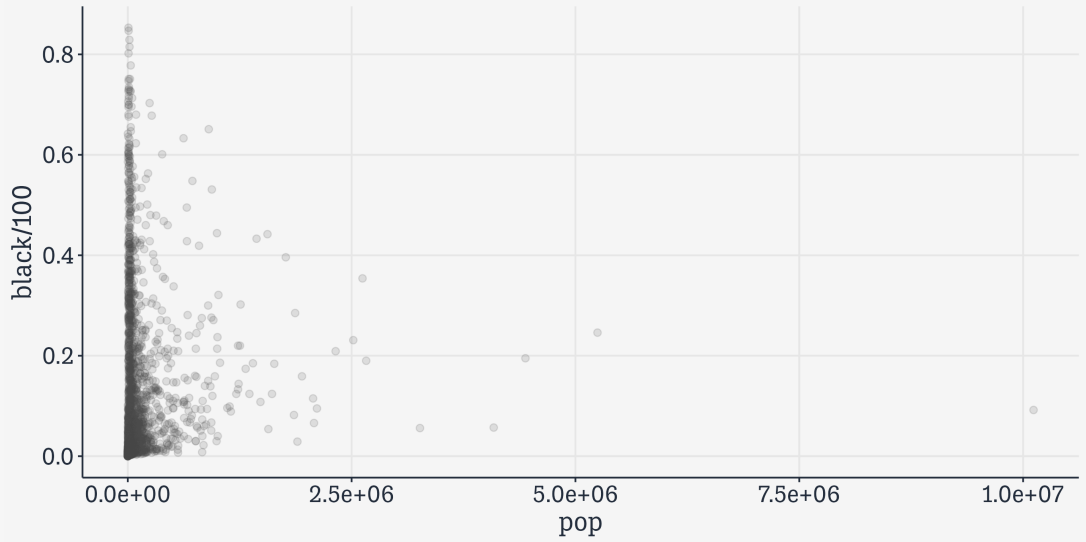
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100))
```



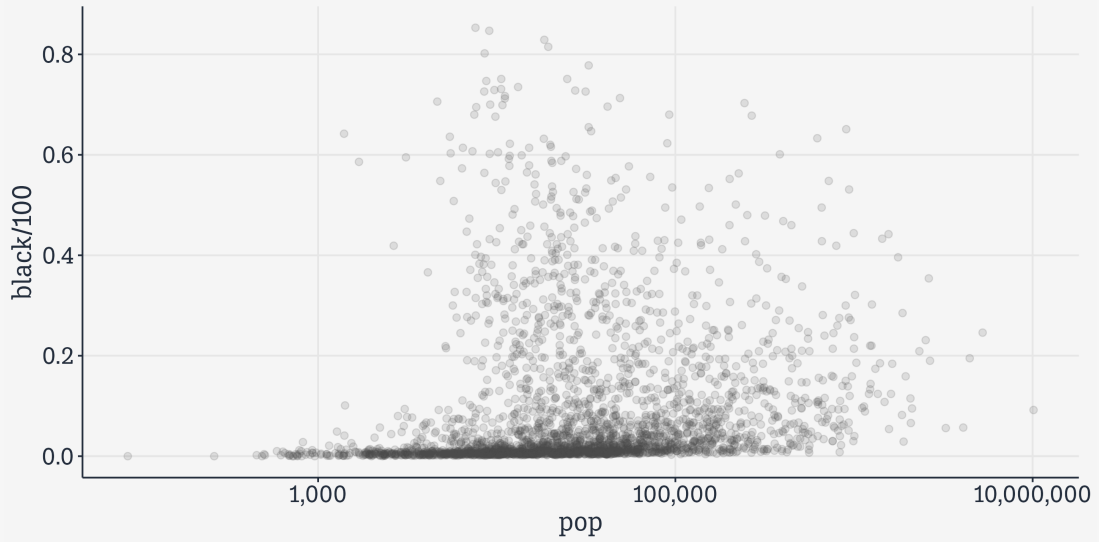
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2))
```



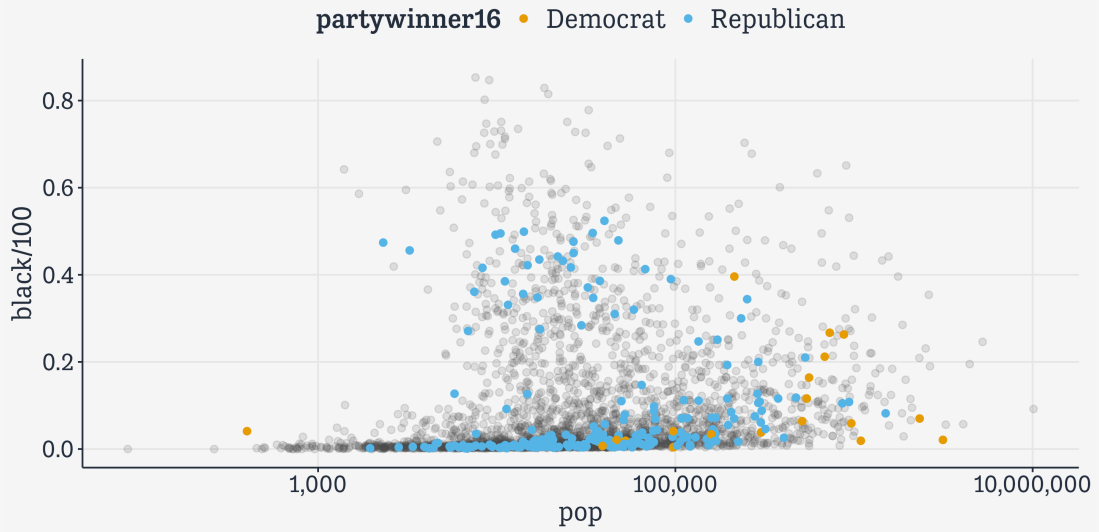
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma())
```



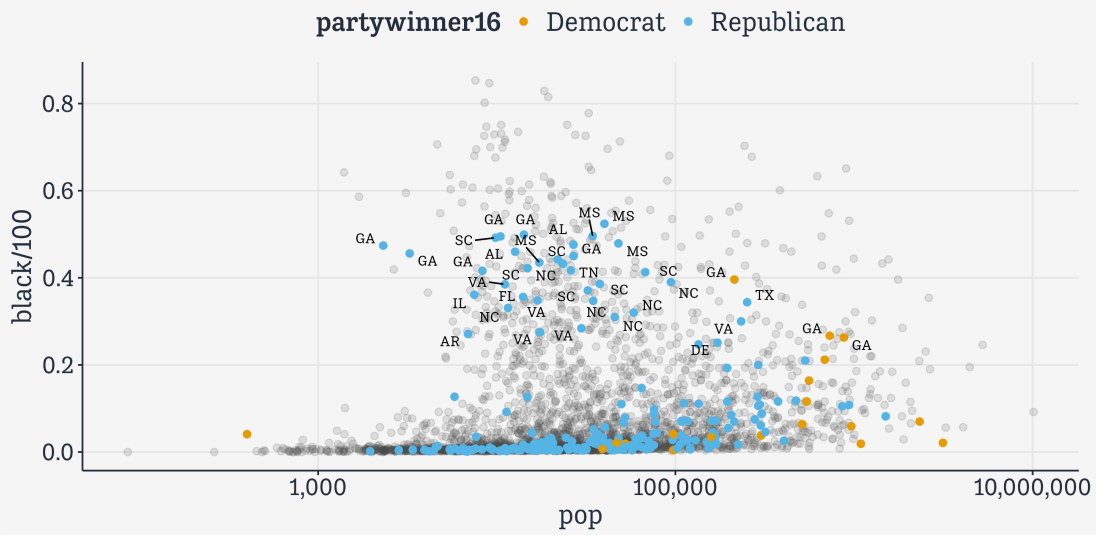
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2))
```



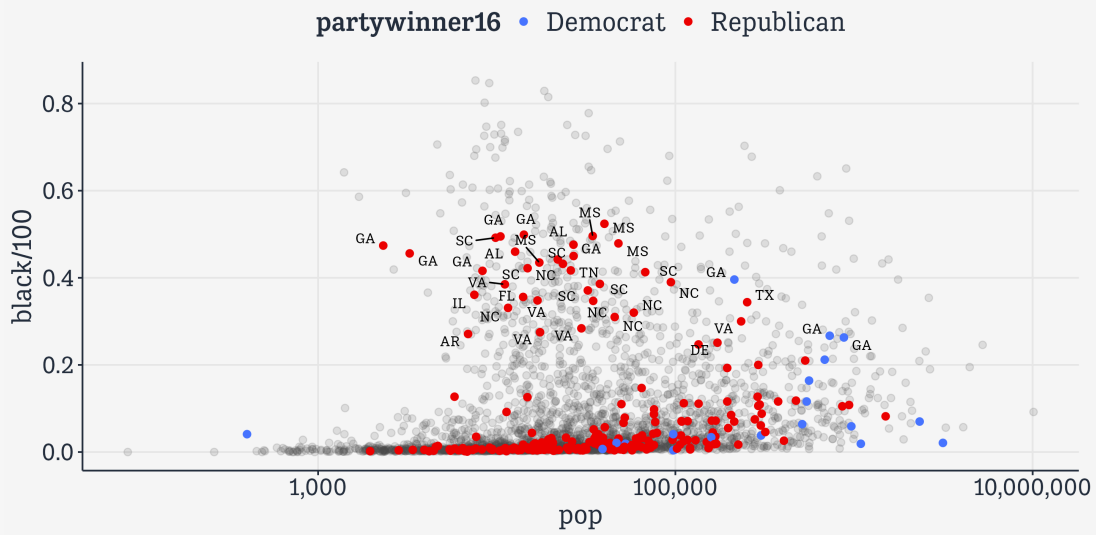
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2)) +
  geom_text_repel(data = df_text,
                  mapping = aes(label = state,
                    family = "Tenso Slide"),
                    size = rel(3.5))
```



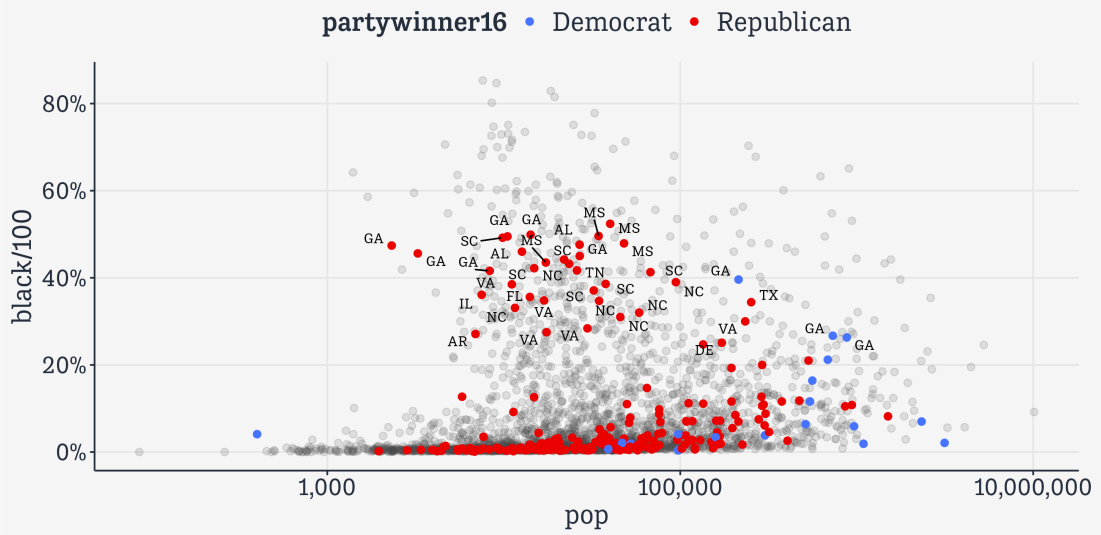
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2)) +
  geom_text_repel(data = df_text,
                  mapping = aes(label = state,
                    family = "Tenso Slide"),
                    size = rel(3.5)) +
  scale_color_manual(values = party_colors)
```



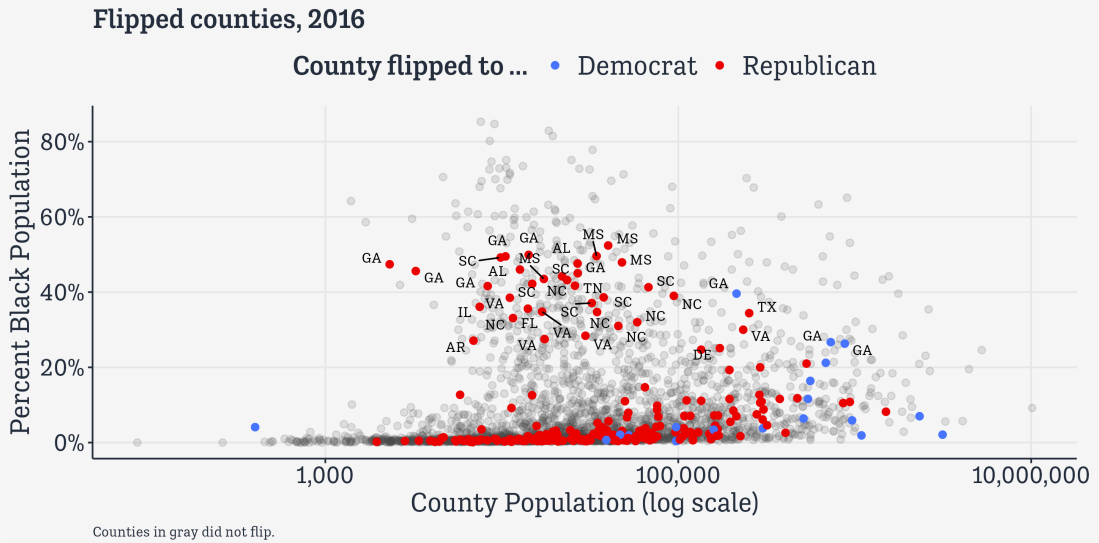
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2)) +
  geom_text_repel(data = df_text,
                  mapping = aes(label = state,
                    family = "Tenso Slide"),
                    size = rel(3.5)) +
  scale_color_manual(values = party_colors) +
  scale_y_continuous(labels = label_percent())
```



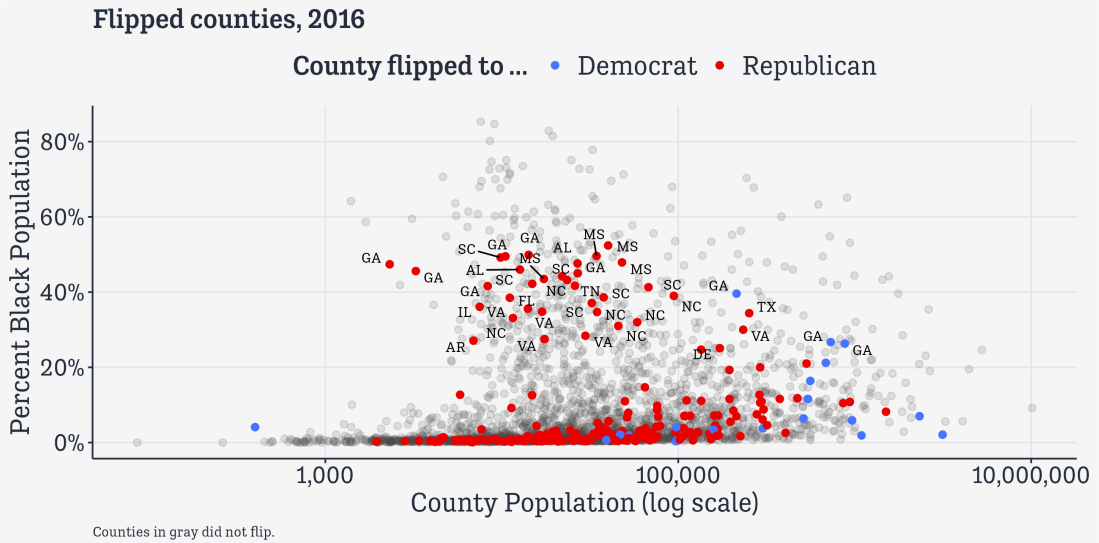
We build it a piece at a time

