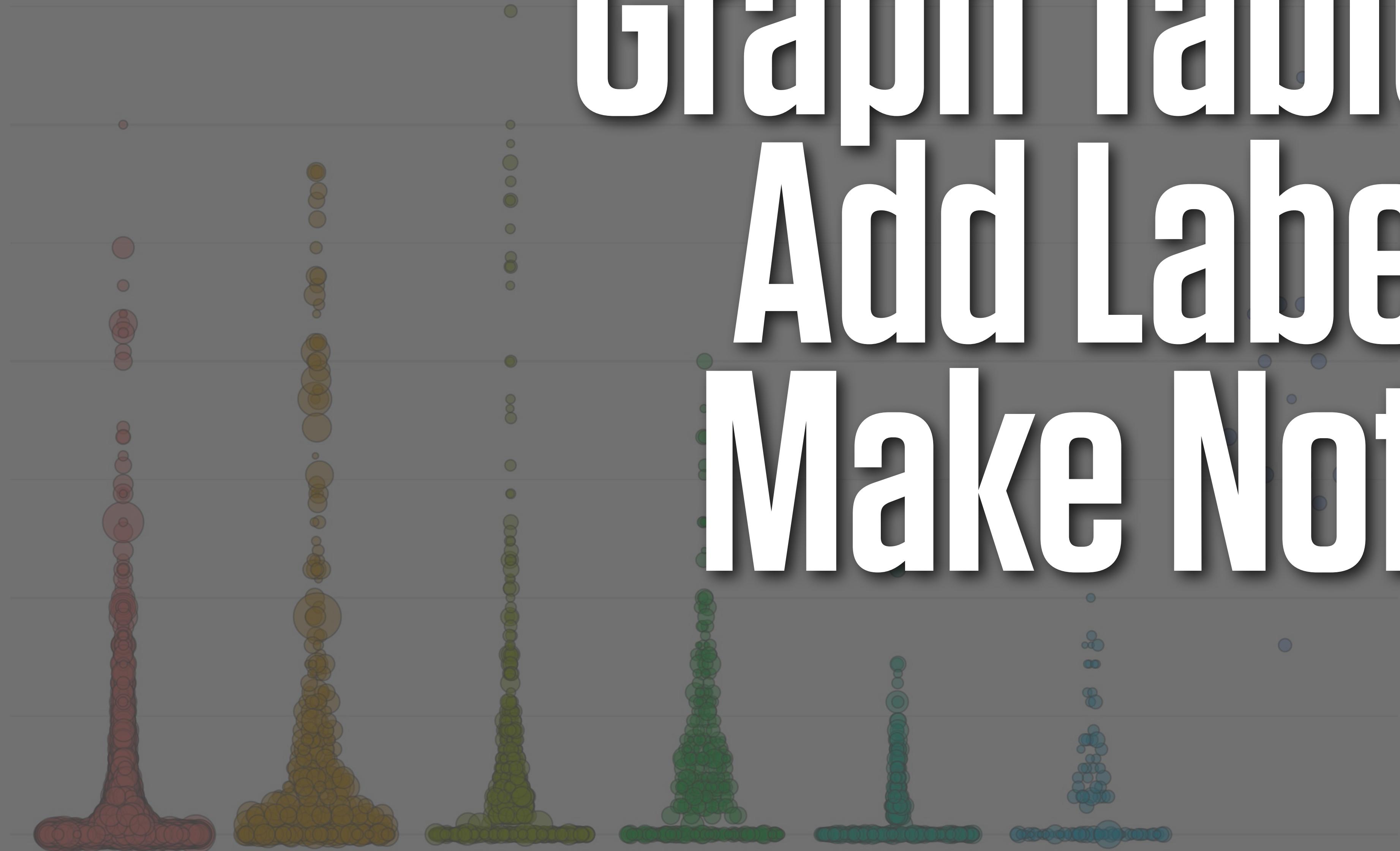
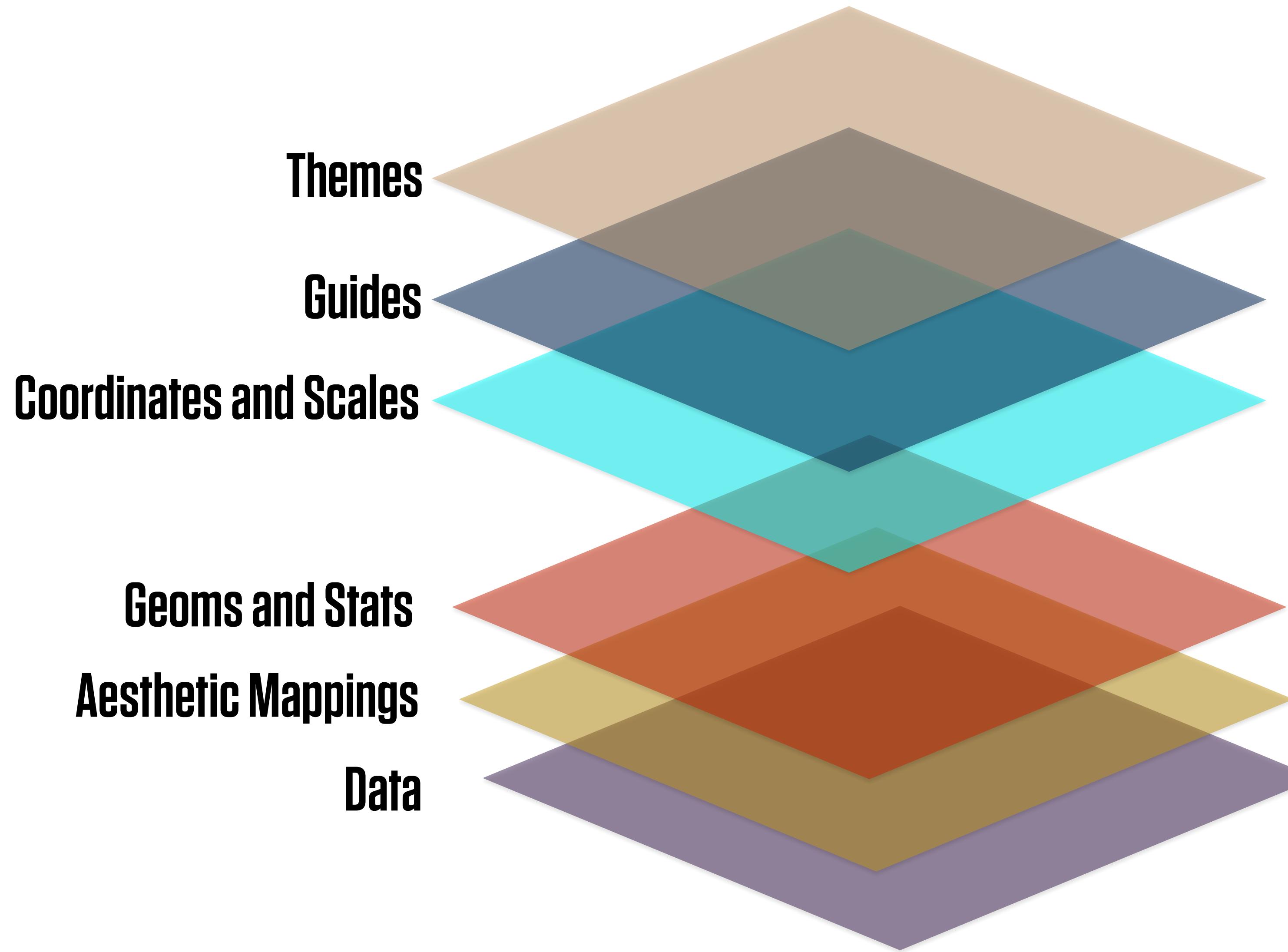


Data Visualization



Graph Tables, Add Labels, Make Notes





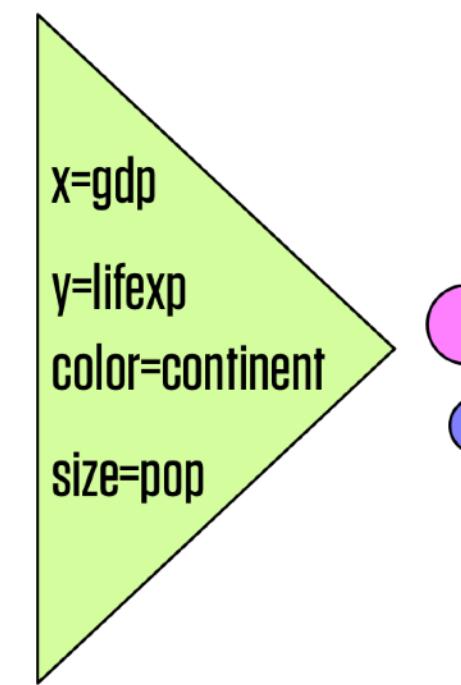
ggplot's FLOW OF ACTION

1. Tidy Data

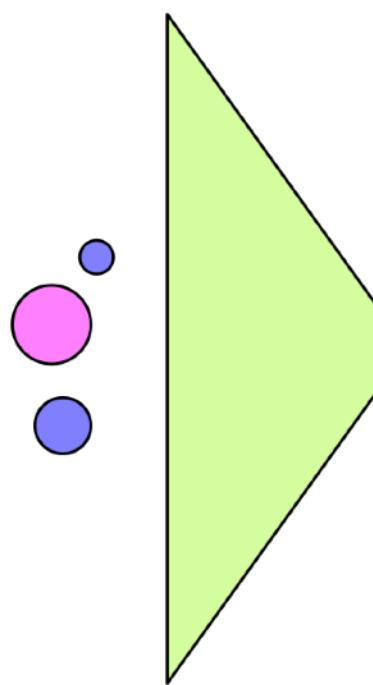
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

```
ggplot(data = gapminder, mapping =  
aes(x = gdp,  
y = lifespan,  
color = continent,  
size = pop))
```

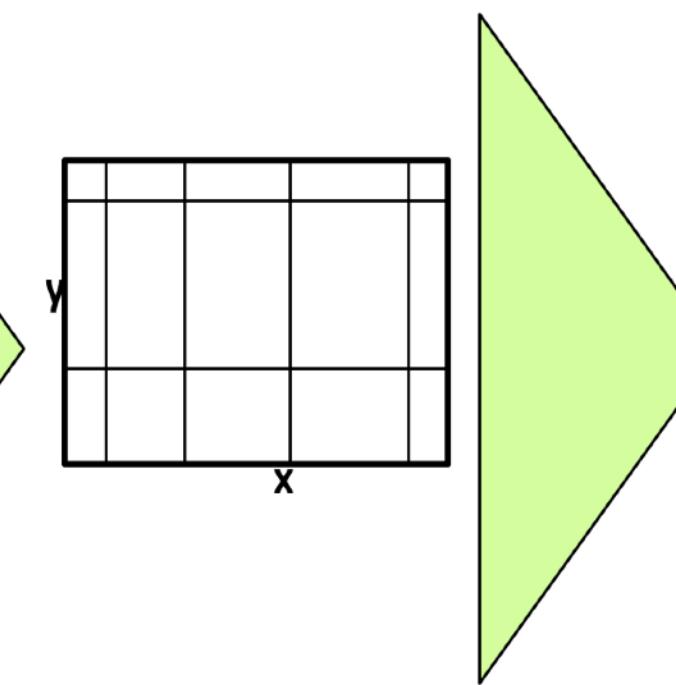
2. Mapping



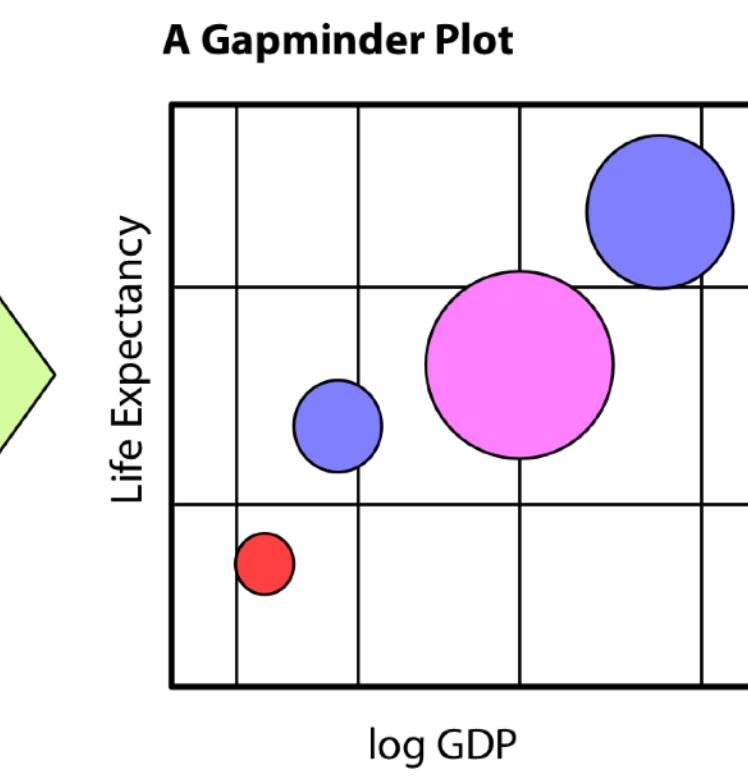
3. Geom



4. Co-ordinates, Scales

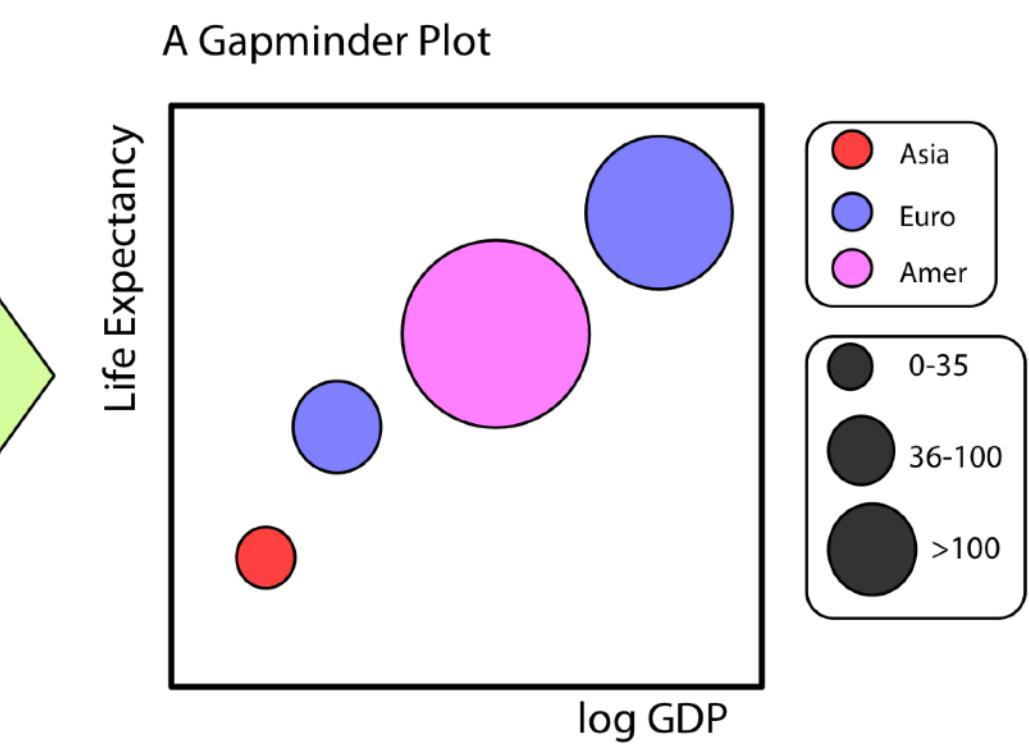


5. Labels & Guides



```
labs()  
guides()
```

6. Themes



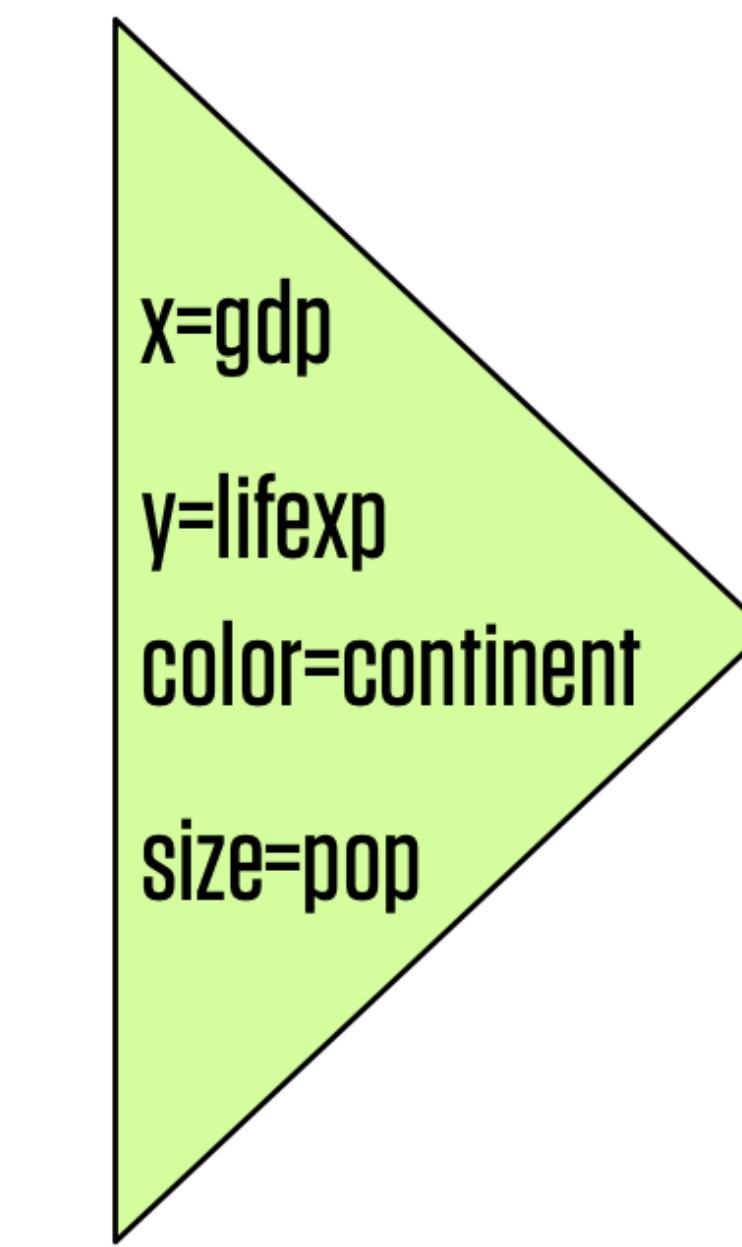
```
theme_minimal()
```

1. Tidy Data

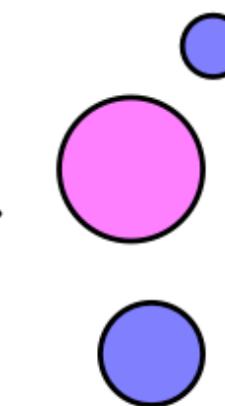
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

```
ggplot(data = gapminder, mapping =  
aes(x = gdp,  
y = lifespan,  
color = continent,  
size = pop))
```

2. Mapping

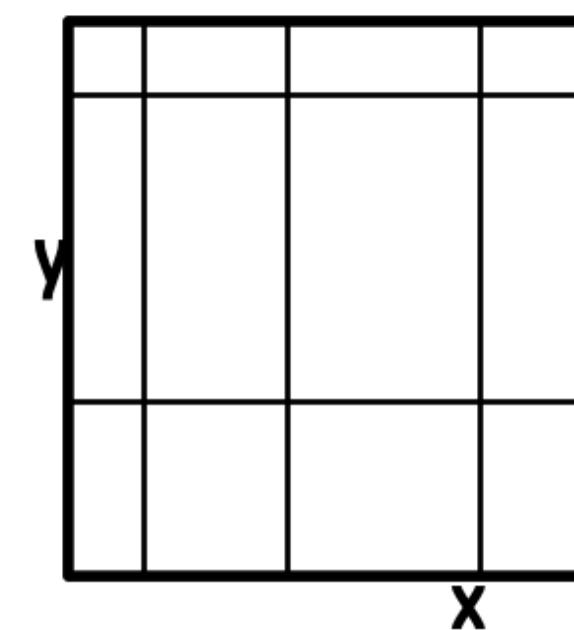


3. Geom



```
geom_point()
```

4. Co-Ordinates



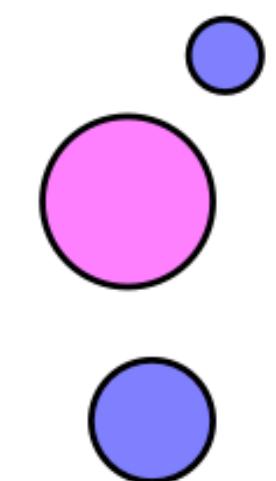
```
coord_cartes(  
scale_x_log
```

2. Mapping

continent
Euro
Amer
Euro
Asia

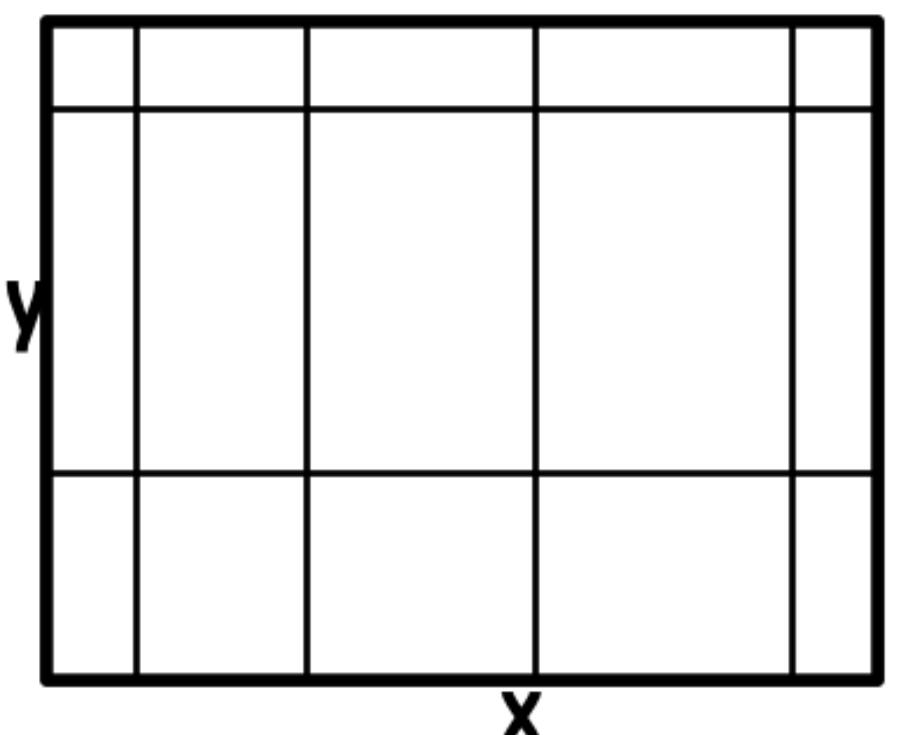
```
x=gdp  
y=lifexp  
color=continent  
size=pop
```