```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2)) +
  geom_text_repel(data = df_text,
                  mapping = aes(label = state,
                    family = "Tenso Slide"),
                    size = rel(3.5)) +
  scale_color_manual(values = party_colors) +
  scale_y_continuous(labels = label_percent()) +
  labs(color = "County flipped to ... ",
       x = "County Population (log scale)",
       y = "Percent Black Population",
       title = "Flipped counties, 2016",
       caption = "Counties in gray did not flip.")
```

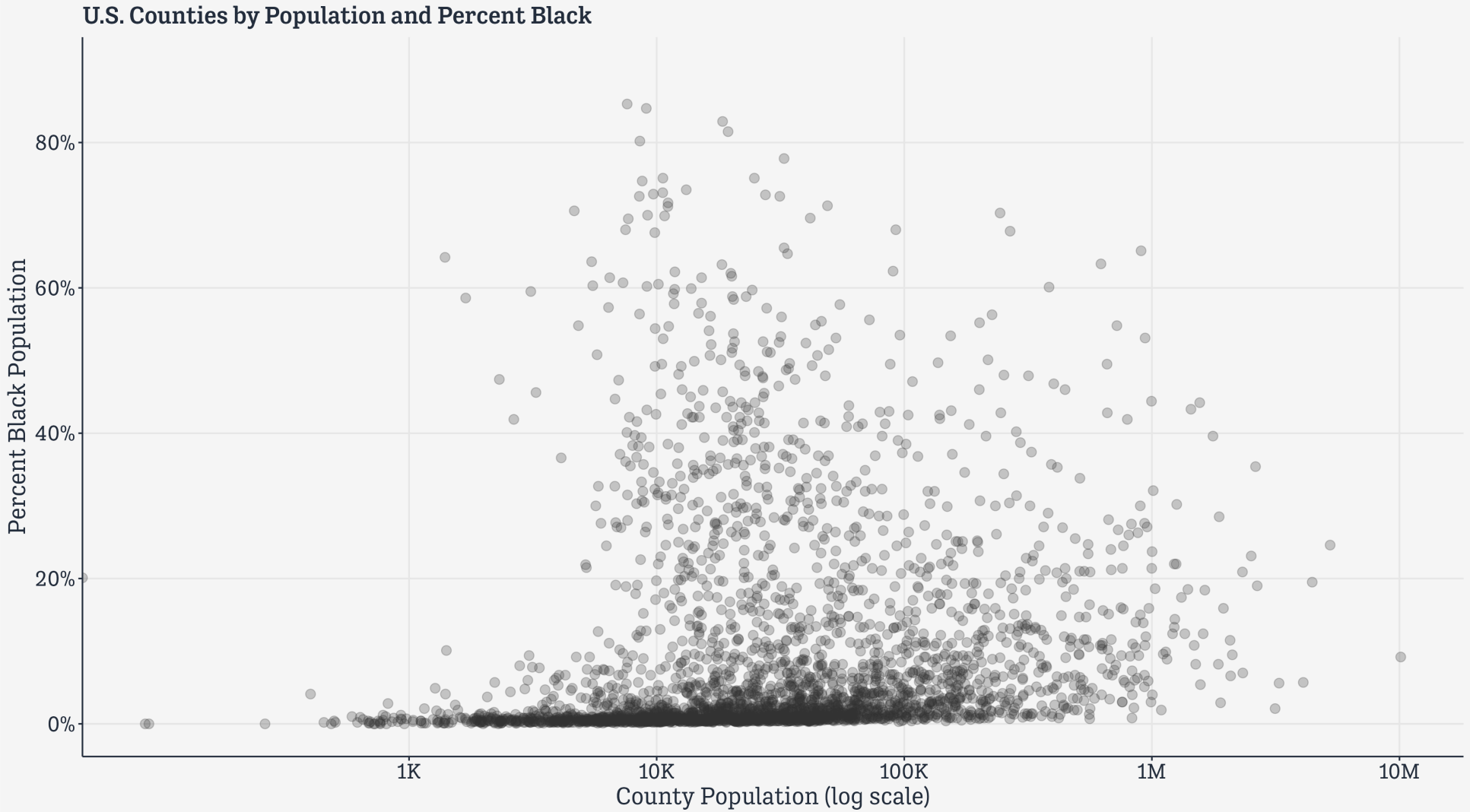


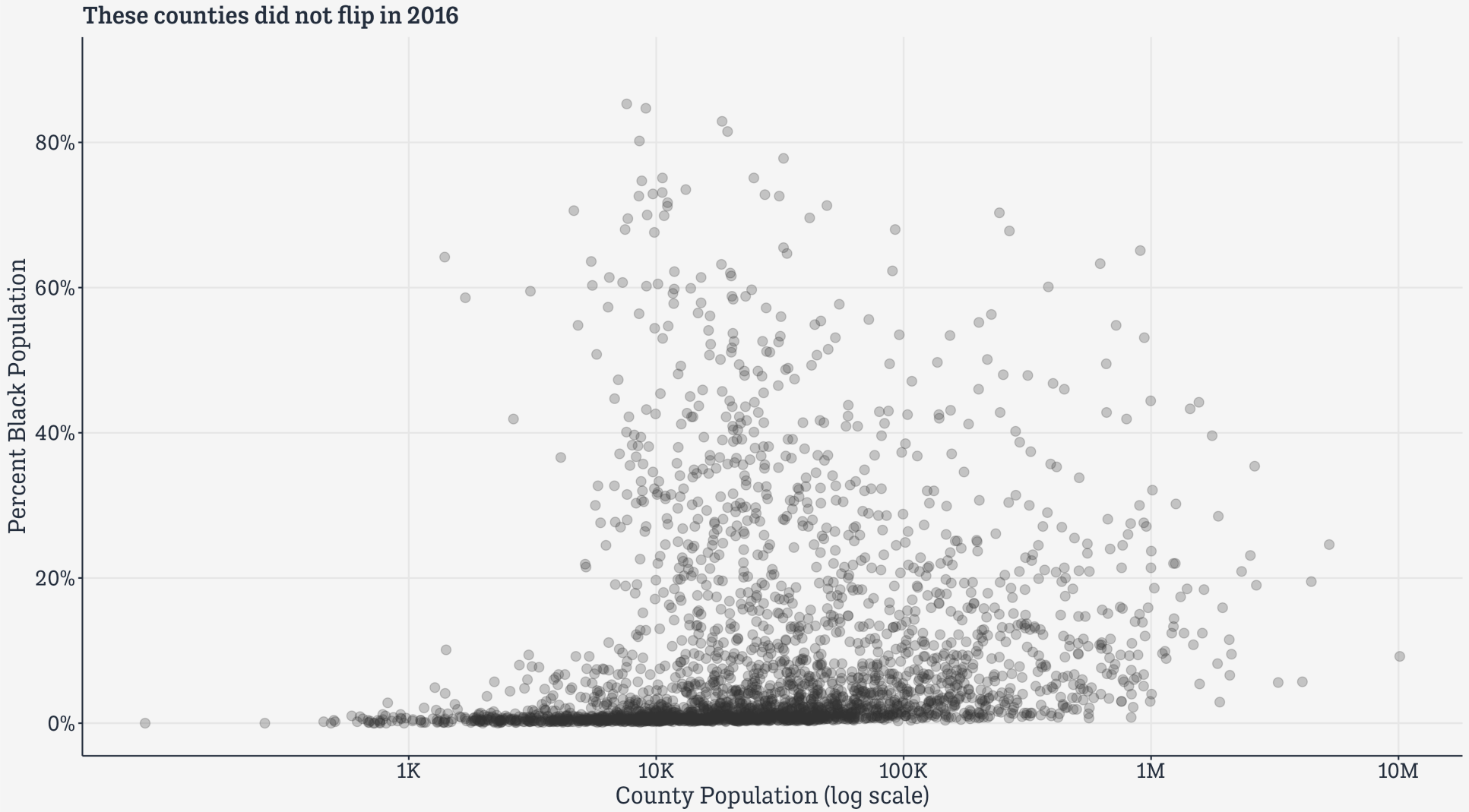
We build it a piece at a time

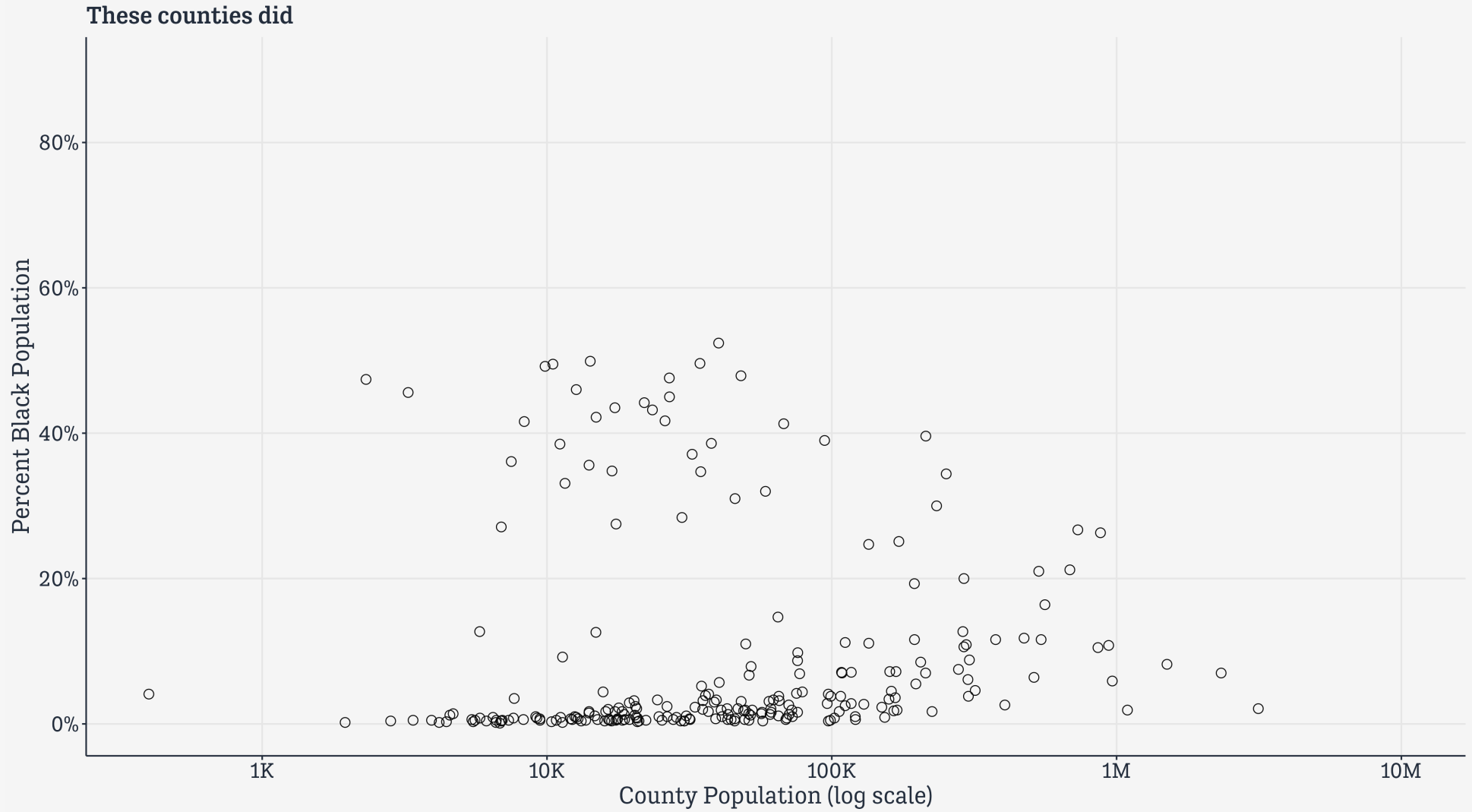
```
## Brighter Blue and Red
party_colors ← c("royalblue1", "red2")