3. Geom



```
geom_point()
```

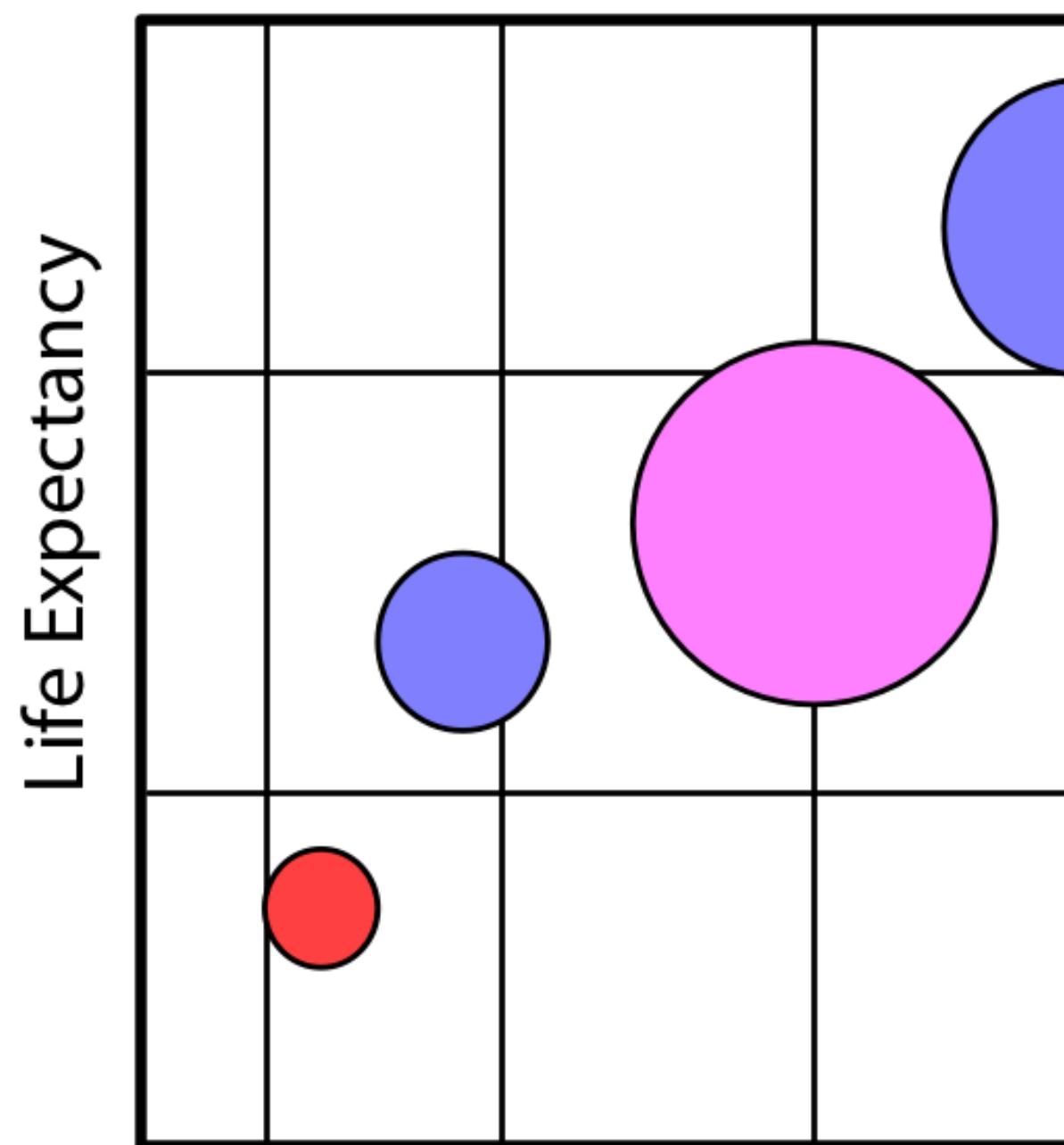
4. Co-ordinates, Scales



```
coord_cartesian()  
scale_x_log10()
```

5. Labels & Guides

A Gapminder Plot



```
= gapminder, mapping =  
aes(x = gdp,  
y = lifespan,
```

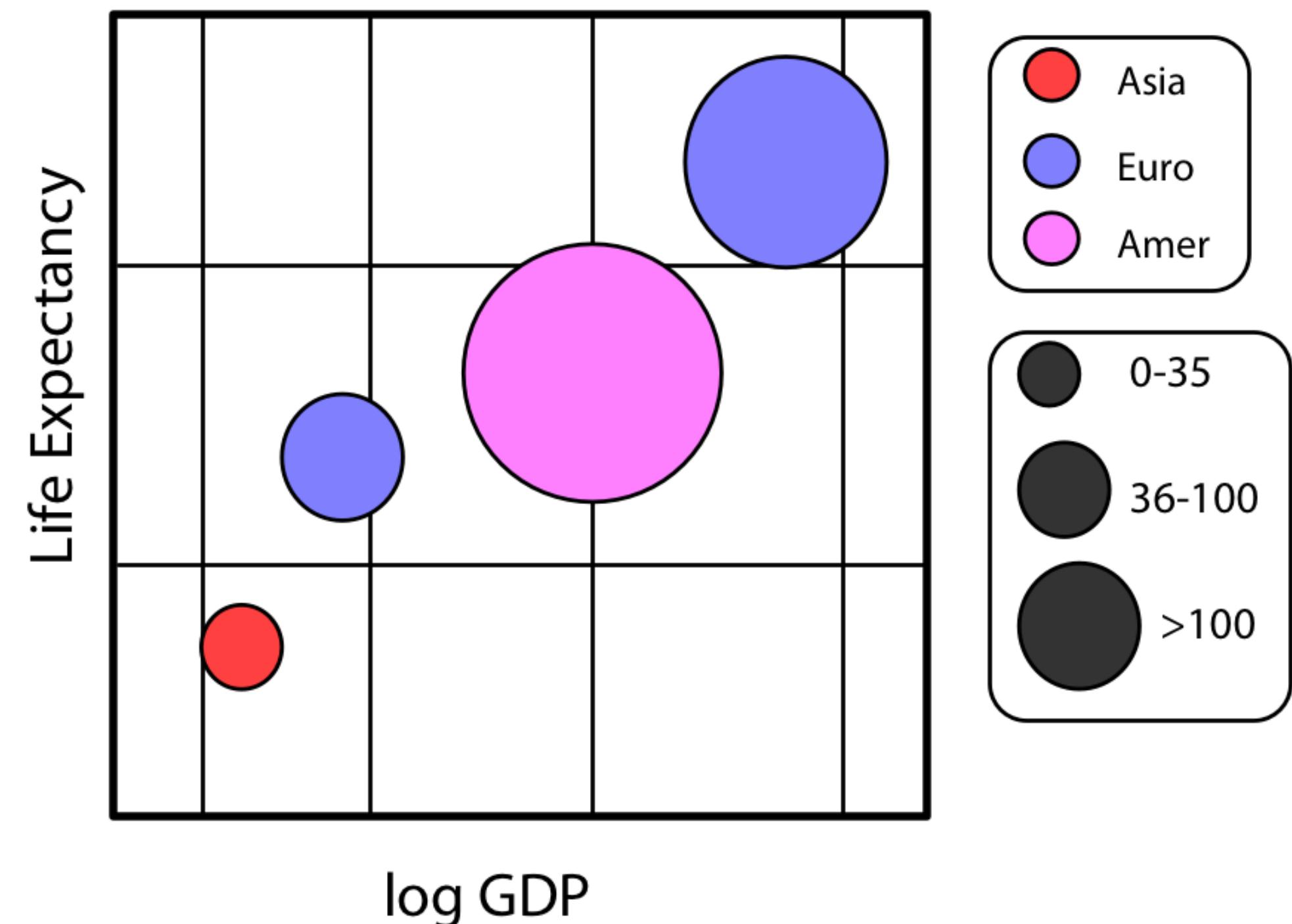
log GDP

labs()

guides()

5. Labels & Guides

A Gapminder Plot

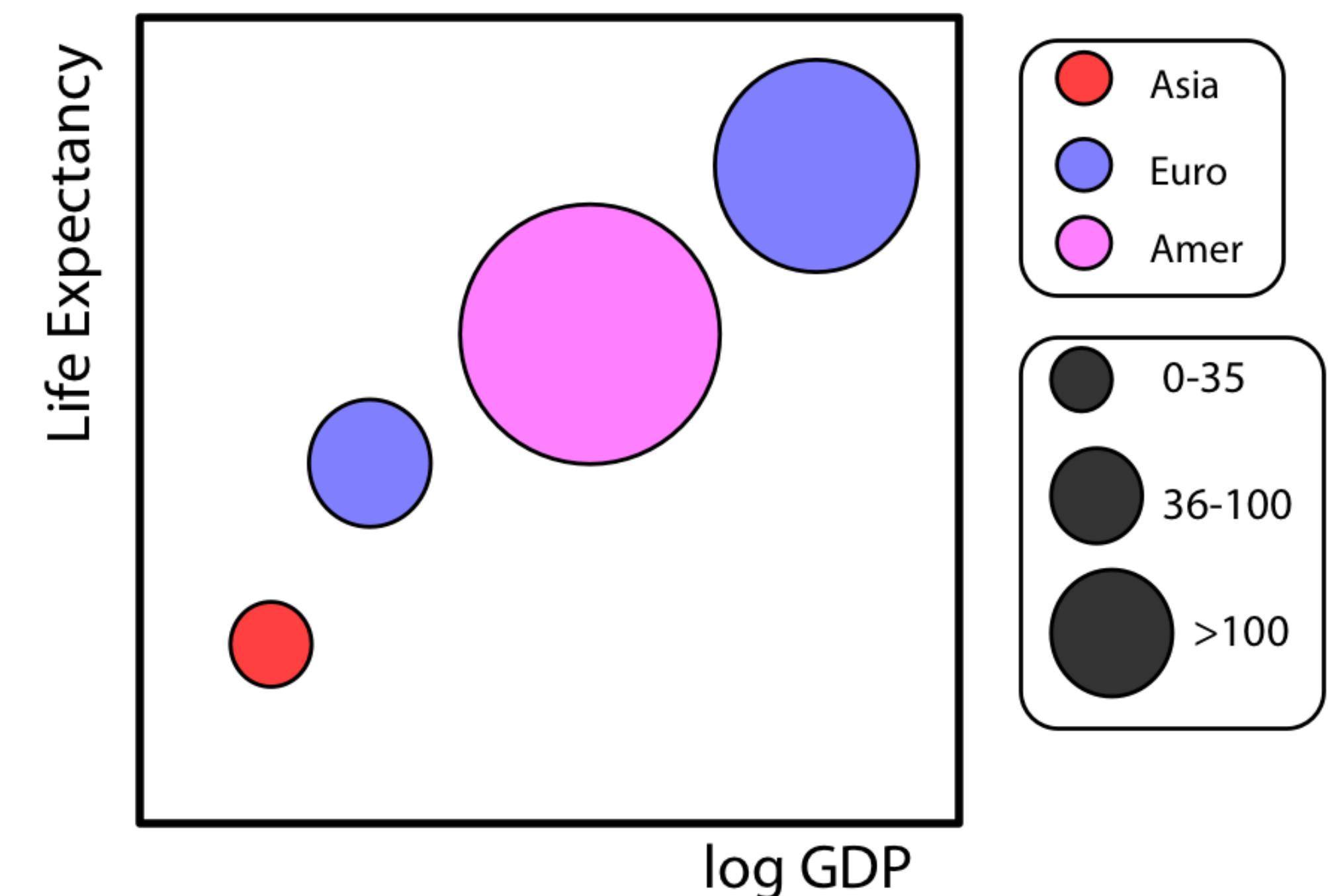


`labs()`

`guides()`

6. Themes

A Gapminder Plot



`theme_minimal()`

**TRANSFORM AND
SUMMARIZE FIRST
THEN SEND CLEAN
TABLES TO ggplot**

	Protestant	Catholic	Jewish	None	Other	NA
Northeast	11.5	25.0	52.9	18.1	17.6	5.6
Midwest	23.7	26.5	5.9	25.4	20.8	27.8
South	47.4	24.7	21.6	27.5	31.4	61.1
West	17.4	23.9	19.6	29.1	30.2	5.6
	100	100	100	100	100	100

	Protestant	Catholic	Jewish	None	Other	NA
Northeast	32.4	33.2	5.5	23.0	5.7	0.2
Midwest	46.8	24.7	0.4	22.6	4.7	0.7
South	61.8	15.2	1.0	16.2	4.8	1.0
West	37.7	24.5	1.6	28.5	7.6	0.2
	100	100	100	100	100	100

dplyr lets you
manipulate tables in
a series of steps,
or **pipeline**

Group the data at the level we want, such as "Religion by Region" or "Authors by Publications by Year".

group_by()

Filter or **Select** pieces of the data. This gets us the subset of the table we want to work on.

filter() rows
select() columns

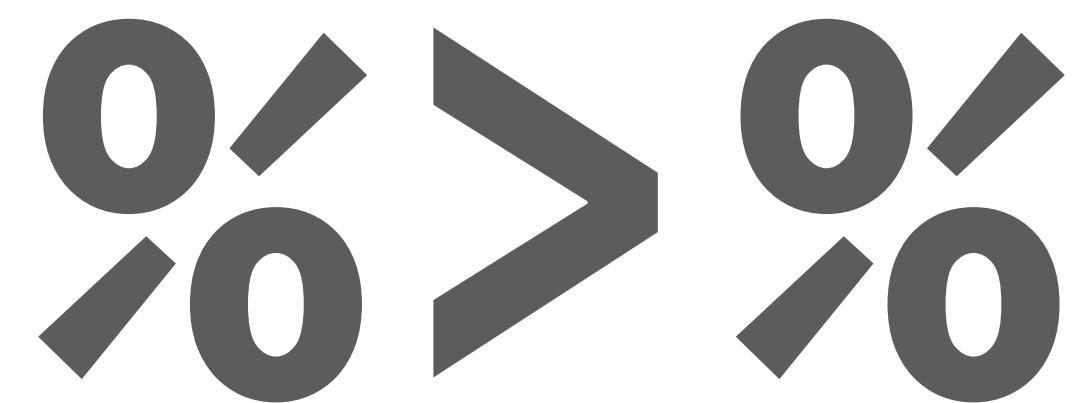
Mutate the data by creating new variables at the current level of grouping. Mutating adds new columns to the table.

mutate()

Summarize the grouped data. This creates new variables at a higher level of grouping. For example we might calculate means with mean() or counts with n(). This results in a smaller, summary table, which we might do more things with if we want.

summarize()

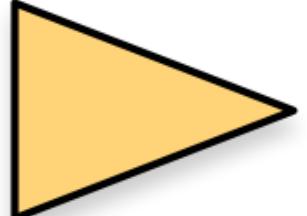
Create a pipeline of
transformations with
the pipe operator



Individual Level GSS Data Table

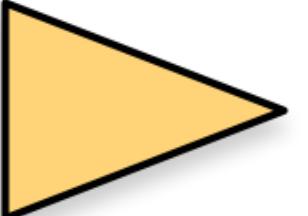
	id	bigregion	religion
1014		Midwest	Protestant
1544		South	Protestant
665		Northeast	None
1618		South	None
2115		West	Catholic
417		South	Protestant
2045		West	Protestant
1863		Northeast	Other
1884		Midwest	Christian
1628		South	Protestant

Summary Count of Religious Preferences by Census Region



bigregion	religion	N
Northeast	Protestant	123
Northeast	Catholic	149
Northeast	Jewish	15
Northeast	None	97
Northeast	Christian	14
Northeast	Other	31

Percent Religious Preferences by Census Region



bigregion	religion	pct
Northeast	Protestant	28.3
Northeast	Catholic	34.3
Northeast	Jewish	3.4
Northeast	None	22.3
Northeast	Christian	3.2
Northeast	Other	7.1

REORGANIZING TABLES WITH `dplyr`

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

```
rel_by_region <- gss_sm
```

```
> rel_by_region
# A tibble: 2,867 x 32
  year    id ballot   age childs  sibs degree race   sex   region income16 relig marital padeg madeg partyid polviews
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <fct>
1 2016     1     1    47      3      2 Bacheli... White Male New E... $170000... None Married Gradu... High... Indepe... Moderate
2 2016     2     2    61      0      3 High ... White Male New E... $50000 ... None Never ... Lt H... High... Ind,ne... Liberal
3 2016     3     3    72      2      3 Bacheli... White Male New E... $75000 ... Cath... Married High... Lt H... Not St... Conserv...
4 2016     4     1    43      4      3 High ... White Fema... New E... $170000... Cath... Married NA     High... Not St... Moderate
5 2016     5     3    55      2      2 Gradu... White Fema... New E... $170000... None Married Bacheli... High... Not St... Slightl...
6 2016     6     2    53      2      2 Junio... White Fema... New E... $60000 ... None Married NA     High... Not St... Slightl...
7 2016     7     1    50      2      2 High ... White Male New E... $170000... None Married High... High... Not St... Slightl...
8 2016     8     3    23      3      6 High ... Other Fema... Midd... $30000 ... Cath... Married Lt H... Lt H... Ind,ne... Slightl...
9 2016     9     1    45      3      5 High ... Black Male Midd... $60000 ... Prot... Married Lt H... Lt H... Strong... NA
10 2016    10     3    71      4      1 Junio... White Male Midd... $60000 ... None Divorc... High... High... Strong... Conserv...
# ... with 2,857 more rows, and 15 more variables: happy <fct>, partners <fct>, grass <fct>, zodiac <fct>, pres12 <dbl>,
# wtssall <dbl>, income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
# bigregion <fct>, partners_rc <fct>, obama <dbl>
```

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion)
```

```
> rel_by_region  
# A tibble: 2,867 x 32  
# Groups: bigregion, religion [24]  
  year    id ballot   age childs sibs degree race   sex   region income16 relig marital padeg madeg partyid polviews  
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>  
1 2016     1      1    47      3      2 Bach... White Male New E... $170000... None Married Grad... High... Indepe... Moderate  
2 2016     2      2    61      0      3 High ... White Male New E... $50000 ... None Never ... Lt H... High... Ind,ne... Liberal  
3 2016     3      3    72      2      3 Bach... White Male New E... $75000 ... Cath... Married High... Lt H... Not St... Conserv...  
4 2016     4      1    43      4      3 High ... White Fema... New E... $170000... Cath... Married NA High... Not St... Moderate  
5 2016     5      3    55      2      2 Gradu... White Fema... New E... $170000... None Married Bach... High... Not St... Slightl...  
6 2016     6      2    53      2      2 Junio... White Fema... New E... $60000 ... None Married NA High... Not St... Slightl...  
7 2016     7      1    50      2      2 High ... White Male New E... $170000... None Married High... High... Not St... Slightl...  
8 2016     8      3    23      3      6 High ... Other Fema... Middl... $30000 ... Cath... Married Lt H... Lt H... Ind,ne... Slightl...  
9 2016     9      1    45      3      5 High ... Black Male Middl... $60000 ... Prot... Married Lt H... Lt H... Strong... NA  
10 2016    10     3    71      4      1 Junio... White Male Middl... $60000 ... None Divorc... High... High... Strong... Conserv...  
# ... with 2,857 more rows, and 15 more variables: happy <fct>, partners <fct>, grass <fct>, zodiac <fct>, pres12 <dbl>,  
# wtssall <dbl>, income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,  
# bigregion <fct>, partners_rc <fct>, obama <dbl>
```

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n())
```

```
> rel_by_region  
# A tibble: 24 x 3  
# Groups:   bigregion [4]  
  bigregion religion     n  
  <fct>    <fct>     <int>  
1 Northeast Protestant  158  
2 Northeast Catholic   162  
3 Northeast Jewish     27  
4 Northeast None       112  
5 Northeast Other      28  
6 Northeast NA         1  
7 Midwest Protestant  325  
8 Midwest Catholic    172  
9 Midwest Jewish       3  
10 Midwest None        157  
# ... with 14 more rows
```

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

```
> rel_by_region  
# A tibble: 24 x 5  
# Groups:   bigregion [4]  
  bigregion religion     n    freq    pct  
  <fct>     <fct>   <int>    <dbl>   <dbl>  
1 Northeast Protestant  158  0.324    32.4  
2 Northeast Catholic   162  0.332    33.2  
3 Northeast Jewish     27  0.0553    5.5  
4 Northeast None       112  0.230    23  
5 Northeast Other      28  0.0574    5.7  
6 Northeast NA          1  0.00205   0.2  
7 Midwest Protestant   325  0.468    46.8  
8 Midwest Catholic    172  0.247    24.7  
9 Midwest Jewish       3  0.00432   0.4  
10 Midwest None        157  0.226   22.6  
# ... with 14 more rows
```

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

Objects in a pipeline carry
forward some assumptions
about context

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

Grouping with `group_by()` carries forward; summary calculations are applied to the innermost group

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

mutate() adds or modifies columns, it
doesn't change the grouping level or
the number of rows in the table

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

Notice how we can create variables
on the fly and use them immediately

rel_by_region

```
## Source: local data frame [24 x 5]
## Groups: bigregion [4]
##
## # A tibble: 24 x 5
##   bigregion religion     n      freq     pct
##   <fctr>     <fctr> <int>     <dbl>    <dbl>
## 1 Northeast Protestant  158 0.32377049 32.4
## 2 Northeast Catholic   162 0.33196721 33.2
## 3 Northeast Jewish     27  0.05532787 5.5
## 4 Northeast None       112 0.22950820 23.0
## 5 Northeast Other      28  0.05737705 5.7
## 6 Northeast NA         1   0.00204918 0.2
## 7 Midwest Protestant  325 0.46762590 46.8
## 8 Midwest Catholic    172 0.24748201 24.7
## 9 Midwest Jewish       3   0.00431655 0.4
## 10 Midwest None        157 0.22589928 22.6
## # ... with 14 more rows
```

```
rel_by_region <- gss_sm %>%  
  group_by(bigregion, religion) %>%  
  summarize(n = n()) %>%  
  mutate(freq = n / sum(n),  
        pct = round((freq*100), 1))
```

Some Shorthand for this ...