## Data subsets for our geoms
df_noflip ← county_data ▷ filter(flipped == "No")
df_flip ← county_data ▷ filter(flipped == "Yes")
df_text ← county_data ▷ filter(flipped == "Yes" & black

ggplot(data = df_noflip,
       mapping = aes(x = pop,
                     y = black/100)) +
  geom_point(alpha = 0.15,
             color = "gray30", size = rel(2)) +
  scale_x_log10(labels = label_comma()) +
  geom_point(data = df_flip,
             mapping = aes(x = pop, y = black/100,
                           color = partywinner16),
             size = rel(2)) +
  geom_text_repel(data = df_text,
                  mapping = aes(label = state,
                    family = "Tenso Slide"),
                    size = rel(3.5)) +
  scale_color_manual(values = party_colors) +
  scale_y_continuous(labels = label_percent()) +
  labs(color = "County flipped to ... ",
       x = "County Population (log scale)",
       y = "Percent Black Population",
       title = "Flipped counties, 2016",
       caption = "Counties in gray did not flip.")
```

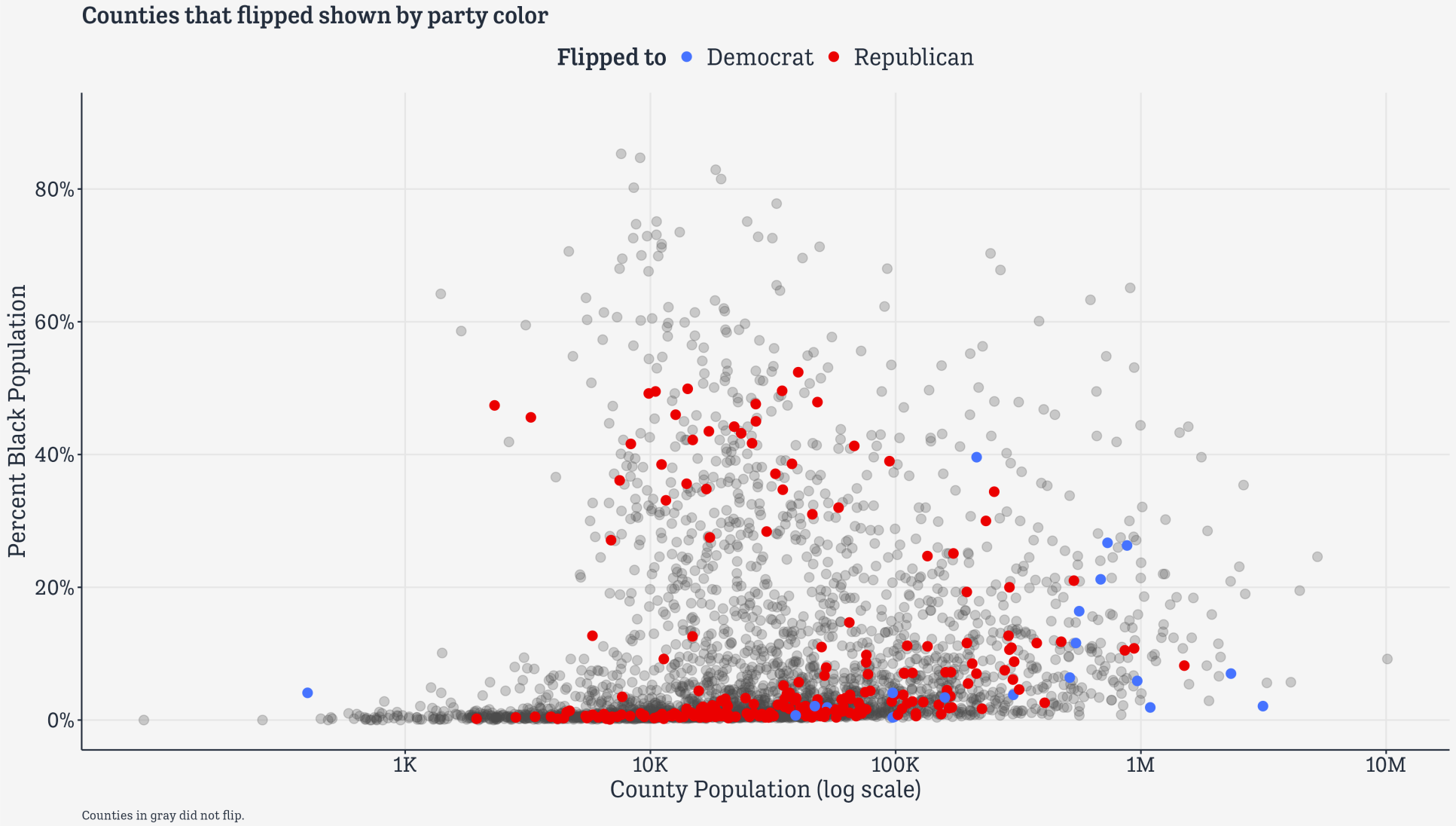


Leverage ggplot's layered approach

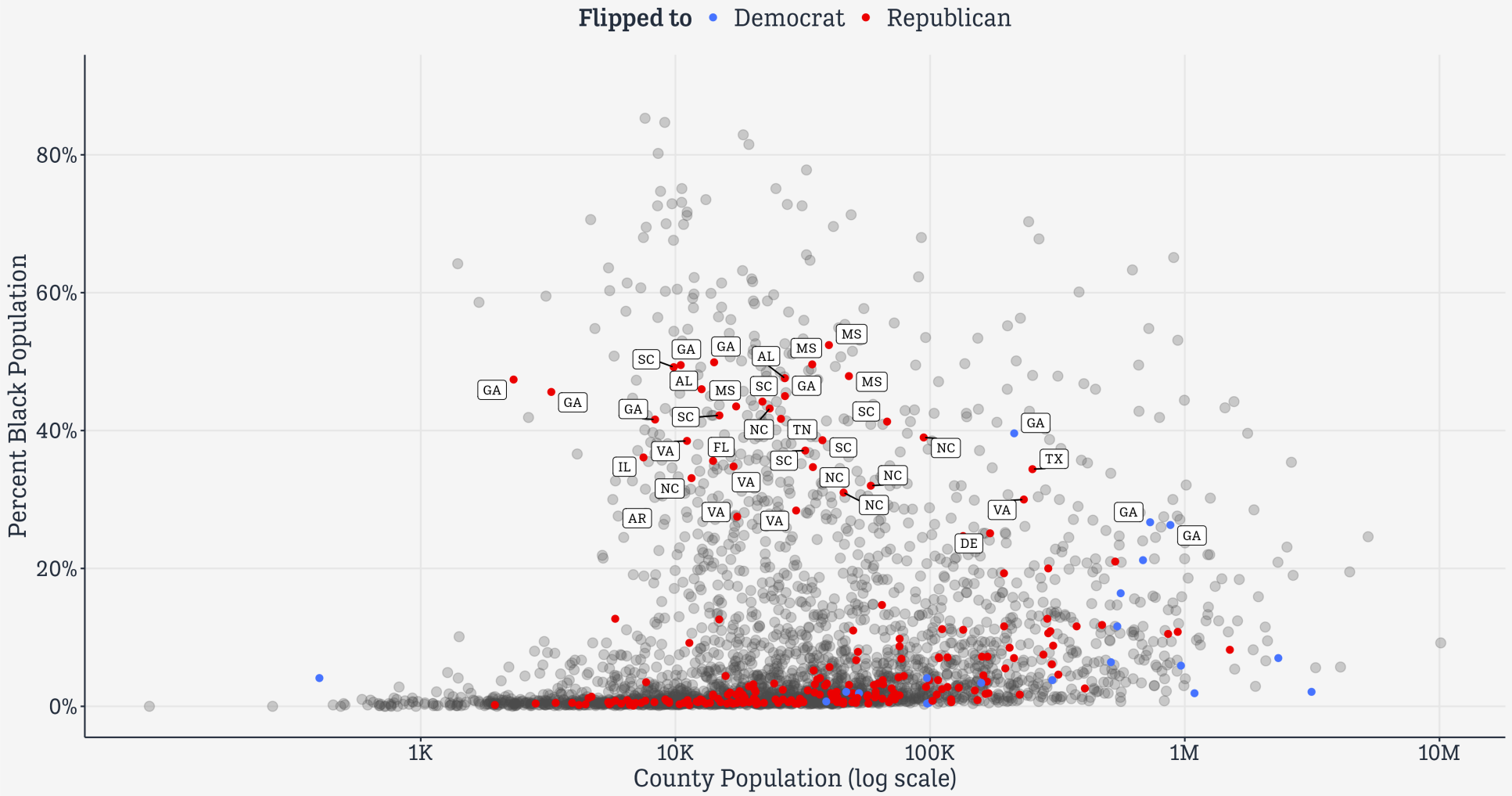

U.S. Counties by Population and Percent Black
Percent Black Population
80%
60%
40%
20%
0%
1K
10K
100K
1M
10M
County Population (log scale)


These counties did not flip in 2016
Percent Black Population
80%
60%
40%
20%
0%
1K
10K
100K
1M
10M
County Population (log scale)

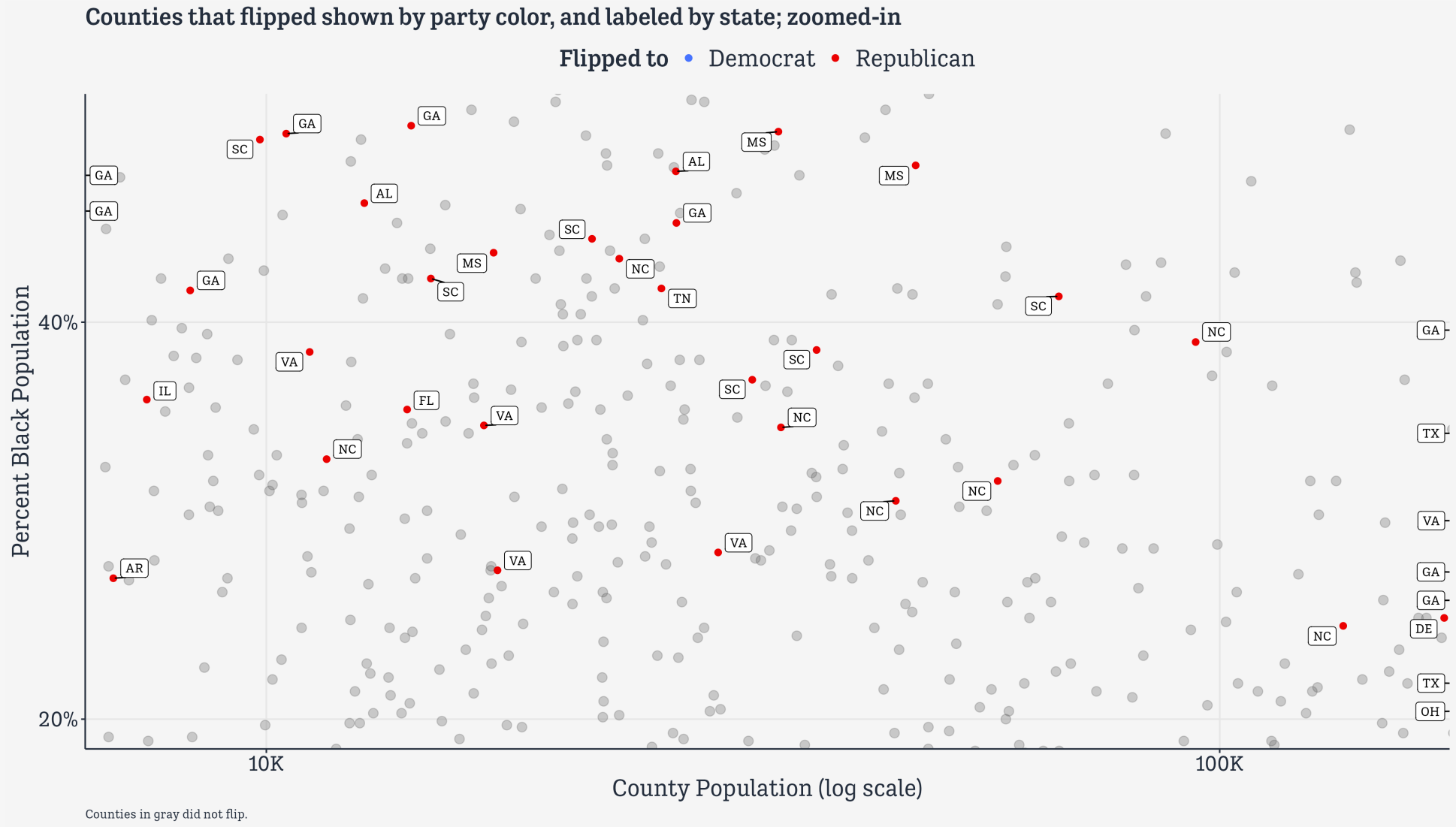

These counties did
80%
60%
40%
20%
0%
Percent Black Population
1K
10K
100K
1M
10M
County Population (log scale)


Counties that flipped shown by party color
Flipped to
Democrat
Republican
80%
60%
40%
20%
0%
Percent Black Population
1K
10K
100K
1M
10M
County Population (log scale)
Counties in gray did not flip.

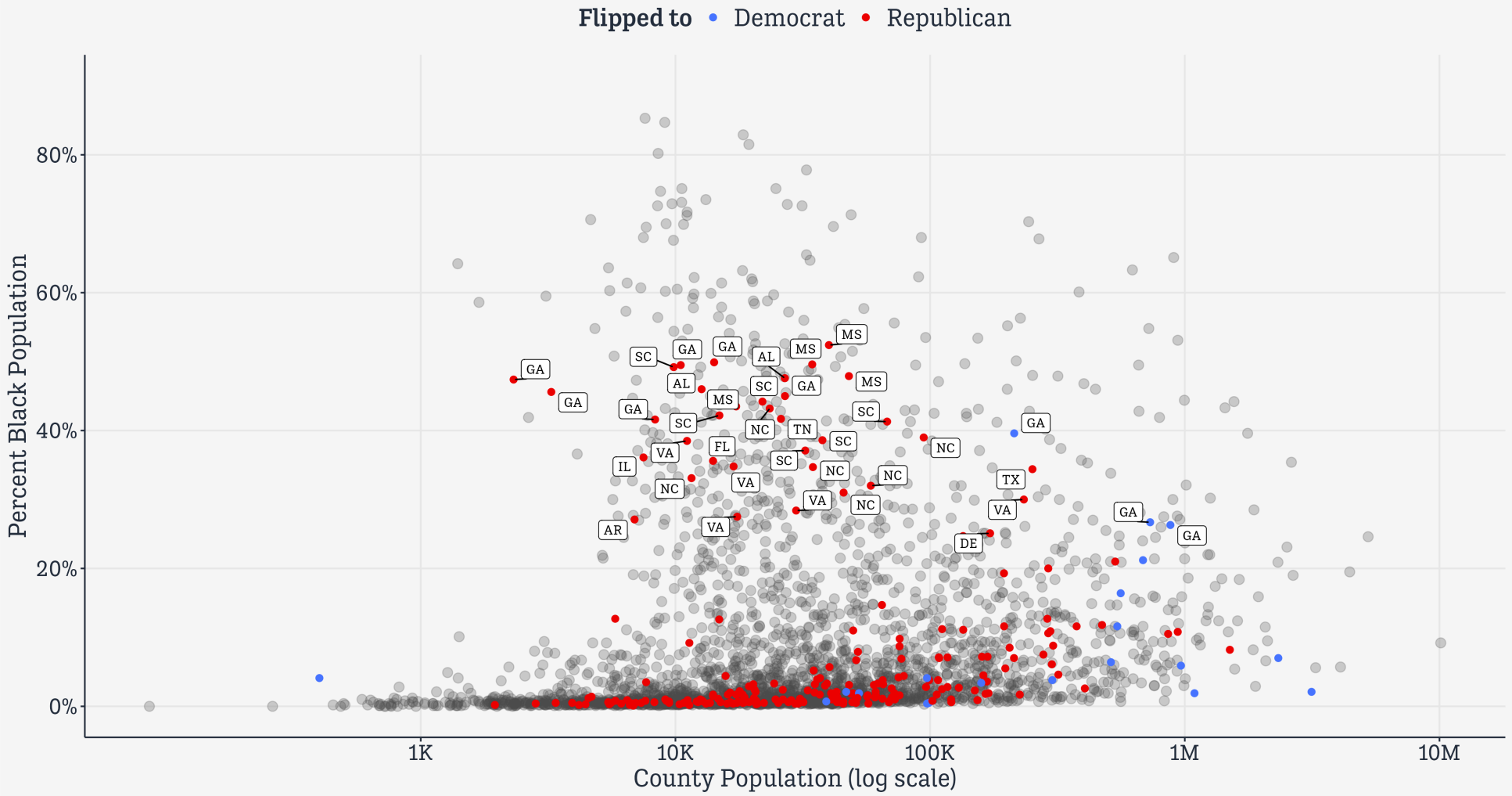
Counties that flipped shown by party color, and labeled by state



Counties in gray did not flip.


Counties that flipped shown by party color, and labeled by state; zoomed-in
Flipped to Democrat Republican
Percent Black Population
40%
20%
10K
100K
County Population (log scale)
GA
SC
GA
GA
AL
GA
GA
MS
AL
MS
GA
SC
MS
NC
SC
TN
GA
SC
VA
SC
NC
GA
IL
SC
FL
VA
NC
TX
NC
NC
VA
AR
VA
VA
GA
GA
NC
DE
TX
OH
Counties in gray did not flip.

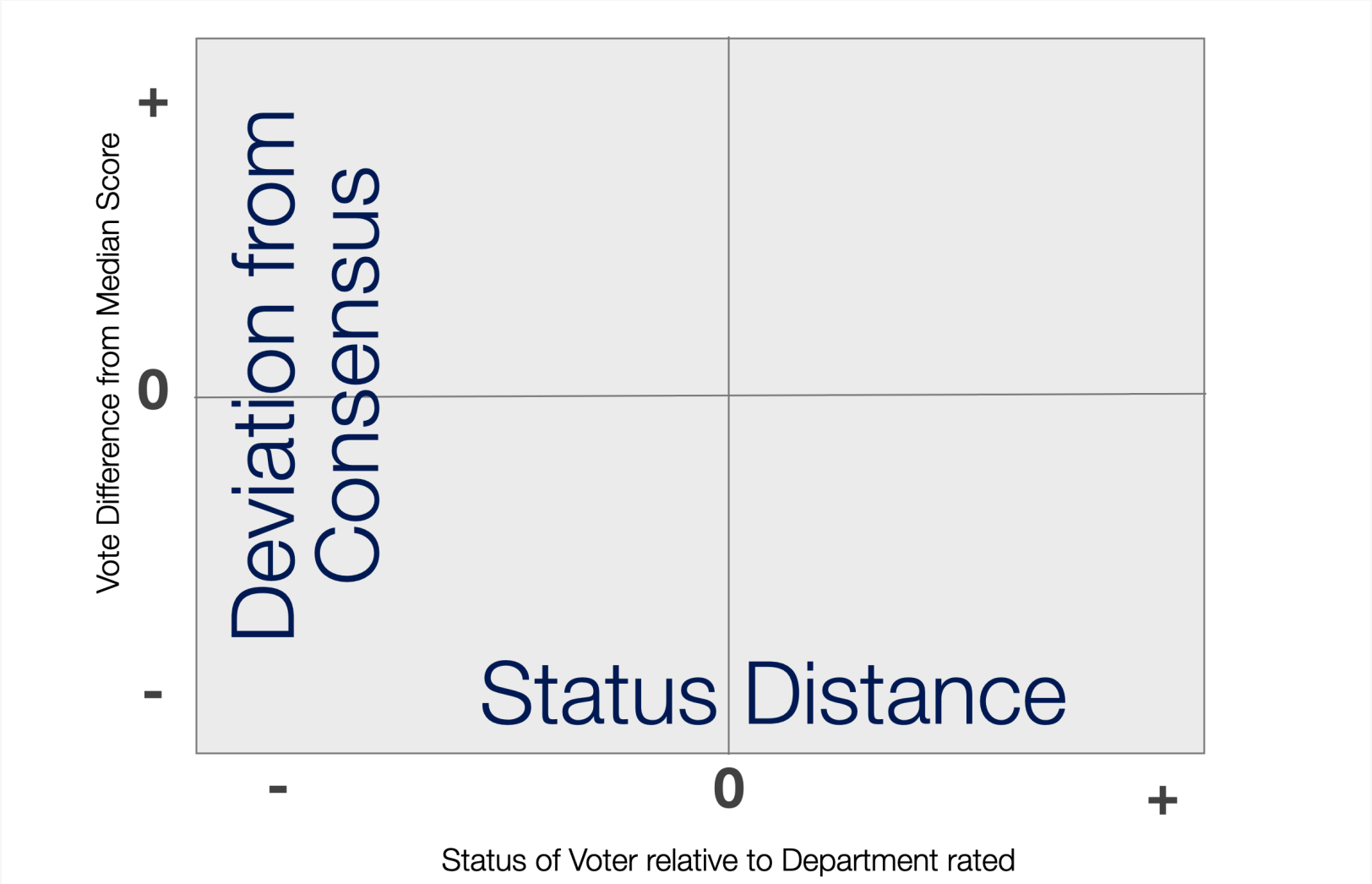
Counties that flipped shown by party color, and labeled by state



Counties in gray did not flip.

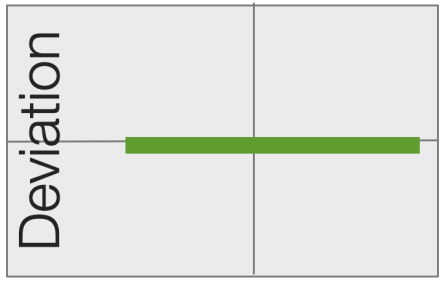
Layer,
Highlight,
Repeat

Build from ideas to data

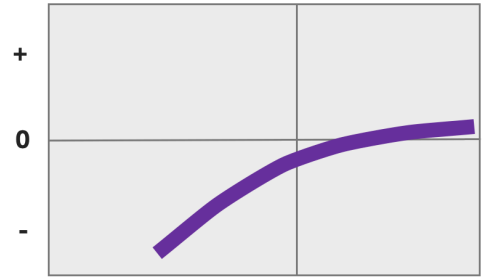


The relationship of interest

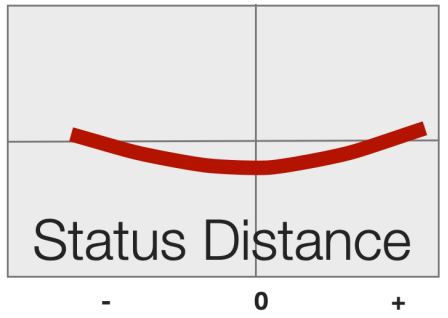
Build from ideas to data