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  summarize(n = n())
```

```
# A tibble: 24 x 3
# Groups: bigregion [4]
  bigregion religion      n
  <fct>     <fct>     <int>
1 Northeast Protestant    158
2 Northeast Catholic      162
3 Northeast Jewish        27
4 Northeast None          112
5 Northeast Other         28
6 Northeast NA             1
7 Midwest Protestant       325
8 Midwest Catholic         172
9 Midwest Jewish           3
10 Midwest None            157
# ... with 14 more rows
```

n()

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  tally()
```

```
# A tibble: 24 x 3
# Groups: bigregion [4]
  bigregion religion      n
  <fct>     <fct>     <int>
1 Northeast Protestant    158
2 Northeast Catholic      162
3 Northeast Jewish        27
4 Northeast None          112
5 Northeast Other         28
6 Northeast NA             1
7 Midwest Protestant       325
8 Midwest Catholic         172
9 Midwest Jewish           3
10 Midwest None            157
# ... with 14 more rows
```

tally()

```
gss_sm %>%
  count(bigregion, religion)
```

```
# A tibble: 24 x 3
  bigregion religion      n
  <fct>     <fct>     <int>
1 Northeast Protestant    158
2 Northeast Catholic      162
3 Northeast Jewish        27
4 Northeast None          112
5 Northeast Other         28
6 Northeast NA             1
7 Midwest Protestant       325
8 Midwest Catholic         172
9 Midwest Jewish           3
10 Midwest None            157
# ... with 14 more rows
```

count()

Use dplyr to create summary
tables, and **then** graph them

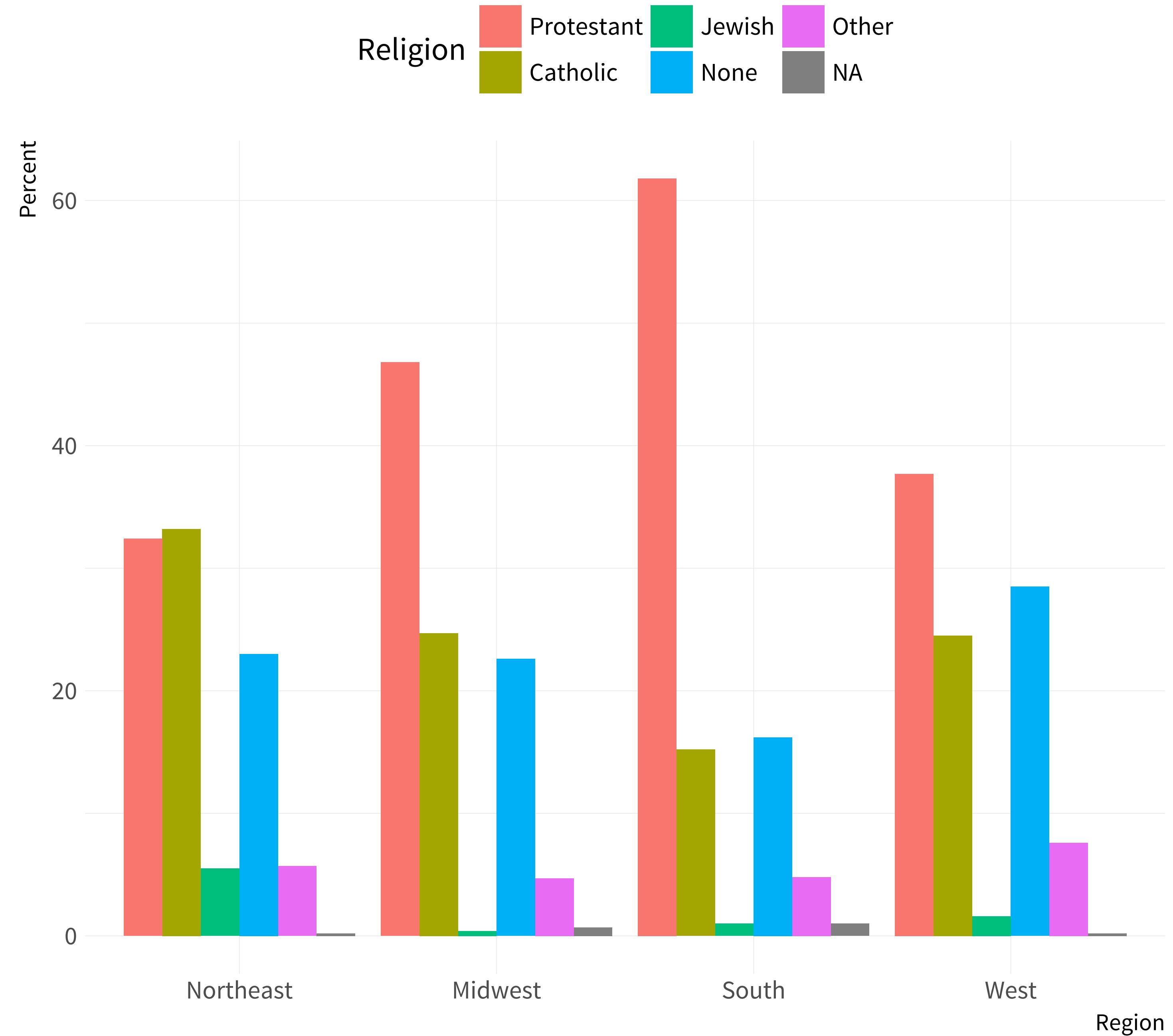
```
rel_by_region %>%  
  group_by(bigregion) %>%  
  summarize(total = sum(pct))
```

```
## # A tibble: 4 x 2  
##   bigregion total  
##   <fctr>    <dbl>  
## 1 Northeast  100.0  
## 2 Midwest   99.9  
## 3 South      100.0  
## 4 West       100.1
```

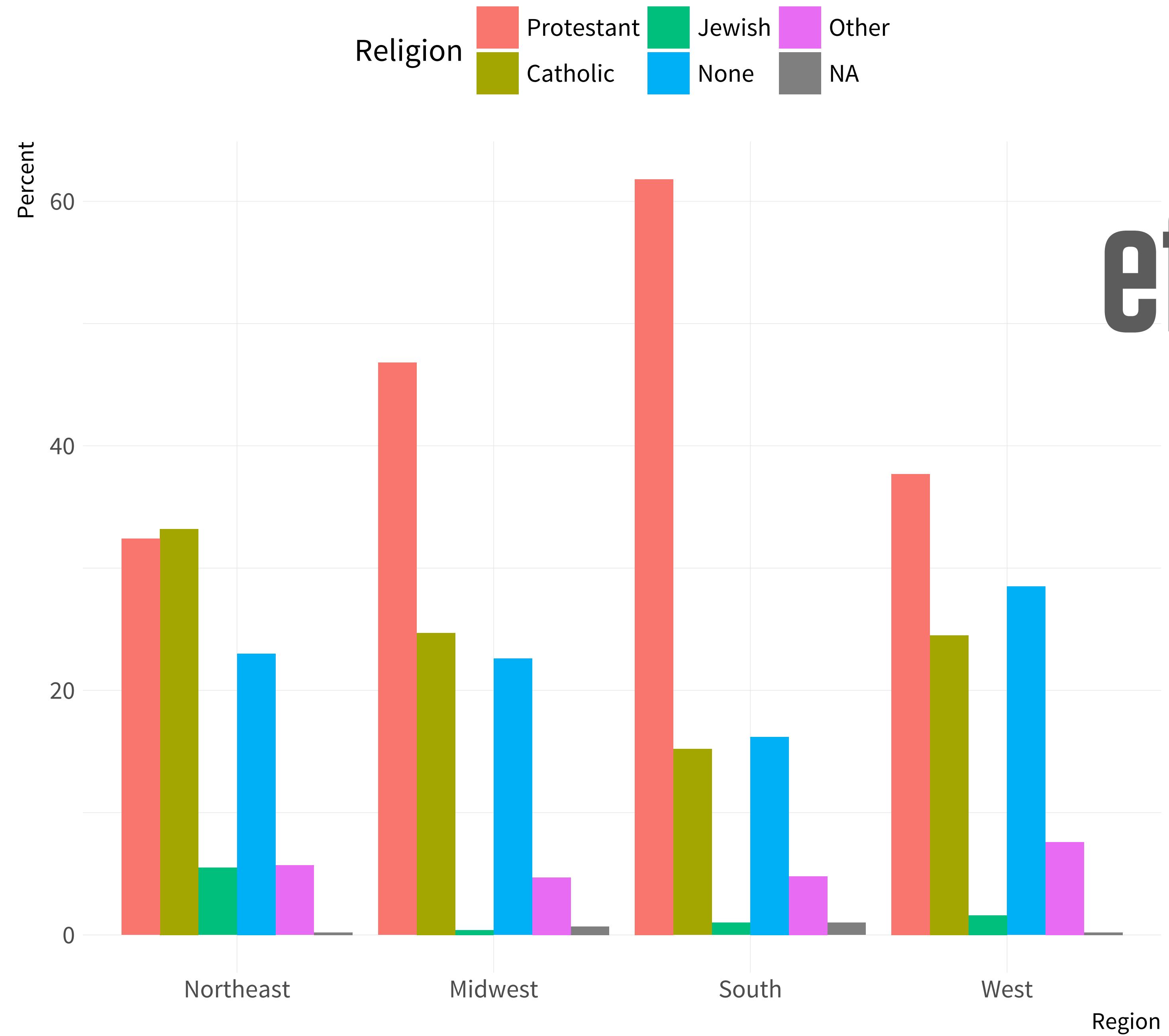
Pipelined tables are
easier to check for errors

```
p <- ggplot(data = rel_by_region,  
             mapping = aes(x = bigregion,  
                           y = pct,  
                           fill = religion))  
p + geom_col(position = "dodge") +  
  labs(x = "Region",  
        y = "Percent",  
        fill = "Religion") +  
  theme(legend.position = "top")
```

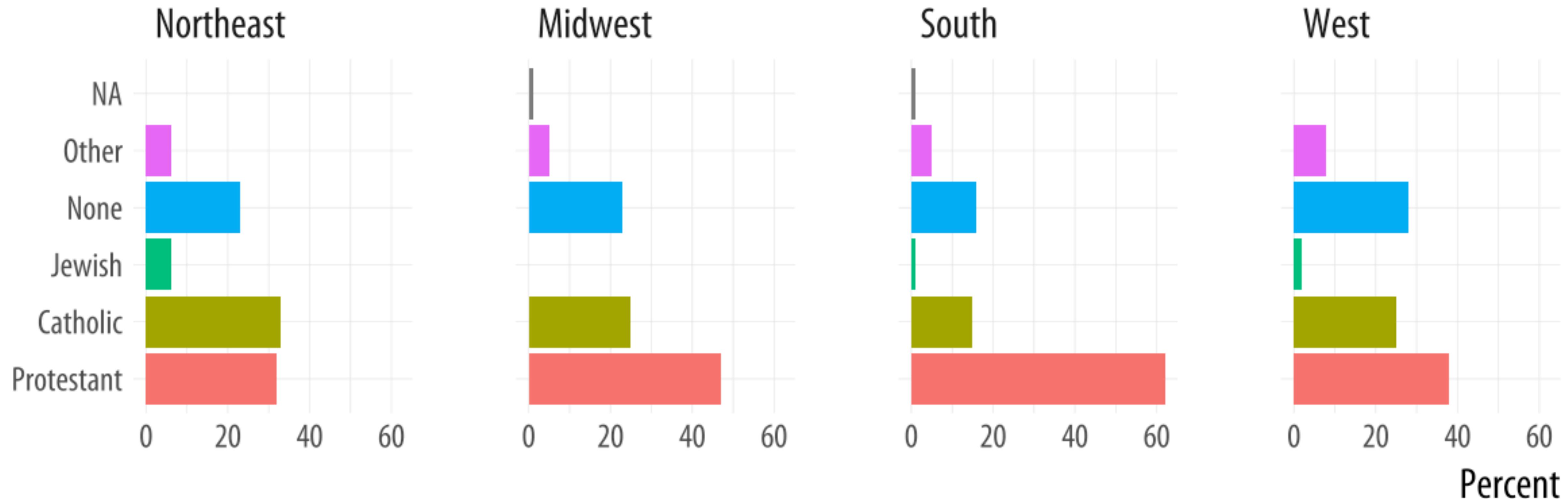
OK, back to making graphs ...



But is this an
effective graph?
Not Really!



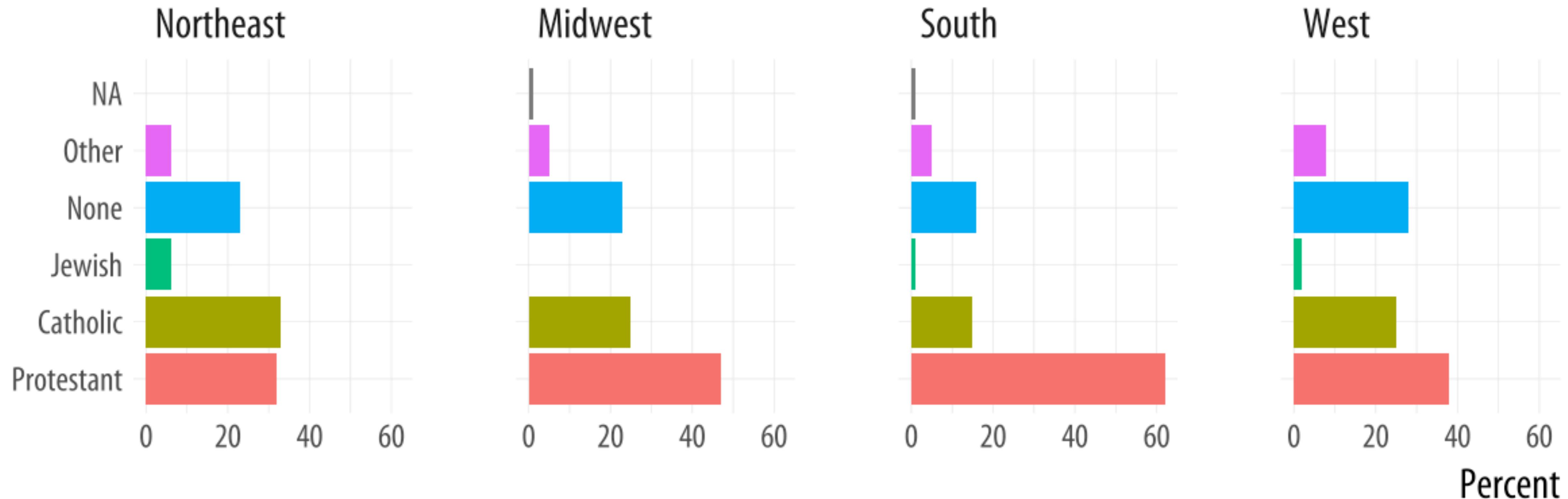
```
p <- ggplot(data = rel_by_region,  
             mapping = aes(x = religion, y = pct, fill = religion))  
p + geom_col(position = "dodge") +  
  labs(x = NULL, y = "Percent") +  
  guides(fill = FALSE) +  
  coord_flip() +  
  facet_wrap(~ bigregion, nrow = 1)
```



Try putting categories on the y-axis

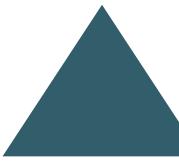
Try faceting variables instead of mapping them

Try to minimize the need for guides and legends

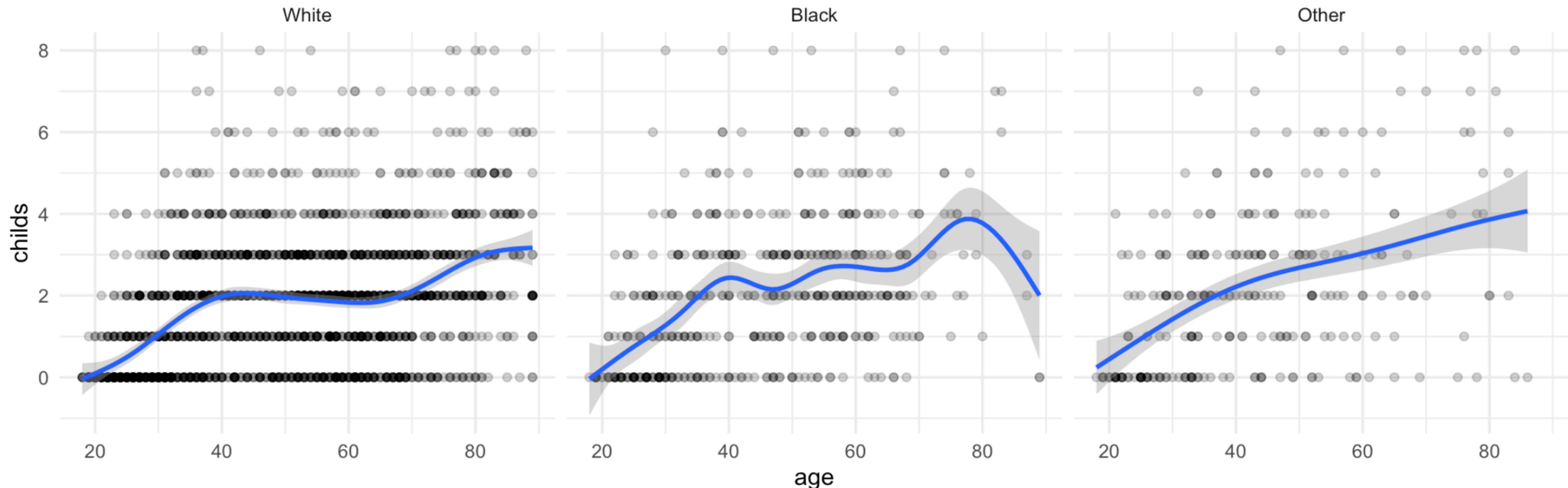


Two kinds of facet

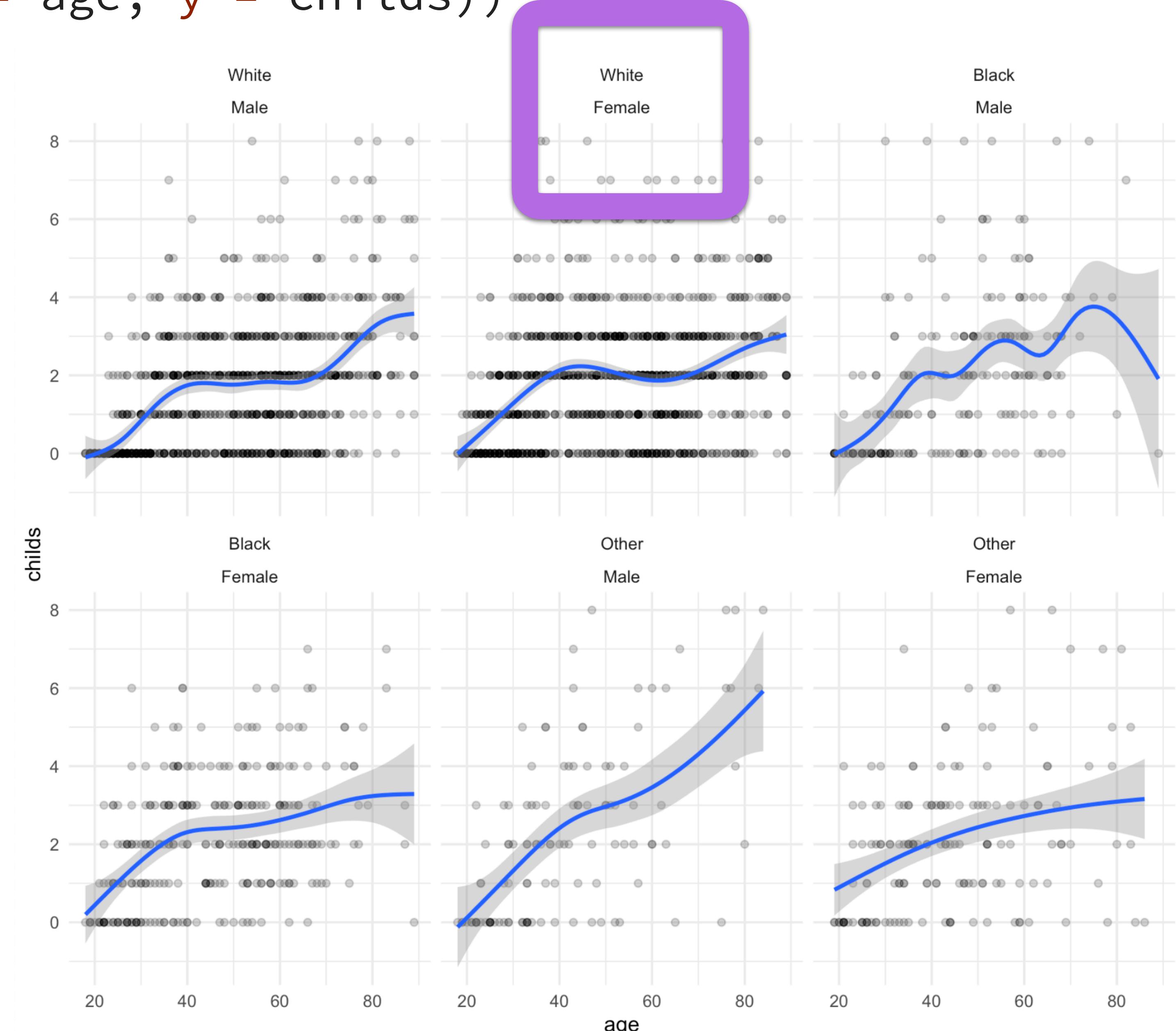
```
p <- ggplot(data = gss_sm, mapping = aes(x = age, y = childs))  
p + geom_point(alpha = 0.2) +  
  geom_smooth() + facet_wrap(~ race)
```



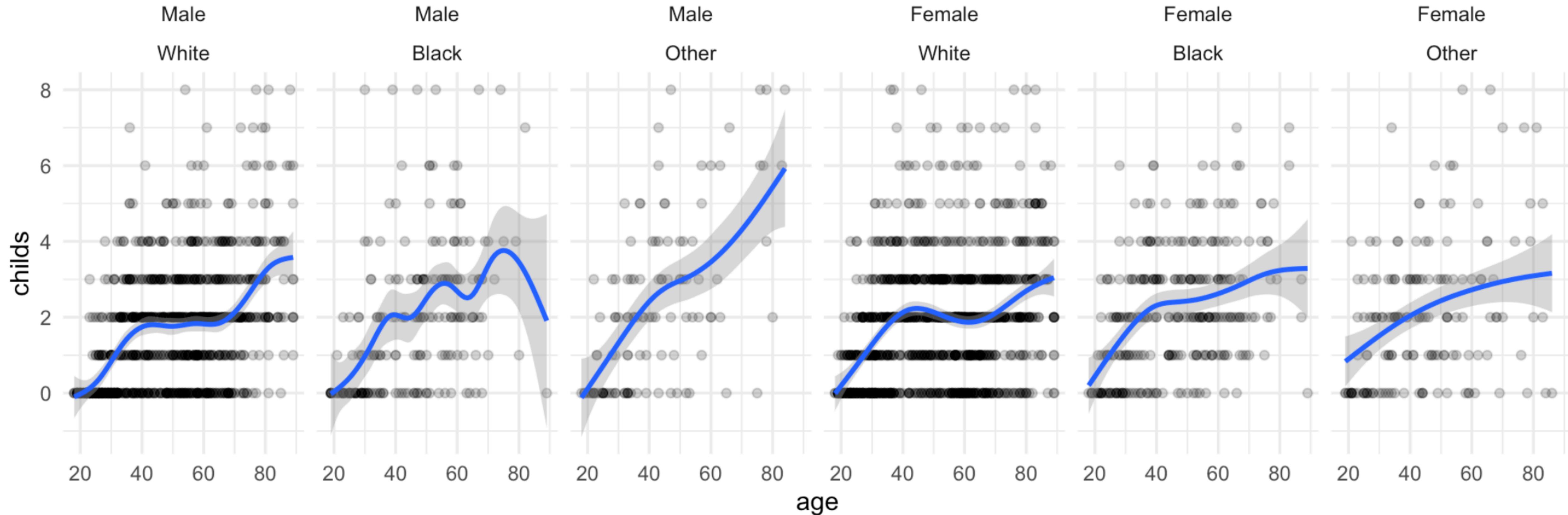
The formula syntax isn't limited to one variable