1. Pure Objectivity



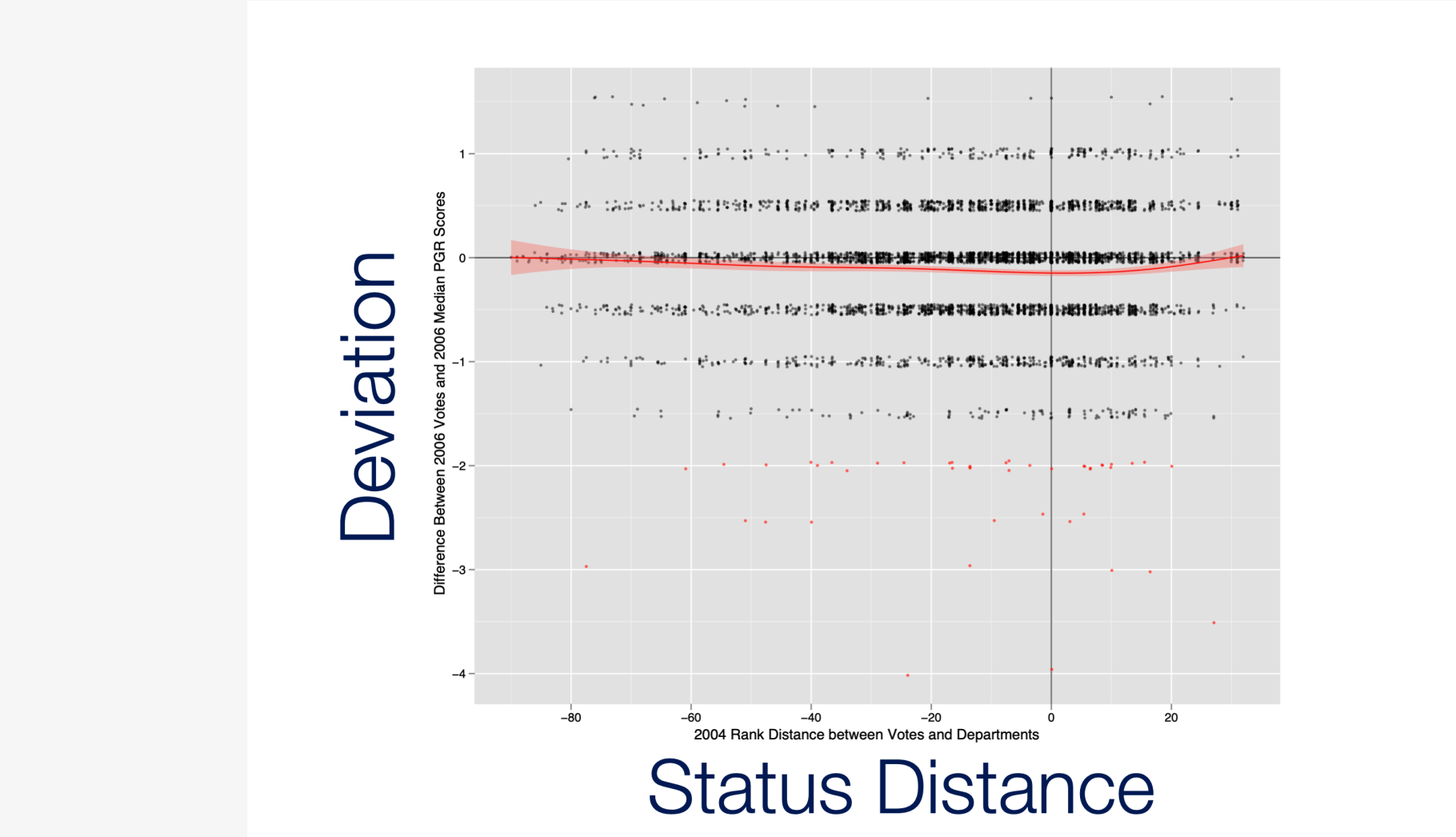
2. Distant Envy



3. Local Competition

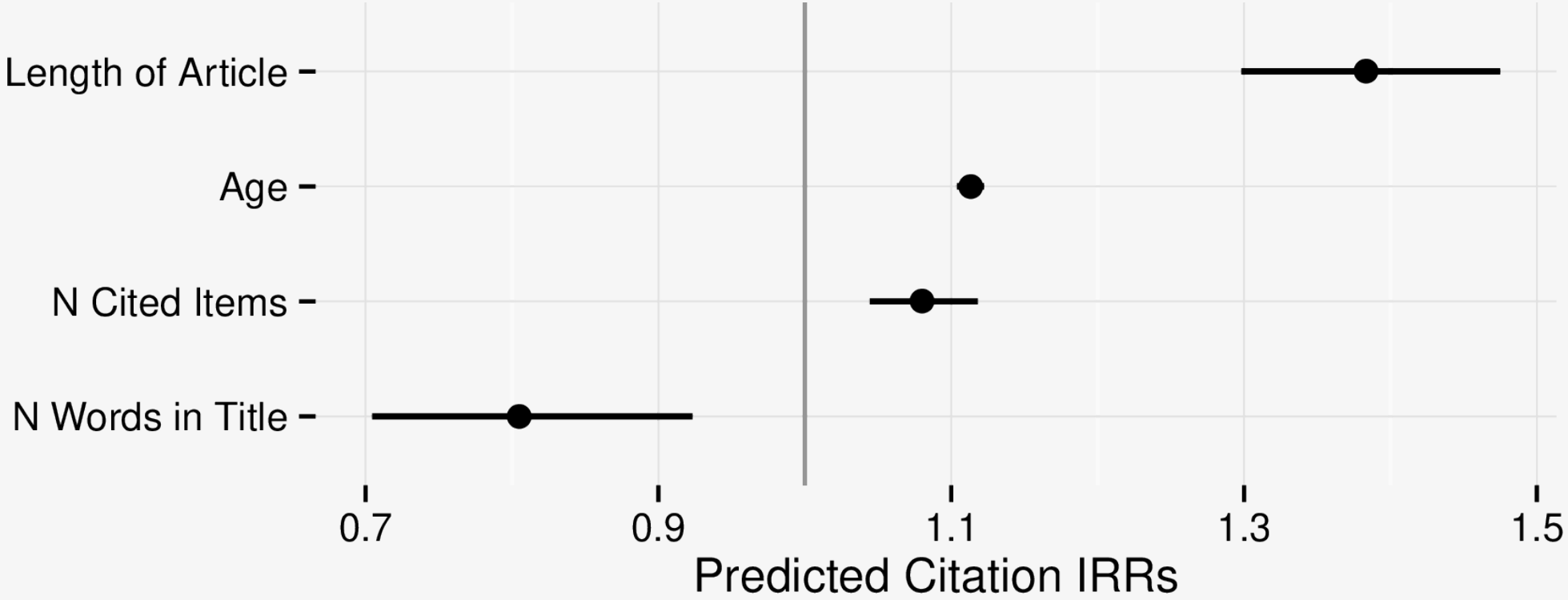
Theory says …

Build from ideas to data



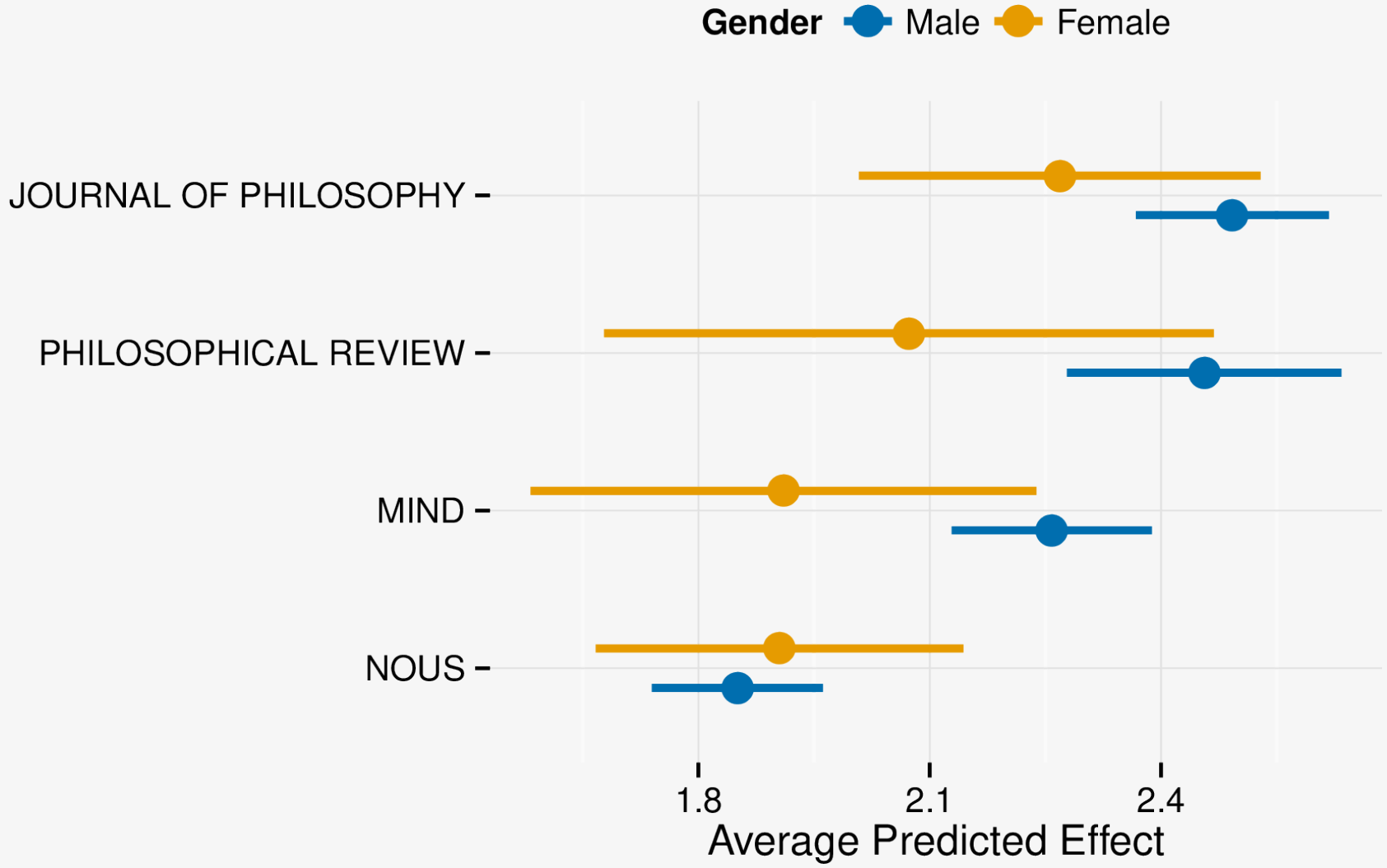
Data suggests ...

Repeat to differentiate



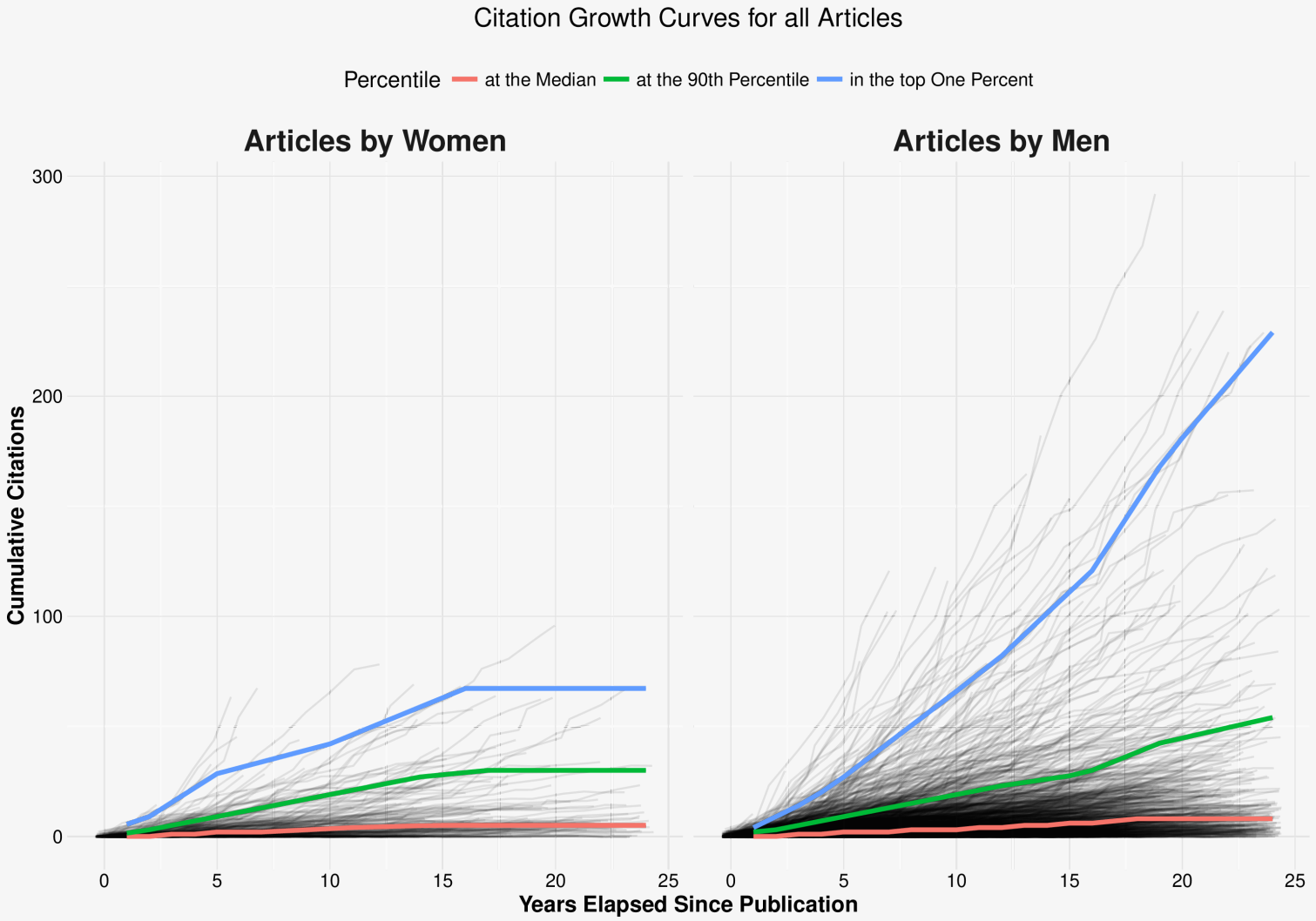
Pointrange

Repeat to differentiate



Add a comparison group with color and `position_dodge()`

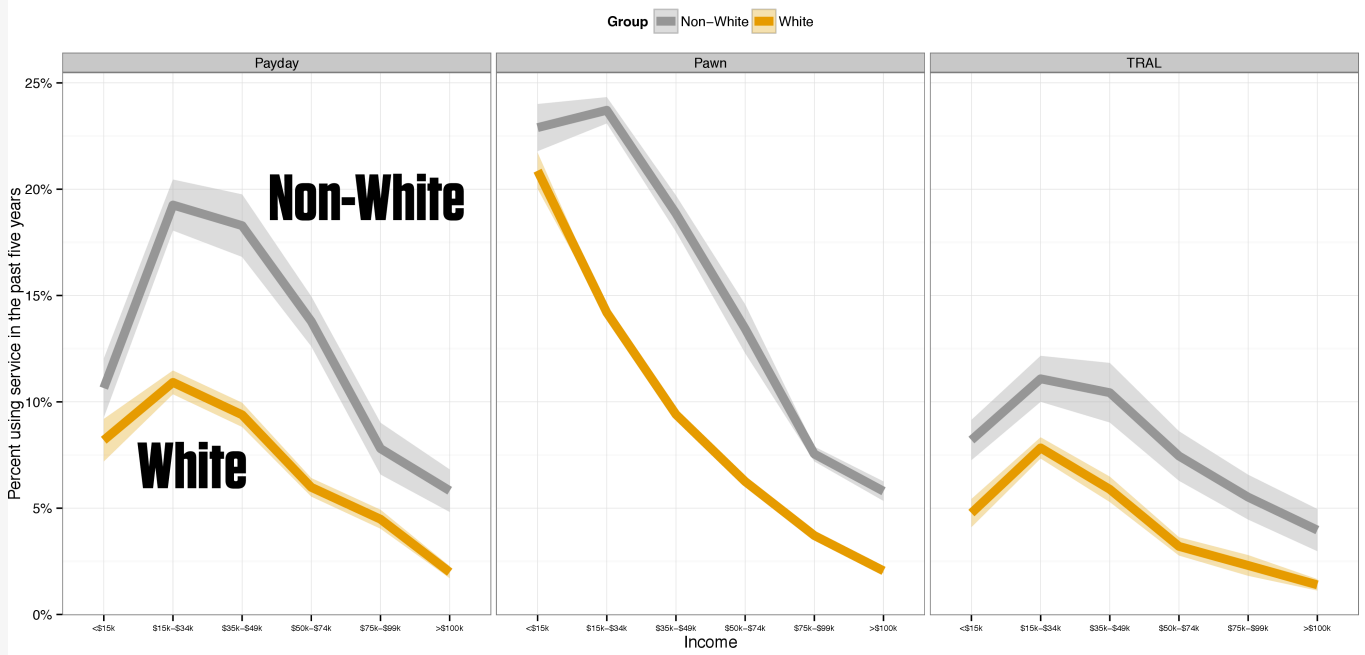
Layer and repeat with facets



Compare across facets

Layer and repeat across facets

Categorical Gaps: Alternative Financial Services

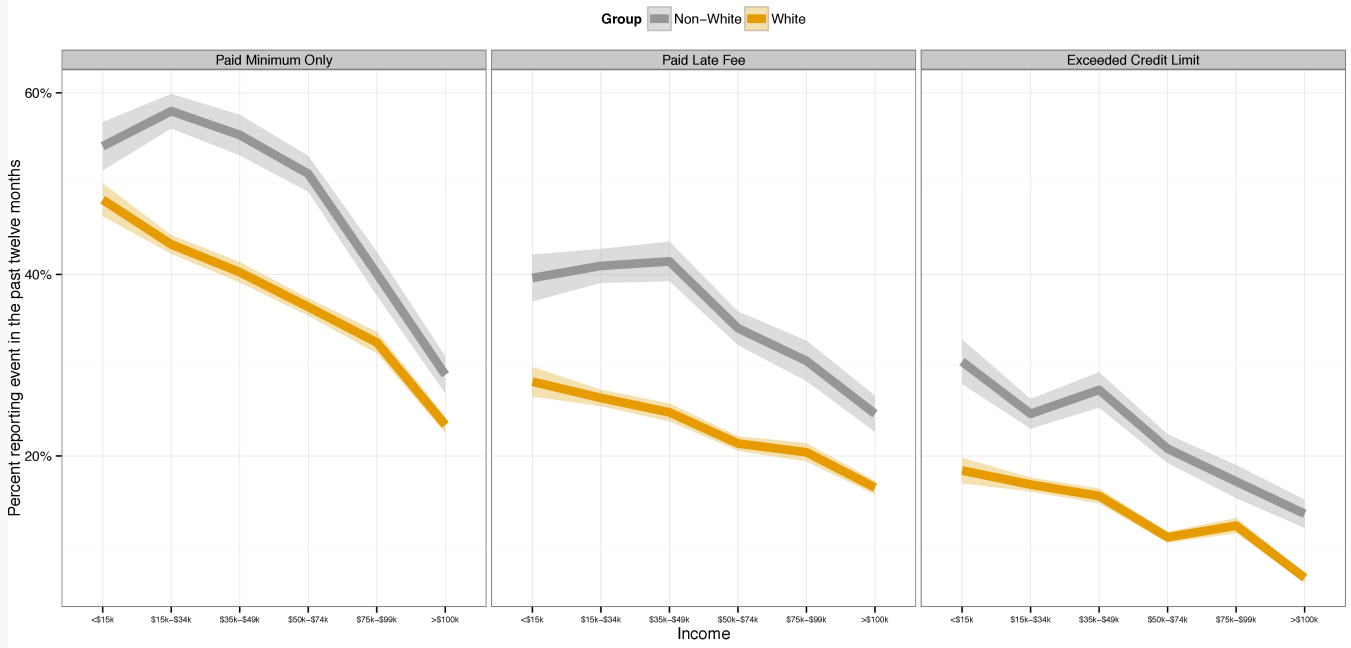


Payday Loan Pawn Shop TRAL

Layer to compare

Layer and repeat across facets

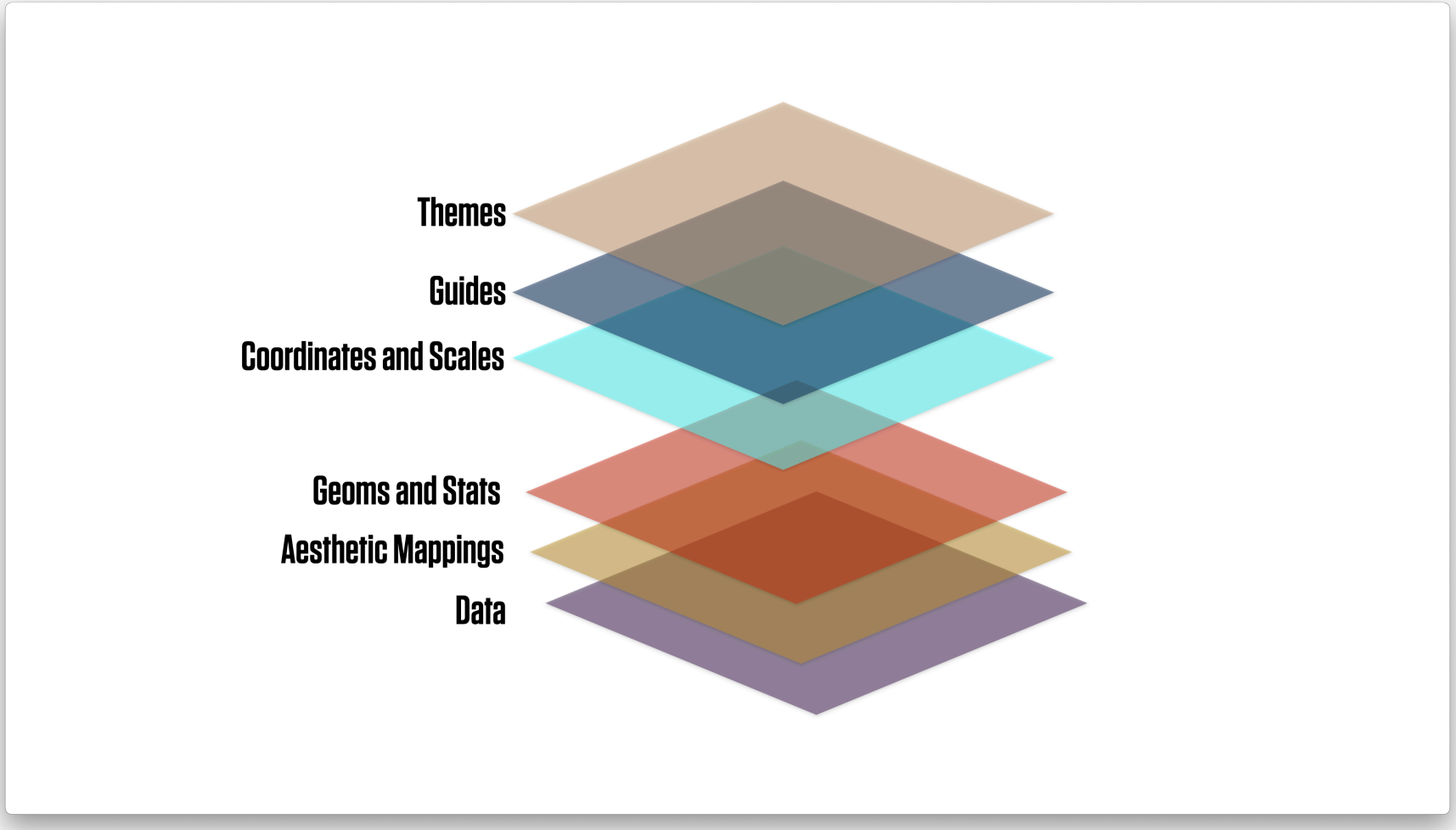
Categorical Gaps: Adverse Credit Events



Minimum Only Late Fee Over Limit

Layer to copmare

X-Ray Vision



Seeing through it

Themes

Themes ...

are controlled by the `theme()` function

can be bundled into functions of their own, like `theme_bw()` or `theme_minimal()`

can be set for the duration of a file or project with `theme_set()`

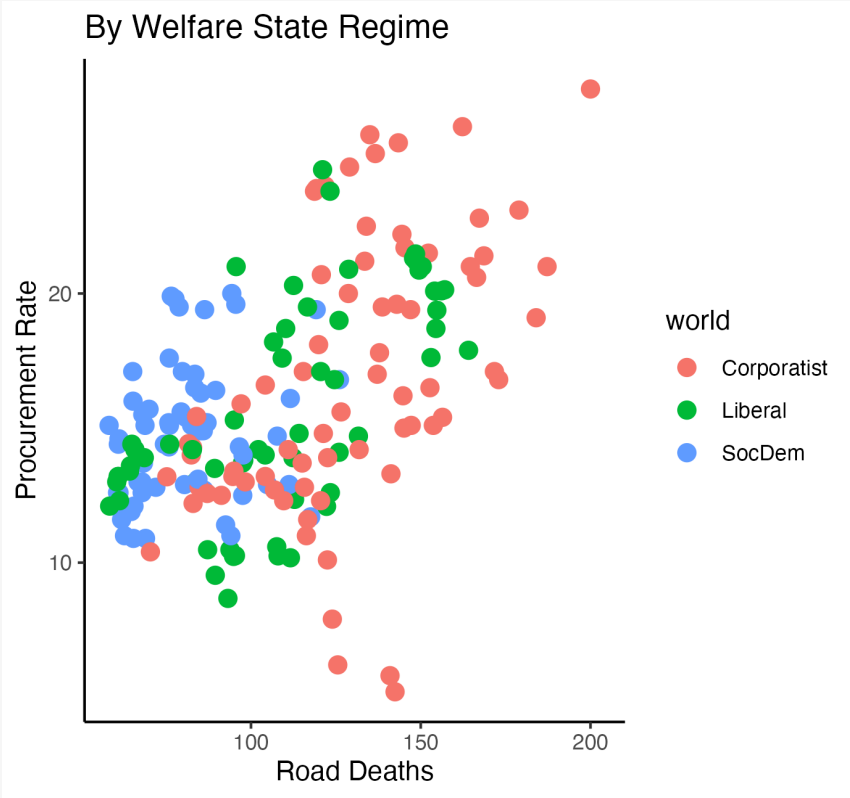
make changes that are applied *additively*

and most importantly ...

Thematic elements do not represent data directly

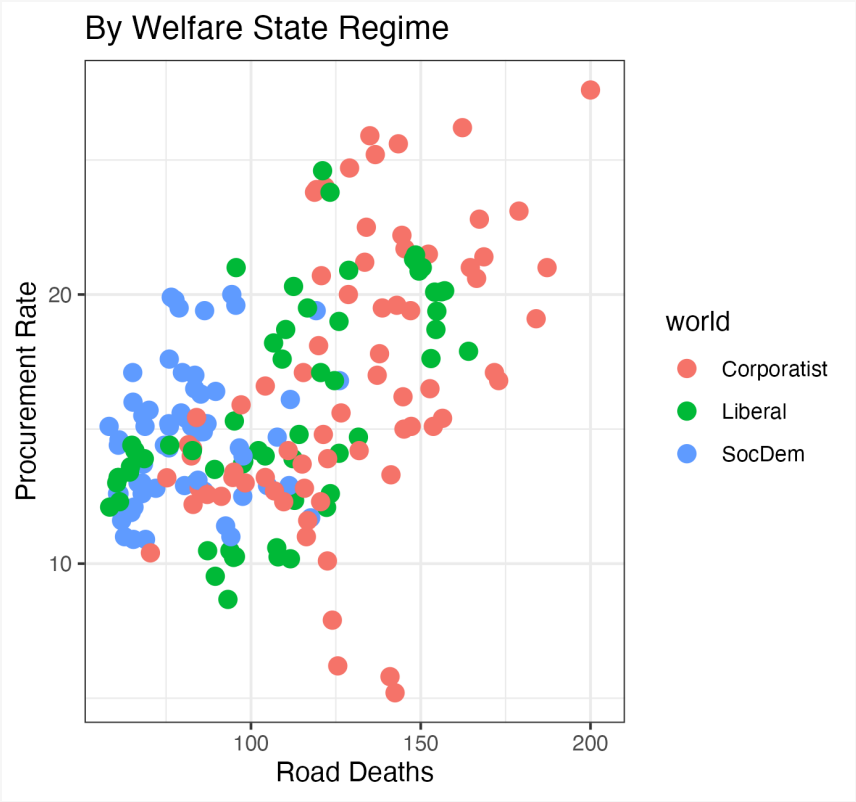
Make a plot