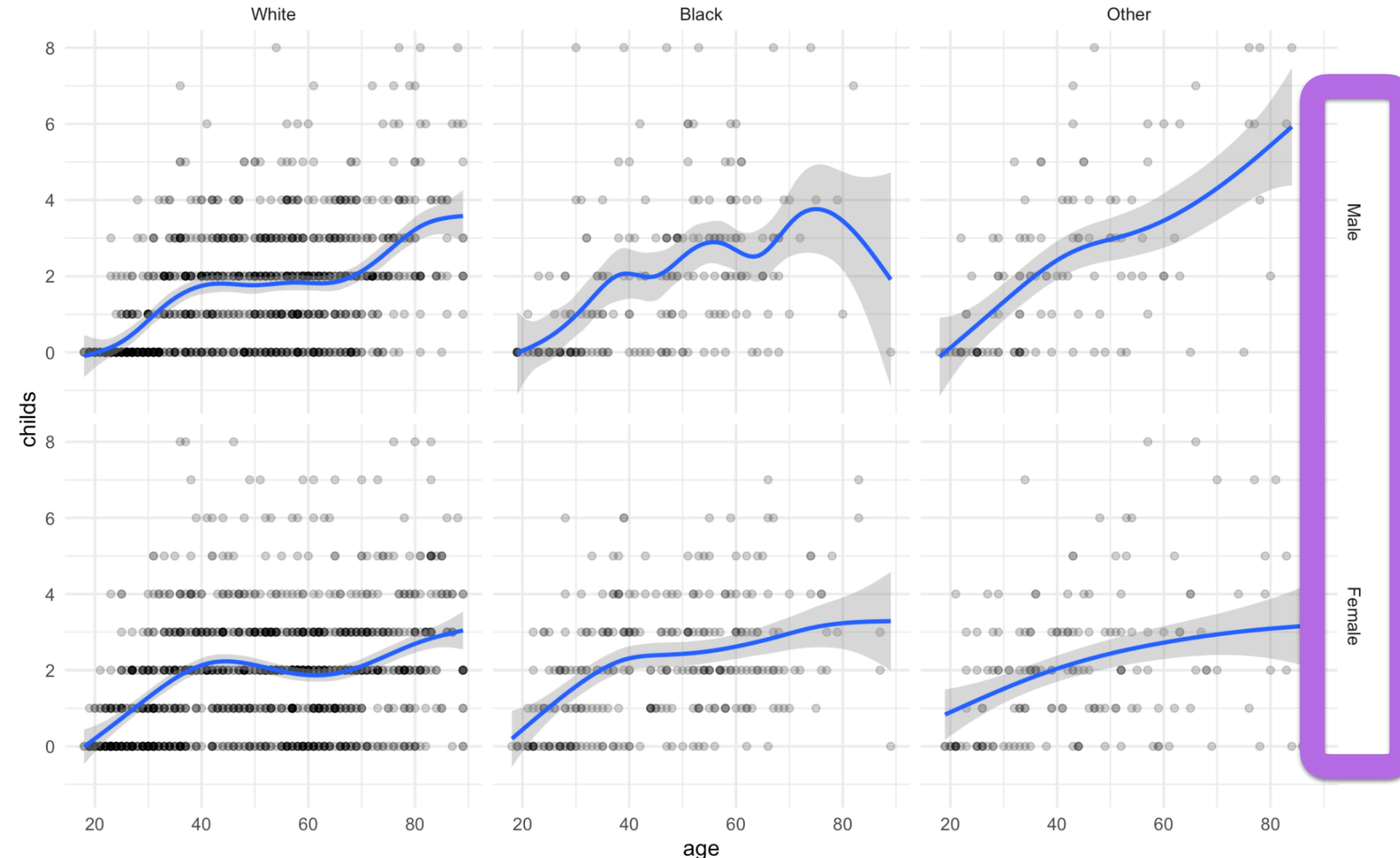
```
p <- ggplot(data = gss_sm,  
             mapping = aes(x = age, y = childs))  
p + geom_point(alpha = 0.2) +  
  geom_smooth() +  
  facet_wrap(~ sex + race)
```



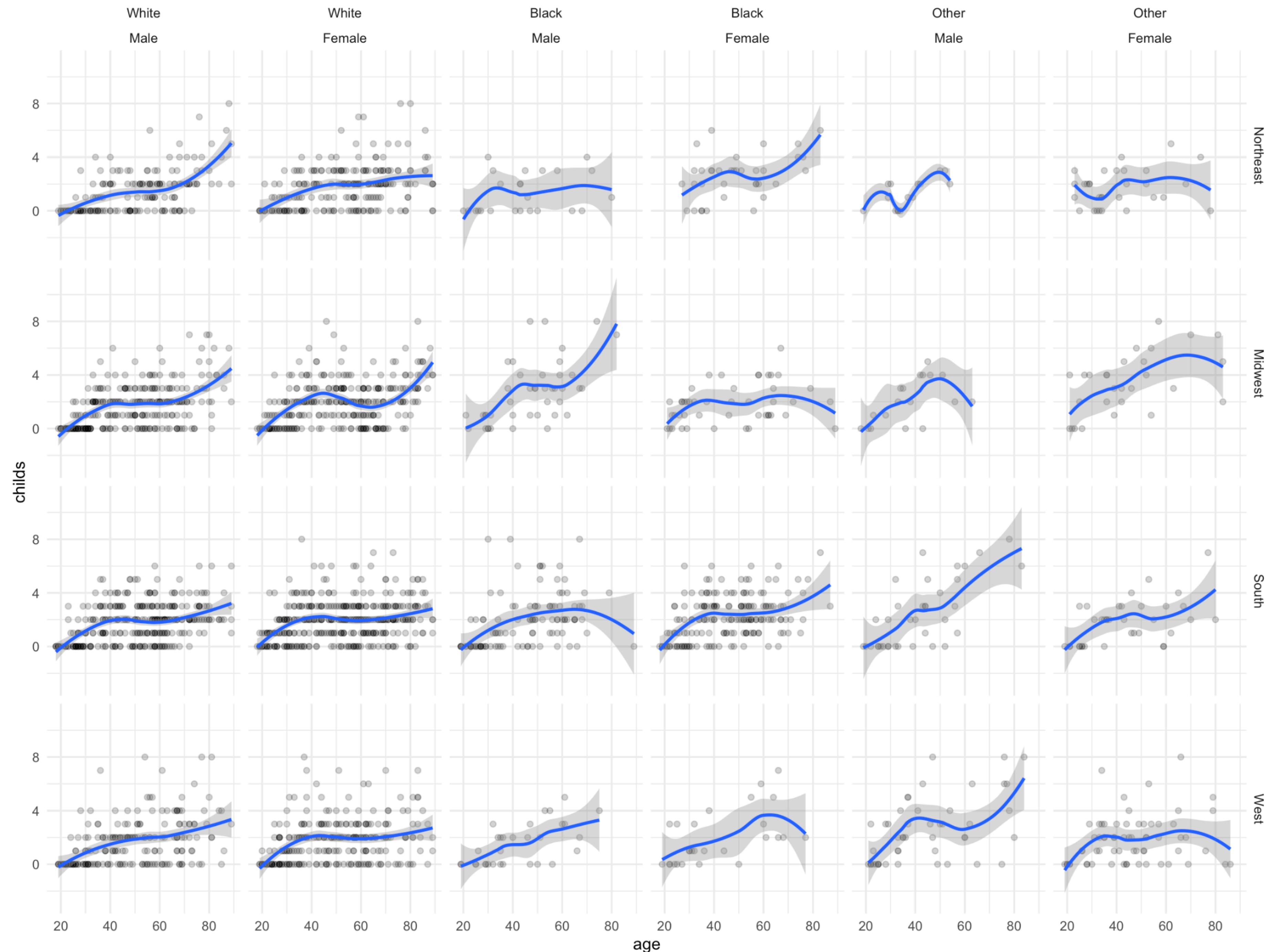
```
p <- ggplot(data = gss_sm, mapping = aes(x = age, y = childs))  
p + geom_point(alpha = 0.2) +  
  geom_smooth() + facet_grid(~ sex + race)
```



```
p <- ggplot(data = gss_sm, mapping = aes(x = age, y = childs))  
p + geom_point(alpha = 0.2) +  
  geom_smooth() + facet_grid(race ~ sex)
```



```
p <- ggplot(data = gss_sm, mapping = aes(x = age, y = childs))  
p + geom_point(alpha = 0.2) +  
geom_smooth() +  
facet_grid(bigregion ~ race + sex)
```



**WHAT
WE'VE NOW
BUILT UP**

```
p <- ggplot(data = <DATA>,  
           mapping=aes(<MAPPINGS>)) +
```

```
      <GEOM_FUNCTION>(  
        mapping = aes(<MAPPINGS>),  
        stat = <STAT>,  
        position = <POSITION>) +
```

```
<SCALE_FUNCTION> +  
<COORDINATE_FUNCTION> +  
<FACET_FUNCTION> +  
<THEME_FUNCTION>
```

```
p <- ggplot(data = gapminder,  
               mapping = aes(x = year,  
                                 y = gdpPercap))
```

```
p + geom_line(aes(group = country)) +  
scale_y_log10() +  
coord_cartesian() +  
facet_wrap(~ continent)
```

geom_point()

geom_line()

geom_smooth()

geom_bar()

geom_histogram()

geom_density()

geom_boxplot()

EXTENDING YOUR VOCABULARY

THE ORGAN DONATION DATA

```
organdata %>% select(1:6) %>% sample_n(size = 10)

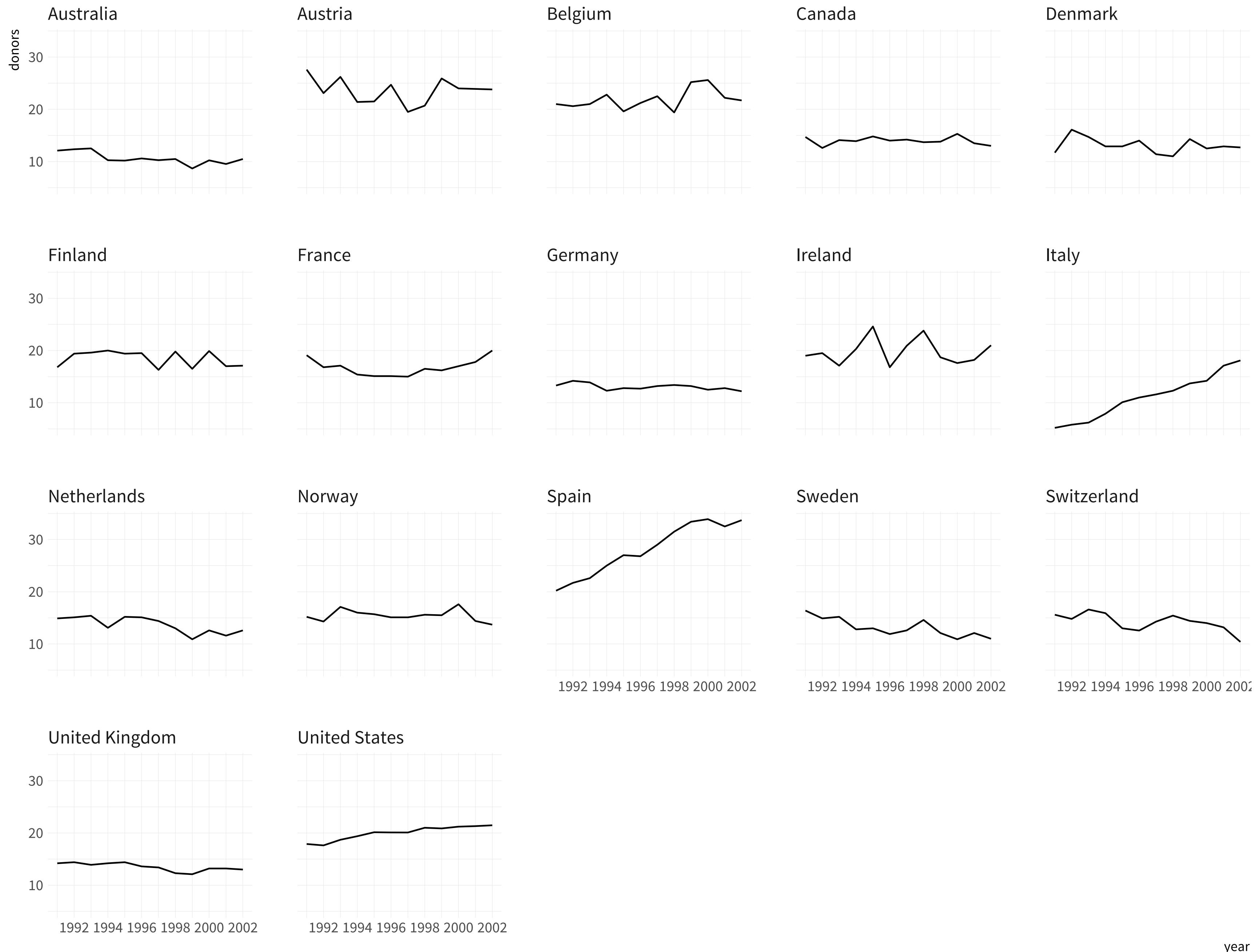
## # A tibble: 10 x 6
##   country       year   donors    pop  pop_dens    gdp
##   <chr>     <date>   <dbl>  <int>    <dbl>  <int>
## 1 Switzerland   NA      NA      NA      NA      NA
## 2 Switzerland 1997-01-01  14.3    7089    17.2  27675
## 3 United Kingdom 1997-01-01  13.4   58283    24.0 22442
## 4 Sweden        NA      NA      8559    1.90 18660
## 5 Ireland       2002-01-01  21.0    3932    5.60 32571
## 6 Germany        1998-01-01  13.4   82047    23.0 23283
## 7 Italy          NA      NA      56719    18.8 17430
## 8 Italy         2001-01-01  17.1   57894    19.2 25359
## 9 France         1998-01-01  16.5   58398    10.6 24044
## 10 Spain         1995-01-01  27.0   39223    7.75 15720
```

Everyday use of dplyr and pipes

```
p <- ggplot(data = organdata,  
             mapping = aes(x = year, y = donors))  
p + geom_point()
```

```
## Warning: Removed 34 rows containing missing values  
## (geom_point).
```

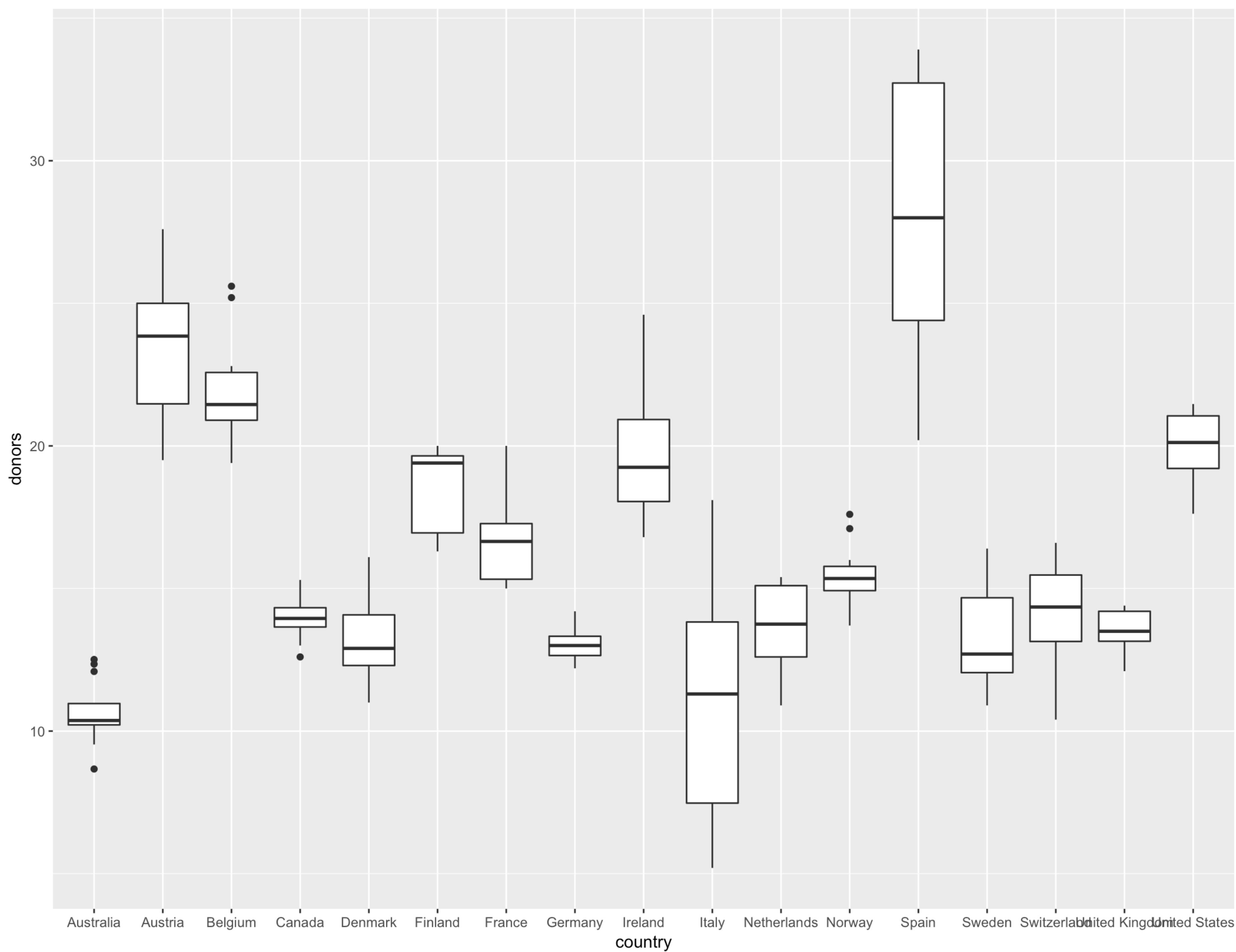
```
p <- ggplot(data = organdata,  
             mapping = aes(x = year, y = donors))  
p + geom_line(aes(group = country)) +  
  facet_wrap(~ country)
```



year

Continuous Variables by Categories

```
p <- ggplot(data = organdata,  
               mapping = aes(x = country, y = donors))  
  
p + geom_boxplot()
```

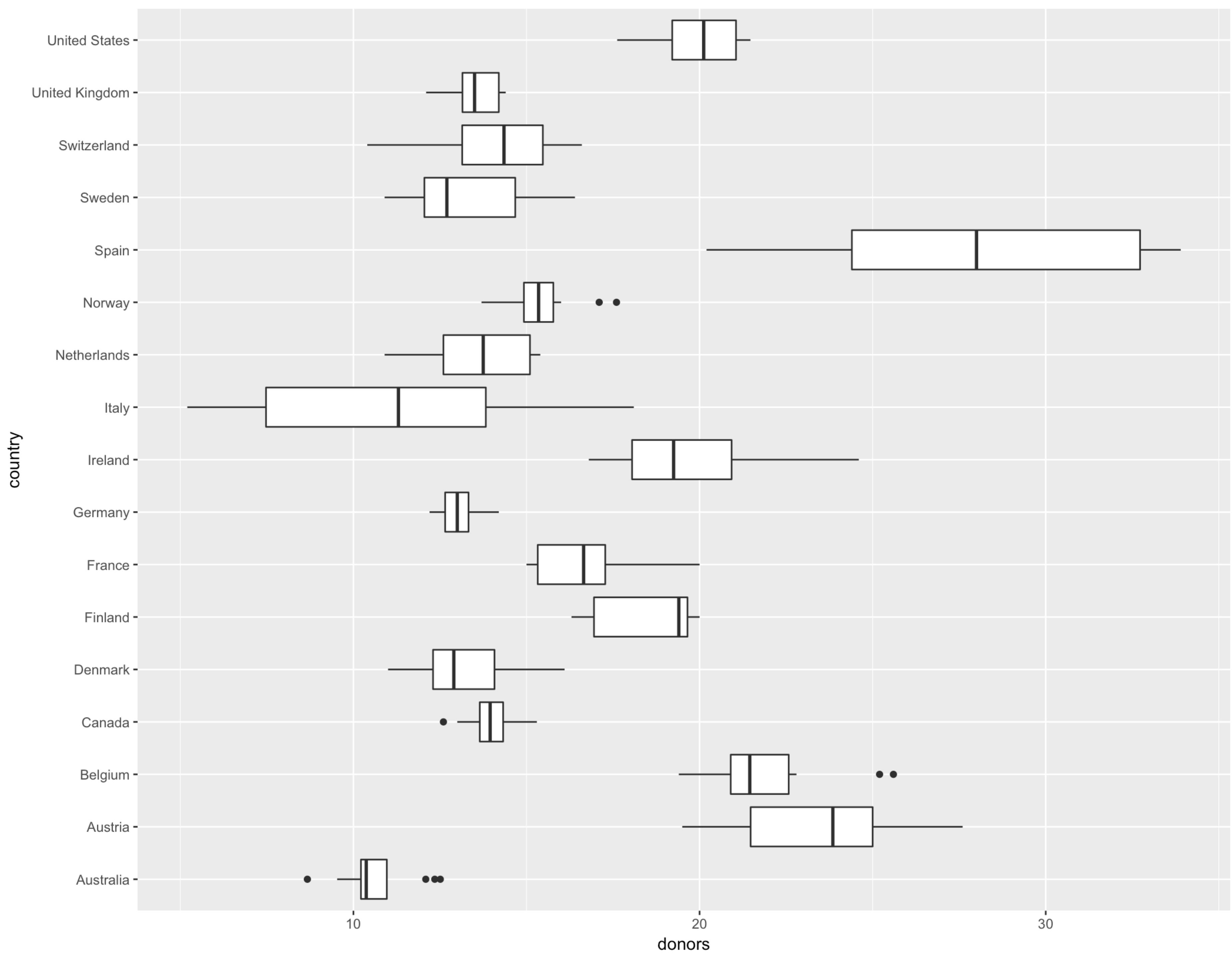


```
p <- ggplot(data = organdata,  
               mapping = aes(x = country, y = donors))
```

```
p + geom_boxplot() + coord_flip()
```



Explicit use of a coordinate
system transformation



```
p <- ggplot(data = organdata,  
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),  
                           y = donors))  
  
p + geom_boxplot() + labs(x = NULL) + coord_flip()
```

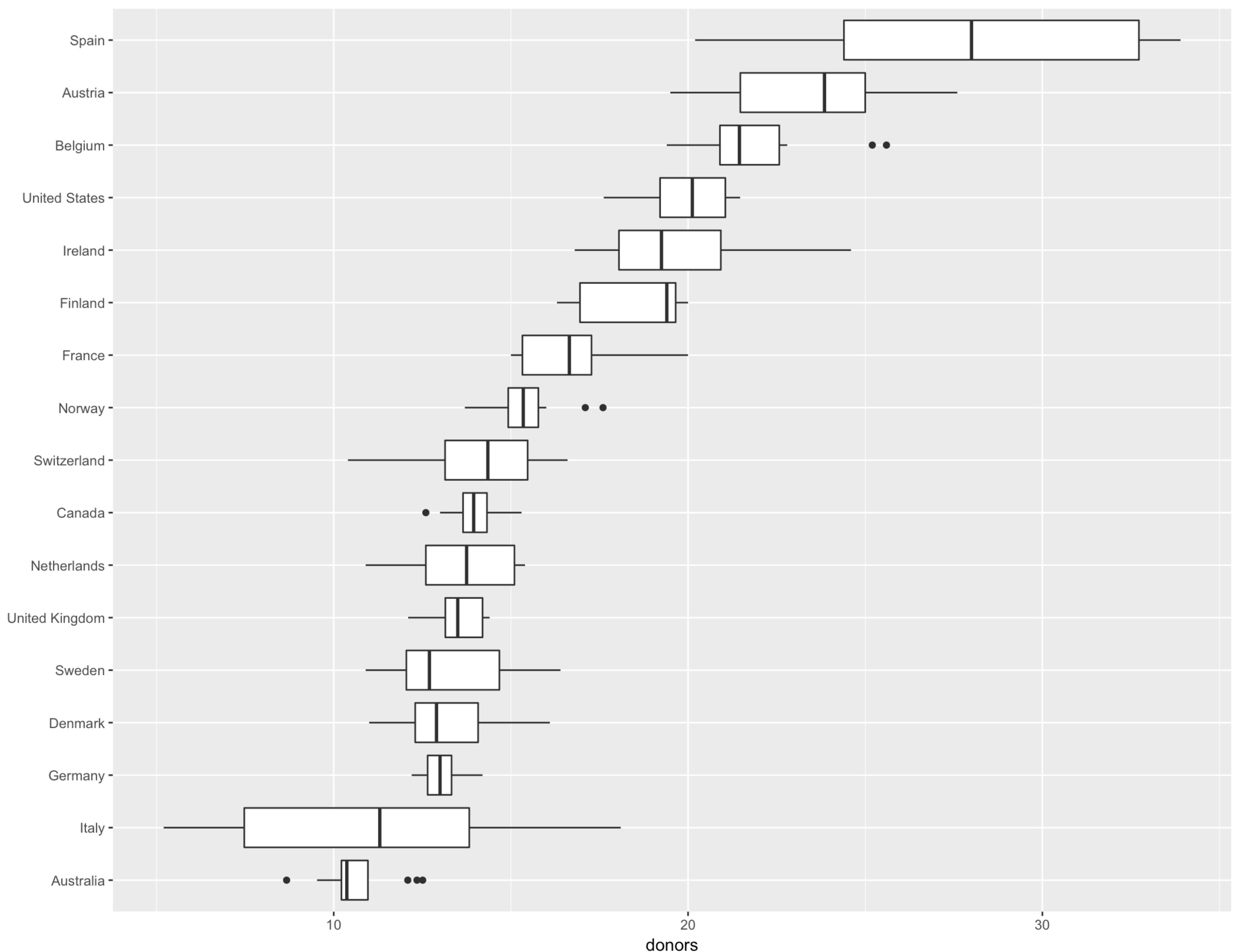
variable

default is mean()

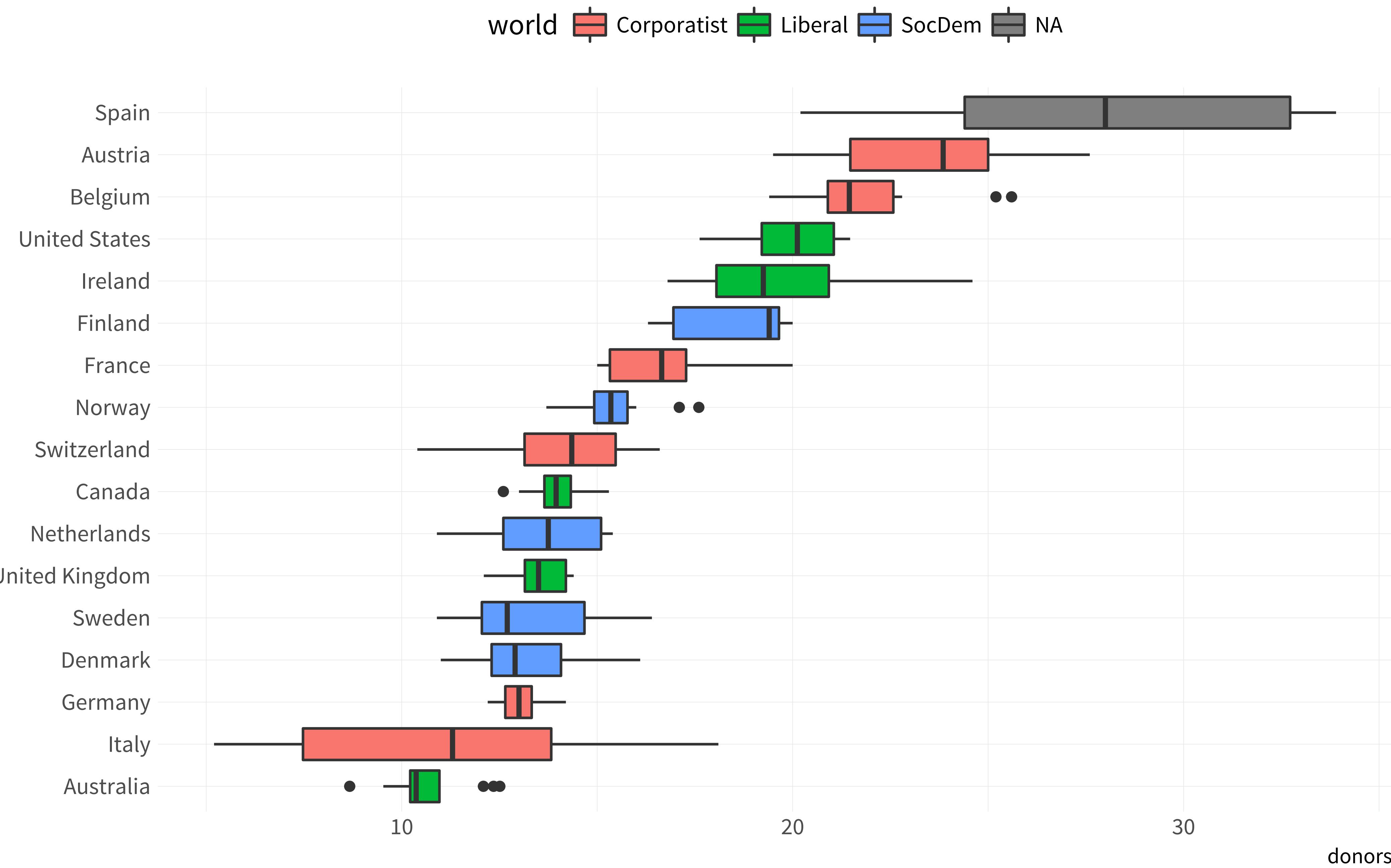
by

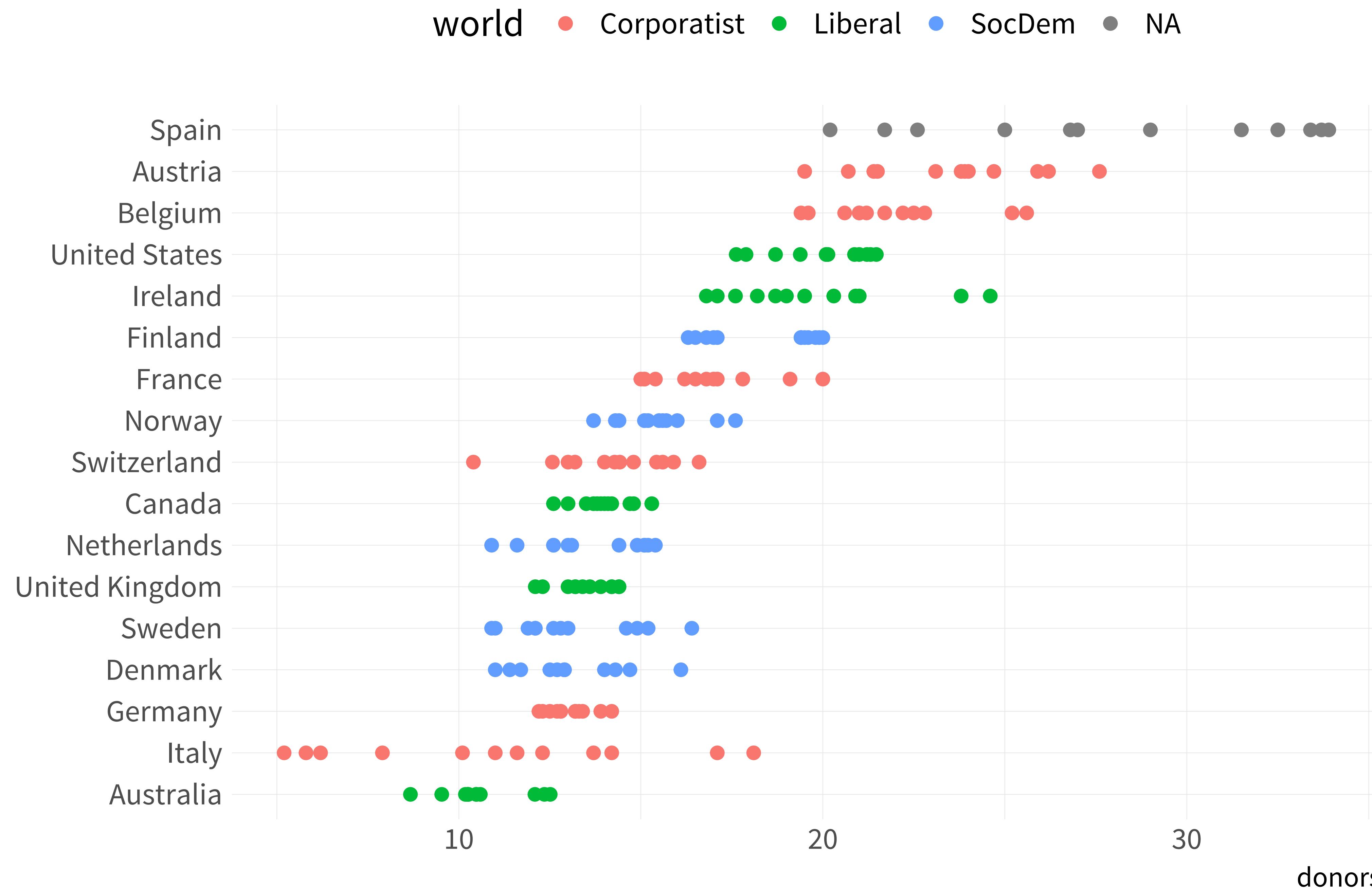
passed
to mean()

reorder() your data
in a sensible way



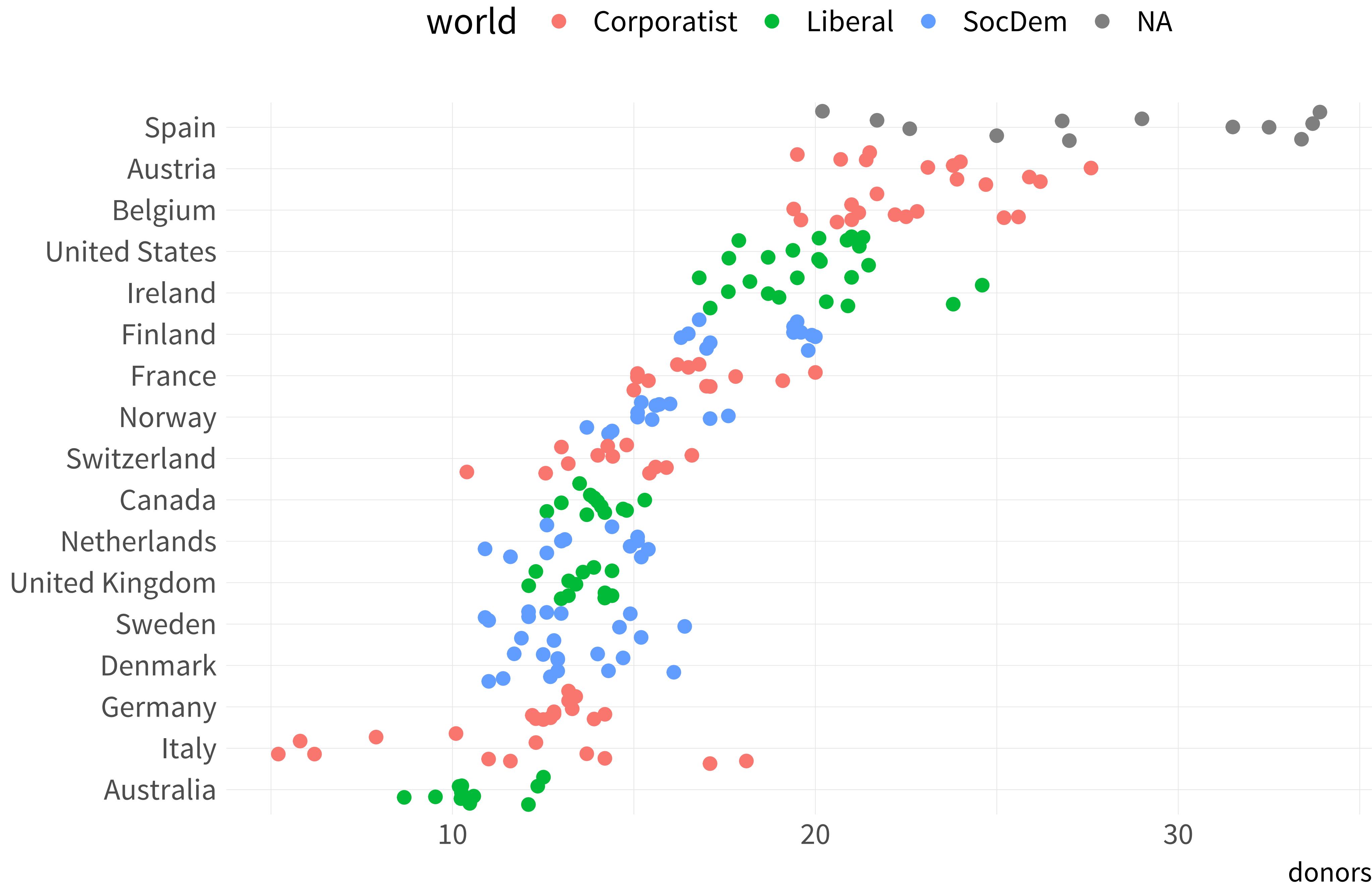
```
p <- ggplot(data = organdata,  
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),  
                           y = donors,  
                           fill = world))  
  
p + geom_boxplot() + labs(x=NULL) +  
  coord_flip() + theme(legend.position = "top")
```





```
p <- ggplot(data = organdata,  
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),  
                           y = donors, color = world))  
  
p + geom_jitter() + labs(x=NULL) +  
  coord_flip() + theme(legend.position = "top")
```

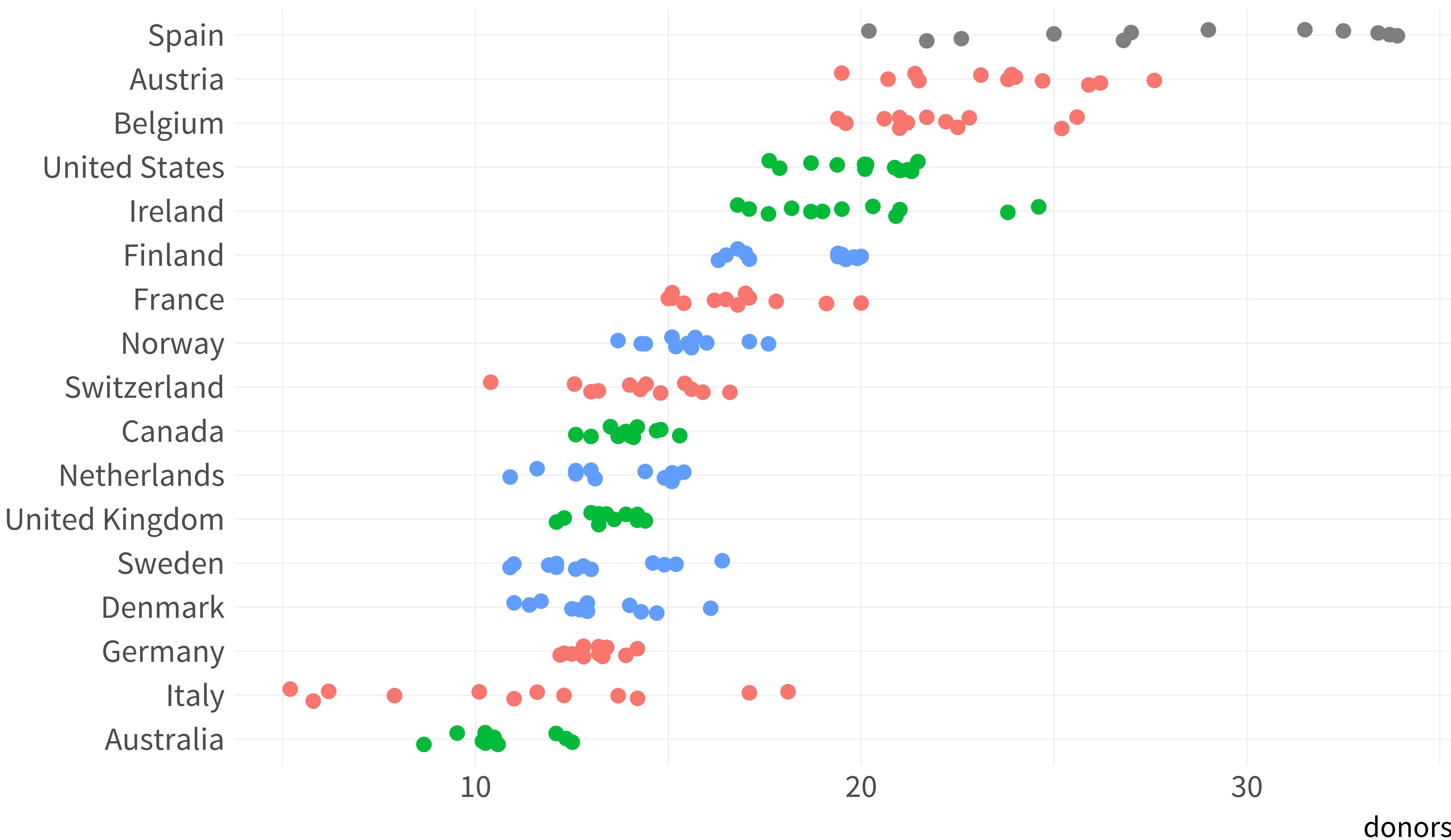
geom_jitter() can help
with overplotting



```
p <- ggplot(data = organdata,  
             mapping = aes(x = reorder(country, donors, na.rm=TRUE),  
                           y = donors, color = world))  
  
p + geom_jitter(position = position_jitter(width=0.15)) +  
  labs(x=NULL) +  
  coord_flip() +  
  theme(legend.position = "top")
```

geom_jitter() can help
with overplotting

world ● Corporatist ● Liberal ● SocDem ● NA



SUMMARIZE
BETTER WITH
dplyr

```
by_country <- organdata %>% group_by(consent_law, country) %>%  
  summarize(donors_mean = mean(donors, na.rm = TRUE),  
             donors_sd = sd(donors, na.rm = TRUE),  
             gdp = mean(gdp, na.rm = TRUE),  
             health = mean(health, na.rm = TRUE),  
             roads_mean = mean(roads_mean, na.rm = TRUE),  
             cerebvas = mean(cerebvas, na.rm = TRUE))
```

This direct method works;
But there's lots of code repetition

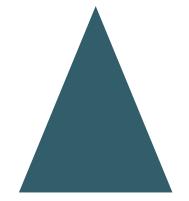
by_country

```
## Source: local data frame [17 x 8]
## Groups: consent_law [?]
##
## # A tibble: 17 x 8
##   consent_law      country donors_mean donors_sd    gdp health roads_mean cerebvas
##   <chr>          <chr>     <dbl>     <dbl> <dbl> <dbl>     <dbl>     <dbl>
## 1 Informed        Australia     11       1.1  22179  1958     105      558
## 2 Informed        Canada       14       0.8  23711  2272     109      422
## 3 Informed        Denmark      13       1.5  23722  2054     102      641
## 4 Informed        Germany      13       0.6  22163  2349     113      707
## 5 Informed        Ireland      20       2.5  20824  1480     118      705
## 6 Informed        Netherlands   14       1.6  23013  1993      76      585
## 7 Informed        United Kingdom 13       0.8  21359  1561      68      708
## 8 Informed        United States 20       1.3  29212  3988     155      444
## 9 Presumed         Austria      24       2.4  23876  1875     150      769
## 10 Presumed        Belgium      22       1.9  22500  1958     155      594
## 11 Presumed        Finland      18       1.5  21019  1615      94      771
## 12 Presumed        France       17       1.6  22603  2160     156      433
## 13 Presumed        Italy        11       4.3  21554  1757     122      712
## 14 Presumed        Norway       15       1.1  26448  2217      70      662
## 15 Presumed        Spain        28       5.0  16933  1289     161      655
## 16 Presumed        Sweden       13       1.8  22415  1951      72      595
## 17 Presumed        Switzerland  14       1.7  27233  2776      96      424
```

Map your functions, instead

```
by_country <- organdata %>%  
  group_by(consent_law, country) %>%  
  summarize_if(is.numeric,  
    list(~ mean(., na.rm = TRUE),  
        ~ sd(., na.rm = TRUE))) %>%  
  ungroup()
```

by_country



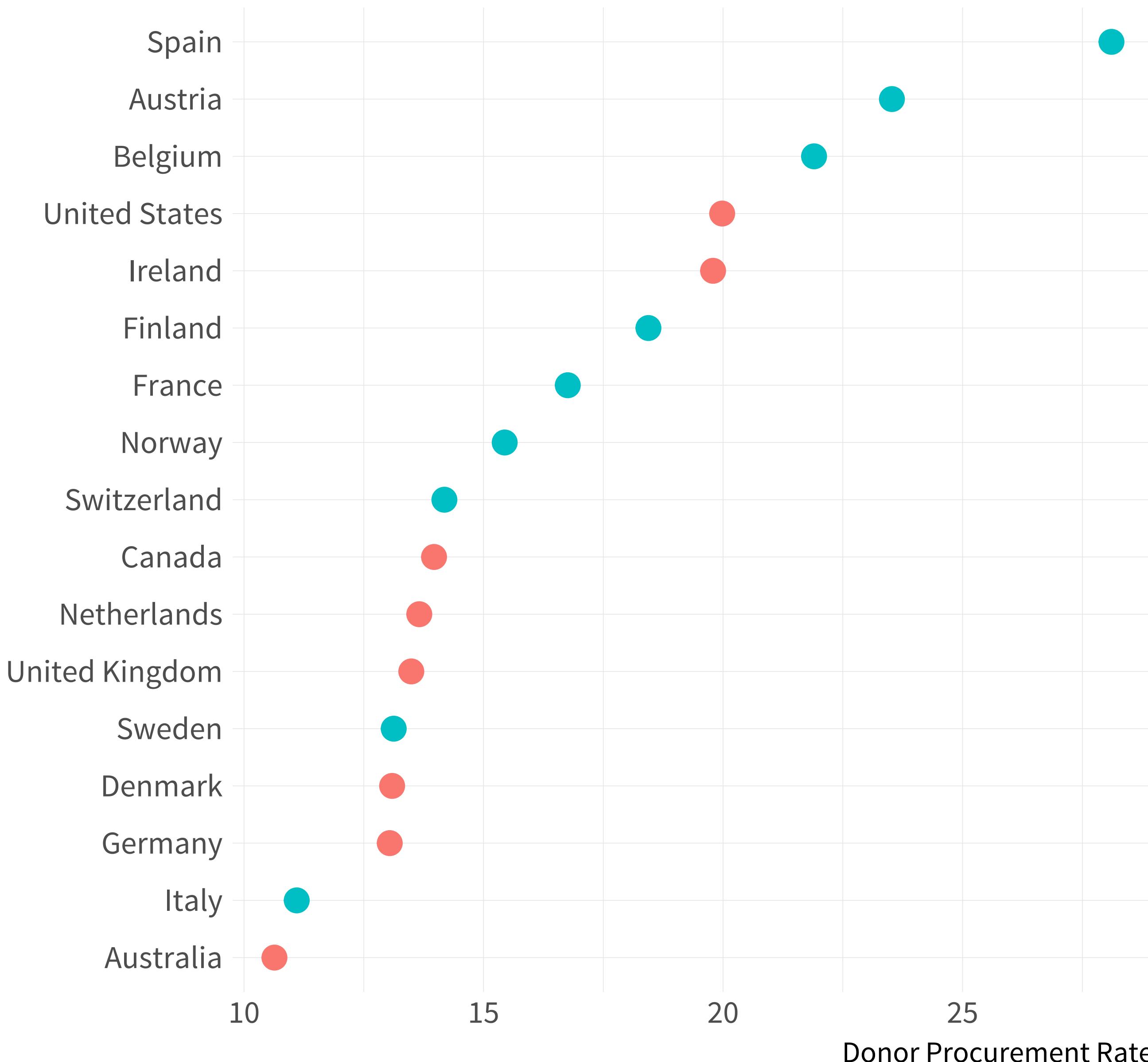
"the thing"