```
p ← organdata ▷
  drop_na(world) ▷
  ggplot(mapping = aes(x = roads, y = donors,
                       color = world)) +
  geom_point(size = 3) +
  labs(x = "Road Deaths",
       y = "Procurement Rate",
       title = "By Welfare State Regime")

p
```



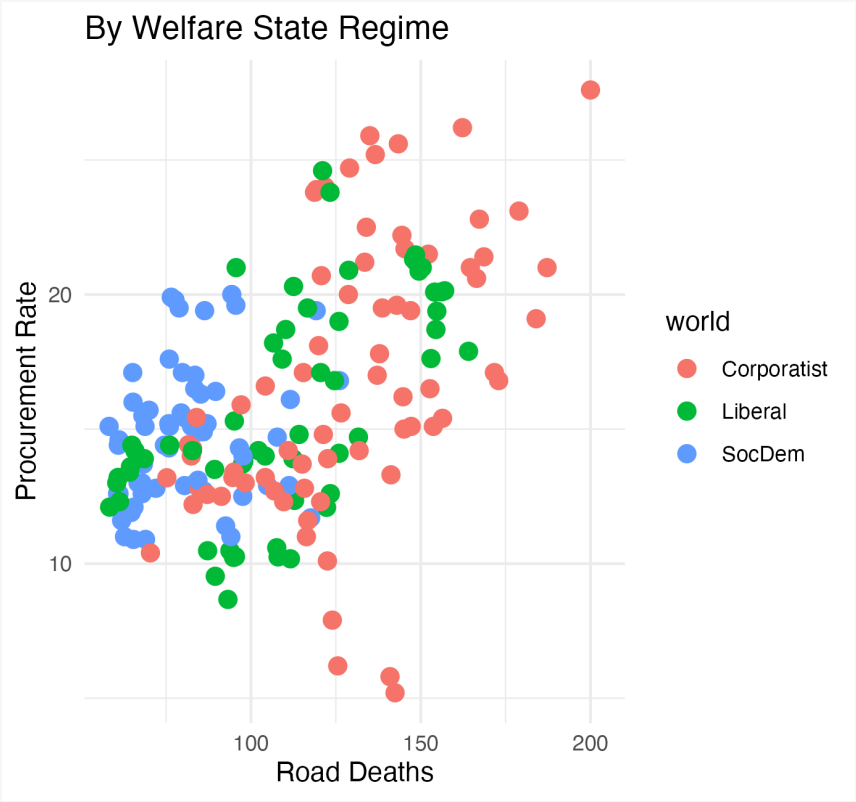
Add a theme ... theme_bw()

```
p + theme_bw()
```



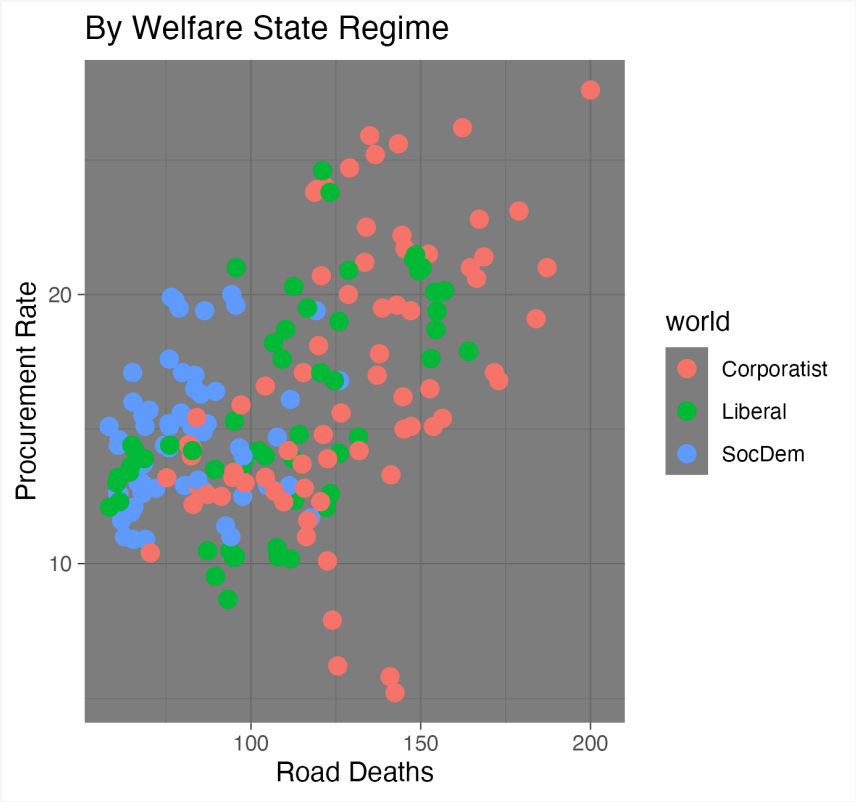
Add a theme ... `theme_minimal()`

```
p + theme_minimal()
```



Add a theme ... theme_dark()

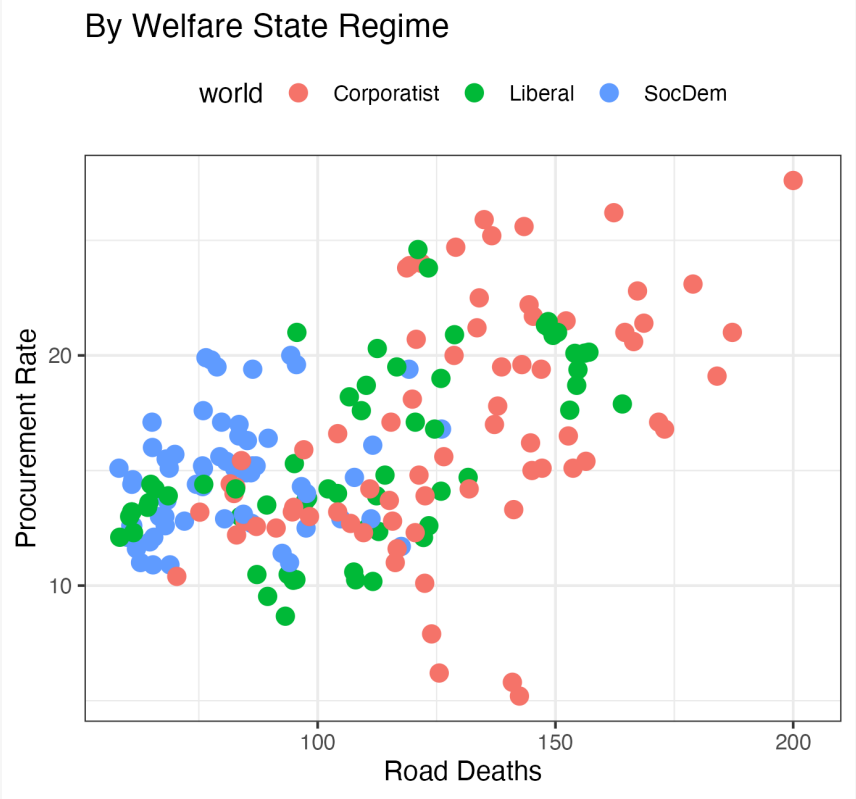
```
p + theme_dark()
```



Adjust with the `theme()` function

None of this directly touches the parts of the plot that are representing your data—i.e. the visual parts that are mapped to a variable, and thus have a scale. Adjusting those is the job of the `scale_` and `guide()` functions.