(More on this later)

```
# A tibble: 17 x 28
  consent_law country donors_mean pop_mean pop_dens_mean gdp_mean gdp_lag_mean health_mean health_lag_mean
  <chr>       <chr>     <dbl>    <dbl>        <dbl>    <dbl>        <dbl>    <dbl>        <dbl>        <dbl>
1 Informed    Australia 10.6      18318.       0.237    22179.       21779.       1958.       1848.
2 Informed    Canada   14.0      29608.       0.297    23711.       23353.       2272.       2163.
3 Informed    Denmark  13.1      5257.        12.2     23722.       23275.       2054.       1973.
4 Informed    Germany  13.0      80255.       22.5     22163.       21938.       2349.       2256.
5 Informed    Ireland  19.8      3674.        5.23    20824.       20154.       1480.       1341.
6 Informed    Nether... 13.7      15548.       37.4     23013.       22554.       1993.       1885.
7 Informed    United... 13.5      58187.       24.0     21359.       20962.       1561.       1464.
8 Informed    United... 20.0      269330.      2.80    29212.       28699.       3988.       3760.
9 Presumed    Austria  23.5      7927.        9.45    23876.       23415.       1875.       1803.
10 Presumed   Belgium  21.9      10153.       30.7     22500.       22096.       1958.       1862.
11 Presumed   Finland  18.4      5112.        1.51    21019.       20763.       1615.       1560.
12 Presumed   France   16.8      58056.       10.5     22603.       22211.       2160.       2066.
13 Presumed   Italy    11.1      57360.       19.0     21554.       21195.       1757.       1689.
14 Presumed   Norway  15.4      4386.        1.35    26448.       25769.       2217.       2125.
15 Presumed   Spain   28.1      39666.       7.84     16933.       16584.       1289.       1220.
16 Presumed   Sweden  13.1      8789.        1.95    22415.       22094.       1951.       1868.
17 Presumed   Switze... 14.2      7037.        17.0     27233.       26931.       2776.       2656.
# ... with 19 more variables: pubhealth_mean <dbl>, roads_mean <dbl>, cerebvas_mean <dbl>, assault_mean <dbl>,
# external_mean <dbl>, txp_pop_mean <dbl>, donors_sd <dbl>, pop_sd <dbl>, pop_dens_sd <dbl>, gdp_sd <dbl>,
# gdp_lag_sd <dbl>, health_sd <dbl>, health_lag_sd <dbl>, pubhealth_sd <dbl>, roads_sd <dbl>, cerebvas_sd
<dbl>,
# assault_sd <dbl>, external_sd <dbl>, txp_pop_sd <dbl>
```

```
p <- ggplot(data = by_country,  
             mapping = aes(x = donors_mean,  
                           y = reorder(country, donors_mean),  
                           color = consent_law))  
  
p + geom_point(size = 3) +  
  labs(x = "Donor Procurement Rate",  
        y = NULL,  
        color = "Consent Law") +  
  theme(legend.position = "top")
```

Consent Law ● Informed ● Presumed

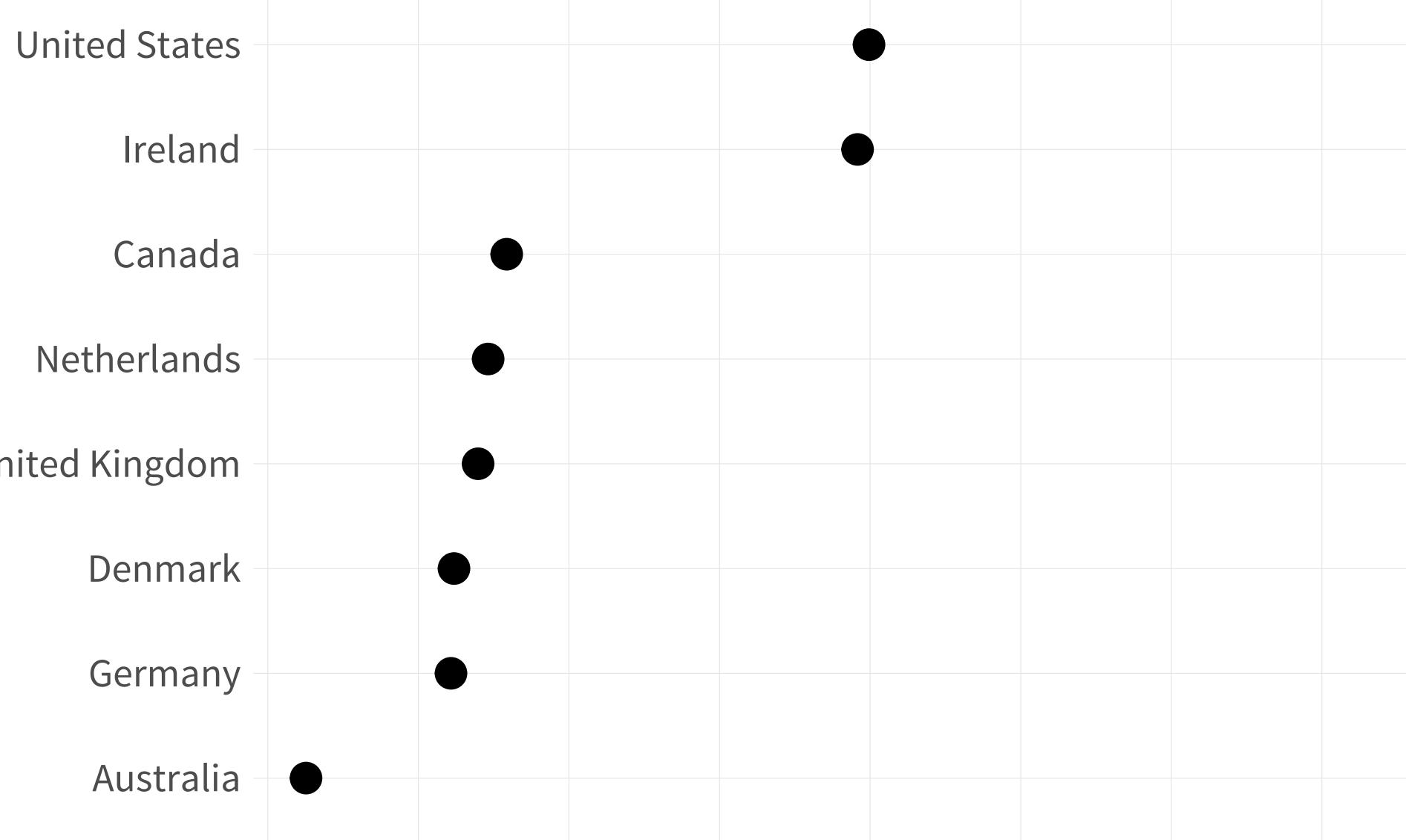


```
p <- ggplot(data = by_country,  
             mapping = aes(x = donors_mean,  
                           y = reorder(country,  
                                       donors_mean)))  
  
p + geom_point(size=3) +  
  facet_wrap(~ consent_law) +  
  labs(x = "Donor Procurement Rate",  
        y = NULL)
```

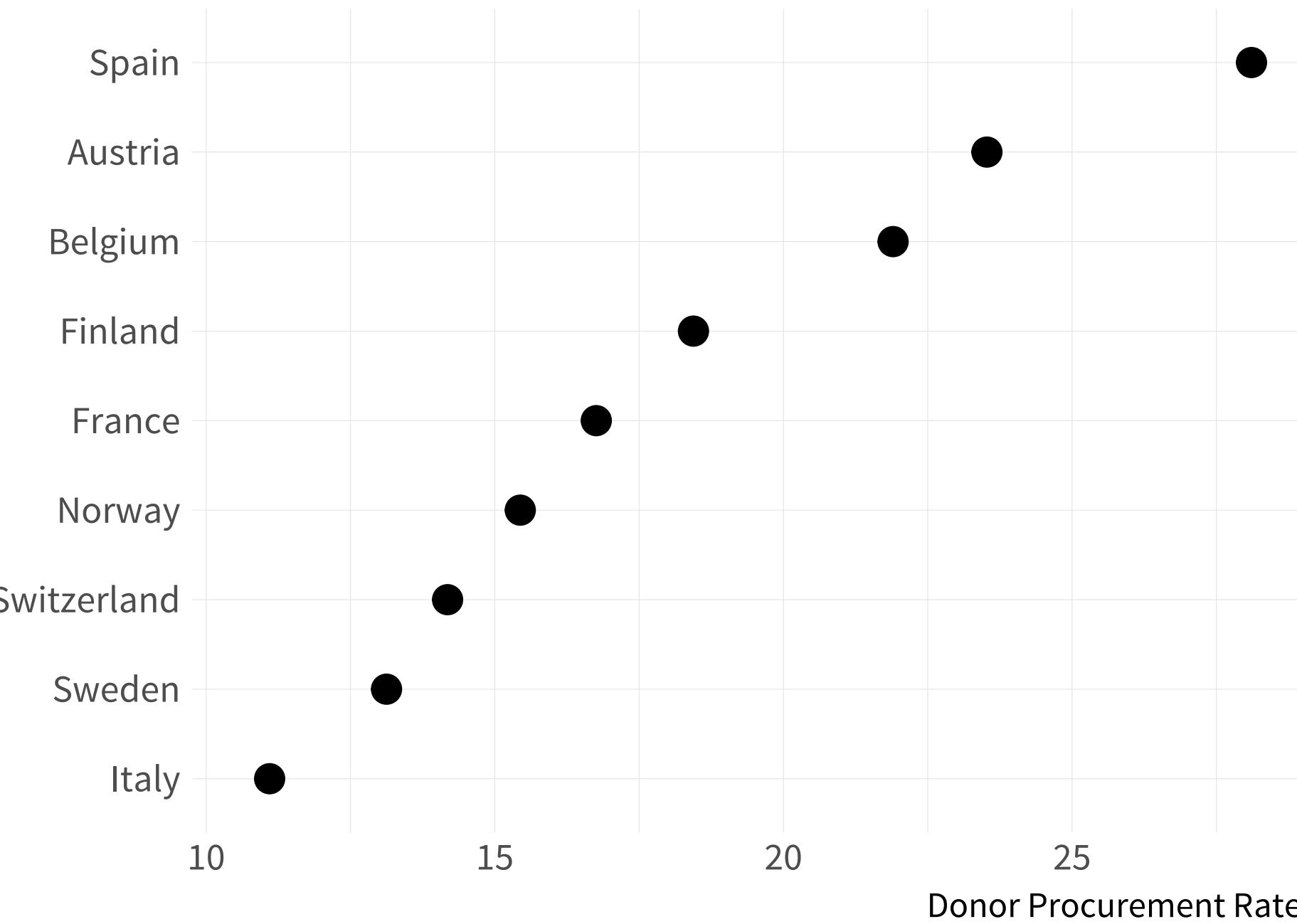
```
p <- ggplot(data = by_country,  
             mapping = aes(x = donors_mean,  
                           y = reorder(country,  
                                       donors_mean)))  
  
p + geom_point(size=3) +  
  facet_wrap(~ consent_law, scales = "free_y") +  
  labs(x= "Donor Procurement Rate",  
        y= NULL)
```

```
p <- ggplot(data = by.country,  
             mapping = aes(x = donors_mean,  
                           y = reorder(country, donors_mean)))  
  
p + geom_point(size=3) +  
  facet_wrap(~ consent_law,  
             scales = "free_y", ncol = 1) +  
  labs(x = "Donor Procurement Rate",  
        y = NULL)
```

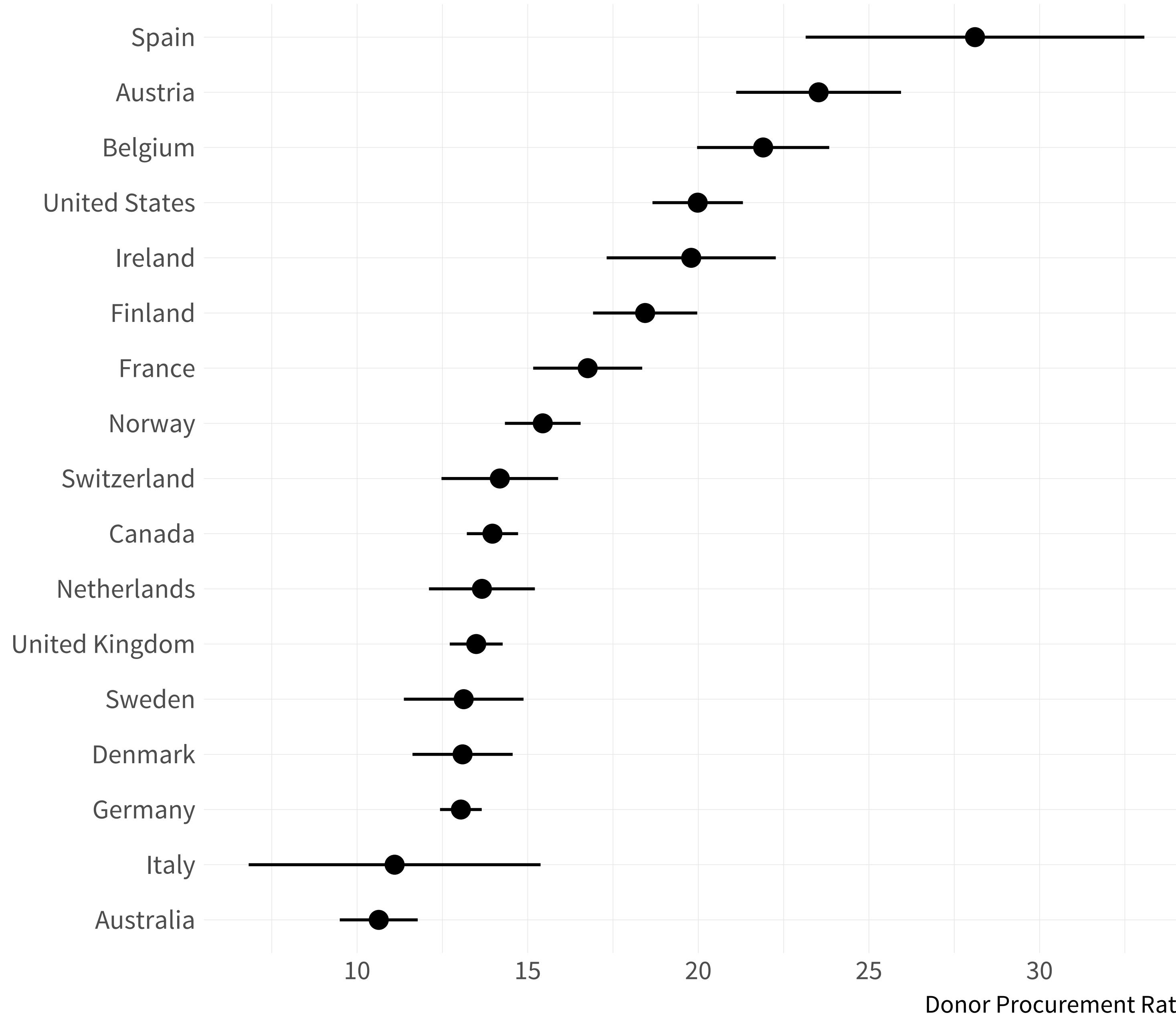
Informed



Presumed



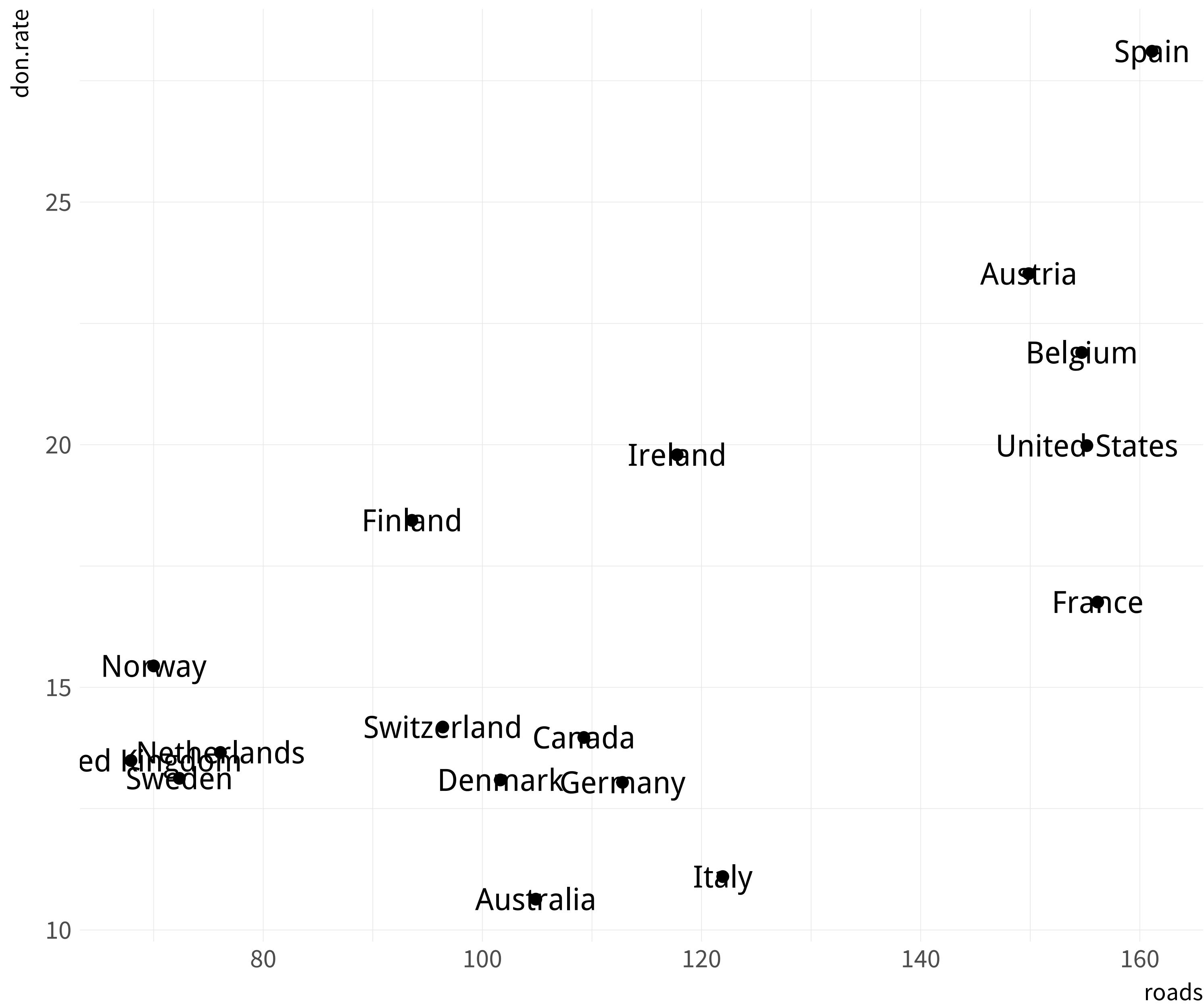
```
p <- ggplot(data = by_country,  
             mapping = aes(x = reorder(country, donors_mean),  
                           y = donors_mean))  
  
p + geom_pointrange(mapping = aes(ymin = donors_mean - donors_sd,  
                                     ymax = donors_mean + donors_sd)) +  
  labs(x = NULL, y = "Donor Procurement Rate") +  
  coord_flip()
```



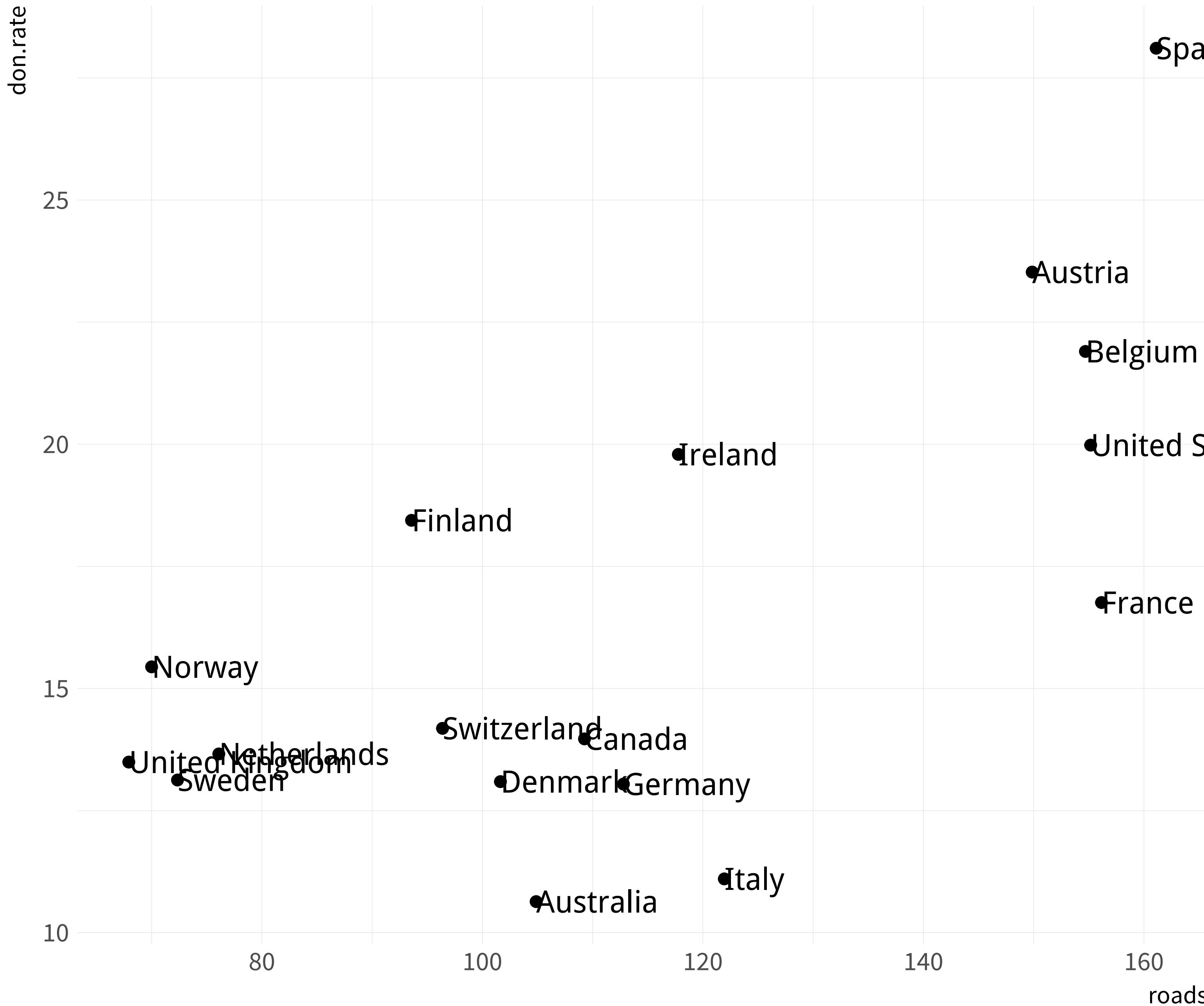
**PLOTTING TEXT
DIRECTLY**

```
geom_text(mapping = aes(label = <VARIABLE>))
```