```
p + theme_bw() +
  theme(legend.position = "top")
```



There are *many* theme elements

line rect text title aspect.ratio

axis.title axis.title.x axis.title.x.top axis.title.x.bottom
axis.title.y axis.title.y.left axis.title.y.right axis.text
axis.text.x axis.text.x.top axis.text.x.bottom axis.text.y
axis.text.y.left axis.text.y.right axis.ticks axis.ticks.x
axis.ticks.x.top axis.ticks.x.bottom axis.ticks.y
axis.ticks.y.left axis.ticks.y.right axis.ticks.length
axis.ticks.length.x axis.ticks.length.x.top
axis.ticks.length.x.bottom axis.ticks.length.y
axis.ticks.length.y.left axis.ticks.length.y.right axis.line
axis.line.x axis.line.x.top axis.line.x.bottom axis.line.y
axis.line.y.left axis.line.y.right

legend.background legend.margin legend.spacing
legend.spacing.x legend.spacing.y legend.key
legend.key.size legend.key.height legend.key.width
legend.text legend.text.align legend.title
legend.title.align legend.position legend.direction
legend.justification legend.box legend.box.just
legend.box.margin legend.box.background
legend.box.spacing

panel.background panel.border panel.spacing
panel.spacing.x panel.spacing.y panel.grid
panel.grid.major panel.grid.minor panel.grid.major.x
panel.grid.major.y panel.grid.minor.x panel.grid.minor.y
panel.ontop plot.background

plot.title plot.title.position plot.subtitle plot.caption
plot.caption.position plot.tag plot.tag.position
plot.margin

strip.background strip.background.x strip.background.y
strip.placement strip.text strip.text.x strip.text.y
strip.switch.pad.grid strip.switch.pad.wrap

But they are structured

line rect text title aspect.ratio

axis.title axis.title.x axis.title.x.top axis.title.x.bottom
axis.title.y axis.title.y.left axis.title.y.right axis.text
axis.text.x axis.text.x.top axis.text.x.bottom axis.text.y
axis.text.y.left axis.text.y.right axis.ticks axis.ticks.x
axis.ticks.x.top axis.ticks.x.bottom axis.ticks.y
axis.ticks.y.left axis.ticks.y.right axis.ticks.length
axis.ticks.length.x axis.ticks.length.x.top
axis.ticks.length.x.bottom axis.ticks.length.y
axis.ticks.length.y.left axis.ticks.length.y.right axis.line
axis.line.x axis.line.x.top axis.line.x.bottom axis.line.y
axis.line.y.left axis.line.y.right

legend.background legend.margin legend.spacing
legend.spacing.x legend.spacing.y legend.key
legend.key.size legend.key.height legend.key.width
legend.text legend.text.align legend.title
legend.title.align legend.position legend.direction
legend.justification legend.box legend.box.just
legend.box.margin legend.box.background
legend.box.spacing

panel.background panel.border panel.spacing
panel.spacing.x panel.spacing.y panel.grid
panel.grid.major panel.grid.minor panel.grid.major.x
panel.grid.major.y panel.grid.minor.x panel.grid.minor.y
panel.ontop

plot.background plot.title plot.title.position plot.subtitle
plot.caption plot.caption.position plot.tag
plot.tag.position plot.margin

strip.background strip.background.x strip.background.y
strip.placement strip.text strip.text.x strip.text.y
strip.switch.pad.grid strip.switch.pad.wrap

And *inherit*

line rect text title aspect.ratio

axis.title axis.title.x axis.title.x.top axis.title.x.bottom axis.title.y axis.title.y.left axis.title.y.right **axis.text** axis.text.x axis.text.x.top axis.text.x.bottom axis.text.y axis.text.y.left axis.text.y.right **axis.ticks** axis.ticks.x axis.ticks.x.top axis.ticks.x.bottom axis.ticks.y axis.ticks.y.left axis.ticks.y.right axis.ticks.length axis.ticks.length.x axis.ticks.length.x.top axis.ticks.length.x.bottom axis.ticks.length.y axis.ticks.length.y.left axis.ticks.length.y.right **axis.line** axis.line.x axis.line.x.top axis.line.x.bottom axis.line.y axis.line.y.left axis.line.y.right

legend.background legend.margin legend.spacing legend.spacing.x legend.spacing.y **legend.key** legend.key.size legend.key.height legend.key.width **legend.text** legend.text.align **legend.title** legend.title.align **legend.position legend.direction legend.justification legend.box** legend.box.just legend.box.margin legend.box.background legend.box.spacing

panel.background panel.border panel.spacing panel.spacing.x panel.spacing.y **panel.grid** panel.grid.major panel.grid.minor panel.grid.major.x panel.grid.major.y panel.grid.minor.x panel.grid.minor.y **panel.ontop**

plot.background plot.title plot.title.position **plot.subtitle plot.caption** plot.caption.position **plot.tag** plot.tag.position **plot.margin**

strip.background strip.background.x strip.background.y **strip.placement strip.text** strip.text.x strip.text.y **strip.switch.pad.grid strip.switch.pad.wrap**

Two kinds of adjustment

It's a single setting.

E.g, `legend.position` can be `"none"`, `"left"`, `"right"`, `"bottom"`, or `"top"`

Hence, e.g., `theme(legend.position = "top")`, which we have seen several times. Similarly for e.g. `legend.direction` (can be "horizontal" or "vertical").

It's a component of the plot that might be styled in several ways. E.g., The text on the axes, or the lines in the plot panel.

If the latter ...

If adjusting a thematic element ask...

Where on the plot is it?

If adjusting a thematic element ask...

Where on the plot is it?

Is it part of an *axis*, part of the *panel*, the *strip* (facet title) box, or the *legend*? This will help you find the name of the thing you want to adjust.

If adjusting a thematic element ask...

Where on the plot is it?

Is it part of an *axis*, part of the *panel*, the *strip* (facet title) box, or the *legend*? This will help you find the name of the thing you want to adjust.

E.g. “I want to adjust the text for the markings on the x-axis”

You want axis.ticks.x

E.g. “I want to adjust the styling of the main y-axis grid lines inside the plot”

You want panel.grid.major.y

If adjusting a thematic element, ask…

What *kind* of element is it?

If adjusting a thematic element, ask…

What *kind* of element is it?

Is it *text*, or a *line*, or a *rectangle*?

This will tell you what function to use to make the adjustment to the named element.

If adjusting a thematic element, ask…

What *kind* of element is it?

Is it *text*, or a *line*, or a *rectangle*?

This will tell you what function to use to make the adjustment to the named element.

If it's text, adjust the element with `element_text()`

If adjusting a thematic element, ask…

What *kind* of element is it?

Is it *text*, or a *line*, or a *rectangle*?

This will tell you what function to use to make the adjustment to the named element.

If it's text, adjust the element with `element_text()`

If it's a line, adjust it with `element_line()`

If adjusting a thematic element, ask...

What *kind* of element is it?

Is it *text*, or a *line*, or a *rectangle*?

This will tell you what function to use to make the adjustment to the named element.

If it's text, adjust the element with `element_text()`

If it's a line, adjust it with `element_line()`

If it's a rectangle, with `element_rect()`

If adjusting a thematic element, ask...

What *kind* of element is it?

Is it *text*, or a *line*, or a *rectangle*?

This will tell you what function to use to make the adjustment to the named element.

If it's text, adjust the element with `element_text()`

If it's a line, adjust it with `element_line()`

If it's a rectangle, with `element_rect()`

If you want to *fully turn off* an element, use `element_blank()`

For example ...

“I want to adjust the styling of the plot title”

For example ...

“I want to adjust the styling of the plot title”

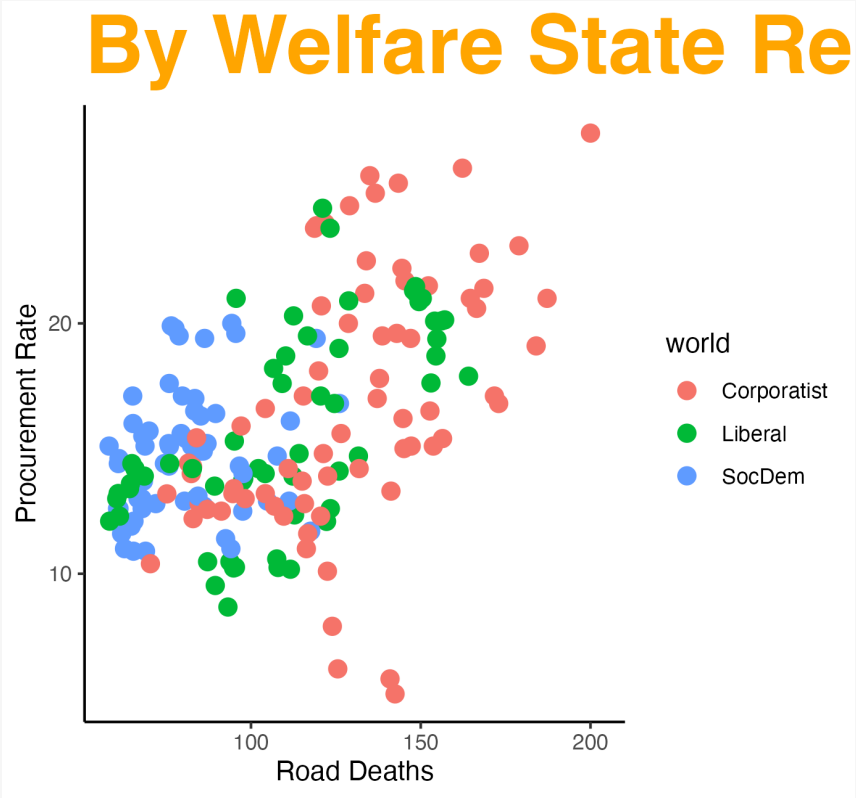
The relevant element is `plot.title`.

It’s *text*.

Inside the theme function, adjust it with `element_text()`.

For example ...

```
p + theme(plot.title =
            element_text(size = rel(3),
                         face = "bold",
                         color = "orange"))
```



For example ...

“I want to adjust y axis grid lines on the plot”

For example ...

“I want to adjust y axis grid lines on the plot”

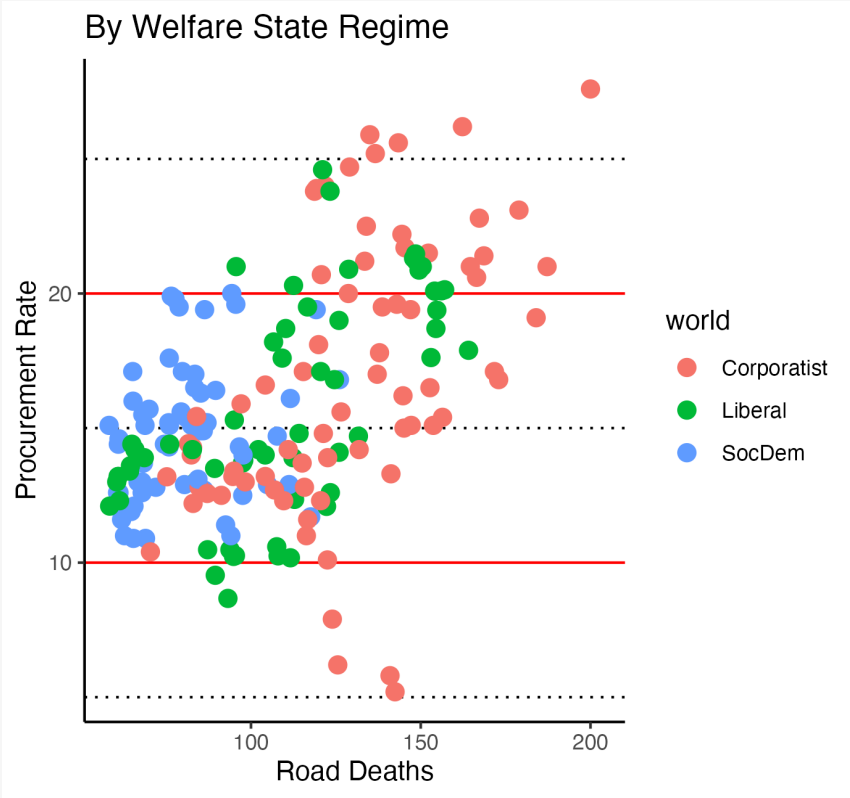
The relevant elements are `panel.grid.major.y` and `panel.grid.minor.y`.

These are *lines*.

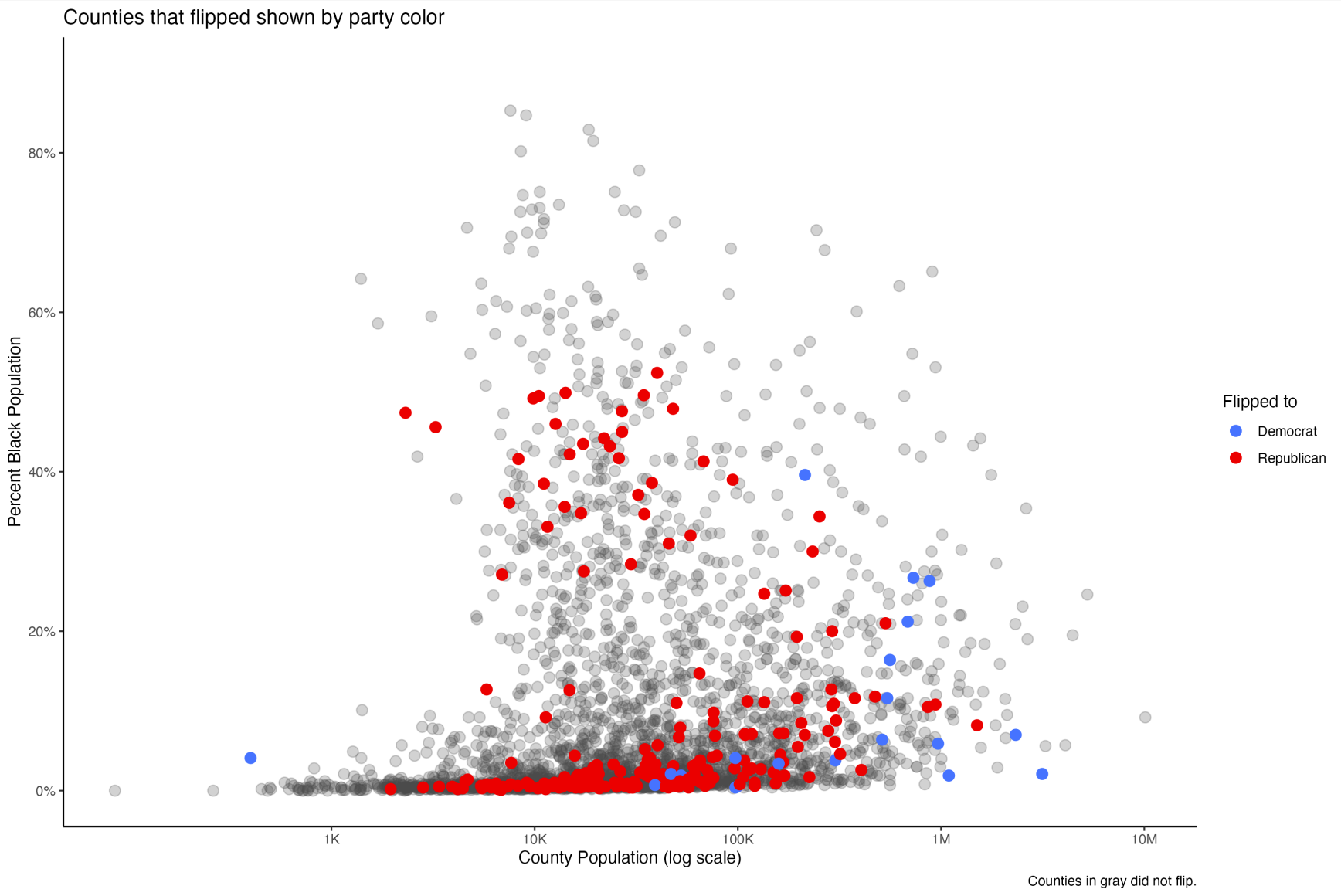
Inside the theme function, adjust it with `element_line()`.

For example ...

```
p + theme(panel.grid.major.y =
            element_line(color = "red"),
          panel.grid.minor.y =
            element_line(color = "black",
                         linetype = "dotted"))
```



The ggthemes package

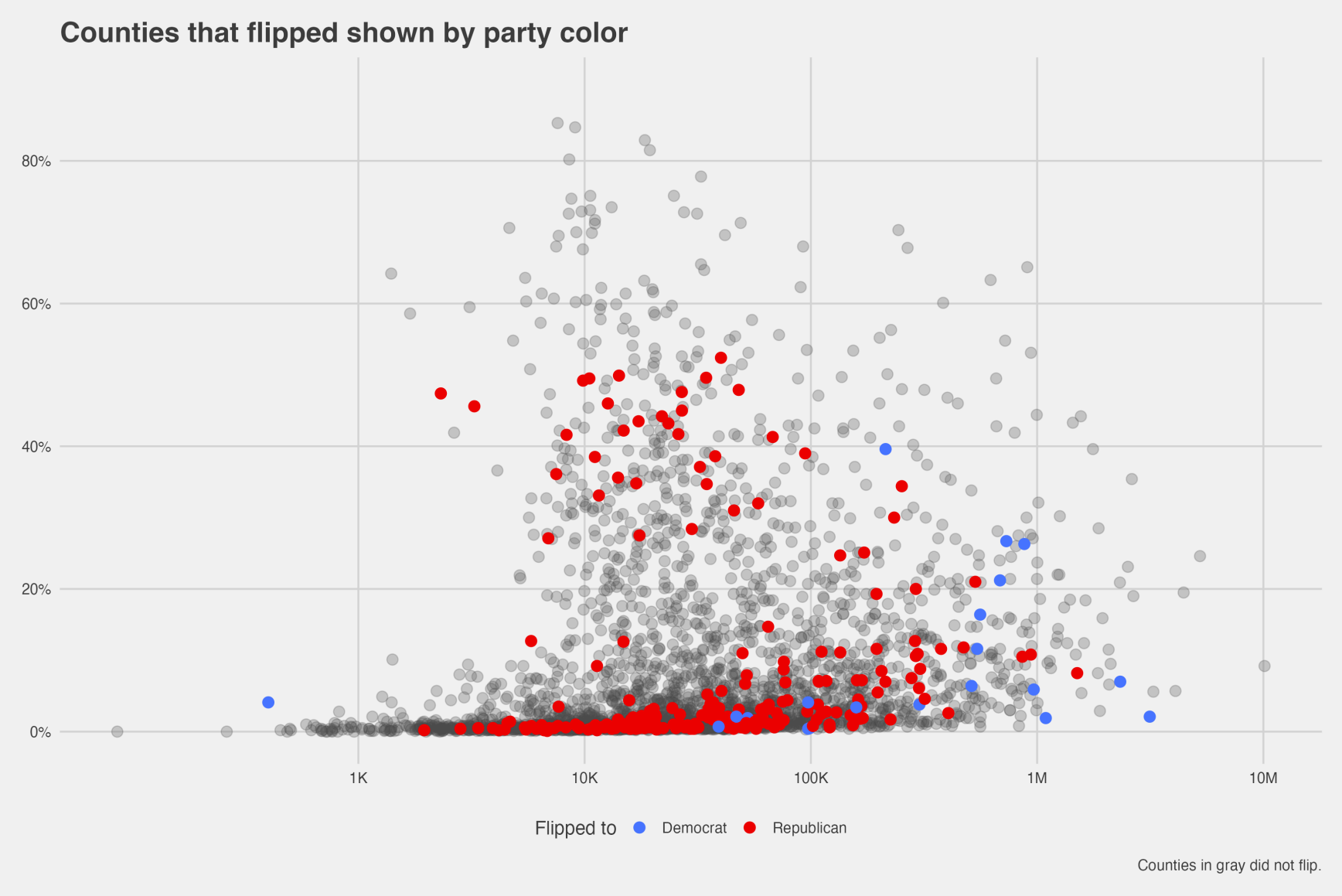


We made this earlier. Here it is in a default theme.

Theming a plot

```
library(ggthemes)
theme_set(theme_fivethirtyeight())
```

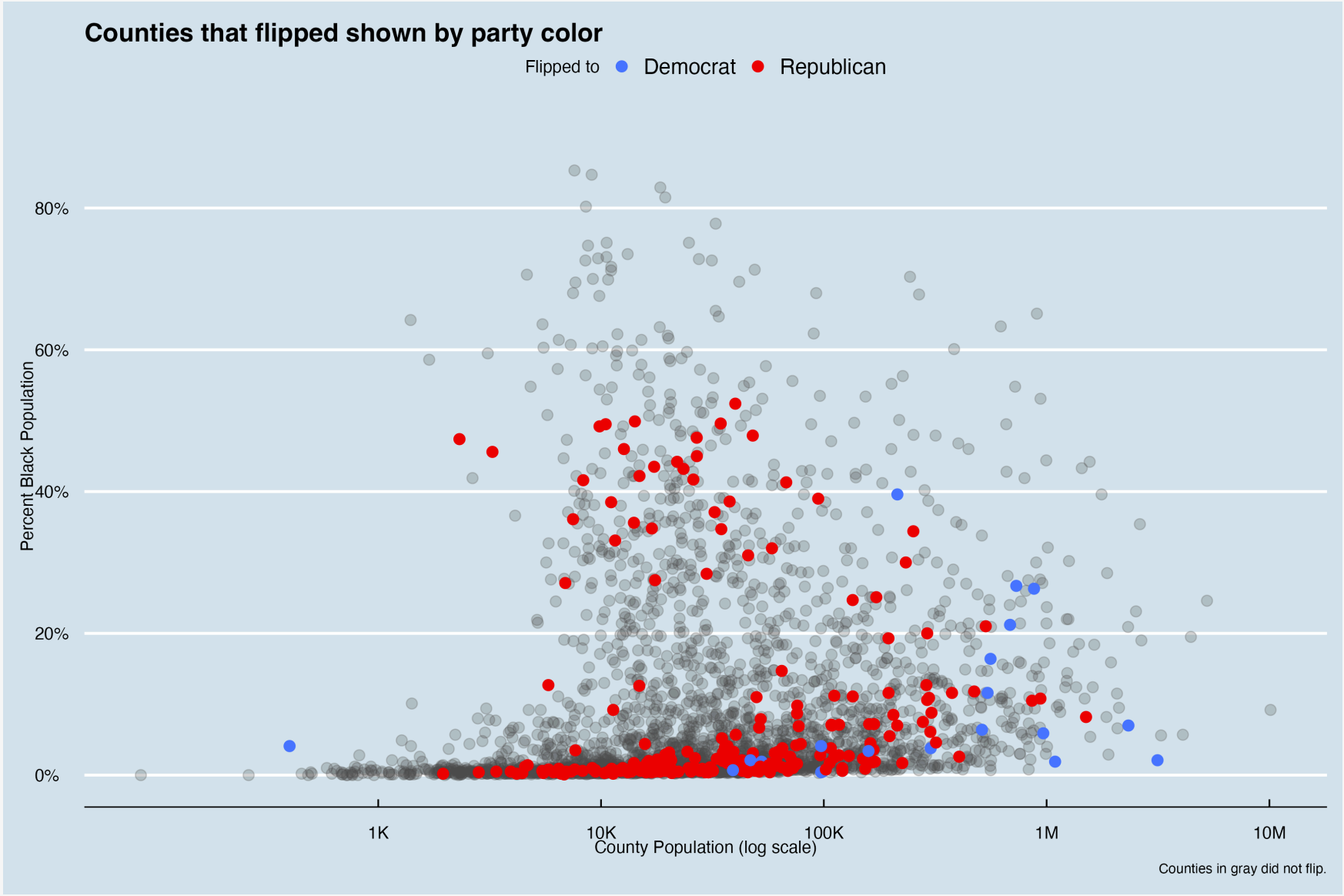
See how the full function call goes inside `theme_set()`, including the parentheses, because we are actually running that function to set all the elements.



Calling the object now draws the plot with the thematic elements added.

Theming a plot

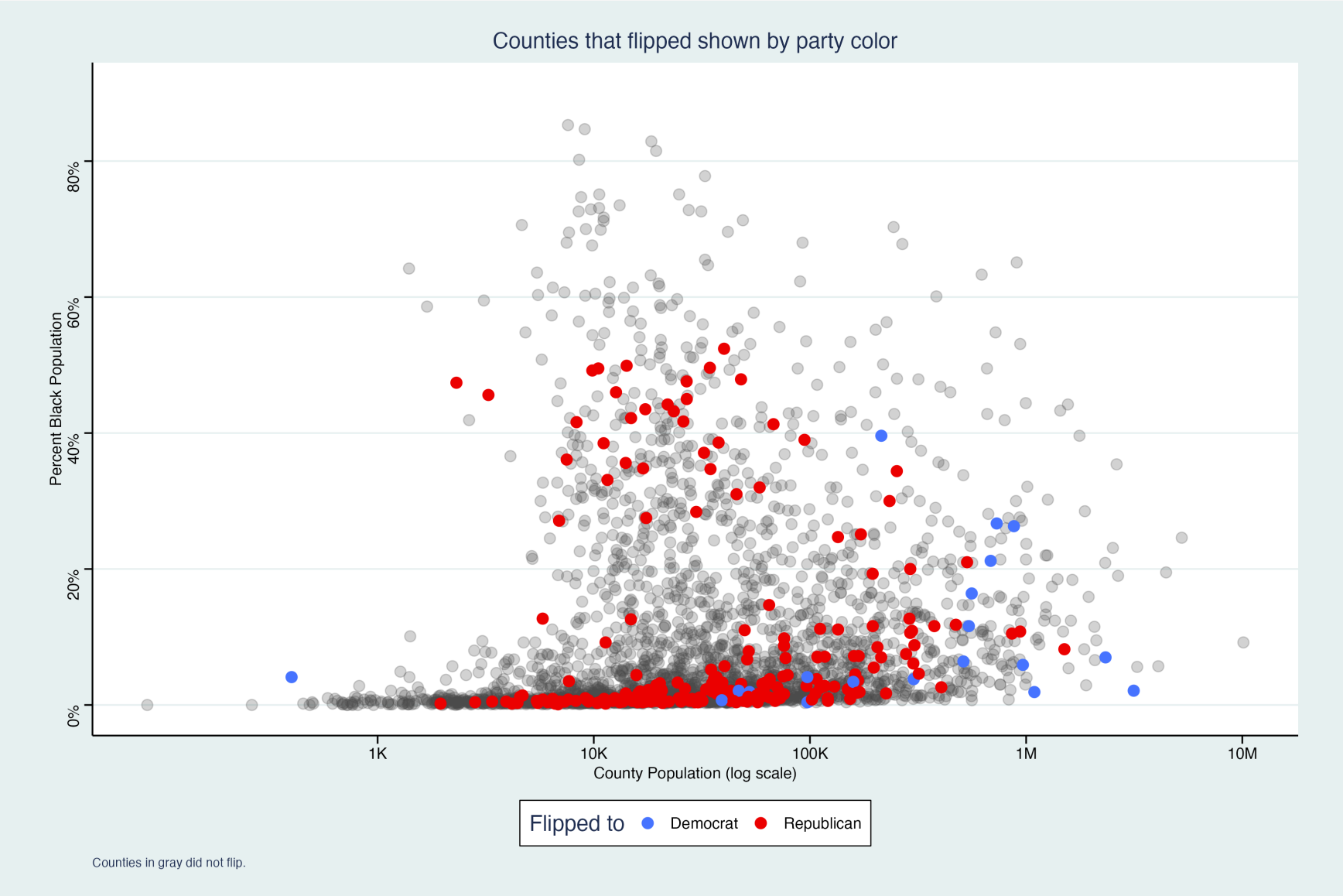
```
theme_set(theme_economist())
```



Calling the object now draws the plot with the thematic elements added.

This seems morally wrong

```
theme_set(theme_stata())
```



Why would you do this to yourself?

Themes are just functions

E.g. the built-in `theme_bw()` modifies `theme_grey()` with a bunch of additional options, expressed in a `theme()` call:

```
theme_bw
```

```
function (base_size = 11, base_family = "", base_line_size = base_size/22,
    base_rect_size = base_size/22)
{
    theme_grey(base_size = base_size, base_family = base_family,
        base_line_size = base_line_size, base_rect_size = base_rect_size) %+replace%
        theme(panel.background = element_rect(fill = "white",
            colour = NA), panel.border = element_rect(fill = NA,
            colour = "grey20"), panel.grid = element_line(colour = "grey92"),
            panel.grid.minor = element_line(linewidth = rel(0.5)),
            strip.background = element_rect(fill = "grey85",
                colour = "grey20"), complete = TRUE)
}
<bytecode: 0x16f368070>
<environment: namespace:ggplot2>
```

but with the special `%+replace%` operator. You can use `+` if you like. The difference (read the help for `theme_get()`) is that `%+replace%` swaps out the *entire* element being updated, whereas `+` does not.

You can make themes

To make a theme yourself, you can begin entirely from scratch or you can adapt an existing theme (like `theme_bw()` does. For example:

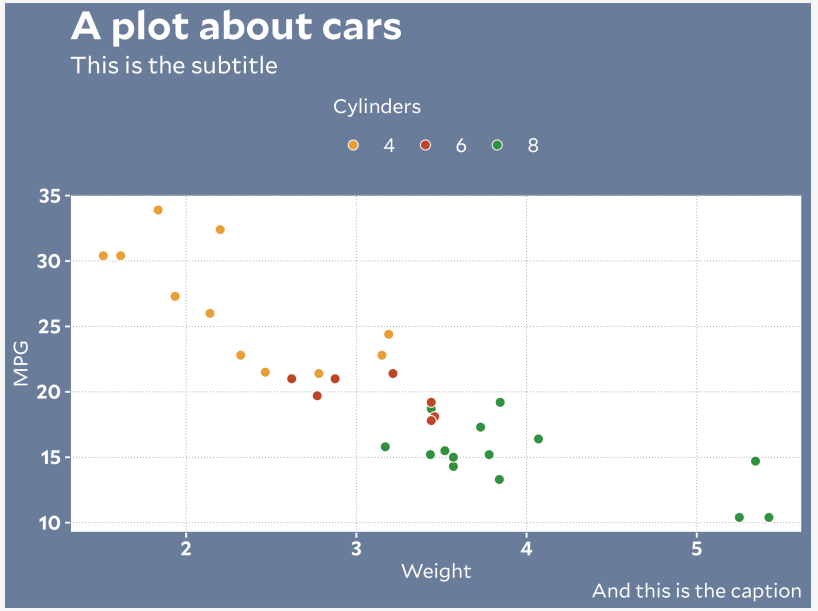
```
theme_rice ← function(base_size = 14,
                      base_family = "",
                      base_line_size = base_size/22,
                      base_rect_size = base_size/22) {

  ## See https://brand.rice.edu/colors/
  riceblue1 ← "#00205B"
  riceblue2 ← "#6D7D9C"
  ricegray1 ← "#686B6C"
  ricegray2 ← "#B0B2B2"
  textcolor ← "#FFFFFF"
  half_line ← base_size / 2

  ggplot2::theme_minimal(base_size = base_size, base_family = base_family,
        base_line_size = base_line_size, base_rect_size = base_rect_size) %+replace%
    theme(
      axis.text = element_text(color = textcolor,
                               face = "bold"),
      axis.ticks = element_line(color = textcolor),
      axis.title = element_text(color = textcolor),
      plot.title = element_text(color = textcolor,
                                face = "bold",
                                size = rel(2),
                                # left-justified text
```

Base Plot

```
p ← ggplot(data = mtcars,
           mapping = aes(x = wt,
                         y = mpg,
                         fill = factor(cyl))) +
  geom_point(size = rel(3),
             shape = 21, color = "white") +
  labs(x = "Weight", y = "MPG", fill = "Cylinders",
       title = "A plot about cars",
       subtitle = "This is the subtitle",
       caption = "And this is the caption")
```

Now we can do, e.g.,

```
rice_accents ← c("#E9A139", "#C04829", "#359245", "#362E52", "#005B50")

p + scale_fill_manual(values = rice_accents) +
  theme_rice(base_family = "Mallory") # See https://brand.rice.edu
```



This is very rough, and I daresay ill-advised. But you see how much control you have over how things look. Also it serves as a reminder that you will likely have to do quite a bit of detail-oriented work if you want a properly good custom theme. Choose one and stick with it.