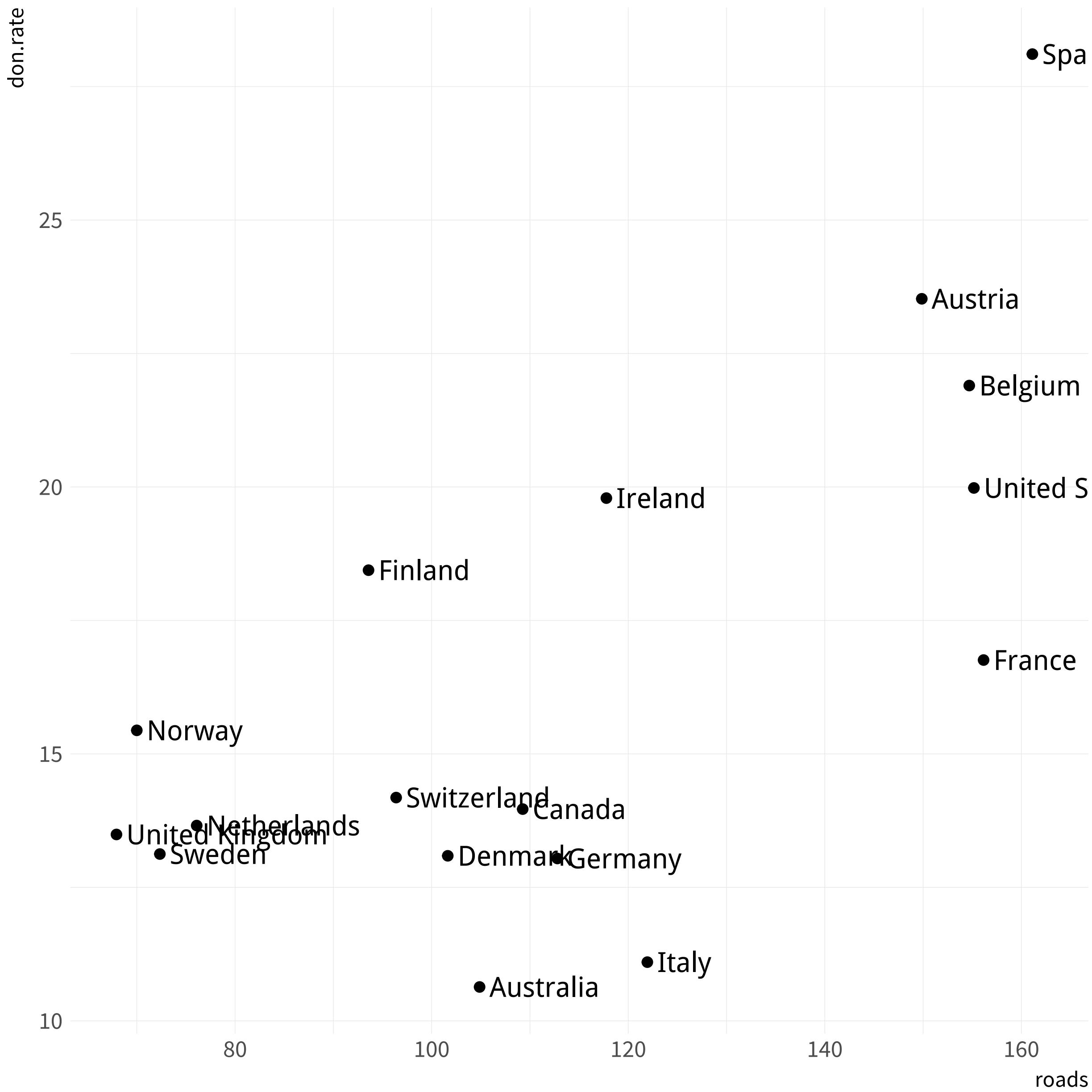
```
p <- ggplot(data = by_country,  
             mapping = aes(x = roads_mean, y = donors_mean))  
  
p + geom_point() +  
  geom_text(mapping = aes(label = country))
```



```
p <- ggplot(data = by_country,  
             mapping = aes(x = roads_mean,  
                           y = donors_mean))  
  
p + geom_point() + geom_text(mapping = aes(label = country),  
                           hjust = 0)
```



```
p <- ggplot(data = by_country,  
             mapping = aes(x = roads_mean, y = donors_mean))  
  
p + geom_point() +  
  geom_text(mapping = aes(x = roads_mean + 1, label = country),  
            hjust = 0)
```



```
library(ggrepel)
```

This library provides
`geom_text_repel()` and
`geom_label_repel()`

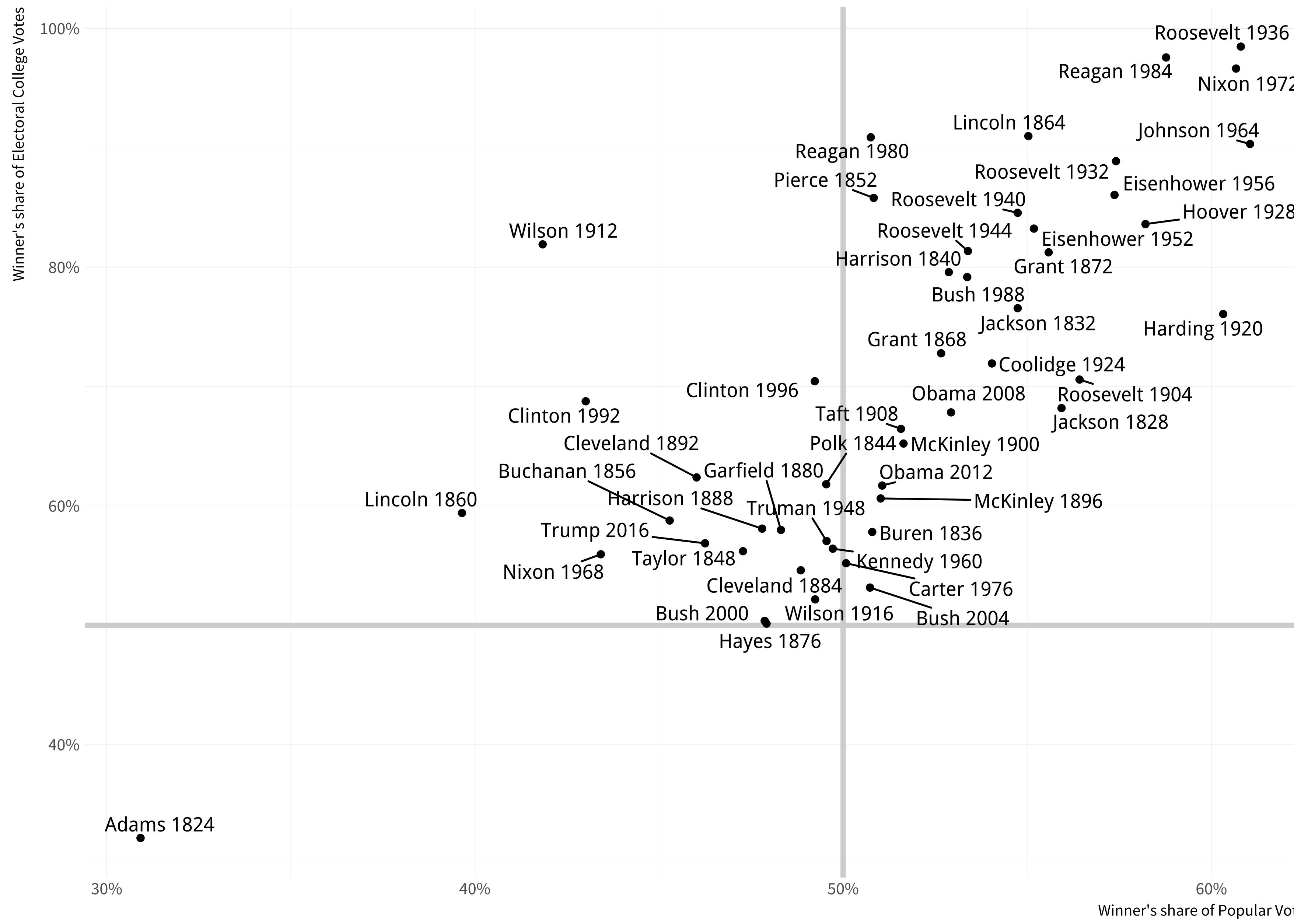
```
elections_historic %>% select(2:7)
```

US Elections Data

```
## # A tibble: 49 x 6
##       year     winner win_party   ec_pct popular_pct popular_margin
##   <int>     <chr>    <chr>    <dbl>    <dbl>            <dbl>
## 1 1824 John Quincy Adams D.-R.      0.322     0.309        -0.1044
## 2 1828 Andrew Jackson   Dem.      0.682     0.559         0.1225
## 3 1832 Andrew Jackson   Dem.      0.766     0.547         0.1781
## 4 1836 Martin Van Buren Dem.      0.578     0.508         0.1420
## 5 1840 William Henry Harrison Whig      0.796     0.529         0.0605
## 6 1844 James Polk       Dem.      0.618     0.495         0.0145
## 7 1848 Zachary Taylor   Whig      0.562     0.473         0.0479
## 8 1852 Franklin Pierce  Dem.      0.858     0.508         0.0695
## 9 1856 James Buchanan   Dem.      0.588     0.453         0.1220
## 10 1860 Abraham Lincoln Rep.      0.594     0.397         0.1013
## # ... with 39 more rows
```

Presidential Elections: Popular & Electoral College Margins

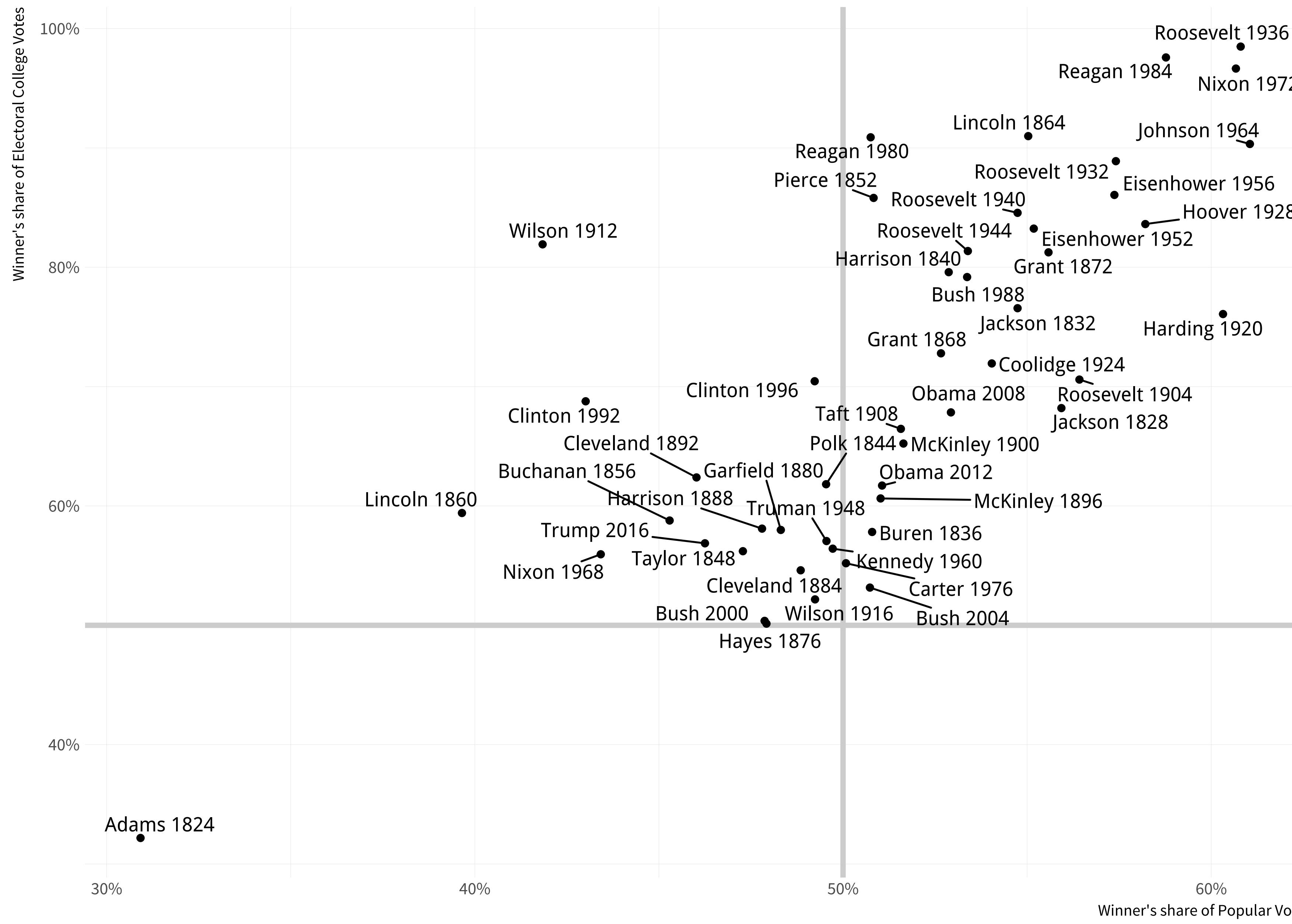
1824-2016



Data for 2016 are provisional.

Presidential Elections: Popular & Electoral College Margins

1824-2016



Data for 2016 are provisional.

```
p_title <- "Presidential Elections: Popular & Electoral College Margins"  
p_subtitle <- "1824-2016"  
p_caption <- "Data for 2016 are provisional."  
x_label <- "Winner's share of Popular Vote"  
y_label <- "Winner's share of Electoral College Votes"
```

**Put labels in objects
to keep your code tidy**

```
p_title <- "Presidential Elections: Popular & Electoral College Margins"  
p_subtitle <- "1824-2016"  
p_caption <- "Data for 2016 are provisional."  
x_label <- "Winner's share of Popular Vote"  
y_label <- "Winner's share of Electoral College Votes"
```

```
theme_set(theme_minimal())
```



Set a theme

**Put labels in objects
to keep your code tidy**

```
p <- ggplot(data = elections_historic,  
               mapping = aes(x = popular_pct,  
                               y = ec_pct,  
                               label = winner_label))  
  
p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray70") +  
geom_vline(xintercept = 0.5, size = 1.4, color = "gray70") +  
geom_point()
```

Base Layer, Grid Lines, Points

```
p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray70") +  
  geom_vline(xintercept = 0.5, size = 1.4, color = "gray70") +  
  geom_point() +  
  geom_text_repel()
```

Add the textual labels

```
p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray70") +  
  geom_vline(xintercept = 0.5, size = 1.4, color = "gray70") +  
  geom_point() +  
  geom_text_repel() +  
  scale_x_continuous(labels = scales::percent) +  
  scale_y_continuous(labels = scales::percent)
```

Add the scale adjustments

```
p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray70") +
  geom_vline(xintercept = 0.5, size = 1.4, color = "gray70") +
  geom_point() +
  geom_text_repel() +
  scale_x_continuous(labels = scales::percent) +
  scale_y_continuous(labels = scales::percent) +
  labs(x = x_label,
       y = y_label,
       title = p_title,
       subtitle = p_subtitle,
       caption = p_caption)
```

Add the scale and guide labels

Use ggsave

```
ggsave()
```

```
ggsave("my_figure.png")
```

```
ggsave("my_figure.pdf")
```

```
ggsave("my_figure.pdf",  
       plot = p5,  
       scale = 1.2)
```

```
ggsave("figures/my-figure.pdf",  
       plot = p5,  
       width = 8,  
       height = 5)
```

With pdf() or other graphics devices

```
pdf(file = "plot.pdf", height = 5in,  
     width = 5in)
```

▲
Open device ...

```
print(p5)
```

▲
dev.off()

... and close when done

Within an Rmd file using knitr's options

```
```{r electionplot, fig.cap="Popular and Electoral College Margins.", out.width="9in", fig.width=9, fig.height=8, fig.fullwidth=TRUE, warning=FALSE, echo=FALSE}

p + geom_hline(yintercept = 0.5, size = 1.4, color = "gray70") +
 geom_vline(xintercept = 0.5, size = 1.4, color = "gray70") +
 geom_point() +
 geom_text_repel() +
 scale_x_continuous(labels = scales::percent) +
 scale_y_continuous(labels = scales::percent) +
 labs(x = x_label,
 y = y_label,
 title = p_title,
 subtitle = p_subtitle,
 caption = p_caption)

```

```

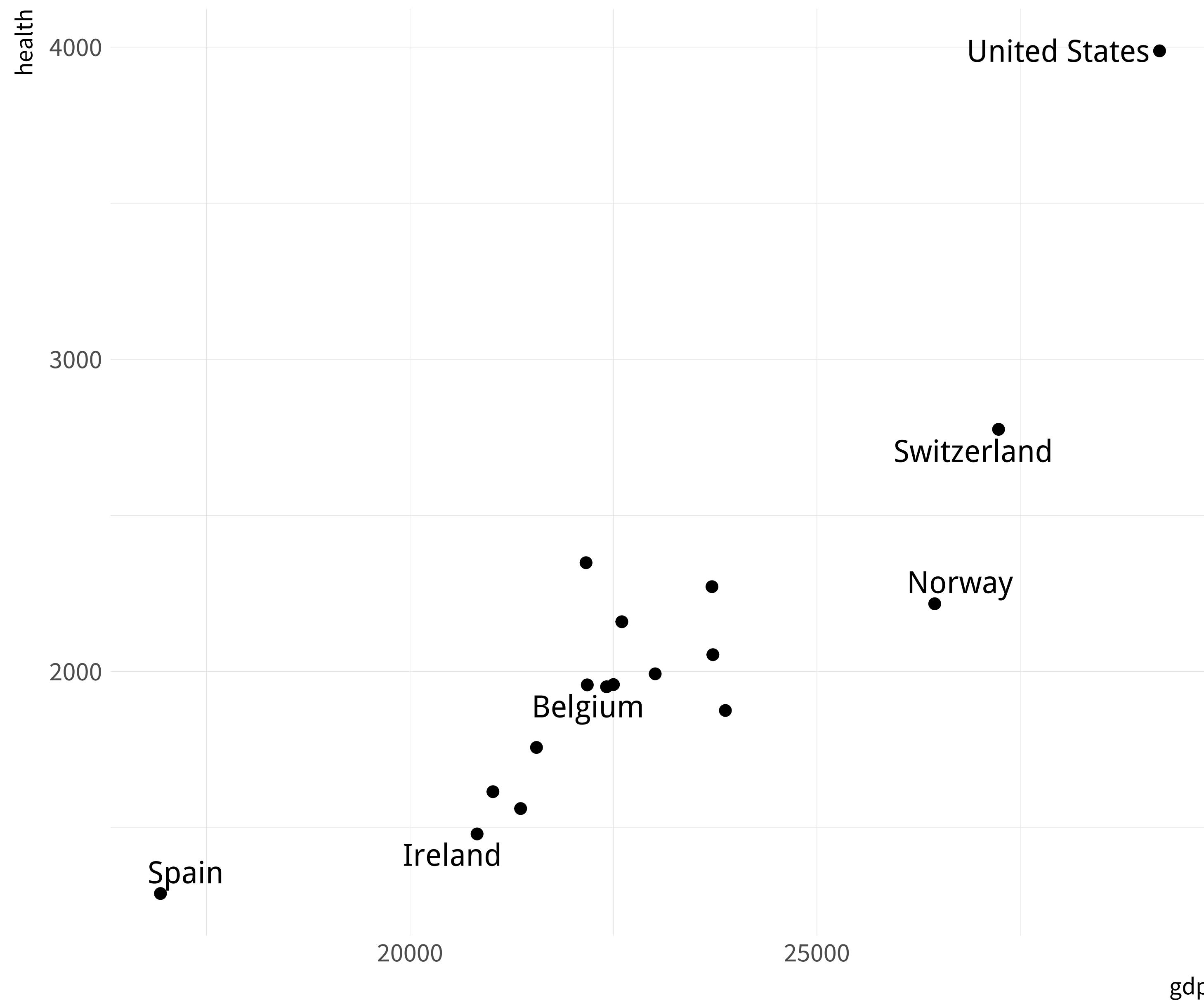
Labeling Points of Interest

```
p <- ggplot(data = by_country,  
             mapping = aes(x = gdp, y = health))
```

```
p + geom_point() +  
  geom_text_repel(data = subset(by_country, gdp > 25000),  
                  mapping = aes(label = country))
```

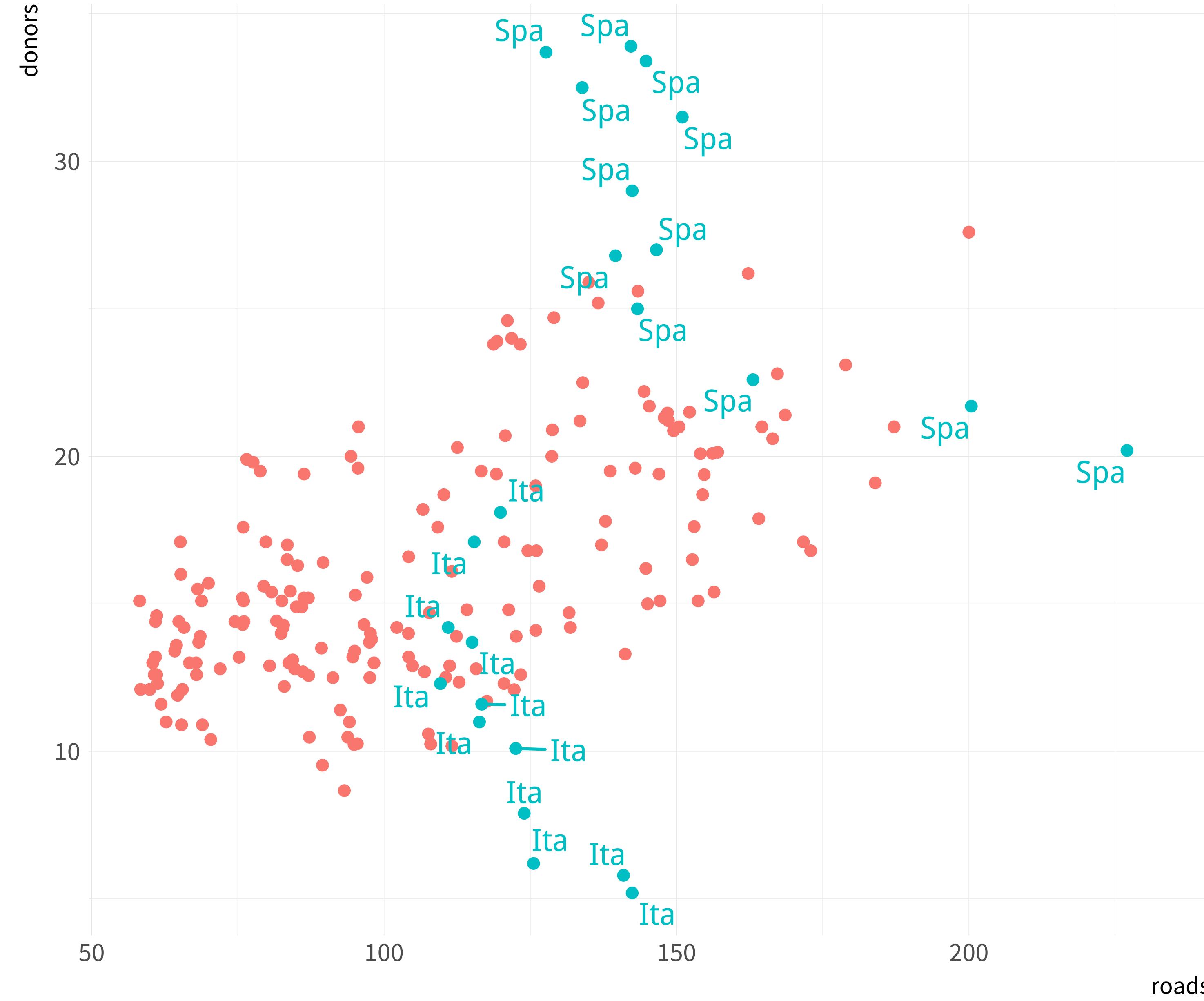
```
p <- ggplot(data = by_country,  
             mapping = aes(x = gdp, y = health))
```

```
p + geom_point() +  
  geom_text_repel(data = subset(by_country,  
                               gdp > 25000 | health < 1500 | country %in% "Belgium"),  
                  mapping = aes(label = country))
```



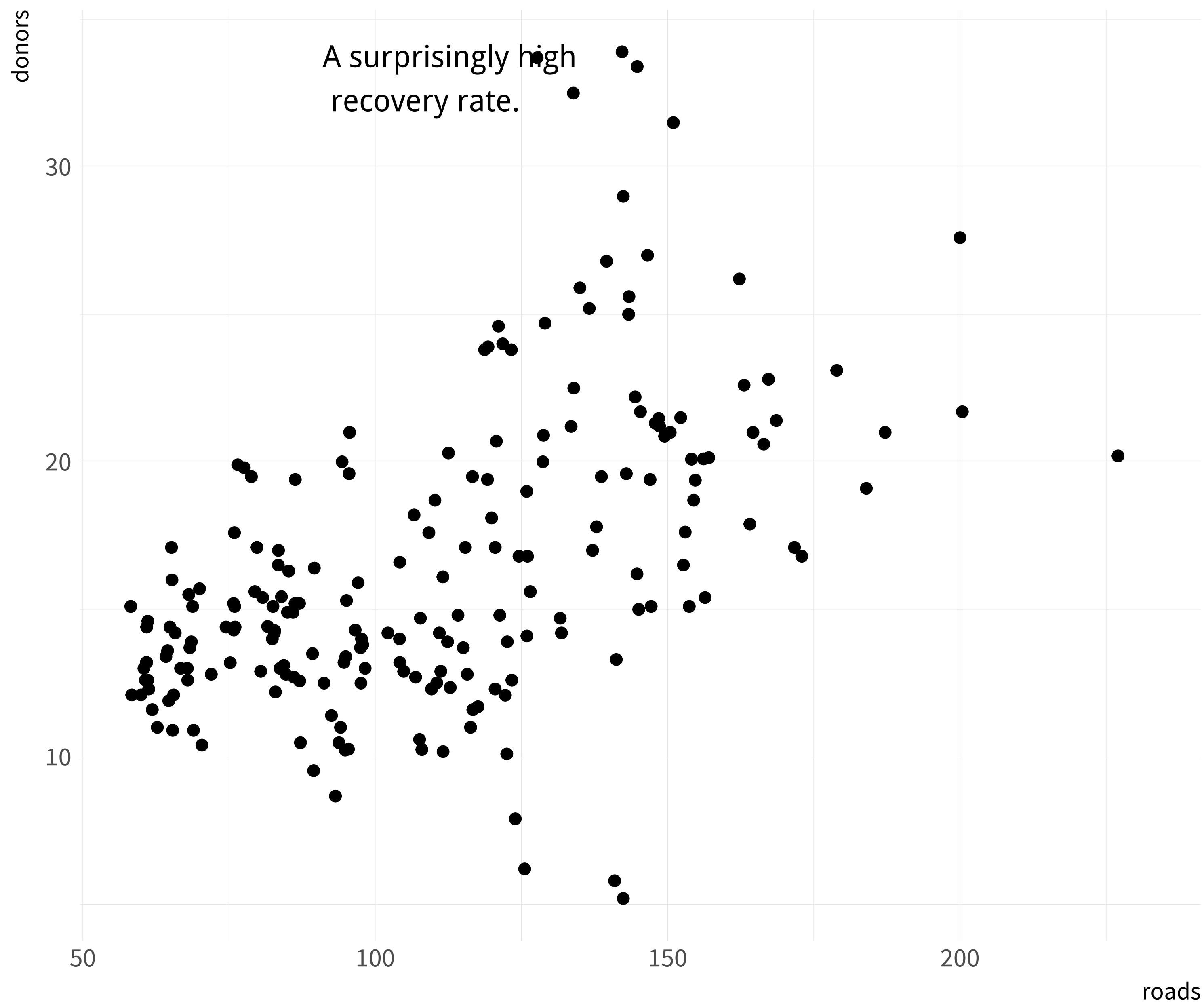
```
organdata <- organdata %>%
  mutate(ind = ccode %in% c("Ita", "Spa") &
         year > 1998)

p <- ggplot(data = organdata,
             mapping = aes(x = roads,
                           y = donors, color = ind))
p + geom_point() +
  geom_text_repel(data = subset(organdata, ind),
                  mapping = aes(label = ccode)) +
  guides(label = FALSE, color = FALSE)
```

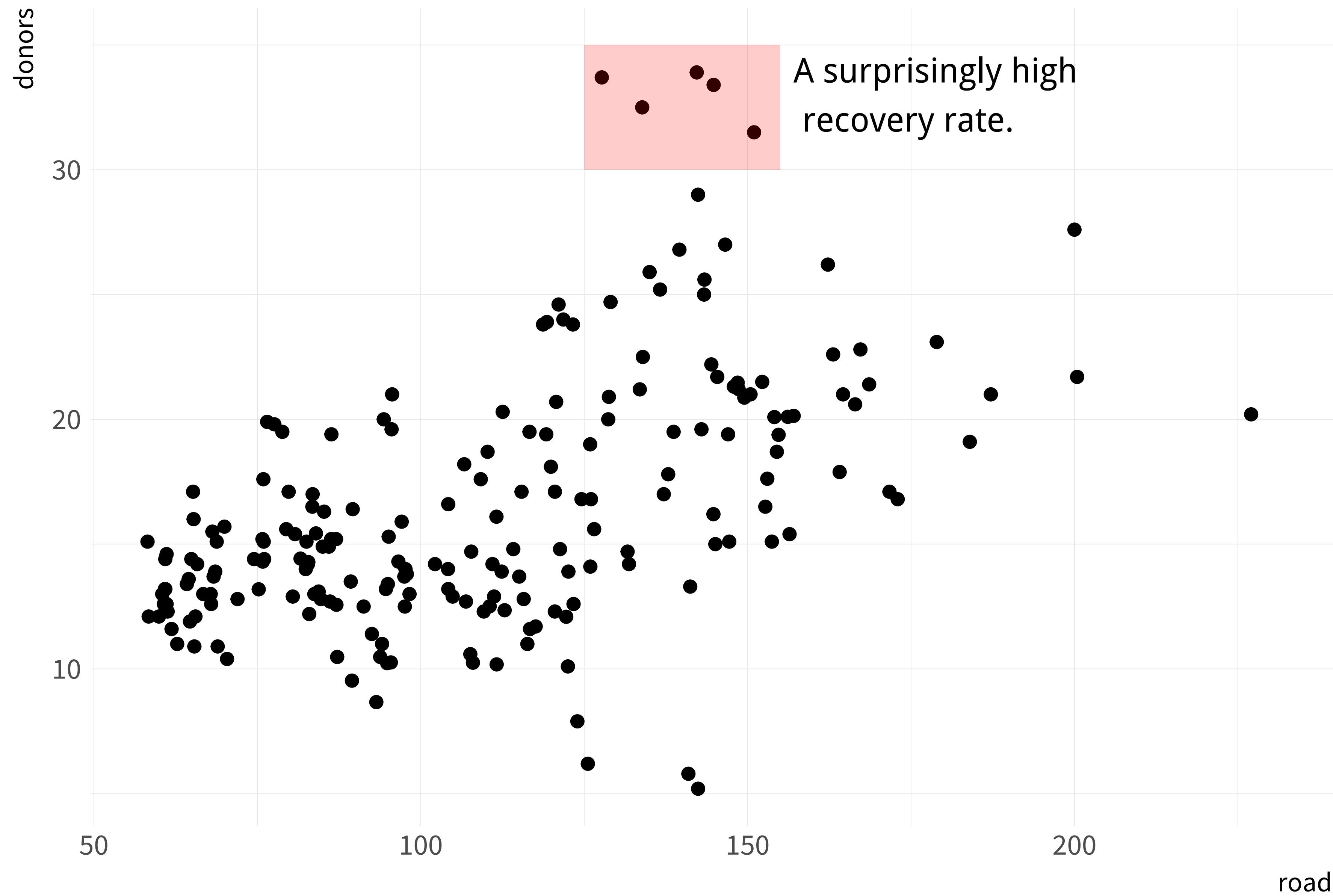


**Write and Draw
in the Plot Area**

```
p <- ggplot(data = organdata,  
             mapping = aes(x = roads, y = donors))  
  
p + geom_point() +  
  annotate(geom = "text", x = 91, y = 33,  
           label = "A surprisingly high \n recovery rate.",  
           hjust = 0)
```



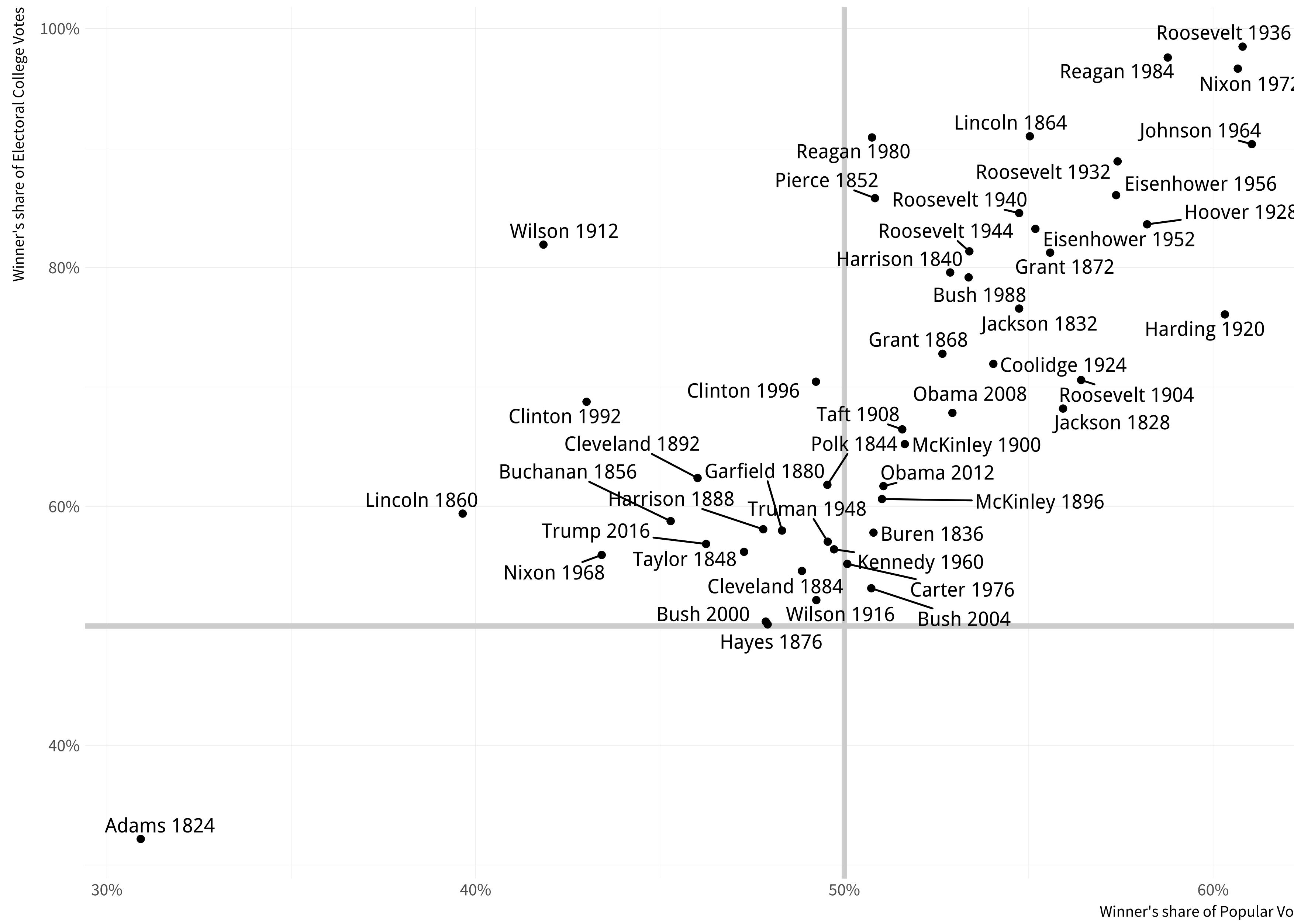
```
p <- ggplot(data = organdata,  
             mapping = aes(x = roads, y = donors))  
  
p + geom_point() +  
  annotate(geom = "rect", xmin = 125, xmax = 155,  
           ymin = 30, ymax = 35, fill = "red", alpha = 0.2) +  
  annotate(geom = "text", x = 157, y = 33,  
           label = "A surprisingly high \n recovery rate.", hjust = 0)
```



**SCALES, GUIDES,
and THEMES**

Presidential Elections: Popular & Electoral College Margins

1824-2016



Data for 2016 are provisional.

```

p <- ggplot(data = gapminder,
             mapping =
               aes(x = gdpPercap,
                   y = lifeExp,
                   color = continent,
                   fill = continent))
p + geom_point() +
  geom_smooth(method = "loess") +
  scale_x_log10()

```



continent



**Scale functions control scale mappings
in geoms. Remember: not just x and y but
also color, fill, shape, and size are scales.**

**They visually represent quantities or
categories in your data—thus, they have
a scale associated with that
representation.**

This means you control things like
color schemes **for data mappings**
through scale functions

```
scale_<MAPPING>_<KIND>()
```

Scale functions are
consistently named, by
mapping and kind

`scale_<MAPPING>_<KIND>()`

`scale_x_continuous()`

`scale_y_continuous()`

`scale_x_discrete()`

`scale_y_discrete()`

`scale_x_log10()`

`scale_x_sqrt()`

`scale_<MAPPING>_<KIND>()`

`scale_color_gradient()`

`scale_color_gradient2()`

`scale_color_hue()`

`scale_fill_gradient()`

`scale_fill_gradient2()`

`scale_fill_gradient()`

scale_<MAPPING>_<KIND>(<ARGUMENTS>)

```
p <- ggplot(data = organdata,  
             mapping = aes(x = roads_mean,  
                            y = donors,  
                            color = world))  
p + geom_point() +  
    scale_x_log10() +  
    scale_y_continuous(breaks = c(5, 15, 25),  
                        labels = c("Five", "Fifteen", "Twenty Five"))
```

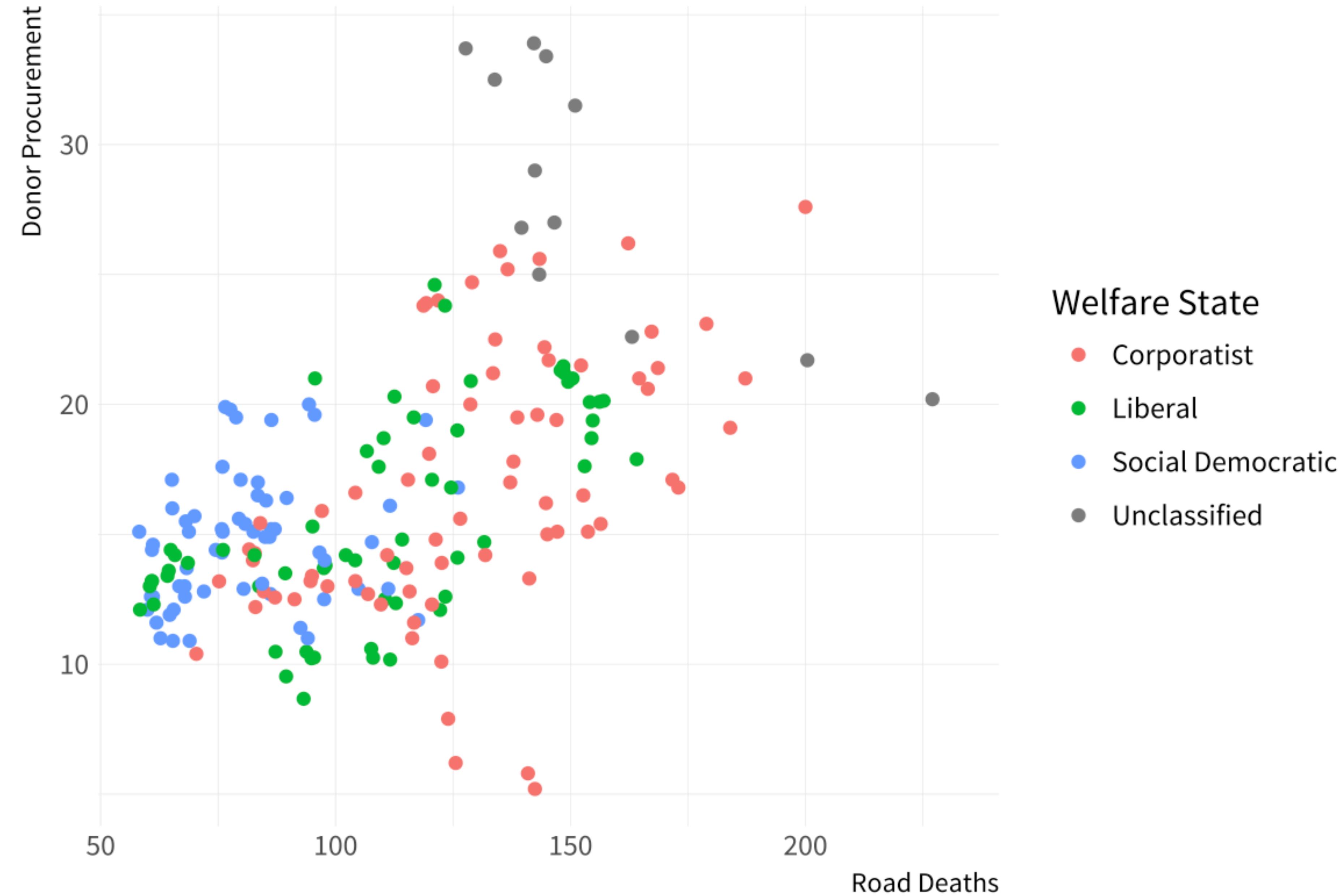
E.g., labels, breaks, and limits



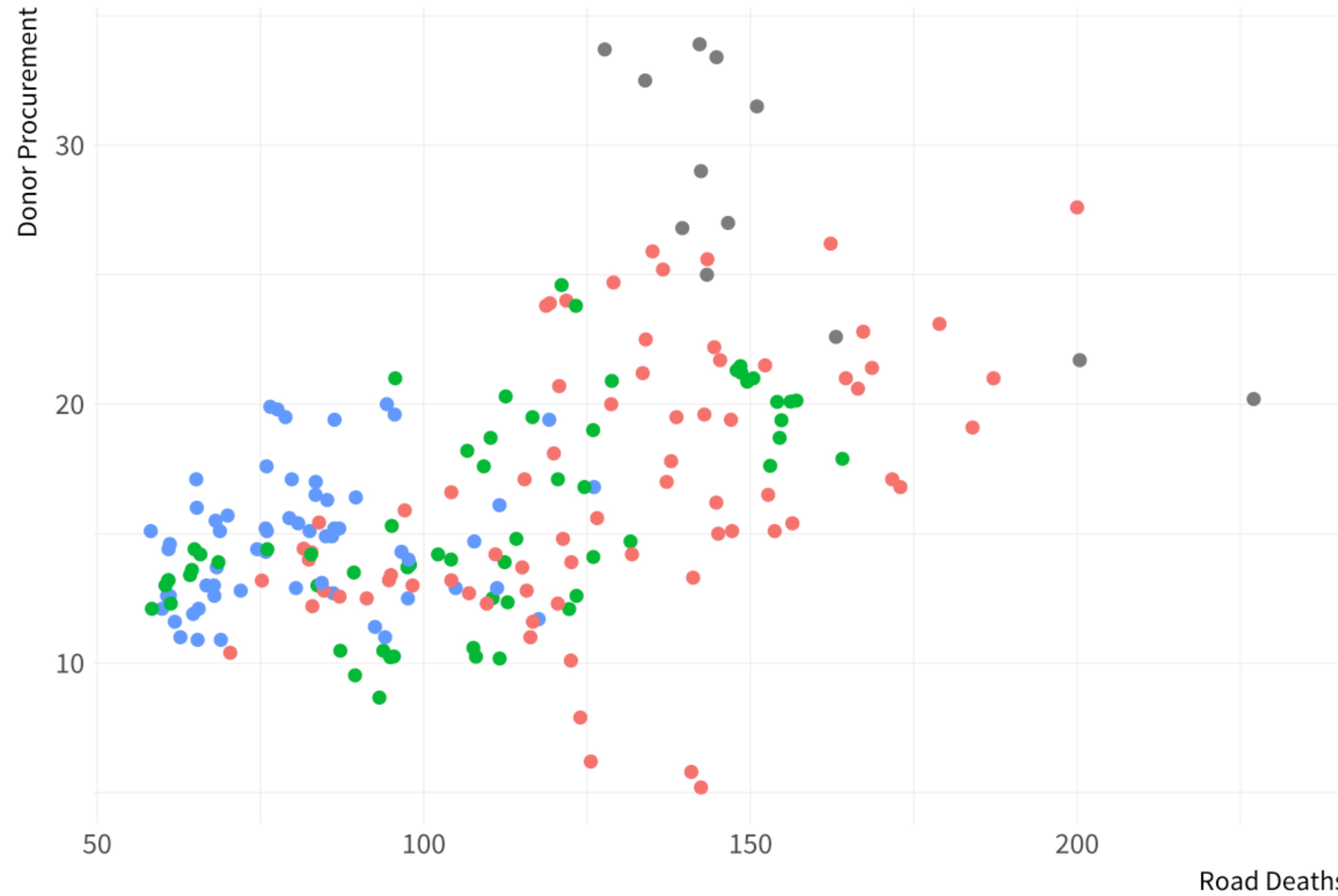
world

- Corporatist
- Liberal
- SocDem
- NA

```
p <- ggplot(data = organdata,
             mapping = aes(x = roads_mean,
                           y = donors,
                           color = world))
p + geom_point() +
  scale_color_discrete(labels =
    c("Corporatist", "Liberal",
      "Social Democratic", "Unclassified")) +
  labs(x = "Road Deaths",
       y = "Donor Procurement",
       color = "Welfare State")
```



```
p <- ggplot(data = organdata,  
             mapping = aes(x = roads,  
                           y = donors,  
                           color = world))  
p + geom_point() +  
  labs(x = "Road Deaths",  
       y = "Donor Procurement") +  
  guides(color = FALSE)
```



```
scale_<MAPPING>_<KIND>()
```